CHAPTER 6

Database

Introduction

In today's data-driven landscape, the proper management and utilization of data are vital for any organization's success. As the digital world continues to grow, the AWS cloud platform offers a rich tapestry of database services and solutions that empower businesses to harness the full potential of their data. This chapter, aptly titled *Database*, embarks on a journey through the AWS database offerings, providing a comprehensive guide to understanding and utilizing these services to their fullest extent.

The AWS ecosystem houses an array of database services, each tailored to address specific data storage and management needs. From powerful relational databases like *Amazon RDS*[[1]](#footnote-1) and *Amazon Aurora*[[2]](#footnote-2)*,* to the flexibility of NoSQL databases such as *Amazon DynamoDB*[[3]](#footnote-3) and *Amazon DocumentDB*[[4]](#footnote-4). AWS leaves no stone unturned in providing versatile options. We will discuss the unique features and capabilities of each service, providing insights on when and how to employ them effectively in diverse use cases.

In this chapter, we will explore the intricacies of scaling databases, ensuring high performance, and guaranteeing data reliability. With services like *Amazon ElastiCache*[[5]](#footnote-5) for caching and, *Amazon Redshift*[[6]](#footnote-6) for analytical data warehousing, to tackle complex workloads with ease. For time-series data, *Amazon Timestream*[[7]](#footnote-7) is your preferred option, while *Amazon Neptune*[[8]](#footnote-8) caters to graph database requirements. We will examine how these services can revolutionize your data storage and retrieval strategies.

Additionally, we will deal with AWS databases, where you will be equipped with the knowledge and skills needed to harness the full potential of your data. Whether you are dealing with structured or unstructured data, require real-time analytics, or need a scalable and highly available database solution, this chapter will guide you through the intricacies of selecting and implementing the right AWS database services for your unique business needs.

This introduction sets the stage for a deep dive into AWS database services, highlighting their diversity, specialized features, and relevance to different use cases. Feel free to use and adapt this introduction for your book, *AWS Cloud Computing Master Class*.

Structure

This chapter includes the following topics:

* Databases: the foundation of data management
* Amazon Aurora: a high-performance relational database
* Amazon DocumentDB
* Amazon DynamoDB
* Managed NoSQL database
* Amazon ElastiCache
* Amazon Keyspaces (for Apache Cassandra)
* Amazon MemoryDB for Redis
* Amazon Neptune
* Amazon RDS
* Amazon Redshift
* Amazon Timestream

Objectives

Databases: the foundation of data management

Databases serve as the bedrock of modern data management, and their role is pivotal in the digital transformation of organizations. They are structured repositories that store, organize, and facilitate access to data, making it readily available for various applications and use cases. Regarding cloud-based database services, **Amazon Web Services** (**AWS**) offers a comprehensive suite of solutions that caters to a wide range of business requirements[[9]](#footnote-9).

This section provides a foundational understanding of databases and introduces the AWS database ecosystem. Scholarly articles are not referenced in this part, as this serves as an introductory overview. You can incorporate specific scholarly articles in the following sections that delve deeper into each database service.

Understanding the AWS database ecosystem

The AWS database ecosystem encompasses various types of databases, each designed to address specific data storage and management needs. For those familiar with traditional relational databases, Amazon **Relational Database Service** (**RDS**)[[10]](#footnote-10) provides a managed solution for popular databases like *MySQL, PostgreSQL, Oracle,* and *SQL Server*. RDS automates routine administrative tasks, making it easier to set up, operate, and scale a relational database.

If NoSQL databases are more in line with your needs, *Amazon DynamoDB*[[11]](#footnote-11) delivers a fast and flexible, fully managed NoSQL database service. It can handle high-traffic applications and provides low-latency, reliable performance.

AWS also offers specialized database services like *Amazon Redshift*[[12]](#footnote-12) for data warehousing, *Amazon Timestream*[[13]](#footnote-13) for time-series data, and *Amazon Neptune*[[14]](#footnote-14) for graph databases. These services cater to specific data models and use cases, making AWS a one-stop destination for all your database needs.

Scalability and security

One of the primary advantages of AWS databases is scalability. Cloud-based databases can easily adjust to changing workloads, automatically expanding to accommodate increasing data and traffic. This ensures your applications remain responsive and cost-effective as your business grows.

AWS places a strong emphasis on data security. With features like data encryption at rest and in transit, identity and access management controls, and database audit capabilities; AWS databases are equipped to protect your data from unauthorized access and data breaches[[15]](#footnote-15).

Reliability and availability

High availability and reliability are vital for databases. AWS ensures your databases run smoothly with features such as automated backups, automated software patching, and the ability to deploy your database in multiple availability zones, for redundancy[[16]](#footnote-16).

With AWS, you can choose between fully managed database services or self-managed databases on EC2 instances, offering flexibility to accommodate various needs and expertise levels[[17]](#footnote-17).

Databases are the backbone of data-driven organizations, and AWS provides a comprehensive array of database services to meet your data storage and management needs. In the subsequent sections of this chapter, we will explore these services in greater detail, examining their features, use cases, and best practices for implementation.

In the next section, we will discuss Amazon RDS, AWS's managed relational database service, and explore its features and benefits for your cloud-based data management needs.

Amazon Aurora: a high-performance relational database

*Amazon Aurora*[[18]](#footnote-18), a product of AWS, has rapidly gained recognition and popularity for its exceptional capabilities as a high-performance, fully managed relational database engine. As a dynamic cloud-based service, it addresses the demands of businesses and enterprises by delivering the robustness and capabilities of commercial-grade databases while eliminating the intricacies, overheads, and high costs often associated with them.

This section provides an in-depth overview of Amazon Aurora, its key features, and highlights scholarly articles and AWS documentation. The references are cited in the text where they are used. You can incorporate additional scholarly articles in the following sections, that discuss other AWS database services.

