Databases and hash tables

A dive into an ancient format

Quentin Godfroy

Trainline

2020-07-08

Presentation

About me

- Connections Architect.
- Paris Office.
- I joined the company in 2012.
- As a Capitaine Train employee.
- I work mainly on the carriers connector.

Outline

Generalities

What do we want from databases

Thought

Ultimately, databases store things for us. Generally on long-term storage. The choice of technology is a matter of compromise.

- speed of reads/writes/updates
- easy queries/relational databases
- replication
- storage
- data lifecycle

Left-out

Data manipulation is another very important feature of modern databases. It is not the subject here.

Generalities

Some examples

Some technology examples:

- MySQL, PostgreSQL, . . . — relational
- Filesystems — large storage, hierarchical, random file access
- Object storage like S3 — object-level access
- NoSQLs — key-value stores

Common denominator

Many sytems use efficient ways to locate records. One way of doing this is the hash table.

Hash table

Essential tool

Definition

A hash table is one way to rapidly locate a record—the `value`—stored under a `key`.

Basic function

A hash function shrinks keys to a fixed size. Keys that hash together are stored into buckets. Buckets are searched sequentially until the key is found.

Virtually every storage tool uses either hash tables or b-trees.

Hash table

Hash function

Some properties:

- fixed output size
- uniformity over the result space
- speed
- deterministic
- generally not injective!
- parametrizable

Warning

Do not confuse this with a cryptographic hash. Cryptographic hashes are overkill for this application and are generally too slow.

