1

Core Data Concepts

Data generation and data processing have been growing exponentially in recent years. Data is being generated and processed everywhere, in information systems, cell phones, smartwatches, Smart TVs, city buses and subways, cars, among others. Knowing how to capture and process this data to generate intelligence is today the main competitive advantage in the market.

To start understanding how these technologies and solutions work, it is necessary to know the concepts of data storage and processing.

By the end of this chapter, you will be able to understand:  
• The types of data and how to store it  
• What is relational and non-relational data  
• What is analytics data  
• How to differentiate the workloads on data

**Understanding the core data terminologies**

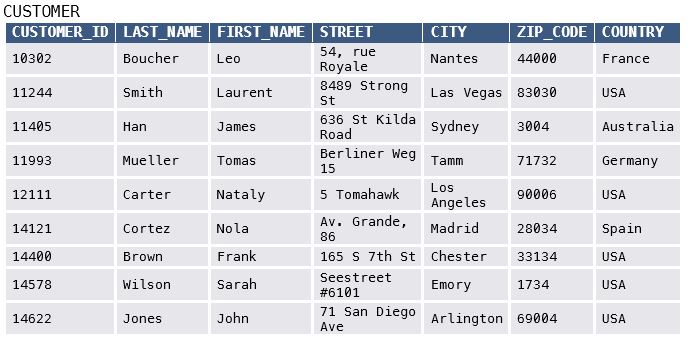
To start, let's understand the terminologies used in the data world, so that all the next concepts are easily interpreted to be applied to technologies.

**What is data?**

A data is a “record”, or also called a “fact”, which can be a number, a text or descriptions used in to take decisions. Data only generates intelligence when processed, these are then called information or insights.

Data is classified into 3 basic formats, *structured*, *semi-structured* and *unstructured* data.

*Structured* data are formatted, typically stored in a table represented by columns and rows. This data is found in relational databases, which organize their table structures in a way that creates relationships between them.

The image below shows an example of a simple table with structured data:

In this example the table called “CUSTOMER” has 7 columns and 9 records (rows) with different values.

The “CUSTOMER” table could be part of a customer management system (CRM) database, for example, a financial and process control system (ERP), among other types of business applications.

*Semi-structured* data are structures where records have attributes like columns but are not organized in a tabular way like structured data. One of the most used formats for semi-structured data is JSON (JavaScript Object Notation) files. Below the example demonstrates a structure of a JSON file also considering the registration of a CUSTOMER:

## JSON FILE - Document 1 ##

{

"CUSTOMER\_ID": "10302",

"NAME":

{

"FIRST\_NAME": "Leo",

"LAST\_NAME": "Boucher"

},

"ADDRESS":

{

"STREET": "54, rue Royale",

"CITY": "Nantes",

"ZIP\_CODE": "44000",

"COUNTRY": "France"

}

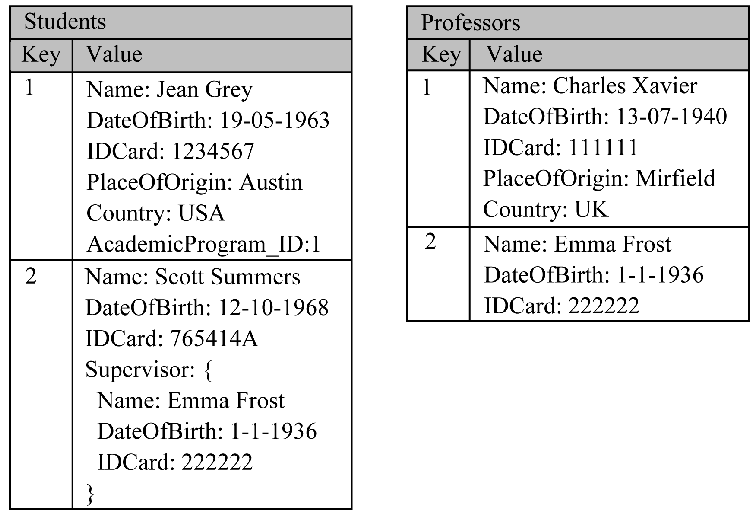
}

In this example we can see that each JSON file contains a record, like the rows of the structured data table, but there are other formats of JSON and similar files, which contain multiple records in the same file.

In addition to formats of *semi-structured* data, there are also data in key-value format and graph databases, which are considered semi-structured data too.

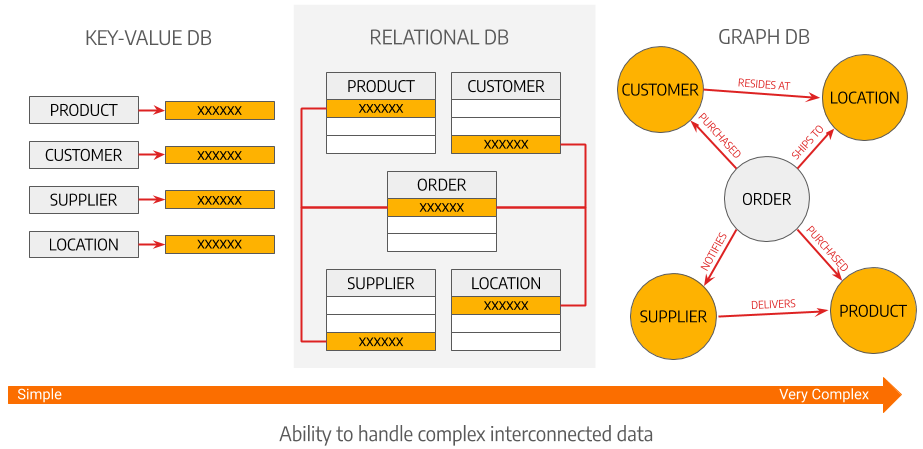
The key-value database stores data in a related array format. These arrays have a unique identification key per record. Values written to a record can have a variety of formats, including numbers, texts, and even full JSON files.

