|  |
| --- |
| IU |
| Data Warehousing Architecture Types |
| DLMBIDWAT |

# Learning Objectives

Well-organized and managed data enables organizations to make informed decisions, determine root causes of problems, and strategically plan for the future. These days, businesses have access to vast amounts of computational power and massive storage capacities, in addition to increasingly fast communication practices. So, data that are available for managerial and decision support are growing exponentially. Hence, in this course, you will gain an understanding of the characteristics and categorizations of big data. You will also learn about different approaches that can be taken to design and develop an efficacious DWH. Moreover, you will learn about RDBMS-based versus NoSQL-based data warehousing.

You will also be introduced to different types of DWH architecture, i.e., a single-layer, two-layer, and three-layer architecture. Moreover, you will learn where to apply each of these architectural designs most effectively. Additionally, various DWH components, including various types of databases, the ETL process components, data marts and a DWH bus architecture, are expanded upon.

# Unit 1 – Introduction

**Study Goals**

On completion of this unit, you will be able to …

… understand the different characteristics and categorizations of big data.

… explain the challenges related to the management of unstructured big data.

… discuss the characteristics of data warehouses.

… distinguish between RDBMS-based and NoSQL-based data warehousing.

# 1. Introduction

## Introduction

The incessant increase in computational power, communication velocity and storage capacity results in continuously increasing volumes of data being created. These large, complex, and diverse sets of data are referred to as big data. Big data presents immense opportunities. However, big data also brings about enormous challenges in terms of aspects such as storage, analysis, and privacy (Magon, 2014). The data can be in a structured, semi-structured and/or unstructured format.

Structured data is typically generated by business operations and transactions; it is stored in a relational format and in a standard, relational database management system (RDBMS). Data that are generated by electronic systems such as automated machine interactions and electronic mail (e-mail) messages produce semi-structured data. Furthermore, businesses often have need to explore ‘other’ data, e.g., data created on social media and the Internet, atmospheric data from satellites, data generated by analog computers, etc., resulting in unstructured data.

Structured data can be effectively stored and managed in a relational data base management system (RDBMS) and associated relational data warehouse (DWH). In this regard, the DWH provides reliable, queryable data, to support and enhance business decision making, typically through business intelligence (BI) portals. However, it is impractical and too expensive to attempt to store semi-structured and unstructured in a RDBMS-type DWH. The data is organized too differently (Inmon & Linstedt, 2015). However, semi-structured and unstructured data can be effectively stored in NoSQL-based data storage (Foster & Godbole, 2016). It can then also be effectively analyzed and queried.

In this unit, we look at challenges associated with unstructured data. We also explore the characteristics of DWHs. Three approaches are available to model a relational DWH: the Corporate Information Factory (CIF) (Inmon et al., 2001); the Kimball Lifecycle Methodology, i.e., the data mart bus architecture (Kimball et al., 2008); and the Data Vault (DV) Model (Linstedt, 2002). These approaches are positioned and briefly discussed. Moreover, relational and dimensional data modelling techniques that are used to model data in the context of DWHs are explored. Lastly, NoSQL data warehousing is elucidated.

## 1.1 Big Data: Structured, Semi-structured and Unstructured Data Types

The term ‘big data’ refers to data sets that are extremely large, complex, and diverse. It entails data that inundate organizations daily. Big data grows exponentially over time. Hence, it is difficult to analyze, manage and maintain. Big data encompasses some (or all) of the following characteristics: volume, variety, velocity, value, and veracity (Gordon, 2014):

* Volume—the amount of data to store and analyze is so large that it requires special considerations.
* Variety—the data consist of multiple types of data and/or data from multiple sources; accordingly, structured, semi-structured and unstructured forms of data must be considered when storing and/or analyzing it.
* Velocity—new data are produced very fast and operating on ‘old’ (stale) data will thus not be sufficiently valuable.
* Value—the data have a perceived or quantifiable benefit for an organization.
* Veracity—the data’s correctness [and truthfulness] can be assessed.

Big data is furthermore classified as structured, semi-structured and unstructured—these categories are elucidated next.

### Structured Data

Structured data entail data that is organized into a formatted repository. Accordingly, a ‘traditional’ relational **SQL** (Standard Query Language) database is a typical example of **repository** containing structured data. The format of structured data is predictable since patterns occur regularly in the data. Furthermore, structured data records are stored in tables that are made up of rows and columns. The tables contain attributes, keys, and indexes.

**Repository**

A managed directory for storing and describing digital objects for a digital archive.

Structured data is associated with well-defined **metadata** to ensure that it can be effectively analyzed and managed. It can therefore be easily managed by a RDBMS (Inmon & Linstedt, 2015). However, SQL-based relational databases do not scale easily and, as a result, have difficulty dealing with data that increase rapidly in quantity, such as in the case of big data (Gordon, 2014). Consequently, RDMBSs are not suitable for the management of big data.

**Metadata**

Information that defines and describes the structure of data, in order to ease the manipulation thereof.

### Semi-Structured Data

**SQL**

A language that is used to manipulate and retrieve data from relational databases.

Semi-structured data comprises some form of a structure, even though the data does not necessarily explicitly conform to a data model. For example, an email contains semi-structured data, since it contains a structure with a sender, recipient, subject, message header and message while the content of the message consists of structureless text. Hence, an e-mail message is an example of unstructured data. So, semi-structured data does not conform to a fixed and rigid schema and is not stored in a relational database; though, it contains organizational properties that makes it easier (than unstructured data) to manage and analyze. For example, metadata (such as tags and elements) are used to group and store semi-structured data. Unfortunately, the metadata of unstructured data is inadequate, in the sense that it cannot be used to easily automate and manage the data by means of a conventional RDBMS (Marr, 2019).

Semi-structured data is basically repetitive unstructured data—the repetition occurs many times, often in the same structure and/or embodiment. Computer generated reports and files generated by means of machine interactions, such as the metering of a consumer’s energy usage, will typically produce a document in a semi-structured data format. Similarly, file formats such as **XML** (eXtensible Markup Language), **JSON** (JavaScript Object Notation) or CSV (comma-separated values) typically contain semi-structured data.

**JSON**

The format is an open standard that is useful to exchange data—it contains text that is easily readable to store and transmit data objects, consisting of values that are serializable.

**XML**

A markup language and file format that is used to store, transmit and reconstruct arbitrary data, based on a set of rules to encode data in a format that is readable by both humans and machines.

Examples of CSV and JSON format files are shown below. The CSV file in the example indicates the respective country names and sales territories of customers, which are identifiable through a customer id. Thereafter, the image of a JSON file gives the details of a customer that is associated with an order (however, the detail of the order is not included in this file, so this JSON file will be associated through the order number to another file).

Example of a CSV File (Venter, 2022)

Graphical user interface, application, table

Description automatically generated

Example of a JSON File (Venter, 2022)

Text, letter

Description automatically generated

### Unstructured Data

Unstructured data records differ substantially from each other, and neither the form nor the content will be repeated regularly. Unstructured data is the most difficult data type to work with and to manage—it is not organized in any predefined manner and the data does not conform to a predefined data model. Unstructured data is unpredictable and, since it does not contain a structure, is essentially unrecognizable by a computer. Hence, it is rather ‘clumsy’ to access, and long strings of data must be sequentially searched (and parsed) to find a specific unit of data (Inmon & Linstedt, 2015). Though, unstructured data in various formats, e.g., PDF and Word documents, images, texts, media logs, audio and/or video files, photographs, etc., are increasingly used and mined by Business Intelligence (BI) and analytics applications.

A recording of call center conversations is an example of unstructured data that can potentially provide rich organizational insights. Furthermore, data gathered by the Internet of Things (IoT) and social media are forms of unstructured data that are extensively collected and analyzed by organizations.

### Self-Check Questions

1. Please complete the following sentence:

Big data is categorized as structured, semi-structured, or *unstructured* data.

## 1.2 Unstructured Big Data Challenges

Unstructured data can potentially offer a wealth of information and insight by means of associations, comparisons, correlations, etc., of data that have been gathered from various and diverse sources. But it is very challenging to work with. Unstructured data is also resource intensive to mine and analyze. It is raw and unorganized, so it requires some processing and indexing to become searchable. Unstructured data is vastly different to the conventional and relational data sets that we have become accustomed to. Various toolsets that can effectively store and manipulate traditional, relational data sets have been readily available for a while already. However, unstructured data demands different and specialized skills and systems. Specific challenges in terms of storage and analysis of unstructured data, as well as privacy aspects to consider, are detailed next.

### Storing of Unstructured Data

Relational databases and DWHs cannot store unstructured data, since this type of data is not associated with the typical format of tables, rows and columns. Furthermore, unstructured data takes up vast amounts of storage space, so it cannot be stored cost effectively using a (commercial) RDBMS. Instead, unstructured data must be stored through alternative ways, using inexpensive or distributed storage systems (Inmon & Linstedt, 2015). Consequently, unstructured data are stored using specialized tools, NoSQL databases and data lakes.

According to Kimball and Ross (2013) big data systems should be scalable to support **petabytes** of data. Furthermore, big data should ideally be stored in the same format that it was originally captured in, whilst query and analysis applications ought to be supported without having to convert and/or move the data. Various new data types that are emerging (e.g., collections of name-value pairs, waveforms, images, etc.) should also be supported. Moreover, the data must be loaded at extremely high rates to have it quickly ready for analysis. Lastly, Kimball and Ross (2013) assert that big data systems must be able to effectively process complex user-defined functions (UDFs).

**Petabytes**

A multiple of a byte, where one petabyte is equal to one quadrillion bytes.

### Analysis of Unstructured Data

Unstructured data is difficult to analyze because the data records are not uniform in terms of shape and size. Furthermore, records that are composed of text are difficult to parse and, consequently, also difficult to analyze. There are technologies available to assist in this regard though: Hadoop can read and parse unstructured data. It can also place the output in any of several available formats. When using Hadoop, the output can remain in the form of the selected (individual) records. Alternatively, the output can also be merged with other records (Inmon & Linstedt, 2015).

When working with unstructured data in the form of text records, the text must first be contextualized so that it can be effectively analyzed—as words can have different meanings and/or interpretations, depending upon the context in which they are used (Inmon & Linstedt, 2015). Technologies such as **textual disambiguation** or MapReduce are useful to contextualize text records. However, these technologies are most effective when used with data that is semi-structured.

**Textual disambiguation**

An analogous process in structured processing, i.e., extract, transform and load (ETL) that reads raw text, and transforms and writes the text to a normalized analytical database, while keeping it linked to the original source.

### Privacy Concerns

Big data analytics seeks to store and process large amounts of unstructured data quickly and accurately. Big data is extracted from various public and private sources, e.g., social media, personal and business accounts used to access the internet, mobile phones as well as health care systems. Data gathered from these systems are used for various purposes, e.g., it can be used to optimize user interactions with a system; or, it can be used to improve the underlying systems, such as when patient records are used to optimize and digitize a hospital’s health care system. However, regardless of what the data is used for, the privacy of the individuals whose data is being used must be considered. It is important to be cognizant that the use of such data can easily result in privacy breaches. Hence, precautions must be taken to ensure the privacy of individuals and organizations. For instance, if the personal information of a hospital patient is accessed, privacy preservation analysis of his/her data must be performed, and the data must be anonymized or encrypted prior to analysis. Though, depending on the format of data elements, it can be very challenging to identify sensitive attributes and anonymize data (Mehta & Rao, 2016).

### Self-Check Questions

1. 1. Please mark the correct statement.
2. Relational databases cannot store unstructured data, since unstructured data...

* *is not associated with tables, rows and columns*
* can easily result in privacy breaches
* must first be contextualized
* contains too many sensitive attributes

## 1.3 Characteristic of Data Warehouses

A DWH provides organizational and managerial decision support—it aims to provide a “single version of the truth” by being the “the basis for *believable* corporate data” (Inmon & Linstedt, 2015, p. 98). Furthermore, Inmon et al. (2001, p. 8) defines a DWH as “a subject-oriented, integrated, time-variant (temporal), and nonvolatile collection of summary and detailed data used to support the strategic decision-making process for the enterprise”. These terms are explained as follows (Inmon et al., 2001):

* Subject-orientated—a DWH is to be organized according to a company’s major entities, e.g., customers, products, vendors, orders, transactions, etc. Hence, it must not be functional or application oriented. Instead, the DWH must focus on the organization’s important business measures.
* Integrated—the data must be physically unified and cohesively stored in the DWH. In view of that, the following aspects ought to be commonly defined throughout the DWH: key structures, structures that will be used for encoding and decoding, definitions of data, data layouts, data relationships and naming conventions. The data will also be altered to be standardized, and to ensure an integrated foundation for analytics.
* Time-variant—the records in the DWH will be, relative to a moment in time, accurate. Moreover, the stored data is historical, so it can be analyzed and compared over time.
* Nonvolatile—changes are captured as time-variant snapshot. This implies that new snapshots are added to reflect changes while all the original records are kept as-is. Historical records therefore remain unchanged.
* Summary and detailed data—detailed data reflect atomic-level transactions, e.g., product usage and inventory movement. Summary data is summarized and/or calculated data, e.g., the average total of products sold monthly.

According to Kimball et al. (2008, p. 602), on the other hand, a DWH is “The queryable data in your DW/BI system”; they further define the DW/BI system as “The complete end-to-end data warehouse and business intelligence system”, thus linking a DWH and BI together to reinforce the dependency between the queryable data in the DWH, and the “value-add analytics” offered by BI applications. Accordingly, Kimball and Ross (2013) state that a DWH entails a database that is typically constructed from multiple source databases, which is designed for query and analysis. A DWH must therefore be optimized for high-performance user queries. As an example, when the user searches through trillions of historical data and transactions, s/he must get the correct answer swiftly. The answer must also be contextual, yet concise and compressed.

Furthermore, Linstedt developed a relational DWH structure that he refers to as a Data Vault (DV). Linstedt (2015a) defines the DV as “a detail oriented, historical tracking and uniquely linked set of normalized tables that support one or more functional area of business” (Data Vault Definition section). So, a DV structure intends to keep detailed track of historical data while also providing flexibility (Yessad & Labiod, 2016). Users access DV data through data mining or BI applications and use it for decision support (Linstedt, 2002).

From the above, there are three schools of thought in terms of the core aim of a DWH. Accordingly, the fundamental beliefs that Inmon versus Kimball versus Linstedt hold regarding the DW environment dictate how it ought to be designed, developed, and managed. Their respective views are represented by Inmon’s Corporate Information Factory (CIF); the Kimball Lifecycle methodology; and Linstedt’s Data Vault (DV). The approaches are deemed best practices, and therefore briefly positioned next.

### The Corporate Information Factory

From Inmon’s perspective, a DWH is designed and developed using the so-called top-down approach (Breslin, 2004). The top-down DWH architecture includes a central repository in the form of a comprehensive normalized relational database along with various departmental data marts that are created and supported from the central DWH (Inmon et al., 2001). The CIF-architecture provides a logical framework to deliver BI. It is built around a central, enterprise-wide DWH. The Inmon DWH approach is considered to be data-driven (instead of business-driven) since it does not explicitly consider specific business requirements and business processes prior to loading all the organizations data into a central DWH (Yessad & Labiod, 2016). Entity-relationship diagrams (ERDs) are used to model the data. The CIF entails a normalized (3NF) data model.

Inmon et al. (2001) state that a CIF is a generic structure that is uniquely implemented by different organizations. Accordingly, each CIF is uniquely dimensionally shaped by the specific company’s business, culture, economics, and technology. So, it represents a physical embodiment of the company’s ‘information ecosystem’. When using the CIF-architecture, the departmental data marts (containing information of individual business areas) and dimensional OLAP (online analytical processing) cubes (to enable fast querying of the data by business users) are distinctly separated from the central DWH. Hence, the central DWH contains all the atomic, detailed data of the organization. The individual DW components such as data marts and cubes are explained in detail later in this Unit. The following image shows a basic reference framework of a CIF.

Framework for a CIF (Venter, 2022)

Diagram, schematic

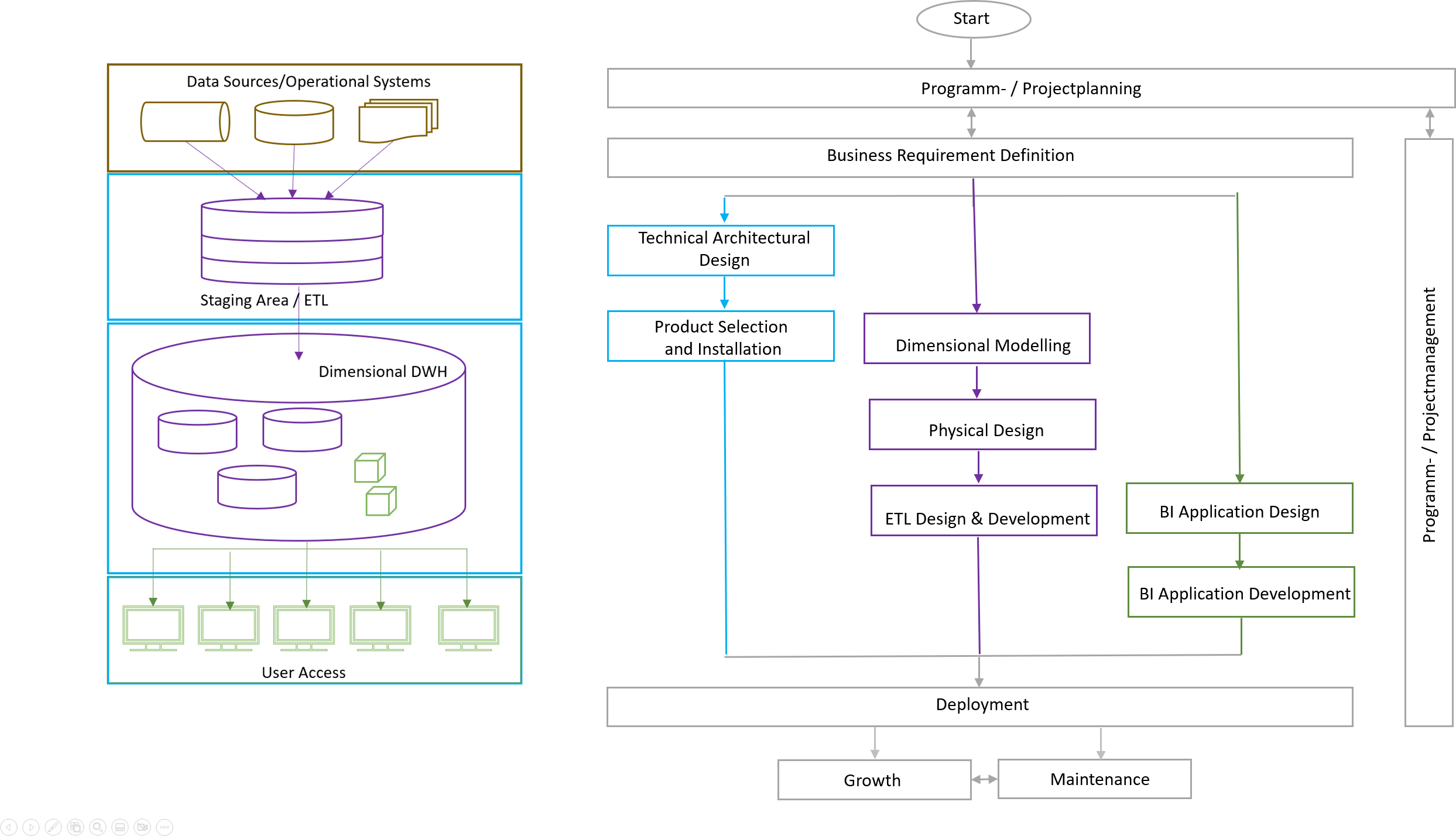
Description automatically generated

### The Kimball Lifecycle Methodology

The Kimball method applies a so-called bottom-up approach to design and develop the bottom-up DWH architecture (Breslin, 2004). Kimball et al. (2008) propose the Kimball Lifecycle methodology as a framework that comprises various program/project planning and management activities. These are aimed at designing, developing, implementing, and maintaining the DW/BI system. The DWH that is developed with this approach is also known as the data mart bus DWH.

