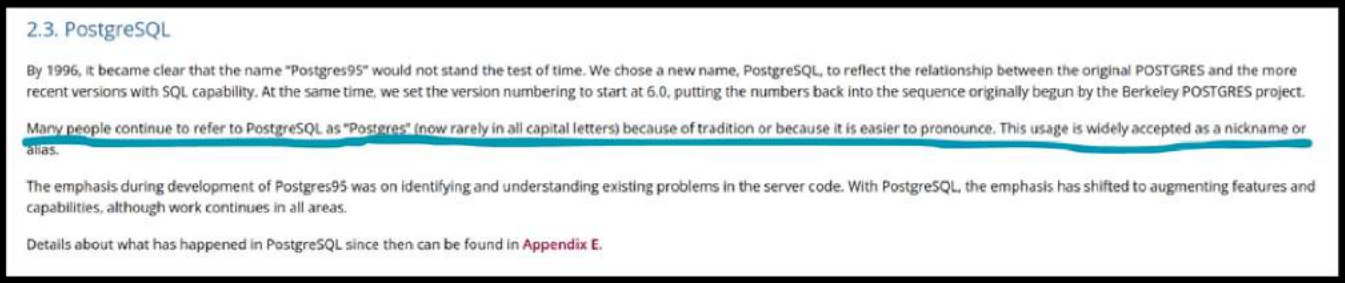

Unlocking PostgreSQL's Potential

Disclaimer

This presentation is flavored with humor. If you spot any technical goofs, help us keep it accurate!

It's Postgres or Post-gres-q-l.
Not
It's Postgrey or Postgres SQL.



