Lesson 2
What Is Data Partitioning? And How It Leads To Efficient Data Processing

Adapted from: https://airbyte.com/data-engineering-resources/what-is-data-partitioning

Discussion questions before the reading.

- How does data partitioning contribute to efficient data processing in distributed systems?
- What are some common strategies for data partitioning, and how do they vary based on the characteristics of the data and the processing requirements?
- Can you discuss some real-world examples where effective data partitioning significantly improved the performance and scalability of data processing pipelines or systems?

Key vocabulary for the reading activity #2.

Data Partitioning: The process of dividing a large dataset into smaller, more manageable subsets called partitions or shards.

Horizontal Partitioning: Also known as row partitioning, it involves dividing the database schema into multiple partitions based on rows or records.

Vertical Partitioning: Also known as column partitioning, it involves dividing the database schema based on columns or attributes.

Functional Partitioning: Involves dividing the database schema based on specific operational requirements or processing needs.

Partition Key: An attribute or criterion used to divide a dataset into partitions or subsets.

Scalability: The ability of a system to handle increasing loads by adding more nodes or resources.

Data Localization: Distributing data based on criteria like geographic regions or functional requirements to optimize data access.

COLOMBIA POTENCIA DE LA VIDA

TIC

Reading activity #2: "What Is Data Partitioning? And How It Leads To Efficient Data Processing"

What is Data Partitioning?

Data partitioning, also known as data sharding or data segmentation, is the process of dividing a large dataset into smaller, more manageable subsets called partitions or shards. Data partitioning criteria and the partitioning strategy decide how the dataset is divided.

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Each partition contains a subset of the data, and these subsets are typically distributed across multiple servers, nodes, or systems.

The main goals of data partitioning are to improve performance, scalability, and fault tolerance. By distributing data across multiple servers, engineers can leverage parallel processing, reduce the load on individual systems, and improve the utilization of computing resources.

Data partitioning is commonly used in database management systems (DBMS), distributed file systems, and big data processing frameworks.

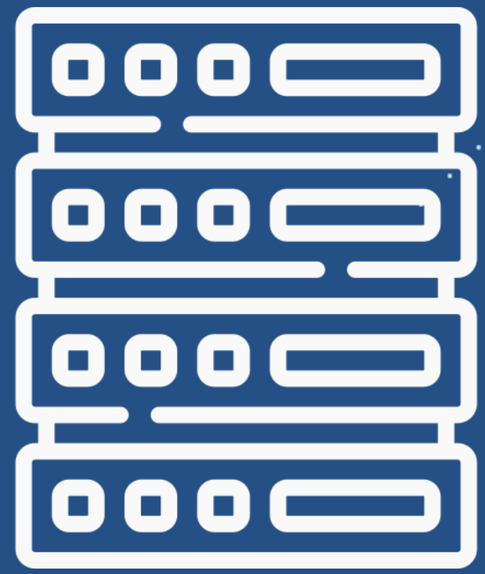
COLOMBIA POTENCIA DE LA VIDA

TIC

What are the different types of data partitioning methods?

Data partitioning can be broadly categorized into three main methods - horizontal partitioning, vertical partitioning, and functional partitioning. These methods can be combined to match different use cases.

The choice of database partitioning method(s) depends on factors like data size, access patterns, processing requirements, and system architecture.



TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

Universidad Tecnológica de Pereira

Let’s take a closer look at each of them:

Horizontal Partitioning (or Row Partitioning)

In horizontal partitioning, database schema is divided into multiple partitions based on rows or records. Each partition contains a subset of rows with a common attribute or value.

This method is used when the dataset is too large to fit on a single system or when there is a need for distributed processing of data. It is useful when different subsets of data are accessed or updated independently.

There are many different strategies for horizontal partitioning, including range partitioning, list partitioning, hash partitioning, and composite partitioning.

For example, in a customer database, the data can be horizontally partitioned based on geographical regions, with each partition containing customer records specific to a particular area.

