What the heck is time-series data (and why do we need a time series database)?

A primer on time-series data, and why you may not want to use a “normal” database to store it.

Here’s a riddle: what do self-driving Teslas, autonomous Wall Street trading algorithms, smart homes, transportation networks that fulfill lightning-fast same-day deliveries, and an open-data-publishing NYPD have in common?

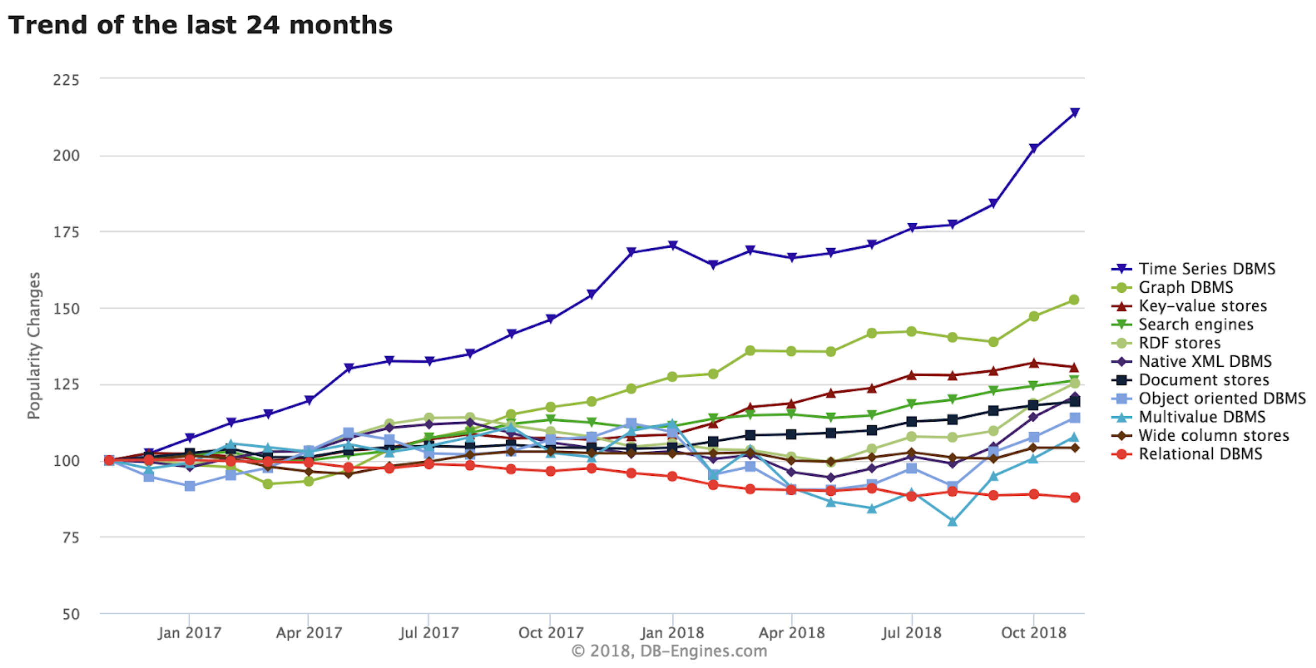
For one, they are signs that our world is changing at warp speed, thanks to our ability to capture and analyze more and more data in faster and faster ways than before.

However, if you look closely, you’ll notice that each of these applications requires a special kind of data:

- Self-driving cars continuously collect data about how their local environment is changing around them.
- Autonomous trading algorithms continuously collect data on how the markets are changing.
- Our smart homes monitor what’s going on inside of them to regulate temperature, identify intruders, and respond to our beck-and-call (“Alexa, play some relaxing music”).
- Our retail industry monitors how their assets are moving with such precision and efficiency that cheap same-day delivery is a luxury that many of us take for granted.
- The NYPD tracks its vehicles to allow us to hold them more accountable (e.g., for analyzing 911 response times).

These applications rely on a form of data that measures how things change over time. Where time isn’t just a metric, but a primary axis. This is time-series data and it’s starting to play a larger role in our world.

Software developer usage patterns already reflect this. In fact, over the past 24 months time-series databases (TSDBs) have steadily remained the fastest growing category of databases:



