Selection of high quality data using a broker model

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Author: **Andrea Lottaroli**

Student ID: 968089
Advisor: Prof. Cinzia Cappiello
Co-advisors: Prof. Danilo Ardagna
Academic Year: 2021-2022

Abstract

Every day in the world a huge amount of data are generated and the necessity to manage and store them is increasing. This great quantity of data open the opportunity for companies and enterprises to exploit them to make profit. Indeed it has become very common that the business based their decisions, regarding the company management, on the knowledge obtained from studies and analysis over the data. The problem is that there are too many sources and it has become difficult for enterprises to select the most appropriate and suitable information. For this reason the figure of a mediator, or broker, is fundamental. The role of the broker is to suggest to a customer the best data sources that can satisfy the requirements imposed. The selection performed examines the quality and the price of the sources, and then, based on the constraints demanded, the broker suggest to the customer the best one to buy. The objective of the broker is to provide the best set of data sources in terms of quality, minimizing the price and the execution time. This thesis aims to describe a new types of broker model, based on previous works, where the sources of data are data streams and the data used come from the stock market. This version of the broker adds new metadata that describes the characteristics of each data source, provenance, similarities, providers, correlations, value distributions and dependencies are some of them. Some types of metadata describe the characteristic of a single data source, while others define the relationships between different sources. All these information, regarding each data stream, can help the broker to determine if a data source is eligible to be part of the final plan offered to the end-user. An optimization over the performances of previous models has been done. In fact a filtering process has been implemented, in particular it acts before the evaluation of the data quality dimension values and before the optimization phase. This procedure rejects all data sources that has a different provider, have a higher price, do not have in common any attribute and have a different granularity, or lower, with respect the requirements imposed by the customer. This method reduces the total execution time of previous broker model, preserving the same quality level of the plan discovered. All these changes are tested and analysed in order to show the actual improvements achieved. To summarise, the usage of the broker can reduce the complexity of the data streams selection, supporting the decision making process and giving the opportunity to save money and time to the companies.

Keywords: Broker, Optimization, Data sources, Information

Abstract in lingua italiana

Ogni giorno nel mondo vengono generate grandi quantità di dati e la necessità di gestirli e conservarli è sempre più cresente. Questo enorme volume di dati apre la possibilià alle aziende e alle imprese di sfruttare queste informazioni a scopo di lucro. Infatti è ormai abituale per le società basare le proprie decisioni, riguardanti la politica aziendale, sulle informazioni ricavate dallo studio e dalle analisi dei dati. La problematica riscontrata in questo campo risiede nella selezione dei dati, infatti, data la grande disponibilità di informazioni, è complesso scegliere le fonti che possono fornire dati utili alle aziende. Per questa ragione la figura di un mediatore, o broker, è fondamentale. Il ruolo del broker è quello di suggerire al cliente le migliori sorgenti di dati che possono soddisfare le richieste. La selezione esamina la qualità e il prezzo di ogni sorgente e, basandosi sui vincoli imposti, il broker consiglia le sorgenti da acquistare. La finalità è quella di trovare una serie di sorgenti che offrono le migliori informazioni in termini di qualità, minimizzando il prezzo e il tempo di ricerca. Questa tesi descrive un nuovo modello di broker, basandosi su lavori precedenti, utilizzando come sorgenti i data streams e considerando come caso di studio i dati ricavati dal mercato delle azioni. Questa versione di broker aggiunge nuovi metadata che descrivono le caratteristiche di ogni sorgente, provenienza, similarità, correlazione, fornitore, dipendenze e distribuzioni dei valori sono alcuni di questi. Ci sono due tipi di metadata: uno descrive le caratteristiche di una singola sorgente, l'altro definisce i rapporti e le somiglianze tra diverse sorgenti. Questa conoscenza, riguardo ogni data stream, può aiutare a determinare se una fonte di dati può fare parte della soluzione finale. Una nuova funzionalità è stata aggiunta al modello di broker, con l'intento di migliorare le prestazioni. Un processo di filtraggio delle sorgenti è stato implementato. Questa procedura scarta tutte le fonti che hanno un fornitore diverso, hanno un prezzo superiore, non hanno in comune nessun attributo e hanno un granularità diverso o inferiore rispetto ai vincolo definiti dal cliente. Grazie a questa nuova funzionalità il tempo di ricerca ed esecuzione è diminuito, mantenendo lo stesso livello di qualità della soluzione trovata. Tutti questi cambiamenti sono stati testati e analizzati in modo tale da mostrare gli effettivi miglioramenti apportati al broker. Riassumendo, l'utilizzo del broker può ridurre sensibilmente la complessità nella scelta delle sorgenti dei dati, facendo risparmiare alle aziende tempo e denaro.

Parole chiave: Sorgenti, Ottimizzazione, Broker, Dati

Contents

1 | Introduction

Nowadays a massive amount of data are collected and continuously generated in every possible domain. All this knowledge provides a great opportunity to generate profit. For this reason companies and enterprises are interested in managing and exploiting the information available, making business decision based on the data gathered. Along with the quantity, it is essential to have also high quality data, otherwise the decision making process of the companies would conduct to a loss of earnings. This means that the selection of the data is crucial to avoid wasting of time, in performing analysis over low quality data, money and resources. A prior examination over the data is necessary and it can be conducted by the mediator. A mediator is a figure that works between the data providers and customers interested to buy data. The broker, after receiving a detailed description of the characteristics that the data must have, from the end-user, filters out all data sources that do not respect the constraints imposed, regarding the quality, price or provider, and it offers to the user the best set of sources to buy that can satisfy the requirements. Another feature that the broker must respect is a low execution time. In fact for companies is essential to take strategic decisions as soon as possible based on reliable information.

1.1. Contents of the work

This thesis will discuss a new version of the broker model, based on the older version described in [3] and [15]. The technologies and tools used will be discussed and the full procedure will be analysed along with some analysis of the experiments done. This broker will exploit the advantages of previous works adding new features to improve the performance. The source of the data will be the data streams and the case study will use data gathered from the stock market, as in [3]. The objective remains always the same, trying to provide the best set of data streams considering, also, the execution time.

1.2. Structure of the document

The structure of the document is reported as follows:

- Chapter 2 describes the tools and technologies used to support the broker.
- Chapter 3 defines the case study considered.
- Chapter 4 presents all the steps that the broker follows to suggest the data streams to buy.
- Chapter 5 illustrates the structure of the data sources and data lake.
- Chapter 6 depicts the results of the experiments conducted.
- Chapter 7 summaries the work done and discusses possible future works.

2 | State of art

Nowadays for enterprises and companies is necessary a structured method to organize, manage and store the massive amount of data that is increasing everyday. For this reason data lake has been used to satisfy this demand, but unfortunately it is not enough to guarantee to the companies a relevant profit, it is needed a tool that exploit all information stored, improving the decision making process. The broker is the solution of this problem, in facts it is an algorithm that choose the best collection of data sources, belonging to a data lake, that satisfy the requests of the customer. Another important issue for the broker algorithm is to evaluate the quality of each data source to be able to choose the most complete and accurate sources, so it important to select a suitable set of data quality dimension to analyze each set of data.

In the following sections an overview of the current technologies and tools is reported, previous broker models are discussed, showing the advantage and disadvantage of each implementation, the main data quality dimension are defined, and the functionality of data lakes are illustrated.

2.1. Data lake

Data lakes are used to manage and organize data received from different sources, but not only, in fact in order to support the broker algorithm in a significant way, data lakes have to provide some other functionalities. In this section data lake functionalities are discussed, showing all feature they can offer. The logical structure of a data lake is shown in Figure 2.1.

2.1.1. Data lake description

A data lake, as described in [26] and [34], is a massive collection of datasets that may be hosted in different storage system, ingest heterogeneously structured or unstructured raw data, stores data in their native format without a strict schema and they can be accompanied by a set of metadata. According to [44], data lakes allow to perform more

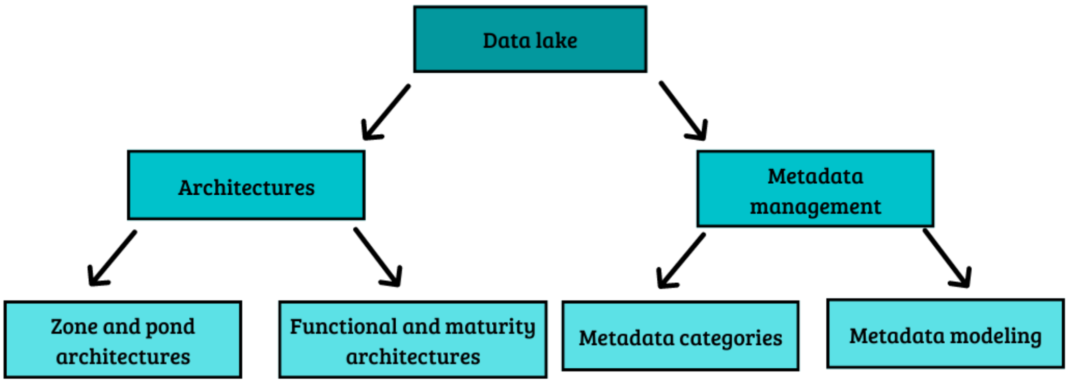


Figure 2.1: Data lake composition

efficient and efficacious data analysis by crossing multiple existing data sources, process and analyses. Furthermore data lakes decouple data producers from data consumers, a very useful feature especially when the operational system are not owned by the users. So a data lake can be an essential resource for companies to store and preserve the value of the data, however a data lake can easily turn into a data swamp. A data swamp is an unmanaged data lake without a proper organization of data and it does not provide the possibility to perform any type of analysis, wasting the opportunity to exploit the value of the data.

The characteristics needed to correctly handle a data lake, in accordance with the work done by [38] include: 1) a metadata catalog that enforce data quality, data governance policies and tools, 2) accessibility to various kinds of users, 3) integration of any type of data, 4) a logical and physical organization, and 5) scalability. Metadata, that will be described in depth, conduct a very important role in data lakes because they provide summary information regarding each data source present in the lake and information regarding the relationships between datasets, improving the selection phase of the data and obtaining better results when the broker algorithm is used to accomplish the request of the user to extract data.

2.1.2. Data lake architectures

Data lake architectures are classified in pond architecture and zone architecture [17], however this categorization may seem to be fuzzy, as said in [38], so a new approach, functional and maturity architecture, is also presented.

2.1.3. Pond and zone architectures

An example of pond architecture is described by Inmod. Inmod [23] designs a data lake as a collection of data ponds, where a data pond is a portion of data lake handling a specific type of data. There are in total five data ponds. The raw data pond contains the data that access the lake and it has a function of a holding cell, in fact analysis on data are not performed in this phase. Then when an analysis is requested and based on the type of data required, raw data pond sends data to other ponds. The analog data pond stores analog data and performs data reduction in order to obtain a useful and manageable volume of data in the pond. In the application data pond are stored data coming from one or more applications. Given the fact that data comes from application, here important information regarding the enterprises are contained, but at the same time is difficult for the analysts to operate due to the fact that data, coming from different sources, are not integrated. The textual data pond includes textual data and before to execute any analysis of the data is essential to disambiguate the text. Then when data are used and no more suitable for analysis, are relocated to the archival data pond.

Adopting zone architecture, data are assigned to a zone according to their degree for refinement [17], one example is described in [34], in which a four zone architecture is adopted. In the raw data zone all types of data are ingested, batch, real time or hybrid, without any kind of processing and then stored in their native format. Users in the process zone can process data, performing operations like union, selection, according to the requirements. The access zone allows the access to the information stored and data are used to perform some statistical analysis, reporting and business intelligence analysis. At last the governance zone implement a monitoring function all over the architecture guaranteeing data quality, data life-cycle, data access and metadata management.

2.1.4. Functional, hybrid and maturity based architecture

In the architectures previously described, when data pre-processing is performed, the analyses are quick and easy, but this cost a data loss in the pond architectures, since part of raw data are lost when transferred to other ponds. Moreover, in the zone architecture the data that flow across the areas may lead to multiple copies of data, producing difficulties in tracking the movement of the data. Furthermore the distinction of data lake architecture into pond and zone approach is not so clear, in fact pond architecture may be considered as an alternative to zone architecture, since data location depends on the refinement level of data, as in zone architectures [37]. Another approach adopted, in order to overcome the drawbacks of previous architectures, is to organize data lake architectures keeping in consideration the criteria used to define components in the data lake [37].

Functional architectures adopts basic functions to define a lake's components, this include: a data ingestion function to connect with data sources, a data storage function to keep raw and refined data, a data processing function and a data access function to allow raw and refined data querying. This method have the good characteristic to clearly highlighting the functions to implement for each data lake, helping the match with the required technologies.

Data maturity based architectures are data lake architecture where components are defined considering the data refinement level, it is constituted of zone architectures as transient zones, raw data zone, trusted zone and refined data zone. This approach, similar to the zone and pond architecture, is useful to plan and organize all stage that data goes through, from generation to the deletion or to its archival.

At last hybrid architectures are data lake architectures where the components rely on both data lake functions and data refinement. So, hybrid architectures merge the two main positive features of functional and data maturity architecture to design a data lake.

2.2. Metadata

Metadata assume a crucial role in data lakes, they avoid the turning of the lake in a collection of inoperable data, a data swamp. In [13], metadata are described as a type of data which provide information regarding other data, processes and systems, while metadata management represents all activities which involve managing an organizations' knowledge on its data. Figure 2.2 depicts a metadata classification.

2.2.1. Usage of metadata

Sawadogo [38] identifies six features that a data lake metadata system should implement to be considered comprehensive. Semantic enrichment involves adding information, such as title, tags and description, making the data meaningful by providing informative summaries. Data indexing organizes a data structure to retrieve information based on keywords or patterns. Link generation and conservation is the process of detecting similarity relationships or integrating preexisting links between datasets. Data polymorphism is the simultaneous management of several data representations in the lake, allowing to store and reuse transformed data, making analyses easier and faster by avoiding the repetition of certain processes. Data versioning expresses a metadata system's ability to manage the update operations, while conserving previous data states and ensures process

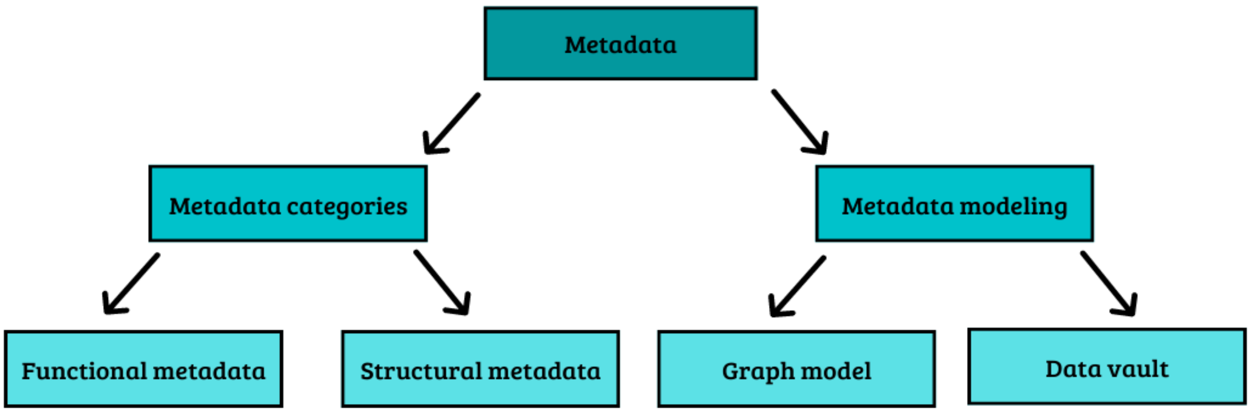


Figure 2.2: Metadata management

reproducibility of analyses, detection and correction of inconsistencies. Usage tracking consists in managing information about user interactions with the lake, where interaction are usually creation, read and update operations.

2.2.2. Metadata categories

In [35], are identified two classification of metadata, functional metadata and structural metadata.

2.2.3. Functional metadata

Functional metadata, illustrated in Figure 2.3 have another differentiation, with respect to the way they are gathered. Business metadata includes business rules, making the data more understandable reporting data field names and integrity constraints. Operational metadata includes information automatically generated during data processing, as for example data quality, data provenance and executed jobs. Technical metadata concerns information regarding data type, format and structure. In [12] is shown that operational and technical metadata sometimes intersect. For instance, data fields relate both to business and technical metadata, since they are defined in data schemas by business users. Similarly, data formats may be considered as both technical and operational metadata, and so on. So functional metadata result quite fuzzy and, even if, this classification is the most cited the structural approach is preferred and it is described as an extension of the functional metadata.

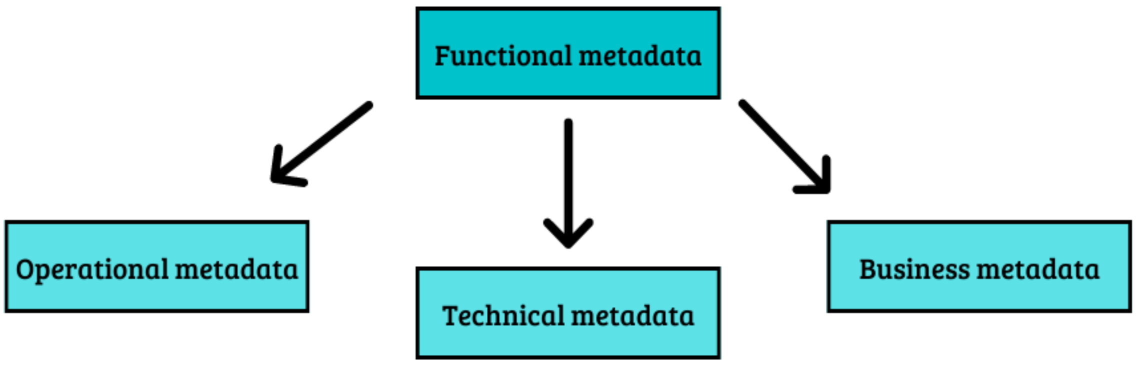


Figure 2.3: Fuunctional metadata

2.2.4. Structural metadata

Structural metadata, shown in Figure 2.4, categorizes metadata with respect to the "objects", which can be seen as a relational or spreadsheet table or a simple document, like an image file or textual document, in a semi-structured or unstructured data context. This classification include intra-object, inter-object and global metadata.

Intra-metadata, as described in [35], allow users to understand the characteristic of each objects, specifying quality, meaning and security level. Intra-metadata are separated in many categories. Data characteristics provides general information of a dataset, this includes name, size, structural type and creation timestamp. Definitional metadata explains the meaning of the dataset. Definitional metadata are classified as semantic and schematic metadata. Navigational metadata gives information regarding the location of the data, this includes file path and database connection urls. Then lineage shows the history of each dataset, from the origin to the last update. In this way the reliability of the datasets increases. At last access, quality and security metadata track each interaction between users and data, ensure consistency and completeness of each dataset and support access verification, improving the safety of sensitive data.

