Datenbankgestützte Messdatenspeicherung: Chancen und Grenzen im Vergleich zur dateibasierten Verwaltung

Anna Christina Hühnerbein (Matr. Nr: 3615719)

Seminar

ATESTEO GmbH & Co. KG

Dezember 2025

1 Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema „Datenbankgestützte Messdatenspeicherung: Chancen und Grenzen im Vergleich zur dateibasierten Verwaltung“ selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, sind kenntlich gemacht. Zudem versichere ich, dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.
Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereichs Medizintechnik und Technomathematik auszuhändigen.

Alsdorf, 15.12.'25, A. Hüttemann
Ort, Datum, Unterschrift Student

Inhaltsverzeichnis

2 Einleitung

2.1 Themenvorstellung

Die Firma ATESTEO GmbH & Co KG ist ein weltweit führender Spezialist im Drivetrain-Testing für die Automobilindustrie mit acht Standorten weltweit, von denen fünf in Deutschland liegen. Bei uns werden Fahrzeugkomponenten auf Prüfständen erprobt und Messdaten erfasst. ATESTEO GmbH & Co. KG wurde im Jahr 1986 in Aachen gegründet, mit dem Ziel, die Automobilindustrie bei der Entwicklung von Fahrzeugkomponenten zu unterstützen. Unser Angebot umfasst die Umweltsimulation im Prüfstand, bei der unter anderem die Lebensdauer, Leistungsfähigkeit sowie die Rückwirkung auf die Umwelt getestet werden, das NVH-Testing, um die auftretenden akustischen oder mechanischen Schwingungen von Fahrzeugkomponenten zu messen, Dauerlaufprüfungen, Wirkungsgradüberprüfungen, Funktionsuntersuchungen, E-Mobility-Testing, Analyse und Befunde beim Testing im Prüfstand sowie Fahrzeugerprobungen und Fahrversuche auf unserer Teststrecke und der Straße. Zudem bieten wir die zertifizierte Kalibrierung von Messtechnik an.

Neben diesen Bereichen haben wir u.A. Werkstätten, eine IT-Abteilung und eine eigene Softwareabteilung.

Die Softwareabteilung schreibt Software ausschließlich für den internen Gebrauch. Unter anderem haben wir unsere Prüfstandautomatisierungssoftware PDES entwickelt. Hierbei nutzen die Ingenieure an den Prüfständen eine eigene, in die Automatisierungssoftware integrierte Skriptsprache, um die Prüfläufe an die Kundenanforderungen und die genutzten Geräte anzupassen. Des weiteren haben wir unser Analyseprogramm für Messdaten DANA sowie viele Programme entwickelt, um die Arbeitsprozesse standortübergreifend zu vereinheitlichen und zu vereinfachen. Der Großteil unserer Projekte ist in der Programmiersprache Delphi geschrieben.

Um zu Untersuchen, ob eine zentrale Ablage von Messdaten geeignet und hilfreich ist, erörtert die vorliegende Arbeit die Probleme und Möglichkeiten einer datenbankgestützten Speicherung von Messdaten.

Im Folgenden werden die derzeit vorliegenden Programmabläufe, die Probleme und Anforderungen sowie mögliche Lösungsansätze dargestellt. Zum Abschluss werden die Ergebnisse zusammengefasst, gefolgt von einem Ausblick auf die weitere Entwicklung.

2.2 Aufbau Prüfstand

Ein Prüfstand verfügt über einen Prüfling, an den mehrere Sensoren und Steuergeräte angeschlossen sind, die in hochfrequenten Abständen Daten liefern. Zudem sind meist Elektromaschinen mit dem Prüfling verbunden, um die realen Krafteinwirkungen zu simulieren. Es können auch noch weitere Geräte für verschiedene Untersuchungen angeschlossen werden.

Das Automatisierungssystem wird auf zwei Computer aufgeteilt: SIU und RTU. Die SIU ist die Schnittstelle zum Benutzer und zum Entwicklungssystem, in dem unsere Ingenieure die Prüfsequenzen als Steuerprogramm schreiben. Dieses wird in einen PCode übersetzt und zur RTU übertragen. Die RTU dient dazu, den zeitgesteuerten Ablauf des Steuerprogramms zu gewährleisten und stellt unter anderem eine Verbindung zu den angeschlossenen Sensoren und Geräten her. Während der Prüfläufe fallen auf der RTU große Mengen an Messdaten an, die alle erfasst und gespeichert werden müssen. Die Messwerte werden auf der lokalen Festplatte oder einem angeschlossenen Netzlaufwerk (NAS) gespeichert. Die Speicherung kann in verschiedenen

Formaten, z.B. MDF4, E.d.a.s. oder FAMOS erfolgen. In der SIU können während des Prüflaufs die Daten außerdem auf der Benutzeroberfläche angezeigt werden.

SIU, RTU und NAS sind über einen Prüfstands-internen Switch verbunden.

Die von uns eingesetzten NAS sind in der Lage, Docker-Images zu installieren, was bisher nicht genutzt wird. SIU und NAS sind mit einem weiteren Switch an das Firmen-LAN für den Austausch von Programmen und Daten verbunden. Die RTU und die anderen Geräte haben aus Sicherheits- und Performance-Gründen keinen Zugriff auf das Firmen-LAN. SIU und NAS sind als einzige Komponenten zusätzlich mit dem Firmen-LAN verbunden. Die Anbindung an das Firmen-LAN ermöglicht den Zugriff und die Übertragung von Daten.

