Webinar on Become a Big Data Engineer

Host Speaker:

Rashedul Alam Shakil

Founder of Study Mart
Founder of aiQuest Intelligence
Automation & Database Programmer,
Siemens, Nuremberg, Germany
Master in Data Science,
University of Erlangen, Germany

Full Video Link: https://youtu.be/BnRv2TXyqAg

Guest Speaker:

A.K.M. Alfaz Uddin

10 Years+ Industry Expert
Enterprise Data Engineering Lead Engineer,
Banglalink Digital Communications Ltd.
Former Lead Engineer, bKash Limited.
Former Senior Software Engineer, IMpulse (BD) Ltd
Former Specialist, BI/DW & CLM Systems, Robi Axiata Limited
Bachelor in CSE at KUET, Khulna, Bangladesh

What is Data?

Data is a collection of **facts**, such as numbers, words, measurements, observations, images, videos, audio, or even just descriptions of things.

- **Types of Data:**
  - Structured Data:
    - Organized and easily searchable data
    - Examples: Databases, spreadsheets
  - Unstructured Data:
    - Not organized in a pre-defined manner
      - Examples: Emails, videos, social media posts
  - Semi-Structured Data:
    - Partially organized but not fully
    - Examples: JSON files, XML files

What is Data?

Data is a collection of **facts**, such as numbers, words, measurements, observations, images, videos, audio, or even just descriptions of things.

- **Sources of Data:**
  - Human-Generated Data:
    - Social media posts, text messages, photos
  - Machine-Generated Data:
    - Sensor data, logs from servers, transaction records

What is Big Data?

Big Data refers to large, complex datasets that traditional data processing software cannot manage and process efficiently.

- **Characteristics of Big Data:**
  - **Volume:** The vast amounts of data generated every second.
  - **Velocity:** The speed at which data is generated and processed.
  - **Variety:** The different types of data (structured, unstructured, semi-structured).
  - **Veracity:** The uncertainty of data accuracy.
  - **Value:** The potential insights and benefits that can be derived from analyzing Big Data.

What is Big Data?

Big Data refers to large, complex datasets that traditional data processing software cannot manage and process efficiently.

- **Importance in Today's World:**
  - Big Data enables better decision-making through data-driven insights.
  - It drives innovation in various fields such as healthcare, finance, and retail.
  - Big Data technologies help in predicting trends, understanding customer behavior, and improving operational efficiency.

Normal Data Vs. Big Data

Aspect	Normal Data	Big Data
Volume	Limited size, usually manageable with traditional tools	Extremely large datasets, often in petabytes or exabytes
Velocity	Data generated at a slower, more predictable rate	Data generated at high speed, requiring real-time processing
Variety	Typically structured data	Includes structured, unstructured, and semi-structured data
Complexity	Lower complexity, easier to analyze	High complexity due to various data sources and formats
Processing Methods	Traditional databases and data warehousing solutions	Advanced tools like Hadoop, Spark, and NoSQL databases

Concepts of
Databases & Data Warehouse

What is a Database?

A database is an organized collection of data, typically stored and accessed electronically from a computer system. Databases are designed to support the storage, retrieval, and management of data in a structured format.

- **Key Characteristics:**
  - **Structured Data Storage:** Data is stored in tables, rows, and columns, making it easy to organize and retrieve.
  - **Data Integrity:** Ensures accuracy and consistency of data over its entire lifecycle.
  - **Scalability:** Can handle increasing amounts of data and users.
  - **Security:** Protects data through authentication and authorization mechanisms.

- **Types of Databases:**
  - **Relational Databases (RDBMS):** Use SQL for defining and manipulating data (MySQL, PostgreSQL, Oracle).
  - **NoSQL Databases:** Designed for unstructured data and large-scale data storage (MongoDB, Cassandra).
  - **In-Memory Databases:** Store data in the main memory for faster access (Redis, Memcached).
  - **Distributed Databases:** Data is distributed across different locations for redundancy and faster access (Apache Cassandra, Amazon DynamoDB).

What is a Data Warehouse?

A **Data Warehouse** is a centralized repository that stores large volumes of data collected from various sources. It is designed to support business intelligence activities, particularly analytics and reporting.