Inter-object metadata represent links between two or more objects and they are distinguished in different types. Dataset containment says if a dataset is contained in other dataset. Partial overlap specifies if some attributes are shared between different datasets. Provenance specifies which dataset is the source of other data. Logical clusters gathers the data that belong to the same domain. The last one, content similarity measures how many attributes are shared between two dataset [35].

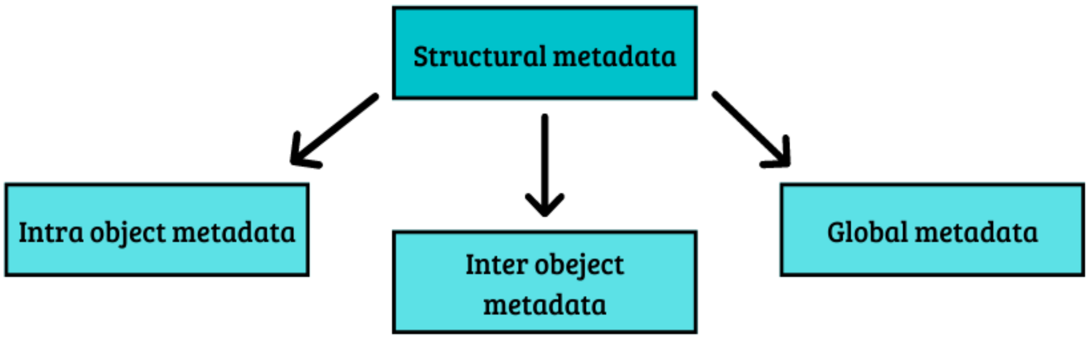


Figure 2.4: Structural metadata

Global metadata, as shown in [37], are data structure not directly associates with each object but makes the data processing and analysis easier. Semantic resource, indexes and logs are the three types of global metadata. Semantic resources are knowledge bases, a technology used to store complex structured and unstructured information, improving the data analyses. Indexes are built and enriched by an indexing system and retrieve data based on terms or patterns. Logs take note of all interaction, like connection and disconnection, between each user and the data lake.

2.2.5. Metadata modeling

In the literature there are two ways to represent metadata systems, the first one uses a graph view and the second one uses data vault modeling.

2.2.6. Graph models

In this section the characteristics of a metadata model are described through graph models, as shown in Figure 2.5. There are three categories: similarity centered graph, composition centered graph and data provenance graph.

The similarity centred graph model is an undirected graph composed by arcs and nodes. Nodes represent the data object while the arcs represent the similarity between two different data objects. Arcs can also have a weight that specify the affinity between two connected data object. This structure allows to determine the importance of each node for the graph and allows schema discovery, the determination of the collection of appro-

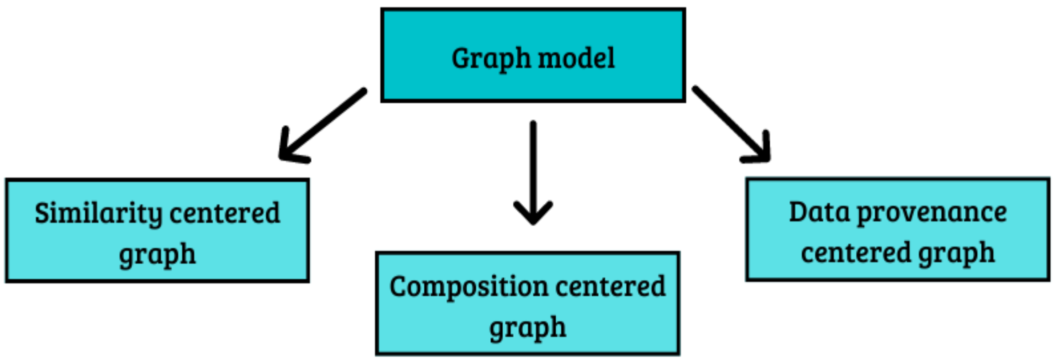

Graph model
Similarity centered graph
Composition centered graph
Data provenance centered graph

Figure 2.5: Graph models

priate nodes to solve an analysis problem, as discussed in [14].

The composition centered graph model builds a hierarchy of data objects, through a directed acyclic graph in which nodes serve as objects or attributes and arcs explicit the constraint $B \subseteq$A, all types of attribute of object B belong also to object A. This hierarchical structure allows the user to have an easier navigation through objects and facilitates the research of data [27].

The data provenance centered graph model represents a directed acyclic graph, in which nodes represent an entity, like user, role or object and arcs describe the interactions between entities, specifying the action performed, like read, updated, deleted or queried. For each interaction, timestamp and a location are annotated [7].

2.2.7. Data vault

A data vault is a conceptual and logical data model. Data vault modeling shows entities and relationships between entities, where entities can be hubs, links or satellites. A hub represents business objects, it contains the business key of the business object it represents, a unique surrogate key hashed from the business key, a load date and a record source. Essentially hubs represent a customer, a vendor, or a product in a business decision system. Links represent associations and relationships between hubs. Links refer to the connected hubs through their surrogate keys. This structure adds flexibility to the model, in fact new entities can be added to the system without changing existing hubs, instead a link is updated or added. Of course different types of links exist and depend on the hubs that connect. Satellites add information to hubs and links. One hub or link may have multiple satellites holding all attributes that are not covered by the hubs and links, while each satellite is attached to a unique hub or link, holding the surrogate key of one of them [18]. In [30], it is illustrated a metadata model using data vaults. To identify hubs, metadata like title, location, date or category are used, because they are more susceptible to constitute keys. For each attribute a satellite, containing hub's attributes, is associated, meanwhile links allow to associate all hubs. An example of data vault model, reported in Image 2.6, shows that metadata common to all objects, like title, date and category, are stored in hubs, and metadata specific to some object, like publisher for books, are stored in satellites.

2.3. Data quality

Until now a description of the management of data and of its technologies are discussed, in fact data lake provides a secure storage where to keep data, then it is up to the broker algorithm to choose suitable data to satisfy the request from a potential customer. The issue here is how a broker should choose a data source rather than another one. For this reason data quality is introduced with the purpose of evaluating, through some metrics, the datasets of each data source and choose the best providers. The metrics used to discern among data sources are data quality dimensions. Data quality dimensions measures the characteristic of datasets, for example they say if data are updated, if data present some kind of errors or if data are reliable. Data quality is crucial for enterprises because decisions or statistical analyses characterized by poor data quality probably will conduct to the wrong actions or conclusions. So, the broker has the demanding role to choose the datasets based on the values of data quality dimensions and satisfying the minimum quality request of the customer. For companies, it is not easy to have high quality data because today the amount of information available is huge, and it is increasing everyday. Another issue to address is how the enterprises should choose the data quality dimension to evaluate data, whose dimensions can assure that data are reliable and useful. All these problems are discussed in the following sections.

2.3.1. Challenges of data quality

In order to describe the challenges to retrieve high quality data, it is important to introduce the concept of big data. Big data [24] is defined as a large amount of data which requires new technologies and architecture to make possible extracting valuable data. In fact due to the great size of data is very difficult to perform an adequate analysis. Big data are characterized by the four V, velocity, volume, variety and veracity. The velocity

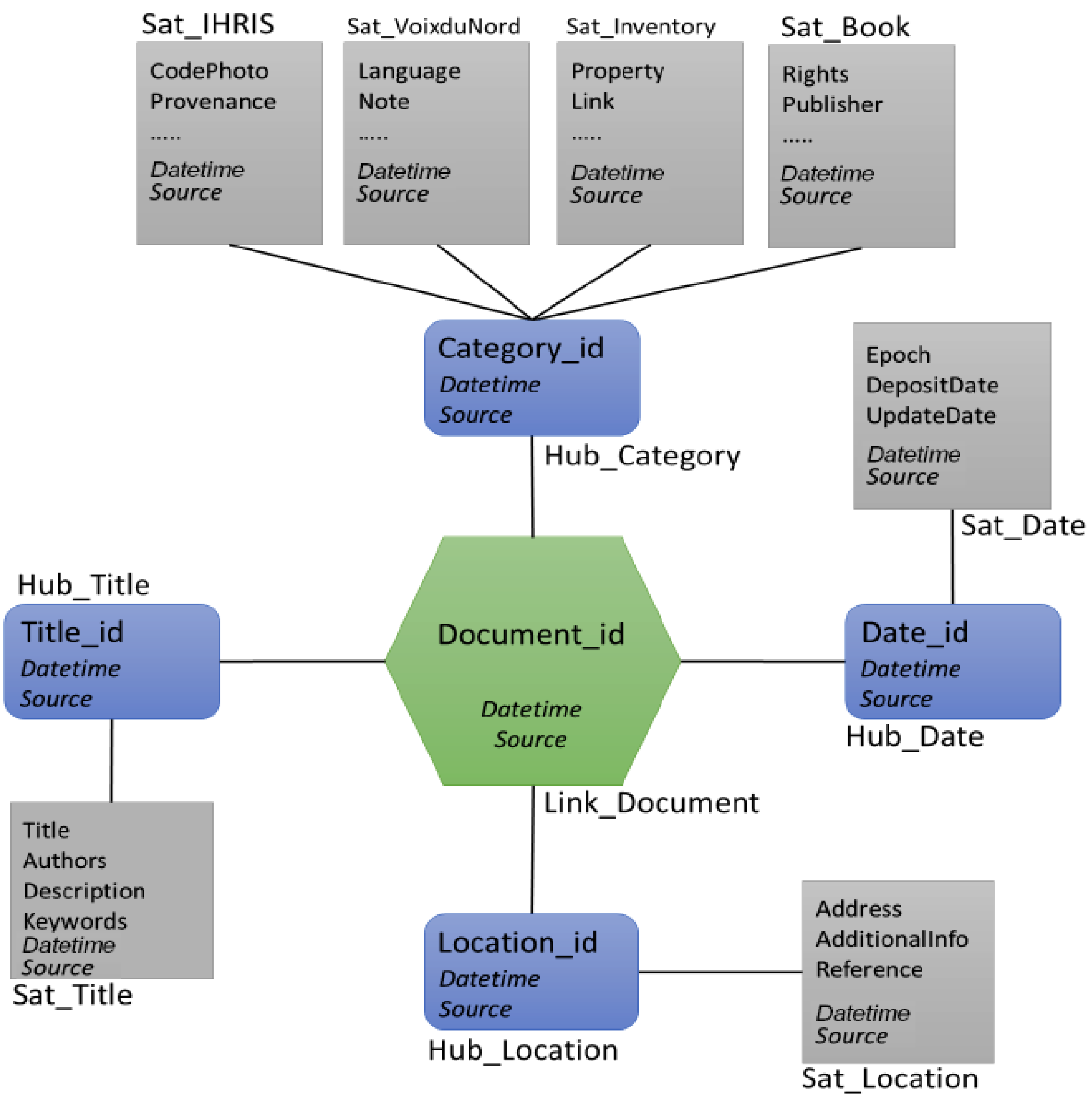


Figure 2.6: Data vault model taken form [30]

stands for the speed in which data comes from various sources, hence enterprises have to deal in a timely manner. The volume refers to the exceptional volume of data, it is estimated that now the data existing are in the order of petabytes and is supposed to increase to zettabytes in the near future, so it is clear that is difficult to handle this amount of data with traditional systems. Variety represents the fact that the data produced do not belong to a single category but include all kinds of data types and they are produced by different sources. Veracity says how truthful data collected are.

The issues of data quality in big data is present also in the broker algorithm, in fact some problem are similar, but of course the broker deals with a tiny amount of data with respect than the volume of big data. Indeed [10] the diversity of data sources, in the data lake, involves disparate data types and complex data structures, increasing the difficulties for the broker to integrate and analyse all kinds of data. For the broker is also difficult to deal with a massive quantity of data and at the same time to judge the values of data quality dimension for each dataset in a reasonable amount of time. Another main aspect of data quality is timeliness, data change very fast and can happen that information become quickly outdated and performing analysis on these types of data will produce useless and inaccurate results leading to poor decision making.

2.3.2. Classification of data quality dimensions

Often data quality is associated with the concept of "fitness to use", the capacity of the data to fulfill the request of the data consumer. Along with this definition is attached the notion of data quality dimensions, a set of attributes that specify every characteristic of the data. One example of dimension could be the accuracy, that measure the reliability and correctness of data, or usefulness, that represents the advantage the user can gain from the use of the information. The evaluation of the data quality and the decision of which dimensions to consider can follow different approaches. In [41] three methodology to study data quality are identified: intuitive, theoretical and empirical approach. The intuitive approach selects the data quality attribute exploiting the researchers' studies or through an intuition on what attributes can be considered appropriate. It is also the most used approach. The empirical approach analyses the features that data consumers consider to verify if a collection of data is useful for their tasks. In this case, scientific researches are not considered and the advantage is that this method embraces the voice of the customers. This approach can reveal characteristics that researchers have not considered, but the correctness and completeness of the results, obtained by the data consumers, cannot be proven. The theoretical approach selects the attribute, to consider for evaluating data quality, based on data deficiency. Data deficiency is an inconsistency between the view of the real-world system that can be inferred from a representing information system and the view that can be obtained by directly observing the real-world system [40]. The three representation error are incompleteness, in which a real world state is not represented in the information system, ambiguity, where two or more real world state are represented by the same information system, and insignificance, when an information system represent a state that is non existent in the real world.

In the literature a classification of the data quality dimension is also proposed, for each categories is chosen a label that captures the essence of the class. In [29], are identified four set: content related criteria, technical criteria, intellectual criteria and instantiation related criteria. Content related criteria concern the information that is retrieved, like accuracy and relevancy, technical criteria measures aspects that are determined by software and hardware, like response time and reliability, intellectual criteria are made up of very subjective criteria like believability and reputation and, the last one, instantiation related criteria concern the presentation of the information and present dimensions as amount of data and understandability. Another classification is shown in [41] and presents four categories: intrinsic, contextual, representational and accessibility dimensions. Contextual criteria highlight the requirement that data quality needed considering the task at hand, representational and accessibility emphasize the role of the system, that own the data, and ensure the accessibility and security of the system. The system must also assure the understandability, interpretability, consistent and concise representation of data. Intrinsic criteria denote that data have quality in their own right.

2.3.3. Data quality dimensions description

As described there are lots of way to classify data quality dimensions, other example are illustrated in [40] and [28]. But the most considered data quality dimensions that summarize the most important characteristic of the datasets are: accuracy, consistency, timeliness and completeness.

2.3.4. Accuracy

There are different definition for accuracy, in [41] it is defined as the degree in which data are correct, reliable and free of error, in [36] accuracy is separated in syntactic and semantic accuracy. Syntactic accuracy measures the closeness of a value v to the components of the corresponding domain. Semantic accuracy is defined as the closeness between a value v and a value v', where v' is considered the correct representation of a real world object. In general, the accuracy is measured as:

$$\text{Accuracy} = 1 - \frac{\text{Number of incorrect data values}}{\text{Total number of data}}$$

2.3.5. Timeliness

The timeliness measures if the data are updated and suitable for the task at hand [41]. In [24], timeliness is expressed as age of data, in particular it is defined as the difference between the current system time and the timestamp of the data. The timeliness is calculated as:

$$\text{Timeliness} = \max\left(1 - \frac{\text{Currency}}{\text{Volatility}}; 0\right)^{s}$$

Where currency [39] specifies how recent are data and is calculated as:

$$\text{Currency} = \text{Age} + (\text{DeliveryTime} - \text{InputTime})$$

Delivery time represents when the information product is delivered to the customer and input time is when the data are obtained. Volatility specifies the duration of the data validity. Then s is a parameter that influence the sensitivity of measurement, the greater s the bigger is the reduction of the timeliness, considering the same ratio between currency and volatility.

2.3.6. Completeness

Completeness in [41] is defined as the extent in which data are of sufficient breadth, depth and scope for the task to fulfill. While in [32] it is described as the degree to which entities and attributes are not missing from the schema. At the data level, for example, one can define column completeness as a function of the missing values in a column of a table. So, completeness calculates the number of missing values that a representation of a real world object can have and it is measured as:

$$\text{Completeness} = 1 - \frac{\text{Number of missing values}}{\text{Number of total values}}$$

2.3.7. Consistency

The last main quality dimension is consistency and measures [41] whether data are always displayed in the same format and are compatible with previous data. Consistency, also, captures the violation of semantic rules defined over a set of data items, where items can be tuple of relational tables or records in a file. Semantic rules are subdivided in: integrity constraint and data edits. Integrity constraint specifies the bonds that a single attribute or attributes from different relation must satisfy. The role of data edits is to detecting inconsistencies by formulating rules that must be respected by every correct set of answers. These rules are defined as edits and denote error conditions. In general, consistency is calculated as the ratio of correct values with respect to integrity and business rules:

$$\text{Consistency} = 1 - \frac{\text{Number of violations}}{\text{Total number of consistency checks performed}}$$

2.4. Data stream and data window

Data stream have emerged as a modern archetype for handling time-varying, volatile, unpredictable and unbounded information through different monitoring applications. Recently, data streams, have been used instead of conventional database management system, due to their different functionalities. Examples of data streams are information generated in the field of finance and in traffic measurements. An essential tool to retrieve knowledge and perform queries from the data stream are windows. In this section the data streams and the different types of windows are described.

2.4.1. Data stream

A data stream S is a mapping $S : T \longrightarrow 2^R$ that at each instant $\tau \in T$ returns a finite subset from the set R of tuples with common schema E. A supplementary attribute A_τ is used as the timestamp of tuples and takes its ever-increasing values form T [31]. In fact, it is commonplace represent data streams as relational tuples. A tuple is represented as a set of elements e_i and each element define an attribute. Every tuple is an instance of the schema and it is described by the values of his attributes. For each tuple is associated a timestamp value use as an unique time index and to provide order among the data items flowing in the system. All the tuples are linked to one timestamp value and different tuples can share the same value. In order to deal with data streams instead of exploiting one-time queries, continuous queries are preferred. In fact there are some differences between traditional database management system and data streams, as reported in [31]. The size of the stream can be unbounded, the state of the data is not known in advance and the replies depend on the set of stream tuples available during the query evaluation. Another common problem in data streaming is the resources limitation that make impossible to store the entire amount of elements accumulated, since streaming data are usually kept in memory. To face these issues the data windows are introduced in the query formulation.

2.4.2. Data window

Dealing with continuous queries required the concept of the sliding windows. A sliding window over a stream is defined as a window that at any point of time contains a historical snapshot of a finite portion of the stream. So, each window is applied over the items of a single data stream and returns a finite, but always different, set of tuples. As reported in [4], there are three classes of sliding window operators in continuous query language: time-based, tuple-based and partitioned.