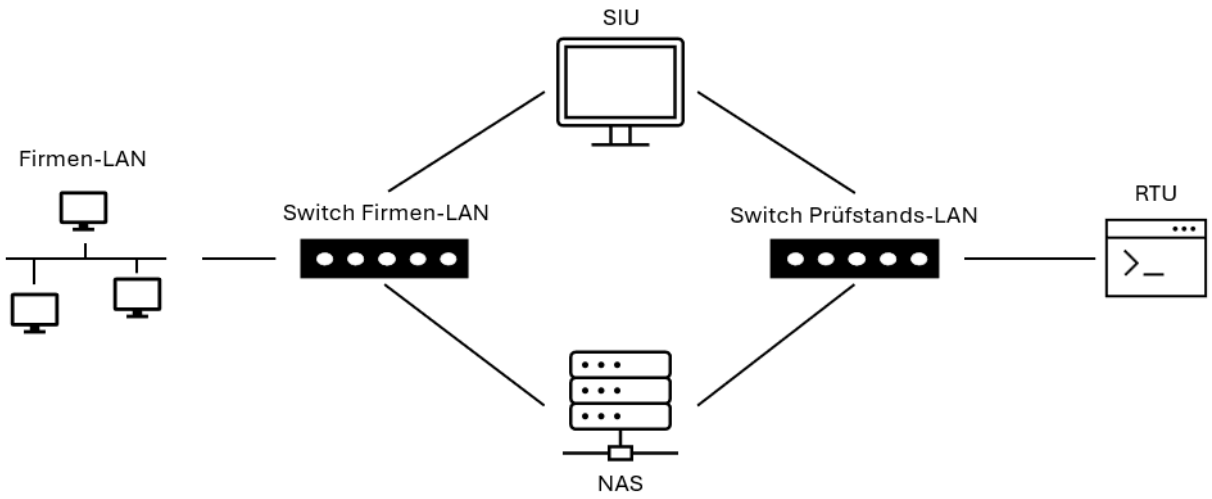

Abbildung 2.1: Netzwerkarchitektur eines Prüfstandes

3 Speichern der Messdaten

Nachfolgend erläutere ich das aktuelle Verfahren zum Speichern der Messdaten sowie mögliche Ansätze für eine zukünftige Speicherung in der Datenbank.

3.1 Derzeitiger Stand

Die Messdaten müssen in Echtzeit erfasst werden. Um dies zu gewährleisten, wird die Windows-Echtzeiterweiterung der Firma Kithara Systems eingesetzt. Diese Erweiterung wird im Folgenden, wie es bei uns intern üblich ist, kurz Kithara genannt. Kithara selbst verfügt unter Anderem über verschiedene echtzeitfähig implementierte Protokolle, um mit Sensoren oder Geräten zu kommunizieren.

Die Messungen werden mit sehr kleinem Jitter, sehr geringen Zeitungenauigkeiten, in das vom Kunden bestimmte Dateiformat geschrieben und auf der lokalen RTU-Festplatte oder auf dem NAS gespeichert.

Da wir oft in einer zusätzlichen Messdatei messen, um die Daten besser analysieren und Debuggen zu können, müssen die Messdaten teilweise doppelt geschrieben werden.

3.2 Potentieller Implementierungsansatz

Bei einer Umstellung auf die Speicherung von Messdaten in Datenbanken darf der Ablauf des Programms nicht beeinträchtigt werden. Zusätzlich muss weiterhin gewährleistet sein, dass die Daten verlustfrei gesichert werden.

Um ein alternatives System zur Speicherung der Messdaten zu verwenden, muss dieses in die Automatisierungssoftware PDES integriert werden.

Folgende Implementierungsmöglichkeiten bieten sich an:

- Eine Möglichkeit besteht darin, dass die Speicherung der Messdaten in Dateien unverändert bleibt. Dabei sendet das Automatisierungsprogramm nach einer abgeschlossenen Messung ein Signal an ein externes Programm, das nun auf die Datei zugreifen kann. Dann ist es dem Programm möglich, die Daten in die Datenbank zu übermitteln. Falls dies erfolgreich ist, wird die Datei gelöscht. Ansonsten bleibt die Datei erhalten. Solche Dateien können nach Behebung der Übertragungsprobleme in die Datenbank transferiert werden. Ein Datenverlust ist damit ausgeschlossen.

- Alternativ wird die Automatisierungssoftware modifiziert, damit die Daten statt in Dateien direkt über eine API an ein externes Programm gesendet werden. Dieses ist für die Speicherung der Daten in einer Datenbank verantwortlich. Wie bei der ersten Möglichkeit muss über einen Cache bzw. Zwischenspeicher in Form von Dateien auf der Festplatte sichergestellt werden, dass keine Daten verloren gehen können. Bereits während einer laufenden Messung kann das Programm Daten an die Datenbank übertragen. Probleme bei der API-Kommunikation sowie Cache-Probleme müssen unserer Automatisierungssoftware unmittelbar mitgeteilt werden, um ggf. einen Prüflauf zu unterbrechen.

  Dadurch würde der RTU-Teil der Automatisierung vom Datei Handling befreit, wodurch die Einhaltung der Echtzeit-Verarbeitung weiter stabilisiert werden kann.

- Eine dritte Möglichkeit wäre, dass die Automatisierungssoftware direkt mit der Datenbank kommuniziert. Allerdings erhöht dies die Komplexität und kann die Echtzeit-Verarbeitung gefährden.

4 Anforderungsanalyse

Im Anschluss nenne ich die Problemstellung und liste die daraus resultierenden Anforderungen auf.

4.1 Problemstellung

Bei den in der Einleitung bereits beschriebenen Prüfläufen werden die Messdaten meist mit einer Frequenz von 1 kHz von unserer Automatisierungssoftware erfasst. Dort werden die Messdaten verarbeitet und in eine Datei geschrieben. Diese Dateien beinhalten nicht nur Messdaten, sondern auch berechnete Werte und teilweise textuelle Metadaten, wie z.B. Getriebe- und Seriennummer sowie Zyklus-Bezeichnungen. Beim Schreiben der Dateien werden verschiedene Dateiformate unterstützt. Der Kunde erwartet Messdateien in einem von ihm bestimmten Format mit den von ihm festgelegten Messkanälen. Wir messen oft in einer zusätzlichen Messdatei in einem für uns ggf. besser unterstützen Dateiformat mit zusätzlichen internen Werten, um in Problemfällen diese besser analysieren und Debuggen zu können.

Ein Prüfdurchlauf besteht meist aus mehreren Zyklen, die oft in separate Messdateien aufgeteilt und auf Festplatten am Prüfstand abgelegt werden. Zu geeigneten Zeiten werden die Daten nun auf unsere Netzwerklaufwerke übertragen. Um die Echtzeitsteuerung und -regelung nicht zu gefährden, geschieht das, wenn möglich, zwischen Zyklen. Teilweise wird der Prüflauf dafür pausiert.

Während des Prüflaufs werden die Dateien momentan nicht komprimiert und aus den oben genannten Debug-Zwecken die Daten teilweise doppelt geschrieben und gespeichert. Daraus folgt, dass die Speicherung der Dateien speicherintensiv ist. An den Kunden werden meist die hier gespeicherten Rohdaten ausgeliefert.

In seltenen Fällen müssen Messungen vor der Auslieferung nachbearbeitet werden, z.B. durch Filter oder Kompression. Diese Nachbearbeitung geschieht mit unserer Analysesoftware DANA.