Below is an example of a key-value database:



As you can see in the illustration, each record can contain different attributes. They are stored in a single collection, with no pre-defined schema, tables and columns, and no relationships between the entities, this differs the key-value database from the relational database.

The Graph database is used to store data that require complex relationships. A Graph database contains nodes (object information) and edges (object relationship information), that is, it predetermines what an object is and the relationships they will have with each other, but the records can contain different formats. Below is a representation of nodes and edges in a Graph database of sales and deliveries:



In addition to structured and semi-structured data, there are also *unstructured* data such as audios, videos, images, or binary records without a defined organization.

These data can also be processed to generate information, but the type of storage and processing for them is different from the first two options. It is common, for example, for unstructured data such as audio to be transcribed using artificial intelligence, generating a mass of semi-structured data for processing.

**How is data stored in a modern cloud environment?**

Depending on the data format, structured, semi-structured and unstructured cloud platforms have different solutions. In Azure we can count on Azure SQL Database, Azure SQL Database for PostgreSQL, and Azure SQL services Database for MySQL and database servers installed on virtual machines such as SQL Server on a virtual machine in Azure, to store *structured data*. These are called relational databases.

*Semi-structured data* can be stored in Azure CosmosDB and *unstructured data* such as videos and images can be stored in Azure Blob Storage in a format called Azure Data Lake Store, optimized for queries and processing.

These services are delivered by Azure in the following formats:

* **IaaS (Infrastructure as a Service)** – Databases deployed at Virtual Machines
* **PaaS (Platform as a Service)** – Managed database services, where the responsibility for managing the virtual machine and the operating system lies with Azure

These database services to be used must be provisioned and configured, to receive the data properly.

One of the most important aspects after provisioning a service is the access control configuration. Azure allows you to create custom access role control, but in general we maintain at least 3 profiles:

* **Read-only** – users can read existing data on that service, but they cannot add new records, edit, or delete them.
* **Read/Write** – users can read, create, delete, and edit records
* **Owner** – Higher access privilege, including the ability to manage permission for other users to use this data.

With these configured profiles, you will be able to add users to the profiles to access the data storages/databases.

An example:

You are an administrator of a CUSTOMER database, you have the Owner profile, and you configure access to this database for the leader of the commercial area to Read/Write, and for salespeople as Read-only.

In addition to the permissions configuration, it is important to review all network configurations, data retention and backup patterns, among other administrative activities that will be addressed in the next chapters of this book.

**Describing a data solution**

To define which service is the best for your data solution, you need to evaluate the details of the use case. After this analysis we can define the solution between a transactional or an analytical database.

**Transactional databases**

Transactional are databases used by systems for basic operations: create, read, update, and delete. Transactional systems are considered the core of the informatization of business processes, as with these basic operations with can create entities such as customers, products, stores, sales transactions, among, to store these important data.

A transactional database is commonly known as OLTP (Online Transaction Processing), for its behavior of online operations between the application and the database.

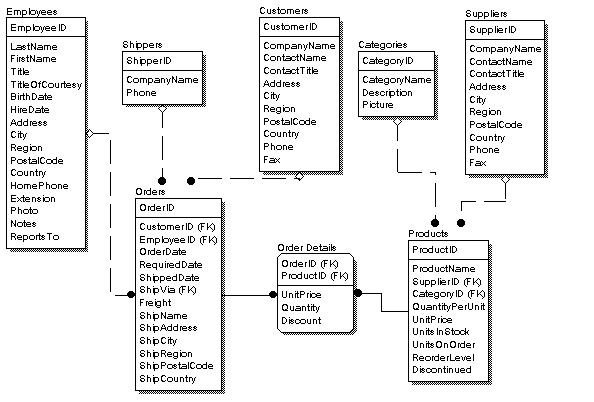
For a proper organization, transactional databases usually have their data segmented into entities, which can be tables (or not), with or without a relationship between these entities to facilitate the correlation between these data.

For example, an e-commerce database can be structured with a table called "Shopping\_Cart", which represents the products that are being selected in the store during user navigation and another called "Purchases" with the completed transaction records.

The process of segmenting entities in a database is called *normalization*, which will be covered in more detail later in this book.

The format of a normalized transactional database is optimized for transactional organization, but it is not the best format for data exploration and analysis.

