UNIVERSITÀ POLITECNICA DELLE MARCHE
FACULTY OF ENGINEERING
Department of Information Engineering
Master in Computer and Automation Engineering

NEW GENERATION DATABASES EXAMINATION PROJECT

Design and implementation of a MongoDB database for the management of the State Omero Museum

Authors
Valerio Morelli
Federico Staffolani
Enrico Maria Sardellini

ACADEMIC YEAR 2024-2025

Contents

List of Figures

List of Tables

CHAPTER 1

The Omero Tactile Museum

This chapter provides an overview of the Omero State Museum. We believe it is important to get acquainted with the internal activities before exploring the NoSQL transition. Furthermore, the reasons for such a transition are explained.

The Omero State Tactile Museum in Ancona is a pioneer in the field of tactile art and a model of accessibility and inclusion for other cultural institutions.

1.1 Foundation

The Omero State Tactile Museum in Ancona is a distinguished institution within the small community of tactile museums worldwide. It was founded by Aldo Grassini and his wife Daniela Bottegoni, a blind couple with a passion for art, with the aim of fostering inclusivity for individuals with visual impairments by bringing them together with others and allowing them to experience the visit equally. In one of his interviews, Grassini stated:

> My wife and I, as avid travellers, have often had to contend with the absurd prohibitions that are placed in all museums: 'You can't touch!', which referring to a blind person is like telling a sighted person he cannot look. Out of this exasperation came my wife's idea, to collect in one place reproductions of the great masterpieces of art, so that even the blind could get to know them and enjoy the beauty of the masterpieces of human genius [NetworkMuseum, 2017].

The project immediately found support from the municipality of Ancona and was established on May 29, 1993, with the contribution of the Marche Region. On November 25, 1999, it was unanimously recognized as a state museum by the Italian Parliament, confirming its unique international significance. Under the agreement signed on August 3, 2001, the museum is managed by the City of Ancona in collaboration with the Ministry of Culture.

1.1.1 The Omero Museum today

The museum has a collection of thoudands of works, including sculptures, paintings, reliefs, and tactile reproductions of famous artworks [MinistryOfCulture, 2016]. It also organizes exhibitions, workshops, conferences, and educational activities for visitors of all ages and abilities.

The exhibition route is structured into rooms arranged in chronological order. It begins with the Greco-Roman room, followed by the Medieval/Renaissance room, and ends in the Contemporary room, which showcases numerous original pieces. All the pieces displayed in the museum's rooms, which visitors can admire during their visits, are provided with comprehensive descriptive information in Braille, as well as in large print for those with low vision. Visitors can use the app to track their route and access extra information about the artworks.

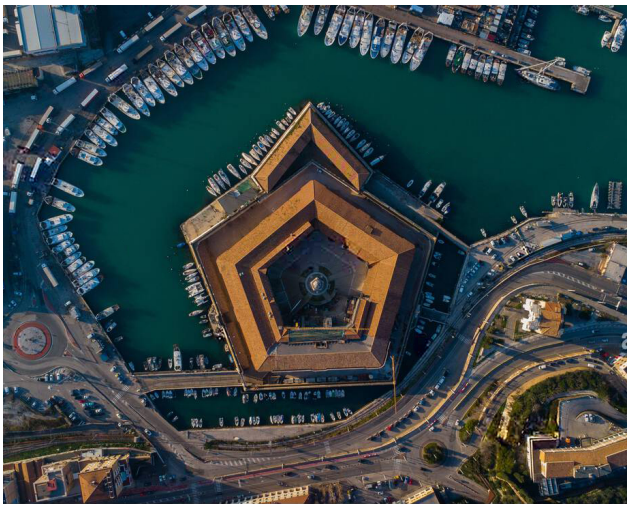


Figure 1.1: The Mole Vanvitelliana where the museum is located today

1.2 Internal organization

As a museum, most customer interactions, such as ticket purchases, take place in person at the physical location. Nevertheless, the current management system provides access to services and information via a website and a mobile app. Staff also have access to dedicated software connected to the same centralized *RDBMS*.

However, each department has its own authority credentials and set of permitted operations through the application. These align with their responsibilities, as shown in Figure 1.2.

1.2.1 Services offered

The museum offers clients three different services:

- Guided and unguided tours for a fixed collection of works;
- Guided and unguided tours for occasionally organized exhibitions;
- Organized laboratory activities for groups of people with a minimum number of participants.

Figure 1.3 depicts the prices of these three types of service.

1.2.2 How a client visit takes place

Upon entering the museum, visitors are greeted at the reception area where they can purchase and validate tickets, an example of which is shown in Figure 1.4. Generally, admission

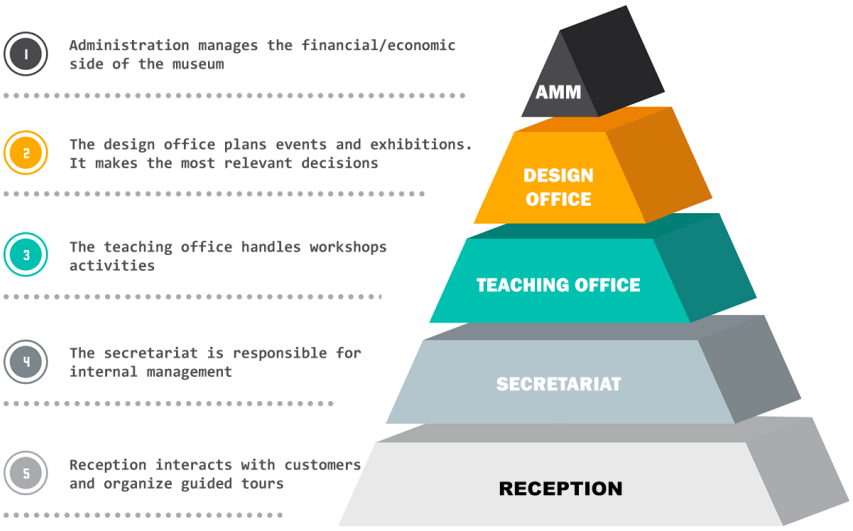


Figure 1.2: The internal organization of the Omero museum. Different departments have distinct duties.

to the museum is free for both individuals and families, with the exception of exhibitions and laboratory activities that may have fees depending on the type of event. At the end of their visit, visitors are encouraged to complete a survey, either on paper or via the mobile app. This survey collects anonymous data about the quality of their experience for statistical purposes.

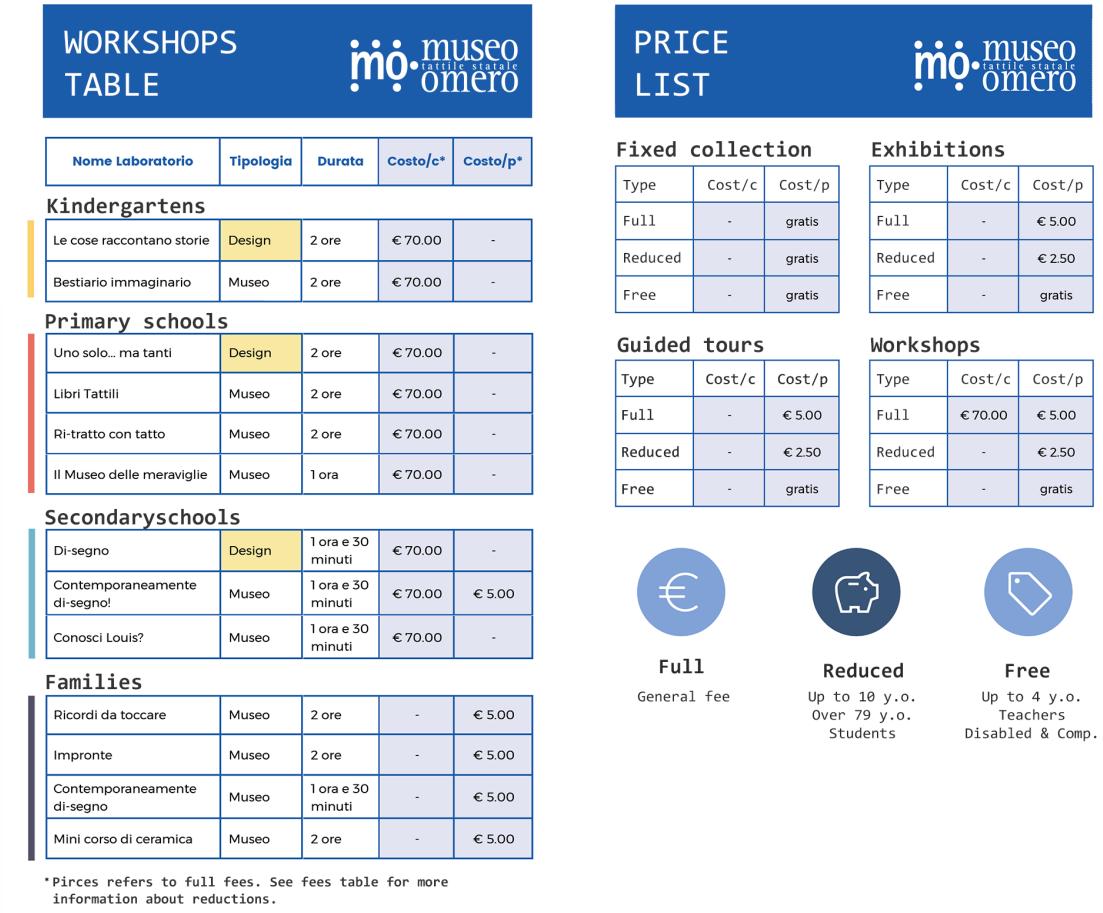
As stated in Section 1.2.1, clients can also request a guided tour, during which they can learn about the exhibited works from a knowledgeable guide and ask questions. Moreover they have the option to make advance bookings to secure a spot for their visit. This can be done by telephone, or through the mobile app. Reservations are required for laboratory as the museum needs to ascertain the number of participants in order to properly organize the event.

To facilitate the requirements collection and analysis, it is beneficial to visualize the sequence of activities involved in the typical interactions between customers and the museum. Figure 1.5 presents a *UML* activity diagram illustrating the key steps that a client must undertake to either purchase a ticket on-site or make a reservation through the website.

1.2.3 The current museum's management software

The museum recently implemented a digitalization project that provided both a desktop app for staff management, in Figure 1.7, and a mobile app for clients, in Figure 1.6 [Morelli, 2023b]. Although it primarily caters to the blind, it is essential to acknowledge that the majority of its customers consists of individuals without significant visual impairments. Moreover, the website provides visitors with easy access to all information pertaining to the museum's artworks and past exhibitions, allowing them to conveniently explore the content at their own pace.

As concerning the data management system, the museum currently relies on a *MySQL* database management system provided by the *WordPress CMS* to store all its data. This data is updated through an interface that executes low-level *MySQL* queries and is used to display information about artworks, rooms and exhibitions on the website.



WORKSHOPS TABLE

museo tattile statale omero

Nome Laboratorio	Tipologia	Durata	Costo/c*	Costo/p*
Kindergartens				
Le cose raccontano storie	Design	2 ore	€ 70.00	-
Bestiario immaginario	Museo	2 ore	€ 70.00	-
Primary schools				
Uno solo... ma tanti	Design	2 ore	€ 70.00	-
Libri Tattili	Museo	2 ore	€ 70.00	-
Ri-tratto con tatto	Museo	2 ore	€ 70.00	-
Il Museo delle meraviglie	Museo	1 ora	€ 70.00	-
Secondaryschools				
Di-segno	Design	1 ora e 30 minuti	€ 70.00	-
Contemporaneamente di-segno!	Museo	1 ora e 30 minuti	€ 70.00	€ 5.00
Conosci Louis?	Museo	1 ora e 30 minuti	€ 70.00	-
Families				
Ricordi da toccare	Museo	2 ore	-	€ 5.00
Impronte	Museo	2 ore	-	€ 5.00
Contemporaneamente di-segno	Museo	1 ora e 30 minuti	-	€ 5.00
Mini corso di ceramica	Museo	2 ore	-	€ 5.00

*Pirces refers to full fees. See fees table for more information about reductions.

PRICE LIST

museo tattile statale omero

Fixed collection

Type	Cost/c	Cost/p
Full	-	gratis
Reduced	-	gratis
Free	-	gratis

Exhibitions

Type	Cost/c	Cost/p
Full	-	€ 5.00
Reduced	-	€ 2.50
Free	-	gratis

Guided tours

Type	Cost/c	Cost/p
Full	-	€ 5.00
Reduced	-	€ 2.50
Free	-	gratis

Workshops

Type	Cost/c	Cost/p
Full	€ 70.00	€ 5.00
Reduced	-	€ 2.50
Free	-	gratis

Full
General fee

Reduced
Up to 10 y.o.
Over 79 y.o.
Students

Free
Up to 4 y.o.
Teachers
Disabled & Comp.