4.2 Anforderungen

Um einen Mehrwert für ein neues, einheitliches und verbessertes System zur Messdatenspeicherung zu bieten, werden im Folgendem Anforderungen aufgelistet.

4.2.1 Technische Anforderungen

Mindestanforderungen:

- Das System soll eine große Menge an hochfrequenten Daten aufnehmen und diese verlustfrei komprimieren können, um Speicherplatz einzusparen.
- Das System soll ausfallsicher sein. Die Daten sollen während oder unmittelbar nach jedem Zyklus in die Datenbank übertragen werden.
- Alle bereits verwendeten Dateiformate sollen gleichermaßen unterstützt werden.

Zusatzanforderung:

- Bestehende Daten können im Nachhinein in neue, später implementierte Formate exportiert werden.

- Beim Export in Dateien sollen ohne großen Aufwand z.B. Kanäle umbenannt, gefiltert und verrechnet werden können.

4.2.2 Weitere Anforderungen

Mindestanforderung:

- Der Datenschutz muss sichergestellt werden. Ein Rechtekonzept muss sicherstellen, dass Mitarbeiter nur auf Daten Zugriff haben, für die sie autorisiert sind.
- Die Daten zu abgeschlossenen Projekten müssen weiterhin archivierbar sein. Vertraglich sind wir dazu verpflichtet, die Dateien zehn Jahre lang für die Kunden aufzubewahren. Die Messdaten werden auf Band gespeichert. Die Metadaten müssen direkt zugreifbar sein.
- Das System soll in unser Automatisierungsprogramm integriert werden können, ohne dass die Echtzeitfähigkeit beeinträchtigt wird.

Zusatzanforderung:

- Das System soll ohne aufwändige Änderungen durch Ingenieure am Prüfstand in das Prüfstands-Steuerprogramm (Skripte) sowie in die Auswerteprogramme integriert werden können.

5 Datenbankarchitekturen und Datenbankformen

5.1 Zentrale- und dezentrale Datenbanken

Im Folgenden erörtere ich die Vor- und Nachteile von zentralen und dezentralen Datenbanken.

5.1.1 Zentrale Datenbank

Eine zentrale Datenbank befindet sich an einem zentralen Ort, z.B. auf einem Server im Serverraum der Firma. Dadurch gibt es nur eine gültige Informationsquelle. Die Backups und die Sicherung der Daten können zentral erfolgen. Wenn diese zentrale Stelle ausfällt oder nicht erreichbar ist, sind jedoch alle Clients betroffen.

5.1.2 Dezentrale Datenbank

Bei dezentralen Datenbanken liegen die Daten auf mehreren Datenbankservern, die an unterschiedlichen Orten stehen können. Es gibt verschiedene Modelle zur Realisierung. Bei einem Modell werden die Daten auf alle Server repliziert. Dabei läuft auf jedem Server eine eigene Datenbank, der Datenbestand ist auf allen Systemen gleich. Dies bietet die Möglichkeit, weiterhin Daten abfragen zu können, selbst wenn einzelne Server nicht erreichbar sind.

Bei diesem Modell unterscheidet man bei der Speicherung neuer Daten zwischen einem synchronen und einem asynchronen Aufbau. Beim synchronen Aufbau werden neue Daten sofort an alle anderen Server weitergeleitet. Sollte dies nicht möglich sein, führt dies zu Fehlern. Im Gegensatz dazu ist es beim asynchronen Aufbau erlaubt, dass die Datenbestände der Datenbanken zeitweise unterschiedlich sind. Dadurch können auch bei einem Verbindungsausfall zu einem Server weiterhin Daten aufgenommen und erst zu einem späteren Zeitpunkt zwischen den Servern synchronisiert werden.

In einem anderen Modell werden die Daten auf mehreren Knoten aufgeteilt, z.B. Aufteilung nach Messkanälen. Diese können zusammenfassend abgefragt werden. Dabei ist die Kommunikation zwischen den Knoten aufwändig, aber die Performance ist bei großen Datenraten höher, da weniger Daten pro Server übertragen und gespeichert werden.

5.2 Systemarchitektur im Zusammenhang mit Datenbanken

Im Zusammenhang zwischen Datenerfassung, Datenspeicherung und Datenabfrage ergeben sich in unserem Fall folgende mögliche Systemaufbauten.

5.2.1 dezentrale Datenerfassung mit zentraler Datenbank

Hierbei wird die Datenbank auf einem zentralen Server betrieben. Bei dieser Architektur senden alle Prüfstände ihre Daten über das Firmen-LAN an die zentrale Datenbank. Allerdings hätte ein Serverausfall zur Folge, dass keine Messdaten mehr gespeichert oder abgerufen werden könnten. Zudem ist die Arbeitslast der Datenbank und des Netzwerks sehr hoch, wenn mehrere Prüfstände gleichzeitig hochfrequente Daten an die Datenbank übermitteln. Um das Netzwerk entsprechend leistungsstark und ausfallsicher auszubauen, wären außerdem große Investitionen nötig.

5.2.2 zentrales System mit verteilter Datenbank

Dabei liegt auf jedem Prüfstands-NAS eine Datenbank. Beim synchronen Aufbau werden die Daten direkt auf die anderen Prüfstanddatenbanken repliziert. Im Gegensatz dazu kann die Replikation bei asynchronen Datenbanken zeitlich versetzt erfolgen. Dies bietet die Möglichkeit, Daten zu puffern, sollte es zu einem Netzwerkausfall kommen. Ein Ausfall einer Datenbank beeinträchtigt nicht die Datenspeicherung anderer Prüfstände. Nach der Synchronisation wären die Daten eines Prüfstands auch auslesbar, wenn die Verbindung zu diesem NAS ausfallen sollte. Nachteilig ist jedoch, wenn Daten überall hin synchronisiert werden, vervielfacht sich die Netzwerkbelastung und der benötigte Festplattenspeicher. Der erforderliche Investitions- und Ressourcenaufwand dieser Variante ist für uns aus wirtschaftlicher Perspektive nicht darstellbar.

5.2.3 Standalone-Datenbanken

Auch bei dieser Architektur verfügt jedes Prüfstands-NAS über eine eigene Datenbank. Diese nimmt nur Daten auf, die am angeschlossenen Prüfstand anfallen. Die Daten müssen über das Firmen-LAN beim jeweiligen Prüfstand direkt abgerufen werden. Dadurch wird das Netzwerk nicht durch Synchronisationen überlastet und der benötigte Speicher an jedem Prüfstand ist geringer, als bei der vorherigen Architektur. Darüber hinaus beeinträchtigt ein einzelner Systemausfall nicht die Datenbanken der anderen Prüfstände.

