MCKNIGHT
CONSULTING GROUP

Edge Computing Evolved: Introducing the Zero-DBA, Zero-ETL Embedded Database

Presented by: William McKnight

President, McKnight Consulting Group

3 X Inc 5000

/in/wmcknight

www.mcknightcg.com
(214) 514-1444

TOP VOICE ★ 2024 ★ thinkers 360 OVERALL

I'VE BEEN FEATURED IN THE 2024 dataIQ™ 100 USA THE MOST INFLUENTIAL PEOPLE IN DATA

McKnight Consulting Group Partial Technology Implementation Expertise

Big/Analytic/Vector/Mixed Data Management

- databricks
- Pinecone
- Google BigQuery
- snowflake
- Azure Synapse
- MySQL HeatWave
- teradata.
- amazon REDSHIFT
- CLOUDERA
- SAP Datasphere
- Exasol
- VERTICA
- Starburst
- ACTIAN activate your data
- Azure HDInsight
- SingleStore
- Apache Impala
- amazon EMR
- Yellowbrick
- Cloud Dataproc
- ORACLE AUTONOMOUS DATABASE
- ORACLE

Data Movement and APIs

- Airbyte
- Informatica
- Qlik Replicate®
- Fivetran
- SYNADIA
- CONFLUENT
- apigee
- talend
- MATILLION
- Stitch
- Kong
- Astera

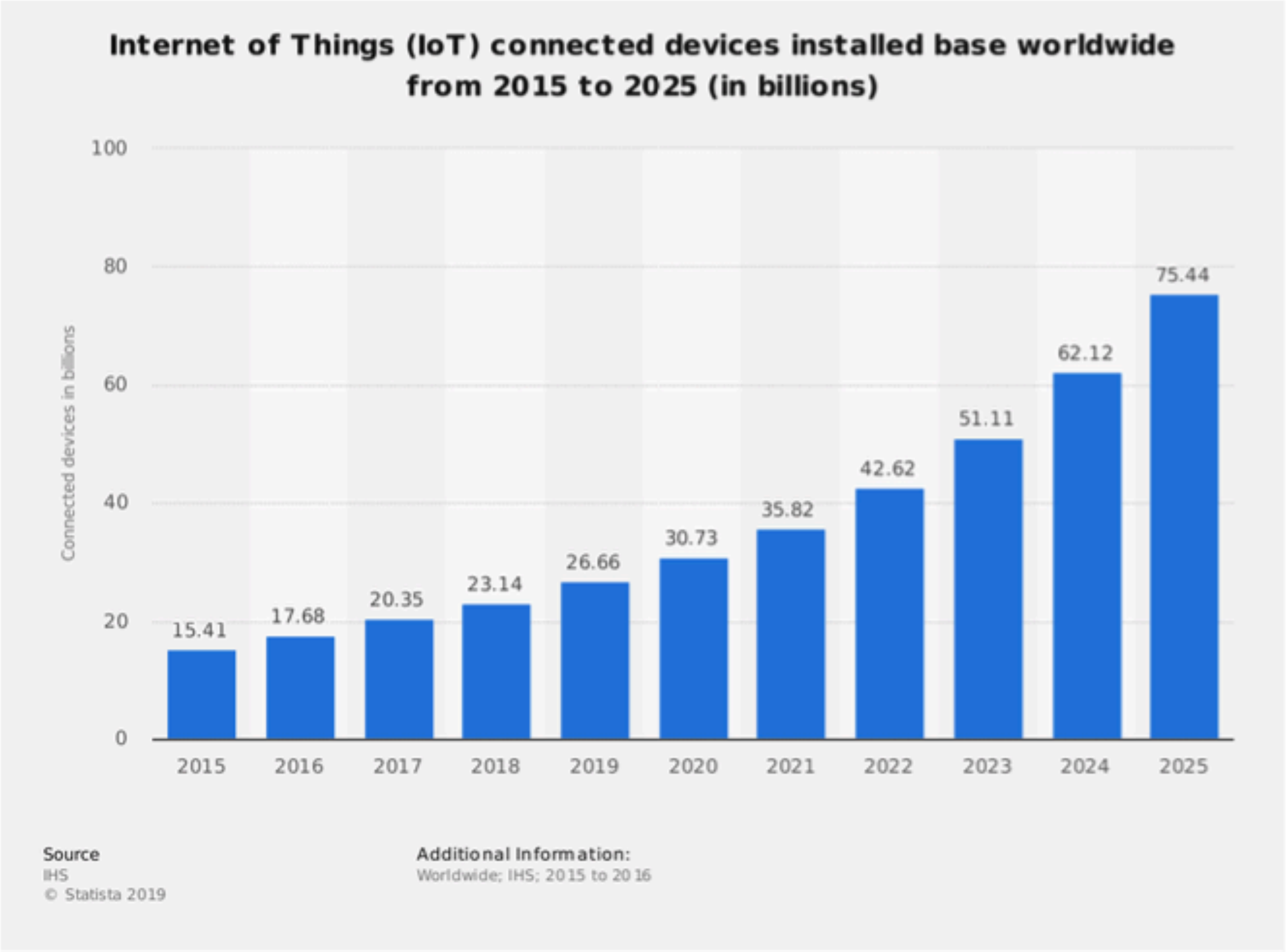
Data Management

- MONTE CARLO
- Anomalo
- SNOWPLOW
- SAP Datasphere
- alteryx DESIGNER CLOUD
- Profisee
- splunk>
- NGINX Part of F5
- Collibra
- IMMUTA
- elasticsearch
- Azure MLOps