The process includes the following activities: gathering of business requirements; the technical architecture design; the logical design (i.e., dimensional modeling); the design of the physical databases; the design and development of the extract, transform and load (ETL) architecture and system; the design and development of BI applications; and lastly the deployment, growth, and maintenance of the DW/BI system (Kimball, 2008). The process phases and activities are shown in the figure below.

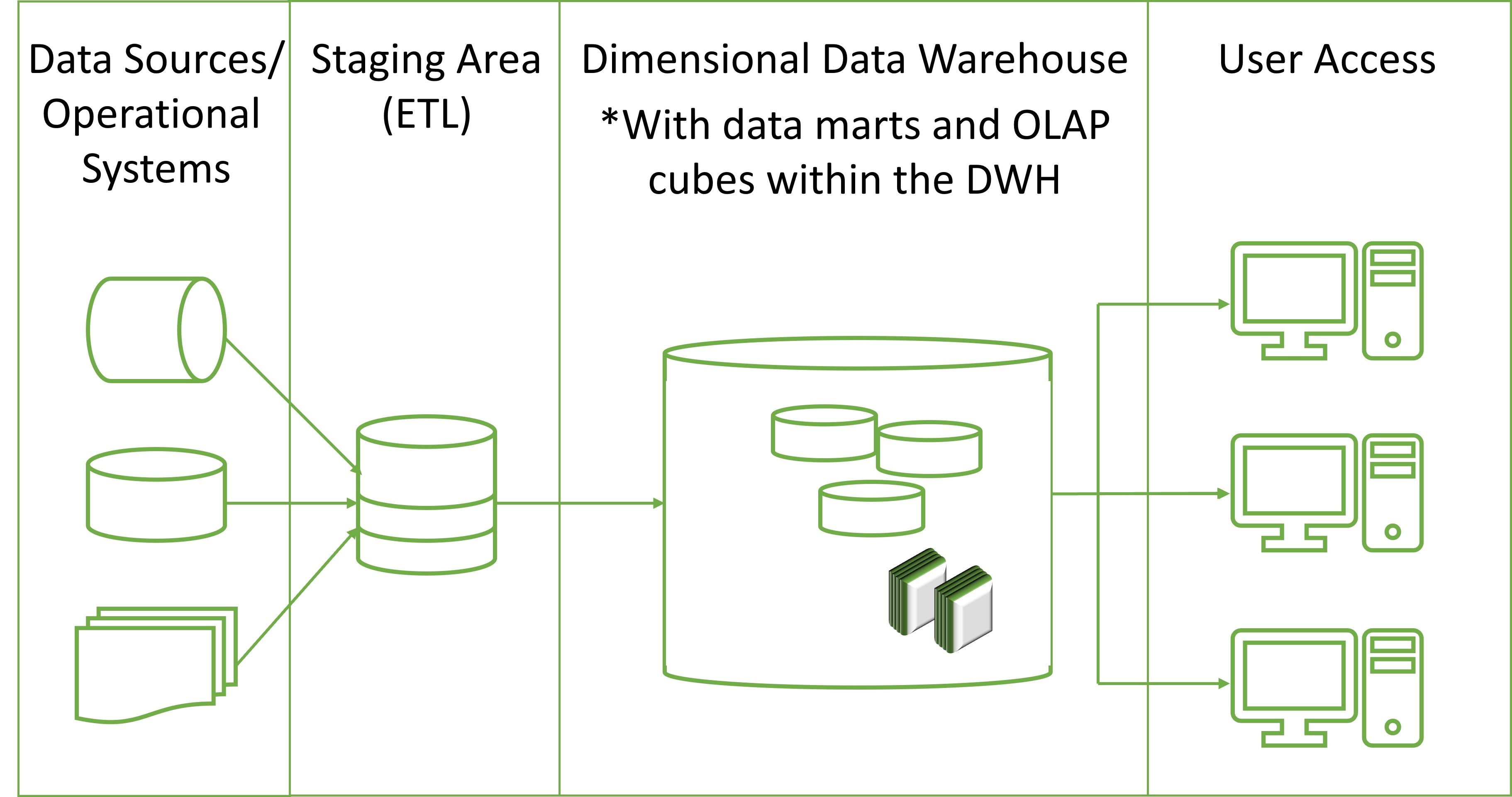
The Kimball Lifecycle Methodology (Vaas, 2022)



The data mart bus DWH is designed using a business-oriented (rather than a data-driven) approach. Dimensional modeling principles are applied to model the data in a denormalized format. When using the Kimball approach, the DWH entails a set of consistent departmental data marts that are based on conformed (uniform) descriptive entities called dimensions. These dimensions are associated with the transaction records that are created when a business process is executed by means of entities that are called fact tables. A DWH also includes multidimensional views of data in the form of cubes. Business users use the views (i.e., the cubes) to access and browse data in a DWH (Yessad & Labiod, 2016).

Since the Kimball method develops the DWH incrementally, enterprise-wide cohesion is achieved through a bus matrix that lists all the organizational business processes and cross-references them with the associated descriptive data, which are called dimensions (Breslin, 2004). The elements that are specific to the Kimball approach are also explained in detail later in this Unit. The following image shows a basic reference framework for the Kimball approach.

Framework for the Kimball Approach (Venter, 2022)



### Linstedt’s Data Vault

The DV architecture aims to overcome apparent shortcomings in the CIF and Kimball approaches. Linstedt (2015a) refers to the DV as a hybrid approach that encompasses the best features of the 3NF CIF DWH and Kimball’s data mart bus DWH that follows dimensional star schema modeling. Hence, the DV follows a hybrid approach where the DWH is designed following a bottom-up design approach; yet it entails a top-down DWH architecture.

Advantages of this approach include the following: Several tables can be loaded simultaneously in parallel, so data can be loaded quickly. Furthermore, dependencies between tables are decreased during the load process. The ingestion process is also simplified by the leveraging of inserts only—they load quicker than upserts or merges. Linstedt (2015a) explains further that, in a DV architecture, business-representative data are received from sources, delivered to a preparation (or staging) area, then to a DV area that comprises the enterprise DWH and business DWH, and lastly to data marts and cubes. Metrics and metadata are collected throughout the process. A DV design separates the structural information (in an enterprise DWH) and descriptive information (in a business DWH) to ensure flexibility in case of organizational changes. The data are not processed in a DV, so data sources can be traced, and historical data remain intact. The DV employs a three-tier architecture that separates the (raw, unprocessed) DWH from end-users and from the data mining layers (Yessad & Labiod, 2016). The image below shows a basic reference framework of the DV approach.

Framework for a Data Vault (Venter, 2022)

Diagram

Description automatically generated

### Features of an Efficacious DW

An efficacious DWH is designed in such a way that it achieves several goals. According to Kimball and Ross (2013) a DWH brings together data from various business sources, and transforms the data to give users easy, timely and consistent access to credible information, i.e., information that is cleansed and quality assured. Furthermore, the DWH should be adaptable, i.e., the DWH must be designed so that it can gracefully adapt when any business needs, business conditions, data and/or underlying technologies change. A DWH must also keep information safe and secure and ensure that access to information is effectively controlled. The DWH should be an authoritative, trustworthy foundation that an organization can use to make informed and improved decisions, and so, business users must accept the DW/BI system to deem it successful (Kimball & Ross, 2013). The DWH must be designed to remain flexible and scalable. In addition, it must consistently meet the needs of the organization (Linstedt, 2015).

## Self-Check Questions

1. Please complete the following sentence.

A data warehouse must provide an authoritative and trustworthy foundation for the organization to make *informed* and *improved* decisions.

## 1.4 RDBMS-Based Data Warehousing

A RDBMS entails a set of programs that facilitates the management of a relational database. It enables the definition of data elements and the manipulation of data. Furthermore, it ensures secure access to the data and supporting systems. Relational database structures have been a leading database approach since the 1970s (Foster & Godbole, 2016). A RDBMS-based DWH is implemented using a relational approach. Examples are Oracle, MS SQL Server and IBM D2.

RDBMS-based DWHs fulfill the ACID (atomicity, consistency, isolation, and durability) requirements. The ACID attributes contribute to ensure data integrity. They are explained as follows (Mullins, 2017):

* Atomicity—preserves the comprehensiveness of a business process. Hence, in the case where some of the instructions of a transaction failed, the entire transaction will be discarded, as if none of the instructions happened.
* Consistency—ensures that all the data in the database is accurate. The state of the data must be considered before and after a transaction occurs.
* Isolation—a locking mechanism is used so that transactions run in parallel, but it appears as if only one transaction is running at a time. So, concurrent transactions will not have visibility in terms of database modifications that have not yet been committed.
* Durability—ensures that data can survive failures. It means that the state of the data will not be impacted by an outage or failure, i.e., when a transaction ends abruptly and abnormally.

### DWH Holding Components

A DWH can include some, or all, the following data holding components: the central or core DWH (C-DW), operational data store (ODS), data marts, and a staging area. When built upon a relational data model, the affected schema contains **granular data** only. The C-DW and ODS consist of relational data models so that they can store data at the highest level of data granularity, as needed by the supported business departments. The C-DW contains all organizational data while the ODS stores loosely integrated operational data that are being prepared for ETL, so that it often serves as a preliminary stage for the C-DW or other components. Furthermore, if it is useful for integration and transformation process, a staging area may also consist of relational data models. However, data marts will be built on multi-dimensional data models. Multi-dimensional models allow for the creation of simpler joins, data aggregation and the implementation of hierarchies in BI-tools.

**Granular data**

The lowest and most detailed level of data captured by a business process, also referred to as atomic data.

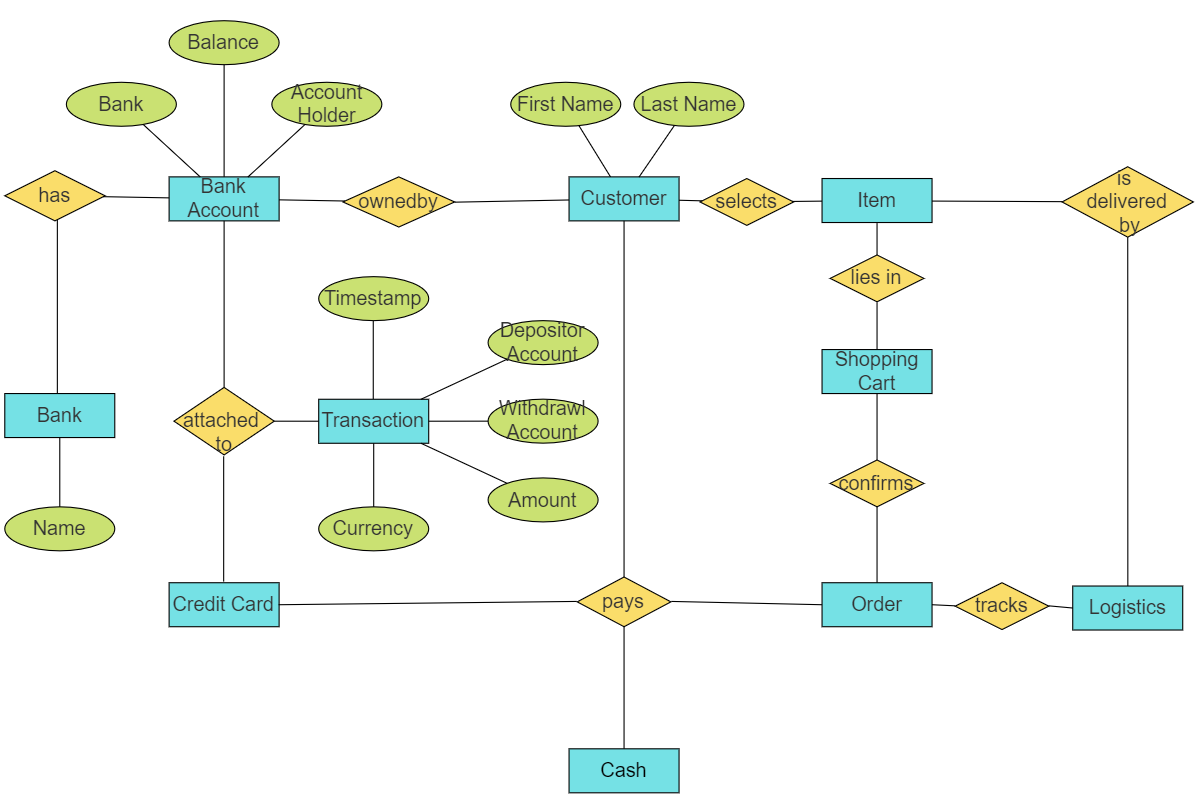
### Relational Data Modelling

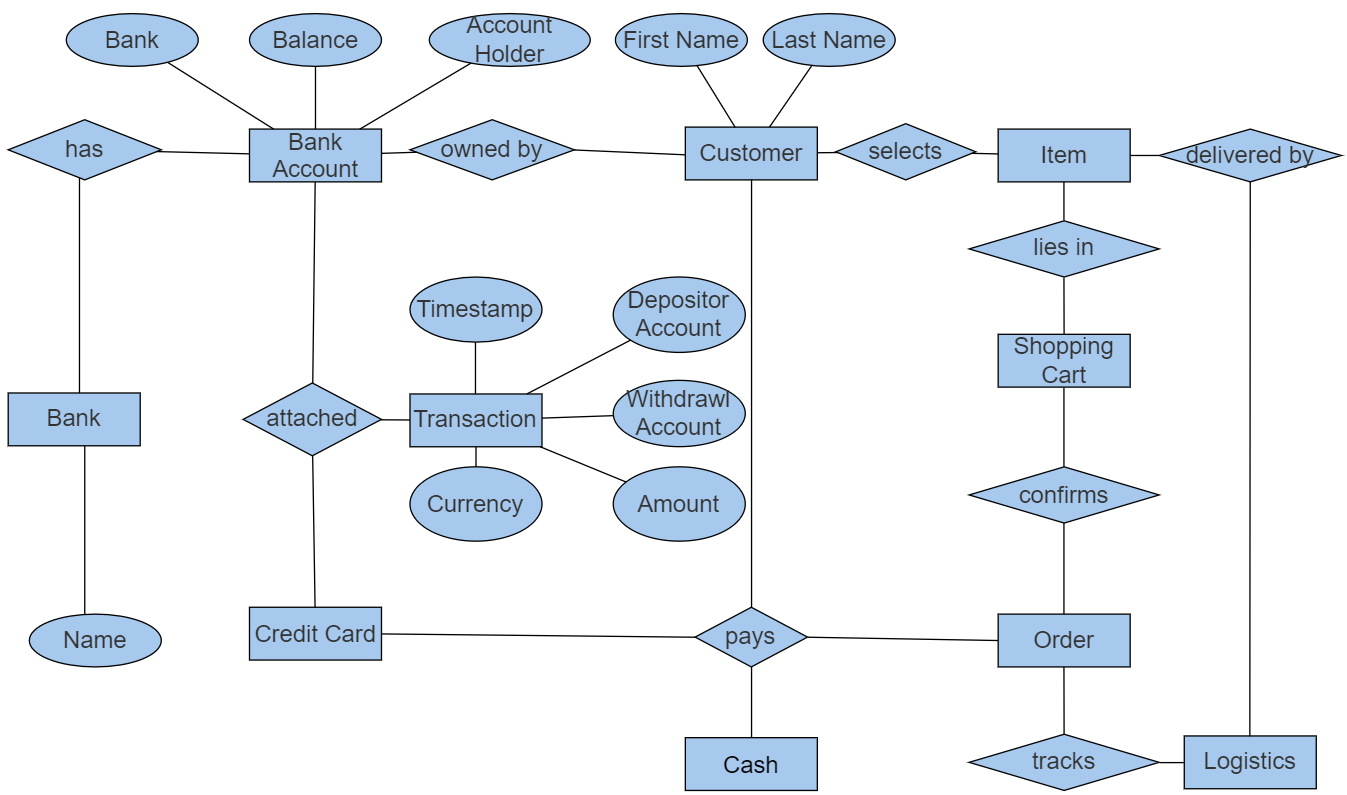
When all the granular data of an organization is identified, it can be abstracted to the highest meaningful level. As an example, an organization that has different types of customers, such as foreign, corporate, and governmental customers, may have need to differentiate between them for purposes such as marketing, legal and reporting. Therefore, a ‘customer’ can be used as an abstraction and an entity ‘customer’ can be created so that all the relevant information of customers can be tied together (Inmon & Linstedt, 2015).

Defining these organizational entities facilitates the design of a predefined schema. It subsequently simplifies the organized classification, structuring and storage of data in associated tables, columns, and rows. In such a schema, the data’s consistency and integrity is assured. When all the relevant organizational entities have been defined, they must be linked together in a data model. An entity relationship diagram (ERD) is useful to show entities and their relationships to one another. An ERD entails a high-level statement of all entities and their relationships but does not include the details contained within the entities (Inmon & Linstedt, 2015). Hence, ERDs visually illustrate relationships between entities. Different visualization types are used in practice. The notation that is applied defines the visualization type. As a result, different notations are used in industry to model databases, e.g., notations from Chen and Barker—these notations are detailed by Brumm (2022).

Bill Inmon is widely known as the “Father of Data Warehousing” (Breslin, 2004, p. 7). It is thus noteworthy that, in the context of DWH modelling, he proposes the following three-level approach: First, draw a high-level ERD that indicates the entities and their relationships (here, any ERD can be used). Second, identify the specific underlying details of each entity and detail that in the associated second (mid-level) data model, i.e., the data item set (DIS). Lastly, the third level entails the physical model (Breslin, 2004). It is illustrated next: The figure below illustrates the first level, as per the explanation above. It shows the high-level ERD of a customer that can order an item. S/he pays with a credit card, which will be verified, or with cash. Next, a shopping cart is created. Then, upon confirmation, the order is dispatched and delivered.