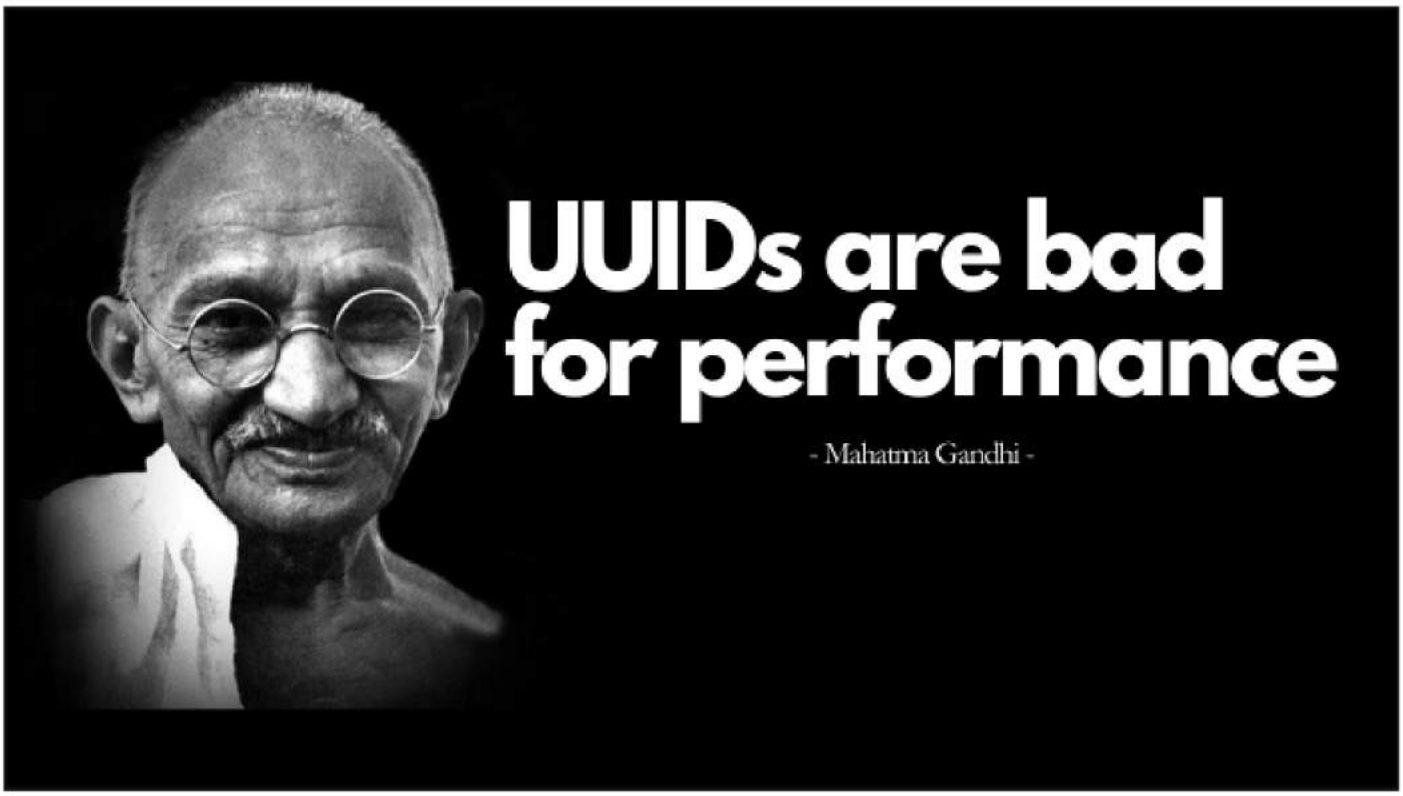
2.3. PostgreSQL

By 1996, it became clear that the name "Postgres95" would not stand the test of time. We chose a new name, PostgreSQL, to reflect the relationship between the original POSTGRES and the more recent versions with SQL capability. At the same time, we set the version numbering to start at 6.0, putting the numbers back into the sequence originally begun by the Berkeley POSTGRES project.

Many people continue to refer to PostgreSQL as "Postgres" (now rarely in all capital letters) because of tradition or because it is easier to pronounce. This usage is widely accepted as a nickname or alias.

The emphasis during development of Postgres95 was on identifying and understanding existing problems in the server code. With PostgreSQL, the emphasis has shifted to augmenting features and capabilities, although work continues in all areas.

Details about what has happened in PostgreSQL since then can be found in Appendix E.


UUIDs are bad
for performance
- Mahatma Gandhi -

So What is UUID and is it truly random?

Mostly Yes. UUID is a 32-char (excluding 4 hyphens) value which is 128 bits. Out of 128 bits, Only 6 bits are always fixed, leaving 122 bits of randomness

Reasons for Using IDs

Cannot let customer know that they placed the first order and let them know that business is bad

Not easy to perform enumeration type attacks on UUIDs

Security

Also, we're moving towards distributed computing everywhere, and distributed DBs usually need a way to generate completely secure, random, unguessable ID which is unique across the distributed network.

What is a B-tree?

B-tree

Article Talk

From Wikipedia, the free encyclopedia

Not to be confused with Binary tree or B+-tree.

In computer science, a **B-tree** is a self-balancing tree data structure that maintains sorted data and allows searches, sequential access, insertions, and deletions in logarithmic time. The B-tree generalizes the binary search tree, allowing for nodes with more than two children.[2] Unlike other self-balancing binary search trees, the B-tree is well suited for storage systems that read and write relatively large blocks of data, such as databases and file systems.

PostgreSQL's default index type for most scenarios is B-tree.

