

# (Big) Data Engineering In Depth

From Beginner to Professional

Mostafa Alaa Mohamed

Senior Big Data Engineer

 MoustafaAlaa  Moustafa Alaa  @Moustafa\_alaa22

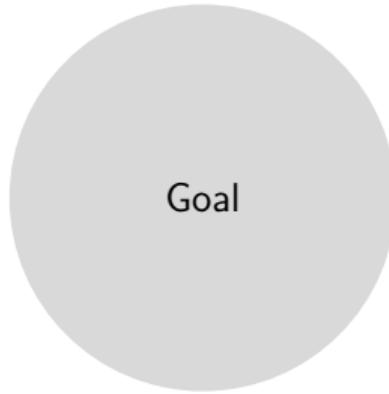
 mustafa.alaa.mohamed@gmail.com

<sup>1</sup>Big Data & Analytics Department, Epam Systems

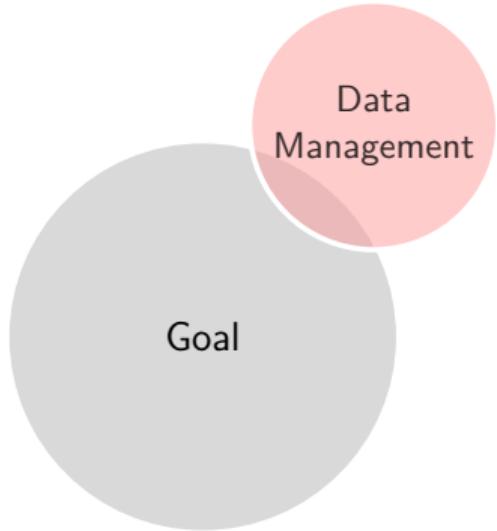
The Definitive Guide to Big Data Engineering Tasks

# Course Introduction

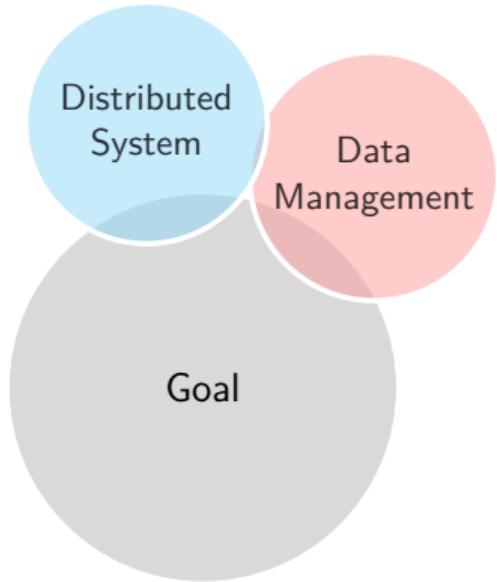
# Course Target



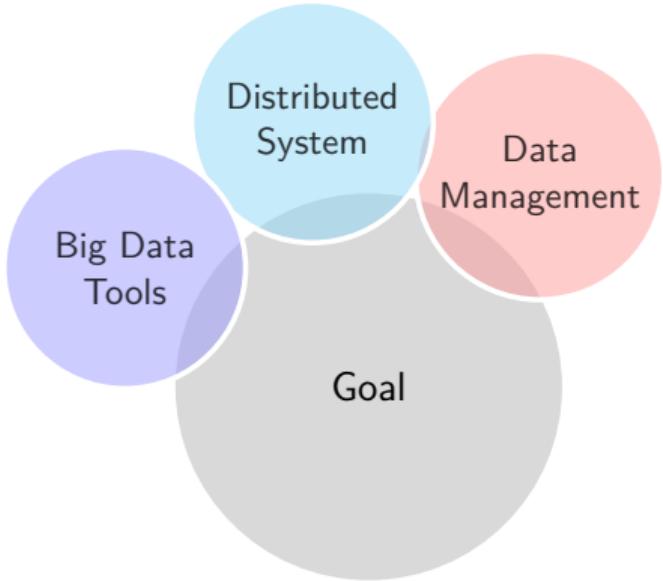
# Course Target



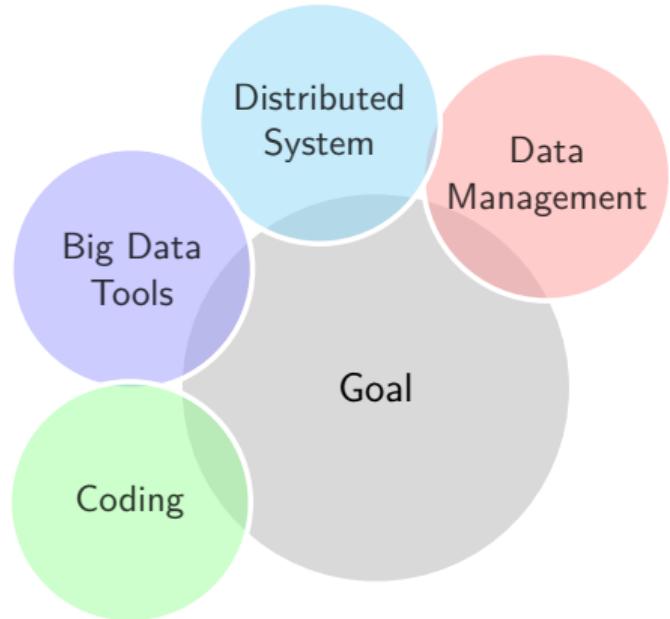
# Course Target



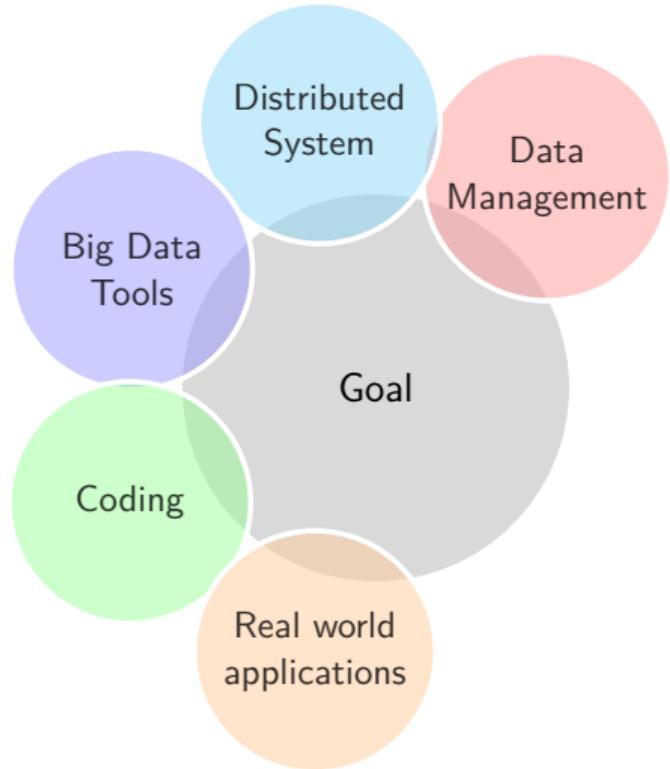
# Course Target



# Course Target



# Course Target



# Course Target



# Course Target



## Learning Objectives and Audience

# Learning Objectives

- Simplify the concepts in data management.

# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.



# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts



# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts
- Be familiar with ETL for (batch/streaming) data over distributed systems ex: Hadoop & Spark.

# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts
- Be familiar with ETL for (batch/streaming) data over distributed systems ex: Hadoop & Spark.
- Apply QA and testing for the data pipeline cycle.



# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts
- Be familiar with ETL for (batch/streaming) data over distributed systems ex: Hadoop & Spark.
- Apply QA and testing for the data pipeline cycle.
- Build and scale your data product.

# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts
- Be familiar with ETL for (batch/streaming) data over distributed systems ex: Hadoop & Spark.
- Apply QA and testing for the data pipeline cycle.
- Build and scale your data product.
- Building real-life examples.



# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts
- Be familiar with ETL for (batch/streaming) data over distributed systems ex: Hadoop & Spark.
- Apply QA and testing for the data pipeline cycle.
- Build and scale your data product.
- Building real-life examples.
- Applying machine learning over big data.

# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts
- Be familiar with ETL for (batch/streaming) data over distributed systems ex: Hadoop & Spark.
- Apply QA and testing for the data pipeline cycle.
- Build and scale your data product.
- Building real-life examples.
- Applying machine learning over big data.
- Automate the data life-cycle process end-to-end (e2e).