5.3 Relationale Datenbank

Neben einer Entscheidung bezüglich zentraler oder dezentraler Datenbankarchitekturen muss zusätzlich erörtert werden, welche Datenbankkonzepte für uns relevant sind. Im Folgenden erläutere ich das Konzept einer relationalen Datenbank.

5.3.1 Eigenschaften

Bei relationalen Datenbanken werden Tabellen mit Eigenschaften angelegt, die durch Beziehungen miteinander verknüpft sind, was über Fremdschlüsselverweise realisiert wird. Jeder Datensatz erhält in der Regel einen eindeutigen Primärschlüssel.

In der Datenbank liegen die Daten strukturiert vor und können dadurch auch mit komplexen Abfragebedingungen abgefragt werden. Eine häufig genutzte und weit verbreitete Abfragesprache ist beispielsweise SQL. Die wesentlichen Operationen sind Projektion, Selektion, kartesisches Produkt, Umbenennung, Vereinigung und Differenz.

Um Anomalien zu vermeiden, muss die Datenbank normalisiert sein. Aufgrund der Normalisierungsverfahren werden Datensätze separiert und über Fremdschlüsselverweise miteinander verknüpft. Dies kann schnell zu vielen Abhängigkeiten untereinander führen und die Komplexität der Datenbank erhöhen, wodurch hochfrequentes Einfügen, Modifizieren, Löschen und Auslesen einen Datenbankserver an seine Leistungsgrenzen bringen kann.

Dieser Datenbanktyp ist bis heute weit verbreitet, da die Tabellenstruktur erweiterbar ist, gut visualisiert werden kann, ein breites Anwendungsfeld abdeckt und SQL eine weit verbreitete Abfragesprache ist.

5.4 Zeitreihendatenbank

Im folgenden Abschnitt erörtere ich das Konzept einer Zeitreihendatenbank.

5.4.1 Eigenschaften

Eine Zeitreihendatenbank ist auf die Speicherung von Zeitreihen optimiert, also auf Wertefolgen, wie z.B. Messwerte, die nur in aufsteigender zeitlicher Abfolge abgelegt werden sollen. Die Daten werden nur angehängt und nur selten verknüpft oder anders sortiert. Für diese Optimierung werden die zeitlich geordnet eintreffenden Daten in Blöcke unterteilt und in einer Baumstruktur sortiert abgelegt [18], um schnelles Suchen zu ermöglichen. Mithilfe dieser Struktur können die relevanten Datenblöcke effizient nach Zeitstempel lokalisiert werden, ohne die gesamte Datenmenge durchsuchen zu müssen. Dies ist ein Unterschied zur Speicherarchitektur von relationalen Datenbanken, bei denen die Daten unsortiert abgelegt und über Index-Tabellen sortiert referenziert werden.

Durch das Fokussieren auf Zeitreihen und entsprechende Optimierungen mithilfe von Memory-Buffern [8] können auch konventionelle Serverarchitekturen hohe Datenraten von mehreren Quellen gleichzeitig, z.B. mehreren laufenden Prüfständen, ohne Leistungsverlust ablegen. Mit diesen Memory-Buffern können außerdem die Daten mit kurzer Latenz von auswertenden oder visualisierenden Clients abgefragt werden.

In Zeitreihendatenbanken ist es üblich, dass die auf der Festplatte abgelegten Daten nach einer konfigurierten Zeit, z.B. einer Woche, komprimiert werden. Hierfür werden für Zeitreihen optimierte Algorithmen verwendet, wie z.B. die Gorilla-Kompression [16]. Dabei werden die Zeitstempel mithilfe der Delta-of-Delta-Kompression komprimiert, die nur die Differenzen der Zeitstempel speichert, da bei wiederkehrenden und gleichbleibenden Intervallen diese Werte oft identisch oder nur leicht verändert sind. Viele Messwerte bleiben bei hoher Abtastrate über zahlreiche aufeinanderfolgende Intervalle hinweg gleich oder weisen nur geringfügige Veränderungen auf und werden mithilfe der XOR-Kompression komprimiert, wobei nur die wenigen veränderten Bits effizient komprimiert werden können.

Im Folgenden habe ich zwei weit verbreitete Zeitreihendatenbanken analysiert, die für uns geeignet wären.

5.4.2 InfluxDB

In diesem Abschnitt erfolgt eine Beschreibung des Systems InfluxDB, einschließlich der funktionalen Merkmale und Unterschiede der einzelnen Produktvarianten. Die InfluxDB wird vom Unternehmen InfluxData entwickelt und betrieben [12]. Die beiden kostenpflichtigen Cloud-Services werden nicht näher erläutert, da diese für ATESTEO nicht in Frage kommen. Kundendaten und Messungen müssen wegen der Informationssicherheit innerhalb von ATESTEO bleiben. Es bleiben die beiden Cloud-losen Varianten InfluxDB 3 Core und InfluxDB 3 Enterprise.

Abfragemöglichkeiten

Alle Influx-Varianten können mit folgenden Abfragearten arbeiten:

- InfluxQL: InfluxQL ist eine Abfragesprache und besonders für grundlegende Abfragen, Aggregationen und Zeitfenster geeignet. Diese ähnelt SQL, sodass die Einarbeitung mit Vorkenntnissen aus relationalen Datenbanken schnell möglich ist.

- Flux: Flux ist eine Skriptsprache. Sie erweitert die Funktionalität gegenüber InfluxQL erheblich, unterscheidet sich jedoch deutlich von SQL und erfordert daher eine intensivere Einarbeitung [2][10].

Die bei Influx genutzten Optimierungen für Zeitreihen lassen aus Performance- und Skalierungsgründen viele Eigenschaften relationaler Datenbanken weg. Einige der von SQL bereitgestellten Funktionen werden deshalb nicht unterstützt, wie z.B. Joins oder Subqueries.

Datenmodell

InfluxDB speichert Daten in zwei Kategorien [15]:

- Tag-Werte: Tag-Werte können nur Strings als Datentyp aufnehmen und sind besonders für Metadaten geeignet, die ein kategorisiertes Abfragen ermöglichen. Sie sind indiziert, wodurch sie innerhalb einer Messung filterbar und durchsuchbar sind. Mit einer steigenden Anzahl von Tag-Werten in einer Messung nimmt die Performance der Datenbank aufgrund des steigenden Verarbeitungsaufwands, ab.
- Feld-Werte: Feld-Werte können die Datentypen Int64, Float64, Uint64, Bool, String und Time aufnehmen und werden in Datenblöcken in einer Baumstruktur abgelegt. Die Spaltenanzahl der Feld-Werte darf sich zwischen den Datensätzen unterscheiden.