Vertical Partitioning (or Column Partitioning)

In vertical partitioning, database schema is divided based on columns or attributes. Each partition contains a subset of columns for each row or record.

This method is used to optimize storage and improve performance by grouping related columns in the same partition. It can speed up data retrieval by reducing the amount of data read from storage.

It is ideal for when different sets of attributes are frequently accessed together. A common use case of vertical partitioning is to separate static, slow-moving data from more fast-paced, dynamic data.

For example, in a product database, the data can be vertically partitioned into two partitions, one containing frequently accessed attributes, like product name and price, and the other containing rarely accessed attributes, like detailed product descriptions and images.

Functional Partitioning

Functional partitioning involves dividing database schema based on specific operational requirements or processing needs. Data partitions are created based on the operations or functions performed on the data.

This method is used in complex systems or distributed processing frameworks where different functions or modules require access to specific subsets of data.

Functional partitions allow for better isolation and encapsulation of data, ensuring that each module operates on its designated data subset.

For example, in a social media platform, the user profile data may be partitioned separately from the posts and comments data to enable independent processing.

COLOMBIA POTENCIA DE LA VIDA

TIC

- What are the benefits of data partitioning?
- When to use Data Partitioning?
- Real-world use cases for data partitioning
- What are Data Partitioning Strategies?
- What is the Partition key?
- Strategies for horizontal partitioning
- Strategies for vertical partitioning
- Strategies for functional partitioning
- Best practices for Data Partitioning

Presionar cada tema para ver si contenido

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

Universidad Tecnológica de Pereira

What are the benefits of data partitioning?

Data partitioning offers eight critical advantages for data processing:

- Improved performance: By dividing a large dataset into smaller partitions, data processing tasks can be performed in parallel across multiple nodes or systems. This parallelization leads to faster processing times and improved performance overall. Moreover, partitions eliminate the need to query the entire database. Queries can be run on smaller components, leading to faster results.
- Enhanced scalability: Data partitioning enables horizontal scaling by distributing data across multiple nodes or systems. Here, data engineers can add more nodes to accommodate increasing loads. This scalability ensures the system can handle increased processing demands without any performance bottlenecks.
- Increased concurrency: Different partitions enable concurrent access to the dataset. Multiple operations, such as reading, writing, and analysis, can be performed simultaneously on different partitions, enabling higher throughput and reducing processing delays.
- Efficient resource utilization: Data engineers can better utilize computing resources using partitioning. They can assign each partition to a specific node or system, enabling efficient use of memory, storage space, and processing power. It also helps distribute the data and workload evenly, preventing any single node from becoming overloaded.

- Improved Fault Tolerance: Data partitioning provides built-in fault tolerance capabilities. Data can be replicated or redistributed across other nodes, ensuring the entire database architecture is available and minimizing the impact of system failures. Replication of partitions also provides data redundancy, reducing the risk of data loss.
- Data isolation and security: By segregating data into multiple partitions, access controls and security measures can be applied at the partition level. So, sensitive data can be stored in a separate partition and better protected using additional security controls.
- Simplified maintenance and operations: Partitioning can simplify maintenance tasks and system operations. It also streamlines data migration or archiving since every partition can be handled independently.
- Data localization: Distributing data based on criteria like geographic regions or functional requirements enables data localization. This can be beneficial for compliance with data privacy regulations or optimizing data access based on consumers' location or usage patterns.

When to use Data Partitioning?