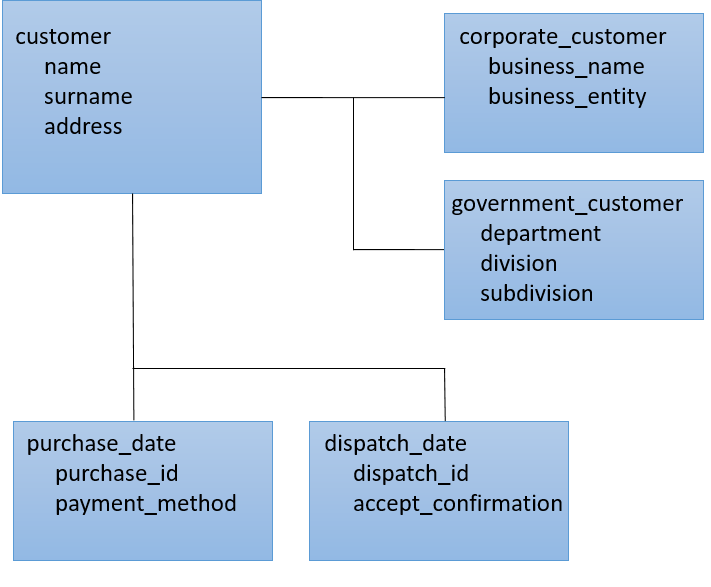
Example of a High-Level ERD (Vaas, 2022)



Alternatively, this or the next figure (depending on the style)

The detailed information of the entities included in the ERD will then be contained in the next (second) level of detail, i.e., in the DIS (Inmon & Linstedt, 2015). It illustrates how the data is organized. Hence, the DIS contains a table’s keys and attributes. It is constructed using so-called boxes. Each box contains the data elements that belong together and are closely related, as follows: The anchor (or primary) box is positioned at the top left of the diagram—it contains the primary and unique data of the entity. More boxes are connected to the primary box with lines. Lines to the right indicate different types of data. The lines that point downwards indicate multiple occurrences of data (Inmon & Linstedt, 2015). An example of a DIS is illustrated next. It shows that a customer can be either a corporate or government customer (thus indicating the customer type). It also portrays that a customer can have made multiple purchases that have been dispatched.

Example of a DIS (Venter, 2022)



When designing a data model, data normalization is an important consideration. When data redundancies have been eliminated in a data model, the data is normalized. The technique to remove data redundancies is known as third normal form (3NF) modeling. Normalization is an important aspect to consider when designing data models for databases. It ensures entity and referential integrity. Modeling according to various other normal forms (in addition to 3NF) are detailed in Foster and Godbole (2016) and Mullins et al. (2021).

Furthermore, normalization is beneficial for transaction processing—it ensures that the process to load and update transactions is simple and fast. However, 3NF models capture all the micro-relationships between data elements, so it may result in too many tables that are linked together in a perplexing spider web of joins. Accordingly, an organization’s hundreds (or even thousands) of logical entities, such as in the case when designing and modeling a DWH, will result in a very complex data model (Kimball et al., 2008). In a DWH ERD model, data must thus be structured around major subject areas, so that the model indicates the (various) relationships between subject areas. The DWH can then serve various different perspectives, using only its atomic, granular data (Inmon & Linstedt, 2015).

### Dimensional Modelling

Dimensional modeling is useful when designing relational DWHs. It simplifies the visual representation of data models. Dimensional modeling makes data models easier and more intuitive for business users, so it makes the data models easier to understand. Dimensional modeling also enhances the query performance in a DWH (Kimball et al., 2008). A dimensional model contains less tables (than a typical ERD model), in addition to less interactions between the tables, so queries execute faster. Moreover, the denormalization of dimension tables makes it easier to find information quickly, thus enhancing query performance further.

A dimensional model is also a type of ERD. It also models various data entities and their relationships. However, a dimensional model is not normalized. The many-to-many relationships in normalized models that contain numeric and additive non-key metrics are called ‘fact tables’. The entities that contain descriptive information are called ‘dimension tables’. The dimension tables are denormalized and modelled using a flat structure. Dimension tables usually resemble second normal form (2NF) tables. They generally contain many low cardinality descriptors. They also contain single-part keys that connect them directly to fact tables. A dimensional model can be connected to more than one fact table if the entity that it represents is shared among more than one business process (Kimball et al., 2008).

Another method that can be used to model dimensional data marts is ADAPTTM (Application Design for Analytical Processing Technologies). ADAPT is useful to model OLAP systems. Bulos and Forsman (2006) explain that it is used to design OLAP databases using two basic building blocks, i.e., hypercubes and dimensions. A hypercube is fundamentally a cube, meaning that it is a basic data storage unit consisting of an *n*-dimensional array. In the ADAPT model, a dimension entails the axis or index of hypercubes. Data contained in hypercubes can be aggregated (rolled up) using predefined hierarchies consisting of levels. The levels define the hierarchical precedence to be applied when aggregating the data. Furthermore, it contains the following dimension objects that are used to model dimensions: a member, attribute and scope.

A member represents an individual dimension value, for example, a specific month, such as January or February, will be a member of a time dimension. Another example of a member is Berlin or Amsterdam—they will be members of a geography dimension. Furthermore, an attribute contains information about a dimension member, for example, the manager of a department will be a textual attribute of the department entity, while the number of workdays within a specific time period will be a numeric attribute. In addition, a scope entails a collection of dimension members, for example, new inventory items or the current year-to-date months. A scope can be enumerated or derived. An enumerated scope contains a list of members, as provided by an external source or managed manually. A derived scope will be computed from other database objects, such as, the current year-to-date months that are derived from the current month (Bulos & Forsman, 2006).

Multidimensional entity relationship models (MERMs) are useful to model OLAP. It enforces simplicity and makes the model easy to understand. It also ensures that databases can be queried fast (Oracle, 2003). A MERM contains logical cubes, measures and dimensions. Measures with the same shape are organized in logical cubes, meaning that they have identical dimensions. The dimensions entail sets of unique values to identify and categorize data. The time dimension should be typically one of the dimensions to include in a cube to meaningfully qualify the measures that are being analyzed through the cube (Oracle, 2003)

Within the cube, measures are used to populate the cells with facts (metric/measurement data) about business events, operations and transactions. Different levels of aggregation can be organized in hierarchies. Accordingly, hierarchies entail different levels of detail of the metric data of business events, so that analysts can drill down to lower levels and thus into more detail. Alternatively, it is possible to roll up to higher levels of details to see less information. Furthermore, additional information about the data can be provided by means of logical attributes. Attributes are useful to select data and answer queries. For example, by including attributes such as colors, shapes, sizes or flavors queries such as: Which color of summer dresses sold the best in the summer of 2022; and how does it compare with previous summers (Oracle, 2003).

Aggregate data is calculated using the hierarchical relationships that are defined in dimension tables. The aggregates are stored in separate tables. These tables are referred to as summary tables or materialized views. Materialized views enable fast querying of data because it is not necessary to search through all the detailed data to obtain answers (Oracle, 2003).

### Data Vault Modelling

DVs are modelled using a logical modelling technique that is called Visual Data Vault (Linstedt, 2015b). It entails three entities, i.e., hub entities, link entities, and satellite entities to model a DV. Hub entities separate business keys to identify business objects such as a customer number, invoice number, etc. Hub entities also store metadata. Link entities are used to model relationships between hubs. So, they link business objects and/or data received from business processes. Satellite entities store the attributes that belong to hubs and links (Linstedt, 2015b).

Business keys have meaning to the business and are accordingly used to uniquely identify, track and locate information. They are also used in databases but must, for the most part, be altered before including them in DWHs. For example, DV modelling distinguishes between two types of keys that are being used: composite keys and smart keys. Composite keys are made up of different columns while a smart key consists of a single column. The individual parts of a business key are combined to create a smart key. On the other hand, a unique combination of columns makes up a composite key (Linstedt, 2015b). However, we choose to use integer-based surrogate keys in DWHs, instead of business keys. It makes it easier to search through various and diverse tables and answer queries faster.

### Diagramming Tools

Diagramming and charting tools are useful to create diagrams and visualizations. For example, Mermaid is a JavaScript based tool to create and modify diagrams. It uses text definitions inspired by Markdown, i.e., a lightweight markup language that is used to create formatted text with a plain-text editor, and a renderer. Mermaid was created and is described by Sveidqvit (2022).

### Self-Check Questions

1. 1. Please mark the correct statement.
2. A dimensional model is a type of ERD, where the many-to-many relationships in models that contain numeric and additive non-key metrics are labelled as...

* dimension tables
* *fact tables*
* relational tables
* normalized tables

## 1.5 NoSQL-Based Data Warehousing

Existing RDBMS vendors initially attempted to include new data types, as required by big data, and store complex structures, unstructured text, images, video and name-value pairs. However, merely hosting these new data types in RDBMS structures is insufficient. They must be processed and analyzed as well. Consequently, alternative architectures such as Hadoop and Hive emerged. These new architectures are usually open source and less expensive (than extended RDBMS) (Kimball & Ross, 2013). They also enable the effective storage and processing of semi-structured and unstructured data. Moreover, a NoSQL (also referred to as Not Only SQL) approach entails “a family of non-relational database approaches that are designed for managing large data sets” (Foster & Godbole, 2016, p. 9). NoSQL offers an alternative storage model to a ‘traditional’ relational model. It seeks to abolish some of the restrictions posed by relational databases.

### Advantages and Disadvantages of NoSQL

A NoSQL approach realizes improved scalability and distributed processing (Gordon, 2014). They are more flexible and provide better availability at lower costs (Foster & Godbole, 2016). However, some of the regular reliabilities associated with RDBMS is lacking when working with NoSQL databases, e.g., since a schema is not applied at the data entry point, NoSQL databases are more difficult to query. It is also more difficult to maintain data consistency and establish relationships between data sets. NoSQL databases do not typically adhere to the ACID principles that aim to ensure data integrity (Mullins, 2017). Furthermore, relationships between individual records are not established, so aggregated data in a record cannot relate to that of another record. It may also be necessary to update an entire record when adding an attribute.

NoSQL implementations support distributed databases and provide redundant data storage on various servers. The databases can, accordingly, easily scale horizontally. It is possible to increase the performance and storage of databases by adding additional nodes (servers) to the server cluster supporting the databases. It enables distributed processing on the servers of a cluster. In addition, multiple nodes in a cluster can be spanned. Furthermore, multiple replicas of a record are kept across several servers or racks. Hence, non-availability of single servers is avoided, and failure of hardware does not affect the availability of data (Mullins et al., 2021).

With the NoSQL approach large volumes of unstructured data are stored directly to achieve faster throughput and read/write speeds. Consequently, NoSQL databases are suited for large volumes of data that must be searched quickly. However, since it is challenging to establish links with other data sets, it is less suited for data that must be associated with other data at a later point.

### Examples of NoSQL Databases

Data in NoSQL databases are typically stored as key-value pairs, graphs, wide-column tables or JSONdocuments. A few examples of NoSQL databases will be discussed next.

#### Key-value stores

In key-value stores data are stored in a schema-less manner. It contains a ‘key-value’ relationship that includes a unique key and a value. The key is typically a string. The value contains the data of interest. The actual data contained in the store can be a datatype or an object (Gordon, 2014). Oracle NoSQL, Aerospike, Riak, and Redis are examples of key-value databases.

#### Document stores

A document store is also a type of a key-value store. The data values consist of native documents. It can include MS Office, PDF, JSON or XML documents. The documents can also contain different types of data items. MongoDB or CouchDB are examples of document-oriented NoSQL databases.

#### Wide-column (or extensible record) stores

These databases consist of ‘tables’, which are referred to as super column families. The tables contain columns, which are referred to as super columns. Each of these columns can contain, like key-value stores, a mixture of various attributes. Column-based solutions are useful for event-driven data. Hadoop is an example of a wide-column store. Cassandra, HBase, Amazon DynamoDB and Clickhouse are also popular column-based solutions.

#### Graph databases

Graph databases consist of elements (nodes) that are connected to each other with an undetermined number of relationships (referred to as edges and properties). Data that represent, e.g., social relationships, road maps or network topologies, are stored in a graph database. Popular graph-based databases include Neo4j, JanusGraph and RedisGraphs.

### Self-Check Questions

1. Please mark the correct statement.

* The data in NoSQL databases conform to a relational data model
* *NoSQL provides improved scalability when working with unstructured data*
* NoSQL databases are designed to manage large sets of structured data
* The Corporate Information Factory (CIF) is an example of a NoSQL database

Summary

The emergence of new data types and, consequently, big data, is a result of increased computational power, communication velocity and storage capacity. Accordingly, we must distinguish between, and cater for, various types of data.

Firstly, ‘traditional’ relational data is known as structured data. It entails data that are (still) generated by day-to-day business operations and transactions. Structured data is effectively managed by means of RDBMS and associated commercial DWHs.

Secondly, semi-structured data, such as e-mail messages and computer-generated reports, can also provide rich business insights. Semi-structured data is analyzable by computers because they contain some metadata. However, the inherent metadata is not enough, and the process can thus not be easily automated. Semi-structured data sets need additional structuring to become easily manageable.

Thirdly, the volume of unstructured data is growing at a tremendous pace. Traditional approaches, such as the CIF, Kimball Lifecycle Methodology and Data Vault Model, cannot effectively manage unstructured data. Hence, alternative approaches such as NoSQL data management and NoSQL data warehousing are rapidly emerging. The NoSQL model provides improved scalability and distributed processing and is more suitable for unstructured data.

# Unit 2 – Classification

**Study Goals**

On completion of this unit, you will be able to …

… differentiate between layer-based and component-based data warehouse (DWH) architectures.

… explain the uses of the different layer-based architectures.

… distinguish between various component-based architectures.

.

# Classification

#### Introduction

A data warehouse (DWH) serves the purpose of supporting organizational decision making. It therefore entails a centralized data repository where an organization’s data are maintained for analysis purposes. Data in a DWH is tailored for business users. It is made available to the users according to their needs. To ensure that only relevant and quality data is presented to users, a DWH is designed and built according to a specific architecture. The architecture is the blueprint that dictates how data strategies are implemented. The DWH architecture can entail a layer-based or a component-based approach. It is vital that it aligns with the requirements of the business. Business requirements will dictate the DWH classification that works best for the organization (Yang et al., 2019).

Traditional layer-based architectures generally include either a single-layer architecture, a two-layer architecture or a three-layer architecture (Senapati & Kumar, 2014). However, DWH reference architectures can, in specific instances, include up to five levels (Winseman et al., 2012). They vary in terms of the level of separation that occurs between operational and analytic components of the DWH. They also differ in terms of the relative complexity of the DWH design.

Component-based architectures are based on the components of the DWH and their relationships. The (traditional) component-based architectures that we explore here include the central DWH with dependent data marts, i.e., Inmon’s Corporate Information Factory (CIF) architecture as well as the bus architecture with several dimensional data marts linked to it, i.e., the Kimball Lifecycle approach (Ariyachandra & Watson, 2010). Furthermore, we also position the Data Vault (Linstedt, 2002).

In addition, we look at the newly emerged approaches such as the big data DWH architecture and data lakes. Big data architectures can entail both layers and components. Both aspects are explained in the context of the architecture. However, in this study unit, the big data DWH architecture is categorized as component-based, according to the definition thereof by Yang et al. (2019).

#### Layer-Based Classification

The layer-based architectures derive their names from the fact that they indicate the number of layers that are used in the DWH architecture. Accordingly, we differentiate between DWH architectures that entails one single layer that holds all the data, two separate data layers or three discrete data layers.

A single-layer architecture shares the identical data with an operational source system. The idea behind this is to store all data sets only once and in the same location, hence the reference in the name to a single layer only. Accordingly, the organization’s databases are used for both the business operations as well as data analytics, at the same time (Yang et al., 2019). It therefore can be regarded as a

**Real-time** **system**

The system is updated continuously and at the time when the transactions occur.

**real-time** **system**, where the analytic data is updated in real-time with the operational data. This may be problematic as data analysis may impact negatively upon the performance of the operational transactions, or vice versa.

### One-Layered Architecture (Vaas, 2022)



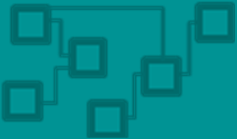
Real-time Data

Operational System

Dispositive /Analytic System

The two-layer architecture design is more complex than a single-layer design. A two- layer architecture can be implemented in cases where it is needed to separate source (operational) data from analytical (historical) data. With a two-layer architecture, the databases are not used concurrently for business operations and data analytics. So, an organization’s operational and analytical processing and databases are done apart from each other (Yang et al., 2019).

### Two-Layered Architecture (Vaas, 2022)



Operational Real-time Data

Operational System



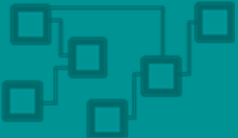
Derived Data

Dispositive /Analytic System

A three-layer architecture consists of three distinct data layers. The first layer stores operational (e.g., real-time) data that are received from production and transactional systems. The second layer entails a reconciled data layer. Accordingly, it contains the data that is being migrated from operational to the analytical databases in order to enable the cleaning and transformation of the data, basically resembling the staging area in a reference DWH. Lastly, the third layer consists of the derived data layer. The derived layer stores data that have been thoroughly cleansed and integrated. Derived data are ready to be used by business users and for analysis purposes (Yang et al., 2019); (Yang et al., 2016).



### Three-Layered Architecture (Vaas, 2022)



Operational Real-time Data

Operational System

Reconciled Data



Derived Data

Dispositive /Analytic System

A five-layer DWH reference architecture is described by Winseman et al. (2012).

The five-layer model extends to accommodate data analysis in the base database also, i.e., in addition to analyses that occur within the data mart level. Furthermore, data transformation can take place in other areas (than the staging area) of the DWH. Accordingly, a scalable five-layer architecture has been proposed by SAP (systems, applications, and products). In this model four of the layers also include a designated area for suitable data transformations. Only the lowest (data acquisition) layer stores the data (temporarily) in the exact same format that they have been extracted from the source systems. The second layer is called the quality and harmonization layer. It is used to integrate the data. The third layer stores the integrated, granular data. It is called the data propagation layer. The fourth layer, i.e., the business transformation layer, is used to transform data according to the needs of business users. Lastly, the

fifth layer provides the data in a format that is ready to be used for reporting and analysis. However, in this model, data can be read from all the preceding layers and not only from the layer directly below it. Moreover, it includes an operational data store that contains near real-time data (Winseman et al., 2012).

##### Self-Check Questions

What is the name of the least complex layer-based data warehouse architecture?

*Single-layer.*

#### Component-Based Classification

The architectures that are classified as component-based include the following: The central DWH architecture with dependent data marts—this architecture is also known as the Corporate Information Factory (CIF), as detailed by Inmon (2006). The Data Vault (DV) architecture, as explained by Linstedt (2015), is also a component- based architecture. Furthermore, the data mart bus architecture with several dimensional data marts that are linked to it (also referred to as a bus architecture) is a component-based architecture—it is also known as the Kimball DWH approach, as detailed by Kimball and Ross (2013); Kimball et al. (2008).