Operational/Transactional Data Management

- DIAMANTI
- Cockroach
- Couchbase
- DataStax
- SQLite
- MongoDB.
- PostgreSQL
- yugabyteDB
- neo4j
- ORACLE
- Google Cloud Spanner
- redis
- Azure Cosmos DB
- Microsoft SQL Server
- Amazon Aurora

MCKNIGHT CONSULTING GROUP

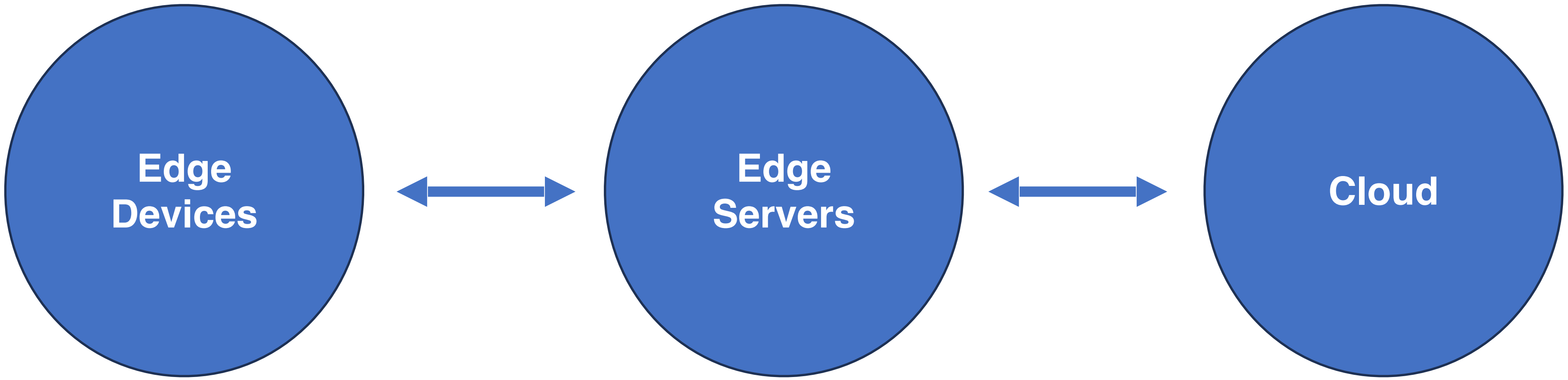

Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)
Connected devices in billions
100
80
60
40
20
0
15.41
17.68
20.35
23.14
26.66
30.73
35.82
42.62
51.11
62.12
75.44
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
Source
IHS
© Statista 2019
Additional Information:
Worldwide; IHS; 2015 to 2016

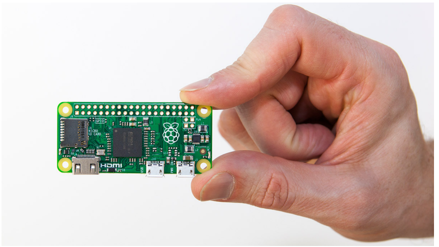
What is the Edge

- The Edge is where “Things” are
- Where highly available processors enable real-time analytics for applications that can't wait a long time for decisions

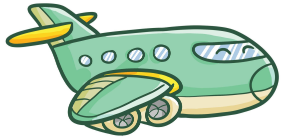
Edge Computing

Distributed computing that processes and stores data at the edge of the network, near the data sources, to reduce latency and improve real-time decision-making.





Use Cases



- **Airlines**: Power in-flight seatback apps without relying on internet connectivity.
- **Retail**: Enable digital kiosks for customers to look up items or make purchases.
- **Restaurants**: Support digital tabletop ordering systems.
- **Event Venues**: Power digital ticket turnstiles to avoid internet slowdowns.
- **Mobile Airline App**
  - Full features online: check in, retrieve your boarding pass, and check your flight status and boarding gate
  - Viewing your boarding pass and upcoming flight details when offline
  - Push gate changes to mobile
- **Socially connected mobile game**
  - Latest leaderboard, position in the game, and chats with friends available when online
  - Continue playing when offline, updating game position and recording new achievements locally
  - When back online, push latest achievements and game status to the backend and synchronize leadership board in both directions
- **Aggregated IoT Sensor Data**
  - Copy sensor data to your backend database when online
  - Aggregate sensor data on IoT gateways, store locally, and sync to the backend
  - In the backend database, aggregate data from thousands of IoT devices to see the big picture

Edge Computing Advantages

- **Real-Time Insights**
  Reduce latency by bringing data processing closer to the source, enabling faster decision-making.
- **Data Protection**
  Minimize risk by keeping sensitive data localized, reducing exposure to potential threats.
- **Optimized Bandwidth**
  Reduce data transmission costs and congestion by processing data locally, minimizing the amount of data sent to the cloud.
- **Resilient Operations**
  Maintain functionality and productivity without reliance on constant network connectivity, ensuring operations remain stable.