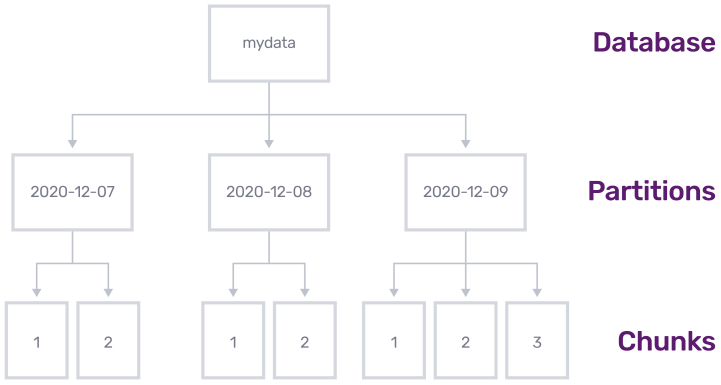

Abbildung 5.1: Speicherung von Daten bei InfluxDB über mehrere Abstraktionsschichten [11]

Sicherheit und Zuverlässigkeit

Der Client bündelt mehrere Datenpunkte in einem Batch-Write, wodurch weniger einzelne Schreiboperationen erforderlich sind. Das reduziert die Netzwerkbelastung und verbessert die Effizienz der Datenspeicherung.

Sobald die Daten erfolgreich übertragen worden sind, werden sie in einen Write-Ahead-Log (WAL) auf die Festplatte geschrieben. So können nach einem unerwarteten Ausfall, z.B. bei einem Stromverlust, die Daten mithilfe des WALs rekonstruiert werden.

InfluxDB 3 Core

InfluxDB 3 Core ist ein selbstverwaltetes System. Das bedeutet, Updates und Sicherheitskonfigurationen müssen vom Administrator durchgeführt werden. Es ist eine Open-Source-Variante von InfluxDB und steht unter der MIT/Apache 2 Lizenz [13] zur Verfügung.

Da es sich um ein Single-Node-System[1] handelt, kann eine Skalierung nur vertikal über die Hardware erreicht werden, wie z.B. durch das Aufrüsten von Arbeits- oder Festplattenspeicher des Servers.

Zudem verfügt das System nur über ein einfaches Benutzerrollensystem, in dem zwischen der Rolle „Administrator“ und „Benutzer“ unterschieden werden kann. Die Authentifizierung erfolgt über die Eingabe des Benutzernamens und des Passworts.

[1]Single-Node-System: Ein Single-Node-System ist eine Systemarchitektur, bei der sämtliche Verarbeitungsschritte, Datenhaltungen und Dienste auf einem einzelnen Server ausgeführt werden. Eine Verteilung auf andere Server ist nicht möglich, sodass jede Verarbeitung zentral abläuft.

InfluxDB 3 Enterprise

Hierbei handelt es sich um eine kostenpflichtige, erweiterte Variante der Open-Source-Variante[14], für die ein Vertrag mit InfluxData geschlossen wird. Dieser umfasst die technische Unterstützung sowie vertraglich definierte Reaktionszeiten bei Störungen. Die Verantwortung für den Betrieb, die Wartung und die Sicherheitskonfigurationen liegen beim Betreiber, in unserem Fall ATESTEO.

Das System zeichnet sich durch ein umfassendes Sicherheitskonzept aus und ist unter anderem nach der ISO 27001[2] zertifiziert. Da auch ATESTEO nach der ISO27001 zertifiziert ist, würde diese Tatsache bei den regelmäßigen Rezertifizierungen von Vorteil sein. Es bietet zusätzlich rollenbasierte Zugriffskontrollen, die Verschlüsselung der gespeicherten Daten und Audit-Logs, was weiter den Datenschutz erhöht.

Darüber hinaus kann InfluxDB Enterprise als Multi-Node-System[3] betrieben werden. Dadurch kann eine Lastverteilung und eine hohe Verfügbarkeit erreicht werden.

Die Engine dieser Variante ist identisch mit der von InfluxDB 3 Core, wodurch die bestehende Datenbank bei einem Umstieg auf Enterprise weiterhin verwendet werden kann.

5.4.3 TimescaleDB

Hierbei handelt es sich um eine Erweiterung für PostgreSQL, die von Timescale Inc entwickelt wird. In diesem Sinne kann TimescaleDB nicht als Datenbank bezeichnet werden. Nachfolgend werde ich TimescaleDB als Synonym für PostgreSQL mit dieser Erweiterung verwenden.

Eigenschaften

Die TimescaleDB ist eine Open-Source-Datenbank, die unter der Apache 2.0/Lizenz angeboten wird. Da sie eine Erweiterung ist, nutzt sie das System von PostgreSQL [7].

In diesem Dokument gehe ich ausschließlich auf die erweiterten Funktionen von TimescaleDB ein.

Datenmodell

TimescaleDB ermöglicht eine performante Speicherung von Zeitreihen in einer optimierten Datenablage. Sie nutzt Hypertables, die intern in Chunks unterteilt sind [5]. Hypertables bezeichnet das Zusammenfassen mehrerer PostgreSQL Tabellen. Diese haben intern Dimensionen, die Regeln darstellen, um die Hypertables in die Chunks aufzuteilen. Eine Dimension stellt den Zeitstempel dar, eine andere den Messwert und es kann zusätzlich eine Dimension von Metadaten hinzugefügt werden. Chunks stellen PostgreSQL Tabellen dar, die in Hypertables zusammengefasst werden. Die Erstellung erfolgt automatisch, basierend auf den zuvor definierten Dimensionen.

Weitere Spalten mit Metadaten können hinzugefügt werden. Da TimescaleDB auf den Datenstrukturen von PostgreSQL aufsetzt, ist die Anzahl der Spalten nach Definition, im Gegensatz zu InfluxDB, nicht veränderbar. Bei geänderten Spaltenanzahl müssen gefüllte Tabellen durch eine Schemamigration angepasst werden. Um dies zu vermeiden und das Schema flexibler zu gestalten, können alternativ weitere Metadaten in einer JSONB-Spalte gespeichert werden [6]. Dies beeinträchtigt jedoch die Performance und reduziert die mögliche Datenrate.