Figure 1.3: The fees of the musum's services. As a museum that fosters accessibility for people with impairments, disabled visitors can always benefit from the free tariff.

1.3 The need for a NoSQL transition

This technological revolution has led to an increase in visitors in recent years. In response to this interest, the museum has organized more appealing events, such as *last-minute exhibitions*, *artist invitations*, and *temporary artwork exchanges* with other museums nationwide. Furthermore, the increased diversity of artworks in the catalog has created the need for a more agile data storage solution. Drawings, statues, digital media, archaeological finds, interactive installations, illustrations, and photographs are just a few of the kind of art the museum hosts today. Each one of them has its own features and often exceed the fixed boundaries of the relational schema. In short, **frequent schema evolution fosters flexibility**. In turn, flexibility requires the data model to adapt without downtime. Costly `ALTER TABLE` SQL and the like operations are out of the question.

In addition to flexibility, there has been an increasing need for scalability. Exceptional, time-limited events, such as auctions, which are hosted on the website and on the mobile app, have proven to attract large audiences. Even though temporarily, the museum's management system is overwhelmed with thousands of requests per second, which is hundreds of times higher than everyday traffic. **A horizontally scalable solution would enable the system to absorb sudden spikes in load** without hindering the user experience during these occasional events.

Figure 1.4: The access ticket of the museum

Furthermore, the inherent redundancy and eventual consistency in NoSQL systems caused by update anomalies are not vital concerns for the museum's use case. **Slight delays in propagating updates across replicas are acceptable** because most schema changes are non-critical and converge quickly.

Last but not least, the administration continuously subjects the huge amount of collected data–tens millions of records–to statistical processes. The data is integrated with affiliated museum and analyzed offline to extract knowledge about client behavior and growing trends. This analysis **often involves subsets of fields in the records** and can be optimized using ad hoc NoSQL solutions. Furthermore, many of the museum's entities, such as artworks, **map naturally into nested objects**. This eliminates the need for many association reification tables and the expensive JOIN operations that follow.

To address these issues, the current relational implementation [Morelli, 2023a] must be upgraded to a more flexible NoSQL solution.

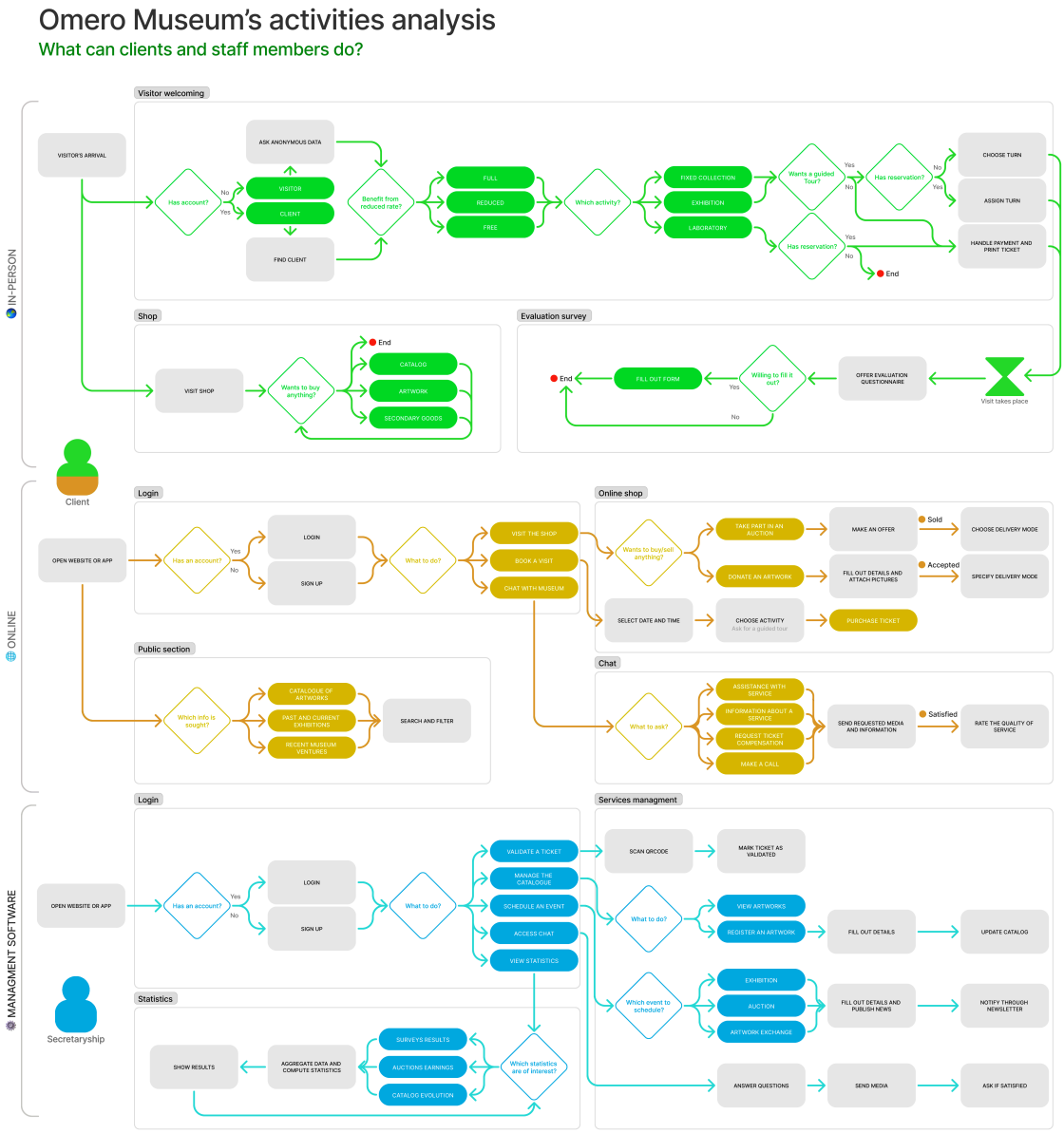


Figure 1.5: The activity diagram depicting the museum's internal activities

Figure 1.6: The mobile app of the Omero museum [Morelli, 2023b]. It offers customers several services, such as ticket purchasing, surveys and catalog browsing.

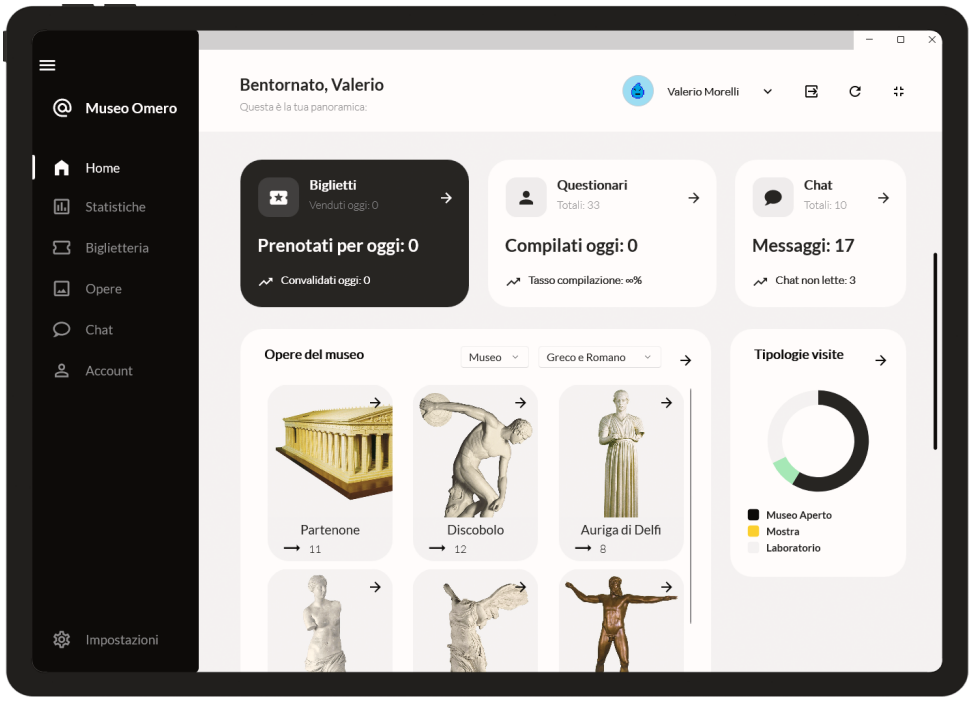


Figure 1.7: The management software for staff members [Morelli, 2023b]. Employees can use it to manage the catalog, generate statistics, validate tickets and chat with customers.

CHAPTER 2

Requirements analysis

The informal requirements provided in the previous chapter need to be refined towards the design of a NoSQL solution. Follows in this chapter, a list of the most frequent operations that the application needs to perform. These, together with the revised conceptual diagram, will drive the logic design of the new database.

2.1 The conceptual model

The museum currently uses a relational database. As such, it was designed using the standard procedure outlined in the literature [Atzeni *et al.*, 2023]. Interested readers can find more information about the decisions that led to the development of the final logical schema in the dedicated documentation [Morelli, 2023a].

Figure 2.1 proposes a redesign of the conceptual schema to accommodate new concepts, such as auctions, unstructured chat media contents, conferences and affiliated museum's data.

Along with the list of the most frequent operations of interest to the mobile and desktop applications, presented in Section 2.4, this diagram serves as the starting point for the *NoSQL* logical design. For the sake of clarity, the ER diagram is colored to show five distinct islands, each representing a specific subdomain of the museum.

2.2 Formal requirements

This section will list the requirements that the new database should meet.

2.2.1 Functional requirements

For the sake of readability, the functional requirements were subdivided according to the color of their respective islands in the entity-relationship diagram shown in Figure 2.1.

Artwork island

The artworks form the core of the museum's collection. New artworks are continuously stored, with some being purchased, some gifted and some borrowed from and loaned to other museums. Artworks are grouped together to create exhibitions, and may be sold at auction.
Although this trade is handled internally, customers can still consult the online catalog to learn its details, such as the author, donor, or seller, the type, the historical period, the date of construction, whether it is authentic, the date of transfer to the museum, the size and scale, if applicable, the room and floor on which it is exhibited, the materials, the techniques used for modeling, and any associated compositions or productions.

Staff Island

The roles of museum employees include administrators, designers, teaching staff, secretaries, and guides. Each one of them has its own duties and responsibilities.
Guides interact with visitors at the reception desk, lead guided tours, and respond to calls and assistance requests via chat. Secretaries manage internal staff and bureaucratic issues. Teaching staff and designers schedule events and strive to bring novelty to the museum's offerings. Lastly, administrators review statistics to develop long-term strategies.
The system stores details about all of the museum's employees, including their past and current roles, shifts, and departments.

Activities island

The offered services include visits to the permanent collection and exhibitions, as well as participation in conferences and workshops. It is preferable to book a guided tour or exhibition visit in advance. This is mandatory for workshops and conferences.
While workshops and guided tours are organized upon request, exhibitions and conferences are scheduled by the museum and have a limited number of participants.

Customer island

Visitors are the museum's primary source of income. As a museum that fosters accessibility, it would be unbecoming to require registration to visit. Thus, visitors may or may not have a digital account when requesting a visit. At the end of the visit, however, all visitors are asked to fill out an optional survey to evaluate their experience. This anonymous data serves as a valuable source of statistics for the administrative staff to monitor the museum's appeal.
Once they have signed up for an account, customers can access additional online services and the system stores their biographical data. They can buy, give, or bid on artwork.
Auctions are held for specific artworks or items. For each auction, the base price, increment rules, and start and end dates are stored. Bids placed by participants are also tracked, including the bid amount and date. The winning bid and related customer are also tracked. In addition to being sold at auction, artworks may be sold directly to customers or received as donations. In the latter case, the donation and processing dates, as well as whether the artwork was accepted into the museum's collection, are recorded.
In addition to regular customers, the museum interacts with private sellers and affiliated museums. Artworks can be borrowed from affiliated museums or purchased from trusted

suppliers.

Chat island

Customers can request assistance via the chat feature on the mobile app. When free from tours, it is the duty of guides to answer requests for assistance through their dedicated desktop software.
The mobile app allows users to send media files along with text messages. For example, this can be useful for scanning a ticket or capturing a problematic application page.

2.2.2 Non-functional requirements

In light of the discussion provided in Section 1.3, the non-functional requirements differ significantly from the design of the current relational DBMS. The new needs can be summarized as follows:

- *Non-structured* digital content, such as images, audio, and video, can no longer be considered a small niche of the database. Therefore, it is mandatory for the system to store them alongside structured content.
- Exhibitions and conferences have a limited number of participants. As a matter of fact, it is common to register system demand spikes coming up to the scheduled date. Therefore, it is essential to ensure both *horizontal scalability* and *fair processing* of requests.
- Support for silent upgrades that do not hinder the ongoing visitor experience is sought to foster *maintainability*.
- The museum must preserve the huge amount of data collected over the years because it has great value. Backups and recovery protocols are necessary to ensure its *persistence*.
- The administration frequently runs statistics to discover interest trends. The system must be able to perform *large-scale statistical analyses* on tens of millions of records.