Time-based sliding window is probably the most common, this type of window requires a time interval as a parameter because the size of the window and the tuples returned belong to the period of time provided as input.

A tuple-based window starts from the current time instant and goes backwards in time, obtaining all the encounter tuples until the maximum number of items, provided as parameter, is reached.

A partitioned sliding window takes as parameter a positive integer and a subset of the attribute of the stream. This window splits the data stream into different streams based on the equality of attributes provided. Then it computes a tuple-based sliding window, with the size defined by the parameter, on each generated streams and, at last, takes the union of all windows to generate the final set of window tuples.

2.5. Optimization phase

Up to now have been discussed the technologies and tools that store, evaluate and keep in shape the data, but we have not yet explained how the actual selection of data is performed. An imperative issue has to be discussed: how broker can analyse, choose and select the set of the best data sources in order to satisfy a user query. Another issue is that the research of the optimal plan is a NP-hard problem, as proved in [5]. In order to have a good solution in a reasonable amount of time, instead of the best solution characterizes with very high computational time, heuristic and metaheuristic algorithm have been adopted. A heuristic is a strategy that ignores part of the information, with the goal of making decisions more quickly, frugally, and accurately than more complex methods, but does not guarantee an optimal solution [19]. A metaheuristic is a searching algorithm able to approach and solve complex optimization problem using a set of several general heuristics. This method does not need knowledge regarding the optimization problem to be solved, in fact most of metaheuristic algorithm are only approximate algorithms because they cannot always provide the global optimal solution [16]. In previous works [3] and [15] three types of algorithm have been used: greedy algorithm, local search and tabù search. Hence, in the following sections greedy, local search and tabù search algorithm are analysed.

2.5.1. Greedy algorithm

The greedy algorithm is a simple approach that takes decision based on the currently available information. The solution found using this algorithm is not the global nor local optimal, it an initial solution used by successive algorithm as starting point to find an admissible solution. In the context of data lake and broker model the greedy algorithm can be useful to perform some initial selection and removal of data sources, based on the constraint imposed by a request, in order to provide an initial solution to be improved with the application of local search and tabù search algorithm.

2.5.2. Local search

Local search is a heuristic search algorithm and, starting from an initial solution, proceeds by applying operations on the current solution to obtain an admissible solution, that respect the level of quality and price requested. The set of solutions that can be generated with a given operator on a solution s is called the neighborhood of s. Local search, from the neighborhood of the current solution try to find one feasible solution that improves the value of the objective function. This mechanism is iterated until there are no improving solutions in the neighborhood of the current solution [2]. At the point in which there are not any further improvement, the current solution is called local optimum. It is important to highlight that the local optimum does not always correspond to the global optimum, hence a better solution can exists and the local search risks to skip it. This method needs an initial solution, provided by the greedy algorithm and meticulous definition of neighborhood. Thus, local search can provide a better solution to the customer in terms of quality and price.

2.5.3. Tabù search

Tabù Search, also called Adaptive Memory Programming, is a metaheuristic algorithm for optimization problems. The objective is to find the best decisions or actions in order to maximize some measure of merit, in this case data quality, or to minimize some measure of demerit, in this case the price. Tabù search can find better solutions, considering the overall quality, with respect than local search, adopting a special strategy designed to exploit adaptive memory [20]. This algorithm implements a queue, a tabù list, in which are stored the last action performed in order to avoid to repeat them and discouraging the search from coming back to previously visited solutions. Of course after a specific number of operations, the actions that are in the queue for the longest period are deleted and then can be executed again. The problem is to choose the size of the queue, how many action should be performed before eliminating the actions in the queue, too many action may result in problems related to the computational time and few moves can preclude the discovery of the global optimum. Another unique feature, that help to find the global optimum, of tabù search is that the algorithm can choose to take an action that conduct to a worse solution, avoiding to get stuck in a local optimum.

2.6. Broker model

Modern business strategies are focused on the understanding of the customer preferences and they aim to model product and services following the demands of the customers. Accordingly, it is necessary a way to collect and analyse the client data. A solution to this problem can be the introduction of a mediator between the interests of the customer and the data provider, the broker. In particular, a broker is a mediator algorithm between the data streams belonged to data sources and the users, its purpose is to satisfy the requests of the customers in terms of quality and price. Essentially a broker takes in input the request of the user, a query, that includes the quality of the data, expresses through data quality dimensions, a maximum price and a list of attributes that data must have. The broker examines each data sources present in the data lake and depending on the constraints of the query and exploiting algorithm like greedy search, local search and tabù search, it offers to the customer a plan to buy, a list of data sources that respect the constraints. An example of what a broker does is depicted in Figure 2.7. Several broker model has been developed during the years with different features and optimization mechanisms. In the following section a brief summary of the previous broker models is presented.

2.6.1. Previous broker models

There are three typologies of broker models. The first model [5] uses as data quality dimension timeliness, accuracy and completeness and it assumes that the quality of the data is normally distributed for each fragment. It focused on minimizing the price of the plan as shown by its optimization method, illustrated in Figure 2.8, where x_i is set to 1 if a specific data source is selected, and it is 0 otherwise. This approach is characterize by a

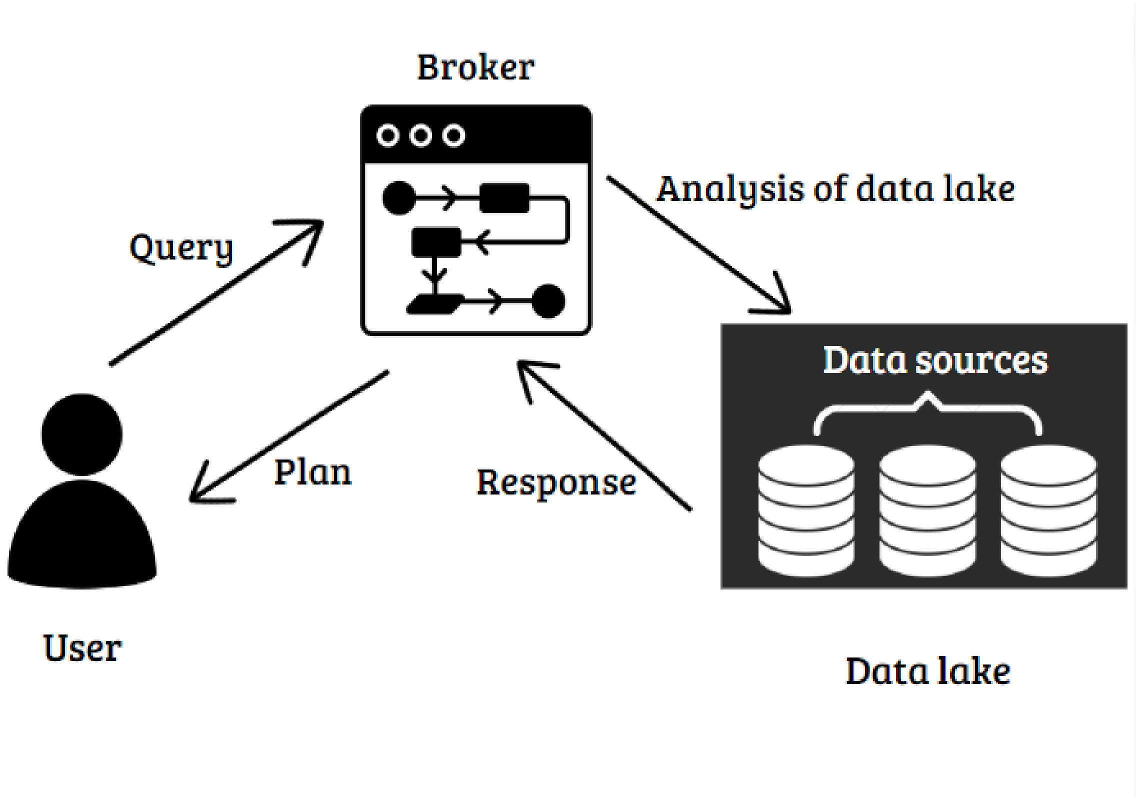


Figure 2.7: Simple broker structure

longer response time, since the selection is based on exhaustive search, and it also returns a low quality plan. The second broker model, as described in [3], has the possibility to analyzing the data and access them in order to improve the quality of the data. There are two types of operations performed: data cleaning technique, to improve the quality of each fragment, and merge algorithm, which improves the quality of the combination of the fragments. Thanks to this improvement the plan returned has a high quality, but this increases the complexity and it provides poor performance. The third broker model does not adopt the exhaustive search, like the other two broker models. It uses greedy algorithm, local search and tabù search. First the greedy algorithm finds a solution and if the solution found does not respect the constraint of price the local search is performed and it tries to find a solution with lower price than $Price^*$. Then the tabù search is executed and the overall quality is improved. As in the second model the quality of the data is improved, but the third model offers also higher performance in terms of execution time.

$$\min \sum_{i=1}^{N} Price(d_i) x_i$$

$$QD_r(\underline{x}) \geq QD_r^*, \forall r \in [1, R]$$

$$q(\underline{x}) \geq q^*$$

$$Price \leq Price^*$$

$$x_i \in [0,1]$$

Figure 2.8: Optimization problem for the fist broker model [5]

3 | Case study: stock market

After the description of the technologies and tools to manage and exploit high amount of data, it is necessary to describe what types of data can be useful to exhibit the features of the broker. In this thesis the data used come from the stock market and in the following sections a description of the characteristics of the stock market is conducted. In Section 3.1 there is a description of the stock market, where the types of analysis, along with their indicators, and the $OHLCV$ char are illustrated. In Section 3.2 are defined the providers and the type of data that will be used, then in Section 3.3 an essential feature of the stock market data is defined, the granularity, and, at last, in Section 3.4 are depicted some models for predicting the price values of the stocks.

3.1. Stock market

A stock market is a collection of buyers and sellers of stocks. A stock, also known as equity, consists of all the shares by which the ownership of a corporation or company is divided. Units of stocks entitle the owner to a fraction of the corporation's assets that is proportional to the total number of stocks possessed by the each shareholder. So the stock markets, also known as equity market or share market, are venues where buyers and sellers meet to exchange equity shares of public corporations. Some examples of stock markets are New York Stock Exchange and Nasdaq. Stock analysis is a method for investors and traders to make buying and selling decisions, as described in [11]. This process involves the study of past and current data and it helps investors and traders to take profitable decisions. There are two types of stock analysis: fundamental analysis and technical analysis. Fundamental analysis determines the value of a stock by analyzing available information, with a special emphasis on accounting information. To conduct fundamental analysis it is necessary to examine information on company's financial statements, balance sheet, income statement and cash flow statement [1]. But the most interesting method, that will be considered in the remainder of this thesis, is the technical analysis. Technical analysis focuses on the study of past and present price actions to predict the probability of future price movements. Technical analysts evaluate the financial market as a whole and

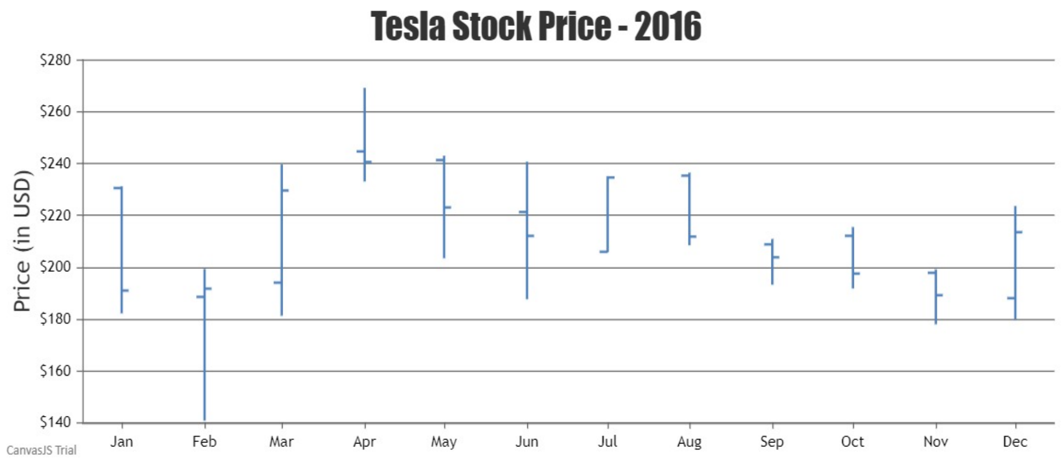


Figure 3.1: Example of OHLC chart with Tesla stocks

they are primarily concerned with price and volume, as well as the demand and supply factors that move the market. Technical analysis can be grouped into: stock prices and stock price indicators. Stock prices show open, high, low, and close prices for each period, with the closing price considered as the most important by many traders. A common representation of these characteristics is $OHLC$ chart, represented in Figure 3.1. $OHLC$ chart is a type of bar chart and it is useful since shows the four major data points over a period. This type of chart is represented by one vertical line and two short horizontal lines extending to the left and right with respect than the horizontal line. The vertical line indicates the volatility, in particular it refers to the maximum and minimum value reached during a specific period, in Figure 3.1 one month. The longer is the line, the more volatility and indecision it occurs in the market. The left horizontal line refers to the initial value in the new interval of time considered, while the right horizontal line shows the final value of the stock in the time interval considered. Another component of the stock prices, not reported by the $OHLC$ chart, is the volume that indicates the number of shares exchanged over a time interval. An example of the $OHLCV$ data belonging of a stock is reported in Table 3.1. Other remarkable stock prices parameters are dividends, the payments to shareholders made in shares, and splits, the issuing of more shares to shareholders.

On the other side, the stock market indicators are quantitative analysis tools used to interpret stocks, current status and future trends in the stock market, as describe in [6]. They are mathematical formulas and ratios used to predict how the market will move based on past patterns. There exist different types of indicators: trend followers and oscillators. A trend indicator checks for the direction and strength of a trend in a determined time interval and it uses price averaging to determine a baseline. Oscillators

Ticker	Date	Open	High	Low	Close	Volume
MSFT	05-02-22	277.71	284.94	276.22	284.47	35150000
MSFT	05-03-22	283.96	284.13	280.15	281.78	25980000
MSFT	05-04-22	282.59	290.88	276.73	289.98	33600000
MSFT	05-05-22	285.54	286.35	274.34	277.35	43260000
MSFT	05-06-22	274.81	279.25	271.27	274.73	37780000

Table 3.1: Example of daily Microsoft Corporation (MSFT) stocks

are another type of indicators whose values oscillate between upper and lower bounds, pointing out the volatility of the stock. They usually do not show the direction of a trend but rather the momentum at which the prices move [8]. An example of the indicators used to a describe a stock is reported in Table 3.2.

Ticker	Date	SMA(20)	SMA(50)	RSI(14)
AAPL	10-28-22	147.24	151.80	59.54
AAPL	10-31-22	145.10	151.42	55.22
AAPL	11-01-22	145.31	151.07	51.53
AAPL	11-02-22	145.51	149.90	51.24
AAPL	11-03-22	145.06	150.12	42.25

Table 3.2: Example of a daily Apple (AAPL) stock market indicators

A typical trend indicator is the simple moving average (SMA) which represents the unweighted mean of the typical stock price of s for n consecutive time interval, a day in Table 3.2, of equal length and it is calculated as :

$$\mathrm{SMA_n(s,t_i)} = \frac{\Sigma_{k=0}^{n-1} TP(s, t_{i-k})}{\mathrm{n}}$$

The parameter n is provided by the users and usually ranges between 10 and 200. Moving averages are used to smooth out short-term fluctuations, highlighting longer-term trends in the price history. When the stock price rises above the current SMA values is the beginning of a positive price trend and it may serve a as buy signal [8]. Other indicators similar to simple moving average are exponential moving average (EMA) and moving average convergence divergence (MACD). The exponential moving average gives more weight to the most recent price data that are considered to be more significant than older data. For this reason the EMA is more sensitive to price changes with respect than SMA. Exponential moving average is calculate as:

$$\text{EMA} = \text{Price(t)} * k + EMA(y) * (1 - k)$$

$$\text{k} = \frac{2}{N+1}$$

Where t refers to the current day, y refers to the previous day and N stand for the number of days in EMA. The moving average convergence divergence is an indicator that shows the relationships between two exponential moving averages and it is calculated as the difference between the 12-period EMA and 26-period EMA:

$$\text{MACD} = \text{12-period-EMA} - \text{26-period-EMA}$$

A common momentum stock market indicator is the relative strength index (RSI), it is used to measure the extend of which the price changes and it determines if the price of a stock is in a state of overbought or oversold. RSI values oscillates between a range of [0; 100], values below 30 is considered oversell, while values over 70 is considered overbought. The RSI is calculated as follow:

$$\text{RSI}_{\text{step one}} = 100 - [\frac{100}{1 + \frac{\text{Average gain}}{\text{Average loss}}}]$$

$$\text{RSI}_{\text{step two}}(n) = 100 - [\frac{100}{1 + \frac{(\text{Previous Average gain } * \ (n-1)) \ + \text{ Current gain})}{-((\text{Previous Average loss } * \ (n-1)) \ + \text{ Current loss})}}]$$

Where average gain indicates the average number of positive price changes, while average loss represents the average number of negative price changes of a stock over a defined period. Previous average gain and loss refer to a previous period. Another important stock price indicator is the momentum, which highlights the rate of change in price movement over a time interval and it is typically used to indicate a bullish momentum, when the price is rising, and a bearish momentum, when the price is falling. Even this indicator is adopted by trades to decide to buy or sell shares. It is calculated as:

$$\text{Momentum} = \text{V} - \text{V} * \text{x}$$

Where V stands for the latest price, $V * x$ stands for the closing price and x represents the number of days ago.

3.2. Data providers and adjustments

In order to be able to operate with the broker, large amount of data is required. The information regarding stocks can come from different sources, they can come directly from the stock market owner but, also, from individual brokers and dealer desks. So, the fundamental idea is to identify the sources that offer the requested data and then searching for the best ones in terms of data quality. To be able to benefit a high number of data providers, the stocks that will be analysed belong to Apple (AAPL) and Microsft (MSFT) because they are renowned companies. Example of data provider for Apple and Microsoft stocks are Nasdaq, the main stock market of Apple, Yhaoo finance api, quandl and kaggle. The data provided from the previous described data sources are characterize by high quality data, in fact they have a very high completeness and only a few datasets, have null-values [3]. The data used are unadjusted values, because the process to obtain adjusted values is very expensive. As a matter of fact to calculate the adjusted version (P_{adj}) of the close price from the unadjusted version (P_{unadj}), dividends and stock splits is calculated as follows:

$$F_{split} = \frac{Split_{new}}{Split_{old}}$$

$$F_{dividend} = 1 - \frac{Dividend_{old}}{P_{old}}$$

$$P_{adj} = F_{split} * F_{dividend} * P_{unadj}$$

Adjustments calculation rely, also, on many other corporate actions such as spin-offs, rights issues, mergers and acquisitions. This corporate actions requires to re-calculating the full stock price history for a specific company. So, this results in the computation of thousand of trading days, times the five parameters of stock price, $OHLCV$, for each stock, every day. This is the operation considering daily granularity but it can increase even more with hourly granularity, for example. The data analysed are characterize by high level of quality and we will consider them in order to choose the best set of sources to satisfy a query by one end-user of our broker system.

