CONVERGE 360 PAPER
An 1105 MEDIA company

Redmond MAGAZINE

EMBEDDED DATABASE MANAGEMENT FOR IOT AND EDGE DEVELOPERS

By John K. Waters

EXECUTIVE SUMMARY

The advent of the Internet of Things (IoT) and the proliferation of Edge Computing have presented software developers with a range of significant challenges—not the least of which is the management of the massive amount of data fueling these disruptive trends. Third parties, organizational systems, and billions of end-point devices—everything from industrial sensors and actuators to smart phones and autonomous vehicles—are generating unprecedented volumes and varieties of data at previously unheard of velocities.

These "three Vs" define *Big Data*, and developers are expected to make the most of it in the IoT and Edge applications they build—not just manage it, but generate near real-time analytics *at the point of action* for business insights, while providing overall data governance and security.

This analysis layer adds complexity for developers, and the old client-server models won't give them what they need in this environment. Whether you're pushing your data to a terrestrial server right next door or the cloud, the performance lag inherent in a system that puts the app on one machine and the data it needs on another becomes a defeating factor when the data needs to be processed in near real time. In manufacturing, health care, and financial services, for example, milliseconds matter. Neither will the practice of simply manipulating data in some memory allocated space using file systems provide a solution going forward.

The solution for a growing number of developers lies in the *embeddable database*, the database and management software that is effectively part of the application itself. This white paper examines the key concepts and enabling technologies of embeddable DBs that IoT and Edge developers need to understand.

DEFINITIONS

Database expertise varies among developers, as does familiarity with the current terms of art in the IoT and Edge Computing space, so it makes sense to define a few terms. The truth is, the database is no longer a hunk of shrink-wrapped software deployed and tended by a database administrator (DBA). Data-intensive apps are becoming the norm, and developers need to understand them.

Embedded

The focus of this paper is the *embeddable database*, which is a database (DB) and various levels of database management software integrated into, or very tightly bundled with, an application that

needs direct or fast access to the data it is manipulating. The DB and the app using it are running on the same machine, and they communicate via procedure calls, so there's very little latency.

For the purposes of this paper, the embedded database category does not include *in-memory databases* (IMDB) in which the data is stored entirely in the main memory. Those systems are light, simple, and fast, but provide no permanent storage.

Edge Computing

Edge Computing refers to distributed information architecture in which the processing of data occurs where it is generated, far from a centralized data-center, at the "edge" of the network. That processing is done on IoT devices and remote gateways. Edge Computing has also been described as a kind of decentralized data persistence that occurs on or near the devices that generate the data.

THE SOLUTION FOR A GROWING NUMBER OF DEVELOPERS LIES IN THE *EMBEDDABLE DATABASE*, THE DATABASE AND MANAGEMENT SOFTWARE THAT IS EFFECTIVELY PART OF THE APPLICATION ITSELF.

Also, an embedded database isn't an *embedded* system, which is special-purpose software with a dedicated function installed on a piece of hardware.

As of this writing, embedded databases are used primarily by independent software vendors (ISVs), original equipment manufacturers (OEMs), and systems integrators (SIs). But the IoT and Edge ecosystem is expanding at warp speed, and the market is likely to expand with it.

Data analytics is the process of assessing data for actionable insights. *Embedded analytics* is the use of reporting and analytic capabilities in transactional business applications. You can't have embedded analytics without an embedded database.

Fog Computing, a term coined by Cisco Systems in 2014, effectively means the same thing as Edge Computing, though it's more directly associated with Cloud Computing. As Cisco defines it, the Fog "extends the cloud to be closer to the things that produce and act on IoT data." The term has been called a marketing take on Edge, but it has been catching on. ("Fogging" has even become a verb.)

IoT

The *Internet of Things* is part of the popular lexicon now, but just to be clear, IoT refers to the vast and ever-expanding network of physical devices linked through the Internet and via enterprise intranets. There are billions of them now,

everything from smart phones to temperature sensors, kitchen appliances to cars. The software installed on them gathers, stores, and analyzes data, either locally or through a connection with a separate database.