- **Key Characteristics:**
  - **Centralized Storage:** Integrates data from multiple sources into a single repository.
  - **Historical Data:** Stores historical data, enabling trend analysis over time.
  - **Optimized for Queries:** Structured for fast retrieval and complex queries.
  - **Data Integration:** Combines structured data from different sources, such as databases, flat files, and external data streams.

- **Components of a Data Warehouse:**
  - **ETL Process (Extract, Transform, Load):** Extracts data from source systems, transforms it into a suitable format, and loads it into the data warehouse.
  - **Metadata:** Data about the data, providing information on data source, usage, and structure.
  - **OLAP (Online Analytical Processing) Engine:** Allows users to perform multidimensional analysis of data.

Database Vs. Data Warehouse

Aspect	Database	Data Warehouse
Purpose	Designed for transactional processing	Designed for analytical processing
Data Type	Current, real-time data	Historical, aggregated data
Normalization	Highly normalized for reducing redundancy	Denormalized to optimize read performance
Read/Write Operations	Handles frequent read/write operations	Primarily read-intensive operations
Query Complexity	Simple, short queries for transactions	Complex queries for analysis and reporting

Components of Big Data Architecture

Components of Big Data Architecture

Main five components of Big Data architecture.

1. Data Sources
2. Data Storage
3. Data Processing
4. Data Analysis
5. Data Visualization

Components of Big Data Architecture

- **Data Sources:** These are the origins of the data collected. Data sources can be anything from transactional databases, social media platforms, sensors, devices, applications, or external data feeds.
  - Examples: Customer transaction records, social media posts, IoT sensor data.

- **Data Storage:** This component is responsible for storing large volumes of data generated from various sources. Storage solutions must be scalable and capable of handling both structured and unstructured data.
  - Examples: HDFS (Hadoop Distributed File System), NoSQL databases like MongoDB and Cassandra, and cloud storage solutions like Amazon S3.

- **Data Processing:** This involves the extraction, transformation, and loading (ETL) of data. Processing is required to clean, aggregate, and transform raw data into a format suitable for analysis.
  - Examples: Apache Hadoop for batch processing, and Apache Spark for real-time processing.

Components of Big Data Architecture

- **Data Analysis:** This component involves analyzing the processed data to extract meaningful insights. Analysis can be descriptive, predictive, or prescriptive, depending on the objectives.
  - Examples: Using tools like Apache Hive for querying large datasets, and machine learning algorithms for predictive analytics.

- **Data Visualization:** This is the final step where the analyzed data is presented in an easily understandable format using graphs, charts, dashboards, and reports.
  - Examples: Visualization tools like Tableau, Power BI, and D3.js.

Data Storage Solutions

Data Storage Solutions

Common data storage solutions:

1. Traditional
2. Big Data Storage
3. Hadoop HDFS
4. NoSQL Databases (Cassandra, MongoDB)
5. Cloud Storage Solutions

Data Storage Solutions

- **Traditional Storage:** Typically relies on relational databases and data warehousing solutions. It is suitable for structured data and smaller datasets. Traditional storage solutions often face limitations in scalability and flexibility when dealing with large volumes of diverse data.

- **Big Data Storage:** Designed to handle massive volumes of data that come in various formats. Big data storage solutions are scalable, and distributed, and can manage both structured and unstructured data efficiently.

- **Hadoop Distributed File System (HDFS):** A scalable, distributed file system that stores large datasets across multiple machines. It is a core component of the Hadoop ecosystem and allows for high-throughput access to large volumes of data. HDFS is fault-tolerant and designed to run on commodity hardware, making it cost-effective for handling big data workloads.

Data Storage Solutions

- **NoSQL Databases (Cassandra, MongoDB):**
  - **Cassandra:** A highly scalable, distributed NoSQL database designed to handle large amounts of data across many commodity servers without a single point of failure. It excels in providing high availability and fault tolerance.

  - **MongoDB:** A flexible, document-oriented NoSQL database that stores data in JSON-like format, making it ideal for handling semi-structured and unstructured data. It offers horizontal scalability and is widely used for applications requiring real-time analytics and rapid data retrieval.