Storage

Memory example

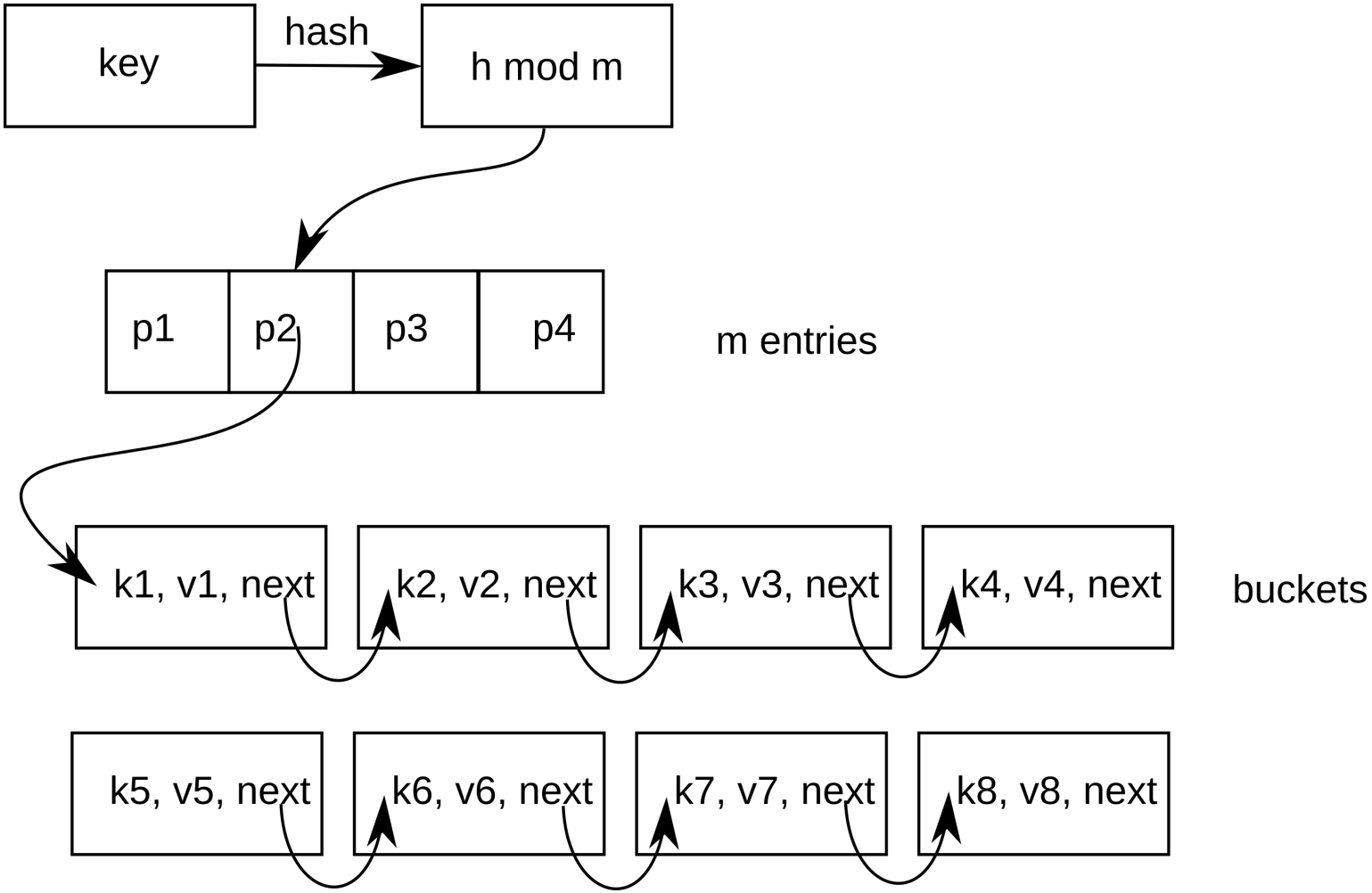


Figure: Typical in memory hash table structure

Storage

Disk storage

Why disk storage is different from memory:

- Orders of magnitude difference in capacity. Think kilobytes for RAM, megabytes for disk—in the 70s. Ratio hasn't changed that much.
- Slow random access for disks. Bad tolerance for fragmentation.
- File API forces allocated space to be reused on disks. No `free()`.

Requirements

We need reusable space, flat storage, low memory footprint, minimize seeks, convenient access to data.

DBM

Description

DBM: DataBase Manager.

- Introduced by Ken Thompson for Unix V7 in 1979.
- Very early example of a NoSQL database.
- Key-value hash table with on-disk storage.
- Page-based. A page contains several key-value pairs.
- Two files: `database.dir` and `database.pag`.
- Has been used with the yellowpages directory service.

Why am I talking about this?

Interesting insight on modern databases. Old piece of technology. Curiosity. Part of being an engineer to understand previous works.

DBM

Principles

- Fixed page size.
- Pages hold the key-value pairs.
- Keys are hashed.
- Keys with the same hash prefix are in the same page.
- When pages overflow they are split in two.
- The directory stores the splitting events.
- Key lookup algorithm walks the history of splits.

Basic principle

Keys are stored in the page # that corresponds to its hash. Only the first bits are considered. More bits are used as split pages are encountered.

Page searching

$hash \leftarrow hash(key)$; $mask = 0$; $page = 0$;
while *isSplit(hash, mask)* **do**
  $mask \leftarrow mask << 1 + 1$;
end
$page = hash \,\&\, mask$;

When the algorithm is finished, the key should be in the page. Or isn't and it's not in the database.

Pages are split when a new key to be inserted cannot fit.

