Introduction

Hive Overview and Important Concepts

- Apache Hive is a data warehouse infrastructure built on top of Hadoop for data summarization, query, and analysis.
- Hive uses a SQL-like language called HiveQL to query large datasets stored in Hadoop Distributed File System (HDFS).
- It abstracts MapReduce complexity, enabling SQL users to perform big data analytics easily.
- Hive supports schema on read, allowing flexible data ingestion without upfront schema enforcement.
- Example Hive query to count records in a table:
- \`\`\`sql
- SELECT COUNT(*) FROM employee_data;
- \`\`\`

Comparison

RDBMS vs Hadoop (Hive)

RDBMS Characteristics

- Designed for structured data with fixed schema
- Supports ACID transactions ensuring data integrity
- Optimized for Online Transaction Processing (OLTP)
- Data stored in tables with relationships and constraints
- Uses SQL for complex queries and joins

Hive on Hadoop Characteristics

- Handles large-scale, semi-structured, and unstructured data
- Schema-on-read model allows flexible data ingestion
- Optimized for Online Analytical Processing (OLAP) and batch jobs
- Uses HiveQL, a SQL-like language for querying data
- Built on Hadoop ecosystem supporting distributed storage and processing

Integration

Importing Tables from RDBMS to Hive

1. Extract Data from RDBMS

Use JDBC connection to extract data from the relational database. Tools like Sqoop can automate this process by connecting to RDBMS and exporting data in a compatible format.

JDBC connection setup
Sqoop export/import commands
Sample query to fetch data from RDBMS

2. Create Hive Table

Define the Hive table schema matching the RDBMS table structure. Use HiveQL to create a new table, specifying columns and data types that correspond to the source data.

HiveQL CREATE TABLE statement
Schema definition file
Data type mapping guidelines

3. Load Data into Hive Table

Load the extracted data files into the Hive table using LOAD DATA command or by specifying external table location. Validate data integrity by running select queries.

LOAD DATA INPATH command
Data validation queries
Example Hive script to automate loading process

Importing data from RDBMS to Hive enables seamless integration for big data processing using familiar SQL-like queries.

Data Types

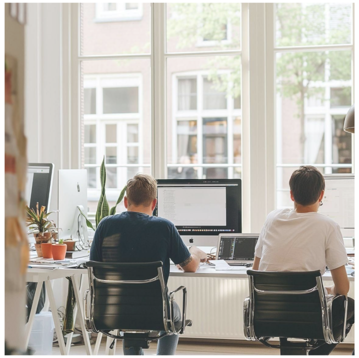
Types of Data

Structured Data

Organized in tabular formats with rows and columns, ideal for SQL queries and traditional databases.

Semi-structured Data

Includes JSON, XML, and Email formats that have some organizational properties but flexible schema.



Unstructured Data

Data without a predefined format such as logs, images, and videos, requiring special processing methods.

Data Frequency Types

Real Time (Streaming)	Data is processed instantly as it arrives, enabling immediate insights and actions. Common in monitoring systems, fraud detection, and live analytics. Example: Kafka streaming data ingested into Hive for real-time analysis.
Near Real Time	Data is processed with minimal delay, usually within seconds to minutes after generation. Balances latency and processing efficiency. Example: Log data collected and updated in Hive every few minutes for dashboard reporting.
Batch Processing	Data is collected over a period and processed in bulk at scheduled intervals. Suitable for large volume data and complex transformations. Example: Daily sales data imported into Hive tables overnight for reporting and analytics.

File Types

Types of Files in Hive

Fixed Width Files

Data is stored in a fixed-length format where each field has a predefined size. Common in legacy systems, these files require precise parsing to extract fields correctly.

Delimited Files

Fields are separated by delimiters such as commas, tabs, or pipes. CSV and TSV are popular examples. Easy to read and widely used for tabular data exchange.

Mainframe Files (EBCDIC)

Data encoded in EBCDIC format typical of mainframe systems. Hive supports importing and processing these files by converting them to ASCII or UTF-8.

Columnar Storage Formats

AVRO, ORC, and PARQUET store data in a compressed, column-oriented manner, optimizing read performance and storage efficiency in Hive for big data workloads.

Compression Techniques in Hive

Gzip Compression in Hive

- Gzip is a widely used compression format supported natively in Hive.
- Compresses data files to reduce storage space and improve query efficiency.
- Easy to enable on Hive tables using TBLPROPERTIES or file format settings.
- Balances compression ratio and decompression speed effectively.

File-Level and Block-Level Compression

- File-level compression compresses entire data files, reducing disk usage significantly.
- Block-level compression compresses data in smaller blocks, improving read/write performance.
- Hive supports block-level compression with columnar file formats like ORC and Parquet.
- Block-level compression allows selective decompression, speeding up query execution.

Partitioning

Partitioning of Data in Hive

Partitioning divides data into segments based on keys, improving query speed by scanning only relevant partitions.

Random Partitioning

Data is distributed randomly across partitions to balance load but offers less efficient query pruning.

Partitioned tables with randomized data
Balanced load distribution
Easy to implement

Hash Partitioning

Data is divided by hashing a column, enabling efficient query filtering by targeting specific partitions.

Partitioned tables based on hash keys
Optimized query performance
Example: HASH(partition_column) % total_partitions

Example Code for Partitioning

Create a table partitioned by region:

```
CREATE TABLE sales(id INT, amount FLOAT)
PARTITIONED BY (region STRING);
```

Load data:

```
LOAD DATA INPATH '/path' INTO TABLE sales
PARTITION (region='US');
```

Hive DDL for partitioned table
Data loading with partition
Partition pruning in queries

Summary

Summary and Key Takeaways

- Hive bridges traditional RDBMS and Hadoop, enabling SQL-like queries on big data.
- Data types include structured, semi-structured, and unstructured, each requiring specific handling.
- Data frequency varies: real-time streaming, near real-time, and batch processing.
- File formats like AVRO, ORC, PARQUET, and EBCDIC support diverse data storage needs.
- Partitioning and compression techniques optimize performance and storage in Hive.

Thank You