Here are some situations where data partitioning is particularly useful:

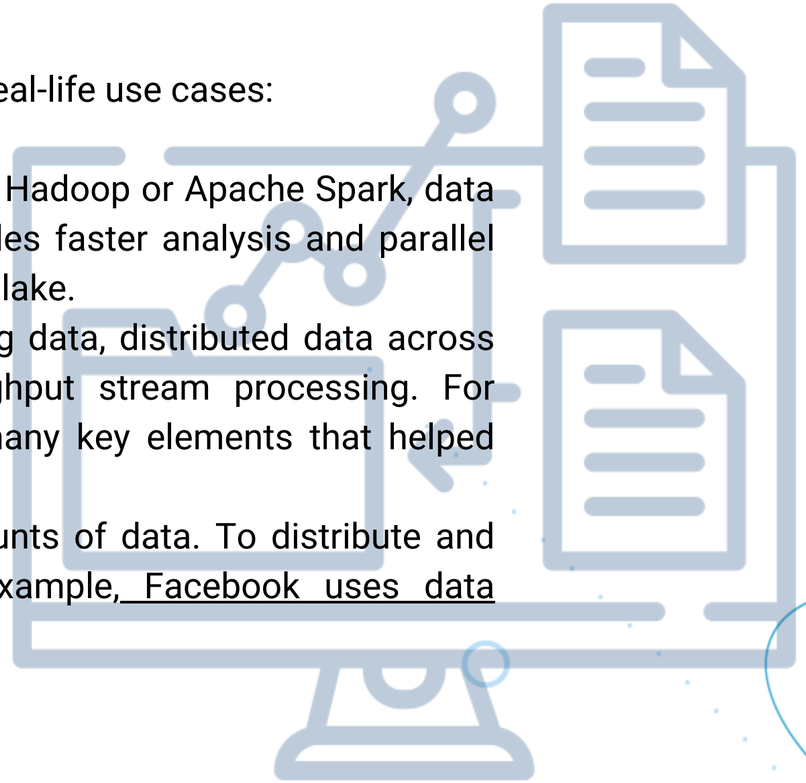
- Large-scale datasets: When dealing with large volumes of data that cannot fit into a single system's memory or storage space, data partitioning becomes essential.
- Distributed computing: In distributed architectures, where multiple servers or nodes work together to process data, data partitioning is crucial.
- Performance optimization: Data partitioning can be effective if your application requires faster data access or reduced query response times. Using partitions, engineers can focus processing resources on specific subsets of data.
- Load balancing: Data partitioning helps distribute the workload evenly across multiple systems or nodes, preventing any system from becoming overloaded.
- Data privacy and compliance: Data partitions help engineers ensure that data access for confidential or sensitive data is restricted based on specific compliance regulations or privacy policies.
- Separation of data access patterns: Dividing data based on access patterns can improve data locality, reduce data transfer across nodes, and enhance overall system efficiency.

It's important to note that data partitioning introduces additional complexity to the system architecture. Your data partitioning schemes must be designed for your application's specific requirements, scalability needs, and performance goals.

Real-world use cases for data partitioning

To illustrate how data partitioning is beneficial, here are some common real-life use cases:

