Generator Page - type 0x09

Every database has at least one Generator Page, even if no generators (also known as sequences in Firebird 2.x) have been defined by the user. A blank database consisting only of system tables and indices already has a number of generators created for use in naming constraints, indices, etc.

Note GENERATOR is a non standard term that originated in InterBase®. The ANSI SQL standard requires the term SEQUENCE instead.

The C code representation of the generator page is:

```
struct generator_page
{
   pag gpg_header;
   SLONG gpg_sequence;
   SLONG gpg_waste1;
   USHORT gpg_waste2;
   USHORT gpg_waste3;
   USHORT gpg_waste4;
   USHORT gpg_waste5;
   SINT64 gpg_values[1];
};
```

Gpg_header: The generator page starts off with a standard page header.

Gpg_sequence: Four bytes, signed. Bytes 0x10 - 0x13. The sequence number of this generator page, starting from zero. If so many generators have been created that new generator pages are required, the sequence number will be incremented for each one.

Gpg_waste: Twelve bytes. Bytes 0x14 to 0x1f. To quote the source code, these values are overhead carried forward for backward compatibility. In other words, most likely unused.

Gpg_values: An array of 64-bit values, one for each generator in the database. If we use isql to create a new blank database, we can dump out the generator page as follows:

```
tux> isql
Use CONNECT or CREATE DATABASE to specify a database

SQL> CREATE DATABASE "../blank2.fdb";
SQL> COMMIT;
SQL> EXIT;
```

back to top of page

We need to find the generator page next:

```
SQL> SELECT RDB$PAGE_NUMBER
CON> FROM RDB$PAGES
CON> WHERE RDB$PAGE_TYPE = 9;
```

```
RDB$PAGE_NUMBER
===============
            148

SQL> COMMIT;
```

Now we can dump out the generator page:

tux> ./fbdump ../blank2.fdb -p 148

```
FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 148
    Page Type: 9
PAGE DATA
    Sequence: 0
    Waste1: 0
    Waste2: 0
    Waste3: 0
    Waste4: 0
    Waste5: 0

    There are 9 sequences defined:

    Sequence[00000]: 9
    Sequence[00001]: 0
    Sequence[00002]: 3
    Sequence[00003]: 0
    Sequence[00004]: 0
    Sequence[00005]: 0
    Sequence[00006]: 0
    Sequence[00007]: 0
    Sequence[00008]: 0
    Sequence[00009]: 0
```

The system table RDB$GENERATORS holds the defined sequence details but no values for each one. It does have an RDB$GENERATOR_ID column and this starts from 1, not zero. And increments by 1 for each new sequence. Where does this number come from?

back to top of page

Looking in the blank database we created, we can see that there are 9 sequences created for system use:

```
SQL> SELECT RDB$GENERATOR_ID, RDB$GENERATOR_NAME
CON> FROM RDB$GENERATORS
CON> ORDER BY RDB$GENERATOR_ID;

RDB$GENERATOR_ID RDB$GENERATOR_NAME
================ ==================
```

```
              1 RDB$SECURITY_CLASS
              2 SQL$DEFAULT
              3 RDB$PROCEDURES
              4 RDB$EXCEPTIONS
              5 RDB$CONSTRAINT_NAME
              6 RDB$FIELD_NAME
              7 RDB$INDEX_NAME
              8 RDB$TRIGGER_NAME
              9 RDB$BACKUP_HISTORY
```

This is a clue, take a look at Sequence[00000], above, and see that it contains the value 9. I suspect therefore, that the very first sequence is used to generate the RDB$GENERATOR_ID value when a new sequence is created. One way to find out is to create a new sequence.

```
SQL> CREATE SEQUENCE NEW_GENERATOR;
SQL> SET GENERATOR NEW_GENERATOR TO 666;
SQL> COMMIT;

SQL> SELECT RDB$GENERATOR_ID, RDB$GENERATOR_NAME
CON> FROM RDB$GENERATORS
CON> WHERE RDB$GENERATOR_ID > 9;

RDB$GENERATOR_ID RDB$GENERATOR_NAME
================ ================
              10 NEW_GENERATOR
```

So far, so good, we see a new sequence. Time to hexdump the database file's generator page again:

```
tux> ./fbdump ../blank2.fdb -p 148

FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 148
    Page Type: 9
PAGE DATA
    ...

    There are 10 sequences defined:

    Sequence[00000]: 10
    Sequence[00001]: 0
    Sequence[00002]: 3
    Sequence[00003]: 0
    Sequence[00004]: 0
    Sequence[00005]: 0
    Sequence[00006]: 0
    Sequence[00007]: 0
    Sequence[00008]: 0
    Sequence[00009]: 0
    Sequence[00010]: 666
```

We can see that Sequence[00010], that a new sequence has been created. The value in this sequence is 666 in decimal. In addition, we can see that Sequence[00000] has increased to 10. So it looks remarkably like the RDB$GENERATOR_ID is itself obtained from a sequence that never appears in RDB$GENERATORS.