# Learning Objectives

- Simplify the concepts in data management.
- Understand the data management life-cycle.
- Illustrate the basics of distributed systems concepts
- Be familiar with ETL for (batch/streaming) data over distributed systems ex: Hadoop & Spark.
- Apply QA and testing for the data pipeline cycle.
- Build and scale your data product.
- Building real-life examples.
- Applying machine learning over big data.
- Automate the data life-cycle process end-to-end (e2e).
- Understanding of the DevOps tools and functions in data life-cycle.

# Audience: Who Should Take This Course?

- Data Engineer who needs to get more knowledge in distributed systems and Big Data.

## Audience: Who Should Take This Course?

- Data Engineer who needs to get more knowledge in distributed systems and Big Data.
- Data Warehouse Engineer who needs to know more about big data.

## Audience: Who Should Take This Course?

- Data Engineer who needs to get more knowledge in distributed systems and Big Data.
- Data Warehouse Engineer who needs to know more about big data.
- Software developer who needs to change to data engineering track.

## Audience: Who Should Take This Course?

- Data Engineer who needs to get more knowledge in distributed systems and Big Data.
- Data Warehouse Engineer who needs to know more about big data.
- Software developer who needs to change to data engineering track.
- DevOps engineers who needs to understand the concepts of big data.



## Audience: Who Should Take This Course?

- Data Engineer who needs to get more knowledge in distributed systems and Big Data.
- Data Warehouse Engineer who needs to know more about big data.
- Software developer who needs to change to data engineering track.
- DevOps engineers who needs to understand the concepts of big data.
- Business or entrepreneur who needs to get more information about how to build or manage a data product.

Getting max benefit from this course

# Getting max benefit from this course

## Take the course advantage

- Follow the videos order as described.

# Getting max benefit from this course

## Take the course advantage

- Follow the videos order as described.
- Read the references for each section (including the implementation of the examples if exists).

# Getting max benefit from this course

## Take the course advantage

- Follow the videos order as described.
- Read the references for each section (including the implementation of the examples if exists).
- Repeat the lecture code with your own.

# Getting max benefit from this course

## Take the course advantage

- Follow the videos order as described.
- Read the references for each section (including the implementation of the examples if exists).
- Repeat the lecture code with your own.
- Do the assignments.

# Getting max benefit from this course

## Take the course advantage

- Follow the videos order as described.
- Read the references for each section (including the implementation of the examples if exists).
- Repeat the lecture code with your own.
- Do the assignments.
- Ask your questions.

# Getting max benefit from this course

## Take the course advantage

- Follow the videos order as described.
- Read the references for each section (including the implementation of the examples if exists).
- Repeat the lecture code with your own.
- Do the assignments.
- Ask your questions.
- Join the online meeting or discussions.

# Chapter Dependencies

 You MUST finish the red chapters first

Ch.01 Introduction

 Finish colors group before move to the next.

Ch.02 Data Management

Ch.03 Distributed Systems

Ch.04 Hadoop and MR

Ch.05 FN and Scala

Ch.06 Spark

Ch.07 Big Data Application

Ch.08 Massging Systems

Ch.09 Data Orchestration

Ch.10 NoSql

Ch.11 Elastic

Ch.12 Data Architecture Design

# Chapter Dependencies (Jump Out Path)

You MUST finish the red chapters first

Ch.01 Introduction

Finish colors group before move to the next.

Ch.02 Data Management

Ch.03 Distributed Systems

Ch.04 Hadoop and MR

Ch.05 FN and Scala

Ch.06 Spark

Ch.07 Big Data Application

Ch.08 Massging Systems

Ch.09 Data Orchestration

Ch.10 NoSql

Ch.11 Elastic

Ch.12 Data Architecture Design

## Assignments and Labs

# Assignments and Labs

## Remark

- Full project code.

# Assignments and Labs

## Remark

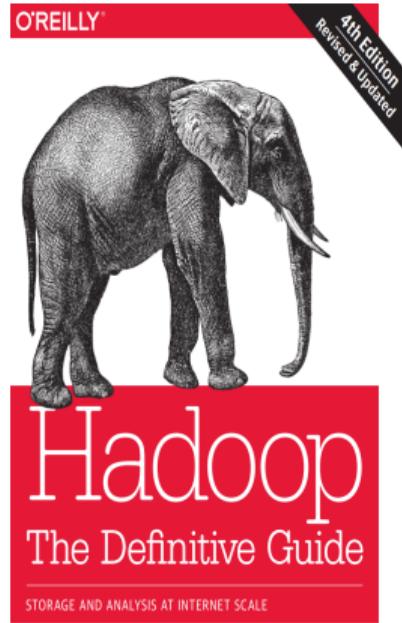
- Full project code.
- Notebooks (Jupyter or Zeppelin).

# Assignments and Labs

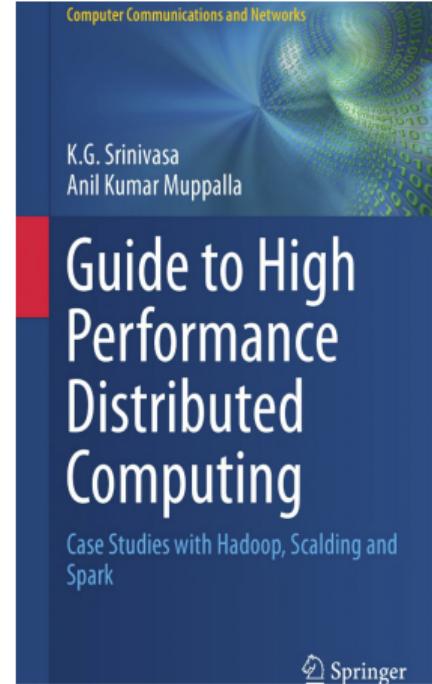
## Remark

- Full project code.
- Notebooks (Jupyter or Zeppelin).
- Read the reference.

# Textbooks-1



Tom White



# Textbooks-2



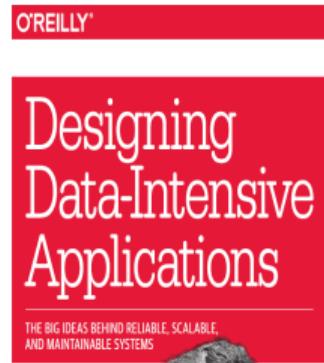
# Textbooks-3



Neha Narkhede,  
Gwen Shapira & Todd Palino



Jeff Carpenter & Eben Hewitt



Martin Kleppmann



# Ugly but important

- User stories or technical discussions are not related to any of my current work or my previous companies.



# Ugly but important

- User stories or technical discussions are not related to any of my current work or my previous companies.
- I am working at EPAM Systems. My company approved me for doing this online course public but the materials are not reviewed or assessed by my company. It is on my own responsibilities.

# Table of Contents I

## 1 Course Introduction

- Learning Objectives and Audience
- Getting max benefit from this course
- Assignments and Labs

## 2 Introduction To Data Management and Data Warehouse

- Data Management
- Data Abstraction
- From Operational DB to DWH
  - Differences Between DWH and Operational DB
  - Types of DWH
  - Use Cases of Operational DB vs DWH
- DWH Characteristics
- Datawarehouse Components
  - ETL Process
  - ETL vs ELT When? Why?



# Table of Contents II

- Data Models
  - Data Model Design
  - DWH Architecture Overview
  - Data Encoding and Formats
- Further Readings and Assignment

## 3 Introduction To Distributed Systems

- Distributed Systems Concepts
- Distributed Systems Architecture
- Distributed Systems Challenges
- Design Simple Distributed System
- Further Readings and Assignment

## 4 Introduction to Hadoop and Map-Reduce

- Hadoop Architecture
  - Storage
  - YARN

# Table of Contents III

- Hadoop I/O
- Processing
- Map-Reduce
  - Map-Reduce Components
  - Word-Count Example
- Pig
- Hive
- ZooKeeper
- Further Readings and Assignment

## 5 Functional Programming

- Why functional programming commonly used in distributed systems?
- Introduction to Scala
- Further Readings and Assignment

# Table of Contents IV

## 6 Spark Framework

- Spark Philosophy towards the Engine and the Programming languages
- Spark Basics
- Spark Programming using RDDs
  - Spark RDD
  - Spark Working With Key/Value Pairs
- Spark Datasets/Dataframe
  - Spark SQL
  - Dataframes/Datasets vs. RDDs
- Spark on Production
- Spark For Batch Processing
- Building custom input and output connector using Spark
- Spark Streaming
- Spark using other Programming Languages

# Table of Contents V

- PySpark for Python Geeks
- RSpark for R Geeks
- Spark For Data Scientist
- Spark Graph Dataframe/Graphx
- Tuning your Spark Jobs
- Further Readings and Assignment