2.3 Volume tables

Volume tables offer a quantitative overview of the estimated number of entries for each entity and relationship. They are shown in Tables 2.1 and 2.2 respectively. Compared to the previous, several-years-old database design ([Morelli, 2023a]), the amount of data has notably increased.

For easier reading, the tables have again been grouped and colored according to the islands in the ER diagram in Figure 2.1.

For clarity, it is important to note that all percentages and averages in the volume tables were calculated using hypothetical estimates that are as logically consistent as possible. Where clear proportionality was not possible, for example, with rooms, which are fixed in number and do not require mathematical justification, the comments field has been left blank.

2.4 Operations analysis

The data modeling design proposed in the following chapter must be tailored to the operations that the mobile and desktop applications define.

Entity	Volume	Comment
ARTWORK	10,000	–
AUTHOR	8,000	1.25 artworks for each author on average
MATERIAL	50	–
ROOM	6	–
TECHNIQUE	20	–
ADMINISTRATOR	5	–
DEPARTMENT	6	–
DESIGNER	5	–
EMPLOYEE	55	Sum of all staff volumes
GUIDE	15	About 30% of employees act as guides
ROLE	5	–
SECRETARY	10	–
SHIFT	3	–
TEACHING STAFF	10	–
ACTIVITY	25,000	Sum of workshops and guided tours
CONFERENCE	2,000	–
EVENT	32,000	Sum of activities and limited events
EXHIBITION	5,000	–
GUIDED TOUR	20,000	About 80% of all activities
LIMITED EVENT	7,000	Sum of exhibitions and conferences
REAL WORKSHOP	5,000	About 20% of all activities
WORKSHOP	15	–
AFFILIATED MUSEUM	10	–
AUCTIONS	4,000	Around 40% of total trades
CUSTOMER	100,000	One in 5 museum's visitors registers
SALE	6,000	Around 60% of total trades
SELLER	2,900	Almost all suppliers participate in trades
SUPPLIER	3,000	–
SURVEY	2,250,000	About 1.5 surveys per visitor
TICKET	800,000	About 1.5 tickets per visitor of this, and not other affiliated museum, on average
TRADE	10,000	Includes both sales and auctions
VISITOR	1,500,000	500,000 from this museum and 1,000,000 from affiliated museums.
AUDIO	30,000	–
CHAT	50,000	About half of all customers open a chat
MESSAGE	300,000	About 6 messages per chat on average
PICTURE	60,000	–
TEXT	210,000	–

Table 2.1: Entity volume table

The operations are listed in Tables 2.3 based on the formal requirements outlined above. The following considerations were made to create the table.

Relationship	Volume	Comment
COMPOSITION	15,000	About 2 materials per artwork on average
CREATION	20,000	About 2 authors per artwork on average
LOCATION	100	–
PRODUCTION	20,000	About 2 techniques per artwork on average
ASSIGNMENT	50	One assignment per employee
CONTRACT	1,000	–
LEADER	30,000	About 1.2 leaders per activity on average
PLANNER	30	About 2 planners per workshop on average
SCHEDULER	5,000	About 100 shifts per employee on average
EXHIBIT	20,000	About 2 exhibitions per artwork on average
REFERENCE	5,000	One reference per workshop
SPEAKER	4,000	About 2 speakers per conference on average
BID	40,000	About 10 bids per auction on average
COMMENT	150,000	Each artwork has 15 comments in average
DONATION	5,000	About 5% of customers make donations
FILL OUT	6,250,000	–
GOOD	10,000	One artwork per trade
PARTECIPATION	200,000	About 40% of this museum's visitors attend events
PURCHASE	800,000	About 1.6 tickets per this museum's visitor on average
SELL	2,000	–
STATISTICS	4,000,000	–
WIN	3,500	About 87.5% of auctions result in a win, the rest remain unsold
CONVERSATION	50,000	One conversation per chat
HELP	50,000	One help request per chat
SUPPORT	50,000	One support request per chat

Table 2.2: Relationship volume table

1. **According to the Pareto's law, the** 80% **of the workload is generated by the** 20% **of the operations** [Atzeni *et al.*, 2023]. Thus, for brevity, we will consider only the most frequent operations.

2. Moreover, we assume that the two **applications implement a caching system** with appropriate Time-To-Live (TTL) parameters. Thus, multiple queries are not made to the DBMS when data is not expected to change.

Lastly, readers should note that the superscripts (c) and (s) in the frequency column distinguish between the operations invoked by the customer and staff apps, respectively.

Operation	Description	Type	Frequency (per day)
CR1	Create a new comment to a specific artwork	I	$1,000^{(c)} + 0^{(s)} = 1,000$
CR2	Create a new survey compilation	I	$250^{(c)} + 2,000^{(s)} = 2,250$
CR3	Create a new ticket for a visitor	I	$3,000^{(c)} + 2,000^{(s)} = 5,000$
CR4	Create a new message	I	$2,500^{(c)} + 2,500^{(s)} = 5,000$
CR5	Create a new trade	I	$0^{(c)} + 10^{(s)} = 10$
CR6	Create a new visitor	I	$3000^{(c)} + 2000^{(s)} = 5000$
CR7	Create a new activity	I	$0^{(c)} + 1^{(s)} = 1$
CR8	Create a new artwork	I	$0^{(c)} + 5^{(s)} = 5$
CR9	Create a new limited event	I	$0^{(c)} + 1^{(s)} = 1$
CR10	Create a new employee	I	$0^{(c)} + 1^{(s)} = 1$
CR11	Create a new shift	I	$0^{(c)} + 50^{(s)} = 50$
CR12	Create a new author	I	$0^{(c)} + 3^{(s)} = 3$
CR13	Create a new workshop	I	$0^{(c)} + 1^{(s)} = 1$
CR14	Create a new supplier	I	$0^{(c)} + 5^{(s)} = 5$
RD1	Read all the information of a given artwork, including its authors, materials, crafting techniques and room placement	B	$10,000^{(c)} + 2,500^{(s)} = 12,500$
RD2	Read the last 10 comments of a given artwork	B	$10,000^{(c)} + 2,500^{(s)} = 12,500$
RD3	Calculate the average rating of a given artwork	B	$10,000^{(c)} + 2,500^{(s)} = 12,500$
RD4	Calculate the average rating of the comments posted by a given customer	B	$500^{(c)} + 1,000^{(s)} = 1,500$
RD5	Read all the comments of a given artwork with a given rating	B	$3,500^{(c)} + 1,500^{(s)} = 5,000$
RD6	Read all the surveys, relative to this museum, filled out in a specific year	B	$0^{(c)} + 10^{(s)} = 10$
RD7	Read all the surveys filled out by a specific customer	B	$500^{(c)} + 250^{(s)} = 750$
RD8	Read all the surveys from a specific affiliated museum	B	$0^{(c)} + 20^{(s)} = 20$
RD9	Read all tickets of a visitor	B	$10,000^{(c)} + 5,000^{(s)} = 15,000$
RD10	Read all tickets for an event	B	$0^{(c)} + 40^{(s)} = 40$
RD11	Count tickets issued in a year	B	$0^{(c)} + 1^{(s)} = 1$
RD12	Read 10 messages of a given chat previous than a specific date	B	$2,500^{(c)} + 2,500^{(s)} = 5,000$
RD13	Read active chats for a guide	B	$0^{(c)} + 200^{(s)} = 200$
RD14	Count chats opened in a year	B	$0^{(c)} + 1^{(s)} = 1$
RD15	Retrieve the artwork of a trade	B	$450^{(c)} + 50^{(s)} = 500$
RD16	Count trades in a given year	B	$0^{(c)} + 1^{(s)} = 1$
RD17	Retrieve the buyer of a given artwork	B	$350^{(c)} + 150^{(s)} = 500$
RD18	Retrieve all the artworks of a given author	B	$0^{(c)} + 10^{(s)} = 10$

RD19	Retrieve all activity of a given workshop type	B	$90^{(c)} + 10^{(s)} = 100$
RD20	Read all the surveys filled out in a specific day	B	$250^{(c)} + 50^{(s)} = 300$
RD21	Retrieve information about a given laboratory activity, including the total number of reservations	B	$350^{(c)} + 150^{(s)} = 500$
RD22	Retrieve all the information about a limited event	B	$350^{(c)} + 150^{(s)} = 500$
RD23	Retrieve all the details of a visitor	B	$3000^{(c)} + 2000^{(s)} = 5000$
RD24	Retrieve the visitor count for each land from most frequent to least frequent	B	$0^{(c)} + 1^{(s)} = 1$
RD25	Retrieve the visitor count for each impairment from most frequent to least frequent	B	$0^{(c)} + 1^{(s)} = 1$
RD26	Count tickets issued for each year	B	$0^{(c)} + 1^{(s)} = 1$
RD27	Calculate the distribution of visitor ages grouped into bands based on ticket activity in a specific year	B	$0^{(c)} + 1^{(s)} = 1$
RD28	Count material usage in all artworks	B	$250^{(c)} + 0^{(s)} = 250$
RD29	Count top 5 visit reasons	B	$0^{(c)} + 1^{(s)} = 1$
RD30	Count how many artworks are original vs. not	B	$250^{(c)} + 0^{(s)} = 250$
RD31	Count how many visitors came back	B	$0^{(c)} + 1^{(s)} = 1$
RD32	Count top 5 days with most ticket activity in a given year	B	$0^{(c)} + 1^{(s)} = 1$
RD33	Average enrollment rate by workshop title	B	$0^{(c)} + 10^{(s)} = 10$
RD34	Count visitor education level breakdown from surveys in a given period	B	$0^{(c)} + 1^{(s)} = 1$
RD35	Show workshop duration distribution	B	$0^{(c)} + 10^{(s)} = 10$
RD36	Count the number of museums vs external suppliers	B	$0^{(c)} + 10^{(s)} = 10$
UP1	Update the description of a material	I	$0^{(c)} + 1^{(s)} = 1$
UP2	Check-in a ticket	I	$0^{(c)} + 5,000^{(s)} = 5,000$
UP3	Update trade information of an artwork	I	$0^{(c)} + 10^{(s)} = 10$
UP4	Update information of an artwork	I	$0^{(c)} + 10^{(s)} = 10$
UP5	Update curriculum of an employee	I	$0^{(c)} + 1^{(s)} = 1$
UP6	Update employee's department	I	$0^{(c)} + 5^{(s)} = 5$
UP7	Update the shift of an given employee	I	$0^{(c)} + 10^{(s)} = 10$
UP8	Change the date of a limited event	I	$0^{(c)} + 1^{(s)} = 1$
UP9	Update a message in a given chat	I	$50^{(c)} + 50^{(s)} = 100$
UP10	Update prices of a workshop	I	$0^{(c)} + 1^{(s)} = 1$
DL1	Delete a posted comment	I	$150^{(c)} + 350^{(s)} = 500$
DL2	Delete an artwork	I	$0^{(c)} + 1^{(s)} = 1$
DL3	Delete a shift of an employee	I	$0^{(c)} + 10^{(s)} = 10$
DL4	Delete a message	I	$30^{(c)} + 20^{(s)} = 50$