- **Cloud Storage:** Refers to storing data on remote servers accessed via the internet, provided by cloud service providers such as Amazon S3 (AWS), Google Cloud Storage, and Azure Blob Storage. Cloud storage solutions offer scalable, flexible, and cost-effective storage options. They support large volumes of data and provide features like automated backups, data replication, and easy integration with other cloud services and big data tools.

Data Processing Frameworks

Batch Processing: Batch processing involves processing large volumes of data all at once (in batches) at scheduled intervals. It is suitable for tasks where immediate real-time processing is not required.

- **Apache Hadoop:** Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from a single server to thousands of machines, each offering local computation and storage.
  - **Key Features:**
    - **Scalability:** Easily scales from a single machine to thousands.
    - **Fault Tolerance:** Automatically handles hardware failures.
    - **Cost-Effectiveness:** Can be run on commodity hardware.
    - **Components:** Includes Hadoop Distributed File System (HDFS) and MapReduce for processing.

Stream Processing: Stream processing involves processing data in real-time as it is generated or received. This is crucial for applications requiring immediate analysis and response, such as fraud detection, real-time monitoring, and event-driven applications.

- **Apache Spark:** Spark is an open-source unified analytics engine for large-scale data processing. It provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.
  - **Key Features:**
    - **Speed:** Processes data much faster than traditional batch processing due to in-memory computing.
    - **Ease of Use:** Provides APIs in Java, Scala, Python, and R.
    - **Unified Engine:** Supports both batch and stream processing.
    - **Components:** Includes Spark SQL, Spark Streaming, MLlib (machine learning), and GraphX (graph processing).

Data Processing Frameworks

- **Apache Flink:** Flink is a stream processing framework that provides high-throughput, low-latency, and exactly-once processing semantics. It is designed to run stateful computations over bounded and unbounded data streams.
    - **Key Features:**
        - **Real-Time Processing:** Optimized for stream processing.
        - **Fault Tolerance:** Ensures exactly-once-state consistency.
        - **Scalability:** Scales to handle large volumes of data.
        - **Event Time Processing:** Handles out-of-order data with sophisticated windowing and state management.

Batch Processing Vs. Stream Processing

Aspect	Batch Processing	Stream Processing
Data Handling	Processes large volumes of data in batches	Processes data continuously in real-time
Latency	Higher latency as data is processed at scheduled intervals	Low latency, near real-time processing
Use Cases	Suitable for historical data analysis, reporting	Suitable for real-time analytics, event detection
Processing Time	Fixed processing window	Continuous, ongoing processing
Fault Tolerance	Fault tolerance through re-processing of entire batches	Fault tolerance with state management and checkpointing

Popular ETL Tools

Popular ETL Tools

- **Apache NiFi:** Apache NiFi is an open-source data integration tool designed for data flow automation. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic.
  - **Key Features:**
    - Web-based user interface
    - Real-time data ingestion and processing
    - Extensive support for various data formats and protocols
- **Talend:** Talend is an open-source ETL tool that provides a suite of data integration and data management solutions. It allows users to connect, transform, and manage data across various sources.
  - **Key Features:**
    - Drag-and-drop interface
    - Pre-built connectors for various data sources
    - Real-time and batch processing capabilities

Popular ETL Tools

- **Apache Spark:** Apache Spark is a powerful open-source processing engine that can perform ETL tasks at scale. Spark provides in-memory data processing capabilities and supports multiple programming languages.
  - **Key Features:**
    - High-speed data processing
    - Support for batch and stream processing
    - Integration with Hadoop ecosystem

- **AWS Glue:** AWS Glue is a fully managed ETL service provided by Amazon Web Services. It simplifies the process of data preparation and loading for analytics.
  - **Key Features:**
    - Serverless architecture
    - Automatic schema discovery
    - Integration with various AWS services

Challenges in Big Data Engineering

Challenges in Big Data Engineering

Scalability Issues

Scalability issues arise when the data volume, velocity, and variety exceed the capacity of current systems and infrastructures. As data grows, it becomes increasingly challenging to scale storage, processing power, and network capabilities efficiently.

- **Examples:**
  - Handling petabytes of data without degradation in performance.
  - Ensuring data processing frameworks can scale horizontally across distributed systems.