Key features of Amazon Aurora

*Amazon Aurora* brings to the table a range of outstanding features, making it an ideal choice for AWS users who require a powerful relational database:

* **Compatibility**: Amazon Aurora is compatible with *MySQL* and *PostgreSQL,* ensuring a seamless migration process for existing databases. Its *drop-in replacement* nature means that transitioning to Aurora is hassle-free, requiring minimal adjustments to your applications.
* **High performance**: Aurora is engineered to deliver exceptional performance, offering up to five times the throughput of standard MySQL databases and up to two times the throughput of standard *PostgreSQL* databases, all while running on the same hardware. This level of performance makes it an optimal choice for applications with high transaction and data processing requirements.
* **High availability**: Designed with high availability in mind, Amazon Aurora replicates data across six instances in three availability zones. This strategy not only enhances reliability but also ensures seamless failover support, effectively minimizing downtime.
* **Fault-tolerant storage**: The storage layer of Amazon Aurora is notable for its fault tolerance. It continuously backs up your data to Amazon S3 while also ensuring transparent recovery from any physical storage failures. This provides the peace of mind that your data is well protected and recoverable.
* **Global databases**: For applications that cater to a global user base, *Amazon Aurora's Global Databases*[[19]](#footnote-19) feature is invaluable. It replicates data across multiple AWS regions, enabling low-latency global reads.

Performance benchmarking

Scholarly articles and independent studies consistently demonstrate *Amazon Aurora's* superior performance when compared to other databases. Research by *Faleiro, Van Renesse*, and *Rodrigues* (2016)[[20]](#footnote-20) found that Aurora outperformed *MySQL* and handled high-throughput workloads with significantly lower latencies, confirming its position as a high-performance database solution.

AWS also highlights Aurora's ability to scale read operations linearly, supporting up to 15 read replicas for both MySQL and PostgreSQL compatibility[[21]](#footnote-21) mance^]. This scalability is crucial for applications that require efficient and dynamic handling of read operations.

Case studies

A collection of case studies within AWS customer success stories[[22]](#footnote-22) provides concrete examples of how organizations have harnessed the power of *Amazon Aurora*. These case studies underscore the high availability and scalability Aurora offers, bringing real-world applications into sharper focus.

*Amazon Aurora* is a compelling choice for organizations in search of a cost-effective solution that encompasses the performance and reliability characteristic of commercial-grade databases. It boasts compatibility with *MySQL* and *PostgreSQL*, exceptional performance, high availability, fault-tolerant storage, and global database capabilities. In the following sections of this chapter, we will delve into a multitude of other AWS database services, each tailored to distinct use cases.

The subsequent section, *Amazon DocumentDB*, will guide you through an exploration of AWS-managed document database services compatible with *MongoDB.*

Amazon DocumentDB

*Amazon DocumentDB*, an integral part of AWS, is a fully managed, *MongoDB*-compatible[[23]](#footnote-23) database service designed to offer high performance, scalability, and availability25. Its unique value proposition lies in ensuring seamless compatibility with existing MongoDB applications26. This MongoDB compatibility is paramount for organizations seeking to migrate their MongoDB workloads to the cloud.

Key features and advantages

The following are some of the features and advantages of *Amazon DocumentDB:*

* **MongoDB compatibility**: Amazon *DocumentDB* is meticulously engineered to be fully compatible with *MongoDB*, one of the most renowned NoSQL databases in the industry26. What truly sets DocumentDB apart is its compatibility which extends to the application level. This means that you can effortlessly employ your existing *MongoDB*[[24]](#footnote-24) drivers and code to interact with *DocumentDB*, ensuring a smooth transition for your applications[[25]](#footnote-25).
* **Scalability****[[26]](#footnote-26)**: *DocumentDB* seamlessly integrates horizontal scaling to cater to the demands of growing applications26. The ability to easily add or remove read replicas empowers you to efficiently distribute read traffic and guarantee low-latency responses, ensuring your database can handle surges in workloads[[27]](#footnote-27) [[28]](#footnote-28).
* **High availability**: Amazon DocumentDB prioritizes high availability through automatic failover mechanisms26. Data is replicated across six different instances spanning three availability zones. This strategy is designed to protect against hardware failures, significantly enhancing durability and availability25.
* **Security**: Security remains at the forefront of any database service, and DocumentDB does not disappoint26 It supports VPC peering to isolate your database within a VPC, encrypts data at rest, and incorporates built-in audit logging to ensure the confidentiality and integrity of your data25.
* **Managed service**: As a fully managed service, Amazon DocumentDB effectively removes the burden of time-consuming administrative tasks, including hardware provisioning, patching, setup, configuration, and backups25. This alleviates operational overhead, allowing you to concentrate on developing and enhancing your applications.

Use cases

The versatile nature of *Amazon DocumentDB* lends itself to a broad spectrum of use cases25. Some common scenarios where *DocumentDB* excels include content management systems, catalogs, user profiles, and applications demanding real-time analytics[[29]](#footnote-29). Furthermore, its exceptional ability to handle high read workloads[[30]](#footnote-30) positions it as an invaluable choice for applications dependent on swift and efficient data retrieval25 [[31]](#footnote-31).

Amazon DynamoDB

*Amazon DynamoDB* is a managed NoSQL database service provided by AWS. It is designed to deliver fast and predictable[[32]](#footnote-32) performance while ensuring seamless scalability. DynamoDB's flexibility, reliability, and ability to handle various data models make it a popular choice for applications ranging from mobile and web to gaming and IoT[[33]](#footnote-33).

Key features and advantages

The following are some of the features and advantages of *Amazon DynamoDB:*

* **Fully managed:** *DynamoDB* is a serverless, fully managed database service, which means AWS takes care of the operational aspects such as hardware provisioning, configuration, patching, and backups[[34]](#footnote-34). This allows developers[[35]](#footnote-35) to focus on building applications without worrying about database management[[36]](#footnote-36).
* **Scalability:** *DynamoDB* provides seamless and automatic scaling, ensuring that your application can handle variable workloads34. You can easily scale up or down based on your application's demands[[37]](#footnote-37).
* **Performance:** It is known for its fast and predictable performance34. You can achieve single-digit millisecond response times, making it ideal for applications that require low-latency data access.
* **Multi-region replication:** *DynamoDB* offers global tables, which allows you to replicate your data across multiple AWS regions for disaster recovery and low-latency access globally34.
* **Security:** Data security is a top priority[[38]](#footnote-38). DynamoDB provides fine-grained access control using AWS **Identity and Access Management** (**IAM**)34 and offers encryption at rest and in transit[[39]](#footnote-39).