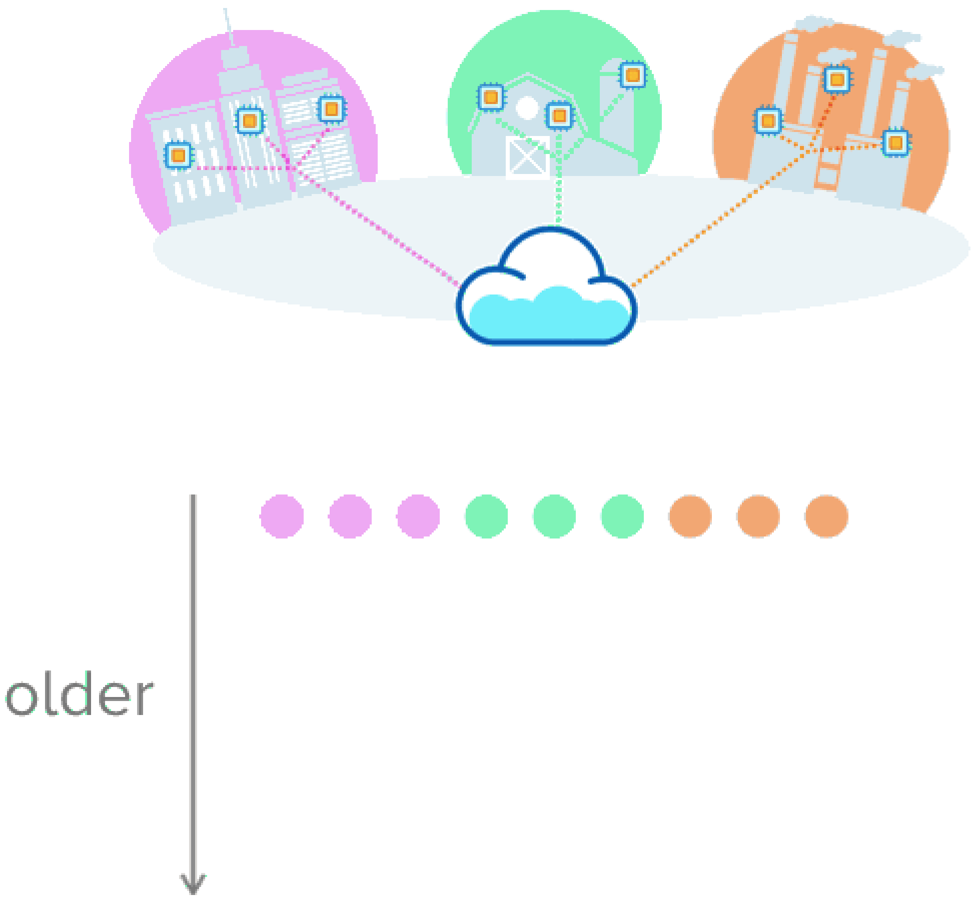
As the developers of an open source time-series database, my team and I are often asked about this trend. So I'll start with a more in-depth description of time-series data and then jump into when would you would need a time-series database.

What is time-series data?

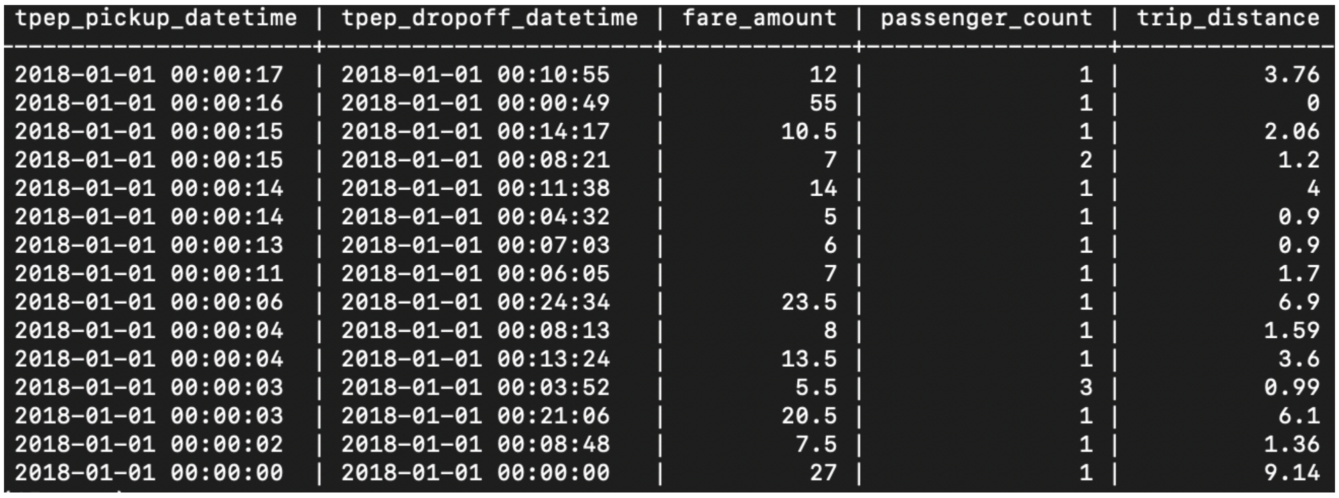
Some think of "time-series data" as a sequence of data points, measuring the same thing over time, stored in time order. That's true, but it just scratches the surface.