Why Embed a Database

- **No DBA; Administration free**: This reason highlights the benefit of reduced administrative overhead, allowing developers to focus on building applications rather than managing databases.
- **Speed up development**: Embedding a database can accelerate development by providing a pre-built, tested, and reliable data storage solution, reducing the time spent on implementing data management.
- **Performance**: A well-optimized embedded database can provide high performance, enabling fast data access and manipulation, which is critical for applications that require low latency and high throughput.
- **Native Integration**: This reason emphasizes the benefit of seamless integration with the application, allowing for a more streamlined development process and better overall performance.
- **Simplified Operations**: Embedded databases often simplify operations by providing a self-contained solution that requires minimal configuration and maintenance, reducing the complexity of managing data storage.
- **Security built in**: This reason highlights the importance of data security, as embedded databases can provide built-in security features, such as encryption and access controls, to protect sensitive data.

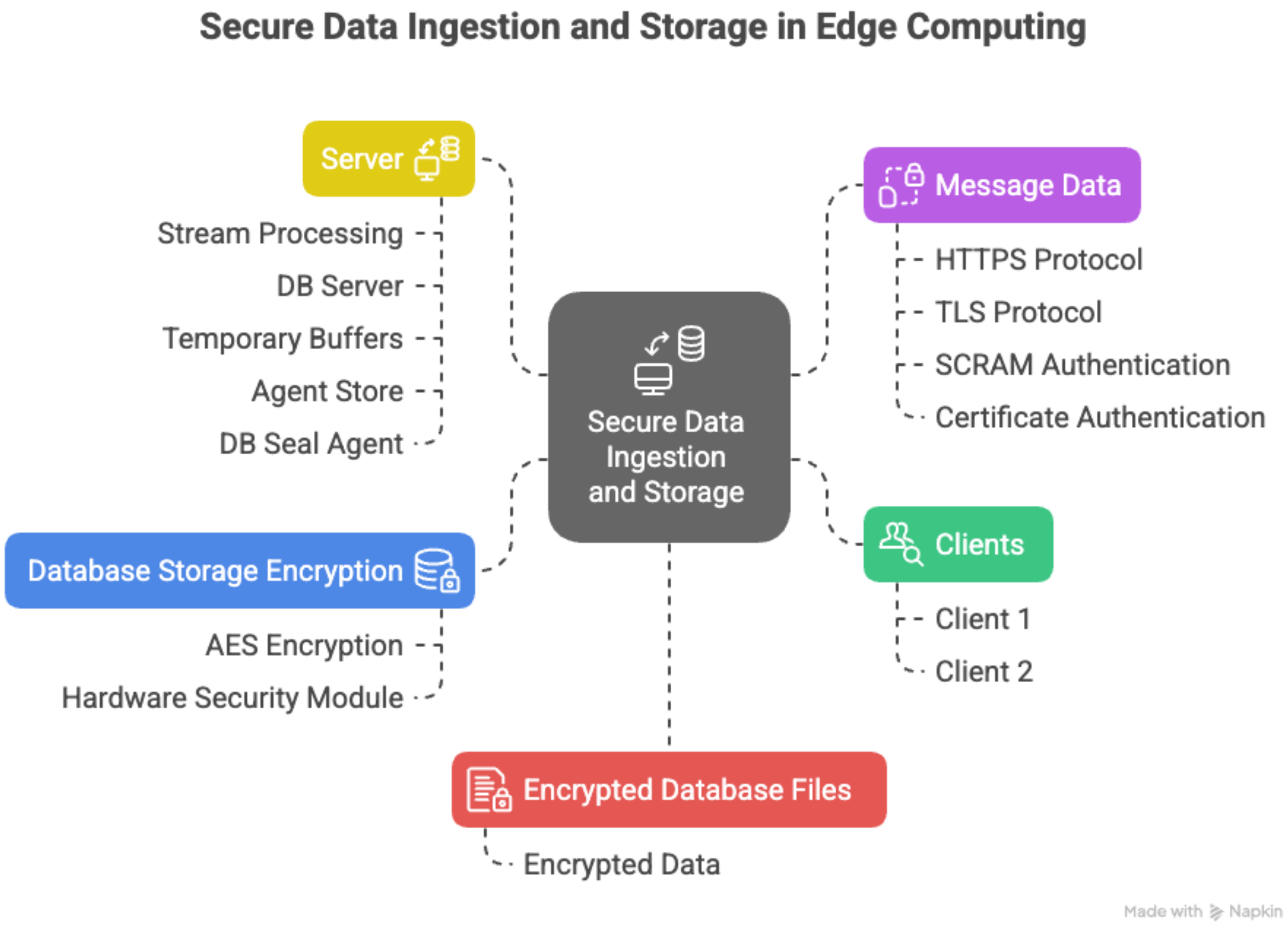
Embedded Databases at the Edge

- **Simplified Deployment**: Embedded databases are self-contained, making it easier to deploy applications.
- **Tight Integration**: Embedded databases are designed to work seamlessly with the application, providing optimized performance.
- **Low Latency**: Embedded databases can provide faster data access since the data is stored locally.
- **Cost-Effective**: Embedded databases can be more cost-effective than traditional database solutions.

