POSTGRESQL VS MONGODB: COMPARING DATABASES

In this article, we compare MongoDB and PostgreSQL.

PostgreSQL is a traditional RDBMS (relational database management system) SQL database, like Oracle and MySQL. PostgreSQL is free.

MongoDB is a no-schema, noSQL, JSON database. MongoDB has a free version, but they also have hosted and enterprise paid versions. Even the free version includes free cloud monitoring hosted on their site for your local installation.

(This article is part of our MongoDB Guide. Use the right-hand menu to navigate.)

PostgreSQL

Here are the basics of PostgreSQL.

A traditional RDBMS

A traditional RDBMS (relational database management system), such as PostgreSQL, has a script schema and requires a primary key. You cannot add data to it unless the data column already exists.

The PostgreSQL shell

The PostgreSQL shell is slightly different from Oracle or MySQL. You log into it like this:

```
sudo su - postgres
postgres@paris2:~$ psql
```

```
psql (9.5.21)
Type "help" for help.

postgres=#
```

Then you use the slash (\) to run commands that are not SQL commands.

Create a table

Here is how you create a table and schema in PostgreSQL:

```
create table expenses (
TransactionDate date,
PostDate date,
Description text,
Category text,
Type text,
Amount float8
);
```

Connect to a database

```
\c expenses
You are now connected to database "expenses" as user "postgres".
```

List tables

```
\dt
                  List of relations
 Schema |          Name           | Type  |  Owner
--------+-------------------------+-------+----------
 public | chase                   | table | postgres
```

Run a query

The query below does aggregation.

```
select  category, sum(amount) from chase group by category;
       category        |  sum
-----------------------+-------
                       | 19389
 Education             |  -216
 Health & Wellness     | -3154
 Personal              | -1582
 Automotive            |   -33
 Shopping              | -4479
 Travel                | -7026
```

```
 Fees & Adjustments    |  -316
 Entertainment         |  -274
 Gas                   |  -139
 Home                  | -1409
 Food & Drink          | -1926
 Bills & Utilities     | -3114
 Professional Services |  -114
 Groceries             | -1720
(15 rows)
```

Add data

```
INSERT INTO expenses(
transactiondate , postdate,  description,   category , type ,amount ) VALUES
('10-July-2020', '10-July-2020', 'coffee shop', 'restaurants', 4.50);
```

MongoDB

Here are the basics of MongoDB.

A JSON database

Unlike PostgreSQL and other RDMBS, a JSON database, like MongoDB, has no schema so you can put anything into it. Contrast that with a SQL database where you must define its structure before you put data.

JSON looks like this:

```
data: {
    attribute:  value
}
```

Or an array of JSON is like this:

```
data:
```

The MongoDB shell

To open the MongoDB shell, you just type **mongo**:

```
mongo
MongoDB shell version v3.6.17
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("fd8960a6-d6fd-44f7-
a6c5-5257438934e3") }
```

Create a database

```
use sales
switched to db sales
```

Create a collection

A **collection** is a set of related tables.

```
db.createCollection("inventory")
{ "ok" : 1 }
```

Add a record

```
db.inventory.insert({product: "blue shoes", add: 2})
WriteResult({ "nInserted" : 1 })
```

Query

```
db.inventory.find({product: "blue shoes"})
{ "_id" : ObjectId("5f0808a6434f5a1831100614"), "product" : "blue shoes",
"add" : 2 }
```

Using JavaScript in the MongoDB shell

One very powerful feature with the MongoDB shell is it supports JavaScript. This means you can define functions and save queries as variables.

For example, here is how you define Connecticut by drawing a square around it on a map. This statement uses the GeoJSON geographical query features of MongoDB to do that.

```
var connecticut = db.address.find ({location:
     {$geoWithin:
       {$geometry: {
       type: "Polygon",
       coordinates:
 ,
   ,
   ,
   ,
   ]]
}}}});
```

Joining tables

In the 1970s, when IBM published the paper which described the SQL language and the database that Larry Ellison later developed into Oracle, disk space and memory was expensive. So, the adopted practice became to not repeat data, as that wasted expensive space and memory. That

practice is called database **normalization**.

Today, of course, storage and memory are cheap. noSQL database designers don't do normalization. That simplifies things somewhat as they don't require foreign keys and all the other design elements that create relations between tables. But it also creates its own set of problems. (For example, SQL databases are still better suited than noSQL databases for transactional systems like accounting because it's easier to maintain what is called **referential integrity**—but that is a whole other topic.)

When data is kept in two tables and you want to bring it together temporarily in a read-only structure such as to create a report you execute what is called a **join**. This makes the intersection of two sets.

It works like this:

```
select sales.productName, product.productName from sales, product where
product = upcCode;
```

The downside is this takes a lot of computing power, memory, and storage to run on a large distributed database.

MongoDB has indices but no joins

MongoDB does not encourage or really allow joins. Instead you would put related documents (records) inside one another so as to have them all in one place. This is kind of awkward for something like a sales system as you would have something like this:

```
{ product : "shoes",
    sales:
}
```

That's awkward to maintain in a transactional system but would work well for other types of systems.

Yet, while MongoDB does not support joins, it does allow indexes, which is a necessary feature of joins. This speeds up queries.

Additional resources

So that is a side-by-side comparison of the two databases. To learn more about this topic, browse the BMC Big Data & Machine Learning Blog and these resources:

- MongoDB Guide, with 10+ articles and tutorials, including:
  - How to Set Up a Cluster in MongoDB
  - Index management in MongoDB
  - MongoDB vs Cassandra: NoSQL Databases Compared
- Using Tableau with PostgreSQL
- DBMS: An Intro to Database Management Systems
- Enabling the Citizen Data Scientists