3.3. Granularity

Another important parameter to consider is the granularity. Granularity indicates the level of detail of the data, with high granularity it is present a high quantity of information to process, while data defined with low granularity have a lower amount of data to deal with. In the context of the stocks, high granularity means to have continuous update of the price of a stock every hour or, even, every minute, whereas the rate of updates of the price indicators with low granularity decrease, reaching one every month or year. It is possible to retrieve low level data from high level data without losing information. This property of granularity is very useful because data sources that support only high level granularity data can, anyway, provide data at a lower granularity. Unfortunately, it is not possible the opposite, from low to high.

3.4. Predicting stock price

The research of the best providers with high quality data at a low price, debated in this thesis, is one of the possible application for the broker and it can represent the starting point for more advanced studies regarding the prediction of the stock prices. Stock price prediction is a current challenge, there have been developed many stock price prediction methods, however the prediction accuracy is still far from being satisfying [43]. One example of models to predict prices is described in [42], where a $LSTM$ (long short-term memory) neural network is used as predictor. The neural network discovers the role of time series analysing the historical information of the stock market and, then, it explores its internal rules through the selective memory advance deep learning function of the neural network model, to obtain the prediction of the stock price trend. One crucial limitation is present in this model, it has been observed time lag of the prediction. Another important factor that can influence the prediction of the stock is the sentiment analysis. This evaluation exploits linguistic technologies and information provided by social media to investigate users' moods and psychological states of people. It has been shown in [9] that the analysis of the text contents of daily Twitter feeds has an accuracy of 87.6% in predicting the daily up and down changes in the closing values of the DJIA, a stock market index that tracks 30 large companies trading on Nasdaq and NYSE, and a reduction of the mean average percentage error by more than 6%. For this reason in [33] has been developed a model that predicts the stock market based on sentiment analysis information. There are other methodologies existing to predict price, for example in [22] is uses an $ARIMA$ (auto regressive integrated moving average) model for prediction of stock market movements.

4 | Broker model and methodology

In this chapter, some example of queries and the methodology of the broker model considering the stock market case of study will be discussed. Stock market data is a good approach to test the broker. The main attributes of the stock are indicators and stock price attributes. For the stock price the attributes are open, close, low, high, volume and date. These attributes describe the maximum, minimum, open, close values of the stocks, the number of stocks traded and the timestamp of the stock, that can be use as the key. Indicators are quantitative analysis tools used by market analysts to illustrate the current status and future trends in the stock market. They are mathematical formulas and ratios used to predict how the market will move based on past patterns. Some example of indicators are simple moving average (SA), relative strength index (RSI) and moving average convergence divergence (MACD). All these information are stored in the data lake with different data quality values and the broker has to choose the set of data streams that satisfy the request from the end-user. In order to make the job done by the broker easier, each data stream has a file that summaries the main characteristic of the data sources, the data catalog. In section 4.1 is illustrated the structure of the data catalog along with the description of all his metadata. In Section 4.3 and 4.4 all the steps of the broker model are explained, starting from the selection of the data sources and ending with the optimization of the plan. In Section 4.5 is specified how to calculate the data quality dimension values for each attribute and in Section 4.2 an example of a query is shown and a summary of the improvements introduced are reported.

4.1. Data catalog

In this section is described what a data catalog is and all the information that can provide, an example of data catalog is reported in 4.1. The metadata are introduced, describing all types of metadata considered and explaining all advantages that can supply for the entire data lake.

4.1.1. Usage of data catalog

The data catalog is a detailed inventory of all data assets in an organization, designed to find the most appropriate data for any analytical or business purposes. In particular the data catalog provides better understanding of data as a consequence of a detailed descriptions of the data, increases the operational efficiency and accelerates the selection of the best data sources. So, the data catalog summaries all the important features of the database presented in the data lake and the broker can exploit them to decide if the database described can be useful in the phase of the creation and optimization of the plan. If all data catalog of each data source are well organise, then is unlikely that the data lake turns into a data swamp, a collection of unusable data, that would result in a loss for the companies in term of earnings. In the data catalog there are metadata that provide the most important information of the data sources, documenting various aspects of data, such as the meaning of its content, the management of the organizations' knowledge on its data and to facilitate the indexing of the data [13]. So the definition of a good set of metadata is essential to guarantee adequate performances. An example of the data catalog with its metadata is reported in Listing 4.1.

```

{
  "databases": [
    {
      "id": "DB1",
      "fileName": "Database1",
      "dataTableName": "DatabaseD1",
      "metaDataTableName": "Database1",
      "granularities": [
        "Daily",
        "Weekly",
        "Monthly",
        "Yearly"
      ],
      "fields": [
        "ID",
        "timestamp",
        "ticker",
        "unadjusted_open",
        "unadjusted_high",
        "unadjusted_low",
        "unadjusted_close",
        "adjusted_close",
        "unadjusted_volume",
        "dividend",
        "split"
      ],

     "lastUpdate": "03-09-2022 20:00:00",
    "provider": "Nasdaq",
    "Value_distribution":"(array of values)",
    "similarities":"DB6, DB9, DB11",
    "provenance":"downstream:DB3, DB7. upstream:DB6, DB9",
    "dependencies":"inclusion:DB13. functional:DB14, DB16",
    "correlations":" (unadjusted_high, unadjusted_volume), (
   unadjusted_low, unadjusted_volume)"
    }
  ]
}
```

Listing 4.1: Data Catalog for the stock market use case

4.1.2. Metadata of the data catalog

As stated before metadata help data lakes to manage, index and enhance the value of data in the storage. When companies or industries are interested in the data belonging to a data lake, the metadata combined with a broker algorithm are used to perform an efficient and search of useful data, satisfying in terms of quality and price the customer. A broker algorithm, in fact, is a mediator between the users and the data sources, that suggest the best data streams to buy considering quality and price. The metadata can support the selection of the source to buy, performing a filtering process that discard data sources that do not provide profitable information. It is important to highlight that at this level it impossible to access the value of the attribute of each database, only the list of attributes is available but not their values. There are two types of metadata: one type describes the characteristic of one database, like the provider or the granularity supported for one database, while the other one describes the relations and similarities between all databases in the data lake. Here the metadata considered in the data catalog are discussed.

4.1.3. Metadata describing the characteristics of a single database

In this section the metadata that describe the information provided by a single data source are defined.

One of these is the granularity, which is a measure of the level of detail in a data structure.

For example, the granularity of measurement might be based on intervals of years, months, weeks, days, or hours. One good aspect of granularity is that it possible to retrieve lower granularity data starting from higher granularity data. A end-user can be interested in data with a precise level of granularity, so the broker should returns all data sources that contains data with that level of granularity or higher, the sources with lower granularity should not be considered.

The provider of the data specifies from which repository the data are taken, some example can be Nasdaq, kaggle and Yahoo Finance Api. It is a useful information used to eliminate all data sources coming from a different data provider with respect than the requested one.

The price of the source is essential for the broker algorithm to provide a good plan for the customer and it can be used to discard all the data sources that have a price higher than the maximum imposed by the request. Every fragment, derived from each data sources, has a different price.

It is also important to highlight when data are loaded or updated in the data source, so the timestamp of the last update of the data source must be considered.

A general description, given by the owners, can be useful for figuring out which datasets are more appropriate for the users and to filter out datasets that are marked as experimental or should not be showed to the user, can happen that some users cannot interact with some types of data.

In the data catalog is reported the list of the attributes that appears in the database. This information is very useful because it helps to discard the data sources that do not contain any required attributes from the request of the customer.

Another important information that metadata should provide are the knowledge regarding the value distribution of each attributes, for example can be helpful to know the maximum and minimun value, the mean and the standard deviation, number of row, number of attributes, distinctness, uniqueness, constancy, the data type and the semantic domain of each column. From this knowledge it can be decided to compute some join or union on datasets that lack on some information, in order to have the possibility to find complementary information or new insight. Distinctness, uniqueness and constancy are calculated as:

$$\text{Distinctness} = \frac{\text{Distinct values}}{\text{Actual values}}$$

$$\text{Uniqueness} = \frac{\text{Distinct values}}{\text{Total number of values}}$$

$$\text{Constancy} = \frac{\text{Most frequent value}}{\text{Total number of values}}$$

4.1.4. Metadata describing relationships between databases

These metadata can be useful in the case the data sources have a low quality data, characterize by null-values and errors. There are a lot of different types of metadata belonging to this category.

The provenance, used also in [21], gives information about which datasets were used to create a given dataset (upstream datasets), and those that rely on it (downstream datasets). This metadata could help the broker to eliminate some data sources that do not provide new or better information. For each data bases, in the catalog, are reported all upstream and downstream datasets if they exist.

For each data sources could be useful to check for some similarities between datasets. Two dataset, for example, can share a primary key column or multiple column. From two similar dataset it possible to execute some join or union in order to gather new information that with the two separate sources cannot be obtained. With this comparison between different datasets could be also useful to check for some inconsistencies or errors, it can happen that from two very similar dataset there is a multiple representation of the same real-world entity, this can help the broker to choose the dataset that do not contain errors or inconsistencies. In the data catalog for each data bases are listed all databases that share the same attributes or the primary key.

Dependencies could provide further information for the broker to accurately choose the best data sources. Inclusion dependency over two relational schema state that all values in one appear also in the other one. Functional dependency, written as $X \rightarrow$A, asserts that all pairs of records with same values attribute combination X must also have same value in attribute A. For example, it is a good choice to discard a dataset if another one contains all his information along with a lower price. This information are in the catalog where are listed all databases that have some dependencies with a given one.

Another metadata to consider are correlations. Correlations aim to establish whether a pair of variables are related, showing which variables influence any particular outcome metric. For example, it can happen that the unadjusted high value can influence the volume of the stock considered. It is important to emphasize that correlations establish a statistical relationship, but does not prove causation. For example, Pearson correlation it is a method to measure the strength of the correlation between continuous numerical variables.

4.2. Queries and constraints

Here the discussion on the management of the queries is described. It is important to outline that, the customers can specify the providers, maximum price, granularity, time interval of the data, minimum value of the data quality dimension and can specify the value and the type of the attributes the user wants. As already said, it is not possible in this phase to access the actual values of the attributes, the only information available is the list of the attributes of each data sources.

4.2.1. Example of a query

Supposing that a user is interested in the stocks of Apple in 2019 and he asks unadjusted open, SMA 20, unadjusted close and ticker equal to Apple as attributes, a weekly granularity and Nasdaq as provider. He requires a final plan in which the minimum value of each data quality dimensions is 0.85 and a maximum price of 20000. The data catalog can be used to help the selection of the best data sources. The first thing is to remove all data sources that have a higher price with respect than the maximum price of the request and the data sources providing data that do not belong to Nasdaq because they don't respect the requirements asked. For example all data streams that have as provider Yahoo finance, or all the sources thas have monthly or lower granularity, are discarded. Then there is the selection of the data sources that share at least one attribute between unadjusted open, unadjusted close, SMA 20 and ticker with the query, whereas the others that do not share any attributes will be not considered anymore. Between these data sources selected and with the metadata that specify the relation between datasets, like dependencies, provenance, value distribution, similarities and correlations, are identified dataset with some null values, inconsistencies or lack of information and are performed some operations, like union or join, in order to fix these errors, trying to gain new insight of the data and gather new information regarding the data sources. For example, if some schema lack information regarding unadjusted close it is possible for other data sources that describe the same stock and that contain the information of unadjusted close, to complete the knowledge of the first considered dataset. All the datasets unused in this phase can be discarded because they don't provide any new information. As already said, this last phase is not considered for stock market because the data have an high quality.

At this point for the remaining databases are associated values in order to describe the data quality dimension of each attribute. Then the data sources selected can be used as a input for the search and optimization phase that will choose the best set of data stream based on the price and the values of the data quality dimensions of each attribute requested.

This is an example of a query that a customer can formulate, but the user can ask different data modifying the query, he just needs to change the time interval of the data, provider, granularity, maximum price or attributes. For example, the user can asks Apple and Microsoft data from kaggle and Yahoo Finance Api, with weekly and hourly granularity, from the first half of 2021 and with volume, unadjusted close and unadjusted low as attributes, specifying also a maximum price and with different quality constraints. The resulting suggested data sources will be different and, if exists, the solution will respect the constraints imposed by the customer.

4.2.2. Improvements with respect previous implementations

This procedure is different with respect than the previous broker model because, formerly, all possible data sources were considered in the phase of the optimization and each attribute of them was evaluated in terms of quality, while in this implementation there is a filtering process, thanks to the information provided by the data catalog and price that discard all data sources that surely do not provide any useful information to satisfy the request of the customer. For this reason the databases that do not supply the provider requested, have a higher price with respect than the maximum, have a lower level of granularity and do not have at least one attribute shared with the request, are rejected. The granularity and the list of attributes were already included in the data catalog, while the provider has been added to make possible to implement this filtering process. This measure reduces the number of data to analyse improving the performance and reducing the total execution time without lowering the quality of the final solution. In particular, our method reduces the measurement time of the data quality dimensions that most affect the total execution time of the broker model. Also the calculation of the price has been slightly modified, with respect previous works.

4.3. Methodology

In this section will be explained the process of selection of the data sources for the end-user, analysing in particular how the candidates databases are chosen and then used as input for the phase of the creation and optimization of the plan.

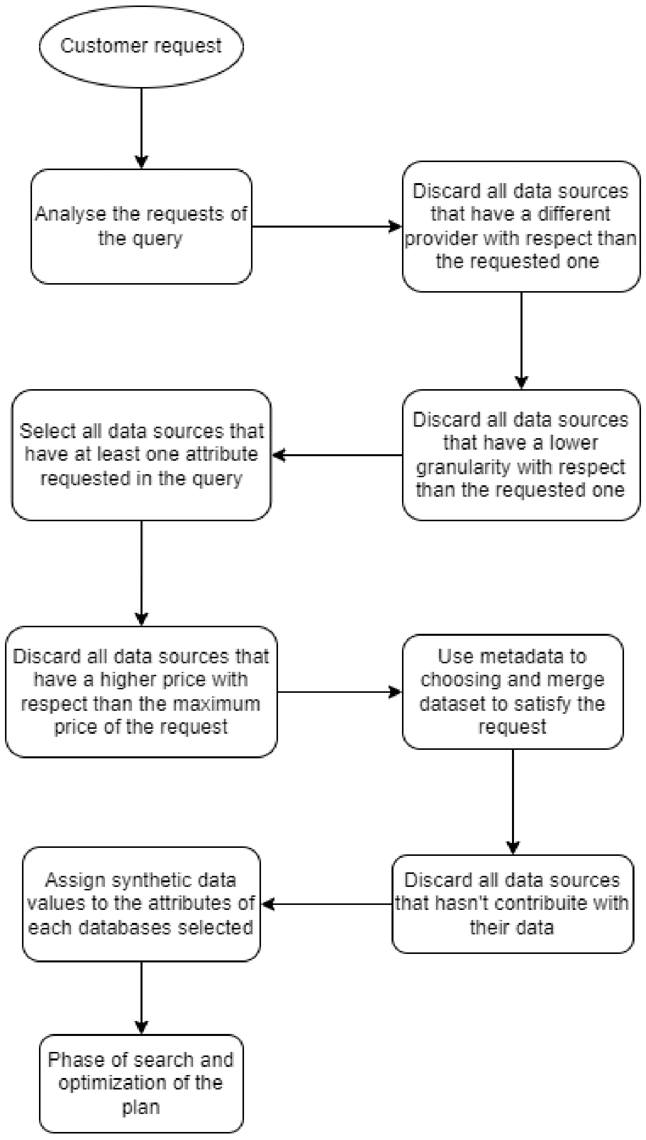

Customer request
Analyse the requests of the query
Discard all data sources that have a different provider with respect than the requested one
Discard all data sources that have a lower granularity with respect than the requested one
Select all data sources that have at least one attribute requested in the query
Discard all data sources that have a higher price with respect than the maximum price of the request
Use metadata to choosing and merge dataset to satisfy the request
Discard all data sources that hasn't contribuite with their data
Assign synthetic data values to the attributes of each databases selected
Phase of search and optimization of the plan

Figure 4.1: Procedure that the broker follows

4.3.1. Procedure

The purpose is to provide the best data sources that satisfy the quality and price constraints. The request, received as a query, is explored in order to find the most general information, like the attributes granularity, maximum price and providers requested. At this point, as it possible to see in Figure 4.1, the data sources that do not satisfy the requirement of the provider section, the data sources that have a price higher with respect than the maximum price of the request and the data sources that provide a lower level of granularity are discarded. Granularity, provider and the list of attributes are reported in the catalog, whereas the price is calculated during the creation of the fragment derived from each data sources, obviously all of them are considered before the phase of the creation and optimization of the plan and before the phase where the data quality values are associated to each attributes of the data stream. There is not interest in these data because they do not provide information that are in compliance with the request. Evaluating their attributes and including them in the phase of the creation and optimization of the plan would result in a increment of the computation time without any benefit. Then, are selected all data sources that share at least one attribute with the request, whereas all data sources that do not share any attribute of the request are discarded. At last the metadata that highlight the similarities between data sources are exploited to have a better understanding of the data sources selected. It can happen, as matter of fact, that some datasets have some lack of information, null values or incomplete columns, so thanks to the information provided by provenance, correlations, dependencies and similarities it is possible to perform join and union operations to fix this problem producing, also, new knowledge and information regarding each data source. All the data sources that do not provide any useful information or they are not involved in any operation are discarded because their information are supplied from other data sources. This last phase, where metadata describing the relationships between databases are used, can be useful in case of data presenting an high number of error or null values, but it is not considered in the case of stock market because the quality of the data is high. At this point the data quality dimension values are assigned to each attributes of the selected databases and then the phase of the search and optimization of the plan can start.

4.4. Optimization phase

The phase of creation and optimization decides the final plan to suggest to the user. Initially a solution is searched by the greedy algorithm in order to allow the local search to take and use it to find an admissible solution and, at last, the tabù search is executed to improve the solution found by the local search. The solution, or plan, is a collection of the data streams provided as input. In Subsections 4.4.2, 4.4.3 and 4.4.4 are depicted the algorithms adopted and in Subsection 4.4.1 is described the method to calculate the global quality of a generic plan.