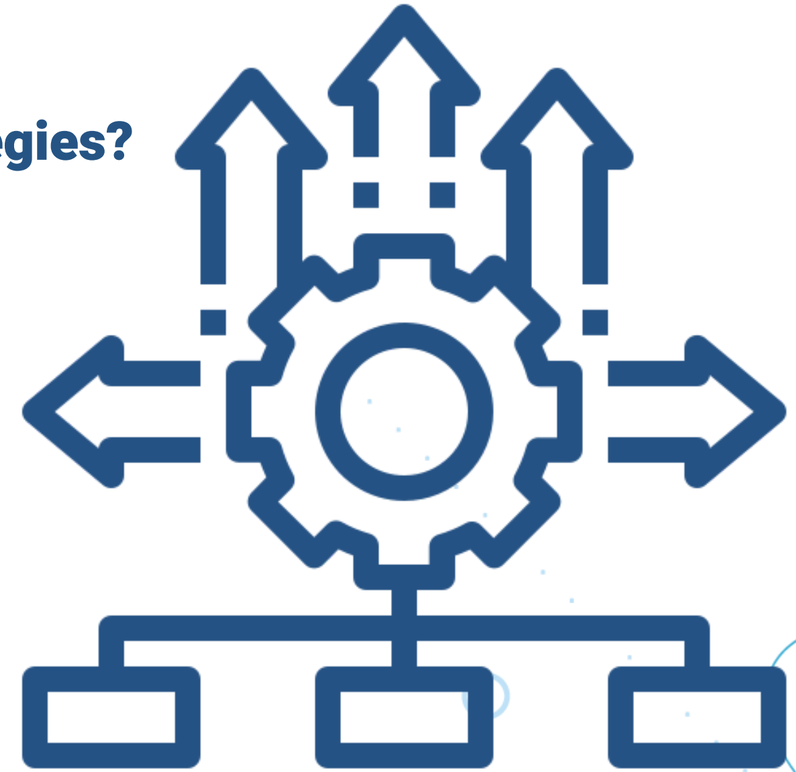
- Big data processing: In big data processing frameworks like Apache Hadoop or Apache Spark, data partitioning is essential for efficient distributed processing. It enables faster analysis and parallel execution of tasks. Uber uses data partitions in its transactional data lake.
- Real-time stream processing: When dealing with real-time streaming data, distributed data across partitions can be processed independently, enabling high-throughput stream processing. For example, upgrading its data partitioning techniques was one of many key elements that helped Netflix improve query performance by up to 60%.
- Social media analytics: Social media platforms generate vast amounts of data. To distribute and process this data efficiently, they use data partitioning. For example, Facebook uses data partitioning techniques to reduce its CPU consumption by half.



- Data warehousing: Data partitioning can make data warehouses more manageable and efficient. By dividing data based on key attributes, data teams can run queries and generate reports faster. Data can also be loaded or purged in a controlled manner. For example, Airbnb's upgraded data warehouse infrastructure uses data partitioning.
- Financial data processing: Partitioning can help efficiently process and analyze different groups of financial data for risk management, fraud detection, and regulatory compliance.
- E-commerce and retail: In e-commerce and retail applications, data partitioning can speed up transaction processing. For example, customer data can be partitioned based on geographical regions to ensure localized processing and minimize data transfer across systems.
- Geographic data analysis: Spatial data, such as geographic information systems (GIS) data, can be partitioned based on geographical regions, allowing for efficient spatial queries and analysis.

What are Data Partitioning Strategies?

There are many database partitioning strategies, each suited for different scenarios and requirements. Before we dive into the strategies, let’s first understand a foundational element - the partition key.



COLOMBIA POTENCIA DE LA VIDA

TIC

What is the Partition key?

A partition key is an attribute or criterion used to divide a dataset into partitions or subsets. It determines how data is distributed and organized within a partitioned system and is a part of data modeling.

When data is partitioned, the key is used to assign each record or data item to a specific partition. For example, in a distributed database, a partitioning key could be a customer ID, geographic region, or timestamp. Each record in the database is assigned to a partition based on the value of the key.

+INFO

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Here are six crucial factors to consider when choosing a partitioning key:

- High cardinality: Partition keys should have many distinct values to ensure an even data distribution across partitions. High cardinality helps prevent data skew and ensures balanced workloads.
- Query efficiency: Choosing a key frequently used in queries or filters improves query performance by reducing the amount of data that must be scanned across partitions.
- Data independence: The key should be independent of other attributes or data segments. This allows for the isolation and efficient processing of data within each partition without the need to access unrelated data.
- Scalability: The partitioning key must allow easy scalability as the dataset grows. It should enable the addition of new partitions without significant disruptions or reorganizing of existing data.
- Data distribution: The key should lead to balanced data distribution across partitions. Uneven data distribution can result in performance issues and inefficient resource utilization.
- Data integrity: The key must ensure data integrity and consistency within each partition. It should maintain the logical relationships and dependencies between data items.