Below is an example of a relational transactional database:



**Analytical databases**

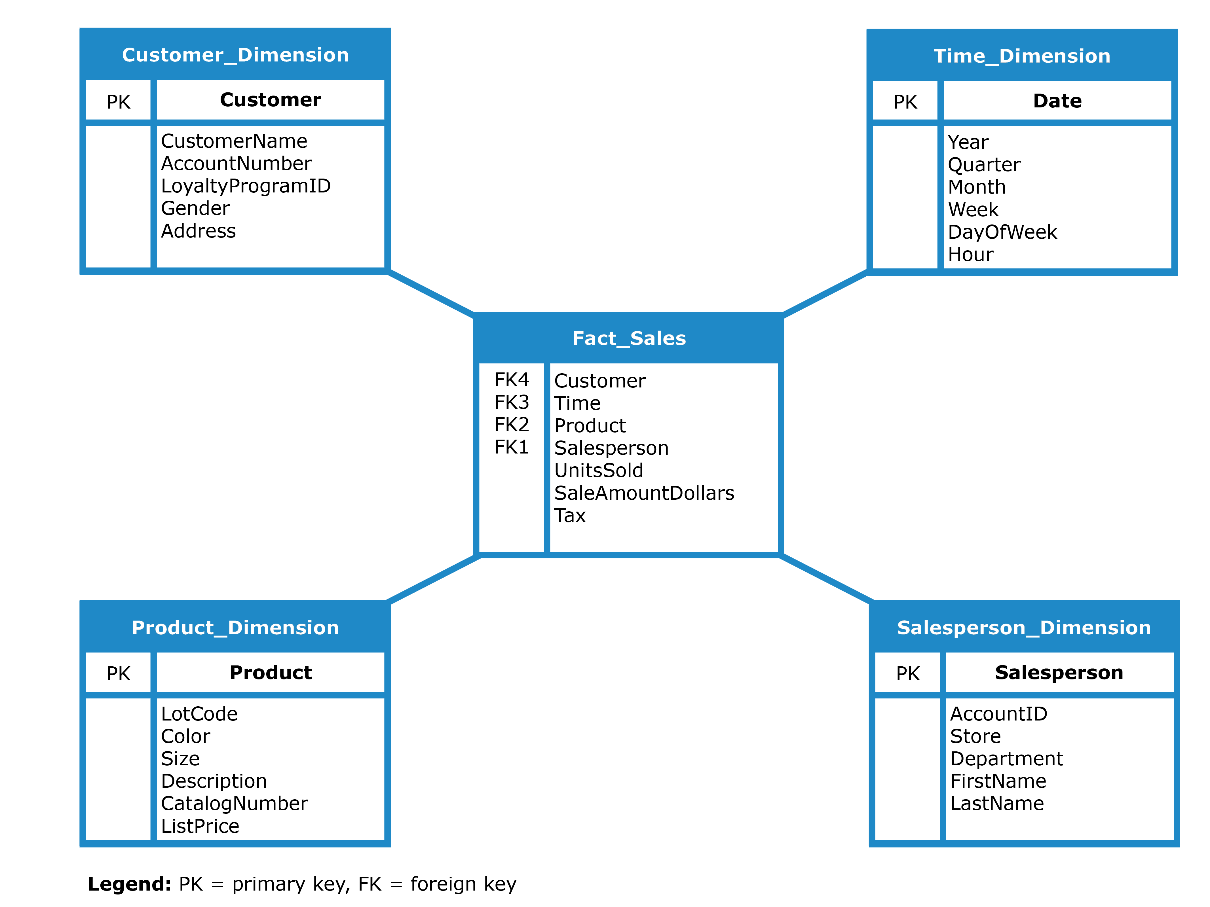
When the data storage must be optimized to perform data exploration and analysis, the data storage organization is different from transactional databases. In this case we prioritize the data aggregations and relationships for data consumption and exploration, these specialized data storages are called Analytical Databases.

Analytical databases are often called OLAP (Online analytical processing) and have undergone a great evolution in recent years with the emergence of Datawarehouse and Big Data platforms.

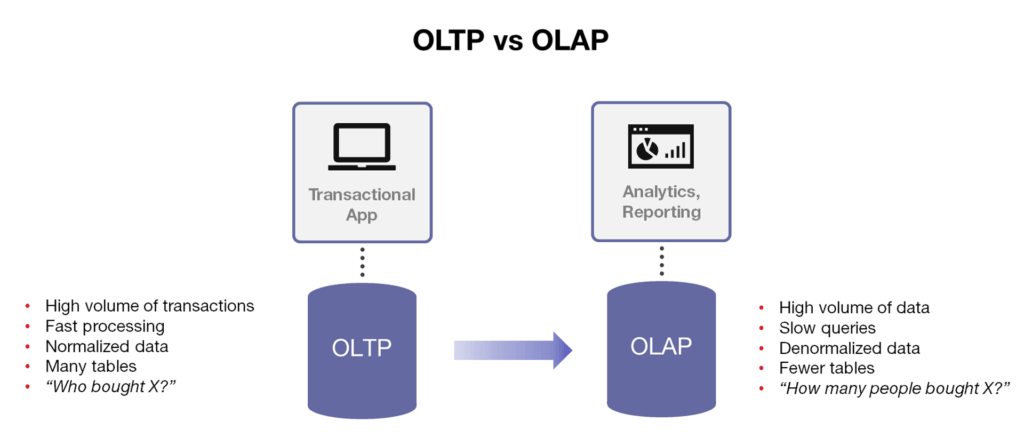
Analytical databases are constituted through a process of data ingestion, then they are responsible for processing and transforming the data into information and then making this processed information available for consumption. Below the concepts of these 3 steps:

* **Data ingestion –** process responsible for connecting to transactional databases or other data sources to collect raw transaction information and include it in the analytical database
* **Data processing –** process performed by the OLAP platform to create a data model, organizing entities, performing indicator calculations, and defining metrics for data consumption
* **Data query –** After the data model is loaded with the proper organization for querying, data manipulation and reporting tools can connect to the OLAP platform to perform your queries.