Its not about UUIDs, its about randomness

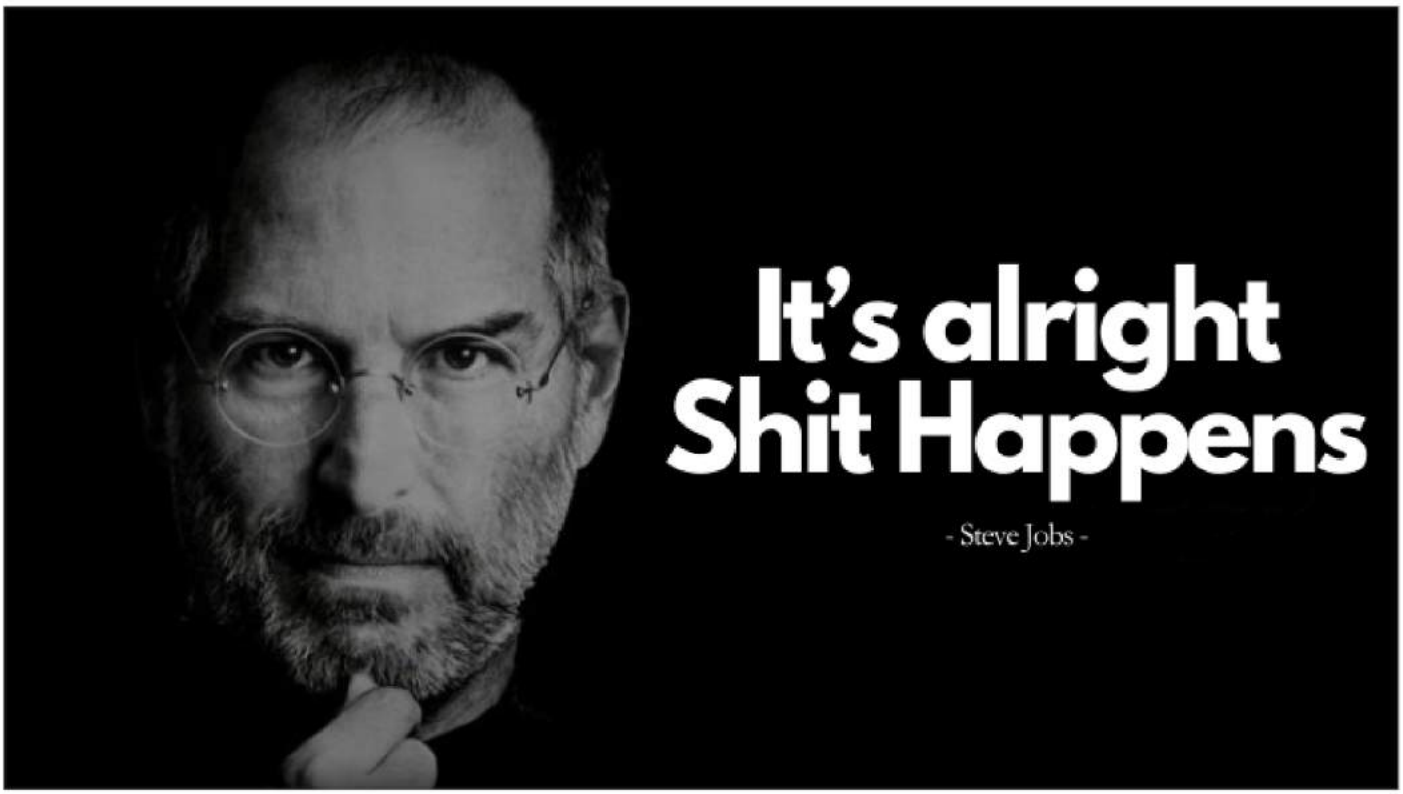
If we as humans cannot sort or understand the order of UUIDs given their randomness, we cannot expect the same from machines obviosuly. As a consequence, the whole index table has to be fetched into memory for lookups.

- UNCLE XAVIER

Cache Evictions

As the number of UUIDs and associated data pages in shared_buffers increases, there's a higher chance of cache evictions. Cache evictions occur when new data pages need to be loaded into shared_buffers, but there's not enough space, so some existing pages must be removed to make room for the new ones. Frequent cache evictions can lead to more I/O operations because Postgres needs to read data from disk, which can impact query performance.