Table 2.3: The table of operations

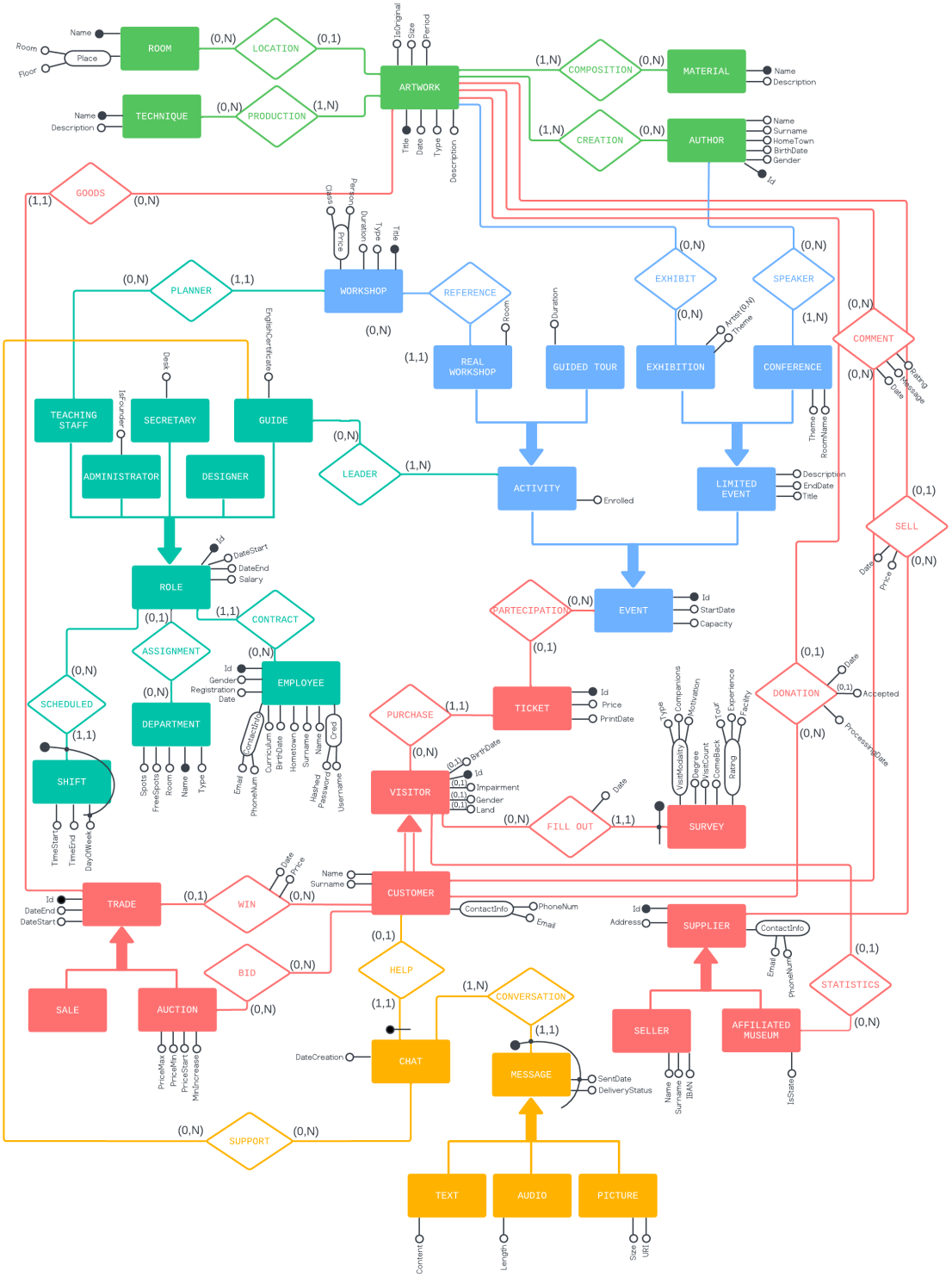


Figure 2.1: The entity-relationship conceptual diagram for the current museum's relational database design [Morelli, 2023a]. (*Adapted*)

CHAPTER 3

The logical and physical design

This chapter discusses the evolution of the ER diagram towards the logic model. During this design phase, choices are made with the aim of optimizing database performance by reducing accesses without denormalizing the data too much.

This analysis involves making several physical-level choices. Tree and hash primary and secondary indexes are defined on specific fields to enable faster access, and eventual consistency is deemed acceptable. The result of this analysis is the NoAM representation, which acts as an abstraction between the NoSQL families.

3.1 Removal of hierarchies

The logic design phase begins with the removal of the hierarchies from the ER diagram presented previously in Figure 2.1. When operations make little distinction in terms of access, one can (i) *unify children in the root entity*. Conversely, when operations treat each child differently, one can do (ii) *the opposite*. Furthermore, one can (iii) *convert the hierarchy into a relationship* or adopt a (iv) *hybrid approach* [Atzeni *et al.*, 2023].

In this case, the high flexibility–result of the *schemaless* property of NoSQL solutions–make it convenient to follow approach (i). This approach requires providing a new "type" attribute on the root entity to distinguish between instances. Additionally, this approach *reduces the number of accesses* at the expense of larger entities. The only exception are the LIMITED EVENT and ACTIVITY entities, which aren't unified with the parent EVENT, but rather the opposite (ii), because of the fundamental different nature of the entities. Moreover, CLIENT and VISITOR are kept isolated with a relationships. The resulting ER diagram in shown in Figure 3.1.

Other than hierarchies removal, the following considerations have been made:

1. Strong entities are highlighted in bold and shown in Figure 3.2. In accordance with the Single Responsibility principle, the first of the SOLID principles, these entities fulfill one specific and distinct duty within the schema and cannot be incorporated into other entities.

2. The COMMENT relationship was reified into a standalone entity. This is because it was a many-to-many relationship with several attributes of its own.

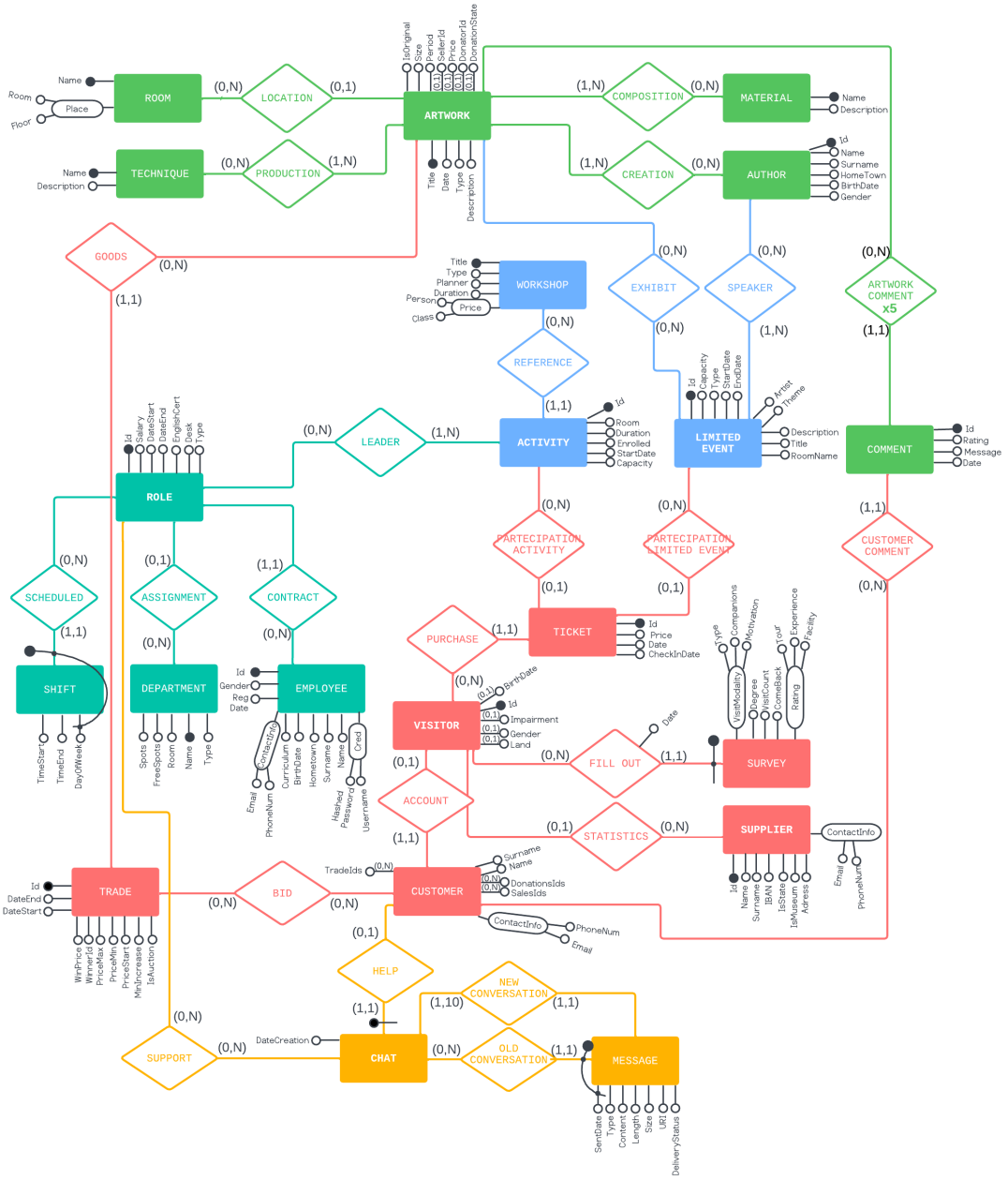


Figure 3.1: The entity-relationship conceptual diagram for the current museum's relational database design [Morelli, 2023a]. (*Hierarchies removed*)

3. The `SELL` and `DONATION` relationships have been removed, with their optional attributes being placed on the `ARTWORK` relationship instead. To make retrieval easier, redundant list attributes called `DonationsIds` and `SalesIds` were added to the `CUSTOMER` entity to facilitate the retrieval of donated and sold artworks. The same applies to the `WIN` relationship.

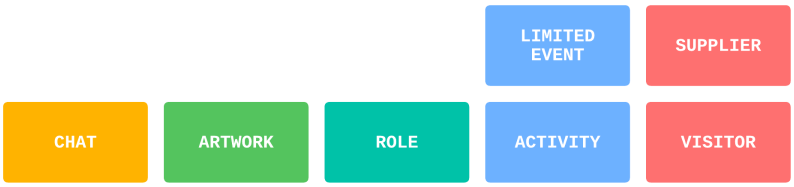


Figure 3.2: The strong entities in the database.

3.2 Horizontal partitioning

Boundless growing entities, i.e. those with more frequent *insertion* operations than *deletion* ones, risk dramatically increasing the size of the aggregates when embedded inside others. Section 3.3 provides a quantitative study of access patterns to determine whether entities should be embedded or not. However, these calculations do not take into account the size of the items. In physical terms, **if an item exceeds the size of a disk block, multiple read accesses are required to read it in full**. This might be the case for the following entities:

- MESSAGE: Chats can grow indefinitely. Although they are designed for occasional use when a problem arises with a museum service, some customers may have a long history of messages. The applications implement a *lazy* scrolling system which requests 10 messages at a time and most users do not look for old messages. Therefore, it would be wise to partition messages based on their insertion date. This means that, in the case of embedding, only the most recent partition is aggregated inside the chat item, with the others being referenced.

- COMMENT: Customers can comment on artworks. The distribution of comments is highly skewed in practice, with a small number of artworks receiving a large amount of comments. Both applications are interested in filtering comments by rating; therefore, having a rating-based partition surely helps. This type of partitioning does not allow for partial embedding because all comments carry equal importance in terms of rating. Therefore, if embedding is preferred, the size of the ARTWORK items isn't reduced.

- VISITOR, TICKET and SURVEY: They are all unbounded in growth and are the entities with the largest volume. However, it is difficult to select a horizontal partitioning strategy given the diversity of the operations around them. **A primary or secondary index, such as a B+-tree defined on the insertion date, can be used to optimize retrieval in ranged queries**.

- BID: Customers can bid multiple times in auctions that interest them. Therefore, the number of bids can grow very quickly, especially for short-lived auctions that are in high demand. However, we consider it excessive to partition bids due to the temporary nature of auctions.

3.3 Aggregates design

The goal of the ER reorganization phase is to reduce accesses. According to the operations listed previously (see Section 2.4), *entities should be unified or separated based on their attribute access patterns*. Therefore, we will consider several sections of the ER diagram and decide how to

translate them into the logical model while optimizing performance in terms of the required write and read accesses. For this analysis, we will refer to the operations (2.3) and volume tables (2.1 and 2.2) . Lastly, the strong entities discovered in Figure 3.2 won't be embedded inside any other entities.

The tables that follow use superscripts to refer to the accessed entity and subscripts to distinguish between write (W) and read access (R). Moreover, we make the assumption that $1_W \approx 2_R$, and that the volume refer to 10 years of activity with the current RDMBS.

MATERIAL and ARTWORK

The relationship between an ARTWORK and a MATERIAL is one-to-few. MATERIAL instances change infrequently and are bounded in growth. These properties hint for embedding materials inside artworks. In more accurate terms,

- The mean number of MATERIAL per ARTWORK is $\frac{20{,}000}{10{,}000} = 2$;
- The mean number of ARTWORK per MATERIAL is $\frac{20{,}000}{50} = 400$;

Operation	**Accesses**	**Freq. (day)**	**Total**
Embedding (MATERIAL in ARTWORK)			
RD1: Retrieve the materials	$1_R^{(a)} = 1$	12,500	12,500
UP1: Update a material's description	$1_W^{(a)} \times 400 = 800$	1	800
			13,300
Referencing (MATERIAL in ARTWORK)			
RD1: Retrieve the materials	$1_R^{(a)} + 2_R^{(m)*} = 3$	12,500	37,500
UP1: Update a material's description	$1_W^{(a)} = 2$	1	2
			37,502

(*) Plus $\lceil log_F(50) \rceil = 1$ buffer accesses per material, assuming an average fanout of $F = 100$.