##### The Central Architecture

A central architecture is based on the data requirements of an entire organization (i.e., an enterprise). It is, in view of that, also known as an enterprise DWH. All the data of the organization is maintained in the central DWH. Hence, the central DWH is built first. Dependent data marts for the business-specific and subject areas are then developed in an iterative manner (Inmon, 2006). Even though this architecture

**Denormalized** **data** The process of denormalization means that precomputed and redundant data are added to a data set that is otherwise normalized, so that the denormalized data is accordingly used to improve a database’s read performance.

**Business** **process**

s developed iteratively, a holistic and enterprise view of the data must be maintained throughout the lifecycle of the DWH. The central DWH typically holds atomic (3NF) level data. On the other hand, the dependent data marts can contain normalized data, **denormalized** **data**, aggregated data or summarized data (Ariyachandra & Watson, 2010). In accordance with the design of a central DWH that is surrounded by the various dependent data marts, this architecture s also referred to as a hub-and-spoke design.

##### The Data Mart Bus Architecture

i

i

In a data mart bus architecture, a single **business** **process** and its associated

A series of activity steps that are performed by a group of stakeholders to achieve a

concrete and typically measurable business goal.

business requirements are identified first. The first mart is then designed and

developed for this business process only (Ariyachandra & Watson, 2010). Hence, this architecture is based on the organization’s business processes (i.e., business areas), rather than on the organization’s data. The data marts contain denormalized data in the dimension tables. Tables with transaction data (i.e., fact tables) are true to a declared grain. The grain can be granular (with normalized data) or summative (with summarized or aggregated data).

To begin with, only dimensions (descriptive and contextual information) and measures (transaction data) associated with this business process are used. Accordingly, where the descriptive information of business processes is common and therefore shared, conformed dimensions must be developed. So, the data are maintained in integrated (and conformed) data marts. Conformed dimensions entail uniform and agreed-upon descriptions and fields (Kimball & Ross, 2013). In this

model, the enterprise-wide view is achieved and maintained through the development of the enterprise bus matrix.

##### The Data Vault

The Data Vault (DV) entails a hybrid approach that includes features of the CIF as well as the data mart bus architecture. For example, the DV applies 3NF modeling of data, in addition to dimensional modelling principles (Linstedt, 2002). Accordingly, in a DV, the structural information that are used for the design of the DWH are kept separate from the contextual and descriptive business-related information.

##### The Big Data Architecture and Data Lake

Big data cannot be effectively managed and analyzed with traditional, relational DWH approaches and systems. Therefore, big data platforms and architectures emerged. Big data architecture refers to the logical and physical structures that facilitate proper ingestion, processing, storage, management and access of high volumes of diverse data. Hence, it enables efficient processing of large volumes of semi-structured and unstructured data, so that the data can be effectively stored and analyzed to be used for business purposes. Big data analytics tools are subsequently used to extract vital business information from data that would otherwise be viewed as quite ambiguous. Accordingly, it can effectively analyze and report on petabyte levels of data by means of familiar SQL-like languages (Yang et al., 2019).

As in the case of traditional, relational DWH architectures, a big data architecture will typically involve several layers and components. For example, it broadly entails the following four levels of layers: first, a layer of data sources; second, layers to ingest, manage and store data; third, analysis and processing layers; fourth, user-facing presentation (or consumption) layers to present the business users with relevant decision support information and business intelligence (BI). Anand (2021) argues that big data sources typically include a mixture of open-source data as well as data from commercial (third-party vendor) data providers. Moreover, the data can be imported

in (static) batches or via (real-time) streams. Data are thus managed using batch processing and/or real-time processing. Furthermore, the data are stored in either distributed storage or a RDBMS (or a mixture of both).

Big data is typically stored in data lakes to keep data from various and diverse sources in its native format. It includes structured as well as semi-structured and unstructured data (Dixon, 2011). Furthermore, an analytical data store generally stores prepared data in an interactive or NoSQL DWH so that the data can be analyzed and queried using big data analytics tools (Anand, 2021). Lastly, the output dimension of a big data architecture provides users with a customizable user interface and BI capabilities. It provides, for example, query results to users (Anand, 2021).

##### Self-Check Questions

Please complete the following sentence:

In a data mart bus architecture, a single business *process* and its associated business

*requirements* are identified first.

**Summary**

The main aim of a DWH is to support various business users with their organizational decision making. Data are therefore maintained in a central data repository, so that it can be used for analysis purposes. To ensure that users are presented with relevant, quality data, a suitable architecture must be used to design and implement the DWH. The business requirements dictate the architecture that will be chosen. It can entail either a layer-based or a component-based architecture.

In this unit we differentiated between the following layer-based approaches: a single- layer, two-layer or three-layer architecture. They mainly differ in terms of separation of the data layers and general complexity of implementation.

Furthermore, we looked at component-based architectures. They are based on their components and the relationships between these components. The centralized DWH with dependent data marts as well as the bus architecture with linked dimensional data marts are examples of relational component-based architectures.

We also positioned newly emerging approaches such as big data DWH architectures and data lakes. They are used to store and manage large volumes of data effectively.

# Unit 3 – Data Warehouse Architecture

**Study Goals**

On completion of this unit, you will be able to …

… differentiate between the three types of architectural DW design.

… explain the purpose and area of application of a single-layer architecture.

… describe the design principles of a two-layer architecture.

… discuss the different tiers of a three-layer architecture.

# 3. Data Warehouse Architecture

## Introduction

The DWH architecture defines the overall architecture that facilitates the processing of data communication and the presentation of information within the organization. A DWH can be implemented by way of a layer-based approach (Bauer & Günzel, 2001). Accordingly, depending on the size and complexity of an organization and its data sources as well as the analytics and reporting requirements that they have, they can decide on one of three different layer-based architectures.

A single-layer DWH architecture is the easiest to implement. It minimizes redundancy, but separation between transaction and analytical processing areas is merely virtual. Moreover, it provides limited flexibility and analytical capabilities (Smith, 2022).

A two-layer architecture provides physical separation between the sources and DWH. It is an enhanced design, compared to the single-layer architecture, but expandability is still limited, and it can accordingly only support a limited number of users (Yang et al., 2019).

The three-layer architecture is well-defined and organized—it entails three tiers, thus separating the storage databases, analytical databases and front-end, i.e., the client-facing layer. This architecture is more complex and expensive, and it is therefore usually implemented for large and complex organizations (Smith, 2022).

## 3.1 Single-Layer Architecture

A single-layer DWH architecture simply aims to remove redundancy—its purpose is to minimize the amount of data stored (Smith, 2022). So, a dense set of data is created, to reduce the overall volume of data. However, a major drawback of the single-layer architecture is that the transactional processing is not separated from analytical processing. The OLTP (online transactional processing) systems that gather detailed raw data, originating from executing the day-to-day business processes, is conjoined with OLAP (online analytical processing) applications that support data requirements such as data analytics, trend analysis, **data mining**, and reporting.

**Data mining**

The process whereby anomalies, patterns and correlations within large data sets are identified.

The single-layer DWH is the most basic architecture of all the possible DWH designs—it excludes a **staging area**, i.e., the area that serves the purpose of cleaning and transforming the (operational) data prior to loading it into the DWH (Smith, 2022). A single-layer DWH still provides a platform for analytical processing. However, in the case of the single-layer DWH, we fundamentally have a virtual DWH structure—the multidimensional view of the organization’s operational data is created by means of middleware (Yang et al., 2019). Consequently, query processing speeds can be slow: DWH queries can adversely affect transactional workloads in the operational systems. Also, the query results may be ‘messy’, in that non-required operational data can be included in the results. It can also be difficult to link effectively to presentation tools, so reporting capabilities can be inadequate. Furthermore, the flexibility and analytical capabilities of the single-layer architecture is limited (Smith, 2022).

**Staging area**

A space where data is processed, i.e., cleansed and transformed, prior to being loaded into the DW.

In view of the above, a single-layer architecture is not the ideal DWH model, especially for large organizations and businesses with complex data requirements, as well as in organizations that have several data streams—it is therefore not frequently used in practice by large and complex organizations. A single-layer architecture is, however, suitable for a business where all the raw operational data is in a standardized format, which does not require complex analytics, and where BI is not used systematically.

### Self-Check Questions

1. 1.Please complete the following sentence:

In the single-layer data warehouse the data are not *cleaned* and *transformed* prior to loading it into the data warehouse.

## 3.2 Two-Layer Architecture

A two-layer DWH architecture physically separates the DWH layer from source systems. The separation is physical, as it uses two servers, i.e., a system as well as a database server.

The term ‘two-layer’ highlights the separation between the sources and DWH. However, the architecture entails four data flow stages: The source layer, data staging, the DWH as well as a layer that enables analysis. Accordingly, it includes a staging area between the source systems and DWH applications to cleanse and transform the data prior to loading it into the DWH (Simic, 2020). Moreover, it includes a discrete layer that facilitates analytical processing. In the two-layer architecture, a data mart level can be added between the DWH and the BI applications that provides the user interface. This way, departmental access can be given to domain-specific information and the overall DWH security is enhanced (Smith, 2022).

To summarize, the two-layer DWH is an enhanced design when compared to the single-layer architecture—it stores, organizes and manages data more efficiently. However, a two-layer architecture design is still not very expandable and cannot effectively scale to accommodate, e.g., organizational growth. In addition, it cannot support many users.

### Self-Check Questions

1. Please complete the following sentence:

A two-layer data warehouse architecture indicates a separation between the source systems and the *data warehouse*.

## 3.3 Three-Layer Architecture

The three-layer DWH architecture is the most modern and, accordingly, also the most used DWH design for large businesses—the data flow, from raw operational data to valuable business insights, is well-defined and well-organized. The three-layer architecture typically contains the following: Multiple source systems, a **reconciled layer** and a DWH system that includes data marts and the multi-dimensional DWH layer, i.e., OLAP cubes are employed. Smith (2022) states that a three-layer DWH architecture entails a top, middle and bottom tier.

**Reconciled layer**

The layer between the physical source systems and the DW; it serves as a standardized, enterprise-wide reference data model.

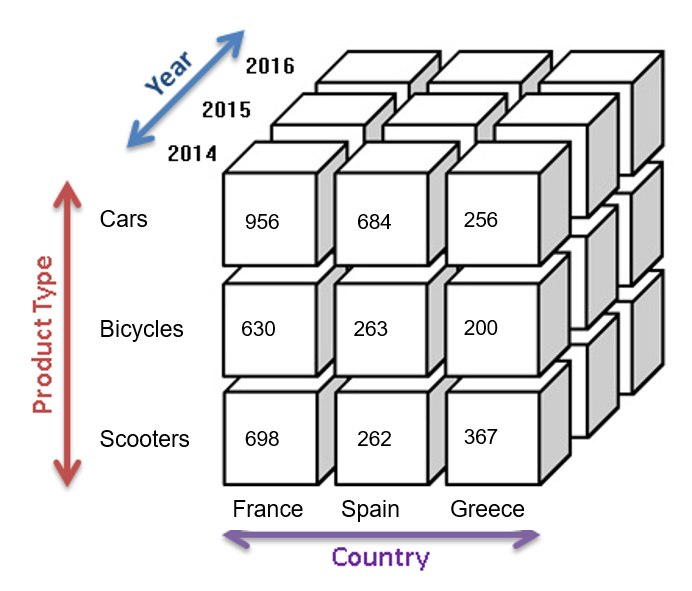
### The Bottom Tier

The bottom tier generally encompasses a relational database system—with data that have been cleansed, transformed, and loaded into a staging area. Data transformation and integration in this tier is executed according to an enterprise-wide reference data model, i.e., the reconciled layer. So, on this level, the architecture entails a database server that consists of an abstraction layer, which is used to consolidate and organize the raw data from the various operational and transactional sources. The bottom tier fundamentally provides (relational) storage and management (Kimball & Ross, 2013).

### The Middle Tier

The middle tier consists of an abstracted view of the analytical database—it is in the form of an OLAP server, which serves to rework the data into a structure that is most suitable for analyses and queries from the business users’ perspectives. Hence, the middle tier can host several data marts, in order to provide departmental and/or business-process specific analysis and reporting information. Accordingly, it contains the OLAP server to present data in an intuitive manner and to facilitate the ‘slicing’ and ‘dicing’ of data, i.e., provide the ability to separate and combine the data in a DW in seemingly endless combinations (Kimball et al., 2008). The OLAP server utilizes cubes to present the data from various perspectives—an example of an OLAP cube with three dimensions, i.e., product type, country, and year, is shown next.

An OLAP Cube (Venter, 2022)



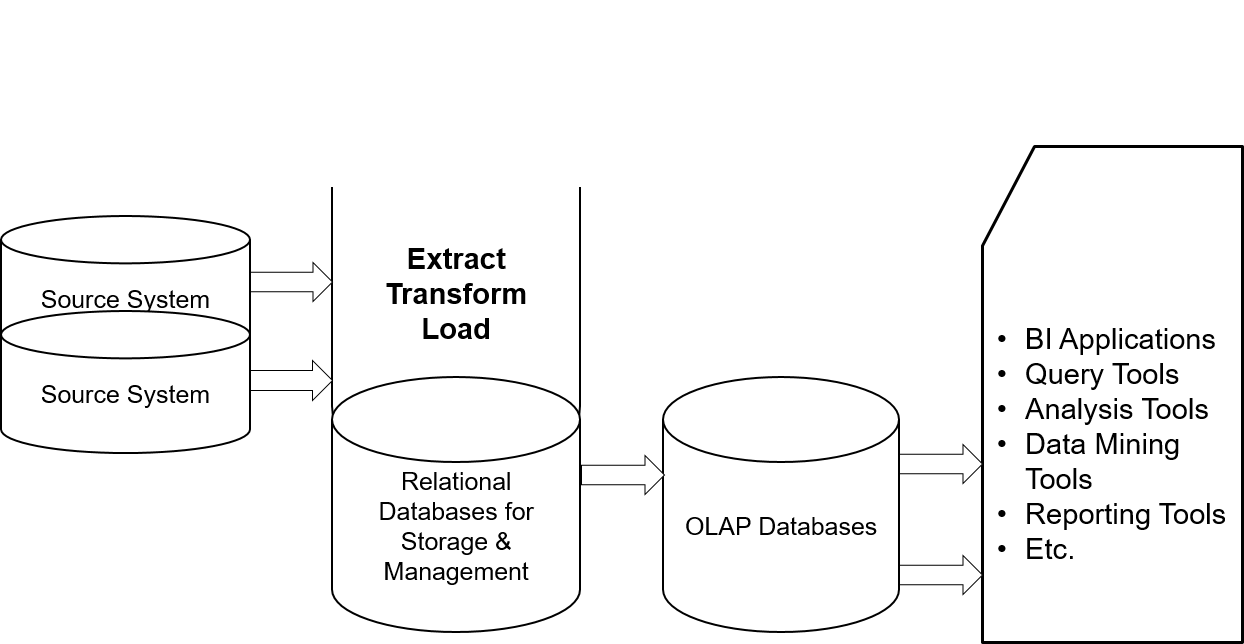
The OLAP server can be implemented using either ROLAP (relational online analytical processing), or MOLAP (multidimensional online analytical processing). Relational databases present data in two dimensions, i.e., columns and rows, whereas OLAP facilitates the compilation of data in multiple dimensions (thus resembling cubes). So, in the case of ROLAP, an extended RDBMS maps functions on multidimensional data to standard relational operations. On the other hand, with the use of MOLAP, multidimensional information and processes are implemented on a specific multidimensional OLAP server (Inmon et al., 2001).

Kimball et al. (2008, p. 610) refer to the bottom and middle tiers collectively as the ‘presentation server’, i.e., an area that provides a “single source for analytic data” in the form of “database platforms where the data is stored for direct querying by business users, reporting systems, and other BI applications”.

### The Top Tier

The top tier involves the client-facing front-end layer that is used to access the data in the DWH. Hence, in the top tier, the client uses various tools and applications, e.g., BI applications and dashboards, query tools, analysis tools, reporting tools, and data mining tools, to interact with the data in the DWH. It is critical that BI applications are correct, perform well, easy to use, and aesthetically pleasing (Kimball et al., 2008). A simplified example of the three-layer architecture, with associated source systems, is illustrated below.

Three-Layer Architecture with Source Systems (Venter, 2022)



### Self-Check Questions

1. Please complete the following sentence:

The bottom tier comprises data that have been *cleansed*, *transformed,* and loaded into a staging area.

Summary

The overall architecture and design of the DWH dictates how data are stored, managed and communicated within an organization. The architecture should be chosen based on the size and complexity of the business and its data sources.

The single-layer DWH architecture is suitable for a business that has its raw data in a standardized format, and where complex analytics is not a requirement. However, it is not suitable for large and complex organizations. The lack of physical separation between the operational and analytical systems and the limited flexibility of a single-layer architecture makes it unsuitable for business with complex and diverse data.

A two-layer architecture can be used physically to separate the operational systems and analytics. However, the two-layer architecture’s expandability is still limited, and it only supports a limited number of users.

The three-layer architecture is the most comprehensive design—it provides data storage and management capabilities that are well-defined and well-organized. The three physically separated tiers provide distinct storage and analytics, in addition to a front-end, client-facing layer. It is usually implemented for organization with large and complicated structures, and complex data sources and requirements.

# Unit 4 Data Warehouse Components

On completion of this unit, you will be able to …

# 4. Data Warehouse Components

## Introduction

A DWH entails various components. At the heart of a DWH is the database system. The operational databases, which are regularly updated via the production OLTP systems, constantly feed the DWH with new data. A DWH contains a full and complete repository of the organization’s historical data, for as long as it is required to keep track thereof. Data are extracted and transformed, and then loaded into a DWH by means of a well-defined and structured process. Only cleansed data are loaded into the DWH. Hence, the extract, transform and load (ETL) process entails extraction of data from various sources, transformation and cleansing of the data, and the loading of cleansed data into the integrated, detailed and aggregated data models of the DWH. So, DWH data can be confidently used for analysis in support of the decision-making processes. The ETL process, as proposed by Inmon et al. (2001), as well as the ETL process according to Kimball et al. (2008) and Kimball and Caserta (2004), are expanded upon.

A DWH is populated with atomic granular data. However, to provide contextual data and analytic capabilities, the data must be summarized and/or aggregated. For this purpose, data marts are used. Data marts provide customized subsets of the data in support of departmental perspectives. Data marts also support analysis and decision-making capabilities (Inmon & Linstedt, 2015). Data marts are therefore discussed in this Unit.

An overarching view of an organization’s DWH can be obtained through a high-level enterprise DWH bus matrix. A bus matrix corresponds to the overall data architecture. It also includes all the organizational business processes and provides an architectural framework that can be used to incrementally design and develop an enterprise wide DWH (Kimball et al., 2008). Hence, we briefly discuss the DWH bus architecture.