Mobile Databases are the Original Database Except Limited/No…

- Sharding and Replication
- Storage Engine Configuration
- Database Authentication
- Encryption/SSL
- Change streams
- Server-side JavaScript Execution
- Transactions

Embedded Database for Real-Time Analytics



Bidirectional Data Synchronization

- **Two-way data exchange**: Data is synced in both directions, ensuring consistency across all systems.
- **Real-time data consistency**: Changes made in one system are reflected in the other system, and vice versa.
- **Conflict resolution**: Bidirectional synchronization can handle conflicts that arise when data is changed in both systems.
- **Data integrity**: Ensures data accuracy and integrity by syncing changes in both directions.
- **Flexibility**: Allows for changes to be made in either system, making it suitable for distributed and collaborative environments.
- **Supports offline-enabled applications**: Enables data to be updated locally and synced when connected to the central system.

Edge Database Challenges

- Resource Constraints
  - CPU
  - Memory
  - Storage
  - Power
- Connectivity
  - Network Connectivity
  - Conflict Resolution
  - Data Synchronization

MCKNIGHT
CONSULTING GROUP

Don’t use Flat Files for embedded data

- Lack of data portability
- No single API portability across programming languages like NoSQL and SQL
- Data integrity problems: missing data, inconsistencies, corruption, etc.
- Indexing, filtering, SQL, metadata, synchronization, support have to be built
- Security needs to be added
- Need to add reporting, auditing, file management, etc.

Architecting for the Edge

Storing data at the edge

- The overall purpose of collecting data on the edge has shifted from purely device control and monitoring to improving various service capabilities through real-time analysis
- Row data timestamped coming from devices to be stored in central database
- Important when connectivity is limited
- Storage Mechanisms
  - Store and forward
  - Twin