Below is an example of a structured data model in an OLAP database:



In the illustration below a simple comparison about OLTP and OLAP databases:



**Important note**

There are modern data storage platforms that aim to unite OLTP and OLAP on the same platform, but these databases, often called NewSQL, still need to mature their structures to deliver the best of transactional and analytical worlds in the same database. The industry standard is to keep transactional and analytical data structures separate.

**Defining the data type and proper storage**

Categorizing the data, to identify its types and best solutions for your storage is an important process, not just evaluating if it is structured, unstructured or semi-structured. In this section you will learn about the characteristics of different types of data.

**Characteristics of relational and non-relational databases**

Relational databases are the most traditional and used database format, as they have an easy-to-understand design and a simple tabular data model like other simple platforms such as Excel spreadsheets. Relational databases have pre-defined schemas, which are the structures of their tables, containing its columns, the data type of each column and other parameters such as primary and secondary keys used in relationships.

However, relational databases with these rigid schemas can pose challenges as presented in the following example.

Your CRM (Customer Relationship Management) system has a database structure with a CUSTOMER table, where you intend to store customer data, CUSTOMER\_NAME, ADDRESS, MOBILE\_PHONE and ZIP\_CODE. To do this, you start by creating a CUSTOMER table with 4 fields:

|  |
| --- |
| **CUSTOMER** |
| CUSTOMER\_ID – Number CUSTOMER\_NAME – Text ADDRESS – Text MOBILE\_PHONE – Text ZIP\_CODE – Number |

However, after setting up this table, you realize that you have clients that have more than one ADDRESS and ZIP\_CODE, and even more than one MOBILE\_PHONE. **How to solve this issue?**

To face problems like this one, a process called *normalization* was created. *Normalization* is done when there is a need to split a table, like CUSTOMER in this example, into more child tables that are correlated to the initial table.

Therefore, we can change the CUSTOMER table as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMER** |  | **CUSTOMER\_ADDRESS** |  | **ADDRESS** |
| CUSTOMER\_ID – Number CUSTOMER\_NAME – Text | CUSTOMER\_ID – Number ADDRESS\_ID – Number | ADDRESS\_ID – Number ADDRESS – Text |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMER\_MOBILE\_PHONE** |  | **MOBILE\_PHONE** |  | **CUSTOMER\_ZIPCODE** |
| CUSTOMER\_ID – Number MOBILE\_PHONE\_ID – Number | MOBILE\_PHONE\_ID – Number MOBILE\_PHONE – Number | CUSTOMER\_ID – Number ZIPCODE\_ID – Number |

|  |  |
| --- | --- |
| **ZIPCODE** |  |
| ZIPCODE\_ID – Number ZIP\_CODE – Number |

Non-relational databases allow you to store data in its original format, without having to pre-defined schema as in relational databases. The most common non-relational storage format is document storage, where each record in the database is an independent file. The benefit is that each file can have different and unique attributes.

On the other hand, the files being independent can bring some challenges such as:

* **Data Duplication** – Going back to our CUSTOMER entity example in a relational database, when 2 or more customers live at 1 address, the database records that relationship, and the normalized database only keeps 1 address record. But in a non-relational database, if 2 customers live at the same address, this address will be present in the record of the first customer and the second customer as well, independently.

Relational Database:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CUSTOMER** |  | **CUSTOMER\_ADDRESS** |  | **ADDRESS** |
| |  |  | | --- | --- | | CUSTOMER\_ID | CUSTOMER\_NAME | | 0001 | MARK HUGGS | | 0002 | KRISTI LAMP | | |  |  | | --- | --- | | CUSTOMER\_ID | ADDRESS\_ID | | 0001 | 0001 | | 0002 | 0001 | | |  |  | | --- | --- | | ADDRESS\_ID | ADDRESS | | 0001 | 1200, Harper Str | | 0002 | 585, Hampton Ave | |

Non-relational database, Document type:

## JSON FILE - CUSTOMER ##

{

"CUSTOMER\_ID": "0001",

" CUSTOMER\_NAME":

{

"FIRST\_NAME": " MARK",

"LAST\_NAME": " HUGGS"

},

"ADDRESS":

{

"STREET": "1200, Harper Str"

}

}

## JSON FILE – CUSTOMER2 ##

{

"CUSTOMER\_ID": "0002",

" CUSTOMER\_NAME":

{

"FIRST\_NAME": " KRISTI",

"LAST\_NAME": " LAMP"

},

"ADDRESS":

{

"STREET": "1200, Harper Str"

}

}

Therefore, to decide between a relational or non-relational data storage solution, you must evaluate the behavior of the application or the user that will use that database, the relationships between the entities and possible normalization processes.

**A transactional workload**

Relational and non-relational databases can be used as solutions for transactional workloads, which is the databases used to perform basic data storage operations, the CRUD (Create, Read, Update and Delete). Transactional operations must be done in sequence, with a transaction control that only terminates, a process called a COMMIT, when the entire operation was successfully executed. If this does not occur, the transaction is canceled, and all processes are not performed, thus generating a process called ROLLBACK.

