



Time-Series Databases and Amazon Timestream

Philip Winston

From the Editor

Audrey Lawrence of Amazon discusses time-series databases and the new Amazon database offering, Amazon Timestream. Philip Winston speaks with Lawrence about data modeling, ingestion, queries, performance, lifecycle management, hot data versus cold data, operating at scale, and the advantages of a serverless architecture. We provide summary excerpts below; to hear the full interview, visit <http://www.se-radio.net> or access our archives via RSS at <http://feeds.feedburner.com/se-radio>.—Robert Blumen

Philip Winston: What is a time-series database?

Audrey Lawrence: It's a database that is purpose built for data points that are measured over time. Time-series databases are useful for measuring events that change over time. They allow for efficient collection, storage, and analysis on top of these data—from doing streaming analytics in real time to training machine learning models off historical time-series data. They allow complex analyses on top of time-series data.

Why can't developers store time-series data in a Structured Query Language

(SQL) database, a key-value store, or some type of long-existing database?

You run into problems scaling those kinds of systems out for performance and costs. Because time-series databases are purpose built for this type of data, they are more efficient at ingesting, storing, and querying. There are also a lot of plug-ins you can use to easily collect and ingest these metrics and visualization tools that help later with querying. The solutions that teams may need to store time-series databases can get complex and take a lot of developer time to build, maintain, and scale out. Often the metrics stored in time-series databases are critical business or operational metrics that must be timely and correct. Time-series databases

can solve problems more cost-effectively with better performance.

How does someone create a data model for a time-series database? How is this different from modeling for an SQL database?

The core unit is a record, a single measurement at a point in time. A series of these measurements over time makes up the time series. For each time series, you have a group of attributes or dimensions that describe that time series. You store those in databases and tables to organize the data and control access, retention, and so on. An example would be a fleet of hosts in a system. You could have a DevOps database and a host-metrics table. The dimensions stored with the time series

would be things like the region where that fleet is located, hostname for the different posts, operating system version, and the actual metrics you're collecting, like CPU utilization, memory utilization, and traffic input-output.

Time-series databases also tend to have a flexible schema. For Timestream, outside of creating a database and a table, you don't declare your measurements and dimensions up front; you do that when you're ingesting data, and Timestream will figure that out and store the data appropriately. In an SQL store, you typically have tables with one row per measurement, so each of these attributes would be in a denormalized view in that row. These dimension values would be repeated for each measurement, which can be costly because you're storing a lot more data. And you also have to read through those data when you're doing queries. Typically, you'll add an index on the Time column and on some of the dimensions, so that you can have queries based on those dimensions. This also gets hard to balance because you don't want to add too many indices. And then ingestion will cost a lot. But you also need to leverage these indices at query time, or sometimes the queries can get way too slow.

What specifically about queries is faster with a time-series database compared to SQL?

Time-series databases optimize how they store data, such as storing all of the data points for one series together and then storing the similar series together. When you're serving a query, you're able to just read the data that are relevant to serve for your query and efficiently index these data as well so that you can quickly store all of the data points. When queries come in, we

want to be able to efficiently look up where the data are and then prune out the tiles that aren't relevant to serving that query. Being able to distribute queries is also important to keep the

so being able to massively scale out data queries over time is important.

What difficulties does scale present to ingestion? If we have many

We do have to deal with this case, and we have a manner of performing batching and load balancing with our ingestion service.

query performance good as the data set grows. With the time-series use case, we know that as time goes on, we'll continue growing that data set,

data sources, all high volume, and the total amount of data being ingested is huge, how can we cope with that?



SOFTWARE ENGINEERING RADIO

Visit www.se-radio.net to listen to these and other insightful hour-long podcasts.

RECENT EPISODES

- 499—Render's Uma Chingunde talks with host Jeremy Jung and compares building a PaaS with her previous experience running the Stripe Compute team.
- 498—Host Felienne and James Socol of Policygenius discuss continuous integration and continuous delivery, and ways to test and deploy software quickly and easily.
- 497—Richard L. Sites talks about his new book *Understanding Software Dynamics* with host Philip Winston.

UPCOMING EPISODES

- Sergey Gorbunov talks to host Philip Winston about blockchain interoperability.
- Omer Katz and host Nikhil Krishna discuss distributed task queues using Celery.
- Host Brijesh Ammanath talks to Bob Ducharme regarding creating technical documentation for software projects.




ABOUT THE AUTHOR



PHILIP WINSTON is a software engineering consultant and contractor through his company, Tobeva Software, Winchester, Virginia, 22602, USA. Contact him at philip@tobeva.com.

What trends have you noticed in time-series databases? Where do you see things evolving in the next three to five years?

We'll see a lot of evolution in time-series databases just in their growth and adoption. We're just getting started in terms of usage of time-series databases and the functionality that they can serve. So much data out there across so many industries are time series in nature. Most data you're collecting have some sort of time element such that you tend to not care so much about the current state of things as you do about historical data. As machine learning expands, and we make devices and different operators smarter, giving them insight into these historical data is really important. Also, as time-series databases become increasingly used, we'll see more use cases across different industries. 

You can't handle that volume of data without distributing the load somehow. We distribute the load by cutting more into our 2D space of tiles. If we're seeing a lot of increased traffic to these tiles, we will further split them down so we can have more resources working on ingestion for that table. Similarly, if we see workload decrease to the table, we can merge tiles back together so we don't have resources not working on a lot of incoming data. Time-series data are usually immutable, but sometimes they are

not. We need to support data updates and data deduplication if we are receiving duplicate data points and also out-of-order data or even data that could be really old. Time-series data often do come in like this. That allows us to scale our system and not have a lot of conflicts when multiple different hosts are getting the exact same data points that they need to ingest. But we do have to deal with this case, and we have a manner of performing batching and load balancing with our ingestion service.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscribe to *IEEE Software* by visiting www.computer.org/software.

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.

Reuse Rights and Reprint Permissions: Educational or personal use of this material is permitted without fee, provided such use: 1) is not made for profit; 2) includes this notice and a full citation to the original work on the first page of the copy; and 3) does not imply IEEE endorsement of any

third-party products or services. Authors and their companies are permitted to post the accepted version of IEEE-copyrighted material on their own web servers without permission, provided that the IEEE copyright notice and a full citation to the original work appear on the first screen of the posted copy. An accepted manuscript is a version that has been revised by the author to incorporate review suggestions, but not the published version with copyediting, proofreading, and formatting added by IEEE. For more information, please go to: http://www.ieee.org/publications_standards/publications/rights/paperversionpolicy.html. Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 or pubs-permissions@ieee.org. Copyright © 2022 IEEE. All rights reserved.

Abstracting and Library Use: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.