ITTIA DB vs. SQLite

Key Critical Differences

ITTIA

WHITEPAPER MARCH ©2022

TABLE OF CONTENTS

INTRODUCTION

ITTIA DB and SQLite are lightweight databases considered for data management on IoT devices. Both are embeddable for high performance and offer benefits to applications using time series data. Developers looking for robust data management options with wide integration capabilities can look towards SQLite or ITTIA DB for solutions to reduce database administration costs, while working within memory constraints and processing capabilities of edge devices.

Solutions must embed silently into applications to allow developers to save time and costs by focusing on their application needs. Data processing solutions must be robust and reliable to automate device data management in real time. ITTIA DB and SQLite offer wide integration support with dynamic APIs for C/C++, Python and Web services. This paper examines the solutions of ITTIA DB versus SQLite.

PERFORMANCE, RELIABILITY, & SCALABILITY FOR THE EDGE

ITTIA DB and SQLite both offer high performance embedded databases with advantages such as: ease of use, flexibility, and data independence. Selecting an available off-the-shelf database is a great alternative that allows developers to focus on application logic and leave data management to a dedicated software library. Embedded and IoT devices have gained significant storage capability in recent years, making new software development options possible.

IoT data management at the edge requires robust processing on a limited memory footprint, without a database administrator, yet features powerful performance. ITTIA DB provides a real time data management platform that offers processing at the edge that allows for storing only critical information and acting on just relevant data, rightfully deemed, stream processing. SQLite's open-sourced code base does not support real time streaming capabilities and developers must create their own strategies for safely grouping, aggregating, and filtering large amounts of data at the edge.

ITTIA DB

ITTIA DB offers a device embeddable data management solution with a footprint as low as 50KB, offering integration with a wide variety of MPUs as well as MCUs. Being an embedded database for the edge, ITTIA DB offers varying levels of security including encryption, replication, and a security agent or protocol called DB SEAL to protect against malicious queries. Planned support for high availability (HA) in ITTIA DB maximizes the protection and availability of data. HA includes peer-to-peer replication, table snapshots, and distributed transactions. The portable file format used by ITTIA DB is highly maintainable, and the stable communication protocols offer easy interoperability with other devices and back-end systems. ITTIA DB also offers development features including DB Console to quickly prototype new schemas and queries before development. This robust and friendly data cockpit allows developers to access data safely and remotely from a web browser.

SQLITE

SQLite offers an embeddable solution with a footprint as low as 300KB. SQLite does not offer encryption and provides no guarantees of protection against malicious queries, putting the burden on developers to protect against threats to their data. SQLite requires developers to implement their own solution to work around timestamped values, like converting UNIX timestamps into integer values. SQLite relies on a network of developers for maintaining its open-source code base and is further supported by a community of software engineers with very limited accountability. Developers working with SQLite may benefit from the open-sourced community, with limited assurance.

DATA AND TESTING METHODOLOGY

Both benchmarks have been written in C/C++ and optimized for each product with equivalent schemas.

DATA MODEL

Value	Type
Sensor Reading (x)	Float
Timestamp (x)	Long Integer
Sensor ID (x)	Integer

Table 1: Data Model for Benchmarking

The dataset consists of a day's worth of data collected from multiple sensors at 1-second intervals, making a total of 4.32 million entries. Therefore, each sensor samples 86,400 values throughout the day, making 3,600 rows per hour. For generalization purposes, each sensor is indicated by an ID but can be represented by any device sensor, including humidity, temperature, pressure, infrared, etc. To further simulate a real-world scenario, the single-threaded performance benchmark uses a total number of 25 sensors, of which 20 are randomly chosen to ingest and query data from. However, the multithreaded benchmark utilizes a total of 62 sensors with 50 sensors chosen randomly to ingest and query data from.