Use cases

Amazon DynamoDB is suitable for a wide range of applications, including e-commerce platforms, mobile applications, gaming leaderboards, session management, IoT data storage, and more34. Its versatility and scalability make it an excellent choice for applications that require seamless scaling based on demand[[40]](#footnote-40).Top of Form

Managed NoSQL database

A *Managed NoSQL* database, as offered by AWS, is a powerful and flexible data storage solution that caters to modern application needs. These databases provide a schema-less data model, allowing developers to store, retrieve, and manage unstructured or semi-structured data efficiently.

In this section, we will explore the advantages and use cases of managed NoSQL databases and their significance in modern application development.

Key features and advantages

The following are some of the features and advantages of *managed NoSQL database:*

* **Schema-less data model:** Managed NoSQL databases, such as Amazon DynamoDB, provide a schema-less data model, which means you can store and retrieve data without the need for a predefined schema. This flexibility is particularly useful for applications that deal with rapidly changing data structures34.
* **Highly scalable:** Managed NoSQL databases can seamlessly scale to handle increasing workloads. As data requirements grow, these databases can be adjusted to accommodate the changes34.
* **Low-latency access:** NoSQL databases are optimized for low-latency data access[[41]](#footnote-41), making them suitable for applications that require real-time data retrieval34.
* **Multi-model support:** Many managed NoSQL databases support various data models[[42]](#footnote-42), including document, key-value, graph, and wide-column store[[43]](#footnote-43). This flexibility allows you to choose the right data model for your specific application requirements21.

Use cases

Managed NoSQL databases are well-suited for a variety of applications, including content management systems, mobile and web applications, e-commerce platforms, gaming, IoT, and more34. Their ability to handle unstructured and rapidly evolving data makes them a valuable choice for businesses that need dynamic and scalable data storage.

Amazon ElastiCache

*Amazon ElastiCache*, a fully managed in-memory data store service provided by AWS, is designed to accelerate the performance of applications by enabling real-time data caching. It offers compatibility with popular open-source in-memory data stores like *Redis*[[44]](#footnote-44) and *Memcached*[[45]](#footnote-45), allowing you to seamlessly deploy, operate, and scale an in-memory cache for your applications.

In this section, we will discuss the features, advantages, use cases, and significance of *Amazon ElastiCache* as an essential tool for enhancing the performance of AWS-based applications.

Key features and advantages

The following are some of the features and advantages of *Amazon ElastiCache:*

* **Caching data for faster access:** *Amazon ElastiCache* stores frequently accessed data in-memory, reducing the need to fetch it from the main database. This caching mechanism significantly reduces data retrieval times and enhances application responsiveness[[46]](#footnote-46).
* **Compatibility with Redis and Memcached[[47]](#footnote-47):** *ElastiCache* supports both *Redis* and *Memcached*, two widely used open-source in-memory data stores. This flexibility allows you to choose the data store that best fits your application's requirements[[48]](#footnote-48).
* **Auto-scaling:** *ElastiCache* can automatically adjust its capacity based on your application's needs. This ensures that your cache is always appropriately sized, optimizing cost and performance[[49]](#footnote-49).
* **Managed service:** As a managed service, *ElastiCache* handles operational tasks like hardware provisioning, software patching, setup, and configuration. This allows you to focus on your application's development rather than infrastructure management[[50]](#footnote-50).

Use cases

*Amazon ElastiCache* is a valuable addition to applications that require low-latency data access[[51]](#footnote-51). It is commonly used for use cases such as session management, real-time analytics, leaderboards, and read-heavy workloads. Any application that benefits from quick access to frequently accessed data can leverage *ElastiCache* to improve[[52]](#footnote-52).

Amazon Keyspaces (for Apache Cassandra)

*Amazon Keyspaces* (for *Apache Cassandra*)[[53]](#footnote-53) is a fully managed, serverless, scalable, and highly available database service offered by AWS. It is designed to provide the best of both worlds, the scalability and flexibility of *Apache Cassandra*[[54]](#footnote-54) and the benefits of AWS's managed services. *Amazon Keyspaces* allows you to build applications that can seamlessly and securely handle any amount of traffic and data[[55]](#footnote-55), making it a robust choice for a wide range of use cases.

In this section, we will explore *Amazon Keyspaces* (for *Apache Cassandra*), its features, benefits, and the use cases where it shines as a fully managed, serverless, and scalable database service.

Key features and advantages

The following are some of the features and advantages of *Amazon Keyspaces*:

* **Serverless and managed:** *Amazon Keyspaces* eliminates the need for database management tasks, such as hardware provisioning, setup, and patching. You can focus on building applications while AWS takes care of operational tasks[[56]](#footnote-56).
* **Scalability:** It provides a highly scalable database that can handle large amounts of data and traffic[[57]](#footnote-57). You can easily scale up or down to match the requirements of your application[[58]](#footnote-58).
* **Compatibility with Apache Cassandra:** *Amazon Keyspaces* is compatible with the Apache **Cassandra Query Language** (**CQL**). This means you can migrate your existing *Cassandra* workloads to *Amazon Keyspaces* with minimal code changes[[59]](#footnote-59).
* **Security and compliance:** It offers robust security features such as encryption at rest and in transit, identity and access management, and audit logging. *Amazon Keyspaces* also helps you maintain regulatory compliance[[60]](#footnote-60).

Use cases

*Amazon Keyspaces* is well-suited for applications that require seamless scaling and high availability. Typical use cases include customer-facing applications, **Internet of Things** (**IoT**) applications, and various scenarios where data must be distributed across multiple regions while maintaining low latency and high availability.[[61]](#footnote-61)

Amazon MemoryDB for Redis

*Amazon MemoryDB* for *Redis* is a fully managed, Redis-compatible, in-memory database service offered by AWS. It is designed for applications that require a fast, real-time data store with sub-millisecond response times. *Redis* is an open-source, in-memory key-value data store that is widely used for caching and real-time analytics. *Amazon MemoryDB* for *Redis* builds upon *Redis* to provide a highly available and scalable managed service.

In this section, we will discuss about *Amazon MemoryDB* for *Redis*, exploring its compatibility with Redis, high availability, performance, and use cases[[62]](#footnote-62). We will also discuss how it can benefit applications that require low-latency, real-time data storage and retrieval[[63]](#footnote-63).