The value, stored in Sequence[n], appears to be the last value that was used and not the next value to be issued. It is also the total number of sequences that have been created thus far in the database, provided, that the value in gpg_sequence is zero.

back to top of page

I wonder what happens when we drop a sequence?

```
SQL> DROP SEQUENCE NEW_GENERATOR;
SQL> COMMIT;

SQL> SELECT RDB$GENERATOR_ID, RDB$GENERATOR_NAME
CON> FROM RDB$GENERATORS
CON> WHERE RDB$GENERATOR_ID > 9;

SQL>
```

We can see that the sequence is dropped from the RDB$GENERATORS table, what about in the generator page in the database?

```
tux> ./fbdump ../blank2.fdb -p 148

FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 148
    Page Type: 9
PAGE DATA
    ...

    There are 10 sequences defined:

    Sequence[00000]: 10
    Sequence[00001]: 0
    Sequence[00002]: 3
    Sequence[00003]: 0
    Sequence[00004]: 0
    Sequence[00005]: 0
    Sequence[00006]: 0
    Sequence[00007]: 0
    Sequence[00008]: 0
    Sequence[00009]: 0
    Sequence[00010]: 666
```

The generator page has not changed. Sequence[00010] still remains at it's previous value - 666 - but this 64 bits of database page representing our recently dropped sequence can never be used again. It

has ceased to be a sequence and has become wasted space.

Given that RDB$GENERATOR_ID is itself generated from Sequence[00000] and cannot therefore reuse any allocated RDB$GENERATOR_ID, it is not surprising that the simplest way of handling a dropped sequence is simply to ignore it.

back to top of page

If you are creating and dropping sequences frequently, you may end up with a lot of unused sequences. You can restore these to a usable state by dumping and restoring the database:

```
tux> # Shutdown & backup the database...
tux> gfix -shut -tran 60 ../blank2.fdb
tux> gbak -backup ../blank2.fdb ./blank2.fbk

tux> # Replace (!) and restart the database...
tux> gbak -replace ./blank2.fbk ../blank2.fdb
```

Warning: The above will cause the loss of the database if anything goes wrong. The commands used overwrite the blank 2.fdb database from the dumpfile. If the dumpfile is corrupt, then we will lose the database as the recovery starts by wiping the database.

If we now dump the generator page as before, we see the following:

```
> ./fbdump ../blank2.fdb -p 148

FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 148
    Page Type: 9
PAGE DATA
    ...

    There are 9 sequences defined:

    Sequence[00000]: 9
    Sequence[00001]: 0
    Sequence[00002]: 3
    Sequence[00003]: 0
    Sequence[00004]: 0
    Sequence[00005]: 0
    Sequence[00006]: 0
    Sequence[00007]: 0
    Sequence[00008]: 0
    Sequence[00009]: 0
```

We now see that the deleted sequence has gone and the value in Sequence[00000] has reduced by one (the number of deleted sequences) to suit. If we now create a brand new sequence, it will reuse the slot previously occupied by our deleted sequence.

```
SQL> CREATE SEQUENCE ANOTHER_SEQUENCE;
```

```
SQL> COMMIT;
```

Dumping the generator page again, we see:

```
tux> ./fbdump ../blank2.fdb -p 148

FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 148
    Page Type: 9
PAGE DATA
    ...

   There are 10 sequences defined:

   Sequence[00000]: 10
   Sequence[00001]: 0
   Sequence[00002]: 3
   Sequence[00003]: 0
   Sequence[00004]: 0
   Sequence[00005]: 0
   Sequence[00006]: 0
   Sequence[00007]: 0
   Sequence[00008]: 0
   Sequence[00009]: 0
   Sequence[00010]: 0
```

Bearing in mind that in ODS 11 onwards, a sequence is a 64-bit value, how many sequences can we store on a page? The answer will be (page size - 32 bytes)/8 and we are allowed a maximum of 32,767 sequences in any one database. With a 4K page size that would mean that sequence 508 would be the first on the next page.