The decision is to embed MATERIAL instances inside ARTWORK ones. The same exact reasoning applies to TECHNIQUE: ARTWORK should embed them both. Concerning ROOM and AUTHOR, we decided to leave them as standalones with references because ROOM is referred to by many entities, and AUTHOR has a many-to-many relationship with LIMITED EVENT.

COMMENT, ARTWORK and CUSTOMER

The relationship between an ARTWORK and a COMMENT is one-to-many, while the one between an CUSTOMER and a COMMENT is one-to-few. COMMENT instances are deleted infrequently and are unbounded in growth. These properties hint for embedding comments inside artworks. In more accurate terms,

- The mean number of COMMENT per ARTWORK is $\frac{150{,}000}{10{,}000} = 15$;
- The mean number of COMMENT per CUSTOMER is $\frac{150{,}000}{100{,}000} = 1.5$;

Operation	Accesses	Freq. (day)	Total
Embedding (COMMENT in ARTWORK)			
RD2: Read last 10 comments	$1_R^{(a)} = 1$	12,500	12,500
RD3: Calculate artwork avg rating	$1_R^{(a)} = 1$	12,500	12,500
RD4: Calculate customer avg rating	$10{,}000_R^{(a)} = 10{,}000$	1,500	15,000,000
RD5: Filter by rating	$1_R^{(a)} = 1$	5,000	5,000
CR1: Post comment	$1_W^{(a)} = 2$	1,000	2,000
DL1: Delete comment	$1_W^{(a)} = 2$	500	1,000
			15,033,000
Embedding (COMMENT in CUSTOMER)			
RD2: Read last 10 comments	$10_R^{(c)*} = 10$	12,500	1,250,000,000
RD3: Calculate artwork avg rating	$100{,}000_R^{(c)} = 100{,}000$	12,500	1,250,000,000
RD4: Calculate customer avg rating	$1_R^{(c)} = 1$	1,500	1,500
RD5: Filter by rating	$3_R^{(c)\dagger} = 3$	5,000	15,000
CR1: Post comment	$1_W^{(c)} = 2$	1,000	2,000
DL1: Delete comment	$1_W^{(c)} = 2$	500	1,000
			2,500,019,500
Embedding (COMMENT in ARTWORK) + **Referencing** (COMMENT in CUSTOMER)			
RD2: Read last 10 comments	$1_R^{(a)} = 1$	12,500	12,500
RD3: Calculate artwork avg rating	$1_R^{(a)} = 1$	12,500	12,500
RD4: Calculate customer avg rating	$1_R^{(c)} + 1.5_R^{(a)\ddagger} = 2.5$	1,500	3,750
RD5: Filter by rating	$1_R^{(a)} = 1$	5,000	5,000
CR1: Post comment	$1_W^{(a)} + 1_W^{(c)} = 4$	1,000	4,000

DL1: Delete comment	$1_W^{(a)} + 1_W^{(c)} = 4$	500	2,000
			39,750
Embedding (COMMENT in ARTWORK) **+ Referencing** (COMMENT in CUSTOMER) **+ Copy** (Rating in CUSTOMER)			
RD2: Read last 10 comments	$1_R^{(a)} = 1$	12,500	12,500
RD3: Calculate artwork avg rating	$1_R^{(a)} = 1$	12,500	12,500
RD4: Calculate customer avg rating	$1_R^{(c)} = 1$	1,500	1,500
RD5: Filter by rating	$1_R^{(a)} = 1$	5,000	5,000
CR1: Post comment	$1_W^{(a)} + 1_W^{(c)} = 4$	1,000	4,000
DL1: Delete comment	$1_W^{(a)} + 1_W^{(c)} = 4$	500	2,000
			36,500

(*) Assuming a secondary tree index is defined on the Date field of COMMENT.
(†) Assuming a secondary hash index is defined on the Rating field of COMMENT.
(‡) Plus $\lceil log_F(150,000) \rceil = 3$ buffer accesses per comment, assuming an average fanout of $F = 100$.

The decision is to embed COMMENT instances inside ARTWORK ones and to reference them inside CUSTOMER instances, by duplicating the Rating attribute together with the COMMENT's id.

SURVEY, VISITOR and AFFILIATED MUSEUM

The relationship between a VISITOR and a SURVEY is one-to-few, while the one between an AFFILIATED MUSEUM and a SURVEY is one-to-many. SURVEY instances are never edited nor deleted, can grow indefinitely and are accessed directly most of the times. These properties hint for avoiding embedding surveys inside customers, but may still be embedded inside affiliated museums. In more accurate terms,

- The mean number of SURVEY per VISITOR is $\frac{2,250,000}{1,500,000} = 1.5$;
- The mean number of VISITOR/SURVEY per AFFILIATED MUSUEM is $\frac{1,000,000}{10} = 100.000$;
- The mean number of museum's SURVEY per year is $\frac{1,250,000}{10} = 125.000$;

Operation	**Accesses**	**Freq. (day)**	**Total**
Embedding (SURVEY in VISITOR)			

RD6: Read museum surveys in a year	$125{,}000_R^{(s)*} = 125{,}000$	10	1,250,000
RD7: Read surveys of a customer	$1_R^{(c)} + 1_R^{(v)} = 2$	750	1,500
RD8: Read surveys of an affiliated museum	$1_R^{(a)} + 100{,}000_R^{(v)} = 100{,}001$	20	2,000,020
CR2: Create survey compilation	$1_W^{(v)} = 2$	2,250	4,500
			3,256,020
Referencing (SURVEY in VISITOR and in AFFILIATED MUSEUM)			
RD6: Read museum surveys in a year	$125{,}000_R^{(s)*} + 125{,}000_R^{(v)\dagger} = 250{,}000$	10	2,500,000
RD7: Read surveys of a customer	$1_R^{(c)} + 1_R^{(v)\dagger} + 1.5_R^{(s)\ddagger} = 3.5$	750	2,625
RD8: Read surveys of an affiliated museum	$1_R^{(a)} + 100{,}000_R^{(v)\dagger} + 100{,}000_R^{(s)\ddagger} = 200{,}001$	20	4,000,020
CR2: Create survey compilation	$1_W^{(v)} + 1_W^{(s)} = 4$	2,250	9,000
			6,502,645

(*) If a secondary tree index is defined on the `CompilationDate` field of SURVEY.

(†) Plus $\lceil log_F(1{,}500{,}000) \rceil = 4$ buffer accesses per visitor, assuming an average fanout of $F = 100$.

(‡) Plus $\lceil log_F(2{,}250{,}000) \rceil = 4$ buffer accesses per survey, assuming an average fanout of $F = 100$.

The decision is to embed SURVEY instances inside VISITOR. It should be noted that AFFILIATED MUSEUM access surveys through visitors.

TICKET and VISITOR

The relationship between an TICKET and a VISITOR is one-to-many. TICKET instances never change, except for check in, and are bounded in growth. These properties hint at embedding tickets inside visitors. In more accurate terms,

- The mean number of TICKET per VISITOR is $\frac{2{,}000{,}000}{1{,}500{,}000} = 1{,}33$;

Operation	**Accesses**	**Freq. (day)**	**Total**

Embedding (TICKET in VISITOR)			
RD9: Retrieve all tickets of a visitor	$1_R^{(v)} = 1$	15,000	15,000
CR3: Create a new ticket for a visitor	$1_W^{(v)} = 2$	5,000	10,000
RD10: Retrieve all tickets for an event	$1_R^{(v)} \times 50 = 50$	10	500
UP2: Check-in a ticket	$1_W^{(v)} = 2$	5,000	10,000
RD11: Count tickets issued in a year	$150,000_R^{(v)} = 150,000$	1	150,000
			185,500
Embedding (TICKET in VISITOR) + **Computed pattern** (on TICKET's count)			
RD9: Retrieve all tickets of a visitor	$1_R^{(v)} = 1$	15,000	15,000
CR3: Create a new ticket for a visitor	$1_W^{(v)} + 1_W = 4$	5,000	20,000
RD10: Retrieve all tickets for an event	$1_R^{(v)} \times 50 = 50$	10	500
UP2: Check-in a ticket	$1_W^{(v)} = 2$	5,000	10,000
RD11: Count tickets issued in a year	$1_R^{*} = 1$	1	1
			45,501
Referencing (TICKET in VISITOR)			
RD9: Retrieve all tickets of a visitor	$1_R^{(v)} + 1_R^{(t)} = 2$	15,000	30,000
CR3: Create a new ticket for a visitor	$1_W^{(v)} + 1_W^{(t)} = 4$	5,000	20,000
RD10: Retrieve all tickets for an event	$50_R^{(t)} = 50$	10	500
UP2: Check-in a ticket	$1_W^{(t)} = 2$	5,000	10,000
RD11: Count tickets issued in a year	$150,000_R^{(t)} = 150,000$	1	150,000
			210,500

(*) **Together with the *Computed* pattern, the *Approximation* pattern is employed**. Strong consistency is not important for this operation, since an annual aggregation can tolerate minor inaccuracies.

The decision is to embed TICKET instances inside VISITOR documents, since visitors own few tickets on average and access patterns are mostly visitor-centric.

CHAT, CUSTOMER and ROLE

The relationship between a CHAT and a CUSTOMER is one-to-one. CHAT instances change frequently and can be considered bounded in growth, since we assume that the chat is only used to seek assistance. MESSAGE instances, related to CHAT instances through NEW CONVERSATION can easily be considered as being embedded inside CHAT, while the ones related through the OLD MESSAGES relationship are referenced inside the CHAT.

- The mean number of CHAT per GUIDE is $\frac{50{,}000}{15} = 3.3$;
- When an RD12 operation is issued, 50% of the time the requested messages are the most recent ones and can therefore be found embedded inside the chat instance.

Operation	**Accesses**	**Freq. (day)**	**Total**
Embedding (CHAT in CUSTOMER) + **Referencing** (CHAT in ROLE)			
RD12: Read 10 messages	$1_R^{(c)} + 10_R^{(m)} \times 0.5 = 6$	5,000	30,000
CR4: Create a new message	$1_W^{(c)} + 1_W^{(m)} = 4$	2,500	10,000
RD13: Retrieve active chats for a guide	$1_R^{(r)} + 1_R^{(c)} \times 3.3 = 4.3$	200	860
RD14: Count chats opened in a year	$53_R = 53$	1	53
			40,913
Referencing (CHAT in CUSTOMER and in ROLE)			
RD12: Read 10 messages	$1_R^{(c)} + 1_R^{(ch)} + 10_R^{(m)} \times 0.5 = 7$	5,000	35,000
CR4: Create a new message	$1_R^{(c)} + 1_W^{(ch)} + 1_W^{(m)} = 5$	2,500	12,500
RD13: Retrieve active chats for a guide	$1_R^{(r)} + 1_R^{(ch)} \times 3.3 = 4.3$	200	860
RD14: Count chats opened in a year	$53_R = 53$	1	53
			48,413

(*) $\lceil log_F(50{,}000) + \frac{5{,}000}{F} \rceil = 53$ blocks of an index B-tree, defined on the insertion date of the chats is needed, assuming an average fanout of $F = 100$.

The decision is to embed CHAT instances inside CUSTOMER documents and to reference them inside ROLE ones.

TRADE and ARTWORK

The relationship between ARTWORK and TRADE is one-to-few. Each trade involves exactly one artwork, while an artwork may be occasionally exchanged in multiple autions – i.e. when an auction fails to reach quota, the artwork is not sold.

Operation	**Accesses**	**Freq. (day)**	**Total**
Embedding (TRADE in ARTWORK)			
CR5: Insert a new trade	$1_W^{(a)} = 2$	10	20
UP3: Update trade information of an artwork	$1_W^{(a)} = 2$	10	20
RD15: Retrieve the artwork of a trade	$1_R^{(a)} = 1$	500	500
RD16: Count trades in a given year	$2,000_R^{(a)} = 2,000$	1	2,000
RD17: Retrieve the buyer of a given artwork	$1_R^{(a)} = 1$	500	500
			3,040
Referencing (TRADE in ARTWORK)			
CR5: Insert a new trade	$1_W^{(t)} = 2$	10	20
UP3: Update trade information of an artwork	$1_R^{(a)} + 1_W^{(t)} = 3$	10	30
RD15: Retrieve the artwork of a trade	$1_R^{(a)} + 1_R^{(t)} = 2$	500	1,000
RD16: Count trades in a given year	$2,000_R^{(t)} = 2000$	1	2,000
RD17: Retrieve the buyer of a given artwork	$1_R^{(a)} + 1_R^{(c)} = 2$	500	1,000
			4,050

From the cost analysis, embedding TRADE instances inside ARTWORK documents proves to be more efficient than referencing. Since trades are strongly tied to artworks and the access patterns primarily involve retrieving information starting from the artwork itself, embedding ensures cheaper query execution.

