1 INTRODUCTION

Today, more than ever, the world lives immersed in data. With ever-increasing rates of change, the data and information derived from them become essential for strategic, tactical and operational decision-making that benefit organizations.

Of the many systems that support Business Intelligence, data warehousing systems play an important role in planning organizational activities, as they allow us to build highly optimized business applications that serve as a basis or support tool for decision making and give users the opportunity to build their own data exploitation paths [1].

This ability, coupled with the fact that tools are becoming more intuitive, makes specialists less dependent on the intervention of IT to create their analysis models on datasets.

Thus, the Business Intelligence (BI) environment becomes more complex and difficult to understand as more and more datasets are integrated.

1.1 Problem Statement

The amount of information associated with the data lifecycle and the need to be agile to understand and act on increasingly tight deadlines puts tremendous pressure on all those involved in processes that require quality data to make the best decisions.

Because documentation is not always accessible to potential stakeholders at the right time and when it is, it is fragmented into different documents and formats.
People need information that identifies, locates, and describes data that enables them to act quickly in an ever-evolving context.

The research follows a bottom-up strategy to tackle the problem.

1.2 Context

The main motivation for this research stems from the fact that, as a computer professional, the researcher performs functions that are closely linked to application development and BI projects.

Additionally, there is a perception that it is possible to increase the value of the BI environment by improving practices in metadata management, more specifically in how it collects, structures and shares technical and functional information about the data used using IT as tools that meet people's expectations and needs.

1.3 Goals and Questions

Recognizing these needs leads us to the question of how we can describe and materialize the information that is associated with the data and thereby benefit people and processes.

To this end, it intends to answer the following research questions:

- What is the need for metadata?
- What metadata is required?
- How metadata can be implemented?
- What benefits can be obtained?

1.4 Action Research Method

Action research is a five-step process repeated as many times as necessary to achieve the objectives initially defined in that cycle [2].

1. Diagnosis: the objective at this stage is to identify the problem and opportunity for improvement while maintaining a holistic view of the domain where the problem fits.
2. Action Planning: this stage explores the references that can help solve the problem, defines the approach and objectives of the intervention, and considers possible alternatives.
3. Implementation: in this phase, the actions defined in the previous phase are executed.
4. Evaluation: The purpose of this phase is to verify that the actions taken have solved the problems identified.
5. Specific learning: the last phase identifies and records the conclusions resulting from the process.

The paper continues in this way:

a) Diagnosis: analyzes the problem and underlies the need to implement metadata;
b) Related Work: presents the theoretical foundations and references that influenced the work;
c) Designing Architecture: describes the process of implementing metadata using the attribute-driven design method;
d) Current State: describes the current situation and outlines the next steps;
e) Conclusions: reflects on the work done and its practical consequences.

2 DIAGNOSIS

2.1 Problem Space

The process of BI application’s creation comprise several tasks that can be grouped by three development tracks, which can be re-executed in parallel, whose ultimate goal is the delivery of the application [3]. Namely, the ETL track (to design and feed databases), application track (to design and build end-reports and queries) and the metadata repository (to create information that enables different kind of users to navigate the BI environment).

Some of process information deliverables, such as databases schemas and data dictionaries, are manually managed by the people involved in the process and are shared to answer questions that arise from other members who were not on the project team or to address integration issues due to failures in data warehouse feeding processes.

When information is held within departmental boundaries, it will inevitably have an undesirable consequence: organizations will have more difficulty in meeting the internal and external challenges. This translates into problems such as decreased productivity of technicians, analysts and managers, increasing labor costs and making hard to achieve personal and organizational goals.

For instance, business users want to understand the objectives, indicators and the data sets of BI apps
already existed on want to create a new one, but do not know if someone has already made. One way to check that is by asking to the person responsible or to the functional support team the rules and data sets on which the application depends.

The functional support team, in turn, will try to identify the scope and related data sources, using their own catalog, which contains all extractions of the transactional system that will feed the data warehouse, namely, the structure and content, along with the filling rules that the system implements.

In case of not being possible to give a proper answer to business, they ask to development teams to investigate. Those teams have their own catalogs to manage some information about data warehouse sources and its dependencies, and can even look to database and programming scripts to explain what were the transformations applied to data sets.

Therefore, we see three kind of users interacting within the context of the BI environment.

2.2 Solution Space

Business wants to know what datasets are available. The functional support team links transactional systems and data warehouse through their own documentation and development team maintains the data structures and technical processes that support the capture and delivery of BI databases and applications.