back to top of page

Because there is no apparent next and previous page numbers on a generator page, how does the database know where to find the actual page that the generator values are stored on? RDB$PAGES is a system table that the main database header page holds the page number for. This allows the system, on startup, to determine where it's internal data can be found. For because sequences live, as it were, in RDB$GENERATORS we can look in RDB$PAGES as follows, to find the actual page number(s):

```
SQL> SELECT *
CON> FROM RDB$PAGES
CON> WHERE RDB$PAGE_TYPE = 9;

RDB$PAGE_NUMBER RDB$RELATION_ID RDB$PAGE_SEQUENCE RDB$PAGE_TYPE
=============== =============== ================= =============
            148               0                 0             9
```

The RDB$RELATION_ID is zero because this is not actually the location of a relation (table) in the

database itself, but the location of a specific page that we are after. Given that RDB$PAGE_SEQUENCE = 0 and RDB$PAGE_TYPE = 9 we see that the first generator page is located on page 148 of the database.

If there are more than one page, then the page that has gpg_sequence set to zero is the first one and the first sequence on that page is the count of all sequences created (and possibly deleted) within the database. If the gpg_sequence is non-zero, then there is no way to tell how many sequences on that page are actually valid and even when the gpg_sequence is zero, unless the database has been restored since any sequences were last deleted, it is not possible to determine which sequences on the page are still valid. (Unless you have access to the RDB$GENERATOR_ID in RDB$GENERATORS of course.

back to top of page

Creating lots of sequences

When you create a new blank database, the first generator page is created as part of the new database. It has to be this way because there are nine system sequences created, as described above. (Well, there are 10 actually, but no-one has access to the first one!)

When the user starts creating new sequences, they will be added to the existing generator page. However, once a new page is required things change!

Given that there can be 508 sequences, in total, on a single 4 Kb database page, then when sequence 509 is created a new page - of type 0x09 - will be required.

If the new sequence is not given an initial value, then the new page is not created yet. An entry will be created in RDB$PAGES with RDB$PAGE_SEQUENCE set correctly (to match what will be in the gpg_sequence field in the page structure when it is finally created) and a new sequence will be stored in RDB$GENERATORS, but nothing will happen to extend the database with the required new page until such time as either:

- The sequence value is read within a transaction; or
- The sequence number is explicitly set to a new value.

It is only now that the required generator page is actually created and written to the (end of) the database file. The following explains the sequence of events that take place when a brand new blank database is extended by the creation of an additional 5,000 sequences.

1. A blank database has 10 pre-created sequences used internally - nine are visible in RDB$GENERATORS, one is hidden. A generator page exists and the details can be found in RDB$PAGES. Page 148 is the first generator page in a 4 Kb page size database. The database file is 161 pages long (659,456 bytes).

```
tux> isql
Use CONNECT or CREATE DATABASE to specify a database

SQL> CREATE DATABASE 'seq.fdb';
```

```
SQL> SHELL;

tux> ls -l seq.fdb
-rw------- 1 firebird firebird 659456 2010-05-12 11:26 seq.fdb

tux> exit

SQL> SELECT RDB$GENERATOR_ID,
CON>        RDB$GENERATOR_NAME
CON> FROM RDB$GENERATORS
CON> ORDER BY RDB$GENERATOR_ID;

RDB$GENERATOR_ID RDB$GENERATOR_NAME
================ ==================
               1 RDB$SECURITY_CLASS
               2 SQL$DEFAULT
               3 RDB$PROCEDURES
               4 RDB$EXCEPTIONS
               5 RDB$CONSTRAINT_NAME
               6 RDB$FIELD_NAME
               7 RDB$INDEX_NAME
               8 RDB$TRIGGER_NAME

SQL> SELECT *
CON> FROM RDB$PAGES
CON> WHERE RDB$PAGE_TYPE = 9;

RDB$PAGE_NUMBER RDB$RELATION_ID RDB$PAGE_SEQUENCE RDB$PAGE_TYPE
=============== ================ ================= =============
            148                0                 0             9

SQL> COMMIT;
```

back to top of page

2. The user creates a set of 5,000 new sequences. The database extends to accommodate the data being written into the system table RDB$GENERATORS, but there are no new generator pages written. The database is now 256 pages long (1,048,576 bytes).