7

## Real World Applications

- Big Data Development Life Cycle
- Template Concept for Data Engineering
  - Template for ETL Application
  - Template for QA
  - Template for Streaming Applications
  - Template for Machine Learning Applications
- Further Readings and Assignment



8

## Massaging Systems

# Table of Contents VI

- Motivation
- Messaging Systems Architecture
- JMS as an example
- Introduction to Kafka
  - Kafka Architecture
  - Kafka Topics
  - Partitions
  - Kafka Producers
  - Kafka Consumers
  - Kafka Connector
  - Kafka Custom Connectors
  - Kafka Configuration
  - Kafka Configuration Optimizations
  - Kafka Operations
  - Kafka Integration with Enterprise tools
- Further Readings and Assignment

# Table of Contents VII

## 9 Data Orchestration

- Motivation
- Enterprise vs Open source tools
  - Open source tools (Oozie as an Example)
  - Enterprise source tools
  - How to choose the right tool?
- Further Readings and Assignment

## 10 NOSQL

- Introduction to NoSQL Databases.
- Cassandra
  - Why Cassandra?
  - Introducing Cassandra
  - The Cassandra Data Model
  - Architecture
  - Reading and Writing Data
  - Integrating Hadoop

# Table of Contents VIII

- Further Readings and Assignment

## 11 Elastic

- Further Readings and Assignment

## 12 Data Architecture Design

- Further Readings and Assignment

## 13 Appendix

- Appendix A- Shell Programming
- Appendix B- Java Programming
- Appendix C- Scala Programming
- Appendix D- SQL Programming
- Appendix E- Oozie Orchestration
- Appendix F- DWH Concepts and Data Modeling Design
- Appendix G- Machine Learning Concepts Data Engineers
- Appendix H- Docker for Data Engineers

# Introduction To Data Management and Data Warehouse

# Chapter Objectives

- Be familiar with data management life-cycle.



# Chapter Objectives

- Be familiar with data management life-cycle.
- Understand the data abstraction and the data layer.



# Chapter Objectives

- Be familiar with data management life-cycle.
- Understand the data abstraction and the data layer.
- Motivation to DWH.

# Chapter Objectives

- Be familiar with data management life-cycle.
- Understand the data abstraction and the data layer.
- Motivation to DWH.
- What is the different types of DWH?

# Chapter Objectives

- Be familiar with data management life-cycle.
- Understand the data abstraction and the data layer.
- Motivation to DWH.
- What is the different types of DWH?
- Use cases for DWH and how it differ from the operational db.

# Chapter Objectives

- Be familiar with data management life-cycle.
- Understand the data abstraction and the data layer.
- Motivation to DWH.
- What is the different types of DWH?
- Use cases for DWH and how it differ from the operational db.
- Explain the data Encoding and Formats.



# Chapter Objectives

- Be familiar with data management life-cycle.
- Understand the data abstraction and the data layer.
- Motivation to DWH.
- What is the different types of DWH?
- Use cases for DWH and how it differ from the operational db.
- Explain the data Encoding and Formats.
- Show what is the challenges to build a DWH?

# Chapter Objectives

- Be familiar with data management life-cycle.
- Understand the data abstraction and the data layer.
- Motivation to DWH.
- What is the different types of DWH?
- Use cases for DWH and how it differ from the operational db.
- Explain the data Encoding and Formats.
- Show what is the challenges to build a DWH?
- What is the data modeling and its design?



# Data Management

# Data Management

- Data are a product.



# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):



# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.



# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).

# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).



# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).
  - Delivery automation (Tools and Process) AKA **DevOps** cycle.



# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).
  - Delivery automation (Tools and Process) AKA **DevOps** cycle.
  - **Extraction** Process (collection).



# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).
  - Delivery automation (Tools and Process) AKA **DevOps** cycle.
  - **Extraction** Process (collection).
  - **Transformation** ex: (cleansing, Apply business logic, Organize).

# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).
  - Delivery automation (Tools and Process) AKA **DevOps** cycle.
  - **Extraction** Process (collection).
  - **Transformation** ex: (cleansing, Apply business logic, Organize).
  - **Loading** or store the transformed data based on our usage or use case.

# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).
  - Delivery automation (Tools and Process) AKA **DevOps** cycle.
  - **Extraction** Process (collection).
  - **Transformation** ex: (cleansing, Apply business logic, Organize).
  - **Loading** or store the transformed data based on our usage or use case.
  - Business Intelligence (**BI**) or data discovery (continues process).

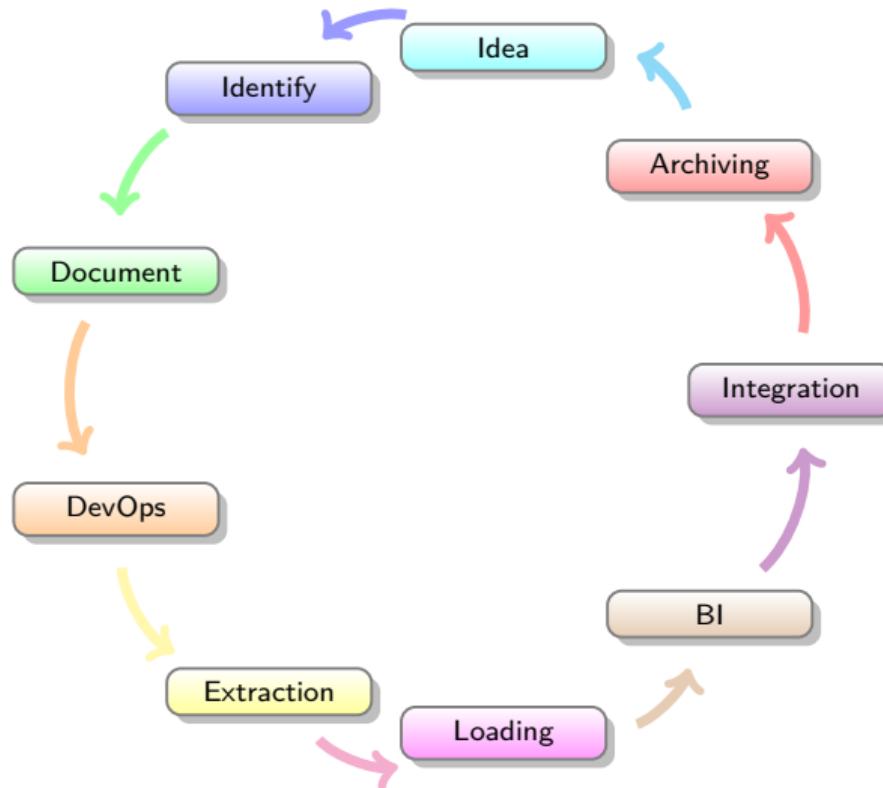
# Data Management

- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).
  - Delivery automation (Tools and Process) AKA **DevOps** cycle.
  - **Extraction** Process (collection).
  - **Transformation** ex: (cleansing, Apply business logic, Organize).
  - **Loading** or store the transformed data based on our usage or use case.
  - Business Intelligence (**BI**) or data discovery (continues process).
  - **Integration** and publishing.

# Data Management

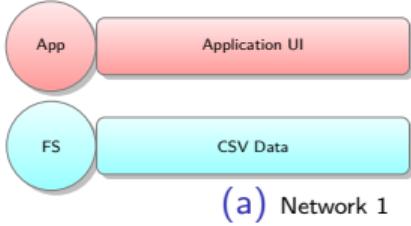
- Data are a product.
- Data product has a life-cycle as following (simplified):
  - **Question**, Idea, or service.
  - **Identifying** the source of information and the data type ex: (text, images, videos, audio, or sensors).
  - **Document** all details regarding the data including quality, security, efficiency, and access (consideration during the cycle).
  - Delivery automation (Tools and Process) AKA **DevOps** cycle.
  - **Extraction** Process (collection).
  - **Transformation** ex: (cleansing, Apply business logic, Organize).
  - **Loading** or store the transformed data based on our usage or use case.
  - Business Intelligence (**BI**) or data discovery (continues process).
  - **Integration** and publishing.
  - Data retention or **archiving** process ex: (Hot or Cold storage).