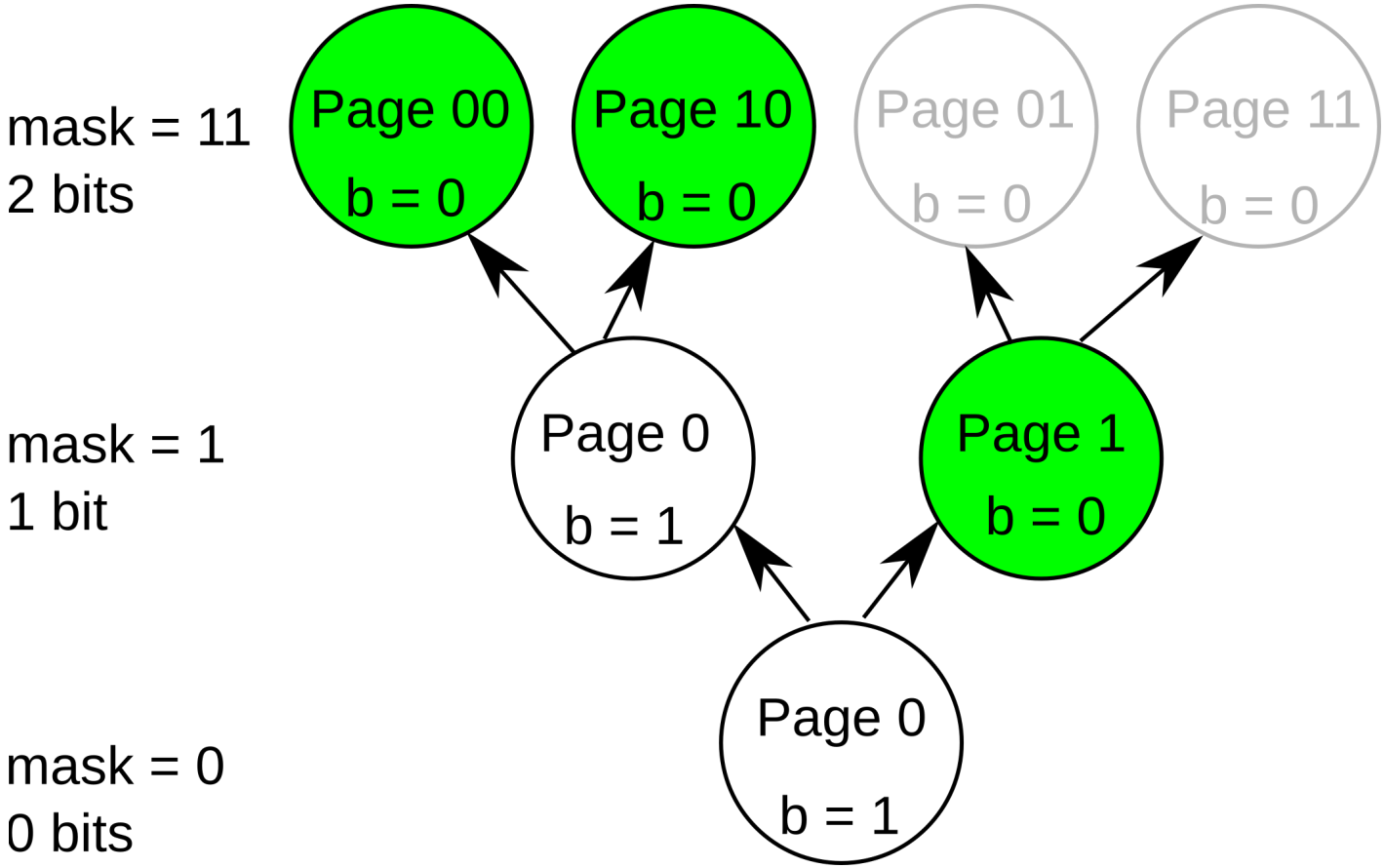


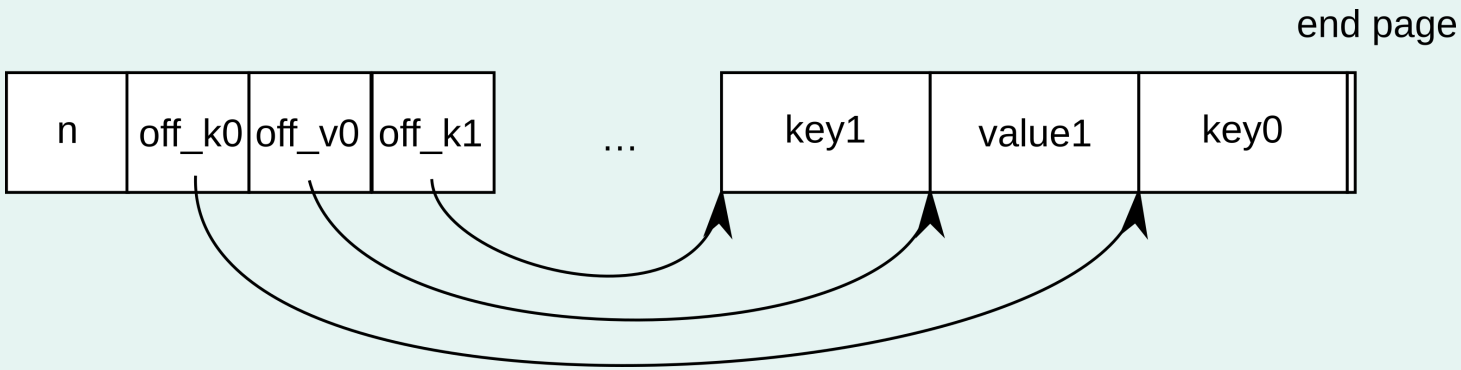
Figure: Split history creates a binary tree. Allows to list all keys with tree traversal.

Directory

Sea of bits. Flattened binary tree. Each level has 2^{ℓ} entries. For a mask m there are exactly m entries for the previous levels.

$$isSplit(hash, mask) : hash \,\&\, mask + mask$$

Pages



Limitations

- Keys that share the same hash have to fit in the same page.
- Hard size limit for key + value.
- It's a single index.
- No concurrency
- Two sparse files. Can appear to be pretty large.
- Initially the library allowed only one database at a time.
- Fixed hash function.

So why am I talking about this thing? Because it's one of the building blocks of modern databases. BerkeleyDB is one of its direct descendents.

Beyond DBM

It's not very complex to imagine how we can remedy to some of the problems.

Compact file

Use more directories and indirection tables. Store the splits in a page itself.

Overhead pages

Store only small data inside pages. Dedicate pages to large payloads.

Hashing

Use parametrizable hashing.

End

- Thank you.
- Don't forget to give feedbacks. `!summit` on slack.