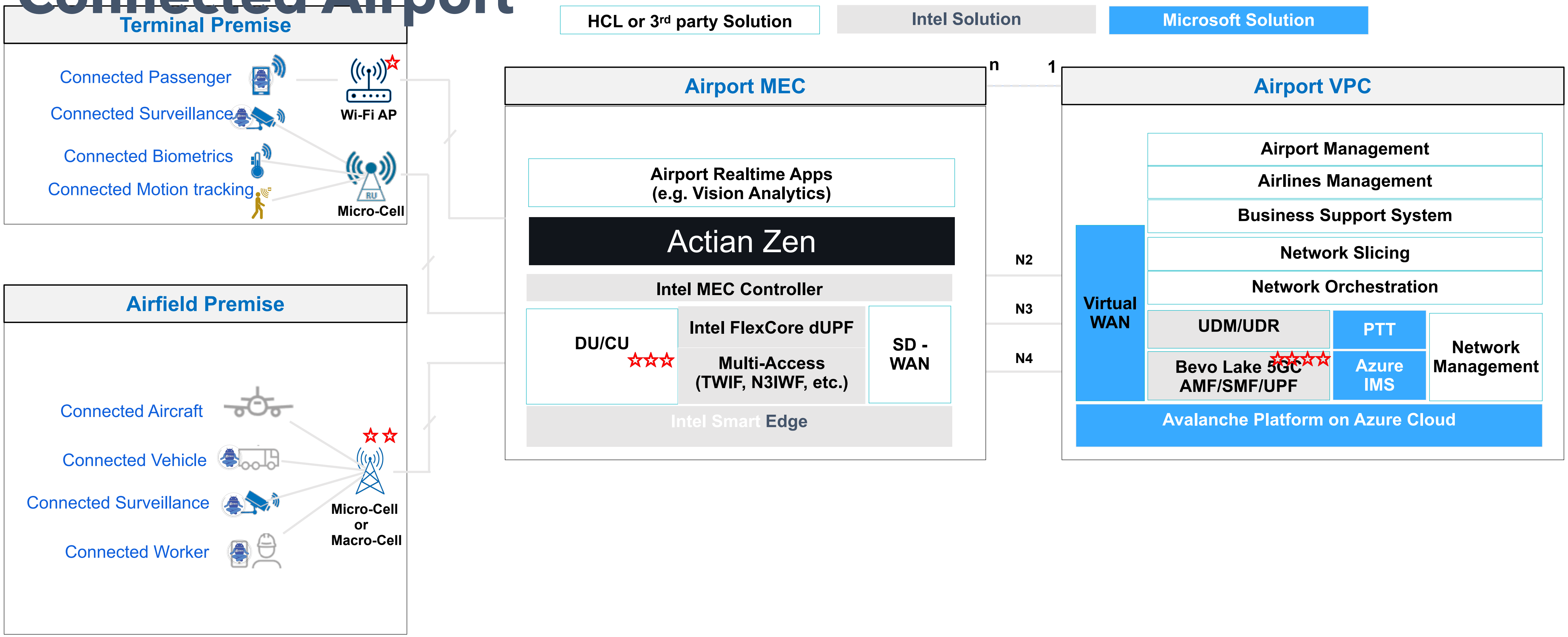
Time-Series Data at the Edge

- Oil & gas in remote location
- Agriculture & mapping
- Black box equipment – i.e., airplanes
- Self-Driving Cars
- Trading Algorithms
- Smart Homes
- Transportation Networks
- Law Enforcement

IoT Device Differences from Traditional Data Requirements

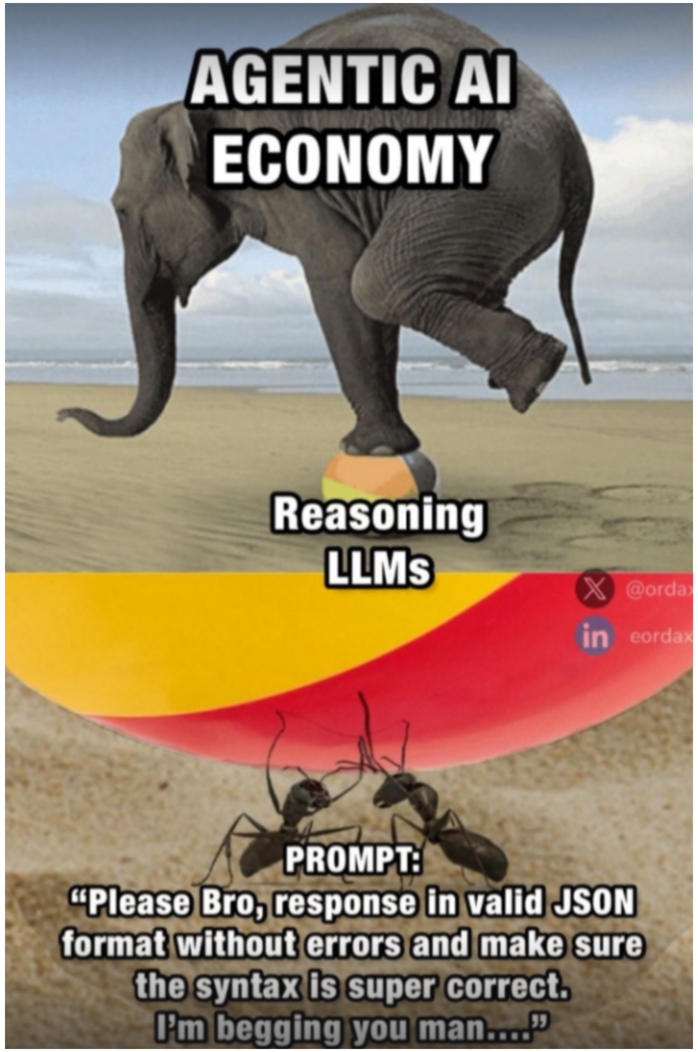
- The high volume of data generated by devices
- New data sources have emerged that generate orders of more data.
- The speed at which data are collected from various streaming sources, sensors, or generated by algorithms to then pass through IoT edge devices and gateways only compounds the demands on a system.
- Modern transponders' frequencies are higher than before, sensors have greater precision
- Location services on mobile devices are routinely used by consumers for everyday chores
- Edge devices may not have enough resources — memory, persistent storage and CPU power - to fully analyze the data on their own, at least not while fulfilling the devices' main purpose
- Edge nodes' physical connectivity is often unpredictable
- The database management system should be able to adjust its data replication patterns automatically based on various application-defined criteria