RDB$PAGES still shows that page 148 is the only generator page in the database.

```
SQL> INPUT gens.sql;

SQL> SELECT *
CON> FROM RDB$PAGES
CON> WHERE RDB$PAGE_TYPE = 9;
```

RDB$PAGE_NUMBER RDB$RELATION_ID RDB$PAGE_SEQUENCE RDB$PAGE_TYPE

```
=============== ================ ================= =============
```

```
           148                0                  0             9

SQL> SHELL;

tux> ls -l seq.fdb
-rw------- 1 firebird firebird 1048576 2010-05-12 11:28 seq.fdb

tux> exit
```

back to top of page

3. A transaction touches the final sequence - which has RDB$GENERATOR_ID = 5,009 - by reading its value (without changing it). At this point a new generator page is created and written to the database. The page has gpg_sequence set to 9, which is the correct page for sequence number 5,009. The database is now 257 pages in size (1052672 bytes).

```
SQL> SELECT RDB$GENERATOR_ID,RDB$GENERATOR_NAME
CON> FROM RDB$GENERATORS
CON> WHERE RDB$GENERATOR_ID = (
CON> SELECT MAX(RDB$GENERATOR_ID)
CON> FROM RDB$GENERATORS
CON> );

RDB$GENERATOR_ID RDB$GENERATOR_NAME
================ ================
            5009 RANDOM_SEQ_4994

SQL> SELECT GEN_ID(RANDOM_SEQ_4994, 0)
CON> FROM RDB$DATABASE;

               GEN_ID
=====================
                    0

SQL> SHELL;

tux> ls -l seq.fdb
-rw------- 1 firebird firebird 1052672 2010-05-12 11:33 seq.fdb

tux> exit
```

RDB$PAGES shows that there are now two pages in the database with type 9. The original page 148 and a new page 256. Looking at the database file itself, however, shows that it is actually 257 pages long. Page 257, the last page, has page type zero - which is not a defined page type and, doesn't appear in RDB$PAGES.

```
SQL> SELECT *
CON> FROM RDB$PAGES
CON> WHERE RDB$PAGE_TYPE = 9
CON> OR RDB$PAGE_NUMBER = 257;
```

```
RDB$PAGE_NUMBER RDB$RELATION_ID RDB$PAGE_SEQUENCE RDB$PAGE_TYPE
=============== =============== ================= =============
            148               0                 0             9
            256               0                 9             9

SQL> SHELL;

tux> ./fbdump seq.fdb -p 257

FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 257
    Page Type: 0
```

The RDB$PAGE_SEQUENCE in RDB$PAGES for the new page, page 256, is set to 9 which corresponds to the gpg_sequence number in the actual page.

```
tux> ./fbdump seq.fdb -p 256

FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 256
    Page Type: 9
PAGE DATA
    Sequence: 9
...
```

back to top of page

4. A separate transaction changes the value of the sequence with RDB$GENERATOR_ID = 520, which is to be found on the second page of sequences. This page doesn't yet exist and is created with page number 257. Looking at RDB$PAGES shows that this new page exists in the database. The database file has extended now to 258 pages or 1,056,768 bytes.

The sequence in question, however, still has the value zero. (The transaction has yet to commit.)

```
SQL> SELECT RDB$GENERATOR_NAME
CON> FROM RDB$GENERATORS
CON> WHERE RDB$GENERATOR_ID = 520;

RDB$GENERATOR_NAME
==================
RANDOM_SEQ_534

SQL> SET GENERATOR RANDOM_SEQ_534 TO 666;

SQL> SELECT *
CON> FROM RDB$PAGES
CON> WHERE RDB$PAGE_TYPE = 9;
```

```
RDB$PAGE_NUMBER RDB$RELATION_ID RDB$PAGE_SEQUENCE RDB$PAGE_TYPE
=============== ============== ================= =============
            148              0                 0             9
            256              0                 9             9
            257              0                 1             9

SQL> SHELL;

tux> ls -l seq.fdb
-rw------- 1 firebird firebird 1056768 2010-05-12 13:07 seq.fdb

tux> ./fbdump seq.fdb -p 257

FBDUMP 1.00 - Firebird Page Dump Utility

DATABASE PAGE DETAILS - Page 257
    Page Type: 9
PAGE DATA
     Sequence: 1
     Waste1: 0
     Waste2: 0
     Waste3: 0
     Waste4: 0
      Waste5: 0

    This is not the first generator page.
    Total generator count unknown.
    There are [a maximum of] 508 sequences located on this page.

    Sequence[00508]: 0
    ...
    Sequence[00520]: 0
    ...
```

Only after a commit does the sequence takes the new value of 666.

```
tux> exit

SQL> COMMIT;

SQL> SHELL;

tux> ./fbdump seq.fdb -p 257

FBDUMP 1.00 - Firebird Page Dump Utility

    ...
    Sequence[00520]: 666
    ...
```