An IoT gateway (sometimes called an intelligent gateway or a control tier) can take the form of a physical hardware appliance or a piece of dedicated software. It serves as the connection between the cloud and the universe of controllers, sensors, and intelligent devices.

Database Management System

A *database management system* (DBMS) is software that controls the storage, retrieval, deletion, security, and integrity of data within a database. Embedded database solutions are sometimes called embedded DBMSs.

Relational Database

A *relational database* is a DB that organizes information into sets of tables with columns and rows. Each table contains data that relates to data in the other tables. The relationships are pre-defined and the data can be accessed or reassembled in different ways without having to reorganize the tables. These types of databases are designed for transactional and strongly consistent online transaction processing (OLTP) applications. Relational database management systems (RDBMSs) are preferred for OLTP, but not for online analytical processing (OLAP).

SQL

The *Structured Query Language* (SQL) is the language most commonly associated with relational databases. In fact, "relational database" is almost synonymous with "SQL Databases." SQL statements are used both for interactive queries for information from a relational database and for gathering data for reports. It is considered a must-master tool for developers working with databases.

NoSQL

A *NoSQL* database is a non-relational database—no tables with data organized in rows and columns—and they come with looser consistency models than traditional relational databases. NoSQL databases use a variety of data models, including document, graph, key-value, in-memory, and search. They're optimized for applications that require large volumes of data, low latency, and flexible data models.

Flat File

A *flat file* is a plain text database from which all word processing and other structure characters or markups have been removed. It contains a single table of data with one record per line. One of the most common flat file examples is a comma-separated values (CSV) file, in which table data is gathered in lines of American Standard Code for Information Interchange (ASCII) text with the value from each table cell separated by a comma and each row represented by a new line.

"AS THE VOLUME AND VELOCITY OF DATA INCREASES, SO DOES THE INEFFICIENCY OF STREAMING ALL THIS INFORMATION TO A CLOUD OR DATACENTER FOR PROCESSING." —SANTHOSH RAO, GARTNER ANALYST

ACID Compliance

ACID (Atomicity, Consistency, Isolation, and Durability) is an acronym for the four attributes of a database transaction that guarantee its validity. All database transactions must be *ACID Compliant* to ensure data integrity. The attributes are:

Atomicity: in a transaction involving two or more discreet parts, if a part fails, the whole transaction fails.

Consistency: data written to the database as part of the transaction must adhere to all defined rules and restrictions.

Isolation: a transaction in process and not yet committed must remain isolated from any other transaction.

Durability: all the changes made to the database are permanent once a transaction is successfully completed.

THE LANDSCAPE

Why should developers care about IoT, Edge Computing, and all this arcane database lore? They're not DBA's, after all, and they're already coping with some significant challenges associated with accelerating application release cadences.

Ignoring these trends is simply not an option. According to the industry analysts at Gartner, within the next four years 75 percent of the data generated in the enterprise will be created and processed outside a traditional datacenter or cloud. Gartner analyst Santhosh Rao summarized the problem this trend is creating for IT organizations in a 2018 report ("What Edge Computing Means for Infrastructure and Operations Leaders"): "As the volume and velocity of data increases, so does the inefficiency of streaming all this information to a cloud or datacenter for processing."

And virtually every industry on the planet is poised to invest significant resources in reaction to these trends. In another recently published report ("Worldwide Semiannual Internet of Things Spending Guide"), IDC analysts predict that IoT spending will reach $1.2 Trillion in 2022.

These predictions should reassure developers considering their own IoT and Edge strategies. While developers may be most interested in leveraging APIs that have functionality behind them that delivers their prescribed outcomes, when it comes to APIs for management of the data within their IoT and Edge applications, investing in technologies that enable data processing closer to the edge of the network allows organizations to analyze that data in near real-time, which has quickly become an essential capability across many industries, including manufacturing, health care, telecommunications, and finance. And those industries

need developers with the skills and the right APIs and underlying data management functionality to make that happen.