Zusätzlich werden Continuous Aggregates unterstützt, bei denen neu eingehende Messwerte automatisch im Hintergrund in aggregierte Werte einbezogen werden [3], z.B. die Berechnung

[2] ISO 27001: ISO 27001 ist eine internationale Norm für Informationssicherheits-Managementsysteme (ISMS). Sie legt Anforderungen an den Schutz von Informationen, Datenschutz und die Gewährleistung der Vertraulichkeit, Integrität und Verfügbarkeit von Daten fest.

[3] Ein Multi-Node-System teilt die Daten af mehrere Server mit eigener CPU, RAM, Speicher und Betriebssystem auf. Mit Redundanzen wird zusätzlich die Ausfallsicherheit erhöht und eine hohe Verfügbarkeit erreicht. Bei Bedarf kann das System um weitere Server skaliert werden, um steigende Lasten zu bewältigen.

eines Mittelwerts. Dadurch können Abfragen auf große Zeitreihen schnell ausgeführt werden, ohne dass die Aggregation bei jeder Abfrage neu berechnet werden muss.

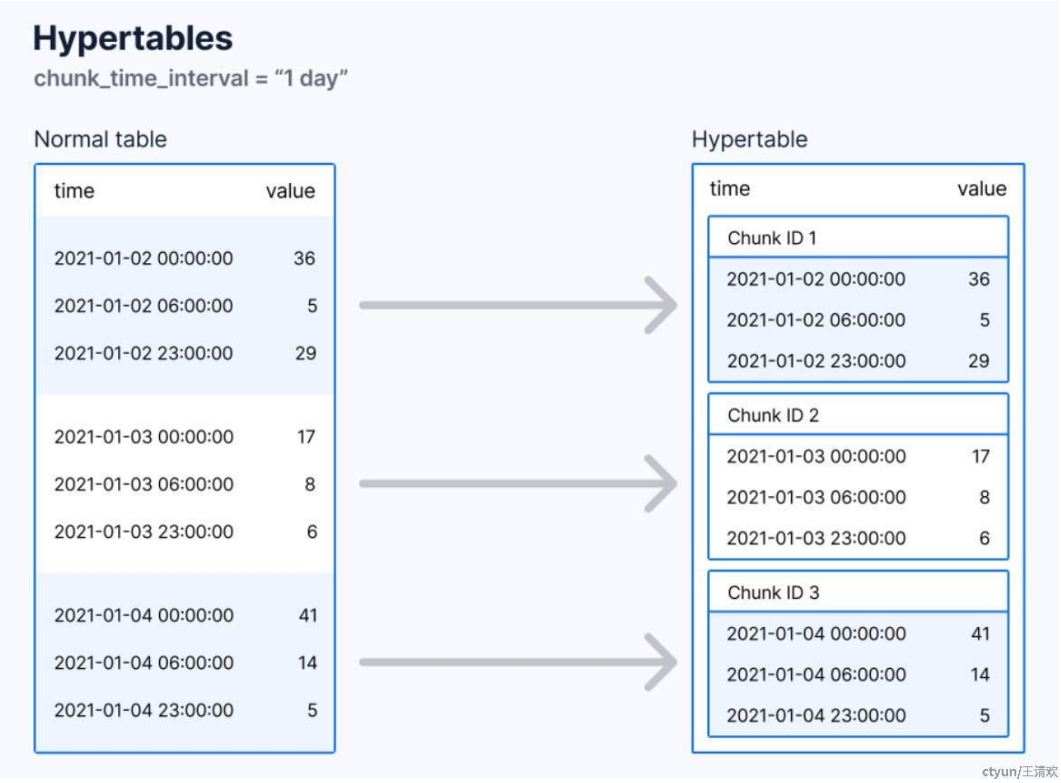

Abbildung 5.2: Von Tabellen zu Hypertables und Chunks[4]

5.5 Vektordatenbank

Diese Datenbank-Variante wurde im Vorfeld der Arbeit in Betracht gezogen, aber aufgrund der in dieser Arbeit erfolgten Einarbeitung zum Speichern von Messdaten stellte sich heraus, dass dieser Datenbanktyp hierfür nicht geeignet ist. Dieses Datenbank-Konzept ist für ATESTEO für zukünftige Einsatzzwecke trotzdem interessant, was im Folgenden erklärt wird.

5.5.1 Eigenschaften

Eine Vektordatenbank ist für das Speichern und Suchen von hochdimensionalen Vektoren optimiert. Jeder Vektor repräsentiert ein Objekt, wodurch es möglich ist, mathematische Ähnlichkeiten zwischen ihnen zu bestimmen [9][17][1].

Die Dimensionen dieses Raums entsprechen den zuvor festgelegten Merkmalen. Alle Vektoren in der Datenbank müssen die gleiche Anzahl an Dimensionen haben.

In der Vektordatenbank werden nicht die Rohdaten der Messungen gespeichert, sondern aus den Messdaten werden durch numerische Verfahren zusammenfassende Kennwerte berechnet und in den Dimensionen eines Vektors abgelegt. Diesen Vektoren können weitere Attribute wie Metadaten zugeordnet werden, nach denen bei Abfragen gefiltert werden kann, bevor die Vektorsuche durchgeführt wird.

Um Datensätze abzufragen, wird aus der Eingabe ein Query-Vektor berechnet, dem die einzelnen Merkmale übergeben werden. Im Anschluss werden mithilfe der Vektorähnlichkeit die ähnlichsten Einträge ermittelt und zurückgegeben.

Vektordatenbanken bieten die Möglichkeit, große Datenbestände zu speichern, systematisch auszuwerten und strukturelle Auffälligkeiten oder Trends zu identifizieren.

Zukünftige weitergehende Datenanalysen und der Einsatz von künstlicher Intelligenz können von der Zusätzlichen Nutzung von Vektordatenbanken profitieren.