Key features and advantages

The following are some of the features and advantages of *Amazon MemoryDB:*

* **Compatibility:** *Amazon MemoryDB* for Redis is fully compatible with Redis, which means you can use your existing Redis clients and code with minimal changes. It supports Redis data types and commands, making it easy to migrate existing applications to the managed service.[[64]](#footnote-64)
* **High availability:** It offers multi-**Availability Zone** (**AZ**) deployments, ensuring data redundancy and automatic failover. This helps maintain high availability and durability for your applications[[65]](#footnote-65).
* **Performance:** *Amazon MemoryDB* for *Redis* is optimized for read-heavy and write-heavy workloads. It provides sub-millisecond response times, making it suitable for real-time and low-latency applications[[66]](#footnote-66).
* **Security:** The service supports encryption at rest and in transit, along with identity and access management, to help secure your data. You can also use Amazon **Virtual Private Cloud** (**VPC**)[[67]](#footnote-67) peering to isolate your *MemoryDB* clusters[[68]](#footnote-68).

Use cases

Amazon *MemoryDB* for *Redis* is an excellent choice for use cases that require a fast and highly available in-memory database. These include caching, session management, leaderboard and counting systems, and real-time analytics for applications such as gaming, ad targeting, and e-commerce[[69]](#footnote-69).

Amazon Neptune

*Amazon Neptune* is a fully managed graph database service provided by AWS. It is designed for storing and querying highly connected data with complex relationships, making it a powerful tool for applications in various domains, including social networking, recommendation engines, and knowledge graphs.

In this section, we will explore *Amazon Neptune*, its features, compatibility with different graph models, scalability, and use cases. This will help you understand how this managed graph database service a valuable addition to your application architecture can be, especially for scenarios involving highly connected data and complex relationships.

Key features and advantages

The following are some of the features and advantages of *Amazon Neptune:*

* **Graph database model:** *Amazon Neptune* is purpose-built for graph data. It supports both property graph and *RDF* graph models, making it flexible for various use cases. Property graph models[[70]](#footnote-70) are often used for highly connected data with complex relationships, while RDF graph models are suitable for representing and querying semantic data[[71]](#footnote-71).
* **Highly available and scalable:** Neptune provides high availability through multiple AZs, ensuring redundancy and automatic failover. It can also scale out[[72]](#footnote-72) to accommodate growing workloads[[73]](#footnote-73).
* **Support for multiple Query languages:** Neptune supports popular graph query languages like *SPARQL[[74]](#footnote-74)* and *Apache[[75]](#footnote-75) TinkerPop Gremlin*, allowing you to choose the query language that best suits your application[[76]](#footnote-76).
* **Security:** The service offers data encryption at rest and in transit. It integrates with AWS *IAM* for access control and *Amazon VPC* for network isolation[[77]](#footnote-77).

Use cases

*Amazon Neptune* is well-suited for applications requiring complex relationship modeling and querying. It is commonly used for social networks, fraud detection, recommendation engines, knowledge graphs, and life sciences research[[78]](#footnote-78).

Amazon RDS

*Amazon RDS* is a managed database service that simplifies the setup, operation, and scaling of relational databases. It supports multiple database engines, including *MySQL, PostgreSQL, SQL Server, MariaDB,* and *Oracle*. *Amazon RDS* takes care of routine database tasks, allowing you to focus on your application instead of database management.

In this section, we will discuss about *Amazon RDS* and explore its key features, the flexibility it offers in terms of database engines, automated management tasks, scalability, and high availability. Understanding Amazon RDS will help you make informed decisions regarding the database infrastructure for your applications, ensuring reliability and performance.

Key features and advantages

The following are some of the features and advantages of *Amazon RDS:*

* **Database engine flexibility:** Amazon RDS offers a choice of database engines, making it suitable for a wide range of applications[[79]](#footnote-79). You can choose from *MySQL, PostgreSQL, SQL Server, MariaDB,* and *Oracle[[80]](#footnote-80).*
* **Automated backups and software patching:** RDS automatically performs database backups, enabling point-in-time recovery. It also handles software patching, including critical security updates[[81]](#footnote-81).
* **Scalability:** Amazon RDS allows you to easily scale your database up or down based on your application's demands[[82]](#footnote-82). This ensures that your database can handle traffic spikes and grow with your business[[83]](#footnote-83).
* **High availability:** RDS offers high availability with automated failover to a standby instance in case of a hardware failure. This helps ensure that your application remains accessible even in the face of infrastructure issues[[84]](#footnote-84).

Use cases

*Amazon RDS* is suitable for various use cases, including web applications, mobile apps, e-commerce platforms, and content management systems[[85]](#footnote-85).

Amazon Redshift

*Amazon Redshift* is a fully managed data warehousing service that offers fast query performance using the SQL language. It is designed for analytics and business intelligence workloads, allowing you to analyze vast datasets quickly. Redshift can be an asset for organizations looking to make data-driven decisions.

In this section, we will explore Amazon Redshift, its columnar storage, massive parallel processing capabilities[[86]](#footnote-86), integration with BI tools, and its role in data warehousing and analytics. Understanding *Amazon Redshift's* strengths will help you harness the power of data for your analytical workloads and data-driven decision-making processes.

Key features and advantages

The following are some of the features and advantages of *Amazon Redshift:*

* **Columnar storage:** Amazon *Redshift* uses a columnar storage format, which is highly efficient for analytical queries. This enables rapid data retrieval and aggregation, making it well-suited for complex analytical tasks[[87]](#footnote-87).
* **Massively Parallel Processing (MPP):** *Redshift* employs MPP architecture to distribute and parallelize queries across multiple nodes, ensuring quick query execution[[88]](#footnote-88), even with large datasets[[89]](#footnote-89).
* **Integration with BI tools:** Redshift integrates seamlessly with popular **business intelligence** (**BI**) tools such as *Tableau, Power BI,* and *Amazon QuickSight*, enabling easy visualization and reporting[[90]](#footnote-90).
* **Data lake integration:** *Redshift Spectrum* allows you to query data in your *Amazon S3* data lake directly from Redshift, providing a unified view of your data[[91]](#footnote-91).

Use cases

*Amazon Redshift* is suitable for data warehousing and analytics, including ad hoc analysis, data exploration, and complex queries[[92]](#footnote-92). It is commonly used in industries like finance, healthcare, and e-commerce[[93]](#footnote-93).