USE CASES

The universe of use cases for embeddable databases is ever expanding, but a list of current notable examples would include:

- In intermittent connectivity and limited bandwidth environments, in which a persistent disconnection would be catastrophic where data is being ingested and, at a minimum, being prepared/processed/formatted for later analysis.
- When analytics are being embedded in the device or grid that require time, frequency, or other series data for real-time decision making (even better if it's against a changing baseline dataset). Increasingly, simple analytics are being used locally to avoid a deluge of data in the cloud (it's not cheap to keep data in AWS/Azure, and fractions of pennies, even for depreciation on private cloud storage, adds up).
- Over the next few years, simple and unsupervised machine learning routines, particularly for nodal processing in deep learning networks, will reside on devices or just above them at a gateway level; this will require local persistent and distributed data management.
- For situations involving high data ingestion rates, such as video streaming.
- Mesh sensor networks with peer-to-peer communication and control.
- Local, intelligent sensor grid or heterogeneous sensors with interdependencies that need data governance, security, or performance monitoring—not the data or device itself, but the metadata that deals with them.
- Intelligent capital equipment with integrated instrumentation, complex process, and regulatory oversight. Increasingly expensive machinery and vehicles will be sold as a service; the only way to effectively do this is by instrumenting them extensively.

> "DATA IS THE FUEL THAT DRIVES THESE ENGINES, AND DEVELOPERS NEED THE RIGHT COMPONENTS AROUND THAT BASIC COMBUSTION ENGINE."
>
> —LEWIS CARR, MARKETING DIRECTOR, ACTIAN

THE KISS CONUNDRUM

Another question these trends raise: why not just use flat files or SQLite, the popular open source database? Developers are admonished to "keep it simple, stupid," and embeddable databases seem like a complication. Why reinvent the wheel? Because the complexities of data management and analysis are growing and the ability to leverage that resource via embedded analytics is becoming a key competitive

advantage. Developers must rethink their options, while still trying to adhere to this adage.

An all-too common strategy today among developers is to rely on flat file systems associated with operating systems running on their platforms, or, in the absence of an OS, in local memory. Flat files are a disaster in IoT and Edge environments. As embedded database provider Actian noted in a recent report ("Top 10 Reasons Friends Don't Let Friends Use Flat Files"), to ensure flat file data consistency and avoid corruption, developers must code create, read, update, and delete (CRUD) logic to store or retrieve data with care, and coding may need to vary across APIs and file systems.

SQLite is a multiplatform database widely deployed in the enterprise as an embedded data management solution. It's popular because it's quick and easy and leverages existing SQL developer expertise. It's definitely a step up from flat file systems. But it has some serious drawbacks, as Actian points out in another report ("Ten Great Reasons to Upgrade to Actian Zen from SQLite"), it wasn't meant as a database for multiple apps to use simultaneously, which becomes a serious limitation as apps evolve, adopt microservice architectures, and spread across virtual server instances and out to well-resourced client devices. Also, using SQLite often requires data conversion and mapping across platforms, which can slow design and coding through multiple APIs, adding ETL overhead, and generating maintenance and support headaches. Because it has no server capabilities, SQLite cannot handle concurrent reads and writes. And doesn't provide full ANSI SQL support, so some SQL calls embedded in application code require workarounds to move to or from SQLite.

Actian's marketing director, Lewis Carr, summed up the situation succinctly in a recent interview: "Data is the fuel that drives these engines, and developers need the right components around that basic combustion engine. They need to replace the old carburetor with an overhead CAM 24-valve fuel injection system, so to speak."

EMBEDDABLE DATABASE ESSENTIALS

So, what should developers look for in a modern, embeddable database solution? Generally speaking, an embeddable database needs to be small, fast, versatile, and reliable, with a short code-path, programmable interface for tight application coupling. But there are some specific features and capabilities IoT and Edge application developers will want to consider as they go about the process of choosing an embeddable DB.

SQL *and* NoSQL Support

Look for support for NoSQL programmatic API-based database access *and* SQL relational database access to a single data set. The SQL access covers reporting and local transactions, while the NoSQL access provides performance and local analytics support and leverage of a wide range of programming APIs. This is a must-have feature.