# Data Management Life-Cycle

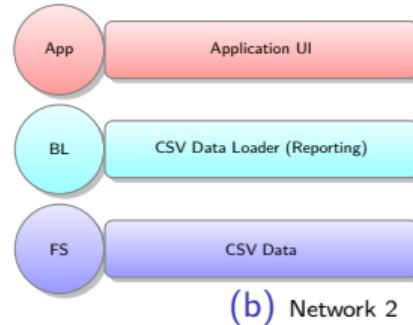


# Data Abstraction

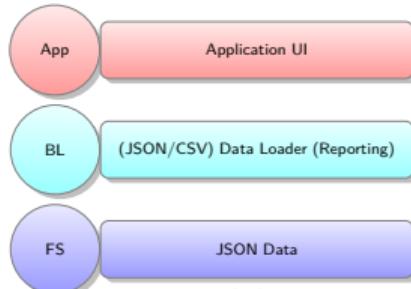
# Motivation to Data Layers (Use Case)



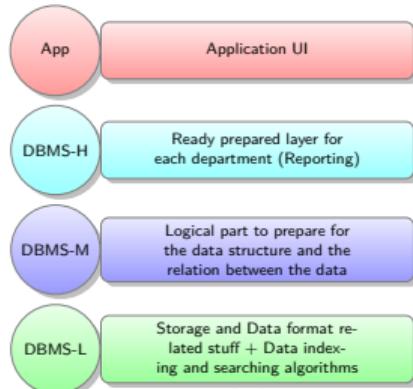
(a) Network 1



(b) Network 2



(c) Network 3



(d) Network 4

Figure: The average and standard deviation of critical parameters: Region R4

# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?



# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.



# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.
  - Identify the problem (challenges).



# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.
  - Identify the problem (challenges).
  - Think about how to overcome the challenges.

# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.
  - Identify the problem (challenges).
  - Think about how to overcome the challenges.
  - Ask your self the following questions:



# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.
  - Identify the problem (challenges).
  - Think about how to overcome the challenges.
  - Ask your self the following questions:
    - Can we solve the problem using the current data structure with adding new features?

# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.
  - Identify the problem (challenges).
  - Think about how to overcome the challenges.
  - Ask your self the following questions:
    - Can we solve the problem using the current data structure with adding new features?
    - What if we enhance/change the data structure or modeling?

# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.
  - Identify the problem (challenges).
  - Think about how to overcome the challenges.
  - Ask your self the following questions:
    - Can we solve the problem using the current data structure with adding new features?
    - What if we enhance/change the data structure or modeling?
    - Could it help if we change the backend engine (ex: DBMS system)?

# Motivation to Data Layers (Solution Thinking)

- How can we think about a data solution or challenges in the data products?
  - Requirements analysis.
  - Identify the problem (challenges).
  - Think about how to overcome the challenges.
  - Ask your self the following questions:
    - Can we solve the problem using the current data structure with adding new features?
    - What if we enhance/change the data structure or modeling?
    - Could it help if we change the backend engine (ex: DBMS system)?
- To answer these questions you need to understand the data layers.

# Data Layers (Abstraction)

- Any data product (database) contains multi-layers.

# Data Layers (Abstraction)

- Any data product (database) contains multi-layers.
- Each layer responsible for different tasks and operations.



# Data Layers (Abstraction)

- Any data product (database) contains multi-layers.
- Each layer responsible for different tasks and operations.
- Each layers interacts with (hardware or software or mixed).

# Data Layers (Abstraction)

- Any data product (database) contains multi-layers.
- Each layer responsible for different tasks and operations.
- Each layers interacts with (hardware or software or mixed).
- To eliminate the complexity for data interactions not all internal processes are shared or available for the user.

# Data Layers (Abstraction)

- Any data product (database) contains multi-layers.
- Each layer responsible for different tasks and operations.
- Each layers interacts with (hardware or software or mixed).
- To eliminate the complexity for data interactions not all internal processes are shared or available for the user.
- The developer for each layer hide internal irrelevant details from developer (users).

# Data Layers (Abstraction)

- Any data product (database) contains multi-layers.
- Each layer responsible for different tasks and operations.
- Each layers interacts with (hardware or software or mixed).
- To eliminate the complexity for data interactions not all internal processes are shared or available for the user.
- The developer for each layer hide internal irrelevant details from developer (users).
- The process of **hiding** irrelevant details from developer (user) is called data **abstraction**.

# Data Layers (Abstraction)

## Definition

**Data Abstraction and Data Independence:** DBMS comprise of complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

- There are 3 levels of data abstraction.

# Data Layers (Abstraction)

## Definition

**Data Abstraction and Data Independence:** DBMS comprise of complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

- There are 3 levels of data abstraction.
  - Physical Level

# Data Layers (Abstraction)

## Definition

**Data Abstraction and Data Independence:** DBMS comprise of complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

- There are 3 levels of data abstraction.
  - Physical Level
  - Logical/ Conceptual Level.

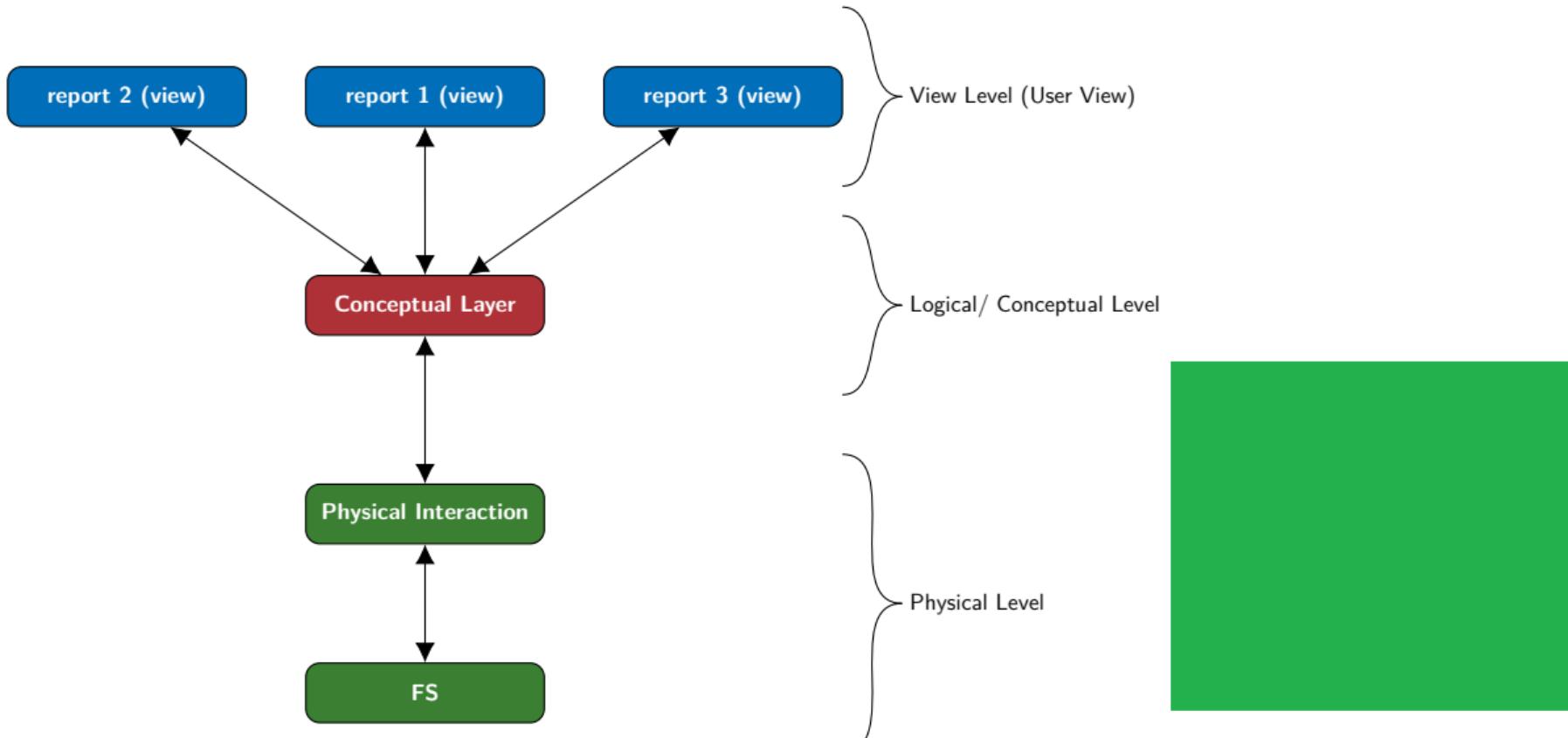
# Data Layers (Abstraction)

## Definition

**Data Abstraction and Data Independence:** DBMS comprise of complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hide irrelevant details from the users. This approach simplifies database design.

- There are 3 levels of data abstraction.
  - Physical Level
  - Logical/ Conceptual Level.
  - View Level.

# Data Layers (Abstraction)



# Physical level

- **Physical level (Internal):**



# Physical level

- **Physical level (Internal):**
  - Lowest level.