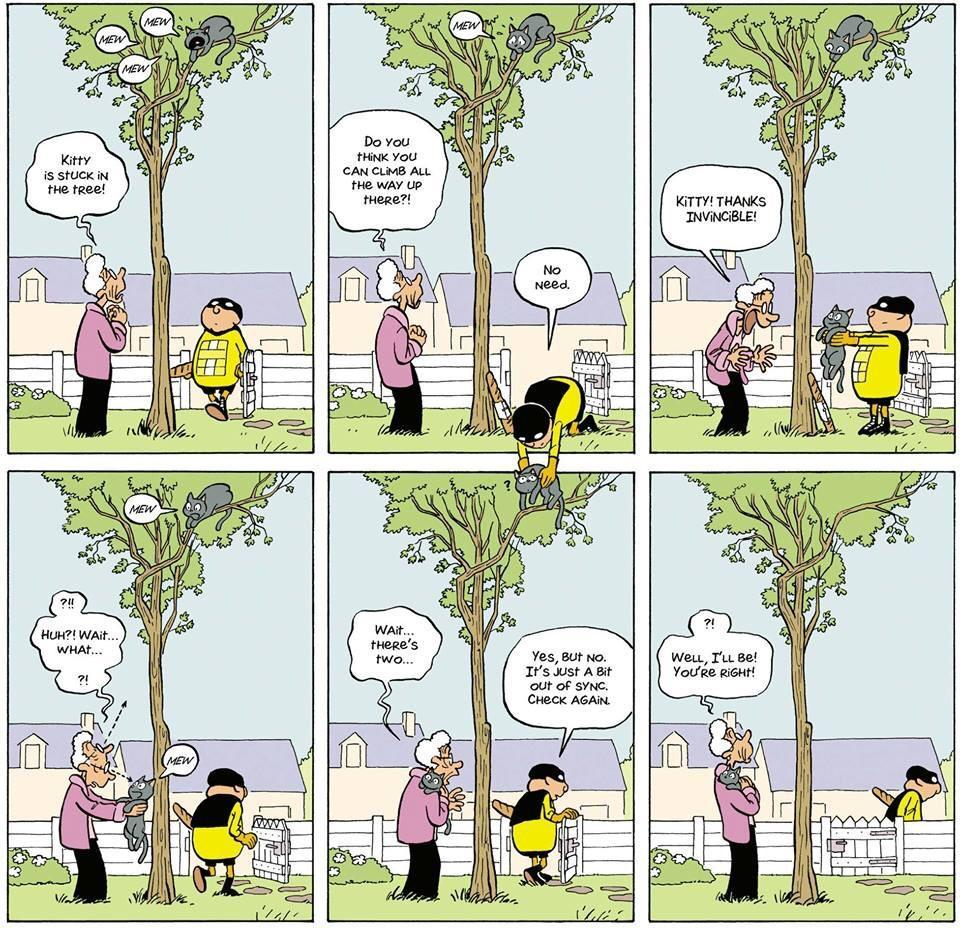
An important item to understand the difference between relational and non-relational databases is the ACID settings, present in most database technologies. These properties are:

* **Atomicity:** Is the property that controls the transaction and defines whether it was successfully performed completely to COMMIT or must be canceled by performing ROLLBACK. A database technology should ensure atomicity.
* **Consistency:** For a transaction running it is important to evaluate consistency between the database state before receiving the data, and the database state after receiving the data. For example, in a bank transfer, when an addition of funds occurs to an account, those funds must have a source. Therefore, it is important to know this source and whether the source funds exit process has already been performed before confirming the inclusion in this new account.
* **Isolation:** This property evaluates if multiple execution of transactions like this that is running keeps the database in the same state, then evaluates if the execution were sequential. In the bank transfer example, if multiple transactions are sent simultaneously, the amounts have already left the source, or you need to evaluate one by one.
* **Durability:** Is responsible for evaluating whether a transaction remains in the committed database even if there was a failure during the process, such as power outage or latency at the time of recording the record.

Database Management Systems (DBMS), which are database software, have ACID properties within their architecture and in addition to performing these controls they need to manage several complex situations. For example, multiple users or systems trying to access or modify database records, the database systems need to isolate transactions, perform all necessary validations quickly, and maintain consistency of data stored after transaction a commit. For this, some DBMS technologies work with temporary transaction locks, so that actions are done sequentially. This lock is done during the process of an action executing in that record, for example in an edition of a field in a table, the lock ends as soon as the COMMIT is executed confirming that transaction.

There are some DBMS that are called distributed databases. These databases have their architecture distributed in different storage and processing locations, which can be On Premises in the company’s datacenter or in a different datacenter around the world in the cloud. Distributed database solutions are widely used to maintain consistency in databases that will serve applications in different geographic locations, but this consistency doesn’t need to be synchronous. For example, a mobile game that can be played in the United States and Brazil, and the database of this game has some entities (categories, game modes, etc.) that must be shared among all players, while the player's online transaction data in the United States does not necessarily need to appear to the player in Brazil, this data will be synchronized, but not online (asynchronous).

All transactions in distributed databases take longer to process than undistributed databases, because it is necessary to replicate the data across all nodes in this distributed system. So, to maintaining an adequate replication speed, the distributed databases only synchronize the data that is needed, the concept of "eventual consistency", which configures ACID to perform replication between the distributed nodes asynchronously, after the confirmation of the transaction on the main node of the database was created. This technique can lead to temporary inconsistencies between database nodes. Ideally, the application connected to a distributed database does not require a guarantee of data ordering, it means that the data relating to this eventual consistency may appear to users with an eventual delay as well. Distributed databases are widely used by social media platforms, for news feed, likes, shares, among other features.