TESTING ENVIORNMENT

These benchmarks are performed for embedded developers on both an Intel NUC mini PC and an NXP i.MX8 M Mini device with the following characteristics:

Intel NUC	NXP i.MX8 M Mini
Operating System: Linux (Debian 11)	Operating System: Linux (Yocto)
Architecture: x86-64	Architecture: aarch64
CPU: Intel i3-7100U (2.40 GHz)	CPU: Arm Cortex-A53 (1.8 GHz)
RAM: 4GB (DDR4 – 2133 Mbps)	RAM: 2GB (LPDDR4)
Storage Media: M.2 SSD (PCI-E 4.0)	Storage Media: Class 10 SD/MMC

Table 2: Benchmark Target Environment Characteristics

	ITTIA DB	SQLite
Version	8.1.0	3.34.1
Locking Mode	File-locking	File-locking
Isolation Level	Read Uncommitted	Serializable
Access Method	C/C++ API	C/C++ API
Database Type	Time Series	Relational

Table 3: Product Configuration Metrics

PERFORMANCE RESULTS

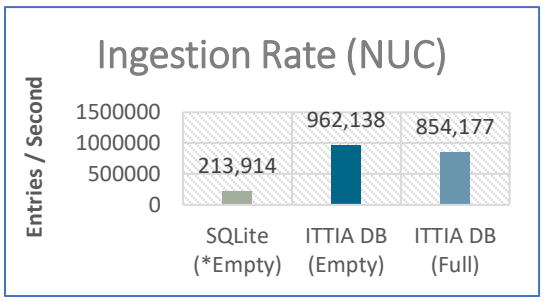
Performance metrics focus on 3 areas:

- Ingestion Throughput - *the number of data entries inserted per second*
- Query Throughput – *the number of queries completed per second*
- Export Time – *the amount of time to export a full dataset*

SINGLE THREADED

Insertion

The ingestion of a full dataset is wrapped in a single transaction for optimized performance. At each sample interval, every sensor is chosen randomly to ingest data from. Both ITTIA DB and SQLite use bindings for batching rows into a single transaction. Throughput is measured by the number of data entries inserted per second, with ITTIA DB measuring 962,138 entries per second from an empty database and 854,177 entries per second from a full database (rollover ingestion rate) on Intel NUC. SQLite measures an ingestion throughput of 213,914 entries per second. SQLite does not offer an option for rollover ingestion. Thus, we did not generate a comparison for that performance.



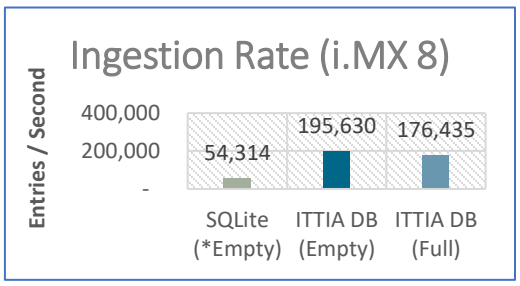
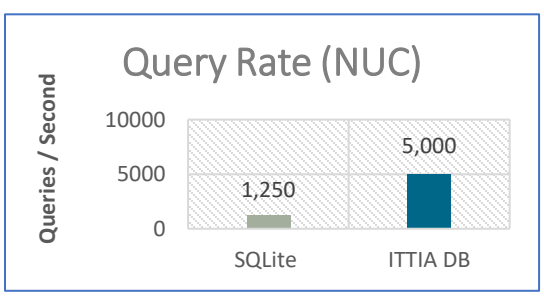


Figure 1. Ingestion Throughput of ITTIA DB *(with empty and full database variants)* alongside SQLite

Querying

After ingesting a full dataset, the metrics for query performance can be collected. To compensate for discrepancies from background tasks, each query is rerun 5 times and the average runtime is reported. Each query selects all the data for some sensor at some hour of the day, both chosen randomly. To optimize query performance, a primary key is used to index the sensor id. SQLite can process 1,250 queries per second on Intel NUC, while ITTIA DB can perform up to 5,000 queries per second, with each query processing 3,600 data entries.

```
SELECT id, timestamp, timestamp_value
FROM table
WHERE id = 15
AND timestamp >= '2020-12-20T03:00:00Z'
AND timestamp < '2020-12-20T04:00:00Z';
```