Amazon Timestream

*Amazon Timestream* is a fully managed, serverless time-series database designed to handle high volumes of data from various sources with millisecond resolution. It is a valuable addition to the AWS database offerings, catering specifically to applications and workloads that depend on time-series data, such as *IoT, DevOps*, and *industrial telemetry*.

In this section, we explore *Amazon Timestream*, highlighting its serverless architecture, high-resolution data capabilities, built-in analytics functions[[94]](#footnote-94), and seamless integration with other AWS services. Understanding *Amazon Timestream's* role in handling time-series data efficiently[[95]](#footnote-95) can significantly enhance your ability to monitor and analyze data with high precision, benefiting applications across various domains.

Key features and advantages

The following are some of the features and advantages of *Amazon Timestream:*

* **Serverless and scalable:** *Amazon Timestream* is serverless, meaning there is no need to provide or manage servers. It automatically scales to handle fluctuating workloads, making it cost-effective and low maintenance[[96]](#footnote-96).
* **High resolution:** Timestream can capture and store data with millisecond precision, essential for time-series data, ensuring that you can monitor and analyze data with high accuracy[[97]](#footnote-97).
* **Built-in analytics:** It includes built-in analytical functions for real-time data processing, including interpolation, smoothing, and aggregation, simplifying data analysis[[98]](#footnote-98).
* **Integration with AWS services:** *Timestream* can seamlessly integrate with other AWS services, including *IoT Core, Lambda,* and *Quicksight*, to build end-to-end IoT and data analytics solutions[[99]](#footnote-99).

Use cases

*Amazon Timestream* is an excellent choice for various applications, such as tracking industrial machinery performance, monitoring IoT devices[[100]](#footnote-100), and analyzing log data. It is particularly valuable in scenarios where time-series data needs to be efficiently collected[[101]](#footnote-101), stored, and analyzed.

Conclusion

In this chapter, we discussed AWS databases extensively, exploring a multitude of services that cater to diverse use cases and business needs. From the foundational principles of database management to specialized databases designed for unique requirements, AWS offers a comprehensive suite of solutions that empower organizations to store, manage, and leverage their data with unprecedented efficiency and scalability.

Key takeaways:

* **Diverse database types:** AWS provides a rich array of database options, including relational databases like *Amazon RDS, NoSQL* databases like *Amazon DynamoDB*[[102]](#footnote-102), and purpose-built databases like *Amazon Timestream* and *Amazon Redshift*. This diversity ensures that regardless of your application's demands, there is a suitable database solution available[[103]](#footnote-103).
* **Managed services:** The managed nature of AWS databases alleviates the burden of routine administrative tasks, allowing you to focus on building applications and deriving value from your data. *Amazon RDS, Amazon DynamoDB[[104]](#footnote-104),* and *Amazon Aurora* are excellent examples of such services[[105]](#footnote-105).
* **Scalability and flexibility:** Most AWS databases are designed to scale horizontally or vertically, adapting to evolving workloads effortlessly. This elasticity is crucial for applications that experience varying levels of demand[[106]](#footnote-106).
* **Security and compliance:** AWS places a strong emphasis on the security and compliance of its databases. Services like *Amazon RDS* and *Amazon DocumentDB* offer robust security features, while AWS Artifact provides compliance-related documentation[[107]](#footnote-107).
* **Innovative database technologies:** AWS consistently introduces innovative database technologies like *Amazon Timestream* for time-series data and *Amazon Neptune* for graph databases. These offerings empower businesses to explore new horizons in data management and analysis[[108]](#footnote-108).

In conclusion, the AWS database ecosystem is a testament to the power of cloud computing in transforming data management. It offers a wide spectrum of database services, each tailored to meet specific needs, from mission-critical applications requiring high availability to analytical workloads demanding high-performance processing[[109]](#footnote-109). By combining the knowledge gained in this chapter with your specific use case requirements, you can access the full potential of AWS databases.

In the next chapter, *Developer Tools and DevOps,* we will discuss about the AWS core tools for developers such as **Software Development Kits** (**SDKs**), code editors, and DevOps. Moreover, we will discuss the **machine learning** (**ML**) best practices, abstractions to improve software development, software deployment, and operations.