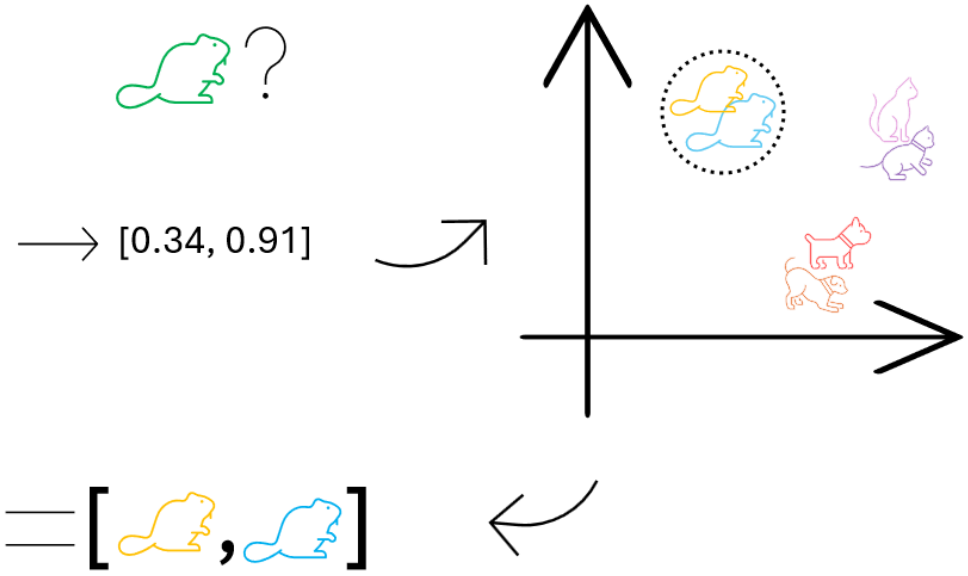

Abbildung 5.3: Abfrageablauf einer Vektordatenbank

6 Daten Extraktion

In diesem Kapitel beschreibe ich die Umsetzung für das Konvertieren in die Dateiformate und erläutere die Änderungen für unsere Auswertungssoftware.

6.1 Dateiformate und Konvertierung

ATESTEO unterstützt verschiedene Dateiformate, wie z.B. e.d.a.s. oder FAMOS, die auch bei einer neuen Umsetzung weiterhin unterstützt werden müssen.

Derzeit werden die Messdaten direkt nach dem Erfassen im vom Kunden gewünschten und für ATESTEO erforderlichen Dateiformat gespeichert. Bei der geplanten neuen Speicherung in einer Datenbank wird das Handling der Dateiformate von unserer Automatisierungssoftware entkoppelt. Somit ist die Formatkonvertierung unabhängig vom restlichen Programm und die Echtzeitfähigkeit kann zusätzlich stabilisiert werden.

Ein weiterer Vorteil wäre, dass Fehler, z.B. die falsche Benennung einzelner Header, einfacher nachbearbeitet werden können, da die notwendigen Daten jederzeit erneut ausgelesen und übertragen werden können. Dadurch wird das Gesamtsystem weniger fehleranfällig und Fehler können effizienter behoben werden.

6.2 Integration in vorhandene Software

Für das Auswerten, Anpassen und Visualisieren der Messdaten nutzen wir ein selbstgeschriebene Auswertungssoftware. Dies ermöglicht es, Dateien mit den erfassten Mess- und Metadaten einzulesen und entweder vorgefertigte Auswertungen, Visualisierungen oder Modifikationen manuell durchzuführen oder mit einem selbstgeschriebenen Skript automatisiert zu verarbeiten.

Für die Verwendung der in der Datenbank gespeicherten Daten stehen folgende zwei Varianten zur Verfügung:

- Es erfolgt keine Programmänderung und die Daten müssen manuell oder über ein Skript exportiert werden, bevor sie vom Auswertungstool genutzt werden können.
- Alternativ wird das Auswertungsprogramm dahingehend erweitert, dass es mit der Datenbank interagiert. Dadurch kann es die spezifischen Kanäle separat abfragen und Auswertungen, Visualisierungen sowie Anpassungen durchführen. Des Weiteren ist es möglich, die Analyseeigenschaften der Datenbank mit einzubeziehen. Bei dieser Umsetzung ist zu beachten, dass die Auswertungen und Visualisierungen der Messdaten nicht in einer Zeitreihen- oder Vektordatenbank gespeichert werden können. Das liegt daran, dass Relationen von Zeitreihen- und Vektordatenbanken nicht im gleichen Umfang unterstützt werden wie bei relationalen Datenbanken und dadurch keine Rückschlüsse auf die Ursprünglichen Daten möglich wären. Daher ist es notwendig, die Ergebnisse extern zu speichern oder sie direkt in Dateien zu übernehmen.

7 Fazit

Abschließend fasse ich die Ergebnisse zusammen und gebe einen Ausblick.

7.1 Zusammenfassung der Ergebnisse

Für den Anwendungsfall, eine Datenbank als alternative Speicherung zu nutzen, eignen sich insbesondere Zeitreihendatenbanken, da diese dafür optimiert wurden, Messdaten in hoher Frequenz aufzunehmen. Hierbei können zusätzlich Metadaten hinterlegt und Speicherplatz durch Datenkompression minimiert werden.

Relationale Datenbanken sind für diesen Anwendungsfall nicht ausgelegt und stoßen bei der hochfrequenten Speicherung von Daten schnell an ihre Grenzen.

Vektordatenbanken nehmen nur die berechneten Eigenschaften auf, aus denen die Rohdaten nicht mehr verlustfrei rekonstruiert werden können.

Vektordatenbanken können jedoch in zukünftigen Analyse- oder KI-Systemen, die auf Messdaten basieren, eingesetzt werden. Für das Training eines solchen Systems müssen im Vorhinein viele Daten beschrieben, Vorgaben sowie gemessene Ereignisse benannt und klassifiziert werden. Hierfür könnten die in der Zeitreihendatenbank abgelegten Messungen hervorragend verwendet werden.

Da ATESTEO vielen Kunden eine ISO27001-Zertifizierung nachweisen muss, müssen diese Anforderungen auch für die Speicherung der Messdaten angewendet werden. Daher sind gegebenenfalls weitere Anpassungen nötig, wie z.B. die Implementierung eines rollenbasierten Zugriffskonzepts, sofern ein solches nicht von der Datenbank unterstützt wird. Zudem muss sichergestellt sein, dass im Fehlerfall kein Datenverlust entsteht und die Systemstabilität gewährleistet bleibt.

Um Investitionen in Hardware- und Infrastruktur zunächst möglichst gering zu halten, wird zunächst die Standalone-Datenbank realisiert, wobei die Datenbank auf dem Prüfstands-NAS betrieben werden kann. Durch die Anbindung des NAS an das Firmen-LAN können aus dem Büro der Projektbeteiligten auf die Messdaten zugegriffen werden. Mit dieser Lösung wird das Netzwerk nicht überlastet und bei einem Systemausfall eines Prüfstands werden die anderen Datenbanken der anderen Prüfstände nicht beeinträchtigt.

Innerhalb der Automatisierungssoftware soll die Messdatenspeicherung an ein ausgelagertes Programmmodul über eine API gesendet werden. Diese Umsetzung ermöglicht eine Entkopplung der Messdatenspeicherung vom restlichen Programm der Automatisierungssoftware. Dadurch kann die Einhaltung der Echtzeit-Verarbeitung der Prüfstandsautomatisierung weiter stabilisiert werden.