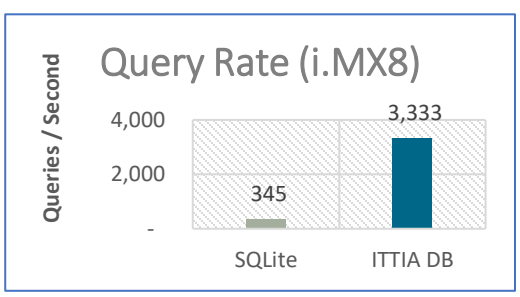


Figure 2. Query Throughput of ITTIA DB and SQLite

Exporting

Given the small memory footprint of many MCUs, a full export of data is often required and within a network of IoT edge devices this can occur frequently. To optimize for export performance, a separate index on the timestamp is used to export an ordered dataset. In this scenario, ITTIA DB shows a 10X performance improvement vs SQLite. Export time for ITTIA DB is recorded at 45 milliseconds compared with 438 milliseconds when using SQLite on Intel NUC. An example of the query used to export the dataset is shown below alongside the amount of time for both products to export a full database.

```
SELECT id, timestamp, timestamp_value
FROM table
WHERE timestamp >= '2020-12-20T00:00:00Z'
AND timestamp < '2020-12-21T00:00:00Z'
ORDER BY timestamp, id;
```

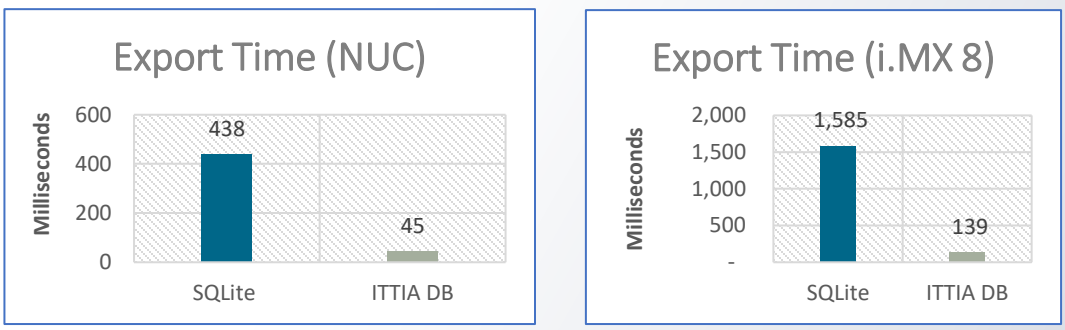


Figure 3. Full Export Time of ITTIA DB and SQLite

MULTI THREADED

ITTIA DB is configured with multithreading support by default. Just open a connection from each thread and begin managing your data. However, with two threads competing for resources, their workloads could become unsustainable. Shown below is a skyline graph representing all the possible workloads that ITTIA DB and SQLite can achieve from two threads, one querying data and one ingesting data. With no ingestion taking place, ITTIA DB can complete 13,299 queries per second on Intel NUC. By increasing the ingestion rate, we see the query throughput fall to 0, with a single thread capable of ingesting up to 9,064 timestamps across 50 series every second.

A green point is used to represent a workload that requires high query throughput at the expense of a higher ingestion rate, such as a data monitoring application. The red point on Figure 4 represents an ingestion-heavy workload, where one sensor is sampling data at a high throughput of 6,200 timestamps per second while another thread can safely query that data at a rate of 1,600 queries per second. Each point within the blue skyline represents a workload achievable through ITTIA DB's engine. The grey data point depicted in Figure 4 represents a workload that is not achievable through SQLite's storage engine, and still well within the limits of ITTIA DB. This performance gap provides the opportunities to save costs and reduce hardware limitations or save extra processing time by using ITTIA DB. Workloads following the purple mark in Figure 4 will find the need to increase hardware resources or limit throughput to achieve their workload. Whereas the black data point in Figure 4 represents a balanced workload of 3,114 queries per second alongside an ingestion throughput of 3,551 timestamps per second.