MODULARITY SHOULD BE CONSIDERED A KEY REQUIREMENT, BECAUSE DIFFERENT IOT AND EDGE SOLUTIONS HAVE DIFFERING LEVELS OF RESOURCES.

Modularity

Look for solutions with modular architectures that scale from a core set of libraries capable of single-user client data management to a direct key-value store engine, and even up to a full-edged, enterprise-grade server capable of supporting thousands of users on multicore, VM cloud environments. Modularity should be considered a key requirement, because different IoT and Edge solutions have differing levels of resources. It might be a 32-bit ARM-based sensor with a 16MB DRAM module, a 4GB flash drive, running Android Things; or a 64-bit Intel-based Single board computer embedded in an MRI machine with 256 GB DRAM, 1 TB SDD drive, running Windows Server 2019. These are literal opposites, but they're part of the IoT app spectrum.

Multiple Platform Support

A critical feature: Unlike the desktop world, which is still dominated by Wintel, the IoT and Edge Computing universe is more fragmented. Developers must accommodate macOS, Android, variants of Linux, and more. This is also true of operation technology (OT) environments in industrial settings. The IoT and Edge Computing platform picture is likely to remain very fragmented for the foreseeable future.

Times Series Data

A survey of IoT developers conducted by the Eclipse Foundation's IoT Group in early 2018 found that a wide variety of data is being collected by these applications across all industries, but about 62 percent is *time series data*. "Time series" are measurements or events that are tracked, monitored, down sampled, and aggregated over time—things like server metrics, application performance monitoring, network data, sensor data, events, etc. Time series data occurs wherever the same measurements are recorded on a regular basis. Device information and log data were second and third in the survey.

Also look for:

- 32-bit and 64-bit support on Intel and ARM
- Callable data management APIs from all popular programming languages
- Free to develop and no hidden surprise costs (often interpreted as Open Source)
- A quick basic set of APIs – subset that's simple, straightforward and easy to use – open, close, read, write, etc.
- A more comprehensive set of APIs that handles things that you don't want to do yourself or reinvent the wheel: Index, sort, search, transpose a matrix, etc.

- No changes to the APIs as development moves from one platform or programming language to another
- Any changes required to deal with underlying file systems, such as encryption/decryption, backup, and recovery on different platforms, should be abstracted and handled by the data management system
- The deployed software, including the embedded DBMS, should be developer configurable, so deployment is simple and the DBMS isn't adding additional configuration and management complexity
- Reporting should be something the developer can do remotely or set-up to be pre-configured and automated, delivering the proper data in the proper formats to the developer, data scientist, or for business and operational analysts
- ACID compliance
- No DBA: Implementing an embeddable database should relieve the developer of the need for database administrator supervision.

CONCLUSION

The world is rapidly filling up with "things." Some industry watchers say there are already more Internet-connected devices on the planet than there are people. And whether it's a refrigerator scanning bar codes to compile a weekly grocery list, a fitness tracker sending heart rate information to a user's smart phone, or industrial machines ordering their own maintenance checks, they're all producing data that must be managed and analyzed *close to the point of action*.

Given the essential requirements for data management and analysis emerging at the edge of the network, it's fair to characterize the embeddable database as the enabling technology of IoT and Edge application development. The question isn't whether developers should employ this technology, but when. At the end of the day, the three Vs of Volume, Variety, and Velocity have to generate a fourth V: Value.

Find out more:
https://www.actian.com/solutions/by-technology/

actian™
activate your data

John K. Waters is a freelance journalist and author based in Silicon Valley. He is editor-at-large for Application Development Trends, and a contributing editor to Campus Technology, T.H.E. Journal, and Law Technology News. He covers a wide range of topics in information technology and is the author of more than a dozen books. He writes news and features about software development trends, application and network security, infrastructure and database technologies, cloud computing, open source software, standards and governance, virtualization, and the people working in, and issues affecting, the information technology business.