**Index JSON In Postgres**

To maximise query efficiency for a relational database is to index the columns that are often used for joining or conditions. The awesome thing about querying JSON in Postgres is that you can index it to further optimise query performance.

In the previous post, we had a look at the [**new JSON data ingestion strategy by using the power of Postgres**](https://www.mydatahack.com/new-json-data-ingestion-strategy-by-using-the-power-of-postgres). If you haven’t tried to load JSON directly into Postgres, you should check out the blog.

To index JSON in Postgres, the JSON column has to be the jsonb data type (stores data in the binary json format), instead of json (stores data as string). If you have your column set as json, you can change the data type as below. To injest JSON, you should always use jsonb for better query performance and indexing capability.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

ALTER TABLE mongodb.restaurants\_json

ALTER COLUMN data

SET DATA TYPE jsonb

USING data::jsonb;

[/cc]

Postgres supports a few different indexing method. If you are indexing a single node (such as id or email) for joining or searching, B-Tree is the best one. Hash indexes are discouraged for not being WAL-logged (meaning not crash-safe and not replicated). If you are interested, [this blog post](http://blog.andrebarbosa.co/hash-indexes-on-postgres/) explains it further.

GiN ([Generalised Inverted Index](https://www.postgresql.org/docs/current/static/gin.html)) is the way to go when the index maps many values to on row (like array or document). For further information about JSON Indexing variations, check out the blog post [here](http://bitnine.net/blog-postgresql/postgresql-internals-jsonb-type-and-its-indexes/?ckattempt=1).

Let’s have a look at how we can index the table we made in the [previous post](https://www.mydatahack.com/new-json-data-ingestion-strategy-by-using-the-power-of-postgres). Let’s index id assuming this is the join key to another table. For a single value, you can index it as string.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

Create Index ind\_btree\_id On mongodb.restaurants\_json Using BTREE ((data->>'Id'));

[/cc]

Let’s also index grades with GIN indexes. For GIN, you have to index it as jsonb data type.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

Create Index ind\_gin\_grades On mongodb.restaurants\_json Using GIN ((data->'grades'));

[/cc]

Now, let’s check out the indexes on the table. To check indexes, you need to query the pg\_indexes table.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

Select \* From pg\_indexes Where tablename = 'restaurants\_json';

[/cc]

Let’s drop indexes. [Drop Index](https://www.postgresql.org/docs/10/static/sql-dropindex.html) command takes a few optional options. I included concurrently and if exists. Concurrently prevents table locks when other processes are querying the table at the same time.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

Drop Index Concurrently If Exists mongodb.ind\_btree\_id;

Drop Index Concurrently If Exists mongodb.ind\_gin\_grades;

[/cc]

May the index be with you!