SQLite is configured with multithreading support by default, but extra steps must be taken from the developer to ensure their database can support multiple threads. SQLite supports various levels of transactions, and it is up to the developer to ensure their proper usage. To avoid locks, each thread starts a transaction immediately upon scheduling. Seen below, SQLite can read up to 230 queries per second. The performance of SQLite suffers from the overhead of multiple threads. A maximum ingestion rate of 147 timestamps per second across 50 series makes a total of 7,350 data entries per second.

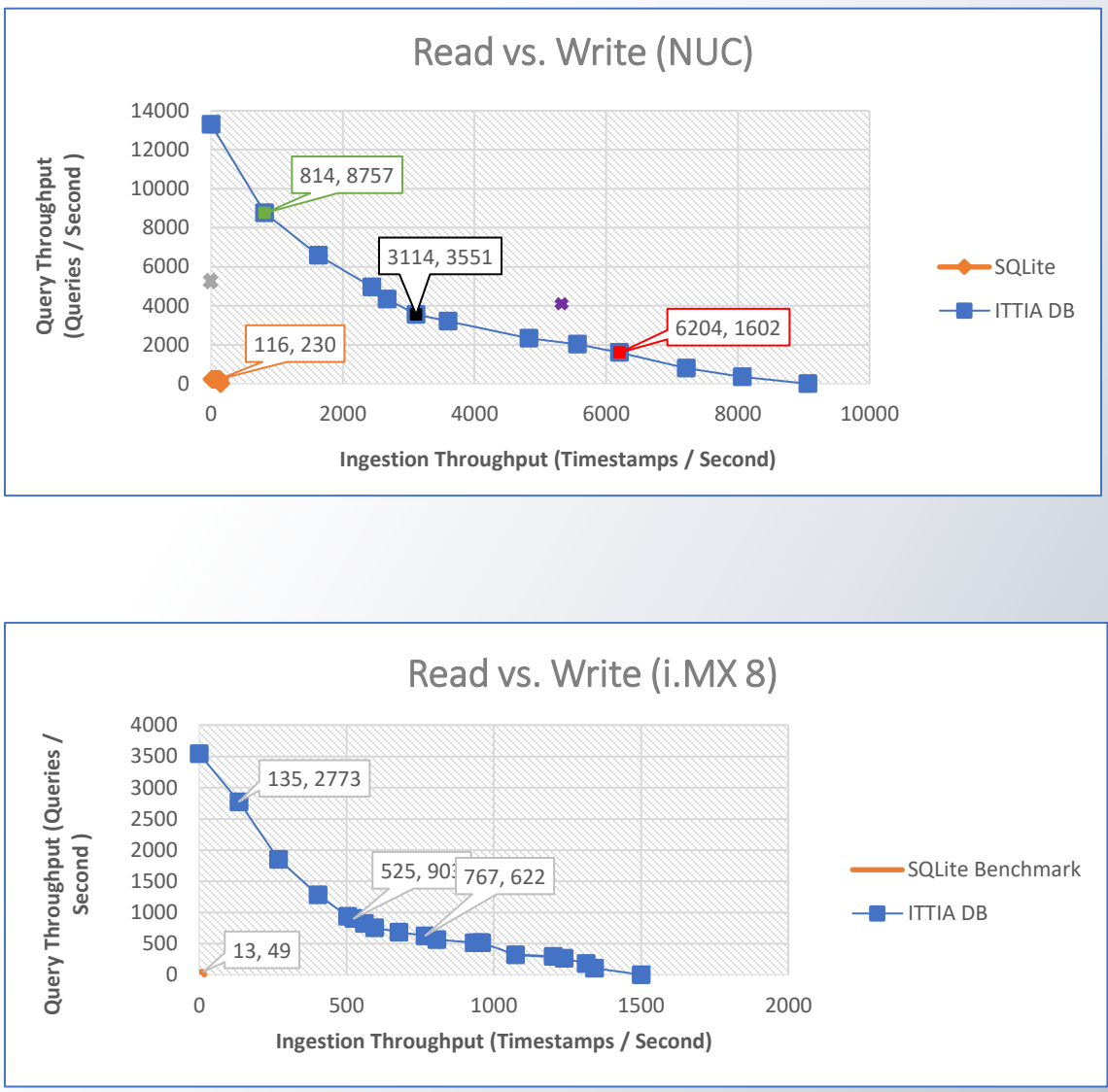


Figure 4: Multithreaded Performance of SQLite and ITTIA DB

FEATURE COMPARISON SUMMARY

Capability	ITTIA DB	SQLite
Low Footprint	50K	300K
Code Base	Proprietary	Open Source
Multi-Thread	Optimized	Basic
API	Proprietary	Standard
Real-Time Streaming	X	-
Secure	X	-
Web browser access	X	-
Simulation GUI Tools	X	-

Table 4: ITTIA DB versus SQLite Feature Comparison Summary

PERFORMANCE COMPARISON SUMMARY

ITTIA DB vs SQLite	
Insertion	4.5X
Querying	4X
Exporting	10X
Multi-Threaded	Exponentially Faster

Table 5: ITTIA DB versus SQLite Performance Summary

CONCURRENCY AND STANDARDS

A solid foundation for multi-threaded and multi-process concurrency is an important part of database performance. SQLite is intended for low-concurrency settings with a small number of users and processes.

Storage-level locking, for example, disables all database access while concurrent modifications are being made. As a result, a more granular locking technique often improves performance for a mix of select, insert, update, and delete actions. ITTIA DB provides both row-level and storage-level locking options, as well as MVCC, whereas SQLite supports only storage-level locking. ITTIA DB's scalable locking options provide you the flexibility you need to get the performance you need in real-world applications.

Because embedded devices require scalability, developers frequently use the same database files across different processes and apps. The inability of SQLite to scale for a large number of processes causes a significant performance barrier. The powerful shared cache design of ITTIA DB, on the other hand, keeps the overhead of each database connection to a minimum.