Specially in case of too many foreign keys


It's alright
Shit Happens
- Steve Jobs -

Shared Buffers

Shared buffers in Postgres are part of the system's RAM (Random Access Memory). They are a portion of memory allocated by Postgres to cache frequently accessed data pages and index pages.

Default Value - 128MB

Recommended - 25% of RAM

Work Mem

It is a configuration parameter in Postgres that determines the amount of memory allocated for sorting and hashing operations within a single query. Increasing work_mem can help with query performance, especially for queries involving sorting large datasets. Setting the correct value of the work_mem parameter can result in less disk-swapping, and therefore far quicker queries

Default Value - 4 MB

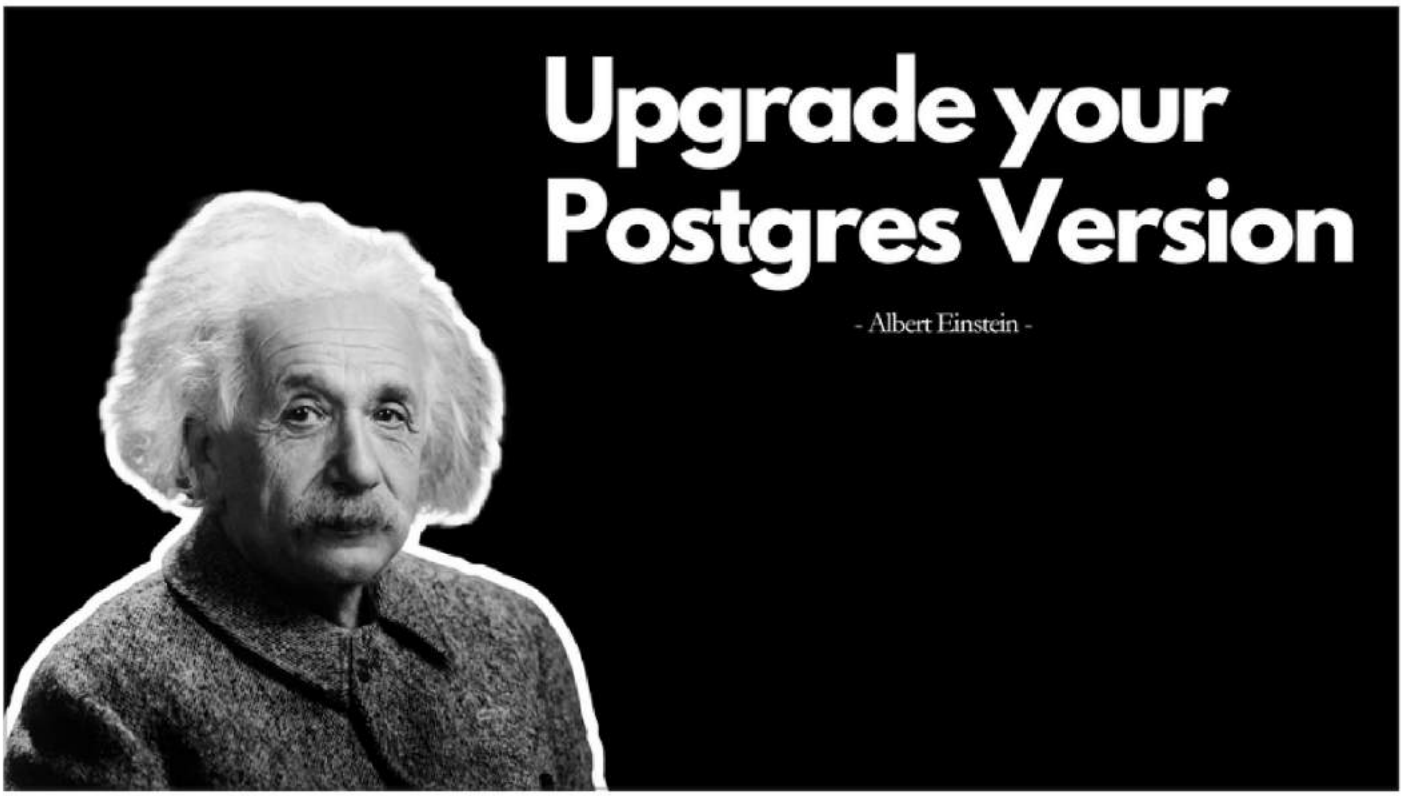
Recommended

*TOTAL RAM * 0.25 / MAX_CONNECTIONS*

Note: You might not want to completely rely on AWS RDS managers

If not UUIDs, what to use then?

An idempotency key needs to be unique for the time we want the request to be retryable, typically 24 hours or less. We prefer using an Universally Unique Lexicographically Sortable Identifier (ULID) for these idempotency keys instead of a random version 4 UUID. ULIDs contain a 48-bit timestamp followed by 80 bits of random data. The timestamp allows ULIDs to be sorted, which works much better with the b-tree data structure databases use for indexing. In one high-throughput system at Shopify we've seen a 50 percent decrease in INSERT statement duration by switching from UUIDv4 to ULID for idempotency keys.


Upgrade your
Postgres Version
- Albert Einstein -

```
db=# CREATE test_btree_dedup (n_unique serial, n_not_unique integer);
CREATE TABLE

db=# INSERT INTO test_btree_dedup (n_not_unique)
SELECT (random() * 100)::int FROM generate_series(1, 1000000);
INSERT 0 1000000

db=# CREATE INDEX ix1 ON test_btree_dedup (n_unique)     WITH (deduplicate_items = OFF);
CREATE INDEX

db=# CREATE INDEX ix2 ON test_btree_dedup (n_unique)     WITH (deduplicate_items = ON);
CREATE INDEX

db=# CREATE INDEX ix3 ON test_btree_dedup (n_not_unique) WITH (deduplicate_items = OFF);