VISITOR and CUSTOMER

VISITOR and CUSTOMER establish an exclusive, partial hierarchy. Due to their similarities, they can be placed under the same collection. **Therefore, in accordance with the *inheritance* design pattern, the visitor collection, will include heterogeneous documents**. An additional `type` attribute is added whereby the two different types can be distinguished.

Figure 3.3 shows the results of the embeddings. The remaining relationships will be translated using references.

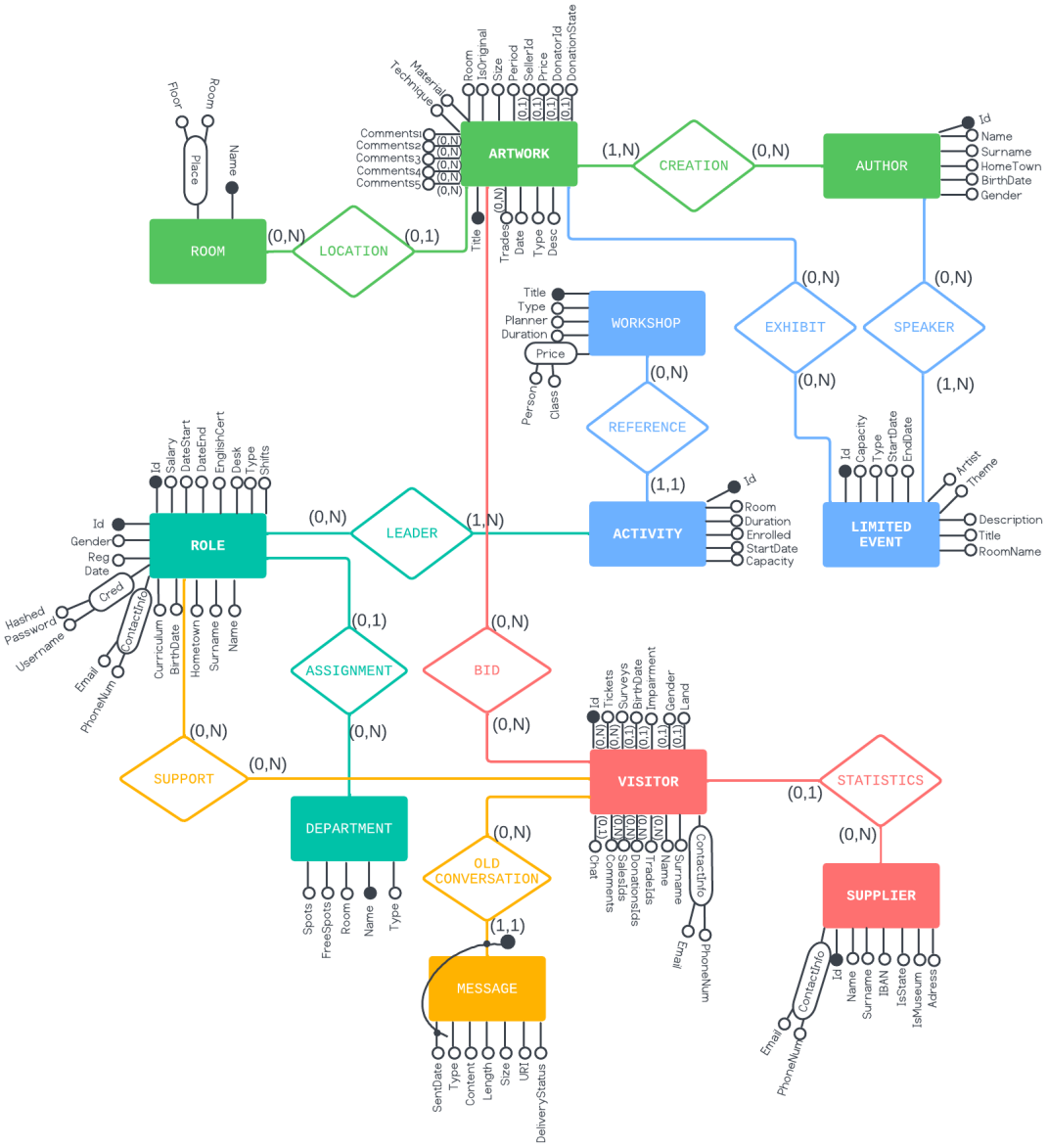


Figure 3.3: The final ER diagram after aggregates design. At this point, this can be considered a logical schema.

3.4 Aggregates representation

Once the aggregates have been selected according to the workload of the two software applications, they are represented using the *NoSQL Abstract Data Model* (NoAM) (Atzeni *et al.* [2020]). NoAM was devised as a logical abstraction of the physical design. As such, it can be translated into any NoSQL solution later on.

The tables below illustrate the different collections within the database, expressed using the NoAM language.

Artwork collection

Artwork		
Venus de Milo	date	1816
	type	Sculpture
	description	Plaster reproduction of the famous ancient Greek statue attributed to Alexandros of Antioch, representing the goddess Aphrodite.
	author	1
	isOriginal	false
	size	202
	period	Greek
	sellerId	34
	donatorId	12
	donationState	accepted
	locationName	Sculpture Hall
	authorIds	{ 21, 30 }

	tecnique	{
		⟨name: Casting, description: ...⟩
		⟨name: Moulding, description: ...⟩
		}

	material	{
		⟨name: Plaster, description: ...⟩,
		⟨name: Resin, description: ...⟩
		}

	commentsStar1	{
		⟨id: 101, message: ..., date: ...⟩,
		⟨id: 102, message: ..., date: ...⟩
		}

	commentsStar2	...
	commentsStar3	...
	commentsStar4	...

	commentsStar5	...

	trade	{
		⟨id: 501, dateStart: ..., dateEnd: ..., winPrice: ..., winnerId: ..., priceMax: ..., priceMin: ..., minIncrease: ..., isAuction: ...⟩
		}

Discobolus	date	1847
	type	Sculpture
	description	Plaster cast of the famous Greek statue by Myron, depicting an athlete in the act of throwing the discus.
	author	2
	isOriginal	false
	size	155
	period	Greek
	sellerId	45
	price	4200.00
	donatorId	19
	donationState	permanent donation
	locationName	Sculpture Hall
	authorIds	{ 79 }

	tecnique	{
		⟨name: ..., description: ...⟩
		⟨name: ..., description: ...⟩
		}

	material	{
		⟨name: ..., description: ...⟩,
		⟨name: ..., description: ...⟩
		}

	commentsStar1	{
		⟨id: ..., rating: ..., message: ..., date: ...⟩,
		⟨id: ..., rating: ..., message: ..., date: ...⟩
		}

	commentsStar2	...
	commentsStar3	...
	commentsStar4	...
	commentsStar5	...

	trade	{
		⟨id: ..., dateStart: ..., dateEnd: ..., winPrice: ...,
		winnerId: ..., priceMax: ..., priceMin: ...,
		minIncrease: ..., isAuction: ...⟩
		}

Author collection

Author		
101	name	Marco
	surname	Bianchi
	homeTown	Florence, Italy
	birthDate	1780-03-25
	gender	Male
	creationTitles	{ Discobolus, Venus de Milo }
102	name	Alice
	surname	Verdi
	homeTown	Paris, France
	birthDate	1982-07-10
	gender	Female
	creationsTitles	{ ... }

Room collection

Room		
Sculpture Hall	room	1
	floor	1
	artworkTitles	{ Venus de Milo, Discobolus }
Renaissance Gallery	room	3
	floor	2
	artworkTitles	{ ... }

Role collection

Role		
001	salary	2500.00
	startDate	2021-01-15
	endDate	2023-12-31
	englishCert	true
	desk	14
	type	guide
	activityIds	{ 001, 002, 005 }
	department	Department of Ancient Sculpture
	chatIds	{ 876, 855, 902 }

	shifts	{
		⟨id: 701, timeStart: 09:00, timeEnd: 13:00, dayOfWeek: ...⟩,
		⟨id: 702, timeStart: 14:00, timeEnd: 18:00, dayOfWeek: ...⟩
		}

	employees	{
		⟨id: 501, gender: ..., regDate: ..., email: ..., phoneNum: ...,
		curriculum: ..., birthDate: ..., homeTown: ..., name: ...,
		surname: ..., hashed: ..., password: ..., username: ...⟩
		}

002	salary	3100.00
	startDate	2022-03-01
	endDate	9999-12-31
	englishCert	false
	desk	21
	type	teacher
	activityIds	{ 003, 004 }
	department	Department of Renaissance Painting
	chatIds	{ 304, 305 }

	shifts	{
		⟨id: ..., timeStart: ..., timeEnd: ..., dayOfWeek: ...⟩,
		⟨id: ..., timeStart: ..., timeEnd: ..., dayOfWeek: ...⟩
		}

	employees	{ ⟨id: ..., gender: ..., regDate: ..., email: ..., phoneNum: ..., curriculum: ..., birthDate: ..., homeTown: ..., name: ..., surname: ..., hashed: ..., password: ..., username: ...⟩ }

Department collection

Department		
Administration	spots	40
	freeSpots	2
	room	4
	type	office
Secretary	spots	15
	freeSpots	1
	room	1
	type	desk

Workshop collection

Workshop		
Things Tell Stories	type	Museum
	planner	Giulia Conti
	duration	2
	class	70
	activityIds	{ 32, 56, 43 }
Imaginary Bestiary	type	Museum
	planner	Paolo Rossi
	duration	2
	class	70
	activityIds	{ ... }

Activity collection

Activity		
001	room	4
	duration	02:00:00
	enrolled	2
	startDate	2025-06-15 10:00
	capacity	10
	ticketIds	{ 2001, 2002 }
	workshopTitles	{ Introduction to Sculpture, Moulding Techniques }
002	room	44
	duration	01:30:00
	enrolled	20
	startDate	2025-07-10 15:00
	capacity	30
	ticketIds	{ ... }
	workshopTitles	{ ... }

LimitedEvent collection

LimitedEvent		
001	capacity	200
	type	Temporary exhibition
	startDate	2023-09-01
	endDate	2023-10-15
	artist	Sandro Botticelli
	theme	Florentine Renaissance
	description	Exhibition dedicated to Botticelli's main works, with a focus on Renaissance symbolism.
	title	Botticelli and the Renaissance
	roomName	Renaissance Gallery
	authorIds	{ 110, 111, 112 }
	artworkTitles	{ The Primavera, Botticelli's Venus }
	ticketIds	{ 4001, 4002, 4003, 4004 }

002	capacity	150
	type	Interactive exhibition
	startDate	2023-11-05
	endDate	2023-11-20
	artist	Michelangelo Buonarroti
	theme	Sculpture techniques
	description	Educational workshop focused on the creative and technical process behind Michelangelo's sculptures.
	title	Inside Michelangelo's Workshop
	roomName	Sculpture Hall
	authorIds	{ ... }
	artworkTitles	{ ... }
	ticketIds	{ ... }

Visitor collection

Visitor		
001	isCustomer	false
	birthDate	1985-04-12
	impairment	visual
	gender	Female
	land	Italy

	tickets	{
		⟨id: 701, price: 8.00, printDate: 2023-06-15⟩,
		⟨id: 702, price: 0.00, printDate: 2024-01-05⟩
		}

	surveys	{
		⟨id: 301, date: ..., type: ..., companions: ..., motivation: ..., degree: ..., visitCount: ..., comeBack: ..., tour: ..., experience: ..., facility: ...⟩,
		
		}

002	isCustomer	false
	birthDate	1992-11-23
	impairment	none

	gender	Male
	land	Germany

	tickets	{
		⟨id: ..., price: ..., printDate: ...⟩,
		⟨id: ..., price: ..., printDate: ...⟩
		}

	surveys	{
		⟨id: ..., date: ..., type: ..., companions: ..., motivation: ...,
		degree: ..., visitCount: ..., comeBack: ..., tour: ...,
		experience: ..., facility: ...⟩,
		}

003	isCustomer	true
	birthDate	1992-11-23
	impairment	none
	gender	Male
	land	Germany

	tickets	{
		⟨id: ..., price: ..., printDate: ...⟩,
		⟨id: ..., price: ..., printDate: ...⟩
		}

	surveys	{
		⟨id: ..., date: ..., type: ..., companions: ..., motivation: ...,
		degree: ..., visitCount: ..., comeBack: ..., tour: ...,
		experience: ..., facility: ...⟩,
		
		⟨id: ..., date: ..., type: ..., companions: ..., motivation: ...,
		degree: ..., visitCount: ..., comeBack: ..., tour: ...,
		experience: ..., facility: ...⟩,
		}

	surname	Rossi
	name	Laura
	donationsId	{ 501, 502 }
	salesId	{ 301 }
	phoneNum	+39 071 1234567
	email	laura.rossi@gmail.com
	tradeIds	{ 201, 202, 205 }

	chat	{
		⟨dateCreation: 2023-03-12 10:15:00,
		newMessages: {
		⟨sentDate: ..., type: ..., content: ..., length: ..., size: ...,
		uri: ..., deliveryStatus: ...⟩,
		