Many database parameters can be tweaked to modify metrics like throughput and latency to achieve the greatest possible performance for a specific application on a given platform. ITTIA DB provides tuning parameters not found in SQLite, allowing developers to accomplish their goals. ITTIA also provides the engineering skills needed to fine-tune the database to meet the needs of certain applications.

SQL, the structured query language, is used in database systems to store and retrieve data in a flexible manner. The SQL standard offers a basic set of characteristics that apply to embedded systems and devices while being created primarily for back-end server databases.

Instead of following the SQL standard, SQLite provides a permissive SQL dialect that is similar to that of several other database solutions. While this makes it simple to apply SQL queries created for another product, slight changes in behavior can lead to unintended consequences that go unnoticed. ITTIA DB adheres to the SQL standard, which ensures that SQL queries have a clear, understandable syntax, predictable type conversion behavior, and correct query results.

Standards and compliance are vital when building mission-critical systems, not just to maintain the database after deployment, but also to prepare you for certifying your product. This is on top of keeping up with updates, implementing fixes, and ensuring that everything related to the database is under your control and technical expertise.

HIGH AVAILABILITY WITH DATABASE REPLICATION

Devices must process data generated by and dispersed across a large number of nodes. Most protocols are suited for conveying a node's current state, and data replication techniques will greatly benefit querying data recorded over a lengthy period of time. To reduce network overhead, nodes must keep track of which information has previously been sent. Conflicts arise when numerous nodes edit the same shared information, jeopardizing the accuracy of recorded data. When mission-critical data is lost, the entire system is jeopardized.

Application developers are solely responsible for communication protocols, tracking database modifications, and keeping metadata to coordinate each node when using SQLite to distribute data.