Challenges in Big Data Engineering

Data Quality and Governance

Ensuring high data quality and proper governance is crucial for making reliable decisions based on big data. Poor data quality can lead to inaccurate insights, while lack of governance can result in data management inefficiencies and compliance risks.

- **Data Quality:**
  - Challenges include dealing with missing, inconsistent, and duplicate data.
  - Ensuring data accuracy, completeness, and timeliness.

- **Data Governance:**
  - Implementing policies and procedures for data stewardship.
  - Ensuring data lineage, data cataloging, and compliance with regulations (GDPR).

Challenges in Big Data Engineering

Security and Privacy Concerns

As the volume of sensitive data increases, so do the risks associated with data breaches and unauthorized access. Protecting big data involves implementing robust security measures and ensuring compliance with privacy regulations.

- **Security:**
  - Protecting data at rest and in transit using encryption.
  - Implementing access controls and authentication mechanisms.
  - Monitoring and auditing data access and usage.

- **Privacy:**
  - Ensuring compliance with data protection laws.
  - Implementing data anonymization and pseudonymization techniques.
  - Addressing concerns related to data ownership and user consent.

Essential Skills to Become a Data Engineer

Essential Skills to Become a Data Engineer

- Programming Languages
- Big Data Technologies
- Database Management Systems
- Data Warehousing Solutions
- ETL Tools
- Data Modeling
- Cloud Platforms
- Data Governance and Security
- DevOps Skills
- Soft Skills

Essential Skills to Become a Data Engineer

- **Programming Languages (1)**
  - Python
  - Java/Scala
  - SQL
- **Database Management Systems (2)**
  - MySQL
  - PostgreSQL
  - Oracle
  - MongoDB
  - Cassandra
- **Big Data Technologies (3)**
  - Apache Hadoop
  - Apache Spark
  - Kafka
- **Data Warehousing Solutions (4)**
  - Amazon Redshift
  - Google BigQuery
  - Snowflake
- **ETL Tools (5)**
  - Talend
  - Apache Nifi
  - Data Modeling
  - Star Schema
  - Snowflake Schema
  - Dimensional Modeling
- **Cloud Platforms (6)**
  - AWS (S3, EC2, Lambda, Redshift)
  - Microsoft Azure (Data Lake, SQL Database, Synapse Analytics)
  - Google Cloud Platform (BigQuery, Cloud Storage, Dataflow)
- **Data Governance and Security (7)**
  - Data Quality Management
  - Data Privacy (GDPR, CCPA)
  - Encryption and Access Control
- **DevOps Skills (8)**
  - Docker
  - Kubernetes
  - CI/CD Pipelines
  - Terraform

Future Trends in Big Data

Future Trends in Big Data

Artificial Intelligence Integration

The integration of artificial intelligence (AI) with big data analytics is transforming the way data is processed and analyzed. AI algorithms and machine learning models are used to uncover patterns, make predictions, and automate decision-making processes.

- **Examples:**
  - Using AI for predictive maintenance in manufacturing.
  - Implementing machine learning models to improve customer personalization in e-commerce.
  - Enhancing fraud detection systems in finance with AI algorithms.

Future Trends in Big Data

Edge Computing

Edge computing involves processing data closer to the source of data generation, such as IoT devices, rather than relying solely on centralized data centers. This reduces latency, saves bandwidth, and allows for real-time data processing.

- **Examples:**
  - Smart home devices that process data locally for faster response times.
  - Autonomous vehicles that process sensor data in real time to make driving decisions.
  - Industrial IoT systems that analyze data on-site to monitor equipment health and performance.

Future Trends in Big Data

Data Fabric

Data fabric is an emerging architecture that provides a unified and consistent way to manage and access data across a variety of environments, including on-premises and cloud. It aims to simplify data management and integration, enhancing data accessibility and security.

- Examples:
  - Enabling seamless data integration across different departments in an organization.
  - Providing a unified data platform that supports both transactional and analytical workloads.
  - Enhancing data governance and compliance by providing a consistent view of data.

Do you have any Questions?

Visit: Become a Big Data Engineering Course by aiQuest Intelligence