# Physical level

- **Physical level (Internal):**
  - Lowest level.
  - Describes how data is stored.



# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.



# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.
- It allows you to modify the lowest level (Physical part) without any change in the logical schema. These changes could be

# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.
- It allows you to modify the lowest level (Physical part) without any change in the logical schema. These changes could be
  - Using a new storage device.



# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.
- It allows you to modify the lowest level (Physical part) without any change in the logical schema. These changes could be
  - Using a new storage device.
  - Change the structure of the data used for storage.

# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.
- It allows you to modify the lowest level (Physical part) without any change in the logical schema. These changes could be
  - Using a new storage device.
  - Change the structure of the data used for storage.
  - Change the file type or use different storage structures.

# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.
- It allows you to modify the lowest level (Physical part) without any change in the logical schema. These changes could be
  - Using a new storage device.
  - Change the structure of the data used for storage.
  - Change the file type or use different storage structures.
  - Changing the access method.

# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.
- It allows you to modify the lowest level (Physical part) without any change in the logical schema. These changes could be
  - Using a new storage device.
  - Change the structure of the data used for storage.
  - Change the file type or use different storage structures.
  - Changing the access method.
  - Modifying indexes.

# Physical level

- **Physical level (Internal):**

- Lowest level.
- Describes how data is stored.
- Describes the data structure.
- It allows you to modify the lowest level (Physical part) without any change in the logical schema. These changes could be
  - Using a new storage device.
  - Change the structure of the data used for storage.
  - Change the file type or use different storage structures.
  - Changing the access method.
  - Modifying indexes.
  - Change the compression algorithm or hashing technique.

# Physical level

## Example

- Database contains product information.

# Physical level

## Example

- Database contains product information.
- Physical layer describes

# Physical level

## Example

- Database contains product information.
- Physical layer describes
  - Storage mechanism and the blocks (bytes, gigabytes, terabytes etc.).

# Physical level

## Example

- Database contains product information.
- Physical layer describes
  - Storage mechanism and the blocks (bytes, gigabytes, terabytes etc.).
  - The amount of memory used.

# Physical level

## Example

- Database contains product information.
- Physical layer describes
  - Storage mechanism and the blocks (bytes, gigabytes, terabytes etc.).
  - The amount of memory used.
  - Usually this layer abstracted from the programmers.

# Logical level

- **Logical level (Conceptual):**



# Logical level

- **Logical level (Conceptual):**
  - Intermediate level.



# Logical level

- **Logical level (Conceptual):**
  - Intermediate level.
  - Describes what data is stored.



# Logical level

- **Logical level (Conceptual):**

- Intermediate level.
- Describes what data is stored.
- Describes what is the relationship between the stored data.

# Logical level

- **Logical level (Conceptual):**

- Intermediate level.
- Describes what data is stored.
- Describes what is the relationship between the stored data.
- It allows you to change the logical view without altering the external view, API, or programs. These changes could be

# Logical level

- **Logical level (Conceptual):**

- Intermediate level.
- Describes what data is stored.
- Describes what is the relationship between the stored data.
- It allows you to change the logical view without altering the external view, API, or programs. These changes could be
  - Add new table.

# Logical level

- **Logical level (Conceptual):**

- Intermediate level.
- Describes what data is stored.
- Describes what is the relationship between the stored data.
- It allows you to change the logical view without altering the external view, API, or programs. These changes could be
  - Add new table.
  - Change the records merge or delete without affecting the running applications.

# Logical level

- **Logical level (Conceptual):**

- Intermediate level.
- Describes what data is stored.
- Describes what is the relationship between the stored data.
- It allows you to change the logical view without altering the external view, API, or programs. These changes could be
  - Add new table.
  - Change the records merge or delete without affecting the running applications.
  - Change attribute (Add, delete) to existing table.

## Example

- Database contains product information.

# Logical level

## Example

- Database contains product information.
- Logical Layer describes

# Logical level

## Example

- Database contains product information.
- Logical Layer describes
  - The product fields and their data types.

# Logical level

## Example

- Database contains product information.
- Logical Layer describes
  - The product fields and their data types.
  - How this product interact with other entities in the database.

# Logical level

## Example

- Database contains product information.
- Logical Layer describes
  - The product fields and their data types.
  - How this product interact with other entities in the database.
  - The programmers design this level based on the business knowledge and the requirements.

# View level

- **View level (External):**



# View level

- **View level (External):**
  - Highest level.



## View level

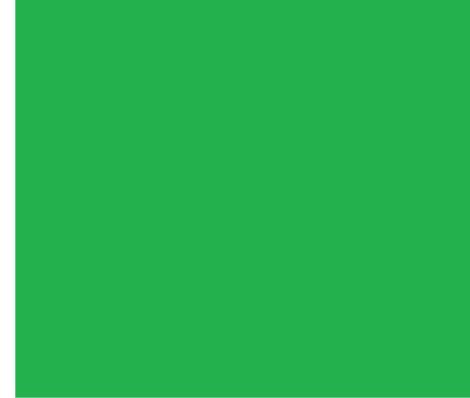
- **View level (External):**
  - Highest level.
  - View of the data stored?



# View level

- **View level (External):**

- Highest level.
- View of the data stored?
- Designed for category of users needs.



# View level

- **View level (External):**

- Highest level.
- View of the data stored?
- Designed for category of users needs.
- It is the final interface for the user.

# View level

- **View level (External):**

- Highest level.
- View of the data stored?
- Designed for category of users needs.
- It is the final interface for the user.
- It could be extended or hidden based on user's role.



- **View level (External):**

- Highest level.
- View of the data stored?
- Designed for category of users needs.
- It is the final interface for the user.
- It could be extended or hidden based on user's role.
- Not all the views is extended to all users and there is an authentication based on the category.



## Example

- Database contains product information.

## Example

- Database contains product information.
- It could be designed to show the sales of product in specific region.

## Example

- Database contains product information.
- It could be designed to show the sales of product in specific region.
- We might hide information about some products based on the teams or users.

# Data solution thinking (Summary)

Let's answer our previous the question, How can we solve data challenges?



# Data solution thinking (Summary)

- Let's split the problem based on the data layers.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - View layer



# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - View layer
    - When we need to add/remove/create new reports it is usually view layer.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - View layer
    - When we need to add/remove/create new reports it is usually view layer.
    - We don't need to change the logical or physical layer to support the view layer.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Logical Layer



# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Logical Layer
    - When you have missing sources into your logical layer and you need to add this source and its structure.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Logical Layer
    - When you have missing sources into your logical layer and you need to add this source and its structure.
    - There is a performance issue in the existing reports and you need to change in the model. For example, reduce the join by creating new join table (*materialized view*).

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Logical Layer
    - When you have missing sources into your logical layer and you need to add this source and its structure.
    - There is a performance issue in the existing reports and you need to change in the model. For example, reduce the join by creating new join table (*materialized view*).
    - Update the data type or the existing relation which could help to fix some data or performance issues.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Physical Layer



# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Physical Layer
    - When our problem is hardly or impossible to be fix by optimize the query (view)/ logical layer it is time for physical change.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Physical Layer
    - When our problem is hardly or impossible to be fix by optimize the query (view)/ logical layer it is time for physical change.
    - If we need to change your storage/compression/structure/access technique.

# Data solution thinking (Summary)

- Let's split the problem based on the data layers.
  - Physical Layer
    - When our problem is hardly or impossible to be fix by optimize the query (view)/ logical layer it is time for physical change.
    - If we need to change your storage/compression/structure/access technique.
    - If we need to change the data orientation structure from row to column or key-value storage, It is time to change the physical layer.

## From Operational DB to DWH

# From Operational DB to DWH

- Data could be a product for some companies.

# From Operational DB to DWH

- Data could be a product for some companies.
- It could be decision support for other products or businesses.



# From Operational DB to DWH

- Data could be a product for some companies.
- It could be decision support for other products or businesses.
- Reporting the results after passing the data life-cycle will be from storage (Database).

# From Operational DB to DWH

- Data could be a product for some companies.
- It could be decision support for other products or businesses.
- Reporting the results after passing the data life-cycle will be from storage (Database).
- There are some challenges facing the people who work on data management backend:



# From Operational DB to DWH

- Data could be a product for some companies.
- It could be decision support for other products or businesses.
- Reporting the results after passing the data life-cycle will be from storage (Database).
- There are some challenges facing the people who work on data management backend:
  - Performance.



# From Operational DB to DWH

- Data could be a product for some companies.
- It could be decision support for other products or businesses.
- Reporting the results after passing the data life-cycle will be from storage (Database).
- There are some challenges facing the people who work on data management backend:
  - Performance.
  - Integration.