High Level Reference Architecture Framework for 5G Connected Airport

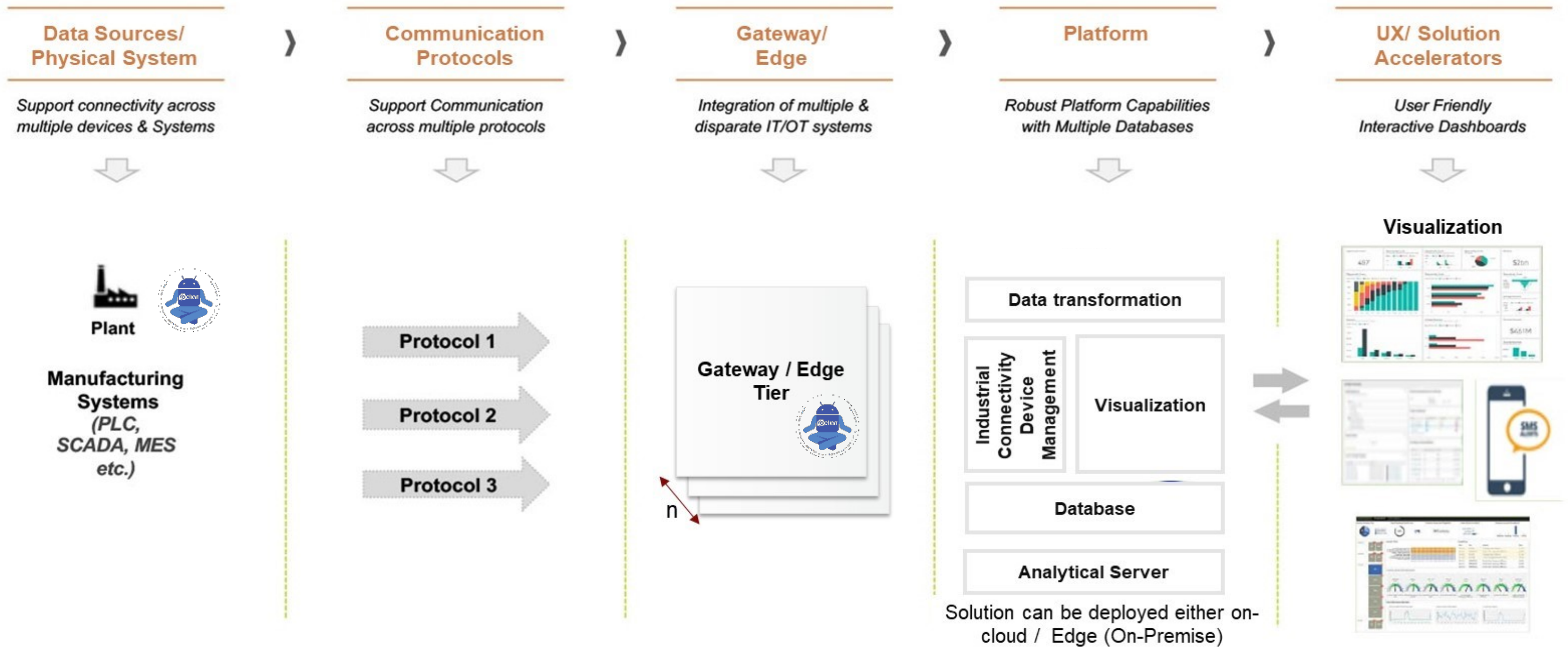


The edge database supports the connected airport architecture in several ways

- Real-time Data Processing
- Edge Computing
- Integration with Intel MEC
- Support for Airport Real-time Apps
- Data Management
- Scalability and Flexibility