ITTIA DB's support for high availability (HA) improves data security and availability. Peer-to-peer replication, table snapshots, and distributed transactions are among ITTIA DB's high-availability features. Embedded applications, for example, use synchronous replication to propagate changes in real-time, automatically sending updates to other devices or backup storage media. Ad hoc replication allows applications to communicate data asynchronously. ITTIA DB protects the application's work from component failure after replication is configured.

ITTIA DB's high availability support ensures that data and databases are always accessible and available, regardless of whether there is a power outage, a catastrophic network failure, or a maintenance issue. Data consistency, data redundancy, fallback, and failure detection are all supported by ITTIA DB SQL for high availability.

RELIABLE DATABASE EXPERTS YOU CAN TRUST

The database's technical assistance is far more involved than in other areas of software development and deployment. Database technology is complex, and its effective application necessitates a high level of technical knowledge. Customers will have instances when they require in-depth knowledge and support for a database feature but cannot wait for public opinion to reach a consensus. If a consumer requires a rapid response, it may cost more than you anticipated. Customers of ITTIA receive personalized, one-on-one technical assistance that goes beyond answering technical problems. ITTIA professionals work hard to understand each application's unique requirements and to determine the most efficient path to success.

Customers don't have to wait to uncover potential dangers and correct early design flaws since ITTIA supports an accurate development environment. A flawless development process ensures that the product arrives on time and on budget. ITTIA will continue to support any changes and upgrades customers make to their apps even after the product has been implemented.

ITTIA has always been responsible for developing a close working relationship with its customers during the development, implementation, and completion of their applications.

DATA TYPES

Embedded databases can hold a variety of data types, including integers, text, time, and raw binary data. When data is stored in the database, ITTIA DB columns are strongly typed to impose domain requirements. SQLite employs dynamic temporal typing, allowing a variety of data types to be stored in the same column. Any value can be retrieved as a character string from both databases.

Strong typing is extremely beneficial when dealing with time data. SQLite stores time types as strings, but does not validate the value until it is used in a time function by the application. ITTIA DB contains the specialized date, time, and timestamp types that integrate the two. Before being committed to the database, time strings are checked, and invalid values, such as 99:00:00, are promptly discarded.

ITTIA DB supports the SQL standard EXTRACT keyword and time interval arithmetic for manipulating time values. Individual components of a time value can be accessed with the EXTRACT keyword: year, month, day, hour, minute, or second. Interval arithmetic allows you to easily add or subtract from a time type, as well as discover the difference between two times with any precision you choose. To alter time strings, SQLite uses nonstandard formatting functions and modifier strings.

API, UTILITIES, AND DATABASE COCKPIT

It was built from the bottom up to provide embedded application developers with the most significant database functionalities without the need for complicated installation or administration tools. It was created with the goal of being simple to use and maintain. It is cross-platform, with easy upgrades and attractive APIs.

While ITTIA DB and SQLite both have terminal utilities for running SQL commands, most applications use an API or a full web server console to access the database. Each product has a C-language interface that can be used to run SQL queries and perform maintenance chores like backups. Control transactions, alter the database schema, conduct ISAM operations on table cursors, and disseminate data to other databases are all available with ITTIA DB.

ITTIA DB provides dynamic, static, and forward-only SQL cursors, whereas SQLite can only fetch rows from a SQL result set in one direction. This allows programs to quickly scroll over query results, which is essential for interactive presentations and processing big data sets.

Standard interfaces such as ODBC, JDBC, and popular scripting languages can be used to access ITTIA DB SQL and SQLite databases. For developers working on a specific platform or programming language, these provide a familiar environment. Bindings are also available for a number of scripting languages, including Python, Ruby, and Lua, which are frequently used to augment the core application.

ITTIA DB Console is a contemporary database cockpit interface that allows developers to build, administer, and monitor embedded database and data processing processes. From the convenience of a web browser, software designers may prototype table structures, generate experimental data in real-time, and execute SQL queries on specified target MCUs and MPUs using a simple visual dashboard.