# From Operational DB to DWH

- Data could be a product for some companies.
- It could be decision support for other products or businesses.
- Reporting the results after passing the data life-cycle will be from storage (Database).
- There are some challenges facing the people who work on data management backend:
  - Performance.
  - Integration.
  - Applying analytical functions.

# From Operational DB to DWH

- Data could be a product for some companies.
- It could be decision support for other products or businesses.
- Reporting the results after passing the data life-cycle will be from storage (Database).
- There are some challenges facing the people who work on data management backend:
  - Performance.
  - Integration.
  - Applying analytical functions.
- Vendors who are working to solve the above challenges creating their own product of DWH and their ultimate work is to optimize the above points.

# Motivation to Data Warehouse (DWH)

## Definition (What is Data Warehousing?)

A DWH is defined as a technique for collecting and managing data from varied sources to **provide meaningful business insights**. It is a blend of technologies and components which aids the strategic use of data.

The real concept was given by Inmon Bill. He was considered as a father of the DWH. He had written about a variety of topics for building, usage, and maintenance of the warehouse & the Corporate Information Factory

# Motivation to Data Warehouse (DWH)

- The DWH is not a product but an environment.

# Motivation to Data Warehouse (DWH)

- The DWH is not a product but an environment.
- It is a process of transforming data into information and make it available to users in a **timely manner** to make a difference.



# Motivation to Data Warehouse (DWH)

- The DWH is not a product but an environment.
- It is a process of transforming data into information and make it available to users in a **timely manner** to make a difference.
- It is an architectural construct of an information system which provides users with current and historical decision support information which is difficult to access or present in the traditional operational data store.

# Motivation to Data Warehouse (DWH)

- The DWH is not a product but an environment.
- It is a process of transforming data into information and make it available to users in a **timely manner** to make a difference.
- It is an architectural construct of an information system which provides users with current and historical decision support information which is difficult to access or present in the traditional operational data store.
- The DWH is the core of the BI system which is built for data analysis and reporting.

# Motivation to Data Warehouse

Data warehouse system is also known by the following names:

- Decision Support System (DSS).
- Business Intelligence Solution.
- Executive Information System.
- Management Information System.
- Analytic Application.
- Data Warehouse.

## Differences Between DWH and Operational DB

# DWH vs Operational databases

Metric	Transactions DB	DWH
Volume	GB/TB	TB/PB
Historical rows	Short-term <1000M	Long-Term 1000M>
Orientation	Product	Subject or multi products
Business Units	Product team	Multi organizational units
Normalization	Normalized	Not required (De-normalized in many use cases)
Data Model	Relational	Star Schema or Multi-dim
Intelligence	Reporting	Advanced reporting and Machine Learning
Use cases	Online transactions & operations	Centralized storage (360°)

# Transnational DB Use cases



# Transnational DB Use cases



# DWH Use cases



# DWH Use cases



# DWH Use cases



## Types of DWH

# Motivation to Data Warehouse

## Types of Data Warehouse

**Enterprise Data Warehouse (EDWH)** It provides decision support service across the enterprise. It offers a unified approach for organizing and representing data (DWH Model). It offers data classifications according to the subject with privileges policy.

**Operational Data Store (ODS):** is a central database that provides an up-to-date (real-time) data from multiple transnational systems for operational reporting into a single DWH.

**Data Mart:** A data mart is a subset of the data warehouse. It specially designed for a particular line of business, such as sales, finance, sales or finance. In an independent data mart, data can collect directly from sources.

# DWH vs ODS vs Data Mart

Metric	DWH	ODS	Data Mart
Latency	Day -1	Real-time	Day -1
Data level	Transnational	Transnational	Summary
Historical	Long-term	Snapshot	Aggregated Long-Term
Size	TB/PB	GB	GB/TB
Orientation	Multi sources	Multi sources	Product
Business Units	Multi organizational units	Product team	Business team

## Use Cases of Operational DB vs DWH

# Use case (Operational DB)

- A telecommunication company named **XTec**.



## Use case (Operational DB)

- A telecommunication company named **XTec**.
- They have lots of systems. One of this systems is a CRM system as example of operational DB.

# Use case (Operational DB)

- A telecommunication company named **XTec**.
- They have lots of systems. One of this systems is a CRM system as example of operational DB.
  - The CRM system handles the customer activities with the company including (sales, change in customer plans, and other activities).

# Use case (Operational DB)

- A telecommunication company named **XTec**.
- They have lots of systems. One of this systems is a CRM system as example of operational DB.
  - The CRM system handles the customer activities with the company including (sales, change in customer plans, and other activities).
  - This system has a backend database (MySQL).



# Use case (Operational DB)

- A telecommunication company named **XTec**.
- They have lots of systems. One of this systems is a CRM system as example of operational DB.
  - The CRM system handles the customer activities with the company including (sales, change in customer plans, and other activities).
  - This system has a backend database (MySQL).
  - CRM team can report their sales and customer activities from their database.

# Use case (Operational DB)

- A telecommunication company named **XTec**.
- They have lots of systems. One of this systems is a CRM system as example of operational DB.
  - The CRM system handles the customer activities with the company including (sales, change in customer plans, and other activities).
  - This system has a backend database (MySQL).
  - CRM team can report their sales and customer activities from their database.
  - Product owner can take a decision based on their system backend reports.

# Use case (DWH)

- What is the need for DWH?



# Use case (DWH)

- What is the need for DWH?
  - This company has other systems for example: billing, charging, signaling.



# Use case (DWH)

- What is the need for DWH?
  - This company has other systems for example: billing, charging, signaling.
  - They need to report information related to the CRM, billing, and signaling source systems in one report.

# Use case (DWH)

- What is the need for DWH?

- This company has other systems for example: billing, charging, signaling.
- They need to report information related to the CRM, billing, and signaling source systems in one report.
- So, they need to ingest (transfer) the data from the source systems to one single database.

# Use case (DWH)

- What is the need for DWH?

- This company has other systems for example: billing, charging, signaling.
- They need to report information related to the CRM, billing, and signaling source systems in one report.
- So, they need to ingest (transfer) the data from the source systems to one single database.
- The decision from the DWH is a **global and strategical decision**.

# Use case (DWH)

- What is the need for DWH?

- This company has other systems for example: billing, charging, signaling.
- They need to report information related to the CRM, billing, and signaling source systems in one report.
- So, they need to ingest (transfer) the data from the source systems to one single database.
- The decision from the DWH is a **global and strategical decision**.
- If the company needs to build a machine learning model which needs data from different sources. They need to load the data from a centralized database rather than read each source alone.

# Use case (DWH)

The Full picture required a DWH. However, we still need the other operational databases for product development perspective.

# Use case (ODS)

- Why do we need the ODS?



# Use case (ODS)

- Why do we need the ODS?
- How does it fit in our system?



# Use case (ODS)

**XTec** has a call center system which handles the customer inquiries.

This system requires some data related to usage, customer information, billing details to be calculated and accumulated in **real-time** to be able to give the customer the right answer for his inquiries.

# Use case (ODS)

- So, What is the challenge for this system?



# Use case (ODS)

- So, What is the challenge for this system?
  - It needs specific information from different source systems.



# Use case (ODS)

- So, What is the challenge for this system?
  - It needs specific information from different source systems.
  - It requires to track the source system database changes or update in real-time.

# Use case (ODS)

- So, What is the challenge for this system?
  - It needs specific information from different source systems.
  - It requires to track the source system database changes or update in real-time.
  - Its functionality is based on the aggregate data not the transactions for example (It needs the total outgoing calls till time or it needs the total charging amounts from prepaid or the available limits from billing if it is postpaid).

## Use case (ODS)

- ODS is based on change data capture (CDC). This approach used to determine the data change and apply action based on this change.



## Use case (ODS)

- ODS is based on change data capture (CDC). This approach used to determine the data change and apply action based on this change.
- ODS uses the real-time aggregations to support the online systems from different source systems.

## DWH Characteristics



# DWH Characteristics

some details about hot vs cold storage,



## Datawarehouse Components

# ETL Process



## ETL vs ELT When? Why?

# Data Models

# What is data model?

Data model is

- An abstract model that organizes elements of data.



# What is data model?

Data model is

- An abstract model that organizes elements of data.
- It describes the objects, entities and data structure properties, semantic, and constraint.



# What is data model?

Data model is

- An abstract model that organizes elements of data.
- It describes the objects, entities and data structure properties, semantic, and constraint.
- It formalizes the relationship between entities.