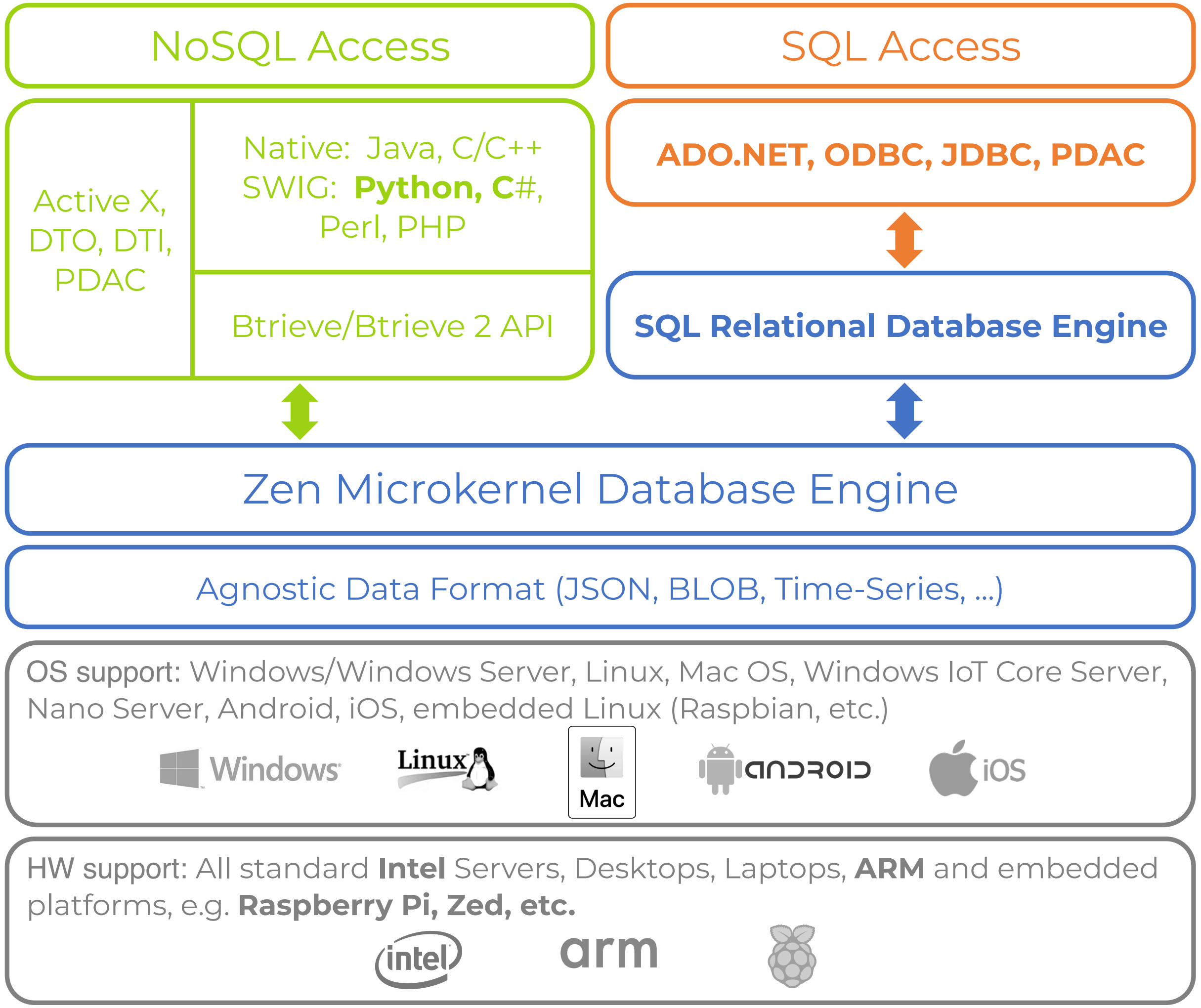


Industrial IOT Architecture



Databases purpose-built for Edge distributed data

- **Flexible development and deployment options**
- NoSQL performance and SQL access
- Wide range of OS platforms, file systems, and SDKs
- **Reliability designed for ZERO management user environments delivers low support costs**
- Self tuning, auto defragmentation, auto reconnect
- Easy upgrades and backward compatibility – new version migration won't break existing applications
- Customizable installation
- Footprint as small as 5MB (embedded engine library) for mobile or IoT devices to enterprise Server for business operations



Security

- Existing internet SSL/TLS technologies do a good job of protecting communications channels between edge nodes and back-end servers
- In addition to secure communications and channel authentication, the integrity of data must be ensured by the database management system, an important component of which is data encryption
- The best database vendors enable use of IoT containers incorporating Federal Information Processing Standard (FIPS) for 140-3, a U.S. government standard that defines cryptographic module security requirements
- Device Security
- Secure Coding

Selecting Your Zero-DBA, Zero-ETL Embedded Database

Key Market Need Categories

1. Vector Processing for Semantic Search and RAG
2. Inference on the Edge
3. Multi-model Support
4. Stream Data Processing and Change Data Capture (CDC)
5. Machine Learning Capabilities
6. Hybrid Transactional/Analytical Processing (HTAP)

1. Vector Processing for Semantic Search and RAG

Purpose: Enable semantic search and Retrieval-Augmented Generation (RAG) by processing and storing vector embeddings for Large Language Model (LLM) applications.

Key Features:

- Support for high dimensionality vectors
- Similarity search techniques (e.g., HNSW, IVF, Exact search)
- Distance/similarity functions (e.g., Euclidean, Cosine, Dot product)
- Efficient metadata filtering and vector compression/quantization

Common Gaps:

- Lack of native vector-specific similarity functions and ANN indexing techniques
- Limited built-in vector compression and tooling for LLM/RAG integration
- Manual methods required for storing and parsing vectors
- Multimodal Support varies by database, often relying on generic BLOB or TEXT fields with size limits.

2. Inference on the Edge

Purpose: Enable AI/ML inference directly on edge devices for offline or ultra-low latency operations.

Key Features:

- Low latency and high throughput
- Lightweight for constrained devices (e.g., ARM, Raspberry Pi)
- Interoperability with popular ML runtimes (e.g., TensorFlow Lite, ONNX, PyTorch Mobile)
- Inference orchestration and caching

Strengths:

- Fast, low-latency data access without network overhead
- Efficient for rapid ingestion of sensor data and quick retrieval of inference outputs
- Suitable for isolated or intermittent-connectivity environments

Common Gaps:

- Inability to run ML models internally within the database engine
- Lack of built-in changefeeds or reactive triggers to invoke inference logic automatically
- Inference Caching: Edge databases can cache inference results, reducing the need to re-run models and benefiting power-constrained devices.

3. Multi-model Support

Purpose: Evaluate a single database's ability to store and manage various data models (relational, document, graph, time-series).

Key Features:

- Native support for different data models
- Cross-model queries using a single language
- Flexibility in schema design (schema-less vs. schema-driven)

Strengths: Strong native support for relational/SQL data and key-value storage.

Common Gaps:

- Limited native JSON functions and graph traversal capabilities
- Lack of advanced SQL features (CTEs, window functions, recursive queries)
- Limited native full-text search and array types

Use Case: IoT and Recommendation scenarios require managing diverse data types (relational, JSON, graph, key-value).

4. Stream Data Processing and Change Data Capture (CDC)

Purpose: Assess databases' ability to power data synchronization, event streaming, and microservice messaging.