## 4.1 Databases

Databases are at the heart of a DWH system and, accordingly, an organization’s mission critical and essential data are stored and managed in “a computerized record keeping system” that aims to maintain and make information available to end-users when required (Foster & Godbole, 2016, p. 3). The organization’s **production databases** are continuously updated via the OLTP systems, such as product and inventory control systems, payroll, accounts payable, etc. In addition, the OLTP systems are the source systems that feed the DWH system and applications. A DWH stores all the historical data and provides access to it. It facilitates OLAP functions such as analyses, ad-hoc queries as well as relevant time-series reporting. Data are periodically extracted from OLTP systems to be transformed (filtered and cleansed). Only cleansed data can be loaded into a DWH. A DWH includes any (or a combination) of the following types of databases:

**Production databases**

The databases that are used in the processing of daily transactions in operative business processes.

* Relational databases—entail row-centered databases that store relational data in (two-dimensional) rows and columns, e.g., IBM DB2, SAP, Oracle, and Microsoft SQL Server.
* Analytics databases—store multidimensional data sets (typically in the form of OLAP cubes) that are used for analytical purposes, e.g., Greenplum and Teradata.
* DWH applications—these are theoretically not storage databases. Instead, the applications entail a combination of hardware and software that facilitates the storage and management of data. Vendors offer DWH applications as off-the-shelf implementable solutions—software is used to manage data and store it on hardware. Examples of such DWH applications include Oracle Exadata, IBM Netezza and SAP Hana.
* Cloud-based databases—entail databases that are hosted on a cloud. It is thus not needed to procure hardware to host the DWH. Google BigQuery, Microsoft Azure SQL and Amazon Redshift are examples of cloud-solutions.

Foster and Godbole (2016) say that a DWH database typically consists of different types of relations—the following are typical relations of DWH databases:

* The base relations that indicate the conceptual schema.
* Virtual relations such as logical views that are derived from the base relations. They are usually simply referred to as views.
* **Snapshots** are the physically defined, named and derived relations. They store physical data.

**Snapshots**

Detailed records that reflect a view of the data in the database, as captured at a particular point in time.

The databases of the central DWH are kept separate from the databases of operational systems. A DWH stores a complete, detailed repository of the organization's historical data. The data is typically at least 24 hours old. However, a DWH may also contain data that is up to a decade old. The length of time that the data is stored in a DWH depends on various legal and regulatory factors (Inmon & Linstedt, 2015). A DWH also includes integrated data, detailed data, summary data, historical data and metadata (Inmon, 1996).

### Self-Check Questions

* + - 1. Please list three types of databases.

*Relational databases.*

*Analytics databases.*

*Data warehouse (or DWH) application databases.*

*Cloud-based databases.*

## 4.2 ETL-Process Components

The ETL process involves extracting data from various source systems, transformation of data into a unified format and loading of the data into a DWH so that it can be used for analysis purposes. The ETL process is executed in a staging area, prior to loading data into the presentation server of the DWH. The presentation server is, according to Kimball et al. (2008), a platform where users can query data directly. The Inmon CIF model versus the Kimball Lifecycle model proposes slightly different approaches in terms of ETL—both are expanded upon next.

### The I & T Layer in Inmon’s CIF Model

A DWH that is built upon the CIF model, applies an integration and transformation (I & T) layer, i.e., an architectural component made up of programs and applications that capture, transform and move data from application (transactional) environments to the ODS and DWH environment. Data are integrated and transformed in the I & T layer to become corporate assets, prior to loading it into the DWH (Inmon et al., 2001). ETL, in the Inmon-model, loads all the organizational data into a giant DWH (Breslin, 2004).

According to Inmon et al. (2001) the I & T interface is quite unstable because the set of programs and applications constantly changes. Also, as a DWH is built incrementally, pieces are added over time. Lastly, the DWH is also built iteratively, so the environment is constantly reshaped and retooled, as required.

The I & T interface uses a logical data model as a blueprint. The logical model defines how the data should be structured in the CIF. A builder is then used to create the I & T programs according to the specifications. Thereafter, the I & T programs transform the raw (source) data that it receives. Metadata is also created in the I & T layer. It provides a reference guide that describes the logic of the transformations that occur (Inmon et al., 2001).

### The Kimball Approach to ETL

When applying the Kimball ETL approach, data are extracted from the original source locations, and transformed and loaded into a final set of tables that users can query, (Kimball & Caserta, 2004; Kimball et al., 2008). ETL, in the Kimball-model loads data into a set of smaller databases. These databases are also called data marts (Breslin, 2004). Metadata describes the flow of the data from sources to targets. The data flow is implemented using a predefined architecture. The ETL architecture is defined and established when all known requirements, realities and constraints that may affect the ETL system have been gathered, and the ETL team is satisfied that they fully understand the effects thereof on the ETL system. The following areas are critical to ensure that accurate requirements are gathered (Kimball et al., 2008, p. 370):

* “Business Needs”—entails the relevant information and content that business users need to make informed business decisions.
* “Compliance”—a proper **due diligence** must be performed to ensure all legal and regulatory reporting requirements have been included.

**Due diligence**

A process to verify, investigate or audit information to confirm, e.g., integrity and authenticity.

* “Data Quality”—as a result of the increased use of distributed data sources as well as the increase in compliance requirements, organizations must ensure that they handle all the data with the utmost care. Organizations should also ensure they share high-quality data only with users.
* “Security”—data must be made available widely to decision makers, while also considering the security interests and constraints that may apply to specific pieces of information.
* “Data Integration”—ensures that data are effectively integrated from various distributed data sources. Hence, common dimension attributes are defined for business processes from different departments that share, e.g., business metrics such as key performance indicators (KPIs).
* “Data Latency”—describes the speed at which the data collected from source systems can be made available to business users, via the DWH.
* “Archiving and Lineage”—entails a process to ensure that data are successfully staged and archived, i.e., so that records are not lost. The accompanying metadata must thus be sufficient to trace the origin of data as well as all the processing steps of the data, until everything has been loaded effectively into the DWH.
* “User Delivery Interfaces”—the content and structure of the DWH data must be sufficient, so that business users can access information quickly and easily through the BI applications.
* “Available skills”—entails the process to ensure that the resources, skills and capabilities required to build and manage the DWH system are available. So, the resources and capabilities must be considered upfront.
* “Legacy Licenses”—involves ensuring a balanced approach in terms of the use of existing legacy licenses versus potentially acquiring new licenses that are better suited for the current as well as future ETL environment. So, informed decisions must be taken in this regard, to ensure that the ETL environment will be suitable and sustainable in the future as well.

The Kimball ETL approach includes the following four major components: extracting; cleaning and conforming; delivering; and managing (Kimball et al., 2008). They are discussed next.

#### Extracting data

During extracting, the main aims are to understand the source data and transfer it correctly into the DWH environment. The DWH environment is independent from the operational systems so that it can be worked on without affecting the data quality and transaction performance in the source systems. Data must be extracted correctly since the data in the DWH will typically not be updated once it has been loaded (Breslin, 2004). During extraction raw data is collected from the source systems and written to the (first) staging environment. This extraction process does not involve significant restructuring of data. However, aspects such as data compression and security must be considered, e.g., if large amounts of data are transmitted over a public network (Kimball et al., 2008).

The extraction process involves several subcomponents such as data profiling, a change data capture (CDC) system and data extraction (Kimball et al., 2008). These are detailed in the next paragraphs.

Data are first profiled. It entails an analysis to describe the content, consistency and structure of the data. The aims of data profiling include the following: determine the data sources to include in the DWH project; and identify the issues that may arise when including source data so that corrective actions can be taken to improve data quality (Kimball, 2004). Data profiling assists the ETL team to determine the scope of the data cleansing effort that must be performed.

Secondly, a CDC system must be developed to ensure that the new data are imported into the DWH, without having to update all the historical data. It necessitates proper isolation of the most recent source data. A CDC system includes any (or a combination) of the following processes (Kimball et al., 2008, p. 377):

* “Audit columns”—make use of appended audit columns in the source systems. They automatically keep track of the precise dates and times when a new database record was added, or an existing record was changed. However, this requires the audit columns to be populated by database triggers.
* “Timed extracts”—entails selecting all the rows where the date indicates that the creation date, or the last date that the record has been modified, is equal to SYSDATE-1, indicating the records that have been created or altered on the previous day. However, this approach is less reliable since a time-based data selection can load duplicate rows when restarting after an intermediate stop.
* “Full “diff compare.””—with this approach the current day's data are compared, record by record, with a snapshot of the previous day's data. This approach is a very thorough. However, it is also very resource intensive.
* “Data base log scraping”—this approach takes a snapshot of the database redo log at a scheduled time. The snapshot is scanned for transactions that have an impact on tables that are of interest to the ETL load.
* “Message queue monitoring”—involves constant monitoring of the queue of a message-based transaction system to identify transactions that affect tables of interest.

Lastly, it is important to note that the data is extracted from various source systems. Sources may include different types of systems such as RDMBSs, mainframe systems, ERP systems, web logs, etc. Depending on the type of source system, the data can be extracted as a file or a stream. When extracting data in the form of files, the following steps should be followed: Extract the data to a file, move the file to the ETL server, transform the file contents and load the transformed data into a staging database. On the other hand, if the extraction is set up as a stream, it can be constructed as a single process. Hence, data flows directly from the source systems through the transformation engine and into the staging database (Kimball & Ross, 2013; Kimball et al., 2008).

#### Cleansing and conforming data

During the cleansing and conforming process, the source data goes through a series of processing steps in the ETL system. The aims of these are to improve the quality of the data, merge source data from two or more sources as well as create and enforce conformed dimensions and metrics. During this process, value is added by cleansing and modifying data to improve data quality. Several methods, such as field mapping and algorithmic comparisons, are used to modify data (Breslin, 2004). Furthermore, metadata is created during the transformation process. Metadata is a crucial aspect of a DWH. It is used throughout the DWH lifecycle. It is also useful to identify problems in source systems, so that they can be improved over time (Kimball et al., 2008).

A cleansing and conforming process includes subcomponents such as data cleansing, an error event schema and audit assembler, and data deduplication and conforming (Kimball & Caserta, 2004; Kimball et al., 2008). They are expanded upon in the next paragraphs.

The cleansing system should provide a comprehensive architecture for data cleansing and the capturing of data quality events, as well as for the measuring and controlling of the quality of the data in the DWH. Furthermore, it is crucial to ensure the following: proactively diagnose and triage data quality issues; ensure that users are given quality data; identify errors that can occur during ETL; capture data quality errors; measure data quality metrics over time; and attach quality confidence metrics to the final data. Hence, visual diagnostic filters should be built into the data flow pipelines. It is also important to rectify errors swiftly. An error event schema is useful to identify and resolve errors. It captures and displays errors that have occurred. It also ensures that all error events are recorded as they happen in the ETL pipeline (Kimball et al., 2008). Additionally, an audit table must be created to capture the metadata context when metric or measurement tables are created in the ETL pipeline—this table is called an audit assembler. The setup and implementation of an error event schema and audit assembler in a practical (and dimensional) format is explained in Kimball et al. (2008).

When data are derived from several sources, the information may have to be merged. However, the different sources may contain contradicting information. Hence, all the data must be evaluated to determine which of the conflicting records must be added to the final data set. The process to combine a set of matching records into a unified representation that combines only the highest quality (or most correct) entries into a conformed row is known as survivorship. It involves establishing clear business rules to define the priority sequence for values from different sources to create a row with the attributes that ‘survived best’ (Kimball et al., 2008). Data standardization as well as data integration tools are available to implement the deduplication process.

Furthermore, entity (or dimension) tables that are shared across divisions, such as a table of customer data used by both the customer service department and the billing department, should be aligned (conformed) in terms of both columns and content (Naeem, 2020). Alignment is achieved by combining and integrating data from multiple systems in a way “that it is structurally identical, deduplicated, filtered of invalid data, and standardized in terms of content rows in a conformed image” (Kimball et al., 2008, p. 386). Domain mappings are therefore made when metadata captures relationships between values from the source systems that are valid, and conformed table values. There are also many ETL tools available to support the data conformation process.

#### Delivering data

During the delivery process data are physically structured, loaded and delivered into the target (dimensional) data tables. When the process is completed the DWH contains a combination of granular data and multidimensional (OLAP) data. At this point, the user can use OLAP tools to analyze multidimensional data from various perspectives. Moreover, new data are continuously loaded into the DWH. ETL routines are used to ensure that newly loaded data are correctly appended. It is essential to ensure data integrity and avoid data redundancy (Breslin, 2004).

#### Managing the ETL environment

This process involves ongoing management and support of the ETL system. It aims to ensure that the ETL systems are reliable and that the ETL processes run consistently to completion (Breslin, 2004). Moreover, it ensures that trustworthy data are made available to users in a timely manner. The DWH should also consistently meet agreed upon service levels. In addition, the DWH must be manageable and the ETL processes should evolve gracefully along with the business (Kimball et al., 2008). Furthermore, a DWH should have a suitable backup and recovery process in place. A suitable version control and version migration system must be implemented. Effective management of the ETL environment requires a robust job scheduler to ensure that all the ETL job streams and workflows are created, managed and monitored in an optimal manner (Kimball et al., 2008). Job schedulers are available with most ETL tool packages.

### ETL Tools

Regardless of the applied approach (Inmon or Kimball), ETL process components are complex and resource intensive. ETL must thus be performed using appropriate tools. The ETL process can be coded manually. However, it is not recommended, especially for large companies with multiple source systems and masses of data. ETL systems must be able to read data from numerous different sources, such as OLE DB (object linking and embedding, database), ODBC (open database connectivity), flat files and various native database drivers (Kimball & Ross, 2013).

The chosen ETL tool dictates the time that will be spent on data extraction, the data extraction approaches, the transformation types that can be applied, the simplicity of the transformation, the definition of business rules to validate and cleanse data to optimize analytics, the methods of ensuring data quality in the DWH and the process of distributing data from the DWH to BI applications (Fatima, 2019).

The use of commercial ETL tools offers the following benefits (Kimball & Ross, 2013):

* Graphical tools facilitate self-documentation.
* Provide a foundation to capture the metadata of all the ETL process steps.
* Simplify version control in multi-developer environments.
* Provide advanced transformation logic.
* Improve system performance at lower levels of expertise.

### Self-Check Questions

1. Please complete the following sentence.

The extract, transformation and load (ETL) process include transformation of data into a *unified* format.

## 4.3 Data Marts

The foundation of a DWH is granular data. It provides a basis for effective analysis and BI. However, the granular data must be summarized and/or aggregated first, so that it can be effectively analyzed. The different users of data usually need to see different perspectives of the same data (Inmon & Linstedt, 2015). As an example, a marketing department needs to see a different perspective (view) of an organization’s data than an accounting department. Consequently, various data views must be implemented to effectively serve the diverse departmental objectives. Hence, data marts are used to serve the diverse and unique needs of various organizational departments (Inmon & Linstedt, 2015). It is “a customized subset of data from the data warehouse tailored to support the specified analytical requirements of a given business unit” (Inmon et al., 2001, p. 8).

Data marts emerge from the DWH and all the data that are contained in individual data marts can be reconciled again with the (central) DWH. However, each data mart shows a unique and business process-oriented perspective of the granular data. It typically reflects departmental business process KPIs (Inmon & Linstedt, 2015). Since different business processes have different KPIs, different perspectives are presented by data marts. Nevertheless, data marts can also be shared among departments when they have similar analytical needs. Within a data mart structure, the granular data is organized to reflect a desired view. The views aggregate or summarize a portion of a DWH. So, the views can be easily updated when any of the KPIs change.

A data mart can contain two different types of data, i.e., first order data and second order data. It receives first-order data directly from a DWH. First order data is detailed, granular data. Second order data, on the other hand, is first order data that have been manipulated. Accordingly, second order data entails summarized or aggregated data. Hence, data in data marts is typically denormalized, pruned and summarized (Inmon et al., 2001). Data marts are flexible, accessible data structures that offer the following advantages (Inmon et al., 2001):

1. Control—data and processing that occurs in a data mart is fully controlled at the departmental level.
2. Cost—it is less expensive to store and process data on a local (departmental) machine, than in the bigger DWH.
3. Customization—data are customized in the data mart according to the needs of the department it serves. Accordingly, keys can be restructured. Data can also be re-sequenced, merged, pruned, summarized, edited and converted.

### Modelling of a Data Mart

According to Inmon and Linstedt (2015) data marts are effectively modelled by means of dimensional models. Each data mart is then associated with a different and unique dimensional model, consisting of a fact table and dimension tables. Fact tables hold transactional or measurement data. On the other hand, dimension tables hold the contextual data that describes relatively static business entities such as a customer or a product. Dimensional modeling is widely used to visualize a logical DWH design. It entails “a logical design technique for structuring data so that it’s intuitive to business users and delivers fast query performance” (Kimball et al., 2008, p. 235).

A dimensional model includes contextual data and measurement data. Kimball et al. (2008, p. 235) explain that the contextual data is intuitively divided into "independent business subject areas" and describes the "who, what, when, where, why, and how" (which is then represented in the dimension tables). In addition, the measurements (metrics) entail data collected by source systems to support the business processes (which is then represented in the fact tables). Accordingly, each business process is presented as a dimensional model consisting of a fact table that is surrounded by a series of dimension tables. It resembles a star-shaped structure and is therefore also known as a star join. The following four-step design process is used for dimensional modelling: firstly, select the business process to model; secondly, declare the grain of the business process; thirdly, choose applicable dimensions; and fourthly, identify the facts, i.e., the metrics (Breslin, 2004).

A simplified dimensional model consisting of order transactions and relevant dimensional data elements (context data) is shown below. It represents a business process that relates to the ordering of transactions (thus containing the order transactions fact table) with measurements order quantity and amount. Furthermore, the model contains descriptive data for the following entities: order data, product, payment terms, sales representative, and customer. The primary keys (indicated by PK) of the dimension tables are included as foreign keys (indicated by FK) in the fact table. The fact table does not contain its own unique identifier in the form of primary key. Instead, the combination of the foreign keys makes up a surrogate key that represents the primary key of the fact table. A fact table can also contain unique identifiers for each business event that it captures. These are called degenerate dimensions (indicated in the image below as DD).

A Dimensional Model (Venter, 2022)

Graphical user interface

Description automatically generated

### Self-Check Questions

1. Please complete the following sentence:

Data in a data mart is *denormalized*, pruned, and summarized.

## 4.4 Bus Architecture

The Kimball methodology proposes the use of a bus matrix to get an overview of the architecture and prioritize business processes to implement first (Naeem, 2020). The high-level enterprise DWH bus matrix consists of a matrix with rows and columns. The rows list the organization’s business processes while the columns resemble and cross-reference the natural groupings of the standardized descriptive business entities (reference data), also referred to as dimensions, with the business processes that use them (Kimball & Ross, 2013; Kimball et al., 2008). Entities and business processes that relate to each other are indicated on the bus matrix, as illustrated below.

Example of a Bus Matrix (Venter, 2022)

A picture containing text, crossword puzzle

Description automatically generated

A bus matrix that contains only conformed dimensions perfectly matches the overall data architecture of the DWH. In addition, it provides an overall perspective and shows how each business process, when implemented, can incrementally build out the overall architecture. A bus matrix effectively establishes an architectural framework to guide the design of the overall architecture while dividing the work into bite-sized implementation sections (Kimball et al., 2008). The bus matrix is also relatively easy to understand from the perspective of non-technical business users. They may find it easier to interpret than a comparatively more complex ERD (Breslin, 2004).