Others may think of a series of numeric values, each paired with a timestamp, defined by a name and a set of labeled dimensions (or "tags"). This is perhaps one way to model time-series data, but not a definition of the data itself.

Here's a basic illustration. Imagine sensors collecting data from three settings: a city, farm, and factory. In this example, each of these sources periodically sends new readings, creating a series of measurements collected over time.



Here's another example, with real data from the City of New York, showing taxicab rides for the first few seconds of 2018. As you can see, each row is a "measurement" collected at a specific time:



```
 tpep_pickup_datetime | tpep_dropoff_datetime | fare_amount | passenger_count | trip_distance
----------------------+-----------------------+-------------+-----------------+---------------
 2018-01-01 00:00:17  | 2018-01-01 00:10:55   |          12 |               1 |          3.76
 2018-01-01 00:00:16  | 2018-01-01 00:00:49   |          55 |               1 |             0
 2018-01-01 00:00:15  | 2018-01-01 00:14:17   |        10.5 |               1 |          2.06
 2018-01-01 00:00:15  | 2018-01-01 00:08:21   |           7 |               2 |           1.2
 2018-01-01 00:00:14  | 2018-01-01 00:11:38   |          14 |               1 |             4
 2018-01-01 00:00:14  | 2018-01-01 00:04:32   |           5 |               1 |           0.9
 2018-01-01 00:00:13  | 2018-01-01 00:07:03   |           6 |               1 |           0.9
 2018-01-01 00:00:11  | 2018-01-01 00:06:05   |           7 |               1 |           1.7
 2018-01-01 00:00:06  | 2018-01-01 00:24:34   |        23.5 |               1 |           6.9
 2018-01-01 00:00:04  | 2018-01-01 00:08:13   |           8 |               1 |          1.59
 2018-01-01 00:00:04  | 2018-01-01 00:13:24   |        13.5 |               1 |           3.6
 2018-01-01 00:00:03  | 2018-01-01 00:03:52   |         5.5 |               3 |          0.99
 2018-01-01 00:00:03  | 2018-01-01 00:21:06   |        20.5 |               1 |           6.1
 2018-01-01 00:00:02  | 2018-01-01 00:08:48   |         7.5 |               1 |          1.36
 2018-01-01 00:00:00  | 2018-01-01 00:00:00   |          27 |               1 |          9.14
```

There are many other kinds of time-series data. To name a few: DevOps monitoring data, mobile/web application event streams, industrial machine data, scientific measurements.

These datasets primarily have 3 things in common:

1. The data that arrives is almost always recorded as a new entry
2. The data typically arrives in time order
3. Time is a primary axis (time-intervals can be either regular or irregular)

In other words, time-series data workloads are generally “append-only.” While they may need to correct erroneous data after the fact, or handle delayed or out-of-order data, these are exceptions, not the norm.

You may ask: How is this different than just having a time-field in a dataset? Well, it depends: how does your dataset track changes? By updating the current entry, or by inserting a new one?

When you collect a new reading for sensor_x, do you overwrite your previous reading, or do you create a brand new reading in a separate row? While both methods will provide you the current state of the system, only by writing the new reading in a separate row will you be able to track all states of the system over time.

Why do I need a time-series database?

You might ask: Why can’t I just use a “normal” (i.e., non-time-series) database?

The truth is that you can, and some people do. Yet why are TSDBs the fastest growing category of databases today? Two reasons: (1) scale and (2) usability.

Scale: Time-series data accumulates very quickly. (For example, a single connected car will collect 4,000 GB of data per day.) And normal databases are not designed to handle that scale. Relational databases fare poorly with very large datasets; NoSQL databases fare better at scale, but can still be outperformed by a database fine-tuned for time-series data. In contrast, time-series databases (which can be based on relational or NoSQL databases) handle scale by introducing efficiencies that are only possible when you treat time as a first class citizen. These efficiencies result in performance improvements, including higher ingest rates, faster queries at scale (although some support more queries than others), and better data compression.

Usability: TSDBs also typically include functions and operations common to time-series data analysis such as data retention policies, continuous queries, flexible time aggregations, etc. Even if scale it not a concern at the moment (e.g., if you are just starting to collect data), these features can still provide a better user experience and make your life easier.

This is why developers are increasingly adopting time-series databases and using them for a variety of use cases:

- Monitoring software systems: Virtual machines, containers, services, applications
- Monitoring physical systems: Equipment, machinery, connected devices, the environment, our homes, our bodies
- Asset tracking applications: Vehicles, trucks, physical containers, pallets
- Financial trading systems: Classic securities, newer cryptocurrencies
- Eventing applications: Tracking user/customer interaction data
- Business intelligence tools: Tracking key metrics and the overall health of the business
- (and more)

Even then, you’ll need to pick a time-series database that best fits your data model and write/read patterns.