# What is data model?

Data model is

- An abstract model that organizes elements of data.
- It describes the objects, entities and data structure properties, semantic, and constraint.
- It formalizes the relationship between entities.
- It describes how application (report) API data manipulation.



# What is data model?

Data model is

- An abstract model that organizes elements of data.
- It describes the objects, entities and data structure properties, semantic, and constraint.
- It formalizes the relationship between entities.
- It describes how application (report) API data manipulation.
- It describes the conceptual design of a business or an application with its flow, logic, semantic information (rules), and how things are done.

# What is data model?

Data model is

- An abstract model that organizes elements of data.
- It describes the objects, entities and data structure properties, semantic, and constraint.
- It formalizes the relationship between entities.
- It describes how application (report) API data manipulation.
- It describes the conceptual design of a business or an application with its flow, logic, semantic information (rules), and how things are done.
- It refers to a set of concepts used in defining such as entities, attributes, relations, or tables.

# What is data model?

Data model is not

- a science.
- a static design for each organization.
- a type of database.
- a new invention which needs to be done for each project.

Data model is

- an engineering design practices.
- a general concepts which lead to build full architecture.
- different based on the use case and the database type.
- customizable and we can utilize some of ready built architecture.
- implementing using different ways.
- affecting the information reporting performance and ways.

# Why does data models are important?

- Data models are currently affecting software design.
- It decides how engineers will think about the problem they are solving.

# Data Model Design

# Data Model Design vs Implementation

- You need to build a home. So, how do we design this home?



# Data Model Design vs Implementation

- You need to build a home. So, how do we design this home?
  - Determine if the home is one level or multi-level and decide man bedrooms and bathrooms for each floor. (User needs)

# Data Model Design vs Implementation

- You need to build a home. So, how do we design this home?
  - Determine if the home is one level or multi-level and decide man bedrooms and bathrooms for each floor. (User needs)
  - Hire an architect to put the architecture in more detailed way for example, the size for each room, the distribution of the wireds, where the plumbing fixtures will be placed, etc. (Architecture phase)

# Data Model Design vs Implementation

- You need to build a home. So, how do we design this home?
  - Determine if the home is one level or multi-level and decide man bedrooms and bathrooms for each floor. (User needs)
  - Hire an architect to put the architecture in more detailed way for example, the size for each room, the distribution of the wireds, where the plumbing fixtures will be placed, etc. (Architecture phase)
  - Decide the decorations, colors for each room, carpets, etc.

# Data Model Design vs Implementation

- You need to build a home. So, how do we design this home?
  - Determine if the home is one level or multi-level and decide man bedrooms and bathrooms for each floor. (User needs)
  - Hire an architect to put the architecture in more detailed way for example, the size for each room, the distribution of the wireds, where the plumbing fixtures will be placed, etc. (Architecture phase)
  - Decide the decorations, colors for each room, carpets, etc.
- What do we do for the implementation?

# Data Model Design vs Implementation

- You need to build a home. So, how do we design this home?
  - Determine if the home is one level or multi-level and decide man bedrooms and bathrooms for each floor. (User needs)
  - Hire an architect to put the architecture in more detailed way for example, the size for each room, the distribution of the wireds, where the plumbing fixtures will be placed, etc. (Architecture phase)
  - Decide the decorations, colors for each room, carpets, etc.
- What do we do for the implementation?
  - Hire a contractor to build (implement the design) the home.

# Data Model Design vs Implementation

- You need to build a home. So, how do we design this home?
  - Determine if the home is one level or multi-level and decide man bedrooms and bathrooms for each floor. (User needs)
  - Hire an architect to put the architecture in more detailed way for example, the size for each room, the distribution of the wireds, where the plumbing fixtures will be placed, etc. (Architecture phase)
  - Decide the decorations, colors for each room, carpets, etc.
- What do we do for the implementation?
  - Hire a contractor to build (implement the design) the home.
  - This phase will implement the design but it also include some detail related to the actual way to build the tools and the material.  
(Physical Design)

# DWH Architecture Overview



# DWH Architecture Overview

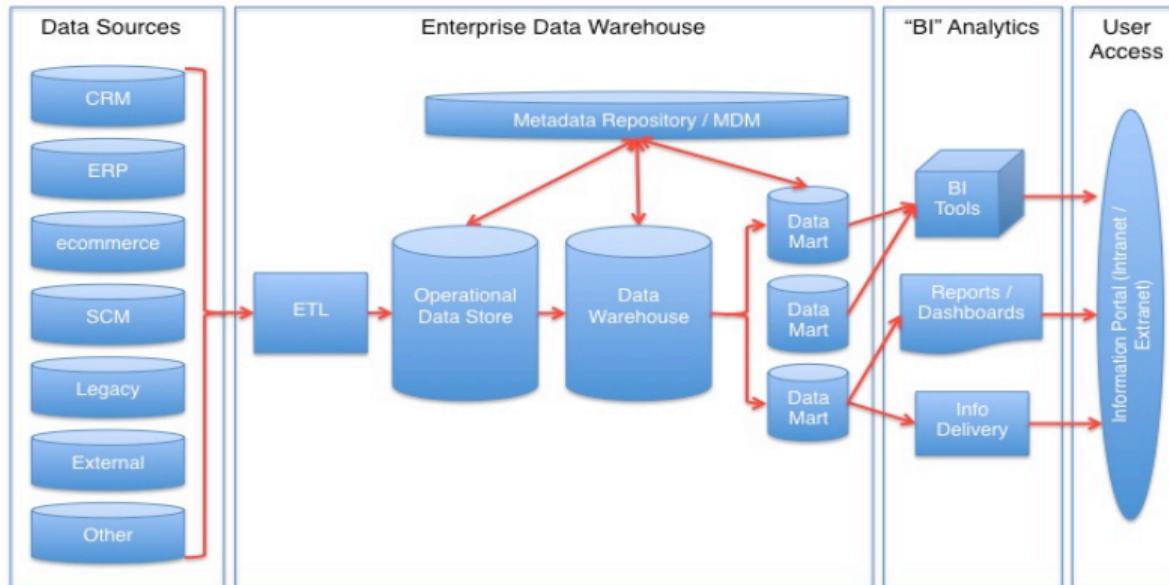


Figure: taken from XXXX

# DWH Architecture Overview

There are mainly three types of Datawarehouse Architectures: -

- Single-tier architecture.
- Two-tier architecture.
- Three-tier architecture.



# Cold storage vs Hot storage

some details about hot vs cold storage,



# Data Encoding and Formats

# Data Models

- Any Big Data solution working based distributed systems.



# Data Models

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Further Readings and Assignment

# Introduction To Distributed Systems

# Chapter Objectives

- Understand the distributed systems concepts.



# Chapter Objectives

- Understand the distributed systems concepts.
- Replication and its usage in distributed systems.



# Chapter Objectives

- Understand the distributed systems concepts.
- Replication and its usage in distributed systems.
- Partitioning and its usage in distributed systems .



# Distributed Systems Concepts

# Distributed Systems Concepts

- Any Big Data solution working based distributed systems.

# Distributed Systems Concepts

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Distributed Systems Architecture

# Distributed Systems Architecture

- Any Big Data solution working based distributed systems.

# Distributed Systems Architecture

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Distributed Systems Challenges

# Distributed Systems Challenges

- Any Big Data solution working based distributed systems.

# Distributed Systems Challenges

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Design Simple Distributed System

# Design Simple Distributed System

- Any Big Data solution working based distributed systems.

# Design Simple Distributed System

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Further Readings and Assignment

# Introduction to Hadoop and Map-Reduce

# Chapter Objectives

- Introduction to Hadoop and its echo-systems.



# Chapter Objectives

- Introduction to Hadoop and its echo-systems.
- Why we need Hadoop?



# Chapter Objectives

- Introduction to Hadoop and its echo-systems.
- Why we need Hadoop?
- Understand the concept of HDFS and Map-Reduce.

# Chapter Objectives

- Introduction to Hadoop and its echo-systems.
- Why we need Hadoop?
- Understand the concept of HDFS and Map-Reduce.
- Developing Map-Reduce applications.

# Chapter Objectives

- Introduction to Hadoop and its echo-systems.
- Why we need Hadoop?
- Understand the concept of HDFS and Map-Reduce.
- Developing Map-Reduce applications.
- Using HiveQL over Map-Reduce.

# Chapter Objectives

- Introduction to Hadoop and its echo-systems.
- Why we need Hadoop?
- Understand the concept of HDFS and Map-Reduce.
- Developing Map-Reduce applications.
- Using HiveQL over Map-Reduce.
- Hadoop advantages and disadvantages with use cases?