Developers can use ITTIA DB Console to run SQL statements and queries, monitor schema definitions, describe tables and sequences, monitor table structures and content (columns, fields, indexes, etc.), monitor and configure replication settings for both databases and tables, and import and export data in XML and JSON formats.

SAFETY AND SECURITY

Using device data processing and management capabilities to ensure data integrity and dependability is a good idea. Sensors continuously provide a huge volume of data for edge gateway devices to consume in a typical IoT data management scenario. Without relying on central storage, an IoT data management platform shares, stores, and analyzes data. These high-performance devices must not only collect data in real-time but also organize and make the data available. But what are the fundamentals for using SQLite to manage data and share it with other systems in a secure and safe manner?

What are the security characteristics of SQLite? Devices can also publish data to any number of nodes over a range of network configurations thanks to connectivity. As a result, how should SQLite be used to encrypt communication pathways to the device?

CPU, memory, storage, and network connectivity are all fixed resources in embedded systems. Because embedded system security mandates the protection of sensitive data throughout the device's life cycle, secure storage and transmission become vital. As a result, a multi-dimensional security technique is required for device internal data management, data protection risks, attack taxonomy, and system vulnerabilities.

ITTIA DB has encryption and authentication capabilities, as well as a security agent named DB SEAL. While the ITTIA DB Console application allows developers and end-users to monitor database activity, ITTIA DB SEAL automatically separates databases stored on devices, chooses amongst mitigation options, and maintains database contents constantly available.

When data management metrics fall outside of the expected range, DB SEAL's proactive monitoring of the data and database will let the device deliver an alarm, prevent access, or shut down. This is a virtual agent that monitors database responsibilities and metrics in real time and responds when an outage or other security problem occurs.

ITTIA DB also supports encryption, including the Advanced Encryption Standard (AES), to safeguard database communication and data transformation. AES stands for Advanced Encryption Standard, and it is a standard for encrypting electronic data. This algorithm is used to secure many communications.

SCRAM (Salted Challenge Response Authentication Mechanism) is a password-based mutual authentication technique compatible with ITTIA DB that makes an eavesdropping attack known as man-in-the-middle interface harder.

ITTIA DB provides authentication and authorization for client-server and distant replication communications. SCRAM provides user and device authentication to ensure the security of M2M data management. Eavesdropping, unlawful interception, and session high jacking are not prevented by authentication. ITTIA DB provides a TLS solution for this and supports SSL and TLS security features.

Abuse of operating systems with no privilege separation, buffer overflow, and SQL injection are all examples of assaults on software integrated inside the device. Control hijacking via SQL is a sort of attack that diverts the normal control flow of the programs operating on the device, usually resulting in the hacker injecting SQL code. SQL code execution is a method by which attackers feed code to an embedded device, such as web scripts and SQL injections, that is not native to the device's application.

These attacks result in a variety of issues, including integrity violations, which are a regular side effect of new codes on a device's database. This assault could result in changes to configuration settings, data, or even firmware updates that aren't legal. In extreme situations, the effect may even result in the disclosure of sensitive information.

Our security agent is a smart mediator that protects embedded data saved in ITTIA DB database files and provides options for a device to operate when it detects an attack. DB SEAL isolates databases on devices, chooses amongst mitigation options in the event of an attack, and always keeps the database contents accessible. Secure remote client/server communications are also available with ITTIA DB. Secure machine-to-machine (M2M) connectivity is also ensured thanks to built-in encryption and authentication.

Developers who are worried about data confidentiality, integrity, and availability should pay attention to database security. As a result, choosing an embedded database with the necessary security characteristics is crucial for the definition and implementation of secure applications.