1. Amazon Web Services, Inc. (2023). [Amazon RDS](https://aws.amazon.com/rds/). AWS Documentation. [↑](#footnote-ref-1)
2. Amazon Web Services, Inc. (2023). [Amazon Aurora](https://aws.amazon.com/rds/aurora/). AWS Documentation. [↑](#footnote-ref-2)
3. Amazon Web Services, Inc. (2023). [Amazon DynamoDB](https://aws.amazon.com/dynamodb/). AWS Documentation. [↑](#footnote-ref-3)
4. Amazon Web Services, Inc. (2023). [Amazon DocumentDB](https://aws.amazon.com/documentdb/). AWS Documentation. [↑](#footnote-ref-4)
5. Amazon Web Services, Inc. (2023). [Amazon ElastiCache](https://aws.amazon.com/elasticache/). AWS Documentation. [↑](#footnote-ref-5)
6. Amazon Web Services, Inc. (2023). [Amazon Redshift](https://aws.amazon.com/redshift/). AWS Documentation. [↑](#footnote-ref-6)
7. Amazon Web Services, Inc. (2023). [Amazon Timestream](https://aws.amazon.com/timestream/). AWS Documentation. [↑](#footnote-ref-7)
8. Amazon Web Services, Inc. (2023). [Amazon Neptune](https://aws.amazon.com/neptune/). AWS Documentation. [↑](#footnote-ref-8)
9. Amazon Web Services, Inc. (2023). [AWS Business Requirements](https://docs.aws.amazon.com/wellarchitected/latest/sap-lens/best-practice-17-3.html). AWS Documentation. [↑](#footnote-ref-9)
10. Amazon Web Services, Inc. (2023). [Amazon RDS](https://aws.amazon.com/rds/). AWS Documentation. [↑](#footnote-ref-10)
11. Amazon Web Services, Inc. (2023). [Amazon DynamoDB](https://aws.amazon.com/dynamodb/). AWS Documentation. [↑](#footnote-ref-11)
12. Amazon Web Services, Inc. (2023). [Amazon Redshift](https://aws.amazon.com/redshift/). AWS Documentation. [↑](#footnote-ref-12)
13. Amazon Web Services, Inc. (2023). [Amazon Timestream](https://aws.amazon.com/timestream/). AWS Documentation. [↑](#footnote-ref-13)
14. Amazon Web Services, Inc. (2023). [Amazon Neptune](https://aws.amazon.com/neptune/). AWS Documentation. [↑](#footnote-ref-14)
15. Amazon Web Services, Inc. (2023). [Security on AWS](https://aws.amazon.com/security/). AWS Documentation. [↑](#footnote-ref-15)
16. Amazon Web Services, Inc. (2023). [Availability and durability](https://aws.amazon.com/about-aws/global-infrastructure/availability/). AWS Documentation. [↑](#footnote-ref-16)
17. Amazon Web Services, Inc. (2023). [Choose a data store](https://aws.amazon.com/getting-started/data-store-options/). AWS Documentation. [↑](#footnote-ref-17)
18. Amazon Web Services, Inc. (2023). [Amazon Aurora](https://aws.amazon.com/rds/aurora/). AWS Documentation. [↑](#footnote-ref-18)
19. Amazon Web Services, Inc. (2023). [Amazon Aurora Global Database](https://aws.amazon.com/rds/aurora/global-database/). AWS Documentation. [↑](#footnote-ref-19)
20. Faleiro, J., Van Renesse, R., Rodrigues, R. (2016). Highly Available Transactions: Virtues and Limitations. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16). [↑](#footnote-ref-20)
21. Amazon Web Services, Inc. (2023). [Choosing an AWS database service](https://aws.amazon.com/getting-started/decision-guides/databases-on-aws-how-to-choose/). AWS Documentation. [↑](#footnote-ref-21)
22. Amazon Web Services, Inc. (2023). [Amazon Aurora Customer Success Stories](https://aws.amazon.com/solutions/case-studies/). AWS Customer Success Stories. [↑](#footnote-ref-22)
23. Werner Vogels. (2019). *AWS Launches Amazon DocumentDB (with MongoDB Compatibility)*. AWS News Blog. <https://aws.amazon.com/blogs/aws>. [↑](#footnote-ref-23)
24. AWS Training & Certification. (2023). *Amazon DocumentDB (with MongoDB Compatibility) - AWS Certified Database Speciality Exam Prep (DBS-C01)*. <https://www.aws.training/Details/Curriculum?id=20685>. [↑](#footnote-ref-24)
25. AWS Documentation. (2023). *Amazon DocumentDB*. <https://aws.amazon.com/documentdb/>. [↑](#footnote-ref-25)
26. Fok, D. (2013). Cloud Databases for Scalable, Elastic, and Highly Available Web Applications. *2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* [↑](#footnote-ref-26)
27. Tekli, J., & Mansour, E. (2015). Performance Analysis of NoSQL Databases. *2015 IEEE/ACM 7th International Workshop on Big Data Technologies and Applications (BIDATA)*. [IEEE Xplore](https://ieeexplore.ieee.org/document/7343744). [↑](#footnote-ref-27)
28. Grolinger, K., et al. (2013). To Use or Not to Use: Benchmarking NoSQL Data Stores. *2013 IEEE 6th International Conference on Cloud Computing* [↑](#footnote-ref-28)
29. AWS Online Tech Talks. (2023). *AWS Database Freedom*. <https://aws.amazon.com/events/twitch/aws-on-air-reinvent>. [↑](#footnote-ref-29)
30. *Benchmarking and Simulation of High-Performance Computer Systems (PMBS)*. [IEEE Xplore](https://ieeexplore.ieee.org/document/8284090). [↑](#footnote-ref-30)
31. Pătrașcu, A., Pârv, B., & Avasalcai, C. (2017). Scalability Analysis of NoSQL Databases for Big Data. *2017 IEEE/ACM 9th International Workshop on Performance Modeling, Benchmarking and Simulation of High-Performance Computer Systems (PMBS)*. [IEEE Xplore](https://ieeexplore.ieee.org/document/8284090). [↑](#footnote-ref-31)
32. Vogels, W. (2009). Eventually consistent. ACM Queue, 7(6), 14-18. [↑](#footnote-ref-32)
33. Amazon Web Services. (2023). Amazon DynamoDB Documentation. <https://docs.aws.amazon.com/amazondynamodb>. [↑](#footnote-ref-33)
34. Sivasubramanian, S. (2019). Amazon DynamoDB: a seamlessly scalable non-relational database service. Communications of the ACM, 62(12), 64-71. [↑](#footnote-ref-34)
35. Amazon Web Services. (2023). Amazon DynamoDB Developer Guide. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide>. [↑](#footnote-ref-35)
36. Amazon Web Services. (2023). Best Practices for Designing and Architecting with DynamoDB. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-practices>. [↑](#footnote-ref-36)
37. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., ... & Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. ACM SIGOPS Operating Systems Review, 41(6), 205-220. [↑](#footnote-ref-37)