### Self-Check Questions

1. Please complete the following sentence:

In a high-level enterprise DWH bus matrix the columns correspond to business *entities*.

Summary

A DWH consists of various components, starting with the database systems that form the core of a DWH. OLTP production databases constantly feed new data into the DW. A DWH stores all the historical data of an organization. ETL processes are implemented to ensure that the data in the DW is kept up to date and that the integrity of the data is such that it can be reliably used for effective analysis and accurate reporting. The DWH must also provide a reliable basis for management decisions. In this context, we explored two aligned, yet slightly different, ETL approaches, i.e., the Inmon approach and the Kimball approach.

Efficient data analytics functions require contextual and focused views of individual departmental and/or business process specific KPIs. Therefore, the granular data in a DWH is summarized and/or aggregated in a separate data structure. Accordingly, data marts are used to create specialized summaries, reports and analyses. In addition, it is necessary to have an all-encompassing view of the enterprise-wide DWH. Therefore, the DWH bus matrix must be kept up to date to represent the overall data architecture and the supported business processes within the architecture framework.

# Unit 5 – Big Data Frameworks

**Study Goals**

On completion of this unit, you will be able to …

… discuss the application of Hadoop as a framework to manage big data.

… portray the functioning of Hive to enable data warehousing capabilities for big data.

… articulate the usefulness of data lakes in managing big data.

… explain the architecture of a data lake.

# 5. Big Data Frameworks

## Introduction

The amount of data available to organizations is increasing exponentially, resulting in big data. Big data can potentially be valuable to organizations. However, due to the sheer magnitude of data as well as the vast variety of types and sources, it is difficult to process. Traditional data processing frameworks have shortcomings that make them inadequate for big data. As a result, alternative frameworks emerged. Accordingly, in this chapter, we look at some of the pertinent frameworks that simplify working with big data.

ApacheTM Hadoop® is a big data framework that provides a platform for affordable, open-source, reliable, and scalable data storage as well as analysis (White, 2015). Since the platform is scalable, it be easily expanded. In addition, Apache HiveTM is an open-source platform that offers data warehousing capabilities for big data. Hive enables effective reading, writing, and analysis of large and diverse datasets that are kept in distributed storage. With Hive, big data can be queried using a familiar SQL-like syntax. The syntax is extended to query data stored in semi-structured and unstructured formats also (Atlassian, 2020). The architecture and components of Hadoop and Hive are discussed in this chapter.

One of the key challenges of big data is in terms of storage. Big data cannot be stored effectively in a relational database management system (RDBMS)-type storage system. So, the concept of a data lake emerged. Data lakes respond to some challenges encountered by traditional data warehousing architectures, such as, traditional RDBMS and data warehousing architectures are hierarchical and cannot store semi-structured and unstructured data (Sawadogo & Darmont, 2021). A traditional DWH is still relevant and powerful when working with structured data but cannot realize the potential of semi-structured and unstructured data. Data lakes store data inexpensively in a distributed manner. They are durable, inexpensive, and scalable, so they offer organizations new opportunities in terms of big data (Sitarska-Buba & Zygała, 2020). So, we also explore data lakes in this chapter.

## 5.1 Hadoop

ApacheTM Hadoop® entails an open-source framework that consists of an ecosystem of technologies. It was initially conceived in 2005 by Doug Cutting and Mike Cafarella; however, at the time, they were working on a technology that was aimed at effectively distributing **Apache NutchTM** (Bappalige, 2014). Two Google technologies, as published in 2003 and 2004, resonated well with the work of Cutting and Cafarella. Hence, they incorporated the (distributed) Google File System (GFS), as described by Ghemawat et al. (2003), as well as MapReduce, as described by Dean and Ghemawat (2004), into a new framework. They called the new framework Hadoop. Hence, two core Hadoop components, i.e., the Hadoop Distributed File System and MapReduce, are based on these Google technologies.

**Apache NutchTM**

A Web crawler that is extensible and scalable, it enables fine-grained configuration and can accommodate various data acquisition tasks.

At present, Hadoop is managed and maintained by a global community of software developers and contributors. They collaborate with the non-profit Apache Software Foundation (ASF). The following modules are (officially) included in this project (Apache Software Foundation, 2022a):

* Hadoop Common—includes all the commonly shared libraries and utilities in support of the other Hadoop modules.
* Hadoop Distributed File System (HDFSTM)—scales large data sets out across a cluster of hosts and stores data on multiple low-cost commodity machines. HDFS provides high aggregate bandwidth across a cluster. It also delivers high throughput access to data by means of replication.
* Hadoop MapReduce—provides a data processing paradigm. It is an optimal programming model for large scale processing. MapReduce enables parallel processing of large data sets.
* Hadoop YARN (Yet Another Resource Negotiator)—entails a cluster resource management platform. It provides a framework to schedule jobs and cluster resource management.

However, it is currently commonly considered that related projects of the ASF, such as, Apache PigTM, MahoutTM, HBaseTM, TezTM, FlumeTM, SparkTM, ZookeeperTM, HiveTM, etc., are also an integral part of the Hadoop platform (Bappalige, 2014).

### Hadoop HDFS

A key shortcoming when working with large data sets stored on numerous disks, such as hard disk drives, is that the disk operations result in **latency**. Data transfer rates generally correspond to the bandwidth of the disks but seek operations must wait for the disk head to be moved to a particular place on the disk. Therefore, seek (e.g., read/write) operations are relatively slower than data transfer operations. However, by storing data on distributed systems, and thus accessing only portions of each, and seeking in parallel from all the disks/systems, data can be read and processed a lot faster. This is exactly the principle that is applied by Hadoop HDFS—very large data sets are stored on various distributed machines (or servers) that are shared among users, for faster processing of data (White, 2015).

**Latency**

The time that it takes to complete one input/output (or a read/write) operation on a device.

In terms of data processing, it is important to take note of the fact that HDFS applies the following fundamental concept: With HDFS, data are optimally processed using a ‘write-once-read-many-times’ pattern. It is thus most suitable for batch processing of data. Furthermore, HDFS does not provide low-latency access to data. Instead, HDFS aims primarily to provide high throughput rates (White, 2015). These limitations are, however, addressed by other Hadoop-compliant platforms—they are discussed later in this chapter.

#### Storage of data in HDFS

Since it is very expensive to purchase and maintain large capacities of highly reliable hardware, HDFS facilitates storing of large sets of data across clusters of inexpensive commodity hardware. It contains built-in mechanisms to identify and work around hardware failures, which occur often times when using such low-cost machines. But, as data are replicated, and multiple copies of all the data stored automatically, a high level of fault tolerance is provided. Jobs are automatically redirected to other nodes upon node failures, without any noticeable interruptions to users (Turkington, 2013).

White (2015) and Turkington (2013) explain that the HDFS architecture comprises of: name nodes, data nodes, and blocks (of data). The name node consists of hardware that contains an operating system and name node software. It acts as a master server to manage the file system, regulate access to files, as well as execute file system operations. Each name node is associated with a cluster of data nodes.

A data node entails hardware containing an operating system as well as the data node software. Data nodes manage storage of data. They also perform the read and write operations, on request of the client. Furthermore, the data node creates, deletes and replicates blocks, on request of the name node. Data nodes also move blocks of data over the network (Turkington, 2013; White, 2015).

In considering HDFS blocks, we can note that a block size is (by default) 128 MB, but block sizes are frequently increased in the HDFS configuration, as is required by the data being stored and managed (Turkington, 2013; White, 2015).

### Hadoop MapReduce

According to White (2015), MapReduce is fundamentally a batch query processor and processing system; it runs ad-hoc queries against large data sets and gets the results reasonably fast. MapReduce runs on top of HDFS, receiving requests and scheduling jobs optimally, and thus minimizing network traffic (Bappalige, 2014). It builds upon the functional programming functions ‘map’ and ‘reduce’, as they are applied to lists of input data. Furthermore, it applies the concept of ‘divide and conquer’ to subdivide a problem into multiple individual subtasks, so that the tasks can be executed faster in parallel (Turkington, 2013). With the MapReduce functionality, both unstructured and semi-structured data are delivered to the ‘map’ function as a series of key value pairs. The output of this ‘map’ function will then be a set of other key value pairs, and a ‘reduce’ function performs aggregation to gather a final set of results (Turkington, 2013). This is illustrated next.

MapReduce Functionality (Venter, 2022)

A screenshot of a computer

Description automatically generated with medium confidence

### Hadoop YARN

YARN significantly improves the power of Hadoop clusters. It separates the management of resources and the scheduling of jobs into two discrete processes, which are in this context also referred to as **daemons** (Apache Software Foundation, 2022b). YARN provides resource management capabilities for scalability and compatibility with MapReduce. However, it also provides support for workloads that are not MapReduce. Hence, any distributed program (not only MapReduce) can operate on the data in a Hadoop cluster (White, 2015). Examples include Apache Spark, Apache Flink and Amazon Kinesis.

**Daemons**

These processes run in the background when using a platform such as Hadoop.

The YARN resource manager arbitrates resources among the system’s applications. It applies a scheduler and an applications manager to, respectively, allocate tasks and accept jobs for execution. However, users can specify resources to be reserved, to ensure that essential jobs are executed first. Moreover, with the YARN federation feature, multiple YARN (sub) clusters can be transparently wired together. This way, they appear to be a single large cluster, and can as such achieve larger scale (Apache Software Foundation, 2022b). Ultimately, YARN facilitates improved utilization of a cluster and provides enhanced agility (Bappalige, 2014). The figure below illustrates the YARN architecture.

YARN Architecture (Venter, 2022)

Graphical user interface

Description automatically generated

### Self-Check Questions

1. Please name the two core components of Hadoop.

*MapReduce.*

*HDFS.*

1. Please complete the following sentence:

The YARN resource manager applies a *scheduler* and an *applications* manager to allocate tasks and accept jobs for execution.

## 5.2 Hive

Apache HiveTM is a data warehousing software that enables the reading, writing and management of large and diverse datasets that reside in distributed storage. The (big) data are queried using a typical SQL syntax. The objectives of Hive are to maximize scalability, performance, extensibility, fault-tolerance, and **loose-coupling** with input formats.

**Loose-coupling**

This approach to interconnecting components ensures that they are weakly associated, so that they have minimal effect on each other.

Hive provides features such as easy access to large sets of data via SQL, thus enabling ETL, reporting and analysis. Moreover, it provides a mechanism to impose a structure on various (unstructured) data formats. Hive also provides access to files stored in distributed storage systems, e.g., HDFS or **HBase**. It also enables execution of queries using tools such as Tez, Spark or MapReduce—more information on these tools can be found in Kannan (2015) as well as in Singh and Kaur (2016). The Hive architecture is illustrated next.

**HBase**

Apache **HBase**™ is the Hadoop database, a non-relational, distributed, scalable, big data store.

Hive Architecture (Venter, 2022)

Diagram

Description automatically generated

Furthermore, it enables the implementation of a procedural language tool, i.e., Hive Hybrid Procedural (HPL) SQL. In addition, Hive ensures the speedy retrieval of queries using Hive LLAP (Live Long And Process), YARN as well as Slider, which is an application that allows users to create and run different versions of heterogeneous long-running applications (Atlassian, 2020). The general Hive query components are illustrated below.

Hive Query Components (Venter, 2022)

Graphical user interface, text, application, chat or text message

Description automatically generated

White (2015) describes Hive as an interactive platform with typical RDBMS-type features such as indexes and transactions. However, it is not required for data to be stored in a standardized format. Instead, it uses built-in connectors for unstructured and semi-structured files, e.g., CSV (comma-separated values) text files, TSV (tab-separated values) text files, in addition to other formats such as Apache ORCTM, Apache ParquetTM, etc.

With Hive, users connect to the data via a command line tool and JDBC (Java Database Connectivity) driver. Data are queried using a SQL-type syntax (i.e., HiveQL) that uses standard SQL functionalities. However, HiveQL is extendable through user code and can thus include user defined functions (UFDs), user defined table functions (UDTFs), and user defined aggregate functions (UDAFs) (Atlassian, 2020).

Hive’s two main components are HCatalog and WebHCat. They are discussed next.

### HCatalog

HCatalog provides a table and storage management layer. It enables the abstraction of tables, so that users are presented with a relational view of data stored in HDFS, regardless of the format of the data (Atlassian, 2018a). Accordingly, Hive data models represent tables in a way that is similar to relational database tables, except, there are no relations between the tables in the model (Bekker, 2014). Furthermore, it supports various data processing tools, including Apache Pig and MapReduce, to facilitate reading and writing of all the data on the grid (Atlassian, 2018a).

### WebHCat

WebHCat (formerly known as Templeton) offers HCatalog a **REST** (REpresentational State Transfer) application programming interface (API). WebHCat enables running of MapReduce (or YARN), Pig, or Hive jobs. It also performs Hive metadata operations through a (REST) HTTP (HyperText Transfer Protocol) interface. Hence, HTTP requests can be made from within any application, to access MapReduce/YARN, Pig, Hive, and HCatalog data definition language (DDL), while the data and code are maintained in HDFS (Atlassian, 2018b).

**REST**

This architectural style provides standards between computer systems and the web to ease communication.

### Self-Check Questions

1. Please complete the following sentence:

Hive provides easy access to large sets of data and enables ETL, *reporting* and *analysis*.

## 5.3 Data Lake

Big data exceeds the capabilities of traditional DWH systems. Accordingly, a hierarchical RDMBS-type DWH cannot effectively collect, store and process big data within reasonable time frames. As a result, organizations may fail to fully utilize the opportunities that are brought about by big data. To overcome some of these challenges, data lakes emerged. James Dixon (2011) introduced the concept of a data lake. At the time, he contrasted it with a data mart that contains cleansed, packaged and structured data, and where a subset of attributes are examined and aggregated, so as to answer pre-determined questions. Dixon (2011) argues that a data lake holds data from various sources in its ‘natural state’, and that data lakes serve various and diverse users. Similarly, Sawadogo and Darmont (2021, p. 2) define a data lake as “a very large data storage, management and analysis system that handles any data format” (p. 2).

It is important to be mindful of the fact that metadata is a significant component of data lakes. Data must be stored with metadata tags and unique identifiers. Metadata is used to locate and retrieve data from various and diverse locations (Taylor, 2022).

Data lakes offer several advantages: Open and object storage is inexpensive. Additionally, various users and applications can make use of the data. Data lakes are implemented using a flat architecture. Different data types can therefore be stored as-is. Furthermore, an organization’s data can be conveniently consolidated in the original format, and without having to impose a rigid schema (Taylor, 2022).

### Data Lake Architecture

According to Taylor (2022), a data lake architecture involves the following six layers: First, an ingestion layer links to data sources and enables loading of data into the data lake. Data can be loaded (‘ingested’) in batches, real-time and/or by means of one-time loads. Second, an insights layer offers insights to facilitate the application of data analysis techniques. Third, a distributed data storage layer (such as an HDFS layer) must be implemented to provide scalable and cost-effective storage for data. Fourth, a distillation layer is required to provide some structure for the unstructured data, to facilitate easier analysis. Fifth, a processing layer is implemented to run algorithms and queries. Lastly, a unified operations layer is implemented to govern and monitor systems—it involves managing aspects such as the availability, usability, security and integrity of the organization’s data.

Furthermore, Sawadogo and Darmont (2021) differentiate between data lake ‘pond’ architectures and data lake ‘zone’ architectures. In a pond architecture, the data are divided into different data ponds. Each pond is associated with a specialized storage system, specific data processing and conditioning, as well as an appropriate analysis service. Inmon (2016) mentions the following types of data ponds:

* The raw data pond—contains raw data that has been newly ingested. The data is conditioned and then transferred from a raw data pond into another pond. A raw data pond is not associated with a metadata system.
* The analog data pond—usually processes semi-structured data. Data enters this pond with a high velocity, i.e., data from social media and the IoT (Internet of Things).
* The application data pond—contains data that corresponds with the data in a DWH and, as such, comprises of data that were previously ingested from various software (e.g., production and operational applications). The data that are managed in the application data pond is in general structured, as it comes from RDBMSs.
* The textual data pond—manages unstructured, textual data from big data sources such as call center transcripts. So, it contains a textual disambiguation process to simplify textual data analysis.
* The archival data pond—saves data that are not actively being used but may still be needed in the future. Data may enter the archival pond from the analog, application or textual data ponds.

In a zone architecture, data are assigned to a zone, based on the degree of refinement of the data. As an example, a six-zone architecture has been adopted by Zaloni’s data lake—it includes the following zones (LaPlante, 2016):

* The transient loading zone—it contains the raw data that is being ingested and performs basic data quality assessments.
* The raw data zone—deals with nearly raw data entering from the transient zone.
* The trusted zone—standardized and cleansed data are transferred here. The data stored in this zone are ready to be moved to the next zone.
* The discovery sandbox—data are moved from the trusted zone into this zone. The data is accessible to data scientists. They may apply data wrangling or data discovery operations to access the data.
* The consumption zone—business users can access the data in this zone for decision support. Data are accessible through dashboard tools and can be used to run ‘what-if’ scenarios.
* The governance zone—facilitates management, monitoring and governance of metadata, and ensures the quality of data. This zone also maintains a data catalog and ensures the security of data.

There are a number of variants of the zone architecture. However, they mostly differ in terms of the number of zones that are incorporated, and the specific characteristics of the individual zones. For example, a transient zone is not included in all the zone architectures.

The lambda architecture is a popular zone architecture. It includes two data processing zones: Bulk data are processed in a batch processing zone; while high velocity data (such as data from the IoT) are processed in a real-time processing zone (Sawadogo & Darmont, 2021). However, the lambda architecture results in duplicate modules and coding overhead and it is also difficult to migrate or reorganize a data set. So, the kappa architecture emerged. With kappa, it is not necessary to maintain different code bases for the batch and high velocity data, thus resolving the lambda’s redundancy issues. However, it is more difficult to implement kappa architecture (Rao, 2021).

Rao (2021) explains that lambda provides a good balance in terms of speed, reliability and scalability. The batch layer provides fault tolerance and distributed storage, so the possibility of errors in the case of a system crash is low. Many data analysis scenarios can be covered since it provides access to real-time and off-line results. Lambda performs well because it offers access to a complete data set in a batch window. Furthermore, lambda is simple to implement. On the other hand, kappa enables direct read and write to the message queue, so batch processing is not needed during ingress. Since all data points are treated as streaming events, it also provides the ability to immediately see the state of all the data in the organization, resulting in faster query results.

### Self-Check Questions

1. Please complete the following sentence:

In a data lake, the ingestion layer links to data *sources* to enable the loading of data.

1. The analog data pond of a data lake processes the following type of data...
2. unstructured data
3. structured data
4. *semi-structured data*
5. What type of data do we use in a data lake to enable location and retrieval of the data from various and diverse locations?