		⟨sentDate: ..., type: ..., content: ..., length: ..., size: ...,
		uri: ..., deliveryStatus: ...⟩
		}
		oldMessages: { 003_1751555841, 003_1751546821 }
		}

	comments	{
		⟨id: 876, rating: 5⟩,
		⟨id: 432, rating: 4⟩
		}

Supplier collection

Supplier		
402	name	National Museum
	iban	FR7630006000011234567890189
	isState	true
	isMuseum	true
	email	info@nationalmuseum.fr
	phoneNum	+33 331 9876543

	surveys	{
		⟨id: 301, date: ..., type: ..., companions: ..., motivation: ...,
		degree: ..., visitCount: ..., comeBack: ..., tour: ...,
		experience: ..., facility: ...⟩,
		
		⟨id: 302, date: ..., type: ..., companions: ..., motivation: ...,
		degree: ..., visitCount: ..., comeBack: ..., tour: ...,
		experience: ..., facility: ...⟩
		}

403	name	Mario
	surname	Gialli
	iban	IT60X0542811101000000123456
	isState	false
	isMuseum	false

	email	mario.gialli@gmail.com
	phoneNum	+39 331 1234567

	surveys	{
		⟨id: ..., date: ..., type: ..., companions: ..., motivation: ..., degree: ..., visitCount: ..., comeBack: ..., tour: ..., experience: ..., facility: ...⟩,
		
		⟨id: ..., date: ..., type: ..., companions: ..., motivation: ..., degree: ..., visitCount: ..., comeBack: ..., tour: ..., experience: ..., facility: ...⟩
		}

Message collection

Message		
003_1751555841	customerId	003
	sentDate	2025-04-23T18:25:43.511Z
	type	text
	content	Hi! I need assistance with the ticked I've purchased...
	deliveryStatus	sent
003_1751546821	customerId	003
	sentDate	2025-04-21T18:31:24.211Z
	type	audio
	length	89
	content	\x06\x00\x1A\xCE\x06\x00\x1A\xCE ...
	deliveryStatus	read

Since the MESSAGE has two identifiers, **the *Composite Key* design pattern was used to create a new, redundant key by concatenating the `customerId` and `sentDate` identifiers**. This means that items will be sorted first by the customer who owns the chat and then chronologically within that customer.

3.5 Aggregates partitioning

Of the available data representation strategies, a hybrid strategy is chosen that is much closer to the *Entry per Top-level Field* (ETF) extreme than the *Entry per Aggregate Object* (EAO). The only exception lies in the comments on the artworks, which are divided into five distinct top-level entries.

CHAPTER 4

MongoDB implementation

The NoAM representation is now ready to be translated into the document store representation. MongoDB has been chosen as the NoSQL solution for this project.

4.1 Translation from NoAM to MongoDB data model

In this final phase, we will translate the NoAM model presented above into the final NoSQL representation. Since we opted for MongoDB, a NoSQL document-oriented database, we will translate each aggregate into JSON document form.

According to [Atzeni *et al.*, 2023], this process is quite straightforward. Each NoAM collection can be mapped to a MongoDB collection, and each NoAM block can be mapped to a MongoDB document. The document is a BSON serialization of the potentially nested entry values of the block, with the addition of MongoDB's special *_id* field for the primary key, which value is the the block key. The following considerations were made during the translation process:

1. MongoDB internally stores data in the *BSON* format [MongoDB, 2025a]. This is to overcome the limitations of the *JSON* format and increase the retrieval and traversal of documents with metadata. Since *JSON* can only directly represent a subset of the types supported by *BSON*, **MongoDB has extended the *JSON* format to create the *EJSON* format** [MongoDB, 2025b]. For the sake of clarity, we chose to use the *relaxed* format variant, which *fosters readability at the expenses of expressiveness*. Therefore, `ObjectId` instences are represented as `{"$oid": <hashed_id>}` and dates as `{"$date": "<ISO-8601 Date/Time Format>"}`

2. References have been translated with the *DBRef* standard. **A *DBRef* is only a common convention, not a proper type. Therefore, any extra keys do not constitute an error**, unlike all the other types defined in *EJSON*. We exploit this property to translate hybrid embedded-referenced documents, such as `comments` within the `visitors` collection.

Only the most relevant collections, i.e. those having embedding docs, are shown below.

Artwork JSON

```
{
  "_id": {"$oid": "Venus de Milo"},
  "date": {"$date": "18160901T175205+0200"},
  "type": "Sculpture",
  "description": "Plaster reproduction of the Greek statue by...",
  "author": 1,
  "is_original": false,
  "size": {"w":202, "h":35, "d":80},
  "period": "Greek",
  "seller_id": 34,
  "donator_id": 12,
  "donation_state": "accepted",
  "location_name": "Sculpture Hall",
  "author_ids": [21, 30],
  "tecniques": [
    {
      "name": "Casting",
      "description": "Made with plaster cast."
    },
    ...
  ],
  "materials": [
    {
      "name": "Plaster",
      "description": "Used for reproduction."
    },
    ...
  ],
  "comments_star_1": [
    {
      "_id": 101,
      "message": "Too simple.",
      "date": {"$date": "20230901T175205+0200"}
    },
    ...
  ],
  "comments_star_2": [...],
  "comments_star_3": [...],
  "comments_star_4": [...],
  "comments_star_5": [...],
  "trade": {
    "id": 501,
    "date_start": {"$date": "20220301T175205+0200"},
    "date_end": {"$date": "20220901T175205+0200"},
    "win_price": 5000,
    "winner_id": 77,
    "price_max": 6000,
    "price_min": 3000,
    "min_increase": 200,
    "is_auction": true
  }
}
```

Role JSON


```
{
  "_id": {"$oid": "57e193d7a9cc81b4027498b5"},
  "salary": 2500.0,
  "start_date": {"$date": "20210901T175205+0200"},
  "end_date": {"$date": "20230901T175205+0200"},
  "english_cert": true,
  "desk": 14,
  "type": "guide",
  "activity_ids": [601, 502, 605],
  "department_name": "Department of Ancient Sculpture",
  "chat_ids": [876, 855, 902],
  "shifts": [
    {
      "id": 701,
      "time_start": "09:00",
      "time_end": "13:00",
      "day_of_week": "Monday"
    },
    ...
  ],
  "gender": "female",
  "reg_date": {"$date": "20200901T175205+0200"},
  "email": "anna.rossi@example.com",
  "phone_num": "+39 345 6789012",
  "curriculum": "MA in Art History",
  "birth_date": {"$date": "19920901T175205+0200"},
  "home_town": "Florence",
  "name": "Anna",
  "surname": "Rossi",
  "password": "a4103c771510bfcd72e9a48b4a876575",
  "username": "arossi"
}
```


Visitor JSON


```
{
  "_id": {"$oid": "57e193d7a9cc81b4027498b5"},
  "is_customer": true,
  "birth_date": {"$date": "19920901T175205+0200"},
  "impairment": null,
  "gender": "Male",
  "land": "Germany",
  "tickets": [
    {
      "price": 12.00,
      "print_date": {"$date": "20230901T175205+0200"}
    },
    ...
  ],
  "surveys": [
    {
```

```
      "date": {"$date": "20230901T175205+0200"},
      "type": "Feedback",
      "companions": 2,
      "motivation": "Art interest",
      "degree": "High",
      "visit_count": 3,
      "come_back": true,
      "tour": "Renaissance Highlights",
      "experience": "Very satisfied",
      "facility": "Accessible"
    },
    ...
  ],
  "surname": "Rossi",
  "name": "Laura",
  "trade_ids": [501, 502],
  "sales_id": [301],
  "phone_num": "+39 071 1234567",
  "email": "laura.rossi@gmail.com",
  "trade_ids": [201, 202, 205],
  "chat": {
    "date_creation": {"$date": "20230901T175205+0200"},
    "new_messages": [
      {
        "sent_date": {"$date": "20230901T175205+0200"},
        "type": "text",
        "content": "Hello, I have a question.",
        "length": 25,
        "delivery_status": "sent"
      },
      ...
    ],
    "old_messages": [
      {
        "$ref": "messages",
        "$id": {"$oid": "003_1751555841"}
      },
      ...
    ]
  },
  "comments": [
    {
        "$ref": "messages",
        "$id": {"$oid": "003_1751555841"}
        "rating": 5
    },
    ...
  ]
}
```

Supplier JSON

```
{
```

```
  "_id": {"$oid": "57e193d7a9cc81b4027498b5"},
  "name": "National Museum",
  "is_state": true,
  "is_museum": true,
  "email": "info@nationalmuseum.fr",
  "phone_num": "+33 331 9876543",
  "surveys": [...]
}
```

4.2 Database seeding

4.2.1 MySQL to MongoDB data migration

Because the database schema has evolved significantly since the creation of the current MySQL database, a subset of the collections was migrated using the *MongoDB Relational Migrator*[1]. After connecting to the MySQL server, the software analyzed the relational schema to assess the simplicity of the translation. The results are shown in Figure 4.1.

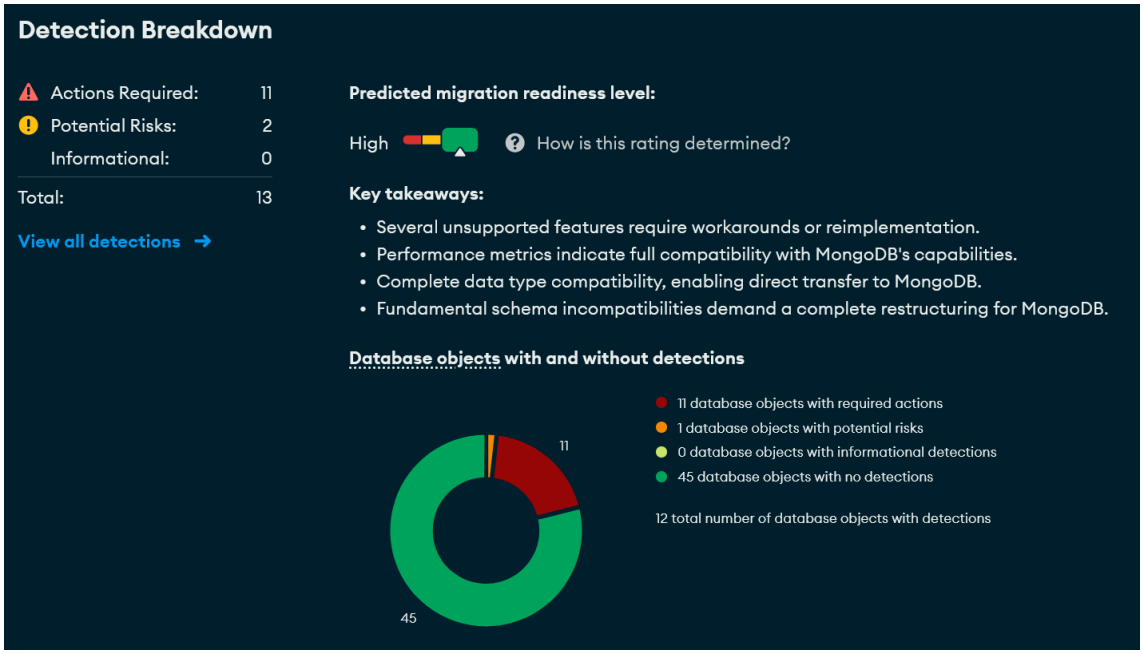


Figure 4.1: The results of the MongoDB Relational Migrator's analysis. The existing MySQL implementation was found to be easily translatable to a MongoDB schema. *Foreign key* inter-document constraints, stand as an exception because they cannot be defined directly in the schema.

The generated schema was translated from Italian to English, adhering to the *snake case* casing. Because of the different schema, only the *departments*, *authors*, *workshops*, and *surveys* collections were kept after the migration process.

[1] https://www.mongodb.com/resources/solutions/use-cases/mysql-to-mongodb

4.2.2 Hand-made seeders

All of the other collections were seeded using ad hoc seeders based on the *Faker*[2] library. **Although very convenient, the generated documents occasionally break the inter-document and intra-document constraints.**

4.2.3 Consistency retrieval

Manual update operations are defined to restore consistency. In particular, the "type" fields of collections resulting from a hierarchy unification are checked to ensure consistency with existing top-level entries.

The final snapshot consist of 11 collections, as shown in Figure 4.2.

Figure 4.2: The overview of the final MongoDB database. The snapshot can be found here.

4.2.4 Secondary B-Tree Indexes definition

In several operations, the presence of an index is essential. Operation RD10 (*read all the tickets of an event*) for instance, performs a *lookup* to retrive the visitor document that holds a ticket which id is given.

Without defining the index `db.visitors.create_index({ "tickets._id": 1})`, the operation would need to scan the entire visitors collection to find the ticket of interest. Thanks to this index, however, **only a document is accessed after first traversing a B-tree to determine which visitor has the ticket**. Figure 4.3 it shows a report proving the performance improvement.