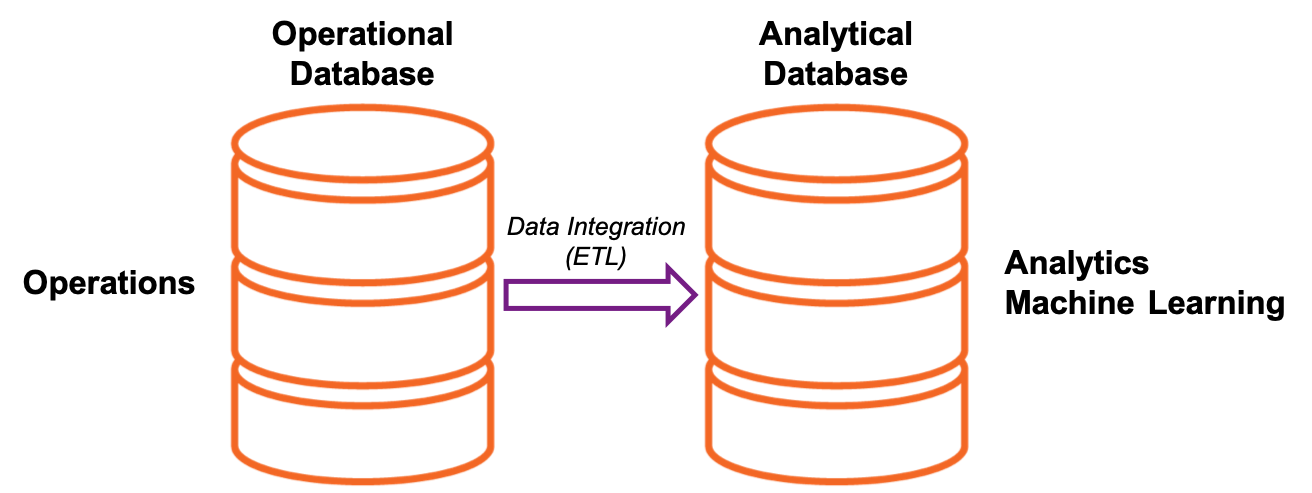
**An analytical workload**

The second category of data solution is the analytical workloads. These analytical solutions are based on high-volume data processing platforms, optimized for querying and exploring, and not for CRUD transactions or with ACID properties. In analytical databases we aggregate various data sources such as more than one transactional database, as well as logs, files, images, videos, and everything that can generate information for a business analyst.

These raw data are processed, aggregated, thus generating summaries, trends and predictions that can support the decision making.

An analytical workload can be based on a specific time or a sequence of dated events. In these workloads its common to evaluate only the data that is relevant to the analysis, for example if you have a sales system with a transactional database (source) with several tables recording all sales, products, categories, customers, among others, it is important to evaluate which of these tables can be used for the analytical database (destination) and then perform the data connections.

To create an analytical database, it is necessary to perform data ingestion, a process of copying the databases of the data to the analytical base. For this, a technique called Extract, Transform and Load (ETL) or the most modern Extract, Load and Transform (ELT) are used.



**Describe the difference between batch or streaming data ingestion**

Data ingestion is the process of copying data from data sources to analytical bases, but there are two different techniques for performing this copy, one batching data and one in online data streaming.

It is important to identify latency requirements between the time the data is generated in the source database, and the data availability in the analytical database.

**Understand batch load**

When batching load the data, the operation is offline. You must define the periodicity for creating the data batch load, collecting data in the data source, and then inserting it into the analytical database.

The periodicity can be hourly, daily, or even monthly, if the requirement of analysis of this data is met. Events that can trigger a batch load can be a new record on a table entity in the database, an action triggered by a user in an application, manual trigger, and more.

An example of batch processing might be the way we get votes count in elections. The votes are not counted one by one exactly after the vote of a voter, but they are inserted in lots that are being processed during election day until the completion of all charges and definition of the results.

**Positive of batch data load:**

* It is the most used method by companies that have multiple transactional systems with large volumes of data, because due to scheduling loads, it can be made at the most convenient time, such as outside business hours, when transactional servers are in lower demand.
* You can monitor the loads to verify where do you need to optimize an script or a method independently, so if you need to prioritize one specific load performance, you can manipulate your computing resources to prioritize that load.

**Constraints of batch data load:**

* There is a delay between the time of data generation on the transactional database and the availability of this data on the analytical database, which sometimes makes it impossible to follow up and make immediate decision-making of the numbers
* The full batch of data must be completed to then begin copying, and if there is any data unavailability, inconsistent data, network latency between transactional and analytical bases, among other situations, the batch load will fail

**Understand data streaming (online)**

In data streaming based data ingestion, there is an online connection between the data source and the analytical database, and the data is processed one by one, in events, right after its generation at this source. For example, for a sales tracking monitor solution, sales managers need to track sales data in "near real time” in a dashboard for immediate decision making. The sales transaction database is linked through streaming load to the analytical database that receives this data, processes it, and demonstrates in the Monitor Dashboard.

Another example could be a stock exchange and its real-time stock tracking panels. These dashboards receive processed information from purchase and sell transaction data for stock papers in a data stream.

Not always the load on data streaming is done online, it can also be done at intervals of 30 in 30 seconds, or even larger, but the behavior of this data is different from the batch load, as it is a continuous window of data ingestion between the source and the destination, while in the batch load each batch opens and closes the connection to process.

**Positive of data streaming:**

* The delay between data creation and analytical processing can be minimal.
* Latency between the source and the target in the order of seconds or milliseconds
* Analytical solutions can both demonstrate past data and perform trends that assist in immediate decision making while events are happening.