Zusammenfassend lässt sich sagen, dass eine solche Umsetzung einen hohen Aufwand erfordert und mehrere Teilbereiche einiger Vorarbeit bedürfen, bevor eine datenbankgestützte Messdatenspeicherung realisiert werden kann.

Für zukünftige Anwendungen im Bereich der künstlichen Intelligenz ist eine solche Lösung also unverzichtbar. Sie schafft eine einheitliche Datengrundlage, auf der Modelle für künstliche Intelligenz aufgebaut, trainiert und weiterentwickelt werden können. Die bisherige Messdatenspeicherung eignet sich aufgrund der heterogenen Dateiformate und der damit fehlenden Standardisierung kaum als zuverlässige Datenbasis.

Der Einsatz einer Zeitreihendatenbank ermöglicht hingegen eine strukturierte, konsistente und skalierbare Ablage der Messdaten. Dadurch werden sowohl die Qualität als auch die Effizienz der weiteren Verarbeitung für eine solche Anwendung erheblich verbessert.

7.2 Ausblick

Aufgrund der langjährigen Nutzung von dateibasierter Messdatenspeicherung könnte es noch unbekannte Vorteile und Anwendungsfälle geben. Daher ist es notwendig, weitere Gespräche mit den Mitarbeitenden an den Prüfständen und der IT-Abteilung zu führen.

Die firmenweite Umstellung stellt einen sehr großen Aufwand dar, diese lässt sich aber über mehrere Ausbaustufen gefahrlos umsetzen. Zu erst werden die Datenbanken aufgesetzt. Die Automatisierungssoftware kann zunächst unveränderte Messungen als Dateien speichern, die über ein eigenes Programm in die Datenbank übertragen wird. Später kann die Automatisierungssoftware über eine API direkt in die Datenbank speichern. Die Lieferung der Daten an den Kunden kann zunächst wie bisher die Dateien der Automatisierungssoftware nutzen und später durch Extraktionsprogramme die Daten aus der Datenbank beziehen.

Das Auswertungsprogramm kann zu Beginn mit den originalen Messdateien arbeiten, später mit extrahierten Messungen und schließlich direkt mit den Daten aus der Datenbank.

Im Anschluss an diese Seminararbeit kann als Prototyp dieses Systems mit seinen Bestandteilen im Büro eine Messbarkeitsanalyse erstellt werden. Mit früheren Messungen als Testdaten kann das Befüllen der Datenbank simuliert, Hardwarearchitekturen, Schreibraten und weiteres untersucht sowie Erfahrung gesammelt werden.

Literatur

[1] Moez Ali. *Die 5 besten Vektordatenbanken*. 1. Dez. 2025. URL: https://www.datacamp.com/de/blog/the-top-5-vector-databases.

[2] Saskia Behn. *Vergleich von relationalen und Zeitreihendatenbanken. Eine Untersuchung der Anfragekomplexität und -effizienz*. 20. Nov. 2025. URL: https://reposit.haw-hamburg.de/bitstream/20.500.12738/17666/1/BA_Vergleich%20von%20relationalen%20und%20Zeitreihendatenbanken.pdf.

[3] Tiger Data. *About continuous aggregates*. 1. Dez. 2025. URL: https://www.tigerdata.com/docs/use-timescale/latest/continuous-aggregates/about-continuous-aggregates.

[4] Tiger Data. *Best Practices for PostgreSQL Data Analysis*. 1. Dez. 2025. URL: https://www.tigerdata.com/learn/postgresql-data-analysis-best-practices.

[5] Tiger Data. *Hypertables*. 15. Dez. 2025. URL: https://www.tigerdata.com/docs/use-timescale/latest/hypertables.

[6] Tiger Data. *JSONB support for semi-structured data*. 1. Dez. 2025. URL: https://www.tigerdata.com/docs/use-timescale/latest/schema-management/json.

[7] Tiger Data. *Self-hosted TimescaleDB*. 1. Dez. 2025. URL: https://www.tigerdata.com/docs/self-hosted/latest.

[8] Marie Fayard. *Zeitreihen-Datenbank(TSDB): Ein Leitfaden mit Beispielen*. 20. Nov. 2025. URL: https://www.datacamp.com/de/blog/time-series-database.

[9] Databricks Inc. *Vektordatenbank*. 1. Dez. 2025. URL: https://www.databricks.com/de/glossary/vector-database.

[10] Influx Data Inc. *Flux vs InfluxQL*. 20. Nov. 2025. URL: https://docs.influxdata.com/influxdb/v2/reference/syntax/flux/flux-vs-influxql/.

[11] Influx Data Inc. *How InfluxDB IOx Manages the Data Lifecycle of Time Series Data*. 1. Dez. 2025. URL: https://www.influxdata.com/blog/how-influxdb-iox-manages-the-data-lifecycle-of-time-series-data/.

[12] Influx Data Inc. *InfluxDB*. 20. Nov. 2025. URL: https://www.influxdata.com/lp/influxdb-database/?utm_source=bing&utm_medium=cpc&utm_campaign=2020-09-03_Cloud_Traffic_Brand-InfluxDB_INTL&utm_term=influxdb&msclkid=03bfae38dff51536972fc72639eb518c.

[13] Influx Data Inc. *InfluxDB 3 Core documentation*. 20. Nov. 2025. URL: https://docs.influxdata.com/influxdb3/core/.

[14] Influx Data Inc. *InfluxDB 3 Enterprise documentation*. 20. Nov. 2025. URL: https://docs.influxdata.com/influxdb3/enterprise/.

[15] Influx Data Inc. *InfluxDB line protocol tutorial*. 26. Nov. 2025. URL: https://docs.influxdata.com/influxdb/v1/write_protocols/line_protocol_tutorial/.

[16] Tuomas Pelkonen u. a. *Gorilla: A Fast, Scalable, In-Memory Time Series Database*. 20. Nov. 2025. URL: https://www.vldb.org/pvldb/vol8/p1816-teller.pdf.

[17] Johannes Ruof. *Was sind Vektordatenbanken?* 1. Dez. 2025. URL: https://digitaleprofis.de/kuenstliche-intelligenz/wiki/was-sind-vektordatenbanken/.

[18] Wikipedia-Autoren. *Zeitreihendatenbank*. 20. Nov. 2025. URL: https://de.wikipedia.org/wiki/Zeitreihendatenbank.