4.4.1. Evaluation of the plan

The aim of the optimization phase is to maximize the overall quality of the plan, observing the conditions imposed. It is possible to model this issue as follows:

$$\begin{aligned}\max_{\vec{x}} \quad & f_{obj}(\vec{x}) \\ \text{s.t.} \quad & \text{Price}(\vec{x}) \leq V_{\text{price}} \\ & \text{Precision}(\vec{x}) \geq V_{\text{precision}} \\ & \text{Timeliness}(\vec{x}) \geq V_{\text{timeliness}} \\ & \text{Completeness}(\vec{x}) \geq V_{\text{completeness}} \\ & \text{Accuracy}(\vec{x}) \geq V_{\text{accuracy}}\end{aligned} \tag{4.1}$$

Where $\vec{x}$ represents the set of the fragments contained in the plan. The constraints force the data quality dimension values of the plan, for example Precision($\vec{x}$), to be greater or equal with respect than the requested values, $V_{\text{precision}}$, and the price, Price($\vec{x}$), of the plan must be lower or equal than the minimum price requested, V_{price}. The value of the objective function, $f_{obj}(\vec{x})$ depends on the four data quality dimensions: accuracy, timeliness, completeness and precision, and it is calculated as:

$$\begin{aligned} f_{obj}(\vec{x}) = (w_{acc} * \text{Accuracy}(\vec{x})) + (w_{time} * \text{Timeliness}(\vec{x})) \\ +(w_{compl} * \text{Completeness}(\vec{x})) + (w_{pre} * \text{Precision}(\vec{x}))\end{aligned} \tag{4.2}$$

Where each data quality dimension is multiplied by a weight factor w and it has to respect the following constraint:

$$1 = w_{acc} + w_{pre} + w_{time} + w_{compl}$$

4.4.2. Greedy algorithm

The optimization of the final result requires a starting solution, even not considering the constraints imposed by the end-user. The greedy algorithm, as describe in [15], builds an initial solution choosing between the fragments that minimize the value of the function, described in 4.3, for each attribute requested. A fragment contains the values of each attribute of a single database. As said, for each attribute requested are gathered all fragments that contain it and then the fragment that minimize the objective function is added to the solution. The function for each fragment evaluates the price and all the data quality dimensions: accuracy, timeliness, completeness and precision along with their weights. This relationship between price and quality is calculated as:

$$f(x) = \frac{\text{Price}(x)}{(w_{\text{prec}} * \text{Prec}(x) + w_{\text{compl}} * \text{Compl}(x) + w_{\text{acc}} * \text{Acc}(x) + w_{\text{time}} * \text{Time}(x))} \tag{4.3}$$

Where x represents the fragment that contains the attribute considered. The final result will be a plan that contain a collection of fragments, but the problem is that the greedy algorithm does not guarantees the constraints imposed, so it is highly probable that the plan will be not admissible. For this reason the local search is executed and, starting from an initial non admissible plan, it search for a plan in accordance with the price and quality desired. Algorithm 4.1 shows the pseudocode of the greedy algorithm.

Algorithm 4.1 Greedy algorithm

```
1: **Input:** Vector of attributes requested $X$, vector of fragments to analyze $Y$
2: **Output:** Initial plan $S$ maximizing the quality of each attribute requested
3: Let $X_{requested} := \{x \mid x_1, \ldots, x_n$ Attributes requested by the user$\}$
4: Let $X_{frag}(y) := \{x \mid x_1, \ldots, x_n$ Attributes inside the fragment y$\}$
5: Let $S_1 := \{y \mid y_1, \ldots, y_n$ Initial plan S$\}$
6: Let $S_1 \leftarrow \emptyset$
7: **for** attribute $x_i \in X_{requested}$ **do**
8:     Find all fragments $y$ contains attribute $x_i$
9:     Select a fragment $y$ that minimize the $f(y)$ score
10:    Add fragment $y$ to plan $S_1$
11: **end for**
12: **return** $S_1$
```

4.4.3. Local search

In this phase the local search considers the quality and price constraints imposed and tries to find a feasible solution starting from the output of the greedy algorithm. The local search can perform three actions, as stated in [15], to improve the quality: substitution, insertion and elimination. All these operations act modifying the components of the plan, the substitution replaces a fragment with another fragment, the insertion adds a new fragment and the elimination deletes one fragment. Each completed action allows to move inside the neighborhood of the plan, improving the quality of the solution. These operations are executed when there are constraints not satisfied in the plan and if the new solution found do not break some constraints satisfied before the execution of the move. At first the most violated constraint is considered and the fragment, that lowers the value the most, is eliminated or substituted, otherwise another fragment can be added to improve the value of that attribute. Once the action is identified the next most violated constraint is studied and the new neighborhood decides the next move to carry out. So, it is essential to measure how much a constraint is violated and it is calculated as the difference between the requested value and the value included in the plan:

$$\begin{aligned}\Delta_{Completeness} &= V_{completeness} - \text{Completeness(x)}\\ \Delta_{precision} &= V_{precision} - \text{Precision(x)}\\ \Delta_{timeliness} &= V_{timeliness} - \text{Timeliness(x)}\\ \Delta_{accuracy} &= V_{accuracy} - \text{Accuracy(x)}\\ \Delta_{price} &= \frac{\text{Price(x)} - V_{price}}{V_{price}}\end{aligned}$$

In the local search is crucial the definition of the neighborhood because it influences the final outcome. The neighborhood is defined as all the possible plans reachable, starting from one solutions and performing any action. If the solution found satisfies the constraints, then it is possible to go through the tabù search, otherwise another iteration of the local search is performed. The algorithm stops without accessing the next phase when the constraints are not satisfied and there are not any actions that improves the current solution or when a maximum number of iterations is reached. At this point it is possible to introduce the negotiation phase where the provider can add new fragment trying to increase the overall quality but increasing the total cost, and eventually, if all constraint are satisfied, it is allowed to access the tabù search phase. Algorithm 4.2 shows the pseudocode of the local search. At the beginning of the algorithm is checked if the plan obtained by the greedy algorithm violates the constraints. Then if the plan is not feasible a loop is accessed where are introduced E, that represents the removed data streams, F, that represents the added data streams, and S_{cycle} that indicates the plan in a determined iteration. Then is checked if all fragments have been analysed, if so the loop ends, otherwise the algorithm tries to replace the fragments that do not respect the constraints, improving the overall quality or reducing the price. If the new solution found is equal with respect than the previous solution the loop ends, otherwise a new iteration is executed. The algorithm ends when the found new plan satisfies the constraints or when a maximum number of iteration is reached.

4.4.4. Tabù search

The solution obtained through the local search is not a global optimum, it can be improved, so the role of the tabù search is to increase the quality of the plan found. As described in [15], the tabù search, given an admissible plan, examines all possible plans

Algorithm 4.2 Local search

```
1: Input: Output of the greedy algorithm $S_1$, maximum number of iteration $n$
2: Output: Admissible plan $S_2$
3: Initialize $step \leftarrow 0$
4: Initialize $S_2 \leftarrow S_1$
5: while isViolatingConstraints($S_2$) and $step < n$ do
6:    Initialize $S_{cycle} \leftarrow S2$
7:    Initialize $E \leftarrow \emptyset$ {Removed data streams}
8:    Initialize $F \leftarrow \emptyset$ {Added data streams}
9:    Initialize $constraint \leftarrow$ findMostViolatedConstraint($S_{cycle}$)
10:   if $S_{cycle}/F \neq \emptyset$ then
11:      if $constraint =$ price then
12:         Try to replace the data stream with the higher price
13:      else if $constraint =$ accuracy then
14:         Try to replace the data stream that is lowering accuracy
15:         If replacement is successful,
16:         try to add a new data stream w.r.t accuracy constraint
17:      else if $constraint =$ completeness then
18:         Try to replace the data stream that is lowering completeness
19:         If replacement is unsuccessful,
20:         try to add a new data stream w.r.t completeness constraint
21:      else if $constraint =$ timeliness then
22:         Try to replace the data stream that is lowering timeliness
23:         If replacement is unsuccessful,
24:         try to add a new data stream w.r.t timeliness constraint
25:      else if $constraint =$ precision then
26:         Try to replace the data stream that is lowering precision
27:         If replacement is successful,
28:         try to add a new data stream w.r.t precision constraint
29:      end if
30:   end if
31:   if $S_2 = S_{cycle}$ then
32:      break loop
33:   end if
34:   $step = step + 1$
35: end while
36: Return $S_2$
```

included in the neighborhood and chooses the action that bring to the solution with the best objective function, where the objective function is calculate in Equation 4.2. All the neighborhoods are based on the moves previously executed. This approach avoids the problem of the local search to get stuck in a local optimum because introduces the concept of the tabù list and allows the execution of action that leads to a worse solution. The tabù list is a vector containing the most recent fragments added to the plan and it prohibits all moves that exclude them from the plan, until a specific number of iterations is performed. In this way the tabù list avoid to return to already visited solutions and it assures a minimum number of iterations before eliminating or changing a fragment from the solution. One issue is the decision of the exact number of moves to perform in order to be able to eliminate a fragment from the plan. A low number of moves banned allow to have a higher number of moves available in the neighborhood of the considered solution, whereas a high number of forbidden moves allow to escape quickly from the local optimum. A solution could be to adopt a dynamic approach, as suggested in [15], a larger tabù list when the objective function is worsening to escape quickly from the local optimum, while a smaller tabù list is useful when the objective function is improving in order to have a higher number of actions available. At each step of the algorithm all plans reachable in the current neighborhood of the solution are considered and if the objective function of one plan is better, then the action is performed and the solution is updated, otherwise a move that worsen the objective function is performed. The algorithm stops when the maximum number of iterations is reached, when a maximum number of moves is reached from the last improvement of the objective function or when all moves are prohibited from the tabù list or conduct to a ineligible plan. Algorithms 4.3 and 4.4 represents the pseudocode of the tabù search. In the algorithm, starting from the solution obtained from the local search, the operations of substitution, insertion and elimination (10,11,12) are performed. At the end is checked if the new solution obtained is higher with respect than the current optimal, if is true the plan found become the optimal best, otherwise it is verified if the current objective function is the best one of the considered neighborhood. If the new objective function is better and the moves performed does not belong to the tabù list then the current optimal is updated.

4.4.5. Complete procedure

All the steps performed by the broker include the initial filtering of the data sources that do not provide suitable data, followed by the assignment of the data quality values, the execution of the greedy algorithm, local search and tabù search. Figure 4.3 shows all the phases for the creation and optimization of the plan, where "Selection of databases that

Algorithm 4.3 Tabù search

```
 1: Let $S_3 \coloneqq \{y \mid y_1, \dots, y_n$ Fragments of the current solution$\}$
 2: Let $S_{best} \coloneqq \{y \mid y_1, \dots, y_n$ Fragments of the best solution find at the end of the algorithm$\}$
 3: Let $S_{neighborhood} \coloneqq \{y \mid y_1, \dots, y_n$ Fragments of the optimal solution found for each move$\}$
 4: Input: Admissible plan $S_2$
 5: Output: Optimized plan $S_3$
 6: Initialize $S_3 \leftarrow S_2$
 7: Initialize $tabuList \leftarrow \emptyset$
 8: while Tabu search stopping criteria are not met do
 9:     Initialize $S_{cycle} \leftarrow S_3$
10:     $S_3 \leftarrow \{S_{cycle} \cup z - \{y\} \mid \forall y \in S_{cycle}\ and\ \forall z \notin S_{cycle}\ and\ X_{plan}(S_{cycle} \cup z - \{y\}) \supseteq X_{requested})\}$
11:     $S_3 \leftarrow \{S_{cycle} \cup z \mid \forall z \notin S_{cycle}\}$
12:     $S_3 \leftarrow \{S_{cycle} - \{y\} \mid \forall y \in S_{cycle}\ and\ X_{plan}(S_3 - \{y\}) \supseteq X_{requested}\}$
13:     $\mathrm{UpdateTabuList}(\mathrm{S_3}, \mathrm{y})$
14: end while
15: Return $S_3$
```

can provide useful data" is described in depth in Figure 4.1. After the elimination of useless data sources and the evaluation of data quality dimension, an initial solution is found through the greedy algorithm. The greedy algorithm is followed by the local search that tries to find a solution that respects all the constraints. The local search is executed until an admissible plan is found or when a maximum number of iteration is reached. If a feasible solution is achieved, it is possible to perform the tabù search, otherwise it is possible to increase the overall quality at the cost of an higher price. If the local search reaches the termination condition and the plan is feasible, then the tabù search is accessible, otherwise the procedure terminates because a suitable plan has not been found. At this point the tabù can be executed and it is performed until the overall quality is improved or the termination criteria are met. After this the final plan is obtained.

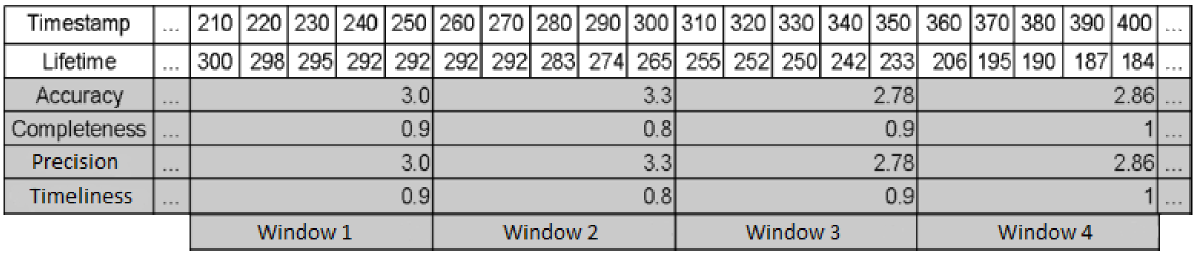
4.5. Data quality in data window

Due to the continuous stream of information, an appropriate data quality management requires a partition into many consecutive and non-overlapping data quality windows each one defined by the windows size, the starting point, the ending point and the list of attributes of the tuples contained [25]. The metrics used to evaluate attributes are accuracy, completeness, timeliness and precision. The evaluation of the data quality dimensions must adapt the new environment characterize by the data stream and the

Algorithm 4.4 Update tabù list

```
1: $updateTabuList(S, y)$
2: if isAdmissible($S$) then
3:     if $fobj(S) > fobj(S_{best})$ then
4:         $S_{best} \leftarrow S$
5:         $S_{neighborhood} \leftarrow S$
6:     end if
7:     if $fobj(S) > fobj(S_{neighborhood})$ and $y \notin tabuList$ then
8:         $S_{neighborhood} \leftarrow S$
9:     end if
10: end if
```

presence of windows, so the data quality assessment is updated, with respect than what described in Section 2.3.3, and discussed in the following sections.



Timestamp	...	210	220	230	240	250	260	270	280	290	300	310	320	330	340	350	360	370	380	390	400	...
Lifetime	...	300	298	295	292	292	292	292	283	274	265	255	252	250	242	233	206	195	190	187	184	...
Accuracy	...					3.0					3.3					2.78					2.86	...
Completeness	...					0.9					0.8					0.9					1	...
Precision	...					3.0					3.3					2.78					2.86	...
Timeliness	...					0.9					0.8					0.9					1	...
		Window 1					Window 2					Window 3					Window 4					

Figure 4.2: Example of data quality evaluation with sliding windows

4.5.1. Accuracy

The accuracy measures if the data are accurate, reflect the true state of the source information and there are no ambiguity in the representation. Considering the data streams and the windows the accuracy are evaluated for each window and then aggregated, if it is requested the overall accuracy of the stream. The accuracy over windows and data streams is calculated as:

$$\text{Accuracy}_{\text{window}}() = 1 - \frac{\# \ of \ records_{inaccurate}}{\# \ of \ records_{window}}$$

$$\text{Accuracy}_{\text{stream}}() = \frac{\sum_{i=1}^{N} Accuracy_{window_i}}{\# \text{ of windows in stream}}$$

4.5.2. Completeness

The completeness is given from the total number of item in a data stream over the actual number of data it should contain. In the context of stock market the completeness is evaluated over the ticker, date and time required. If all this attributes are present, the the data stream is considered complete. To achieve this is necessary to know the total number of unique timestamp in data lake, belonging to the given range of time, then the total number of unique timestamp is calculated for each data stream. As a result the completeness is defined as:

$$\text{Completeness} = \frac{\#\ of\ unique\_datetime_{Stream}}{\#\ of\ unique\_datetime_{Total}}$$

4.5.3. Timeliness

The timeliness can measure the age of a data item as the difference between the recording timestamp of the item and the current system time. It can be also considered as the punctuality of the data item with respect to the application context [25]. The formula to calculate it is the same described in 2.3.3.

$$\text{Timeliness} = \max\left(1 - \frac{\text{Currency}}{\text{Volatility}}; 0\right)^{s}$$

Where currency specifies how much data are recent, volatility specifies the duration of the information validity and the parameter s indicates the sensitivity of the measurement.