*Metadata*

Summary

Ever-increasing volumes of big data are available to organizations. It is therefore vital to know the relevant frameworks and technologies that can be applied to effectively manage big data.

Hadoop is an open-source big data framework that is widely used to store and analyze big data. It enables reliable and scalable distributed data storage and analysis. Furthermore, Hiveprovides users with inexpensive data warehousing capabilities for semi-structured and unstructured data. Large and diverse data sets that are stored in distributed storage can be easily read, written and analyzed with Hive. Therefore, we discussed Hadoop and Hive.

Furthermore, since big data cannot be stored effectively in traditional DWH systems, the concept of a data lake emerged as a suitable and inexpensive alternative. Semi-structured and unstructured data are effectively stored in data lakes. In addition, organizations can store data in its raw and native format, and without having to first impose a rigid structure. Data lakes are modern architectures that are increasingly used by organizations to address their data management requirements. We therefore also explored data lakes in this Unit.

# 6. Data Warehouse Architecture (DWHA) Types

## Introduction

A data warehouse architecture (DWHA) describes the structure of a data warehouse (DWH). It describes relevant guidelines, standards, and services to effectively link the organization’s strategic requirements with its systems and applications, so that it can achieve business goals. A fitting DWHA ensures that up-to-date and relevant decision support information is consistently and accurately derived from the operational data. Furthermore, it promotes the efficient and cost-effective sharing of data and, as such, improves organizational productivity (Perkins, 2022).

There are several types of DWHAs to choose from. Selecting the architecture that is suitable for the organization is a key factor influencing the overall success of the DWH project. The organization’s requirements, data sources, data types, data formats and existing systems will dictate the choice to be made in terms of a DWHA (Sarad, 2022). The wrong architecture decision will result in issues such as a lack of scalability and problems with the performance of the DWH.

In this unit, we discuss the following pertinent DWHAs: hub-and-spoke; data mart bus; centralized; independent; federated; virtual; distributed; as well as a big DWHA. We also give an overview of what a DWHA entails. Lastly, we explore distribution options.

## 6.1 Hub-and-Spoke DWHA

The hub-and-spoke DWHA corresponds with the DWH approach that is presented by Inmon (2006). Accordingly, a hub-and-spoke DWH is also referred to as the Corporate Information Factory (CIF). The DWH is implemented by means of a top-down process. It is thus developed iteratively, meaning that the central DW is built first; data marts are then built and added, one-by-one, afterwards (Yang et al., 2019).

The organization’s atomic level data is maintained in a relational, normalized (3NF) format in the central DWH. Data marts are created and added for the subject and/or functional areas, based on the needs of the business users. A data mart can thus serve a department, functional business area or a specialized purpose such as data mining. The hub-and-spoke topology is illustrated next.

Hub-and-Spoke Topology (Venter, 2022)

Diagram

Description automatically generated

Data marts can be modelled using dimensional modeling as well as entity-relationship diagram principles. Data mart models can be visualized by notations such as ADAPT (Application Design for Analytical Processing Technologies), ERDs (entity-relationship diagrams) with detail in associated DISs (data item sets), or MERM (Multidimensional Entity-Relationship-Model). Regardless of how they are modelled and visualized, as the purpose of a DWH is to maintain the single version of the truth, they must always source their data from the central DWH (Inmon, 2006). Data marts can contain data in various forms, for example, a data mart’s data can be normalized, denormalized, summarized or aggregated (Matouk & Owoc, 2012).

The hub-and-spoke approach is suitable to develop a relational enterprise-wide DWH that stores structured data. Accordingly, traditional and relational database and DWH tools are used (Breslin, 2004).

### Self-Check Questions

1. Data marts can contain data in various forms. Please name two of the applicable forms.

*Normalized.*

*Denomarlized.*

*Summarized*

*Aggregated.*

## 6.2 Data Mart Bus DWHA

The data mart bus DWHA is represented by the Kimball approach to design a DWH—it is detailed in Kimball and Ross (2013). It entails a bottoms-up method to develop a DWH, meaning that business-oriented data marts are implemented incrementally and in an asynchronous manner. It includes a central bus matrix that links to several data marts. The data marts are based on organizational business processes. They are accordingly implemented to support the stakeholders and participants of business processes (Matouk & Owoc, 2012). The process thus starts by identifying the business requirements of a specific business process, for example, orders, billing or deliveries. Dimensional modeling principles are used to model the data mart bus DWH. The topology of a data mart bus data warehouse is illustrated next.

Data Mart Bus Topology (Venter, 2022)

Diagram

Description automatically generated

Enterprise-wide cohesion is achieved through the central bus matrix that indicates all the business processes and conformed dimension (Breslin, 2004). The bus matrix with the conformed dimensional tables ensures enterprise-wide consistency because the conformed dimensions include detailed descriptors and metrics that are commonly shared among business processes. So, the bus matrix and conformed dimensions also gives an enterprise view of a company’s data (Matouk & Owoc, 2012). In addition, it ensures that the data marts are logically integrated.

### Self-Check Questions

1. Please complete the following sentence.

The data marts in the data mart bus DWHA are based on the organizational *business* *processes*.

## 6.3 Centralized DWHA

The topology of the centralized DWHA is simple—it does not include any dependent data marts. Data are gathered from various organizational sources and stored in an enterprise-level, cross-functional information system (Bontempo & Zagelow, 1998).   
It applies the same data model to serve the needs of various individual business units or departments. The data model must therefore extend to take account of the needs of all the multiple business areas (Kashfi & Hajmoosaei, 2014). A centralized topology is illustrated below.

Centralized Topology (Venter, 2022)

Diagram

Description automatically generated

The data in a centralized DWH are generally stored in a presentation-ready format so that it can be used as-is by the business. However, data can also be made available in a format that can be analyzed further if needed (Yang et al., 2019). Both atomic level and summarized data are thus stored in the DWH. It can also include various logical dimensional views (Matouk & Owoc, 2012).

The centralized DWHA is suitable for an organization that has a fragmented decision support data environment but, at the same time, wants to integrate some of the data. The architecture offers advantages such as economies of scale and centralized system management capabilities (Bontempo & Zagelow, 1998).

### Self-Check Questions

1. Please mark the correct statement.
2. The centralized DWH stores only atomic data.
3. *The centralized DWH stores both atomic and summarized data.*
4. The centralized DWH stores only summarized.

## 6.4 Independent DWHA

The independent DWHA entails various standalone systems that are developed using a bottom-up approach. Each of the systems draws data directly from source systems. It does not include a central DWH or operational data store (ODS). Hence, it typically involves various hybrid and/or loosely coupled components such as individual DWHs and/or data marts (Yang et al., 2019). The data marts and/or different organizational databases and/or DMBSs can be loosely linked. Alternatively, they are not linked at all, so the individual components and systems operate on their own. Moreover, the individual structures execute analyses independently from each other. A topology with unlinked independent data marts is illustrated below.

Independent Topology (Venter, 2022)

Diagram

Description automatically generated

This DWHA is typically created when the organization must have a solution (or solutions) available in a very short space of time. However, over time, as more data structures are created in an organization, data redundancy will increase. Scalability of an independent DWHA is also limited (Naeem, 2019). An example of a practical implementation of an independent DWHA is given by Sahama and Croll (2007).

### Self-Check Questions

1. Please complete the following sentence:

The independent DWHA will result in data *redundancy* over time.

## 6.5 Federated DWHA

A federated DWHA is a variation of a distributed DWHA, where the global DWH servers as logical DWH for all local data warehouses. Accordingly, the federated DWHA leaves existing decision support structures such as data marts, DWHs and operational systems as they are. Hence, the organization does not attempt to integrate the complex decision support environment that is already in place into a single solution (Song, 2018).

The data in the federated DWH are stored in **autonomous** **data stores**. Furthermore, data stores are heterogeneous in nature, meaning that they have different storage structures as well as access languages and application programming interfaces (APIs). Federated DWHs do not consist of integrated data. Instead, the data are integrated on demand, i.e., in real-time and upon demand of the user. On-demand integration implies that a user views the data as integrated and accessed from a single, integrated data store. However, the data are not stored in such a way. Instead, data are accessed and integrated upon request of the users. Hence, the (source) data always remain in their original location and format. Although data are not integrated, it is presented to users as if residing in a single, integrated, and homogenous data store. Users can also access the data using the language and API of their choice (Van der Lans, 2012). A federated topology is illustrated next.

**Autonomous data stores**

The data stores can be used independently as well, i.e., they can operate outside the scope of the federated data model.

Federated Topology (Venter, 2022)

Diagram

Description automatically generated

A federated model is typically used by complex organizations, e.g., in the case where different companies merged or for an organization that went through a significant reorganization process. It will therefore be confronted with too many political and/or implementation-related challenges upon integration. Accordingly, it is typically used by companies that already have a preexisting, complex decision support environment in place and it will not be cost effective to rebuild all (Song, 2018).

### Self-Check Questions

1. Please complete the following sentence:

The data in a federated DWH are stored in data stores that are both *autonomous* and *heterogeneous* in nature.

## 6.6 Virtual DWHA

Traditional and relational DWHs cannot adapt rapidly to the changes that occur in the business environment. If a DWH was not built specifically for real-time processing or streaming, it cannot effectively provide real-time decision support capabilities. So, virtual DWHAs emerged. Virtual DWHs aim to overcome these shortcomings and provide real-time analytics. Accordingly, they provide real-time analytical processing and decision support capabilities. To achieve this, data virtualization techniques are applied (Ghosh et al., 2021). Consequently, a virtual DWH is basically a logical DWH that allows users to access the operational data directly via middleware tools. The operational data are thus used for analytical purposes as well. Hence, middleware software provides an interface between the operational systems and databases and front-end tools (Yang et al., 2019). It uses table views, if necessary and applicable, for to provide analytical capabilities.

Virtualization facilitates the use of various and diverse resources without concerning the user with technical details of where it resides, what interface or platform it uses, etc. (Ghosh et al., 2021). The user is unaware that data are accessed from various data stores (Van der Lans, 2012) and of the physical storage location of data, database platforms languages applicable API used, etc.

Data virtualization means that various and diverse data sources are presented to users as one integrated source; however, the data sources are not physically integrated. Hence, operational data sources are manipulated and analyzed using (limited) data integration functions to offer a unified view of the data to users. This may lead to performance problems in the operational systems. Data virtualization is illustrated below.

Data Virtualization (Venter, 2022)

A picture containing chart

Description automatically generated

According to Van der Lans (2012) there are many alternatives available to implement the data virtualization layer: A dedicated data virtualization server can be used so that multiple data stores appear to the users as if they all reside in a single data store. Or an enterprise service bus can be implemented to develop a layer of services that offer standardized access to data. It hides the technical details of how and where data are stored from front-end users. As another alternative, the DWH can be stored in a cloud storage solution so that users access the data through the cloud API. The actual data storage location of data will be hidden to users. However, cloud solutions may pose a data security risk. There may also be latency issues when accessing data in the cloud. Various vendors offer complete cloud DWH solutions, such as, AWS Redshift, Microsoft Azure, Google BigQuery and Snowflake (Sarad, 2022).

A virtual DWH is useful for an organization that do not want to redesign and change their existing underlying infrastructure; or when they have raw data in a standardized form and do not require complex analytics to be performed on the data. However, this approach has some drawbacks, such as, multiple databases may need continuous and expensive maintenance. Also, complex queries may take long to execute in cases where it is necessary to fetch the data from several separate databases. In addition, transformation software may still be needed to make data digestible for end users and reporting tools (AltexSoft, 2019).

### Self-Check Questions

1. Please mark the correct statement.
2. *In a virtual DWH the operational data is also used for analytical purposes.*
3. In a virtual DWH the operational data is hidden from the user.
4. In a virtual DWH the source data is integrated on-demand.

## 6.7 Distributed DWHA

A distributed DWHA entails various distributed DWHs that work in parallel. The DWHs can, as an example, be built on multi-node cloud computing platforms (Yang et al., 2019). So, the topology then entails various DWHs that are connected via a network. Alternatively, it can also include distributed data marts that are logically integrated. Each data mart will then address a specific departmental business problem. However, the data marts must adhere to predefined enterprise data and metadata models. This way, the data marts can still be shared among business processes and departments at a later stage (Kashfi & Hajmoosaei, 2014). It is illustrated below.

Distributed Topology (Venter, 2022)

Diagram

Description automatically generated

According to Yang et al. (2019) the data that reside in a distributed DWH can also be pre-processed, based on a pre-defined schema. Consequently, the data are physically distributed in RDBMSs that are structured according to the predefined schema. In addition, partitioning algorithms will be applied to fragment the DWH schema and allocate the fragments over computing nodes. The fragments can then be duplicated and stored in different computing nodes to increase performance—since data can be accessed faster. However, regardless of the fragmentation, from a user’s perspective, it will appear as if the data are stored in a single central enterprise DWH (Bontempo & Zagelow, 1998).

Distributed DWHs offers advantages such as they can be built fast. Furthermore, they are economical and inexpensive. They also provide a good return on investment while being relatively low risk. In addition, they are expandable and adapt easily when the business needs change (Kashfi & Hajmoosaei, 2014). However, a distributed DWHA is only suitable for organizations with light data loads (Bontempo & Zagelow, 1998).

### Self-Check Questions

1. Please complete the following sentence.

In a distributed DWHA the various distributed DWHs work in *parallel*.

## 6.8 Big DWHA

The big DWHA is used for big data, i.e., data that are semi-structured or unstructured. It is implemented on a platform such as Hadoop and applies a distributed file system, such as HDFS, to store data. Also, a technology such as MapReduce is used to process the data. Complimentary tools, e.g., Spark and Hive, can also be used to analyze the data and provide reporting mechanisms (Yang et al., 2019).

A big DWHA can include data that is processed using the Hadoop ecosystem as well as structured data from a RDBMS-type DWH. It can be implemented in various ways. For example, Hadoop can be used to ingest all types of data (including unstructured, semi-structured and structured). Thereafter, it will process and distribute all the data to a RDBMS-type environment. The consumption environment, i.e., the BI tools, are populated from the RDBMS. Alternatively, an Hadoop system can be used in parallel with the RDBMS environment (Mohanty et al., 2013). Both approaches are illustrated next.

Big DWHA Approach: Hadoop to RDBMS (Venter, 2022)

Diagram

Description automatically generated

Big DWHA Approach: Parallel Hadoop and RDBMS (Venter, 2022)

A screenshot of a computer

Description automatically generated with medium confidence

### Distributed Storage of Big Data

Big data are stored in the Hadoop distributed file system (HDFS) in a Hadoop cluster. It automatically replicates data across the nodes in the cluster. HDFS keeps a default of three copies of the data (Mohanty et al., 2013). This is illustrated next.

Replication of Data in HDFS (Venter, 2022)

A picture containing diagram

Description automatically generated

HBase is a non-relational, distributed wide-column store and another component of the Hadoop ecosystem that is useful to optimize data storage. It runs on top of HDFS and provides compression, in-memory-operations, indexing and transactional capabilities. Accordingly, multiple large, non-relational tables can be stored (distributed) beneath HDFS (Mohanty et al., 2013).

### Ingestion and Distribution of Data

The data stored in HDFS are acquired (ingested) from various sources, e.g., log files, data streams, online data, through ETL processes, and others. Different tools will be applied to distribute semi-structured and unstructured data versus structured data.

Semi-structured and unstructured are distributed using a tool such as Flume, a framework that is used to populate Hadoop with large amounts of data, especially streaming data. Flume entails a pipeline with sources, channels, and sinks. The Flume source consumes events from an external source (such as a web server) and store them in a passive storage system that is called a channel. The channel keeps events until they are consumed by the sink. When the sink consumes events, they are deleted from the channel and delivered to an external target that is a distributed file system (Apache Software Foundation, 2022).

Structured data are distributed using a platform such as Sqoop. It is useful to transfer bulky quantities of structured data between relational data stores and the distributed file system. It can import data from external relational databases to HDFS and HBase; Sqoop can also export data from a Hadoop cluster to a relational database or DWH (Azarmi, 2016). It is used with various relational databases platforms such as Teradata, Oracle, MySQL, Postgres, Data Vault, or Snowflake.

The Apache Sqoop project is officially retired; however, the tool is still widely used for bulk transfers of relational data.

### Processing of Data in HDFS

Different processing languages can be used to process raw bulk data in HDFS. At first MapReduce jobs had to be programmed using languages such as Java and Python to process data in Hadoop clusters—since it had to be processed using the MapReduce processing model. However, with the introduction of YARN, it became possible to use other data processing models such as Spark and Hive as well (Azarmi, 2016). Hive is typically used for to process batches of data while Spark is used to process real-time data streams. The formal query capabilities that HiveQL offers means that Hadoop can be used as a DWH-type system. Accordingly, Hive facilitates the summarization of data, ad-hoc queries, and analysis of data in both HDFS and HBase. It also enables easy integration with BI and data visualization tools (Mohanty et al., 2013).

### Self-Check Questions

1. Please mark the correct statement.
2. HDFS facilitates the use of other data processing models as well, i.e., in addition to MapReduce.
3. *YARN facilitates the use of other data processing models as well, i.e., in addition to MapReduce.*
4. Flume facilitates the use of other data processing models as well, i.e., in addition to MapReduce.

## 6.9 Architecture Overview and Distribution

The DWHA should include various items to ensure successful implementation of the DWH. It includes a DWH data model that documents the organization’s data elements that are used to inform decisions. The data elements encapsulate the company’s key performance indicators. So, at any point in time, they can be used to get an indication of how well the organization is performing. Furthermore, every key data entity should be clearly and unambiguously defined, meaning it must include a description of how and where each data entity is used. In addition, the formulas and methods that are used to derive, aggregate, and summarize data must be defined. The ETL processes to cleanse, transform, and integrate data must also be clarified (Perkins, 2022). Data sources, types and formats should be examined and defined to ensure that ETL logic and interfaces are developed correctly (Sarad, 2022).

Metadata is another important aspect of a DWHA. Structural metadata is useful to manage and control the creation of a DWH. It describes the structure and content of the DWH. Moreover, access metadata is used to regulate access to the DWH. It is vital to ensure quality and keep data secure (Perkins, 2022).

The applications and data in a DWHA must be distributed and managed according to specific distributions. Depending on the platform used, different distribution options are available. Azure distributions as well as Hadoop distributions are discussed next.

### Azure Distributions

Azure Synapse is a massively parallel processing (MPP) database system, consisting of **synapse instances** that store data. The data in the instances are dispersed across 60 underlying databases—these databases are called distributions. It is necessary to organize the data so that queries are handled quickly and efficiently.

**Synapse instance**s

Data stores that store data in columnar formats; it enables distributed querying capabilities that suits the performance of OLAP workloads better.

Kaushikk (2020) and Kumar (2021) explain that Azure distributes data as follows:

* Round robin—is the default method that is used when no strategy is specified. It distributes data sequentially and equally among underlying distributions. It provides fast performance for the staging phase of ETL.
* Hash—uses a hash function. It involves a **deterministic** **algorithm** to assign a row to a distribution. It provides a good query performance for large tables that contain joins and aggregations.