CREATE INDEX

db=# CREATE INDEX ix4 ON test_btree_dedup (n_not_unique) WITH (deduplicate_items = ON);
CREATE INDEX
```

Next, compare the sizes of the four indexes:

COLUMN	DEDUPLICATION	SIZE
Not unique	Yes	6840 kB
Not unique	No	21 MB
Unique	Yes	21 MB
Unique	No	21 MB

Data deduplication

Starting at PostgreSQL 13, when B-Tree deduplication is activated, duplicate values are only stored once. This can make a huge impact on the size of indexes with many duplicate values and improves query performance significantly

In PostgreSQL 13 index deduplication in enabled by default, unless you deactivate it

```
-- Activating de-deduplication for a B-Tree index, this is the default:
CREATE INDEX index_name ON table_name(column_name) WITH (deduplicate_items = ON)
```

If you are migrating from PostgreSQL versions prior to 13, you need to rebuild the indexes using the REINDEX command in order to get the full benefits of index de-deduplication. You can rebuild the indexes concurrently so that you don't end up locking the table.

Another Disclaimer

Deduplication might now work for all data types. please check official Postgres documentation for more details

Pgx Scripts

https://github.com/pgexperts/pgx_scripts/blob/master/indexes/needed_indexes.sql

A collection of useful little scripts for database analysis and administration, created by our team at PostgreSQL Experts.

Table bloat
Index bloat
Unused Indexes
Needed Indexes
Foreign keys with no Indexes

Maybe the next time we can talk about above mentioned topics but for the time being if anybody is aware of what MVCC is and what vacuum is you'll understand what index bloat and table bloat are. So some good people on the internet have already written some scripts to identify potential issues. We can make use of them. Postgres also provides a lot of tools built in to monitor bloat and a lot of extensions have been created already to check the same.

References

https://hakibenita.com/postgresql-unused-index-size

https://www.cybertec-postgresql.com/

And many more blogs

Thank You