**Constraints of streaming data load:**

* Most transactional database technologies do not have a native streaming data export technology, so you need to implement this technique through manual control of what has already been ingested and what has not yet been ingested. This generates great complexity.
* The size of each event is usually small, to avoid having a very robust infrastructure to maintain this events queue during the streaming. This makes it impossible to ingest large files, videos, audios and photos, etc. These loads are often best implemented in batch loads.

In summary, we typically use batch data loads for the most structuring operations of the analytical base, the ingestion of the largest volumes of data and unstructured data. Data streaming is responsible for ingesting some specific data that will generate monitoring indicators such as aggregations and calculations of moving averages.

**Summary**

In this chapter, we reviewed the core data concepts about data storage and processing, the different data types, and data solutions. We went through the explanation of relational, non-relatable, transactional, and analytical data, their particularities and application cases.

In this module, you learned to:

* Identify how data is defined and stored
* Identify relational and non-relational data characteristics
* Describe and differentiate data workloads
* Describe and differentiate batch and streaming data

Now you already know how to differentiate a transactional database from application, and an analytical database. In the following chapters we will go into the details of each of these workloads and of the Azure services that are implemented for this. But before we detail these structures, in the next chapter we will understand the different roles and responsibilities in a Data domain.

2

Exploring different roles and responsibilities in Data domain

Neste módulo, você aprenderá sobre os diferentes cargos de trabalho associados à criação, ao gerenciamento e ao uso de bancos de dados. Aprenda sobre as principais responsabilidades dessas funções e as ferramentas que elas usam.

Na última década, a quantidade de dados gerados por sistemas e dispositivos aumentou consideravelmente. Devido a esse aumento, novas tecnologias, funções e abordagens de trabalho com os dados estão afetando os profissionais de dados. Os profissionais de dados normalmente desempenham diferentes funções ao gerenciar, usar e controlar dados. Neste módulo, você aprenderá sobre as várias funções nas quais as organizações costumam usar profissionais de dados, bem como as tarefas e responsabilidades associadas a esses cargos.

To start understanding how these technologies and solutions work, it is necessary to know the concepts of data storage and processing.

By the end of this chapter, you will be able to understand:  
• Explore job positions that works with data

• Explore common tasks and tools for data roles

**Understanding the core data terminologies**

To start, let's understand the terminologies used in the data world, so that all the next concepts are easily interpreted to be applied to technologies.

**What is data?**

A data is a “record”, or also called a “fact”, which can be a number, a description or observations used in decision making. Data only generate intelligence when processed, these are then called information or insights.

Data is classified into 3 basic formats, *structured*, *semi-structured* and *unstructured* data.

3

Working with Relational Data concepts

4

Working with Non-relational Data concepts

5

Data Analytics concepts

**1.**Como os dados são organizados em uma tabela relacional?

1. Linhas e colunas
2. Cabeçalho e rodapé
3. Páginas e parágrafos

**2.**Qual das alternativas a seguir é um exemplo de dados não estruturados?

1. Uma tabela de Funcionários com as colunas ID do Funcionário, Nome do Funcionário e Designação do Funcionário
2. Arquivos de áudio e vídeo
3. Uma tabela dentro do banco de dados do SQL Server

**3.**Qual das alternativas a seguir é um exemplo de conjunto de dados de streaming?

1. Dados de sensores e dispositivos
2. Dados de vendas do mês anterior
3. Lista de funcionários que trabalham para uma empresa

Make sure your source code is working, then paste it into your Word template and format it in Packt code style. Don’t type your code directly into the document editor, as capitalization and auto correct can break it.

We need to ensure that every code block and image is directly preceded by a “lead-in.” A lead-in sentence tells readers what the following code is or does. It’s important that they know this information before, not after seeing the code/image. You should also tell readers where the code should be placed, if it’s not already clear. For example, the file might need to be in a certain folder or be in the event handler for a particular button before it will work. Use SC – Source to add the code block format. If you need to highlight any piece of the code block, use SC – Highlight like it has been done for function (P – Code for code in text) in the code block below.

const set = **function**(...items) {  
 this.arr = [...items];  
 this.add = function(item) {  
 if( this.\_arr.includes(item) ) {  
 return false; (SC - Source)

Further explanation or discussion of how the code works can come immediately after your code block.

Given the practical nature of Packt lessons, you’ll often want your readers to run code, so your lead-ins will be written as instructions. These instructions should denote what the code will achieve, rather than simply state “run the following code.”

**Important note**

Avoid using comments in code. The best way to format code explanation or instruction is as regular body text. Comments tend to get lost in code blocks. They're not immediately distinguishable from the code, they can end up being poorly formatted, they may contain spelling mistakes that are difficult to correct, or they may be written in a shortened form that is difficult to interpret.

The closing paragraph or sentences of sections shouldn’t end abruptly end with code/a screenshot or simply ‘this completes the task.’ Readers require a summarizing sentence or two - a signpost. This signpost should reiterate what skills they now have, or mention how what they've just done/learned links to the next task or the overarching purpose of the chapter.

**Header 2 - Subtopic or subtask (H2 – Heading)**

Let’s (P – Regular) look at how to present commands in your chapter. Like with code, make sure the commands are working, then paste it into your Word template and format it in Packt command style. To format command lines, use P – Source. Make sure to add a lead-in telling readers exactly what we are doing with the command.

$ apt-get install node.js

$ apt-get uninstall node.js

Make sure to explain what the command will do. If readers are meant to take action based on the output, show the output as part of the command block above or use a screenshot and then discuss what they should look for.