**Deterministic algorithm**

An algorithm that always produces the same output when given a specific input.

* Replicated tables—caches a complete copy of each compute node. It uses additional storage and overheads, so it is impractical when working with large tables. It is useful for small tables, such as lookup tables.

### Hadoop Distributions

When Hadoop-related technologies are used, Azarmi (2016) suggests that a Hadoop distribution be used to assemble and create the required technologies. It entails more than the distribution of data. It includes the complete setup of tools and technologies as well. Cloudera CDH and Hortonworks HDP are examples of popular Hadoop distributions.

Both Cloudera CDH and Hortonworks HDP use the open-source Hadoop components. However, Cloudera also offers vendor-specific, **proprietary software** that can be used in conjunction with the open-source components (Cloudera, 2021). Accordingly, they offer open-source projects that are free; and they also offer proprietary software to be purchased and implemented on top of the open-source components. The vendor-specific components are Cloudera Manager, Impala and Cloudera Search. Cloudera Manager is used to manage the suite of products. Impala provides an interface that can process SQL queries fast. Cloudera Search provides real-time access to products (Savaram, 2022).

**Proprietary software**

Non-free software where the proprietor (legal owner) reserves and restricts the licensing rights to use, modify or share the software.

On the other hand, Hortonworks offers a fully open-source distribution. They employ the following open-source (and thus free) software: Ambari, which is used to manage the suite; Stinger is used to handle queries; and Apache Solr is used to search for data. Though, additional proprietary software is used for data warehousing—they partner with a vendor (Teradata) to provide specific DWH capabilities. Cloudera, on the other hand, claims to be an enterprise data hub; it is therefore not necessary to purchase and employ additional DWH software with a Cloudera distribution (Savaram, 2022). Integrating big data architectures with Data Vault 2.0, to create big data DWHAs, are become increasing popular—a data vault DWH is flexible and scalable. It is illustrated next.

Data Vault and Big DWHA Integration (Venter, 2022)

Diagram

Description automatically generated

### Self-Check Questions

1. Please complete the following sentence:

*Structural* metadata is useful to manage and control the creation of a DWH.

Summary

The DWHA must be documented clearly at the beginning of the DWH project. It must unambiguously describe the structure of the DWH, including all applicable guidelines, standards and services that links the organization’s strategic data requirements with all the integrated systems and applications that are used to achieve business goals. Furthermore, it must ensure that analytical information is consistently and accurately derived from the operational data. It is therefore vital to choose a fitting DWHA for a specific organization. The organizational requirements in terms of decision support capabilities, as well as existing data and information-related ecosystem and systems, must be considered—it is crucial to ensure that the right architecture choice is made. We therefore discussed various architectures, i.e., a hub-and-spoke, data mart bus, centralized, independent, federated, virtual, distributed, and big DWHA. An overview of what a DWHA entails is also given in this unit. Lastly, we explored the distribution options for well-known technologies such as Azure and Hadoop. These are positioned so that the best option can be chosen for an organization.

# Unit 7 – Application-Specific Data Warehouses (DWHs)

**Study Goals**

On completion of this unit, you will be able to …

… explain the key differences between top-down and bottom-up data warehouse (DWH) approaches.

… differentiate between real-time, closed-loop, and active DWHs.

… discuss practical DWH implementations.

# 7. Application-Specific Data Warehouses (DWHs)

## Introduction

Various approaches are available to design, develop and implement DWH solutions. We generally differentiate, on the highest level, between the top-down and bottom-up approaches. A well-known top-down method is the Corporate Information Factory (CIF) approach that is proposed by Inmon (2006). Alternatively, a well-known bottom-up method is the data mart bus approach. It is also known as the Kimball Lifecycle Methodology (Kimball & Ross, 2013). A third, increasingly applied, architecture is the Data Vault methodology from Linstedt. It uses a hybrid design methodology which follows a top down architecture and bottom-up design. Therefore, some advantages and differentiating factors of all three methods are discussed in this unit

We also explore real-time DWHs, closed-loop DWHs and active DWHs. The aim of these are to overcome latency issues and shortcomings of traditional DWHs. Hence, a real-time DWH offers immediate (real-time) updating of DWH data. The closed-loop DWH takes it one step further. It facilitates the updating of DWH data in real-time, in addition to updating of the operational systems with relevant changes that occured in the DWH. The active DWH is proposed and implemented by Teradata (2022). It also entails updating of the DWH in real-time. Moreover, it enables the automation of routine tasks and decisions. Lastly, we discuss practical DWH implementations.

## 7.1 Top-Down and Bottom-Up Approaches

The implementation of a DWH will essentially be done according to either a top-down or a bottom-up approach. The top-down approach is characterized by the CIF method of Inmon (2006). On the other hand, the data mart bus DWH structure, also known as the Kimball Lifecycle Methodology, is a bottom-up approach that is prescribed by Kimball and Ross (2013). Both methods offer several advantages and differentiating factors. Both approaches are accordingly widely used in industry. Some of the key differentiating factors are considered next.

The data models of these approaches differ as follows: The CIF uses a normalized data model that is designed first. It is a data-oriented method. The CIF comprises a central DWH with atomic data. Hence, the DWH stores granular data, i.e., at the lowest level of detail. The dependent data marts are created from the central DWH. The central DWH offers a logical structure for business intelligence (BI) (Inmon et al., 2001; Inmon & Linstedt, 2015). Since the CIF DWH stores analytical data of the entire organization, it is complex and hence takes a long time to design and implement. On the other hand, a data mart bus DWH is business process oriented. The data modeling process is relatively straightforward (in comparison to the CIF) and the business-focused data marts can be designed and implemented in a relatively short period of time (George, 2012). The data mart bus DWH’s data model is denormalized and the data marts are designed for one business process at a time, and in such a way that they provide high levels of query performance. As the approach is business process focused, business users find the dimensional data mart structure easy to understand and access (when compared to a CIF model) (Kimball & Ross, 2010, 2013). The data marts are integrated by means of uniform descriptive attributes that are shared among business process.

In a data mart bus DWH we develop data marts individually. However, the data marts are still integrated and combined in the long term to form a consistent DWH structure for an enterprise. Each data mart links to detailed, denormalized and descriptive tables (that are called dimension tables). They define and describe the data entities. When a dimension table contains multiple valid relationships between itself and other tables, it is called a role-playing dimension. Date dimensions are often role-playing dimensions because they are used by multiple facts. The table will have a different meaning in each fact that uses the dimension data. Each data mart is also associated with transaction data. Transactional data is true to declared grain and can be normalized or denormalized, i.e., it can store data that are atomic, summarized, or aggregated. The grain of transaction tables (that are called fact tables) must be explicitly defined upfront.

Data are updated in the CIF based on a continuous and discrete time frame (George, 2012). Consequently, the historical data are updated at regular intervals (in batches) and when specific events happen. In contrast, new transactional data are included in the data mart bus DWH in scheduled batches that typically run during off-peak times. The schedule can be, e.g., daily, weekly, or monthly. On the other hand, the data mart bus DWH’s descriptive (dimension) tables are updated based on business events. For example, when data entries must be updated due to an address change of a client or a new product to be added. Dimension tables are updated by means of pre-defined slowly changing functions. These functions are defined as part of the ETL process.

The three main types of slowly changing dimension methods that are used to update dimension tables in the data mart bus DWH include the following: overwrite an attribute value; add a row in a dimension table; and add a column in the dimension table. Kimball and Ross (2010) and Kimball et al. (2008) explain these methods as follows:

* Type 1:

Overwrite an attribute value—overwrites an old value with a new value. Changes are not tracked, and the old value is not preserved. This method is used, e.g., when a customer's mobile number changes and must be updated in the database.

* Type 2:

Add a row in the dimension table—creates a new row in the dimension table to capture new values of a changes attribute. The attributes in the new row are in effect as of the time of the change and moving forward. Each row documents with two timestamps, from when until when the changed dataset is valid. Each new changed dimension dataset changes the value of the “valid\_to”-field of the predecessing dataset to the valueIt of the “valid\_from” of the new dataset. Therefore, this SD-type preserves the previous attributes that are still associated with measurement (fact table) data prior to the change. This method is used, e.g., when a bus route changes. Historical entries must still be associated with the previous bus route, and entries from the moment that the change happened must be associated with the new bus route that is in effect.

* Type 3:

Adds a column in the dimension table—changes the table by widening it. An additional column is added to keep track of additional attributes that have not been included previously. This method is used, e.g., when a department wants to keep track of e-mail addresses, in addition to postal addresses, going forward.

The choice in terms of a top-down of bottom-up development approach that will be best suited for an organization will be determined by the organizational business objectives, nature of the business, the time and cost that can be spent, complexity of the business, and the dependencies that exist between various organizational functions. The CIF takes generally take a long time to design and is most suitable for large and complex organizations. It is also expensive to implement so the organization must have a large budget for the DWH project (George, 2012). On the other hand, the data mart bus method focuses on quick wins. The development of single, individual data marts is also less expensive. It is therefore suitable for implementations that must be completed relatively fast and at a lower cost. It will also be the preferred approach where local optimization is a key consideration (George, 2012).

A hybrid approach, combining a top-down architecture with a bottom-up data model design, is the Data Vault (DV), as prescribed by Linstedt (2015). The latest version is called the Data Vault 2.0. methodology. This approach adapted core standards from the software engineering field and applied them in the field of data warehousing. The major activities included in a DV project are project planning, project execution, as well as review and improvement.

Within each of these activities, major roles and responsibilities are defined. Within the project planning phase, a business sponsor, technical business analyst and project manager, information technology manager, ETL developer, report developer, data/ information architecture, metadata manager as well as change manager collaborate to ensure that both the business and technical requirements are established and incorporated in the project plan (Linstedt, 2015).

During the project execution phase aspects of the traditional software development lifecycle (SDLC) approach and the modern (agile) Scrum methodology is combined. Accordingly, a (short) iteration of a SDLC is executed within Scrum sprints that are aimed to achieve the following: First, it is vital to envision the initial architecture. Second, details are fleshed out (based on just-in-time principles). Third, the team focuses to continuously prove that the architecture (still) works. Fourth, the team focuses on the usage and inclusion of any newly emerging facts and data. Fifth, they continue to organize work according to requirements. Last, the team ensures that all business and technical stakeholders are continuously and actively involved in the DV project (Linstedt, 2015).

The review and improvement phase aims to determine whether the product meets all the expectations and documented requirements. Core principles from practices from process improvement as well as quality control and management fields, such as Six Sigma and Total Quality Management (TQM), are incorporated here. The aims of this phase are to reduce costs and increase profits, in addition to improving customer satisfaction. TQM principles are applied to ensure the quality of data in the DV, and the quality of the overall DWH structure (Linstedt, 2015).

### Self-Check Questions

1. Please complete the following sentence:

The central DWH offers a logical, enterprise-wide structure for *business intelligence*.

## 7.2 Real-Time DWHs

The real-time DWH has a relatively low latency, meaning that the time delay between the occurrence of an event, and importing the data that describe the event into the DWH, is relatively short, usually below a minute. A real-time DWH is not updated according to a schedule and in batches. Instead, triggers in the operational systems are used to initiate real-time updates in the DWH (Valencio et. Al. 2013) ). Data associated with business activities are captured at the time when they occur; and data of completed business activities flow directly into the DWH.

Advantages of the real-time DWH include the following: Operational data is available immediately for analysis purposes as well. Furthermore, the DWH reflects the most current (real-time) situation, so the answer to a query always reflects the actual and current state of the entity, at the time of running the query (Mohania et al., 2009).

Real-time data warehousing offers several benefits, as explained by Abdullahi, (2021): It minimizes operational overhead because it limits manual data extraction. It offers instant decision-making to support the business. Furthermore, it improves governance and data security because real-time data integration means fewer updates and reconciliations. Active DWHs offers real-time validation, quality assurance and checking of errors, so data quality is improved. Costs are reduced through the offering of predictive analytic capabilities and automated diagnostic reporting. Manual processing errors are reduced because they are detected early and resolved. In addition, it increases operational efficiency because data can be retrieved fast. Faster response times result in higher customer satisfaction. The flexibility of the architecture leads to increased competitiveness and lower capital expenditures since infrastructure is utilized efficiently. Also, business agility is augmented, and business resilience increased because the organization is less dependent on manual processing (Abdullahi, 2021).

### Self-Check Questions

1. Please complete the following sentence:

The data of *completed* business activities flow *directly* into the real-time DWH.

## 7.3 Closed-Loop DWHs

Traditional DWHs generally have a one-way relationship with the operational systems that supply them with data. However, it is possible to loop an operational system and DWH so that they feed each other with data. In view of that, the closed-loop DWH involves a bidirectional process to optimize transactional processing—the data that the operational systems receive from the DWH is applied to optimize the operational environment (Radding, 2005).

The closed-loop DWH can also be referred to as an integrated DWH. Triggers in the operational systems initiate immediate updating of the DWH (like a real-time DWH). However, in addition, the DWH updates the operational systems as well. Operational systems are thus updated with relevant changes that occurred in the DWH (Finnestad et al., 2009).

This DWH is challenging and expensive to implement, due to the various and diverse operational and other systems as well as APIs that are used in an organization. These are rarely integrated, so they may have difficulty communicating with each other. In addition, the level of detail in the DWH may differ from the level of detail of the data stored in the operational systems—since a DWH can contain summarized and/or aggregated data as well. This causes structural problems with the writing of data from a DWH back to the operational systems. As an example, enterprise resource planning (ERP) systems are tightly integrated with business process and may not be compatible with the summarized or aggregated data from the DWH. It may also introduce logical errors that are difficult (or impossible) to fix if the original data in the source systems have been overwritten with data from the DWH (Finnestad et al., 2009).

### Self-Check Questions

1. Please mark the correct statement.
2. A closed-loop DWH is difficult to implement where an organization has many off-line systems.
3. *A closed-loop DWH is difficult to implement where an organization has various and diverse operational systems.*
4. A closed-loop DWH is difficult to implement where the organization uses many heterogenous data stores.

## 7.4 Active DWHs

The active DWH is a “logically consistent store of detailed data available for strategic, tactical, and event-driven business decision making” through the provision of “a single up-to-date view of the enterprise” (Teradata, 2014, chapter 1). It allows the capturing of transaction details, at the exact moment when they occur or change, as well as timely integration thereof into the DWH. Hence, updates to critical data happens very close to real-time. The term active DWH was coined by Teradata in 2001 (Teradata, 2022).

The DWH offers users the option to automate routine tasks and decisions. In addition, it automatically exports operational decisions to OLTP systems. However, scheduled batch and cycle refreshes are maintained and therefore continue to run (Mohania et al., 2009). So, the DWH supports both **tactical queries**, that should be returned within seconds, alongside traditional, strategic decision support.

**Tactical queries**

Short queries aimed at enabling quick action-taking and/or decision-making in time sensitive environments.

The active DWH includes the following functionalities (Teradata, 2014):

* Active load—entails active loading of data in a non-disruptive way, while at the same time continuing to process other workloads.
* Active access—facilitates processing of tactical queries, to access analytical information quickly and consistently. It provides recurring decision support information for operational business processes.
* Active events—enables automatic detection of a business event, followed by the appropriate application of business rules to update both current and historical data. In addition, applicable operational actions can be initiated. The operational actions can be initiated automatically by applying business rules. Or, as an alternative, different options can be presented to users for review, so that the users can manually choose the actions to implement.
* Active workload management—facilitates real-time management of mixed workloads to dynamically optimize the utilization of system resources.
* Active enterprise integration—simplifies the coordination of applications and business processes on an enterprise level.
* Active availability—considers the effect of enterprise-wide downtime and, in view of that, identifies the application-specific requirements in terms of availability, recoverability, and performance.

### Self-Check Questions

1. Please give the name of the functionality of an active DWH that facilitates automatic detection of a business event, so that business rules can be used to automatically update data.

*Active events*

1. Please complete the following sentence:

The active DWH supports *tactical queries* that should be returned very fast.

1. Please mark the correct statement.
2. The active DWH does not provide strategic decision support information.
3. Scheduled batch refreshes are not executed in the active DWH.
4. *In the active DWH scheduled batch refreshes are maintained and continue to run.*

## 7.5 Practical Implementations

Some key trends emerged around the usage of data that impacted on the implementations of DWHs and DWHAs. Reinsel et al. (2018) argue that data is at the center of digital transformation and drives digitization processes. Accordingly, organizations continue to leverage data to improve their core business in terms of, e.g., improving customer experience, penetrating new markets, increasing efficiency of employees and processes, etc. For this, trillions of data pieces must be stored and analyzed. Many organizations are moving to cloud solutions to store data. Furthermore, organizations are increasingly moving towards real-time data analytics. Historical and real-time data are also increasingly infused into business workflows, as in the case of active DWHs.

To ensure that a DWH remains efficient over time, Abdullahi (2021) posits that the following use cases will ensure that the DWH continues to offer value in an ever-changing environment:

* Tactical reporting—since data are stored in a DWH for reporting purposes, and DWHs are optimized for high-performance queries, they should also be ideally suited for ad-hoc reporting. It should be able to provide instantaneous answers to a wide range of queries.
* Integration with big data—the appropriate architecture will enable integration with big data and automate big data analysis.
* Natural language processing—many organizations are implementing basic robotic process automation (RPA) and gather data through, e.g., live chat bots interacting with customers. Collected data must be integrated into existing data related to customers profiles, so that it can be analyzed.
* Auditing and compliance—electronic copies of data stored in DWHs eases the tedious task of adhering to regulatory aspects of a business.
* Data-mining analytics—stores massive amounts of data centrally so that it can be analyzed easily. It provides valuable business insights, e.g., indicating optimal promotional strategies.
* Ensuring data quality—having a DWH enables teams to automate the processes to identify and correct errors in their databases.

### Self-Check Questions

1. Please complete the following sentence:

An *active* data warehouse infuses the use of historical and real-time data into business workflows.

Summary

In this unit we discussed advantages and discerning aspects of the popular top-down DWH approach, i.e., the CIF DWH, versus the equally popular bottom-up method, i.e., the data mart bus DWH and the hybrid approach, i.e. the Data Vault. All three approaches have strenghts and weaknesses. We notice that the CIF is most suitable for large and complex organizations that are data-driven; whereas the data mart bus DWH is suitable where the organization is business process-driven and the aim is local optimization.

We also differentiated between types of DWHs, including real-time DWHs, closed-loop DWHs and active DWHs. These DWHs aim to overcome some of the issues and limitations of the more traditional DWHs. Accordingly, a real-time DWH is updated when transactions are completed in operational systems. Closed-loop (interactive) DWHs also facilitate updating of the DWH in real-time. Additionally, in a closed-loop DWH, the operational systems are updated with applicable changes that happened in the DWH. The active DWH is also updated in real-time. Furthermore, routine tasks and decisions are automated in the active DWH.

Lastly, we noticed that digital transformation and digitization is driven by data. So, DWHs play a crucial role and must be kept up to date, to ensure that they continuously add value to organizations. We therefore identify use cases that can ensure the continued relevance of DWHs for organizational improvement