4.5.4. Precision

Precision refers to how close two or more measurements are to each other and it is characterized in terms of the standard deviation of the measured values. The level of precision in the data should be smaller or finer than the margin of error. Precision can be used along with accuracy to provide a better analysis because, it can happen, that a high accuracy is not enough and the precision can support the evaluation. The precision is calculated as :

$$\text{Precision}_{\text{window}} = 1 - \frac{1}{n} * \sum_{i=1}^{N} (v_n - \mu)^2$$

$$\text{Precision}_{\text{stream}} = \frac{\sum_{i=1}^{N} Precision_{window_i}}{\#\text{ of windows in stream}}$$

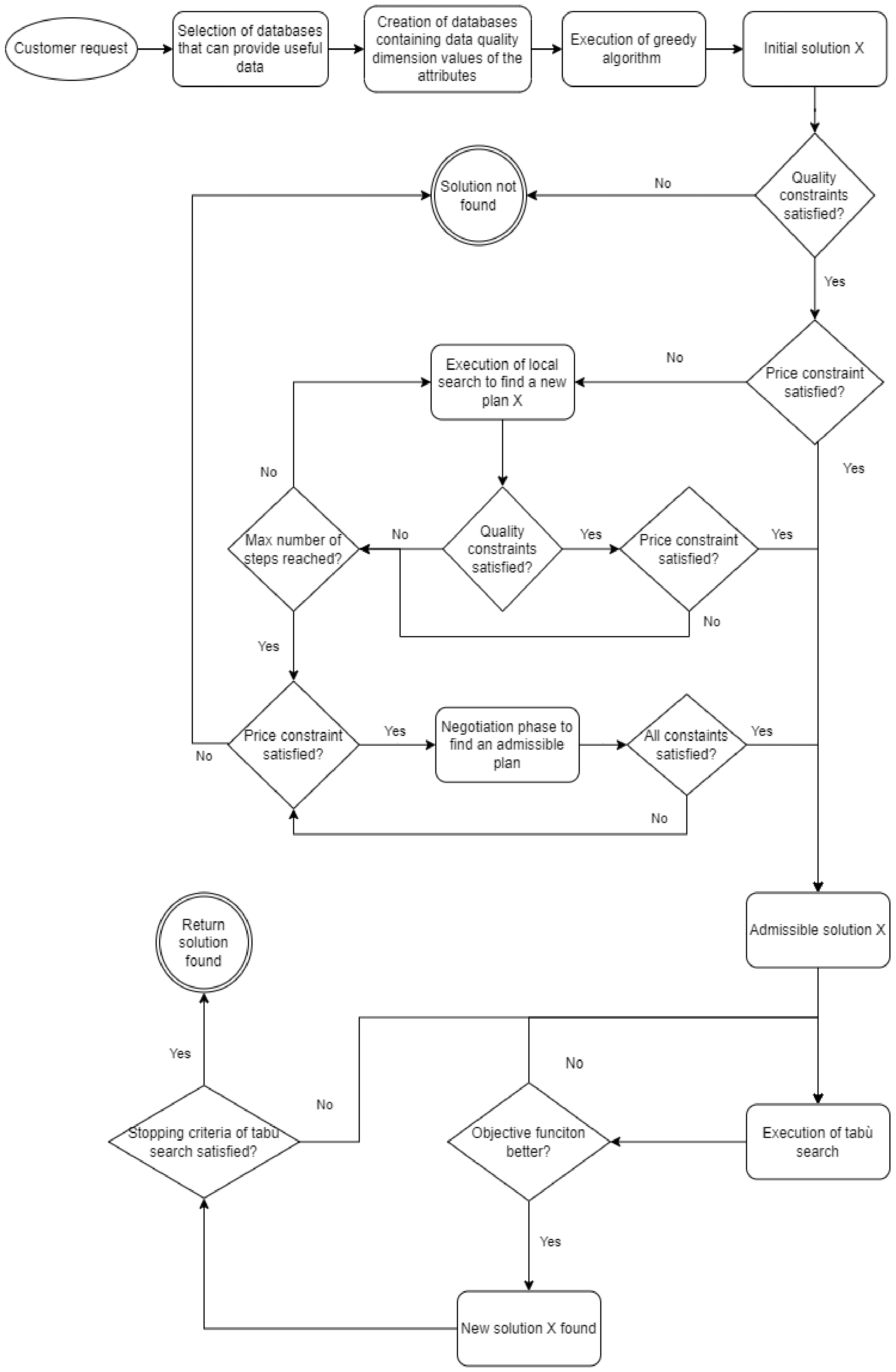

Customer request
Selection of databases that can provide useful data
Creation of databases containing data quality dimension values of the attributes
Execution of greedy algorithm
Initial solution X
Quality constraints satisfied?
No
Solution not found
Yes
Price constraint satisfied?
No
Execution of local search to find a new plan X
Yes
Quality constraints satisfied?
No
Max number of steps reached?
No
Yes
Price constraint satisfied?
Yes
No
Yes
Price constraint satisfied?
No
Yes
Negotiation phase to find an admissible plan
All constaints satisfied?
Yes
No
Admissible solution X
Execution of tabù search
Objective funciton better?
No
Yes
New solution X found
Stopping criteria of tabu search satisfied?
No
Yes
Return solution found

Figure 4.3: Representation of the complete procedure

5 | Application context

In the previous chapters, the technique used to improve the selection of the plan is shown, while here are described all the components that help the broker to carry out his task. The structure of data lake and data sources are described and it is explained how the values of the data quality dimensions are assigned to each attribute analysed because the data sources do not provide the quality of the attributes. These tools are considered and modeled for the stock market case study.

5.1. Data lake

The data lake is used to manage and organize massive collection of datasets belonging to the data sources. So, data sources offer their information to the lake. In the data lake considered there are a total of 36 data streams. There are some rules that the data lake has to follow in order to guarantee a correct preservation of the information and avoid to turn into a data swamp, a collection of inoperable data. Between all the guidelines that a good data lake should follow, the most important one is the presence of a catalog that summaries all the useful information offered by the data lake. The data catalog shows the knowledge that each data sources can provide to the lake, the most important are the fields, which include the attributes offered by a specific data source, granularity, which specifies the granularity that a data source can support, provider, which specifies the name of the data provider, and the value distribution, which specifies values like mean, variance and standard deviation of the attributes. Beyond these information, in the data catalog are present metadata that describe the relationships between data sources. These metadata are provenance, similarity, correlations and dependencies, these metadata are all described in Subsection 4.1.4. These information provided for each data source could be very useful because they can help the broker filtering all data sources that does not respect the constraints imposed by the customer. For example, if a data source has an higher price with respect than the maximum imposed by the user, then surely the data source cannot provide useful data, so it is possible to discard it from list of the possible solutions. The general structure of the data catalog of a single data sources is reported in Listing 5.1.

```
{
  "type": "object",
  "properties": {
    "lastUpdatedAt": {
      "type": "date",
    "description": "Date of last update"
    },
    "id": {
      "type": "string",
    "description": "Unique id for this database inside datalake"
    },
    "fields": {
      "type": "array",
    "description": "List of attribute contained in a data source"
    },
    "description": {
      "type": "string",
    "description": "General info about the data source"
    },
    "provider": {
      "type": "string",
    "description": "Name of the data provider"
    },
    "granularities": {
      "type": "array",
    "description": "High-level information about list of supported
   date-time field frequencies"
    },
    "provenance": {
      "type": "array",
    "description": "List of database that create a given database and
   those rely on it"
    },
    "Value distribution": {
      "type": "array",
    "description": "List of value that summarizes the values of each
   attributes"
    },
    "Dependencies": {
      "type": "array",
    "description": "List of database with a connection"
    },
    "Correlations": {
      "type": "array",
    "description": "List of attributes that influence each other"
    },
    "Similarity": {
      "type": "array",
    "description": "List of similar databases"
    }
  },
  "required": [
    "fields",
    "granularity"
  ]
}
```

Listing 5.1: Data lake catalog

5.2. Data sources

The data sources are part of the data lake and they provide the information to satisfy the customer. These data are collected from stock market data providers like Nasdaq, Kaggle, Yahoo Finance Api and Alphavantage. There are two parameters used to define the characteristic of the stock: stock price attributes and stock price indicator. Stock price attributes data are information about the current and historical price of a share of stock, it includes information like dividends, splits, open, close, low, high, volume and date. Open and close values indicate the value at the opening and at the end of the trading day. Maximum and minimum show the higher and lower value that a stock assume during a time interval. The volume represents the number of the stocks traded over a specific period and the date represents the moment in which the stock is analysed. Indicators are quantitative analysis tools used by market analysts to illustrate the current status and future trends in the stock market. They are mathematical formulas and ratios used to predict how the market will move based on past patterns. There are lots of indicators, but the most used are: simple moving average (SA), momentum (MOM) relative strength index (RSI) and moving average convergence divergence (MACD). The list of attributes contains an identifier in order to distinguish the stocks. This identifier is composed by a combination of the timestamp and ticker because they can provider an unique identifier for each stocks, as illustrated in Table 5.1 and Table 5.2. Before the phase of the optimization of the plan, for each attribute of all selected databases selected are associated data quality dimension values in order to evaluate the accuracy, precision, timeliness and consistency of the information provided by each fragment. In this thesis the data quality dimension values are synthetic data generated through a random function. In Listing 5.2 is reported the global schema along with all attribute that a stock can have, considering, also, the time variant of indicators in order to simulate different granularity support [3]. The other characteristics of a data source, like provider and last update, are reported in the data catalog of the data lake, as illustrated in Listing 5.1.In this thesis it has been considered 36 data sources, 12 describe only the stock prices ($OHLCV$), 12 describe only the indicators (RSI, MOM, MA, MACD) and the last 12 describe both the indicators and stock price attributes. The providers considered are Nasdaq, YahooFinanceApi and Kaggle. In the data lake there are 21 data sources with Nasdaq as provider, 9 from YahooFinanceApi and 6 from Kaggle.

ID	timestamp	ticker	...	unadjustedClose	unadjustedOpen
msft-43465	2018-12-31	msft	...	0.7737	0.761
msft-43467	2019-01-02	msft	...	0.7737	0.761
msft-43468	2019-01-03	msft	...	0.7737	0.761
msft-43469	2019-01-04	msft	...	0.7737	0.761

Table 5.1: Example of a microsoft stock showing the stock price

ID	ticker	timestamp	...	SMA 20	RSI 14
aapl-43830	aapl	2019-12-31	...	0.6797	0.6704
aapl-43832	aapl	2020-01-02	...	0.6797	0.6704
aapl-43833	aapl	2020-01-03	...	0.6797	0.6704
aapl-43836	aapl	2020-01-06	...	0.6797	0.6704

Table 5.2: Example of a apple stock showing the stock price indicators

```
CREATE TABLE ‘GlobalSchema‘ (
  ‘ID‘ VARCHAR NOT NULL,
  ‘ticker‘ VARCHAR(15) NOT NULL,
  ‘country‘ VARCHAR,
  ‘exchange‘ VARCHAR,
  ‘date‘ DATE DEFAULT NULL,
  ‘time‘ DATE DEFAULT NULL,
  ‘unadjusted_open‘ DOUBLE DEFAULT NULL,
  ‘unadjusted_high‘ DOUBLE DEFAULT NULL,
  ‘unadjusted_low‘ DOUBLE DEFAULT NULL,
  ‘unadjusted_close‘ DOUBLE DEFAULT NULL,
  ‘unadjusted_volume‘ DOUBLE DEFAULT NULL,
  ‘dividends‘ DOUBLE DEFAULT ’0’,
  ‘splits‘ DOUBLE DEFAULT ’1’,
  ‘adjusted_open‘ DOUBLE DEFAULT NULL,
    ‘adjusted_high‘ DOUBLE DEFAULT NULL,
    ‘adjusted_low‘ DOUBLE DEFAULT NULL,
    ‘adjusted_close‘ DOUBLE DEFAULT NULL,
    ‘adjusted_volume‘ DOUBLE,

    ‘50_day_moving_average‘ DOUBLE,
    ‘100_day_moving_average‘ DOUBLE,
    ‘200_day_moving_average‘ DOUBLE,

    ‘daily_dist_high‘ DOUBLE, /* Daily Distribution - High, Using a distribution based on daily information, the price value of the top/high level of the band*/
    ‘daily_dist_med‘ DOUBLE, /* Daily Distribution - Mid, Using a distribution based on daily information, the price value of the midpoint level of the band */
    ‘daily_dist_low‘ DOUBLE, /* Daily Distribution - Low, Using a distribution based on daily information, the price value of the bottom/low level of the band*/

    ‘weekly_dist_high‘ DOUBLE, /* Weekly Distribution - High, Using a distribution based on weekly information, the price value of the top/high level of the band*/
    ‘weekly_dist_med‘ DOUBLE, /* Weekly Distribution - Mid, Using a distribution based on weekly information, the price value of the midpoint level of the band */
    ‘weekly_dist_low‘ DOUBLE, /* Weekly Distribution - Low, Using a distribution based on weekly information, the price value of the bottom/low level of the band*/

    /* Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of a stock’s price. */
    ‘daily_MACD_Signal‘ DOUBLE,
    ‘daily_MACD‘ DOUBLE,
    ‘daily_MACD_Hist‘ DOUBLE,

    /* The Relative Strength Index (RSI) describes a momentum indicator that measures the magnitude (0-100) of recent price changes. */
    ‘daily_rsi‘ DOUBLE,
    ‘weekly_rsi‘ DOUBLE, /* The Relative Strength Index (RSI) describes a momentum indicator that measures the magnitude (0-100) of recent weekly price changes  */
    ‘monthly_rsi‘ DOUBLE, /* The Relative Strength Index (RSI) describes a momentum indicator that measures the magnitude (0-100) of recent monthly price changes  */

    ‘daily_mom‘ INT, /* Momentum direction, based on daily information. Values are either 1 (positive) or 0 (negative)*/
    ‘daily_mom_calc‘ DOUBLE, /* The momentum value, based on daily information. */

    ‘weekly_mom‘ DOUBLE, /* Momentum direction, based on weekly information. Values are either 1 (positive) or 0 (negative). */
    ‘weekly_mom_calc‘ DOUBLE, /* The momentum value, based on weekly information. */

    ‘monthly_mom‘ DOUBLE, /* Momentum direction, based on monthly information. Values are either 1 (positive) or 0 (negative). */
    ‘monthly_mom_calc‘ DOUBLE, /* The momentum value, based on monthly information. */

    PRIMARY KEY (‘ID‘)
  ) ENGINE=InnoDB;
```

Listing 5.2: Global Schema DDL

5.3. Population of quality tables

One step to perform, before the phase of the optimization, is to populate the quality table with measured or simulated quality dimensions values of the attributes, as explained in [3] and [15]. During the simulation the broker analyzes the data tables and generates their respective quality tables to use in the next phases to build the best plan. This stage is very important because it can conduct to a bottleneck, improving the execution time. In fact, for k columns subjected to quality analysis, q quality tables and n rows, the broker needs to update $q * n * k$ cells. For this reason, in [3], it has been developed a method to accelerate this operation. The solution reported combines all column updates into one row update since the where clause was same for all of them. Moreover, all update operations are combined into one SQL Update batch in order to minimize table accessing. To summarize, the broker combines all cell updates $q * k$ of a single row into one query and, when all updates are available, batched query ensures the number of database access to be one.

6 | Use cases and experiment results

In this chapter will be analysed how the data are extracted from the data sources and some use cases to test the broker, along with the analysis of the performances obtained. The queries are used to extract data from the sources, but they need to be defined in the context of data stream, so their structure and their usage will be discussed. Here, another objective is to show how the new functionalities, described in previous chapter, work and the advantages of using them, showing if an actual improvement, from previous implementations, has been obtained. In the following sections the management of the query is faced and it is explained on which data sources the queries are performed. In addition three different use cases are presented to show and simulate the execution of the broker, considering the data coming from the stock market, then an analysis on the performances obtained is carried out. In Section 6.1 is explained how the query are performed in the context of stock market data and in Section 6.2 some example of use cases is presented where the end-user asks for specific data. The environment where the tests are execute are reported in Section 6.3 and in Section 6.4 are described and analysed the results obtained. At last in Section 6.5 is reported a comparison, regarding the performances, between this broker model and previous broker model described in [3].

6.1. Definition of the queries

6.1.1. Query structure

A query is a process of retrieving information, that satisfy some criteria, from a data source. When the customer decides on which types of data is interested, a query is performed for each data source, whose data satisfy the user requirements, and the resulting fragments of data will be used as input for the next phases including the evaluation of the data quality dimensions and the creation and optimization of the plan. The information needed to execute the query are taken from the request of the user. Usually the user can ask for the time interval of the data, so he can specify the *windowStartDate* and *windowEndDate* and he can specify the *windowSize*, i.e. the granularity. Another parameter that is user-provided, as described in [3], is the ticker. The ticker is used as a global query variable to ensure that all operation will return homogeneous results. The structure of the query is reported in:

```
SELECT*
FROM DatabaseX_ticker
WHERE granularity_requested <= windowSize AND timestamp
BETWEEN windowStartDate AND windowEndDate
```

The *windowsSize* starting from the *windowStartDate* determines how often the data are retrieved from the data sources until the *windowEndDate* is reached. An example is reported, considering a weekly *windowSize*, the ticker equal to Microsoft and the data coming from March 2022:

```
SELECT* FROM DatabaseX_msft WHERE timestamp
BETWEEN 2022-03-01 AND 2022-03-07
SELECT* FROM DatabaseX_msft WHERE timestamp
BETWEEN 2022-03-08 AND 2022-03-14
SELECT* FROM DatabaseX_msft WHERE timestamp
BETWEEN 2022-03-15 AND 2022-03-21
SELECT* FROM DatabaseX_msft WHERE timestamp
BETWEEN 2022-03-22 AND 2022-03-28
SELECT* FROM DatabaseX_msft WHERE timestamp
BETWEEN 2022-03-29 AND 2022-03-31
```

6.1.2. Query and data catalog

The differences between previous implementation is that not all data sources provide data to use for the next phases. When all constraints imposed by the customer are inserted and before asking data from the lake, the data catalog is inspected. This step is very important because the data catalog describes the information retained in each data source, including the list of attributes, provider and granularity. It is easy to check from the data catalog if there are some data sources that do not respect any constraints. If some databases violate even one constraint, then the query on that databases is not executed because surely they can not provide useful data. Also the price of the data sources, even if it is not included in the catalog, is evaluated depending on the characteristics of each fragment and if it exceeds the maximum imposed price the data sources related can not be included in any plan. This improvement, as describe in 4.3.1, reduces the number of operation to perform lowering the total execution time. Therefore, before acquiring the data from the sources is appropriate to check the information provided by the data catalog.

6.2. Use cases

Here, are illustrated some examples of use cases where new characteristics of the broker are exposed. At first are illustrated the values that is possible to assign to the weights of the objective function and then it is explained how the attributes values of each data quality dimension are generated. Then, some example of queries are described, specifying all the parameters and the quality constrains imposed by the customer, that the final plan must respect.

6.2.1. Weights and data quality dimension values

Before the analysis of the uses cases, it is necessary to define the values of the weights used to calculate the objective function and how the data quality dimension values are assigned to each attribute of the selected databases. Usually the weight are set all to 0.25 in order to distribute the score, of each attribute, equally as follows:

$$1 = 0.25_{acc} + 0.25_{pre} + 0.25_{time} + 0.25_{compl}$$

But in the context of the stock market, the data have a very high level of completeness, so it could be possible to lower the weight of the completeness in favour of the other dimensions, an example is:

$$1 = 0.30_{acc} + 0.30_{pre} + 0.30_{time} + 0.10_{compl}$$

In this thesis the the values associate to each data quality dimension are synthetic values, in fact they are generated through a random function.

6.2.2. Use cases examples

In order to show the functionalities of the broker, some use cases are considered and executed along with a analysis of the results obtained. Then it is performed also a comparison between the new broker model and the old one, in order to see if the new functionalities introduced have some advantages. This scenarios will show how the request are managed and the potentiality of this broker.

6.2.3. First scenario

In the first scenario the user is interested to know the stock price attributes and the indicators values of the apple stocks. He asks for data with $startDate$ as 01/01/2019 and with $EndDate$ as 01/01/2020, with a $windowSize$ of 7 days. He requires specifically for data coming from Nasdaq stock market, an American stock exchange, and he set the maximum price at 10000. As attributes the broker has to retrieve all the $OHLCV$ unadjusted values, split, dividends, moving average, relative strength index, moving average convergence divergence and, of course, the ticker equal to Apple. The quality constraints imposed by the user, over accuracy, completeness, precision and timeliness, are all set to 0.86. The weight of all data quality dimensions have been set equal to 0.25 in order to distribute the score equally. In this case the data must be provided only from Nasdaq so it is possible that some data streams will be discarded. Same argument holds for the price and granularity, in particular the data sources that provide higher granularity with respect than weekly, like daily or hourly, are permitted while the sources that provide lower granularity are discarded. If the price of a single sources is over the constraint, 10000, then is eliminated because surely cannot be included in the final plan. The results of this tests are reported in the Section 6.4.1.