Strategies for horizontal partitioning

Here are some common strategies for horizontal partitioning or sharding:

- Key partitioning: In key sharding, data items are distributed across partitions based on the value of the partitioning key.
- Range partitioning: In Range partitioning, datasets are divided into multiple partitions, each storing data that falls within a particular range or interval.
- Hash partitioning: Hash partitioning distributes data based on a hash function applied to a key attribute. The hash function generates a hash value for each data item, and the item is assigned to a partition based on the calculated hash value.
- List partitioning: In List partitioning, datasets are divided into partitions based on specific values or a list of discrete values. Each partition is assigned a list of values, and any data items matching those values are stored in the corresponding partition.
- Round-Robin partitioning: Round-robin sharding evenly distributes data items across partitions cyclically. Each data item is assigned to the next available partition in sequence.
- Composite partitioning: Composite sharding, also known as compound sharding, combines multiple partitioning methods or criteria. It involves combining partitioning keys or attributes to determine data placement across partitions.

Strategies for vertical partitioning

Here are five standard strategies for vertical partitioning:

- Normalization: Normalization is a widely used technique in database schema where data is organized into multiple related tables to eliminate redundancy and improve data integrity. Each table represents a vertical partition containing a subset of columns related to a specific entity or concept.
- Access frequency: In this vertical partitioning strategy, columns are split based on their access frequency or usage patterns. Frequently accessed columns critical for query execution are placed in one partition, while less frequently accessed columns are placed in another.
- Data sensitivity: In some cases, specific columns may contain sensitive or private information that requires stricter access controls. Vertical partitioning can separate sensitive columns into a dedicated partition or table.
- Data size: If a dataset contains a mix of large and small columns, vertical partitioning can separate the larger columns into a dedicated partition.
- Functionality: Columns can be split based on their functional dependencies or the specific tasks they support.

Strategies for functional partitioning

Here are six strategies for functional partitioning:

- Service-Based Partitioning: In this strategy, each service or component of a system is responsible for managing a data set related to its functionality.
- Module-Based Partitioning: In module-based partitioning, each module or subsystem is responsible for its own data partition, containing the data necessary for its functionality.
- Business Process Partitioning: In business process partitioning, each partition contains the data required for a specific business process or workflow.
- User or Customer-Based Partitioning: User-based partitioning involves splitting data based on different users or customers of a system. Each partition contains data specific to a particular user or customer.
- Geographic Partitioning: In geographic partitioning, every partition contains data specific to a particular geographic region.
- Compliance or Regulatory Partitioning: Engineers can partition data based on specific compliance requirements or regulatory standards.

Best practices for Data Partitioning

To create and implement an effective data partitioning strategy, you can use the following best practices :

- Understand data characteristics: Gain a deep understanding of your data, including its structure, access patterns, and relationships. Analyze the data distribution, size, and growth patterns to make informed decisions.
- Define clear partitioning criteria: Identify the appropriate partitioning key(s) to distribute data evenly, minimize data skew, and align with the most common query filters or join conditions.
- Consider scalability: Ensure that the partitioning strategy allows for easy addition or removal of partitions as the dataset grows or workloads increase.
- Balance data distribution: Aim for even data distribution across partitions to prevent hotspots and ensure optimal utilization of system resources.
- Optimize performance: Analyze the common query patterns and optimize the partitioning strategy to align with those patterns. Ensure frequently executed queries can leverage partition pruning and minimize data access across partitions.
- Implement monitoring and maintenance: Establish monitoring mechanisms to track the health and performance of the partitions. Regularly analyze data distribution, performance, and resource utilization to identify imbalances or bottlenecks.
- Plan for partitioning changes: Anticipate the possibility of changing partitioning requirements or data distribution patterns. Construct a flexible system design that allows easy modifications to the partitioning strategy, including adjusting partition boundaries.
- Documentation and communication: Document the rationale behind the chosen approach and any specific considerations. Communicate the partitioning strategy to the development and operations teams to ensure proper understanding and adherence.

COLOMBIA POTENCIA DE LA VIDA

TIC

Gap fill activity about the previous reading.

Para consolidar las respuestas ingrese al cuestionario online.

INICIO

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

Universidad Tecnológica de Pereira