Table 1 presents some of their concerns.

<table>
<thead>
<tr>
<th>Who</th>
<th>Concerns</th>
</tr>
</thead>
</table>
| Business Teams       | **Context:** Identification and use of quality data is important to prevent errors arising from misinterpretation of the data that may be shared with third parties (e.g., regulators, shareholders).  
|                      | • Reduce time in data source selection  
|                      | • Reduce number of complaints |
| Support Teams        | **Context:** Improve communication between business areas and IT; Reuse information about transactional systems data sources.  
|                      | • Increase responsiveness to business requests |
| Development Teams    | **Context:** Database creation, documentation, and ETL processes are the most time-consuming activities negatively affecting time-to-market.  
|                      | • Reduce ETL program development time  
|                      | • Reduce technical documentation time |

In order to support the sharing of information about the data sets available in the BI environment, the action taken was to develop a metadata repository that will allow:

- Business teams know what data sets exist, who is responsible for them, and who maintains them;
- Functional support teams explain the content of information sources sent to the data warehouse;
- Development teams explain the databases and ETL rules about the structures of these databases.

3 RELATED WORK

3.1 Metadata

Several definitions and categories have been proposed for the characterization of metadata, in the context of information systems. Mainly, is divided into three categories: technical, business and operational metadata ([4], [5], [6],[3]).

**Technical metadata** provides a description of technology-related concepts. Provides technical teams with information about application systems and data warehousing systems that enable them to maintain and evolve these systems over time.

**Business metadata** provides a description of systems in a business language and helps to map the
information available in business intelligence systems in a non-technical way.

Operational metadata is related with the use that is made of data, both in terms of access and processing.

Some metamodels have been suggested to specify metadata representation for multiple domains such as CWM™ - Common Warehouse Metamodel being de facto standard in the data-warehousing domain or ISO/IEC 11179 is considered a de jure standard, since it lays the foundation for representing metadata for an organization. In addition to these, vocabularies have been created to describe information objects on the Web, but the complexity of this models and restrictions to the use of relational databases technologies prevents us to explore them more deeply.

3.2 Metadata Architecture

Several architectural approaches can be used to provide, store, integrate, maintain and make metadata accessible to consumers [7].

Centralized: a centralized architecture consists of a single repository (a relational or object-oriented database) with the copy of metadata from various sources.

Distributed: a distributed architecture does not have a repository that permanently stores metadata. While maintaining a single access point, queries are made to source systems in real-time.

Hybrid: a hybrid architecture combines characteristics of centralized and distributed architectures.

Bi-Directional: a bi-directional architecture allows metadata to change in any part of the architecture (source, data integration, user interface) and then feedback is coordinated from the repository into its original source.

3.3 Software Architecture

Software system architecture is a high-level view that links business requirements to the new software system that implements those requirements [8]. It allows us to see how system functionality is distributed across software and hardware components, how software is mapped to hardware, and how people interact with those components, and is intended to ensure certain qualities, such as modifiability, availability or performance.

The attribute-driven design is a five-step method to design software architectures:

1. Choose an element of the system to design.
2. Identify the ASR (architectural significant requirements) for the chosen element.
3. Generate a design solution for the chosen element.
4. Inventory remaining requirements and select the input for the next iteration.
5. Repeat steps 1–4 until all the ASRs have been satisfied.

The inputs to the method are listed in Table 3, Table 4 and Table 5.

4 DESIGNING ARCHITECTURE

The design process took into account some architectural goals (Table 2) and use cases.

4.1 Business Goals

<table>
<thead>
<tr>
<th>Goal</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to market</td>
<td>Faster information about data sets means faster and safer decisions for the business; Support teams, developers, and database administrators are keen to document the data sets created in the development of BI apps to improve the process of building those applications; Increasing cohesion between teams by sharing information.</td>
</tr>
<tr>
<td>Cost and benefit</td>
<td>Misuse of data entails reputational and financial risks; Using open-source packages whenever possible; Get motivated developers working on business logic.</td>
</tr>
<tr>
<td>Roll-out schedule</td>
<td>The aim is to have a system that evolves over time. That is, allow continuous increases in functionality within a proportional time window.</td>
</tr>
</tbody>
</table>
4.2 Architectural Drivers