6.2.4. Second scenario

In the second scenario the user is interested to know the stock price indicators values, for each trading day, related to Microsoft stocks. He asks for the data with $startDate$ as 01/01/2020 and with $EndDate$ as 01/01/2021, with a $windowSize$ of 7 days. The user does not specify the provider, so all data provider are accepted, and he sets the maximum price at 35000. As attributes the broker needs to retrieve simple moving average, relative strength index, momentum, moving average convergence divergence and, of course, the ticker equal to Microsoft. The quality constraints imposed by the user for accuracy, precision, timeliness and completeness are all 0.85. The weights of accuracy, precision and timeliness are set to 0.3, while for the completeness is set to 0.1, this is done because usually the completeness has a very high quality, in real data quality dimension value in stock market, and so it could be interesting to increase the influence of the other quality dimension that can affect the objective function in different ways. The price is quite high so it is difficult that some data sources have been discarded. The granularity this time is weekly, hence only monthly and yearly granularity are not considered. The results of this tests are reported in the Section 6.4.2.

6.2.5. Third scenario

In the third scenario the user is interested to know the stock price values ($OHLCV$) of Apple, for each trading day. The user asks for the data with *startDate* as 01/01/2018 and with *EndDate* as 01/01/2019, with a *windowSize* of 7 days. He asks specifically for data coming from Nasdaq and Yahoo Finance Api stock market and he sets the maximum price at 35000. The customer asks the ticker, equal to Apple, and all attributes related to the stock prices, including: unadjusted close, unadjusted open, unadjusted low, unadjusted high, unadjusted volume, dividends and split. The quality constraints imposed by the user are 0.85 for accuracy, precision and completeness, while it is 0.91 for timeliness. The weight of all data quality dimensions has been set equal to 0.25 in order to distribute the score equally. The price is quite high so it is difficult that some data sources will be erased due to price. The granularity requested is daily, so all data sources that have higher or equal granularity are accepted, otherwise they are eliminated. The results of this tests are reported in the Section 6.4.3.

6.3. Test environment

The experiment conducted are executed in the following environment:

- **Processor**: Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz, 2208 Mhz, 8 core, 16 processori logici
- **Memory**: 16 GB DDR4 SDRAM.
- **Disk**: 1TB SSD

6.4. Experiment results

Before the introduction of the results of the experiments, it is important to outline some domain assumptions. The values of the data quality dimensions are associated to the each attributes with a random values generated from a probability range. If the data sources support a lower granularity with respect than the requested one, the data are aggregated to the granularity demanded. As a consequence also the indicator data values are auto-aggregated. For the price of data sources, since they can come in different ways as a subscription, one time fees or usage based fees, it has been decided to calculate it based on the number of rows in the dataset. Then based on the provider of the data the price is increased by a percentage, for example if the provider is Nasdaq the price is increased by a percentage between 10% and 40%. Moreover, each scenarios are tested with different number of stream as input and it could be interesting to see the difference between the results.

6.4.1. First experiment

In this scenario, described in Section 6.2.3, the accuracy input generates uniform scores between (0.76,0.95) and, for sure, all the databases have at least one attribute useful because the request include both the stock price attributes and the indicators. The price, since the only provider is Nasdaq, can be increased by a percentage between 10% and 40%. As expected, considering the test with 9 different input streams, the output identifies as a solution only one provider $Database8$, due to fact that the maximum price is low and the price of the plan with two different data streams is higher. The filtering process has eliminated 4 data sources because they come from different provider, Kaggle or YahooFinanceApi, with respect than the requested one, meanwhile the granularity and price of the remaining databases respect the restriction. The plan found respects all the constraints imposed, in fact the quality metrics of the solution are:

- precision: 0.8570
- timeliness: 0.9308
- accuracy: 0.8567
- completeness: 0.8390
- price: 8275
- objective function: 0.8709

Comparing the results with the tests run with 18, 27 and 36 data streams, it shows that there is no huge difference, in term of quality, between the experiment done. In fact the higher objective function is 0.8855 reached by the test including 18 data streams, whereas the lower value of the objective function is 0.8709 reached by the test with 9 data streams. The best values is only 1.65% better than the worst one. There are not big differences in price, between tests with distinct number of input, since the final plan can included at most one data source due to the light constraint in price. The considerable difference between all the tests is the total execution time. The total execution time is influenced by the quality measurement time of the databases and plan calculation time. While the quality measurement time is greatly influenced by the number of the input streams, the plan calculation time is not so much influenced by that. With 9 data streams the time necessary to associate all data quality dimension values to each attribute is 128 seconds, while with 36 data streams the time needed is 343 seconds, there is an increase of 167.98%. In order to show the differences between the objective functions of the plans obtained with different number of input, in Figure 6.1 all the values are reported.

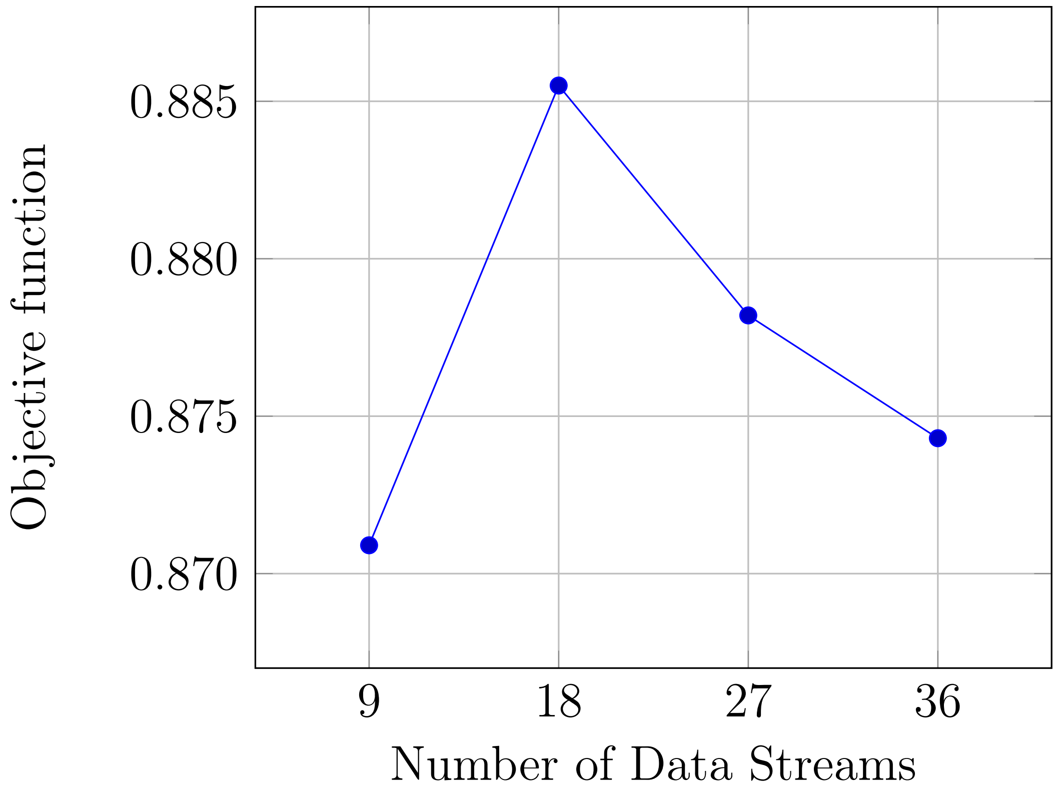


Figure 6.1: Objective function considering different number of data streams

6.4.2. Second experiment

In the second experiment, described in Section 6.2.4, the accuracy input of precision and accuracy are included between (0.75,0.95), while the accuracy input of precision and timeliness are included between (0.70,0.99). Here all providers, Nasdaq, YahooFinanceApi and Kaggle, are accepted and the price increment is between 10% and 40%. In the lake there are data sources providing only stock price attributes that, in this scenario, have been discarded, along with the data streams with lower granularity than weekly (only the database 36). For this reason, before the phase of optimization, 6 databases out 18, considering the test that use the 50% of the lake, have been eliminated. The solution provided by the broker include $Database4$, $Database10$, and $Database14$, and the quality metrics are:

- precision: 0.8474
- timeliness: 0.8581

- accuracy: 0.8508
- completeness: 0.8542
- price: 25869
- objective function: 0.8523

One interesting thing is to evaluate and visualize the actual improvement of the objective function from the temporary plan made by the greedy algorithm and the final plan optimized by the tabù search, considering the tests including 25%, 50%, 75% and 100% of the data lake. For this reason in Figure 6.2 are reported the objective functions discovered by the greedy algorithm and by the tabù search. It can be seen that the tabù has improved the quality of the final generated plans respectively of 0.351%, 0.458%, 2.730%, and 3.448%.

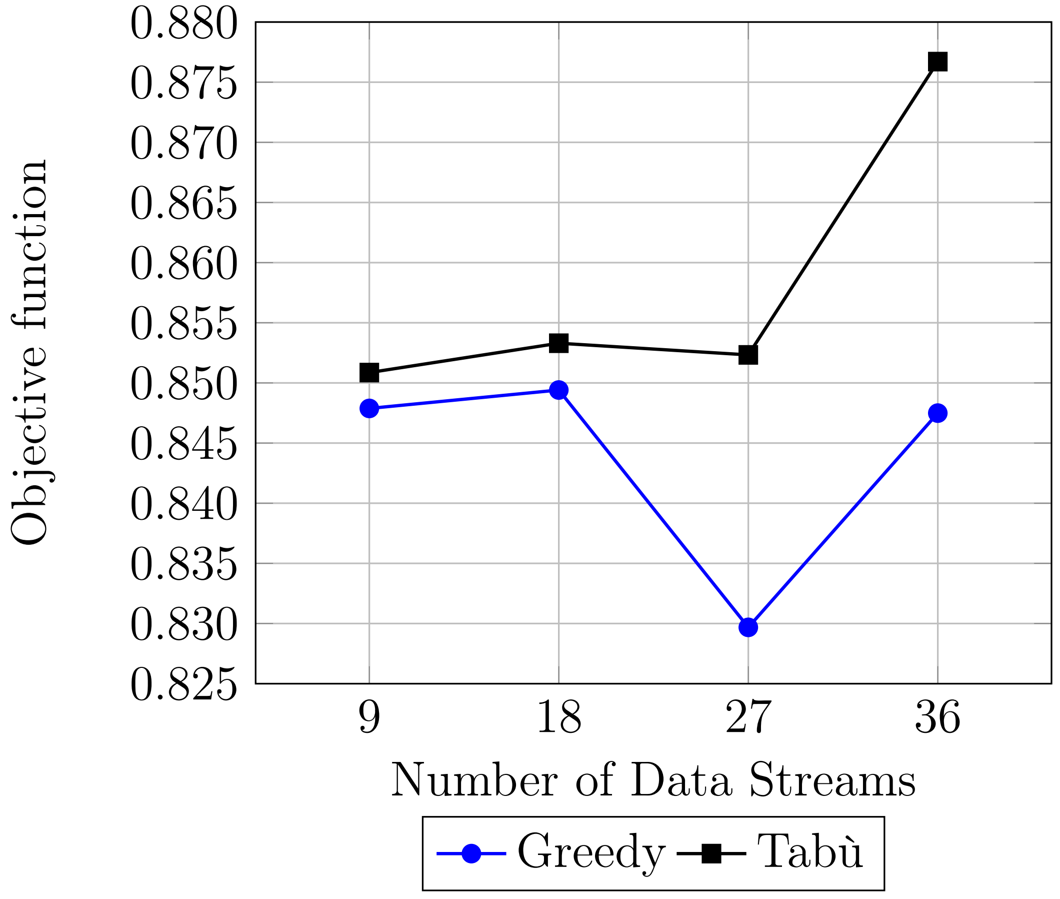


Figure 6.2: Comparison of the objective funciotn of greedy algorithm and tabù search

6.4.3. Third experiment

In the third experiment, described in Section 6.2.5, the accuracy input of precision and accuracy are included between (0.76,0.99), while the accuracy input of precision and timeliness are included between (0.74,0.96). The providers are Nasdaq and YahooFinanceApi and there is an increment in the price that is between 10% and 40%. Considering the test with 18 data streams as input, the resulting plan includes $Database2$ and $Database11$. Since the attributes requested ask only for stock price attributes, all data sources containing only indicators are discarded, the same it holds for provider where all data coming from Kaggle are not considered, meanwhile there are not data streams eliminated due to price, because the constraints are light, and granularity, because all the sources have higher granularity. Therefore the total number of data sources removed are seven. The quality metrics of the solution found are:

- precision: 0.8505
- timeliness: 0.9253
- accuracy: 0.8528
- completeness: 0.9035
- price: 18862
- objective function: 0.8830

The solution proposed two data streams in the plan because the two sources individually can not fulfill the constraints, in fact $Database2$ has a timeliness equal to 0.9025, that is below the minimum requested 0.91, and $Database11$ has a completeness equal to 0.8492, that is below the minimum requested 0.85. The presence of two sources in the solution affects the price of the plan, in fact it is 227.94% higher than the price of the plan found with 36 input streams. This big difference is explained due the presence of more sources that can improve the possibility to find a fragment satisfying alone the constraints imposed. This is reflected also in the objective function obtained from the two test, using 18 data streams as input the objective function is 0.8830, while using 36 data source is 0.9029. The big disadvantage to have more sources available for the optimization phase is the increase of the execution time, indeed the execution time of the broker using 50% of the lake is 223 seconds and it is 61.94% faster than using the 100% of the lake, i.e. 36 data sources. In the following Figure 6.3, the diversities of the total execution time between the test performed using different number of input streams is illustrated.

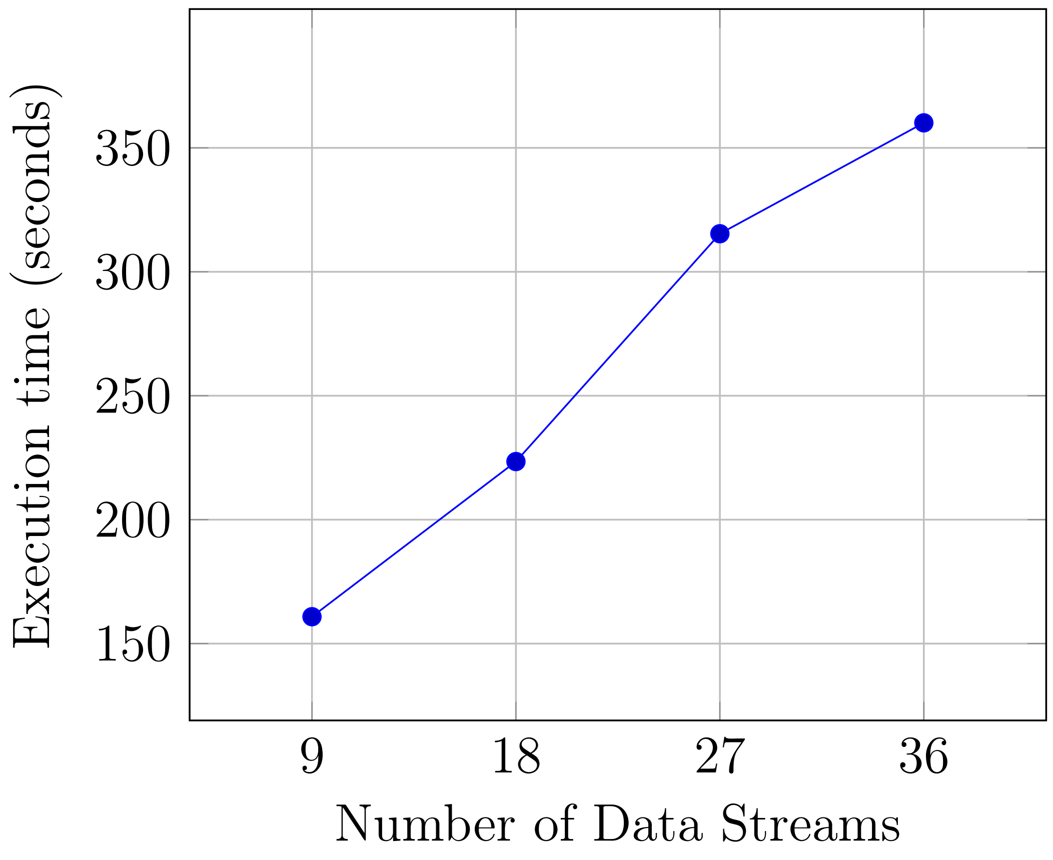


Figure 6.3: Execution time considering different number of data streams

6.5. Comparison with previous broker model

In order to emphasize the improvements done, a comparison between the performance of the broker model discussed in this thesis and the broker described in [3] is conducted. The focus is on the execution time, that should be lower in the new broker model, and, also, the objective functions of the two models are reported. The constraints requested in the query will be the same, of course in the new version is specified also the provider, while in the older version it is not considered because it is not supported.

6.5.1. Query formulation

The query considered, to test the broker, requires as attributes the stock prices values and as ticker Apple. The providers requested are Nasdaq and Kaggle, granularity is daily, maximum price is 25000, *startDate* is 01/01/2020 and *endDate* is 01/01/2021. The quality constraints desired are all equal 0.80 for all data quality dimensions. The weight of all data quality dimensions have been set equal to 0.25. The accuracy input, same for both evaluations, assigns values in the range between (0.70.0.95).

6.5.2. Analysis result

As expected the total execution time is lower in the new broker with respect than the older one. This happens because the information regarding attributes requested, granularity, price and provider are exploited before the insertion, for each databases, of the data quality dimension values related to each attribute and before the phase of the optimization of the plan. As a matter of fact is possible to see from the Figure 6.5, that the new broker model is, considering the different number of input, respectively %52.25, %40.43, %50.94 and %58.95 faster than the older version. Regarding the objective function it is not possible to perform an accurate analysis because the data quality values are assigned randomly, anyway Figure 6.4 are reported the values computed of the objective function.