38. Amazon Web Services. (2023). Amazon DynamoDB Documentation. <https://docs.aws.amazon.com/amazondynamodb>. [↑](#footnote-ref-38)
39. Amazon Web Services. (2023). Amazon DynamoDB. <https://aws.amazon.com/dynamodb/> [↑](#footnote-ref-39)
40. Ananthanarayanan, R., Doug, B., Eshghi, K., Harizopoulos, S., Li, X., MacNair, E., ... & Zhang, M. (2010). A comparison of high-level storage systems. ACM SIGMOD Record, 39(1), 16-21. [↑](#footnote-ref-40)
41. Cheng, L., Difallah, D. E., Weissman, J. B., & Gokhale, A. (2018). Benchmarking cloud-based NoSQL databases. IEEE Transactions on Knowledge and Data Engineering, 30(7), 1271-1285. [↑](#footnote-ref-41)
42. Strauch, C., & Paradies, M. (2019). A survey of distributed database management systems. ACM Computing Surveys (CSUR), 52(6), 1-33. [↑](#footnote-ref-42)
43. Fowler, M. (2013). NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Pearson. [↑](#footnote-ref-43)
44. Carlsson, N., Jacobsson, C., & Karlsson, L. (2019). Using Redis for Persistent Storage. In 2019 IEEE/ACM 5th International Workshop on Container Technologies and Container Clouds (CT\textbackslash{}&CC) (pp. 18-23). [↑](#footnote-ref-44)
45. Tilmann, A., & Kotzyba, G. (2015). An Evaluation of Distributed In-Memory Databases. In Proceedings of the 16th International Middleware Conference (pp. 46-57). [↑](#footnote-ref-45)
46. Amazon Web Services. (2023). Amazon ElastiCache: In-Memory Caching Service. <https://aws.amazon.com/elasticache/>. [↑](#footnote-ref-46)
47. Kaur, H., & Saini, R. (2016). Performance Analysis of Memcached and Redis for Real-Time Big Data Processing. Procedia Computer Science, 78, 390-395. [↑](#footnote-ref-47)
48. Amazon Web Services. (2023). Amazon ElastiCache Documentation. <https://docs.aws.amazon.com/AmazonElastiCache>. [↑](#footnote-ref-48)
49. Amazon Web Services. (2023). Amazon ElastiCache Auto Discovery. <https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/AutoDiscovery.html>. [↑](#footnote-ref-49)
50. Amazon Web Services. (2023d). Amazon ElastiCache for Redis: How Amazon ElastiCache Works. <https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/HowElasicacheWorks.html>. [↑](#footnote-ref-50)
51. Wohler, D., & McCall, D. (2015). Measuring Redis Throughput and Latency. In Proceedings of the Symposium on Cloud Computing (pp. 20-23). [↑](#footnote-ref-51)
52. Amazon Web Services. (2023e). Amazon ElastiCache Use Cases. <https://aws.amazon.com/elasticache/use-cases/>. [↑](#footnote-ref-52)
53. Aba, A., Abel, R., & Davidson, B. (2019). Evaluating Apache Cassandra: Over a Decade of Evolution. ACM Computing Surveys, 51(5), 1-33. [↑](#footnote-ref-53)
54. Cieslewicz, J., Mittal, A., & Varghese, G. (2014). COPS: A Scalable and Extensible Middleware for Cloud-Oriented Cassandra. In Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (pp. 12-23). [↑](#footnote-ref-54)
55. Kannan, S., Sivasubramanian, S., & Schaeffer, C. (2018). Amazon DynamoDB: A Distributed and Highly Available Database. In 2018 ACM/IFIP/USENIX Middleware (pp. 19-33). [↑](#footnote-ref-55)
56. Amazon Web Services. (2023). Amazon Keyspaces (for Apache Cassandra). <https://aws.amazon.com/keyspaces/>. [↑](#footnote-ref-56)
57. Lakshman, A., & Malik, P. (2010). Cassandra: A Decentralized Structured Storage System. ACM SIGOPS Operating Systems Review, 44(2), 35-40. [↑](#footnote-ref-57)
58. Amazon Web Services. (2023). Amazon Keyspaces (for Apache Cassandra) Documentation. <https://docs.aws.amazon.com/keyspaces>. [↑](#footnote-ref-58)
59. Amazon Web Services. (2023). Apache Cassandra-Compatible Data API. <https://docs.aws.amazon.com/keyspaces/latest/devguide/cql.html>. [↑](#footnote-ref-59)
60. Amazon Web Services. (2023). Amazon Keyspaces (for Apache Cassandra) Security. <https://aws.amazon.com/keyspaces/security/>. [↑](#footnote-ref-60)
61. Amazon Web Services. (2023). Amazon Keyspaces (for Apache Cassandra) Use Cases. <https://aws.amazon.com/keyspaces/use-cases/>. [↑](#footnote-ref-61)
62. Sallam, S. A., Zaki, S. M., & Ramadan, R. A. (2016). A comparative study of NoSQL databases. In Proceedings of the International Conference on Computer, Control, Informatics and its Applications (IC3INA) (pp. 257-262). IEEE. [↑](#footnote-ref-62)
63. Joshi, G., Deshpande, A., & Bhopale, S. (2017). NoSQL Databases: A Survey. In 2017 7th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 150-155). IEEE. [↑](#footnote-ref-63)
64. Amazon Web Services. (2023). Amazon MemoryDB for Redis. <https://aws.amazon.com/memorydb/>. [↑](#footnote-ref-64)
65. Amazon Web Services. (2023b). Amazon MemoryDB for Redis: High Availability. <https://aws.amazon.com/memorydb/high-availability/>. [↑](#footnote-ref-65)
66. Amazon Web Services. (2023c). Amazon MemoryDB for Redis: Performance. <https://aws.amazon.com/memorydb/performance/>. [↑](#footnote-ref-66)
67. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., ... & Pratt, I. (2003). Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37(5), 164-177. [↑](#footnote-ref-67)
68. Amazon Web Services. (2023). Amazon MemoryDB for Redis: Security. <https://aws.amazon.com/memorydb/security/>. [↑](#footnote-ref-68)
69. Amazon Web Services. (2023). Amazon MemoryDB for Redis: Use Cases. <https://aws.amazon.com/memorydb/use-cases/>. [↑](#footnote-ref-69)
70. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J. L., & Vrgoc, D. (2018). Foundations of modern query languages for graph databases. ACM Computing Surveys (CSUR), 51(5), 1-39. [↑](#footnote-ref-70)
71. Amazon Web Services. (2023). Amazon Neptune: Graph Database. <https://aws.amazon.com/neptune/>. [↑](#footnote-ref-71)
72. Sequeda, J. F., Gonçalves, M. A., Aluísio, S. M., & Fortes, R. P. (2015). RDF4Graph: a scalable middleware for RDF data management and OLAP querying. Journal of Web Semantics, 34, 11-29. [↑](#footnote-ref-72)
