

# Haskey - Design Document

Henri Verroken

August 2017

## 1 Binary format

This section describes the on-disk binary format. Figure 1 shows the binary format of a generic page. The binary format of each page type starts with the xxHash 64-bit checksum<sup>1</sup> of all the data after the checksum field. It is followed by a field that indicates the page type. The binary format of the remaining data is page type specific. The type field can be one of the following values:

- **Empty (0x00):** An empty, unused page. The remaining data is garbage.
- **Meta (0x20):** A meta-page of the database (Section 1.1).
- **Overflow (0x40):** An overflow page, containing overflow data (Section 1.2).
- **Leaf node (0x60):** A leaf node page, containing actual key-value pairs (Section 1.3).
- **Index node (0x80):** An index node page, containing a branch node of the tree (Section 1.4).

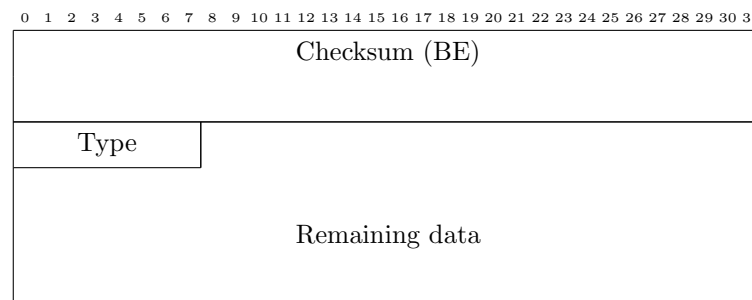


Figure 1: Binary format of a generic page, of a certain type.

---

<sup>1</sup>Seeded with a zero word.

The remaining data in Figure 1 can be compressed using lz4. In that case Figure 2 shows the binary format of a generic packet. The first field is the xxHash 64-bit checksum<sup>2</sup> of all (partially compressed) data after the checksum field. The type field indicates the page type, which is ORed with 0x01 to indicate that the page contents are compressed. The data is compressed using the lz4 library.

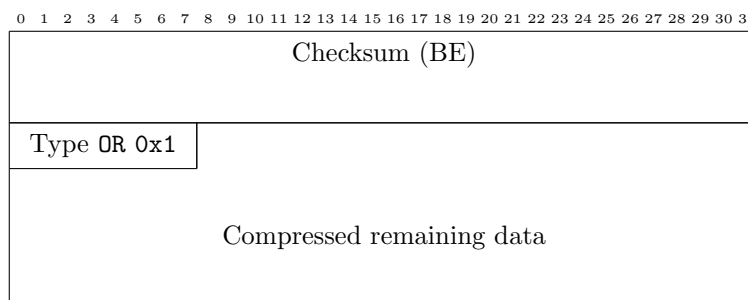


Figure 2: Binary format of a generic page with compressed data, of a certain type.

## 1.1 Meta pages

A meta-page is used to store meta-data about the database. Figure 3 shows the binary format of a meta-page. The remaining data field is encoded using the standard **Generic**-derived **Binary** instance of a **ConcurrentMeta** record, and contains the following data. No further optimizations were made to the binary format.

- The transaction id of the most recent committed transaction
- The number of pages in both the data file and the index file
- A pointer to the root of the main tree, along with its height.
- Pointers to the roots of the free trees of both the data file and the index file, along with their heights.
- A pointer to the root of the overflow tree.
- Two collections of page IDs, one for the data file and one for the index file, that are immediately ready for reuse by the next transaction.

---

<sup>2</sup>Seeded with the zero word.

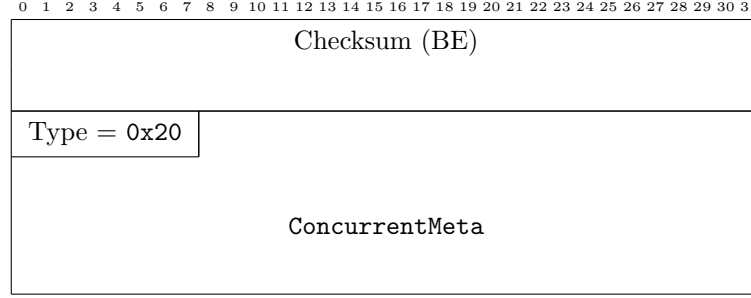


Figure 3: Binary format of a meta page, the binary format of the `ConcurrentMeta` record is not optimized.

## 1.2 Overflow pages

An overflow page is used to store data that is too big to store in a leaf node. Figure 4 shows the binary format of an overflow page. The remaining data field is simply the encoded version of the value using its user-provided `Binary` instance.

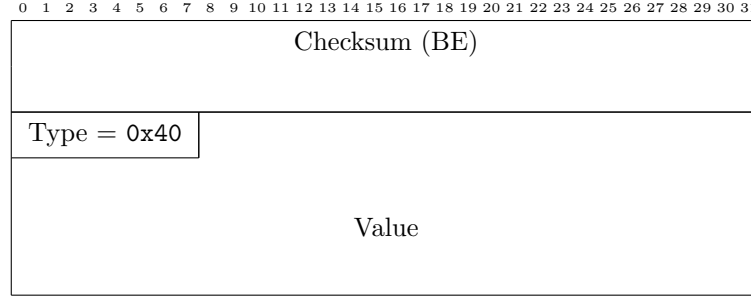


Figure 4: Binary format of an overflow page, the value field is simply the encoded version of the value using its user-provided `Binary` instance.