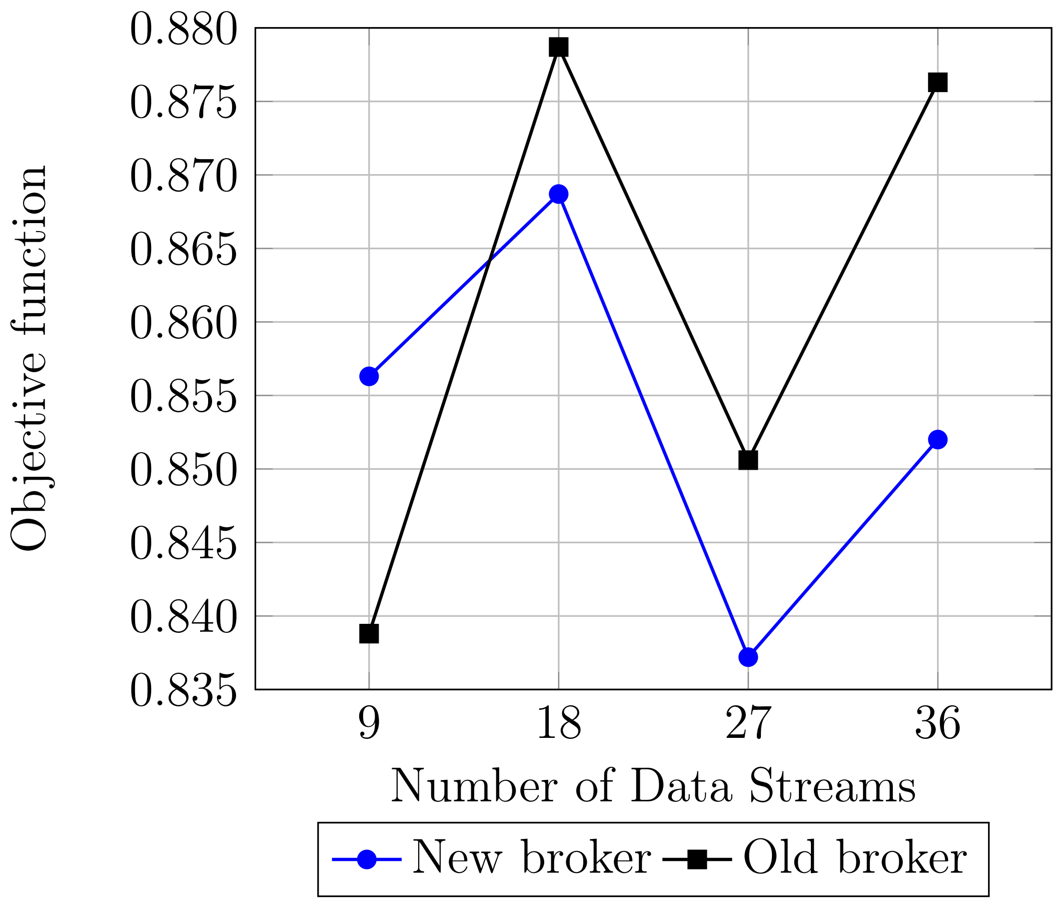


Figure 6.4: Comparison of the objective function of the two broker models

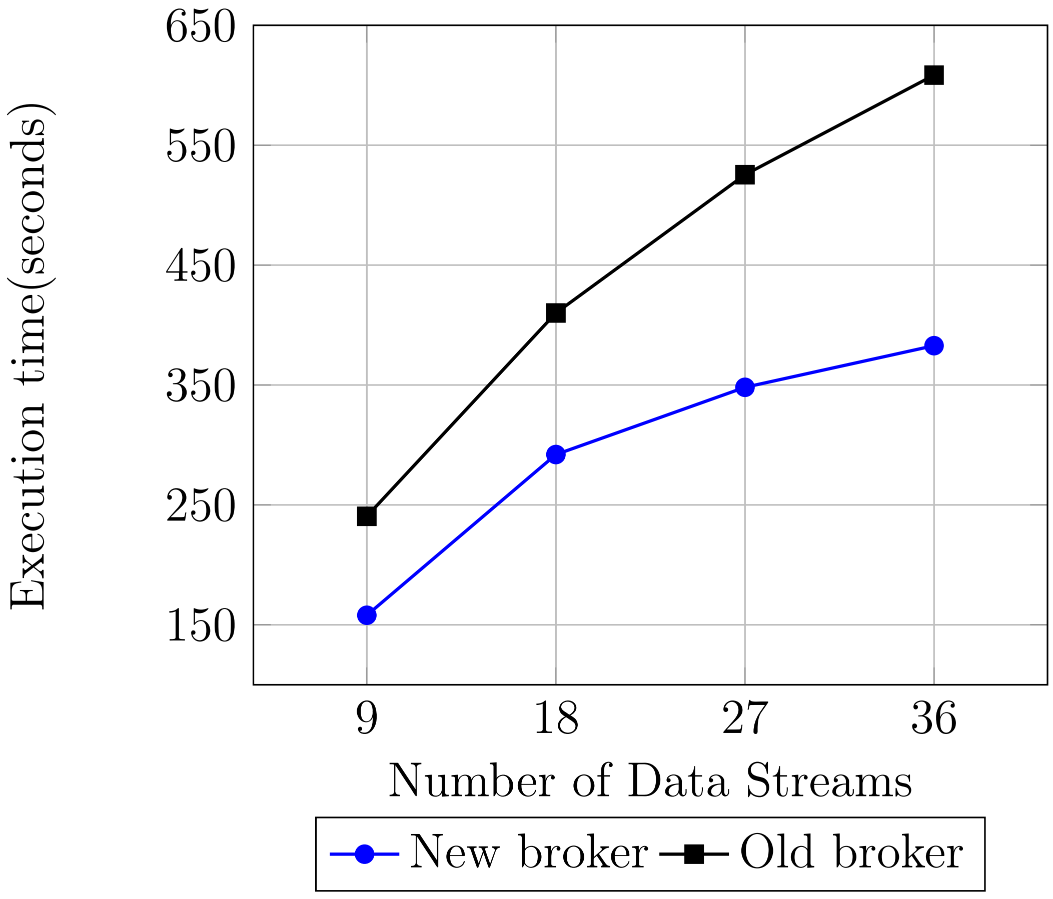


Figure 6.5: Comparison of execution time between the two models

7 | Conclusion

The necessity of having high quality data is fundamental for the corporations to make profitable business decision. For this reason it a good choice to rely on the broker, in fact it analyses and selects the data sources, included in the data lake, to please the request of the customer regarding the type and the level of quality of the data. This new version add some new feature that aim to reduce the total execution time and to increase the information provided by the data catalog. Indeed a classification of the metadata belonged to the data catalog is done and some new metadata has been added. There are two types of metadata, one describes the characteristics of the source and the other one describes the relations between different data sources. The new metadata added are: provider, similarities, value distributions, correlations, provenance and dependencies. The provider indicates from which repository the data are taken. The value distributions shows some statistic data regarding the values of the attribute, specifying for example mean, standard deviation, minimum and maximum values. Meanwhile the metadata including provenance, correlations, dependencies and similarities aim to show if there relationships between the data sources in the lake. Similarities indicate if two or more datasets share the same attributes, provenance give information about which datasets were used to create a given one, and those that rely on it, dependencies states if the knowledge supplied from a given data source is not include in other sources and, at last, correlations specify whether a pair of variables are related, influencing any particular outcome metric. The procedure of broker algorithm has been modified. In previous works all the data sources, selected as input, were taken in consideration for the assignment of the data quality dimension values and for the optimization of the plan. In this thesis a filtering process is implemented and only the data streams eligible to be part of the final plan are considered. Indeed before the assignment of the values and optimization all the data streams that do not have the same level of granularity or higher, do not come from the provider requested by the customer, do not share any attribute with the constraint and have an higher price with the respect the maximum imposed, are discarded and not considered for the consecutive steps. This alteration allow to reduce the total execution time required to build the plan. Also the estimate of the price has been modified, in [3] was calculated based on the number of record of the data source, while in this thesis the price relies, again, on the number of number of record of the data source but also on the provider of the data. All this new features are described in Chapters 4, 5 and 6.

7.1. Future works

The possible improvements that can be done in future works involve the introduction of the broker model in different field, rather than the stock market, or the adoption of a new procedure and heuristic to guarantees a better execution time and level of quality of data. Another topic that can be explore is the field of data cleaning, in fact could be useful the implementation of some data cleaning technique that can increase the overall quality of the data, obtaining better suggestion from the broker.

Bibliography

[1] C. Abad, S. A. Thore, and J. Laffarga. Fundamental analysis of stocks by two-stage dea. *Managerial and Decision Economics*, 25(5):231–241, 2004.

[2] H. F. Amaral, S. Urrutia, and L. M. Hvattum. Delayed improvement local search. *Journal of Heuristics*, 27(5):923–950, 2021.

[3] M. Andac. Quality-aware data stream selection in stock market scenario, 2021.

[4] A. Arasu, S. Babu, and J. Widom. The cql continuous query language: semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142, 2006.

[5] D. Ardagna, C. Cappiello, M. Comuzzi, C. Francalanci, and B. Pernici. A broker for selecting and provisioning high quality syndicated data. In *International Conference on Information Quality (ICIQ 2005)*, pages 262–279. MIT Press, 2005.

[6] A. Bagtas. Stock market indicators. `https://review42.com/resources/stock-market-indicators/`.

[7] A. Beheshti, B. Benatallah, R. Nouri, V. M. Chhieng, H. Xiong, and X. Zhao. Coredb: a data lake service. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2451–2454, 2017.

[8] A. Behrend, C. Dorau, R. Manthey, and G. Schüller. Incremental view-based analysis of stock market data streams. pages 269–275, 01 2008. doi: 10.1145/1451940.1451978.

[9] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

[10] L. Cai and Y. Zhu. The challenges of data quality and data quality assessment in the big data era. *Data science journal*, 14, 2015.

[11] J. CHEN. Stock analysis. `https://www.investopedia.com/terms/s/stock-analysis.asp`.

[12] C. Diamantini, P. L. Giudice, L. Musarella, D. Potena, E. Storti, and D. Ursino. A new metadata model to uniformly handle heterogeneous data lake sources. In

A. Benczúr, B. Thalheim, T. Horváth, S. Chiusano, T. Cerquitelli, C. Sidló, and P. Z. Revesz, editors, *New Trends in Databases and Information Systems*, pages 165–177, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00063-9.

[13] R. Eichler, C. Giebler, C. Gröger, H. Schwarz, and B. Mitschang. Modeling metadata in data lakes—a generic model. *Data & Knowledge Engineering*, 136:101931, 2021.

[14] A. Farrugia, R. Claxton, and S. Thompson. Towards social network analytics for understanding and managing enterprise data lakes. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1213–1220. IEEE, 2016.

[15] GaeGigs. Broker model.

[16] M. Gavrilas. Heuristic and metaheuristic optimization techniques with application to power systems. *Technical University of Iasi, D. Mangeron Blvd., Iasi, Romania*, 2010.

[17] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang. Leveraging the data lake: Current state and challenges. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 179–188. Springer, 2019.

[18] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang. Modeling data lakes with data vault: practical experiences, assessment, and lessons learned. In *International Conference on Conceptual Modeling*, pages 63–77. Springer, 2019.

[19] G. Gigerenzer and W. Gaissmaier. Heuristic decision making. *Annual review of psychology*, 62(1):451–482, 2011.

[20] F. Glover and M. Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[21] A. Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang. Goods: Organizing google's datasets. In *Proceedings of the 2016 International Conference on Management of Data*, pages 795–806, 2016.

[22] S. M. Idrees, M. A. Alam, and P. Agarwal. A prediction approach for stock market volatility based on time series data. *IEEE Access*, 7:17287–17298, 2019.

[23] B. Inmon. *Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump*. Technics publications, 2016.

[24] A. Katal, M. Wazid, and R. H. Goudar. Big data: issues, challenges, tools and good

practices. In *2013 Sixth international conference on contemporary computing (IC3)*, pages 404–409. IEEE, 2013.

[25] A. Klein and W. Lehner. Representing data quality in sensor data streaming environments. *Journal of Data and Information Quality (JDIQ)*, 1(2):1–28, 2009.

[26] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment*, 12(12): 1986–1989, 2019.

[27] F. Nargesian, K. Q. Pu, E. Zhu, B. Ghadiri Bashardoost, and R. J. Miller. Organizing data lakes for navigation. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1939–1950, 2020.

[28] F. Naumann. *Quality-driven query answering for integrated information systems*. Springer, 2002.

[29] F. Naumann and C. Rolker. Do metadata models meet iq requirements? 1999.

[30] I. D. Nogueira, M. Romdhane, and J. Darmont. Modeling data lake metadata with a data vault. In *Proceedings of the 22nd International Database Engineering & Applications Symposium*, pages 253–261, 2018.

[31] K. Patroumpas and T. Sellis. Window specification over data streams. In *International Conference on Extending Database Technology*, pages 445–464. Springer, 2006.

[32] L. L. Pipino, Y. W. Lee, and R. Y. Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.

[33] A. Porshnev, I. Redkin, and A. Shevchenko. Machine learning in prediction of stock market indicators based on historical data and data from twitter sentiment analysis. In *2013 IEEE 13th International Conference on Data Mining Workshops*, pages 440–444. IEEE, 2013.

[34] F. Ravat and Y. Zhao. Data lakes: Trends and perspectives. In *International Conference on Database and Expert Systems Applications*, pages 304–313. Springer, 2019.

[35] F. Ravat and Y. Zhao. Metadata management for data lakes. In *European Conference on Advances in Databases and Information Systems*, pages 37–44. Springer, 2019.

[36] T. C. Redman and A. B. Godfrey. *Data Quality for the Information Age.* Artech House, Inc., USA, 1st edition, 1997. ISBN 0890068836.

[37] P. Sawadogo and J. Darmont. On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 56(1):97–120, 2021.

[38] P. N. Sawadogo, E. Scholly, C. Favre, E. Ferey, S. Loudcher, and J. Darmont. Metadata systems for data lakes: models and features. In *European conference on advances in databases and information systems*, pages 440–451. Springer, 2019.

[39] M. Scannapieco and T. Catarci. Data quality under a computer science perspective. *Archivi & Computer*, 2:1–15, 2002.

[40] Y. Wand and R. Y. Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95, 1996.

[41] R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33, 1996.

[42] D. Wei. Prediction of stock price based on lstm neural network. In *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pages 544–547. IEEE, 2019.

[43] S. Wu, Y. Liu, Z. Zou, and T.-H. Weng. S_i_lstm: stock price prediction based on multiple data sources and sentiment analysis. *Connection Science*, 34(1):44–62, 2022.

[44] Y. Zhao, F. Ravat, J. Aligon, C. Soulé-Dupuy, G. Ferrettini, and I. Megdiche. Analysis-oriented metadata for data lakes. In *25th International Database Engineering & Applications Symposium*, pages 194–203, 2021.

A | Code

```


for (DatabaseInfo db: Options.catalog.databases) {

              //Check that the db has at least one provider requested
   by the customer
          String dbProvider = Arrays.toString(db.provider);
          sameProvider = false;
          String[] noProviderSelected = {};
          if (Arrays.asList(Options.args.provider).equals(Arrays.
   asList(noProviderSelected))  ) {
            sameProvider = true;
          }
          else {
            for (String argsProvider : Arrays.asList(Options.args.
   provider)) {
              argsProvider = "[" + argsProvider + "]";
              sameProvider = argsProvider.equals(dbProvider);
              if (sameProvider) {
                break;
              }
            }
          }

          //Check if at least one attribute requested is contained in
   the db selected
          dbContainsAtLeastOneAttributeReq = false;
          for (int k = 0; k < Options.constraints.
   getRequestedAttributes().length;  ++k) {
            String attributeRequested = Options.constraints.
   getRequestedAttributes()[k];
            if (db.fields.contains(attributeRequested)) {
              dbContainsAtLeastOneAttributeReq = true;
              break;
            }
```

```
        }

        //Check if the granularity requested is supported by the db selected
        dbSupportGranReq = false;
        for (int l = 0; l < db.granularities.size(); ++l) {
          int granularitySupported = GranularityType.fromGranularity(db.granularities.elementAt(l));
          if (Options.args.granularity >= granularitySupported) {
            dbSupportGranReq = true;
            break;
          }

        }

        if(numberOfDbSelected < Options.numberOfFragments && sameProvider && dbContainsAtLeastOneAttributeReq && dbSupportGranReq && Arrays.stream(db.supportedTickers).anyMatch(value -> value.equalsIgnoreCase(Options.currentTicker))) {
          newList.add(db);
          System.out.println(db.fileName + " can be used to create a plan");
        }
        else {
          if (numberOfDbSelected < Options.numberOfFragments) {
            if (!dbSupportGranReq) {
              System.out.println(db.fileName + " is discarded because do not support the same or lower level of granularity (" + GranularityType.fromInteger(Options.args.granularity)  + ") " + " requested");
            }
            else {
              if (!dbContainsAtLeastOneAttributeReq) {
                System.out.println(db.fileName + " is discarded because do not contain at least one attribute (" + Arrays.asList(Options.constraints.getRequestedAttributes()) + ") " + "requested");
              }
              else {
                if (!sameProvider) {
                  System.out.println(db.fileName + " is discarded because do not contain at least one provider (" + Arrays.asList(Options.args.provider) + ") " + "requested");
                }
                }
              }
            }
          }
          numberOfDbSelected ++;
        }
```

Listing A.1: Selection of eligible data sources based on provider list of attributes and granularity

A | Appendix B

```

{
  "databaseCount": 36,
  "databases": [
    {
      "id": "Database1",
      "fileName": "Database1",
      "dataTableName": "DatabaseD1",
      "supportedTickers": [
        "aapl",
        "msft"
      ],
      "metaDataTableNames": {
        "aapl": "Database1_aapl",
        "msft": "Database1_msft"
      },
      "granularities": [
        "Daily",
        "Weekly",
        "Monthly",
        "Yearly"
      ],
     "provider": [
        "Nasdaq"
      ],

      "fields": [
        "ID",
        "timestamp",
        "ticker",
    "unadjusted_open",
        "unadjusted_high",
        "unadjusted_low",
        "unadjusted_close",
        "adjusted_close",
        "unadjusted_volume",
    "dividend",
        "split"
      ],
      "lastUpdate": "29-08-2021 20:00:00"
    },
    .
    .
    .
    {
      "id": "Database36",
      "fileName": "Database36",
      "dataTableName": "DatabaseD36",
      "supportedTickers": [
        "aapl",
        "msft"
      ],
      "metaDataTableNames": {
        "aapl": "Database36_aapl",
        "msft": "Database36_msft"
      },
      "granularities": [
        "Weekly",
        "Monthly",
        "Yearly"
      ],
    "provider": [
        "Nasdaq"
      ],
      "fields": [
        "ID",
        "timestamp",
        "ticker",
        "SMA_20",
        "SMA_50",
        "SMA_200",
        "RSI_14",
        "MOM_10",
        "MACD_12_26_9",
        "MACD_Hist_12_26_9",
        "MACD_Signal_12_26_9"
      ],
      "lastUpdate": "29-08-2021 05:00:00"
    }
```

```
  ]
}
```

Listing A.1: Data Catalog used to execute the tests

List of Figures

List of Tables

Acknowledgements

I would like to thank Prof. Cinzia Cappiello and Prof. Danilo Ardagna whose precious knowledge, expertise and assistance have been indispensable during the writing of this thesis. I would like also to thank my friends for the constant help. Last but not least, I would like to thank my family for the continuous support.