4.2.1 Functional requirements and use cases

1. The system must support metadata extraction from SQL Server relational databases.
2. The system shall support user access through a portal made available on the intranet.
3. The system must support application access to the metadata repository through an API.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1: Produce technical metadata</td>
<td>The use case starts when the Technical Team actor wants to extract the technical metadata from a DBMS. The system shows the actions that user can perform to extract metadata to the screen or file. If an error occurs while attempting to access the metadata, the system records and issues an alert. If there is no error, the case ends when the system extracts the metadata for the screen or file (indicating the number of metadata read to the screen).</td>
</tr>
<tr>
<td>UC2: Publishing metadata</td>
<td>The use case starts when the Technical Team actor wants to update the repository with technical metadata. The system asks user to choose the file he wants to submit. The system validates the existence of the file and checks the format; In case of errors, a message is displayed. The case ends when the system updates the repository with the DBMS technical metadata.</td>
</tr>
<tr>
<td>UC3: Add business metadata</td>
<td>The use case starts when the user has already chosen a data set that they want to modify. The system presents a form asking the Support Team actor to fill in the metadata associated with the dataset. When the actor chooses the &lt;save&gt; option, the system stores these changes and redirects the user to the dataset view, which ends the case.</td>
</tr>
</tbody>
</table>

4.2.2 Quality Attributes

<table>
<thead>
<tr>
<th>Quality Attribute</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>AQM-1</td>
<td>Quality: Modifiability</td>
</tr>
<tr>
<td></td>
<td>Support for a new relational database provider is added to the system (producer) as part of the incremental process of new features. The change is made at initiation time, affecting the smallest number of artifacts.</td>
</tr>
<tr>
<td></td>
<td>Tactics: Restrict dependencies, encapsulate</td>
</tr>
<tr>
<td>AQS-1</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td>An unknown person or program attempts an unauthorized access to the metadata of relational database systems. The system resists and records the attack.</td>
</tr>
<tr>
<td></td>
<td>Resist attack (authentication and authorization), recover from attacks (maintain an audit trail)</td>
</tr>
<tr>
<td>AQI-1</td>
<td>Interoperability</td>
</tr>
<tr>
<td></td>
<td>The data producer sends metadata from database management systems to the repository. The request is accepted, correctly interpreted and data is sent to the repository in 99.9% of cases.</td>
</tr>
</tbody>
</table>
The database administrator wants to capture metadata from relational databases. The system assists him by showing the options he can choose. The administrator can obtain the metadata he has chosen, completing the task successfully.

### 4.2.3 Constraints

<table>
<thead>
<tr>
<th>ID</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRN-1</td>
<td>The system must be accessed via the intranet with the browser</td>
</tr>
<tr>
<td>CRN-2</td>
<td>The system must provide an API for interoperation with other applications.</td>
</tr>
<tr>
<td>CRN-3</td>
<td>Harness the programming expertise of the development team who is familiar with the DotNet platform.</td>
</tr>
<tr>
<td>CRN-4</td>
<td>The repository is a relational database installed on the SQL Server.</td>
</tr>
</tbody>
</table>

### 4.3 Attribute-driven design

#### 4.3.1 First iteration

Step 1: The ADD method starts with the part of the system that has not been designed yet. In this case, the “element” is the system as a whole and the first step is to establish the overall architecture of the solution.

Step 2: The choice of elements is influenced by the type of metadata architecture, in this case the centralized one, since it is independent of the source systems, favoring the availability and allowing the creation of new data attributes, which favors the modifiability of the system. The initial step was to decompose the system, identifying the elements according to the functionality of the system and the processes it supports.

Step 3: Create the initial decomposition of system.

Step 4: There were no quality requirements to satisfy at this early stage.

#### 4.3.2 Second iteration

Step 1: Producer. This element is responsible for capturing and submitting data from source systems (UC-1, UC-2).

Step 2: The chosen ASRs are AQM-1 and AQU-1.

To achieve the quality of the AQM-1 was used well-known established patterns, such as the layered architectural pattern, as a starting point for instantiating the architecture, allowing programmers to work on tasks relatively autonomously (Figure 3).

To achieve AQU-1, the methods created mirror the actions that DBAs are used to, such as defining the connection to the database instances or applying data selection criteria.

In the implementation of FR1 (functional requirement 1), it was considered that these systems manage data through well-defined structures such as tables and columns that are not changeable (Figure 2). Therefore, the business logic layer has been organized into table modules representing these information structures, and, therefore, only the implementation needs to be changed (Figure 1).

Step 3: Generate design solution

![Figure 1 – Data access module usage view](image)
Step 4: The scenarios AQM-1 and AQU-1 were satisfied.

4.3.3 Third iteration

Step 1: Repository. This element is responsible for storing metadata in a persistent medium.