## 1.3 Leaf nodes

A leaf node contains actual key-value pairs or key-overflow pairs. Figure 5 shows the binary format of a leaf node page. It contains a 3-byte value  $N$  indicating the number of key-value pairs. The (key, value) tuple is encoded using its `Binary`-instance, but the value is preceded by a byte indicating whether or not it is an overflow value, as shown in Figure 6 and 7.

The 3-byte  $N$  value means that we need at least 3-byte keys to get  $2^{24} - 1$  unique keys. An extra byte to encode the overflow information, means every leaf entry requires at least 4 bytes to fill up the 3-byte counter. This sets an upper bound on the page size of  $2^{24} \times 4$  bytes = 67 MB, which is deemed plenty.

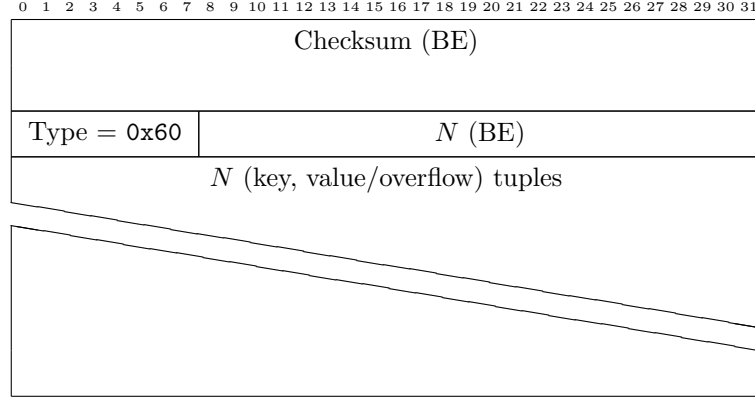


Figure 5: Binary format of a leaf node page.

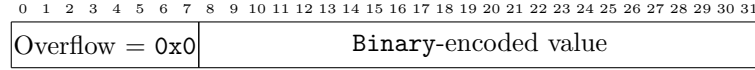


Figure 6: Binary format of a non-overflow value.

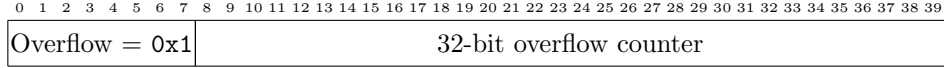


Figure 7: Binary format of an overflow value.

## 1.4 Index nodes

An index node contains key-page pairs. Figure 8 shows the binary format of an index node page. It contains a byte indicating the height of the branch node, and a 3-byte  $N$  value indicating the amount of keys in the index node.

The 3-byte  $N$  value means that we need at least 3-byte keys to get  $2^{24} - 1$  unique keys, along with  $2^{24}$  8-byte page IDs. This sets an upper bound on the page size of  $2^{24} \times 11$  bytes = 180 MB, which is deemed plenty.

The 1-byte height value, means the maximum amount of leaf nodes is  $2^{255}$ , which is also plentiful.

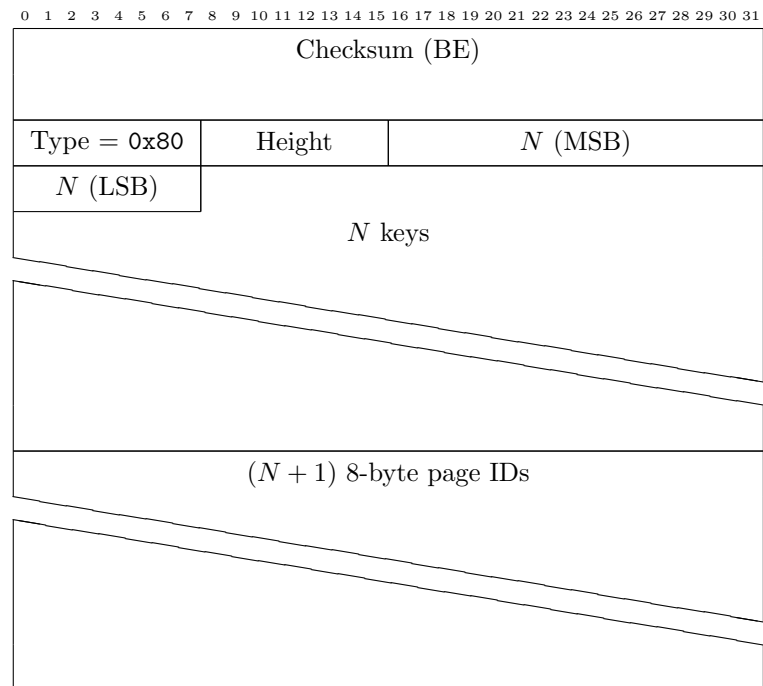


Figure 8: Binary format of an index node page.