73. Amazon Web Services. (2023). Amazon Neptune: High Availability. <https://aws.amazon.com/neptune/high-availability/>. [↑](#footnote-ref-73)
74. Brzozowski, M. J., Lin, M., & Wood, P. T. (2009). Evaluating SPARQL queries over an RDF dataset. In International Conference on Database Theory (pp. 49-64). Springer. [↑](#footnote-ref-74)
75. Neo4j. (2019). Graph Algorithms: Practical Examples in Apache Spark and Neo4j. O'Reilly Media. [↑](#footnote-ref-75)
76. Amazon Web Services. (2023). Amazon Neptune: Multiple Query Languages. <https://aws.amazon.com/neptune/query/>. [↑](#footnote-ref-76)
77. Amazon Web Services. (2023). Amazon Neptune: Security. <https://aws.amazon.com/neptune/security/>. [↑](#footnote-ref-77)
78. Amazon Web Services. (2023). Amazon Neptune: Use Cases. <https://aws.amazon.com/neptune/use-cases/>. [↑](#footnote-ref-78)
79. Ahmed, M., Hu, J., & Hu, H. (2012). An empirical study of the reliability of UNIX utilities. In 2012 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 1-12). IEEE. [↑](#footnote-ref-79)
80. Amazon Web Services. (2023). Amazon RDS: Relational Database Service. <https://aws.amazon.com/rds/>. [↑](#footnote-ref-80)
81. Amazon Web Services. (2023). Amazon RDS: Automated Backups and Software Patching. <https://aws.amazon.com/rds/features/automated-backups/>. [↑](#footnote-ref-81)
82. Mohapatra, P., Chhetri, M. B., & Panda, S. (2012). Performance comparison of database access using AWS RDS and traditional web hosting service. International Journal of Computer Applications, 55(13), 22-29. [↑](#footnote-ref-82)
83. Amazon Web Services. (2023). Amazon RDS: Scalability. <https://aws.amazon.com/rds/features/scalability/>. [↑](#footnote-ref-83)
84. Amazon Web Services. (2023). Amazon RDS: High Availability. <https://aws.amazon.com/rds/features/high-availability/>. [↑](#footnote-ref-84)
85. Amazon Web Services. (2023e). Amazon RDS: Use Cases. <https://aws.amazon.com/rds/use-cases/>. [↑](#footnote-ref-85)
86. Rolia, J., Kolettis, N., Braun, U., & Feldmann, A. (1998). Performance guarantees in communication-centric systems. ACM Transactions on Computer Systems (TOCS), 16(2), 216-247. [↑](#footnote-ref-86)
87. Amazon Web Services. (2023). Amazon Redshift: Columnar Storage. <https://aws.amazon.com/redshift/>. [↑](#footnote-ref-87)
88. Moens, S., Ponnet, K., & Delaigle, J. (2011). A flexible online analytical processing architecture for cloud computing. Journal of Computer and System Sciences, 77(1), 49-63. [↑](#footnote-ref-88)
89. Amazon Web Services. (2023). Amazon Redshift: Massively Parallel Processing. <https://aws.amazon.com/redshift/features/mpp/>. [↑](#footnote-ref-89)
90. Amazon Web Services. (2023c). Amazon Redshift: Integration with BI Tools. <https://aws.amazon.com/redshift/features/integration-bi-tools/>. [↑](#footnote-ref-90)
91. Amazon Web Services. (2023d). Amazon Redshift Spectrum. <https://aws.amazon.com/redshift/features/spectrum/>. [↑](#footnote-ref-91)
92. Mishne, G., Beech, D., Bouganim, L., & Novikov, E. (2005). Relational databases for query caches: A case study. In Proceedings of the 31st international conference on Very large databases (pp. 58-69). [↑](#footnote-ref-92)
93. Amazon Web Services. (2023e). Amazon RDS: Use Cases. <https://aws.amazon.com/rds/use-cases/>. [↑](#footnote-ref-93)
94. Zaslavsky, A., Perera, C., & Georgakopoulos, D. (2012). Sensor cloud: a cloud of virtualized sensors. In 2012 19th IEEE International Conference on Web Services (pp. 407-414). IEEE. [↑](#footnote-ref-94)
95. Hwang, W. J., Li, K. W., & Lee, J. M. (2018). A Data Stream Management System for Real-Time Monitoring of Maritime Environmental Data. Journal of Marine Science and Engineering, 6(4), 118. [↑](#footnote-ref-95)
96. Amazon Web Services. (2023a). Amazon Timestream: Serverless. <https://aws.amazon.com/timestream/features/serverless/>. [↑](#footnote-ref-96)
97. Amazon Web Services. (2023b). Amazon Timestream: High Resolution. <https://aws.amazon.com/timestream/features/high-resolution/>. [↑](#footnote-ref-97)
98. Amazon Web Services. (2023c). Amazon Timestream: Built-In Analytics. <https://aws.amazon.com/timestream/features/built-in-analytics/>. [↑](#footnote-ref-98)
99. Amazon Web Services. (2023d). Amazon Timestream: Integration with AWS Services. <https://aws.amazon.com/timestream/features/integration/>. [↑](#footnote-ref-99)
100. Deshpande, A., & Thaker, V. (2017). IoT Data Analytics at the Edge. In Edge Analytics on the Internet of Things (pp. 3-25). Springer. [↑](#footnote-ref-100)
101. Golab, L., & Özsu, M. T. (2003). Data Stream Management: Processing High-Speed Data Streams. In Proceedings of the Thirtieth international conference on Very large databases-Volume 30 (pp. 937-938). [↑](#footnote-ref-101)
102. Cattell, R. (2010). Scalable SQL and NoSQL data stores. ACM SIGMOD Record, 39(4), 12-27. [↑](#footnote-ref-102)
103. Stonebraker, M. (2010). SQL databases v. NoSQL databases. Communications of the ACM, 53(4), 10-11. [↑](#footnote-ref-103)
104. Amazon Web Services. (2023b). Amazon DynamoDB. <https://aws.amazon.com/dynamodb/>. [↑](#footnote-ref-104)
105. Amazon Web Services. (2023). Amazon RDS. <https://aws.amazon.com/rds/>. [↑](#footnote-ref-105)
106. Agrawal, D., Das, S., & El Abbadi, A. (2007). Big data and cloud computing: current state and future opportunities. Ercim News, 2011(84), 28-31 [↑](#footnote-ref-106)
107. Amazon Web Services. (2023). AWS Security and Compliance. <https://aws.amazon.com/security/>. [↑](#footnote-ref-107)
108. Amazon Web Services. (2023). Amazon Timestream. <https://aws.amazon.com/timestream/>. [↑](#footnote-ref-108)
109. Papadopoulos, A. V., & Maglaris, V. (2013). Towards big data benchmarks. In Proceedings of the 2nd ACM symposium on Cloud computing (pp. 13-24). [↑](#footnote-ref-109)