Step 2: The chosen ASRs are AQM-1 and AQU-1

Step 3: The segregation of development, quality and production environments requires the ability to change the names of database connections. One way to do this is through a configuration file that exposes the repository link variable. This way, when you boot up, you can know where the repository is. However, in order to be able to “talk”, there must be an agreement between the data representation and the semantic accuracy of the elements exchanged, that is, the compatibility between the data models must be assured.

The mapping between data sent by the producer and the repository is almost straightforward. The repository data model has structures that represent the technical metadata (DBMS) and structures for describing the data sets that these structures store (business metadata).

The starting point of the model consists of classes that represent the DBMS:

- Relational Database, Relational Table, and Relational Column describing the structures of a Relational Server.
- Value Domain describes the content, form, and structure of relational column data. If enumerated, the values are broken down through the Permissible Value.

The data model of the repository was inspired by the Data Catalog (DCAT) vocabulary, which despite the language difference, may be a solution to the problem identified in this work, namely the (semantic) use of classes and properties (Figure 4), such as:

- Catalog represents a data catalog, where each record describes a collection of datasets (e.g., a record that associates extractions provided by a system or a record about DWH);
- Dataset represents a data set. A data set is a “collection of data, published by a single agent and available for access in one or more formats” (e.g., elements of the IN environment: customers, products, services, documents);
- Distribution represents how you can access the dataset, such as a relational table or file. It is the physical manifestation of the dataset through a type of medium;
- Agent represents a person or entity related to the catalog and/or datasets through a role;
- Role represents the role an agent can play in datasets (e.g., creator, owner, other...

---

1 <https://w3.org/TR/2014/REC-vocab-dcat-20140116/>
Step 4: The scenarios AQI-1 and AQM-2 were satisfied.

5 CURRENT STATE

The vision for sharing information about the data sets available in the BI environment is based on four key components: the data producer, the centralized repository, the portal for research and exploration, and an API for reading metadata.

One scenario related to the easiness of adding a metadata source, a scenario of interest to development team, which daily lives with various data sets stored in the DBMS. The aim was to provide the team with a tool that would capture the SQL Server metadata initially and be able to later include other database management systems.

This was done on the one hand, by reusing the business module implementation for SQL Server, on the other hand, through the data layer/infrastructure override mechanism that can be activated at run time.

We ask to two developers to interact with the tool without explaining how it works. They answer to questionnaire based on the Technological Acceptance
Model [9]. The evaluation was very positive, although these results have to be interpreted with a series of constraints, namely, the small sample size, the close working relationship and the common influences on the use of tools of this nature.

6 CONCLUSIONS

This research aimed to show the design and implementation of a metadata architecture motivated by the information needs of user groups participating in the process of creating BI applications.

We have tried to understand which user groups interact in a typical information-consuming environment, what tasks they perform, and which interests could benefit from the existence of a metadata repository, even before we begin designing the architecture. We have identified and reported some issues that hinder more fluid collaboration between teams.

In architectural design, the attribute-based architectural design method was used. Functional requirements, desired quality attributes and constraints allowed the early design phase of the architecture to choose tactics and standards to meet some of the requirements. This method proved to be effective in that it was able to break down the system into the main elements and achieve the desired qualities such as modifiability, safety and interoperability in the design of some of these elements.

We were able to use open source technology for develop a tool3 that allows the technical team to start producing data.

The modular form gives the ability to evolve the application and, as such, will allow extending the life cycle of the system.

While it is true that the centralized repository allowed us to choose the metadata we wanted, the effort devoted to create bridges to the metadata sources need to be considered.

We illustrate how metadata integration can occur by defining and mapping the data producer DBMS format and the repository data model. In this way, the technical and business metadata were able to coexist and interconnect, enhancing the communication between the teams.

It’s expected that this metadata will bring some benefits to the processes of creating BI applications because it allows reducing the analysis time of data that can be considered suitable for use by the areas. It will be easier to understand the data and, if necessary, to know who to talk to for further clarification.

6.1 Future work

Although research has answered the questions we have raised, it has also provided for two new learning cycles.

The first relates to the implementation of the remaining use cases. For example, you will need to choose the approach to data model creation: using object-oriented programming techniques or creating tables in the database, or, if the front-web will communicate with repository via Web API or database.

The second relates to the definition of the roles, responsibilities and users involved in repository data management and the necessary authentication and authorization and change tracking mechanisms to be present to safeguard the data.

7 REFERENCES


3 <https://tinyurl.com/u5eqfp3>