# Hadoop Architecture

# Hadoop Architecture

- Any Big Data solution working based distributed systems.

# Hadoop Architecture

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Storage

# Storage

- Any Big Data solution working based distributed systems.



# Storage

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# YARN

- Any Big Data solution working based distributed systems.



- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Hadoop I/O

# Hadoop I/O

- Any Big Data solution working based distributed systems.

# Hadoop I/O

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Processing

# Processing

- Any Big Data solution working based distributed systems.



# Processing

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Map-Reduce

# Map-Reduce

- Any Big Data solution working based distributed systems.



# Map-Reduce

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Map-Reduce Components

# Map-Reduce Components

- Any Big Data solution working based distributed systems.

# Map-Reduce Components

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Word-Count Example

# Word-Count Example

- Any Big Data solution working based distributed systems.



# Word-Count Example

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Pig

- Any Big Data solution working based distributed systems.

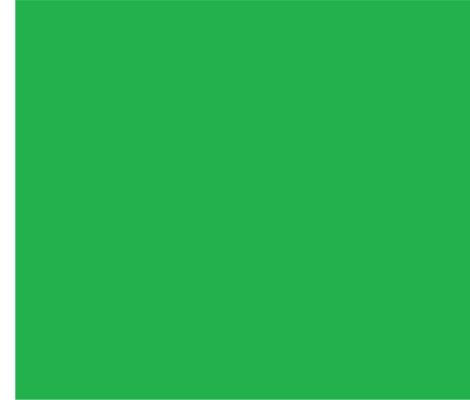


- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Hive

- Any Big Data solution working based distributed systems.



- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# ZooKeeper

- Any Big Data solution working based distributed systems.



- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Further Readings and Assignment

# Functional Programming

Why functional programming commonly used in distributed systems?

# Introduction to Scala

## Further Readings and Assignment

# Spark Framework

# Spark Philosophy towards the Engine and the Programming languages

# Spark Framework: Spark Philosophy towards the Engine and the Programming languages

- Any Big Data solution working based distributed systems.



# Spark Framework: Spark Philosophy towards the Engine and the Programming languages

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Basics

# Spark Framework: Spark Basics

- Any Big Data solution working based distributed systems.



# Spark Framework: Spark Basics

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Basics

- Any Big Data solution working based distributed systems.



# Spark Basics

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Programming using RDDs

# Spark RDD

# Spark Programming using RDDs

- Any Big Data solution working based distributed systems.



# Spark Programming using RDDs

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Spark Working With Key/Value Pairs

# Spark Programming using RDDs

- Any Big Data solution working based distributed systems.

# Spark Programming using RDDs

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Programming using RDDs

- Any Big Data solution working based distributed systems.

# Spark Programming using RDDs

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Datasets/Dataframe

# Spark Datasets/Dataframe

- Any Big Data solution working based distributed systems.

# Spark Datasets/Dataframe

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark SQL

# Spark Datasets/Dataframe

- Any Big Data solution working based distributed systems.

# Spark Datasets/Dataframe

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Dataframes/Datasets vs. RDDs

# Spark Datasets/Dataframe

- Any Big Data solution working based distributed systems.

# Spark Datasets/Dataframe

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark on Production

# Spark on Production

- Any Big Data solution working based distributed systems.

# Spark on Production

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark on Production

- Any Big Data solution working based distributed systems.

# Spark on Production

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark For Batch Processing

# Spark For Batch Processing

- Any Big Data solution working based distributed systems.



# Spark For Batch Processing

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Building custom input and output connector using Spark

# Building custom input and output connector using Spark

- Any Big Data solution working based distributed systems.

# Building custom input and output connector using Spark

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Streaming

# Spark Streaming

- Any Big Data solution working based distributed systems.



# Spark Streaming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Streaming

- Any Big Data solution working based distributed systems.



# Spark Streaming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Streaming

- Any Big Data solution working based distributed systems.

# Spark Streaming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark Streaming

- Any Big Data solution working based distributed systems.



# Spark Streaming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Spark using other Programming Languages

# PySpsark for Python Geeks

# Spark using other Programming Languages

- Any Big Data solution working based distributed systems.

# Spark using other Programming Languages

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark using other Programming Languages

- Any Big Data solution working based distributed systems.

# Spark using other Programming Languages

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# RSpark for R Geeks

# Spark using other Programming Languages

- Any Big Data solution working based distributed systems.



# Spark using other Programming Languages

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark For Data Scientist

# Spark For Data Scientist

- Any Big Data solution working based distributed systems.



# Spark For Data Scientist

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark For Data Scientist

- Any Big Data solution working based distributed systems.

# Spark For Data Scientist

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Spark For Data Scientist

- Any Big Data solution working based distributed systems.



# Spark For Data Scientist

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Spark Graph Dataframe/Graphx

# Spark Graph Dataframe/Graphx

- Any Big Data solution working based distributed systems.



# Spark Graph Dataframe/Graphx

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



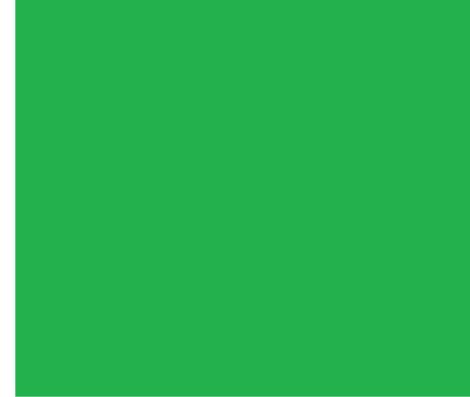
# Spark Graph Dataframe/Graphx

- Any Big Data solution working based distributed systems.



# Spark Graph Dataframe/Graphx

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Tuning your Spark Jobs

# Tuning your Spark Jobs

- Any Big Data solution working based distributed systems.



# Tuning your Spark Jobs

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



# Tuning your Spark Jobs

- Any Big Data solution working based distributed systems.



# Tuning your Spark Jobs

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Further Readings and Assignment

## Real World Applications

# Big Data Development Life Cycle

# Template Concept for Data Engineering

## Template for ETL Application

## Template for QA

## Template for Streaming Applications

# Template for Machine Learning Applications

## Further Readings and Assignment

# Massaging Systems

# Motivation

# Massaging Systems Architecture

## JMS as an example

# Introduction to Kafka

# Kafka Architecture



# Kafka Topics

# Partitions



# Kafka Producers

# Kafka Consumers



# Kafka Connector

# Kafka Custom Connectors

# Kafka Configuration

# Kafka Configuration Optimizations

# Kafka Operations

## Kafka Integration with Enterprise tools

## Further Readings and Assignment

# Data Orchestration



# Motivation

## Enterprise vs Open source tools

## Open source tools (Oozie as an Example)

## Enterprise source tools

## How to choose the right tool?

## Further Readings and Assignment

# NOSQL

# Introduction to NoSQL Databases.

# Cassandra



# Why Cassandra?

# Introducing Cassandra

# The Cassandra Data Model

# Architecture

# Reading and Writing Data

# Integrating Hadoop

## Further Readings and Assignment

# Elastic



## Further Readings and Assignment

# Data Architecture Design

## Further Readings and Assignment

# Appendix

## Appendix A- Shell Programming

# Appendix A- Shell Programming

- Any Big Data solution working based distributed systems.

# Appendix A- Shell Programming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Appendix B- Java Programming

## Appendix B- Java Programming

- Any Big Data solution working based distributed systems.



## Appendix B- Java Programming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Appendix C- Scala Programming

## Appendix C- Scala Programming

- Any Big Data solution working based distributed systems.



## Appendix C- Scala Programming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Appendix D- SQL Programming

## Appendix D- SQL Programming

- Any Big Data solution working based distributed systems.



## Appendix D- SQL Programming

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Appendix E- Oozie Orchestration

## Appendix E- Oozie Orchestration

- Any Big Data solution working based distributed systems.

## Appendix E- Oozie Orchestration

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Appendix F- DWH Concepts and Data Modeling Design

## Appendix F- DWH Concepts and Data Modeling Design

- Any Big Data solution working based distributed systems.

## Appendix F- DWH Concepts and Data Modeling Design

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Appendix G- Machine Learning Concepts Data Engineers

## Appendix G- Machine Learning Concepts Data Engineers

- Any Big Data solution working based distributed systems.

## Appendix G- Machine Learning Concepts Data Engineers

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?



## Appendix H- Docker for Data Engineers

## Appendix H- Docker for Data Engineers

- Any Big Data solution working based distributed systems.

## Appendix H- Docker for Data Engineers

- Any Big Data solution working based distributed systems.
- What is distributed systems in brief?