Key Features:

- Native CDC methods (replication, trigger-based, log scraping)
- Support for various CDC event types (insert, update, delete, schema drift)
- Native stream processing capabilities (aggregations, window functions)
- Integration with Kafka

Strengths: Trigger-based CDC and replication support.

Common Gaps:

- Lack of native stream processing engine and Kafka integration
- Limited support for complex event processing and schema drift handling

Use Case: Real-time alerting involves capturing live changes, applying business logic, and triggering alerts via webhooks or Kafka.

5. Machine Learning Capabilities

Purpose: Assess a database's ability to support outlier detection and trend prediction using embedded or external ML models.

Key Features:

- Model support (locally stored, SQL-native, extensibility via extensions)
- Data federation capabilities
- Integrations with popular ML tools (Pandas, Jupyter, Airflow)

Strengths: Excellent for local ML inference caching and data management.

Common Gaps:

- Limited native support for running or training ML models internally
- Few built-in tools for ML workflow automation

Use Case: Building a user behavior ML app involves setting up a feature store, performing real-time inference, and training/retraining models, often requiring external Python scripts.

6. Hybrid Transactional/Analytical Processing (HTAP)

Purpose: Assess a database's capability to efficiently combine transactional processing (OLTP) and analytical workloads (OLAP) within a single engine.

Key Features:

- Deployment/architecture options (local, embedded, distributed)
- Storage modes (row-oriented, columnar, hybrid)
- Multi-threading optimizations

Strengths:

- Support for concurrent transactional and analytical access
- Lightweight OLAP-style queries via SQL engines
- Excellent multi-threaded processing

Common Gaps:

- Limited native columnar or hybrid storage
- Variability in multi-threading capabilities
- Often require workarounds for materialized views and automatic refresh

Use Case: Real-time dashboarding involves ingesting high-velocity transactional data and generating up-to-the-second KPIs and operational metrics.

Benchmarking Embedded Databases

MCKNIGHT
CONSULTING GROUP

Benchmark a Workload

- TPCx-IOT
  - With custom TPCx-IoT-like benchmark driver developed using Python, referring to a 2018 IEEE ICDE paper by Poess et al. to maintain the spirit of TPCx-IoT and ensure a fair, objective "apples-to-apples" comparison.
- Execute using **pairs of Amazon Web Services (AWS) EC2 instances**
  - Specifically, use **c5.4xlarge EC2 instance types** for both the client and server instances, placed in the same availability zone to ensure close network proximity
  - Each instance with 16 vCPUs (3.6GHz Intel Xeon Scalable Processors - Cascade Lake), 32 GB RAM, running Ubuntu 22.04 (Jammy Jellyfish) with EBS gp2 disk storage.
  - Database software for each platform installed on these client-server pairs.

Testing Methods

- A **custom-built Python application serves as the test driver**
- The application performs **three operational functions**:
  - **Insert**: IoT data was inserted into the Client database, measuring insert throughput and averages
  - **Query**: IoT data was queried on the Server database, measuring query throughput and averages
  - **Synchronization Latency**: The latest data synced to the Server was queried to measure synchronization latency
- Use Python multiprocessing pools were to **simulate multiple sensors**
- Each IoT sensor reading consists of a timestamp, sensor identifier, measurement, and padding, totaling **1,024 bytes per record**. The iot table schema included ts (UNIX timestamp), sensorid (random UUID), measurement (random normal variate), and padding (943 random ASCII characters)
- The tests simulate **various numbers of sensors** (16, 32, 64, 128, 256, 512, and 1,024), with each simultaneously producing sensor readings to the client database for ingestion
- Send Records to the server database using **each system's native synchronization utility**
- **Implement Analytical queries**, representative of dashboard-like queries (Maximum Reading, Minimum Reading, Average Reading, Reading Count) on the server side and run concurrently with ingest operations to simulate a live system.
  - These queries were filtered on a random five-second interval
- Run each platform at each sensor count for **two hours** to ensure performance consistency over time

Measurements

- **Sensor reading throughput**: Measured as **IoT records per second**, similar to the TPCx-IoT's IoTps metric.
- **Ingest latency**: Measured as the **average time elapsed for each ingested row**.
  - Not part of the official TPCx-IoT but was measured for comparison purposes.

Summary

Edge Computing with a Zero-DBA, Zero-ETL Embedded Database…

- Process data closer to its source to reduce delays and boost performance.
- Select lightweight databases that thrive in resource-limited settings.
- Implement reliable synchronization strategies to overcome intermittent connectivity challenges.
- Leverage edge databases to process data in real-time while maintaining consistency across distributed systems.

MCKNIGHT
CONSULTING GROUP

Edge Computing Evolved: Introducing the Zero-DBA, Zero-ETL Embedded Database

Presented by: William McKnight

President, McKnight Consulting Group

3 X Inc 5000

/in/wmcknight

www.mcknightcg.com
(214) 514-1444

TOP VOICE
★ 2024 ★
thinkers 360
OVERALL

I'VE BEEN FEATURED IN THE 2024
dataIQ™ 100 USA
THE MOST INFLUENTIAL PEOPLE IN DATA