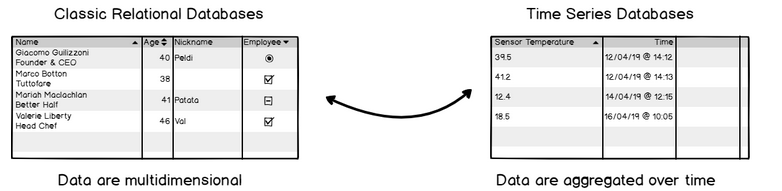
All your life, you have dealt with relational databases like MySQL, or SQL Server. You may also have dealt with NoSQL databases like MongoDB or DynamoDB.

Those systems are based on the fact that you have tables. Those tables contain columns and rows, each one of them defining an entry in your table. Often, those tables are specifically designed for a purpose : one may be designed to store users, another one for photos and finally for videos. Such systems are efficient, scalable and used by plenty of giant companies having million of requests on their servers.

Time series databases work differently. Data are still stored in ‘collections’ but those collections share a common denominator : **they are aggregated over time**.

Essentially, it means that for every point that you are able to store, you have a **timestamp** associated with it.



But.. couldn’t we use a relational database and simply have a column named ‘time’? Oracle for example includes a TIMESTAMP data type that we could use for that purpose. You could, but that would be **inefficient**.

### Why do we need time series databases?

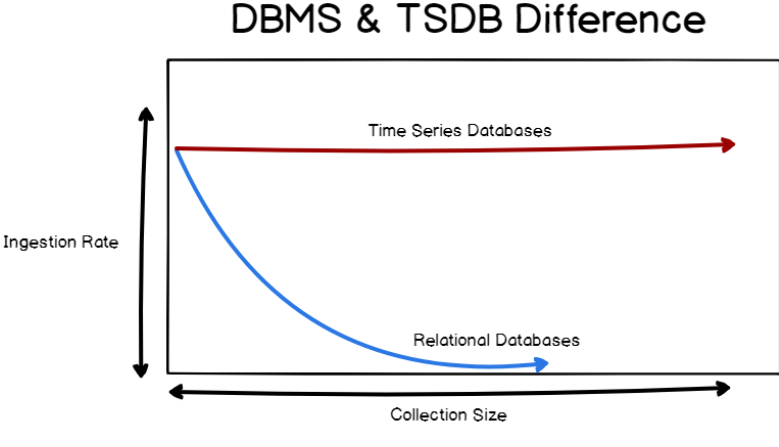
Three words : **fast ingestion rate.**

Time series databases systems are built around the predicate that they need to ingest data in a fast and efficient way.

Indeed, relational databases do have a fast ingestion rate for most of them, from 20k to 100k rows per second. However, the ingestion is not constant over time. Relational databases have one key aspect that make them slow when data tend to grow : **indexes**.

When you add new entries to your relational database, and if your table contains indexes, your database management system will repeatedly re-index your data for it to be accessed in a fast and efficient way. As a consequence, **the performance of your DBMS tend to decrease over time**. The load is also increasing over time, resulting in having difficulties to read your data.

Time series database are optimized for a **fast ingestion rate**. It means that such index systems are optimized to index data that are aggregated over time : as a consequence, the ingestion rate does not decrease over time and stays quite stable, around **50k to 100k lines per second on a single node**.



**Specific concepts about time series databases**

On top of the fast ingestion rate, time series databases introduce concepts that are very specific to those technologies.

One of them is **data retention**. In a traditional relational database, data are stored permanently until your decide to drop them yourself. Given the use-cases of time series databases, you may want not to keep your data for too long : either because it is **too expensive** to do so, or because you are not that interested in old data.

Systems like InfluxDB can take care of dropping data after a certain time, with a concept called **retention policy** (explained in details in part two). You can also decide to run **continuous queries** on live data in order to perform certain operations.

You could find equivalent operations in a relational database, for example ‘jobs’ in SQL that can run on a given schedule.

### A Whole Different Ecosystem

Time series databases are very different when it comes to the ecosystem that orbits around them. In general, relational databases are surrounded by applications : web applications, softwares that connect to it to retrieve information or add new entries.

Often, a database is associated with one system. Clients connect to a website, that contacts a database in order to retrieve information. TSDB are built for **client plurality** : you do not have a simple server accessing the database, but a bunch of different sensors (for example) inserting their data at the same time.

As a consequence, tools were designed in order to have efficient ways to **produce data or to consume it**.

### Data consumption

### Data consumption is often done via monitoring tools such as Grafana or Chronograf. Those solutions have built-in solutions to visualize data and even make custom alerts with it. Those tools are often used to create live dashboards that may be graphs, bar charts, gauges or live world maps.

### 

### Data production

### Data production is done by agents that are responsible for targeting special elements in your infrastructure and extract metrics from them. Such agents are called “monitoring agents“. You can easily configure them to query your tools on a given time span. Examples are Telegraf (which is an official monitoring agent), CollectD or StatsD

### 