**Exploring different roles and responsibilities in Data domain**

Use (P – Regular) numbered steps for sequential instructions. Remember, readers will often be practically implementing your lessons, looking between your content and their own computer screen. Numbered steps help them to keep their place.

1. Step one. (L – Numbers)
2. Step two.
3. Step three.

If you need to add paragraphs under specific numbered steps or if you need to use a block of code or commands, use L – Regular for paragraph text and L – Source for the code or commands. You can use these options to add paragraph text and code or commands under specific bullet points as well.

1. Step one. (L – Numbers)

This paragraph appears underst step 1.

1. Step two.

const set = function(...items) {  
 this.arr = [...items];  
 this.add = function(item) {  
 if( this.\_arr.includes(item) ) {  
 return false; (SC - Source)

1. Step three.

**Summary**

In your Summary section, reiterate the key lessons covered and skills learned in the chapter. Remind readers why these lessons or skills are useful. Finally, add a sentence or two on what we’ll be covering in the next chapter, highlighting how this is the next natural step from what we’ve just covered.

**Here’s a set of visual aids for other styles as well.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#2 This is how P – Bold style will look:

This chapter reviews the many ways **Wireshark** can filter traffic.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#3 This is how P – Italics style will look:

To help your learning of the different ways to *refine* your view.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#4 This is how P – Keyword style will look:

We'll cover when to filter **Traffic** and outline the difference between display and capture filters.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#5 This is how P – Code style will look:

So that you can refine your java.exe file when filtering traffic.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#6 This is how P – URL style will look:

We'll review ways to create more complex filters (https://www.google.com) by using the expression builder.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#7 This is how P – Source style will look:

apt-get install node.js

apt-get uninstall node.js

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#8 This is how P – Callout style will look:

We'll then go through capture filters and how they use syntax that is different than display filters. Finally, because filters are so handy, we'll cover some tricks, shortcuts, and common filters that will help you achieve a more effective analysis.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#9 This is how P – Callout Heading style will look:

**Important Note**

While in the course of your daily routine, the network starts to experience a significant slowdown. You check your Intrusion Detection System (IDS) and anti-malware protection, and there is no evidence of intrusion. At that point, you grab a quick capture to determine the source of the slowdown. Wireshark, along with many other packet analysis tools, has the ability to take a large capture, filter on specific traffic, and refine your view to help with analysis.

Always use P – Callout Heading and P – Callout together.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#10 This is how L – Bullets style will look:

* Filtering network traffic
* Comprehending display filters

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#11 This is how L – Numbers style will look:

1. Creating capture filters
2. Understanding the expression builder

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#12 This is how L – Regular style will look:

* Discovering shortcuts and handy filters

Filtering network traffic

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#13 This is how L – Source style will look:

* Wireshark has several options to filter traffic:

Display filters: Used during an active capture or on a pre-captured packet

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#14 This is how SC – Source style will look:

Capture filters: Applied prior to capture to only display a certain type of traffic

Expressions: Creates complex filters using logical operators

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#15 This is how SC – Highlight style will look:

When filtering **traffic**, there is a difference between display filters and **capture filters**. In the next section, let's **explore** the difference.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#16 This is how SC – Heading style (Filename above code snippet) will look:

**Cjava.py**

When working with packet captures, it appears as if the capture and display filters are the same. However, although the two work in similar ways, capture and display filters each use their own syntax.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

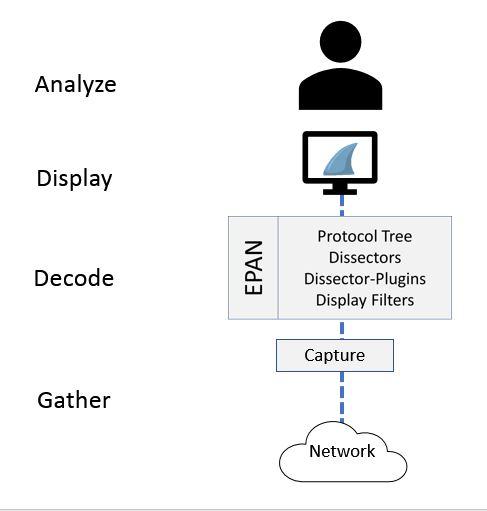
#17 This is how SC – Link style (GitHub Link below the code snippet) will look:

While using Wireshark, there are four main phases of packet analysis as discussed in [Chapter 2](https://cdp.packtpub.com/learn_wireshark___fundamentals_of_wireshark_/wp-admin/post.php?post=30&action=edit#post_25), Using Wireshark NG, which are Gather, Decode, Analyze, and Display, as shown in the following diagram:

**https://github.com**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#18 This is how IMG – Caption style (Figure and Figure caption) will look:

****

**Phases of packet analysis**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#19 This is how H1 – Section style (Heading 1) will look:

**Gathering network traffic**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#20 This is how H2 – Heading style (Heading 2) will look:

**The packets pass through the appropriate capture engine**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#21 This is how H3 – Subheading style (Heading 3) will look:

**NPcap or WinPcap**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#22 This is how H4 – Subheading style (Heading 4) will look:

**Capture filters**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#23 This is how P - Quote style will look:

*“use the Berkley packet filter syntax, and when used, Wireshark drops any packets that are not in the filter. You can read more about this in The BSD Packet Filter: A New Architecture for User-level Packet Capture”*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#24 This is how SP – Editorial style (Heading 1) will look:

**Please do not delete this paragraph.**