In today's cyber hacking world, it is important for companies to proactively address security issues and remain informed about security options. We at ITTIA focus our efforts on embedded systems data management and our main goal is to help our customers by creating a secure database that protects them from rapidly evolving threats, vulnerability, and web services.
In today's age of cyber hacking, it's critical for businesses to handle security issues ahead of time and stay aware of security alternatives. At ITTIA, we concentrate our efforts on embedded systems data management, with the primary goal of assisting our customers by developing a secure database that protects them from quickly evolving threats, vulnerabilities, and web services.

HOW TO AVOID BECOMING A DATABASE COMPANY

SQLite, unlike many other open-source projects, does not allow community contributions. Your key add-ons will not be included in the next supported release if your experts offer new development to SQLite that is suited for your application. Merging your customizations with each new SQLite version is a time-consuming and costly task.

Customers decide the vision for ITTIA DB, and the solution has evolved organically to satisfy the needs of embedded and IoT device application developers.

When you choose SQLite, you are effectively becoming a database specialist or a database firm, however when you choose ITTIA DB, you are relying on database expertise.

CONCLUSION

ITTIA DB and SQLite offer edge data processing solutions with dynamic APIs for real time data processing on IoT device applications. Both solutions offer data independence, but only ITTIA DB offers the ability to leverage SQL for stream processing time series data and remote data access via web browser. Both solutions offer fast data exports, however, ITTIA offers 10X the performance vs SQLite in exporting data and over 4X the ingestion performance from a single thread. While both ITTIA DB and SQLite offer the freedom to build applications across many platforms, ITTIA DB's small memory footprint of 50KB offers a wide range of platforms to embed your application, and leverage SQL for stream processing of time series data.

Although SQLite offers a familiar SQL interface for writing queries and ITTIA DB uses a proprietary API, ITTIA DB provides significant performance, footprint, and security advantages. Sponsorships, consulting, and paid support customers are the main sources of money for the SQLite project. As a result, only those who make such a commitment and pay will receive your development team's full attention. SQLite does not disclose or share a product roadmap, thus developers that choose SQLite are responsible for filling in any gaps. They must locate or create the necessary functionality, test it, document it, and ensure that it is compatible with future releases. A database has suddenly become a part of your development environment. What is the total cost of ownership (TCO) for SQLite embedding?

Although the open-source community provides many SQLite add-ons, there is no guarantee of continued maintenance or product quality. Third-party developers cannot create database features that are incompatible with SQLite's general architecture.

For mission-critical applications, maintaining an SQLite-based system is costly and unreliable. Many businesses begin using SQLite and, after years of significant investment, find themselves in need of database maintenance and assistance from a genuine business. At that point, these businesses must effectively start over and investigate other database solutions, a more robust structure, and SQL statements that comply with industry standards. The decision to include SQLite will shift a large percentage of your development emphasis away from product creation and innovation and into database management and integration.

Because relying on a community is not the greatest practical solution, it is becoming evident that this free product will require significant maintenance, upgrades, and new feature development resources. As a result, you'll need to hire, train, and retain your own in-house database experts to help with the development, testing, and maintenance of new database features. As a result, you must factor in the additional development and maintenance costs and responsibilities.

ITTIA DB is a cost-effective option that allows manufacturers to concentrate on application business logic while data management is handled by a dedicated software library and database professionals.

Information in this document is provided solely to enable system and software implementers to use ITTIA products. No express or implied copyright license is granted hereunder to design or implement any database management system software based on the information in this document. ITTIA reserves the right to make changes without further notice to any products described herein. ITTIA makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does ITTIA assume any liability arising out of the application of or use of any product, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Statistics and parameters provided in ITTIA white papers and data sheets can and do vary in different applications and actual performance may vary over time. All operating parameters must be validated for each customer application by customer's technical experts. ITTIA and the ITTIA logo are trademarks or registered trademarks of ITTIA L.L.C. in the U.S. and other countries. All other product or service names are the property of their respective owners.

Copyright (c) 2022 ITTIA L.L.C. All rights Reserved. References in this document to ITTIA products and services do not imply that ITTIA intends to make them available in every country.