[2] https://fakerjs.dev/

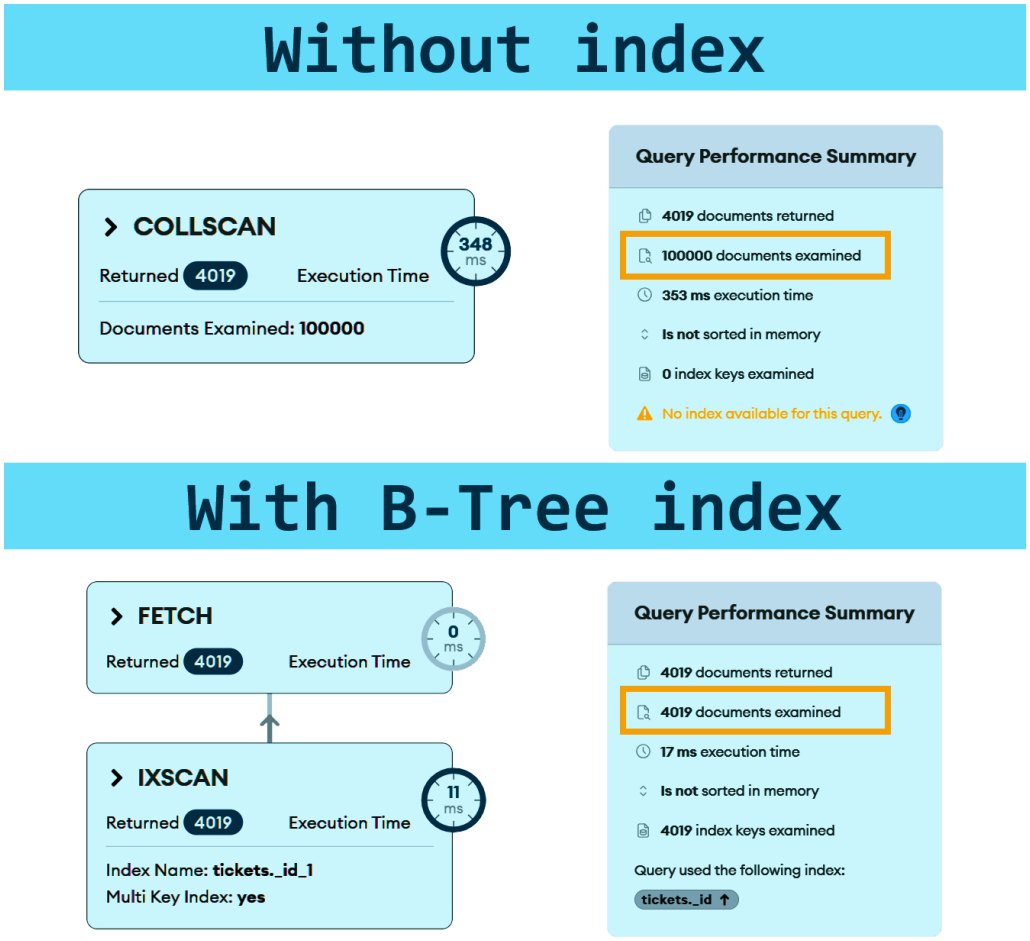


Figure 4.3: The improvement of the *lookup* (query RD10) performance after having defined an index on `ticket._id`. Thanks to the index, only one visitor document is accessed for each ticket id.

B-tree indexes are also valuable for ranged-query. Operation RD10 (*read all the surveys, relative to this museum, filled out in a specific year*) is a bulk operation that needs to retrieve a bunch of documents based on a *datetime* field. `db.visitors.create_index({ "surveys.date_of_compilation" : 1})` defines an index that ensures the fast retrieval of the physical position of a large number of visitor documents based on a date range specified on the embedded survey documents. However, since it is a *secondary* index, **contiguous allocation cannot be assumed. A random disk access is required for each visitor document.** At least, though, scanning the entire collection is avoided.

4.3 Operations coding

The operations listed in Table 2.3 have been implemented in *PyMongo*, Python API. *PyMongo* opens a TCP socket to communicate with the server–`mongodb://localhost:27017` in our local playground–formatting the queries as BSON.

Many operations use *aggregate pipelines* to take advantage of optimized server-side computation and minimize the number of requests and the amount of data received in response. This is particularly true when inter-collection joins are required–such as retrieving the chats of a guide in RD13–or when aggregated values are needed–such as computing the average rating of

artworks in RD3.

4.3.1 Create operations

CR1: Create a new comment to a specific artwork

```
db.artworks.update_one(
    {"_id": <artwork_id>},
    {
        "$push": {
            f"comments_star_{rating}": {
                "_id": <new_comment_id>,
                "rating": <rating>,
                "date": datetime.now(),
                "message": "Capolavoro assoluto, emozionante da vedere dal vivo!
    ",
            }
        }
    },
)
db.customers.update_one({"_id": <customer_id>}, {"$push": {"comments": <
    new_comment_id>}})
```

CR2: Create a new survey compilation

```
db.visitors.update_one(
    {"_id": <visitor_id>},
    {
        "$push": {
            "surveys": {
                "accompanying_persons_visit": "amici",
                "date_of_compilation": datetime(2025, 9, 5, 15, 30),
                "reason_for_visit": "Interesse per l'arte moderna",
                "number_of_visits": 2,
                "return": "si'",
                "type_of_visit": "visita guidata",
                "title_of_studies": "Diploma",
                "evaluation_of_experience": 9,
                "evaluation_of_facility": 8,
                "evaluation_of_visit": 10
            }
        }
    }
)
```

CR3: Create a new ticket for a visitor

```
db.visitors.update_one(
    {"_id": <visitor_id>},
    {
        "$push": {
            "tickets": {
```

```
                "date": datetime(2025, 9, 6, 11, 0),
                "price": 18.0,
                "check_in": False
            }
        }
    }
)
```

CR4: Create a new message

```
visitor_id = db.visitors.find_one({"chat": {"$exists": True}})["_id"]

new_message = {
    "customerId": visitor_id,
    "sentDate": datetime(2025, 9, 5, 16, 0),
    "type": "text",
    "content": "Benvenuto al Museo Omero!",
    "deliveryStatus": "sent",
    "length": 24,
    "uri": None,
}

visitor = db.visitors.find_one({"_id": visitor_id})
chat = visitor["chat"]
new_msgs = chat.get("new_messages", [])
old_msgs = chat.get("old_messages", [])

new_msgs.append(new_message)

# If there are more than 10 new messages, move the oldest to old messages
if len(new_msgs) > 10:
    oldest = new_msgs.pop(0)
    oldest_id = ObjectId()
    old_msgs.insert(0, oldest_id)
    db.messages.insert_one({"_id": oldest_id, **oldest})

db.visitors.update_one(
    {"_id": visitor_id},
    {"$set": {"chat.new_messages": new_msgs, "chat.old_messages": old_msgs}},
)
```

CR5: Create a new trade

```
db.artworks.update_one(
    {"_id": <artwork_id>},
    {
        "$set": {
            "trade": {
                "date_start": datetime(2025, 9, 10, 10, 0),
                "date_end": datetime(2025, 9, 15, 18, 0),
                "win_price": 12000,
                "winner_id": <customer_id>,
```

```
                "price_max": 15000,
                "price_min": 8000,
                "price_start": 5000,
                "min_increase": 200,
                "is_auction": True,
            }
        }
    },
)
db.visitors.update_one({"_id": <customer_id>}, {"$addToSet": {"tradeIds": <
    artwork_id>}})
```

CR6: Create a new visitor

```
db.visitors.insert_one({
    "isCustomer": False,
    "birthDate": datetime(1990, 5, 15, 2, 0),
    "gender": "male",
    "land": "Italy",
    "tickets": [
        {
            "date": datetime(2024, 7, 1, 10, 0),
            "price": 15.0,
            "check_in": True
        }
    ],
})
```

CR7: Create a new activity

```
db.activities.insert_one({
    "room": "Rinascimentale",
    "duration": 90,
    "enrolled": 5,
    "startDate": datetime(2025, 9, 4, 15, 0),
    "capacity": 30,
    "ticketIds": [34, 46, 9, 7, 12],
    "workshopTitle": "Pittura su tela"
})
```

CR8: Create a new artwork

```
db.artworks.insert_one(
    {
        "_id": "La Notte Stellata",
        "date": datetime(1889, 6, 1),
        "type": "painting",
        "description": "Opera celebre di Van Gogh",
        "isOriginal": True,
        "size": 73,
```

```
        "period": "Post-Impressionismo",
        "sellerId": 9,
        "donatorId": 13,
        "donationState": "refused",
        "locationName": "Parigi",
        "authorIds": [<author_id>],
        "tecniques": ["mosaic"],
        "materials": ["Glass", "Wood"],
        "comments_star_3": [
            {
                "rating": 3,
                "date": datetime(2025, 8, 12, 10, 30),
                "message": "Un buon lavoro, ma si pu  migliorare",
            }
        ],
    }
)
```

CR9: Create a new limited event

```
db.limited_events.insert_one(
    {
        "_id": "The Future of Mosaic",
        "capacity": 40,
        "type": "exhibition",
        "startDate": datetime(2025, 10, 1, 10, 0),
        "endDate": datetime(2025, 10, 1, 18, 0),
        "theme": "Innovation",
        "description": "Mostra temporanea di opere moderne",
        "roomName": "Contemporaneo",  # room id
        "authorIds": [<author_id>],
        "artworkTitles": ["The Scream", "Guernica"],
        "ticketIds": [42, 51, 8, 6, 7],
    }
)
```

CR10: Create a new employee

```
db.roles.insert_one(
    {
        "phone_number": "+39 0922684697",
        "surname": "Gozzano",
        "curriculum": "http://monteverdi.com/",
        "birth_date": datetime(2025, 8, 23, 7, 6, 2),
        "date_start": datetime(2025, 9, 2, 18, 14),
        "email": "carmelo87@example.com",
        "hometown": "Trevenzuolo",
        "name": "Amedeo",
        "gender": "F",
        "username": "ludovico03",
        "password": "67d71a0a46db3ea107aa55cab0f8de7d",
        "salary": 40000,
```

```
        "desk": 2,
        "department": "Amministrazione",
        "type": "teaching staff",
        "english_cert": "A1",
    }
)
```

CR11: Create a new shift

```
db.roles.update_one(
    {"_id": out.inserted_id},
    {
        "$push": {
            "shifts": {
                "time_start": "08:50:23",
                "time_end": "16:58:45",
                "day_of_week": "martedi'",
            }
        }
    },
)
```

CR12: Create a new author

```
db.authors.insert_one({
    "surname": "Rossi",
    "birth_date": datetime(1975, 4, 12),
    "home_town": "Milano",
    "name": "Giovanni",
    "gender": "M"
})
```

CR13: Create a new workshop

```
db.workshops.insert_one({
    "_id": "Pittura su tela",
    "duration": datetime(1970, 1, 1, 2, 0),
    "price_class": 80.00,
    "price_person": 25.00,
    "type": "Design"
})
```

CR14: Create a new supplier

```
db.suppliers.insert_one(
    {
        "name": "Maurizio Conte",
        "iban": "IT60X0542811101000000123456",
        "isState": True,
```

```
        "isMuseum": False,
        "email": "conte_mauto@outlook.it",
        "phoneNum": "+390276543210",
    }
)
```

4.3.2 Read operations

RD1: Read all the information of a given artwork, including its authors, materials, crafting techniques and room placement

```
db.artworks.aggregate(
    [
        {"$match": {"_id": <artwork_id>}},
        {
            "$lookup": {
                "from": "authors",
                "localField": "author_ids",
                "foreignField": "_id",
                "as": "authors",
            }
        },
        {
            "$project": {
                "title": 1,
                "date": 1,
                "type": 1,
                "description": 1,
                "is_original": 1,
                "size": 1,
                "period": 1,
                "location_name": 1,
                "tecniques": 1,
                "materials": 1,
                "authors": {
                    "$map": {
                        "input": "$authors",
                        "as": "a",
                        "in": {"$concat": ["$$a.name", "$$a.surname"]},
                    }
                },
            }
        },
    ]
)
```

RD2: Read the last 10 comments of a given artwork

```
db.artworks.aggregate(
    [
        {"$match": {"_id": <artwork_id>}},
        {
```

```
            "$project": {
                "all_comments": {
                    "$concatArrays":
                        [f"$comments_star_{s}" for s in range(1, 6)]
                }
            }
        },
        {"$sort": {"all_comments.date": -1}},
        {"$limit": 10},
    ]
)
```

RD3: Calculate the average rating of a given artwork

```
db.artworks.aggregate(
    [
        {"$match": {"_id": <artwork_id>}},
        {
            "$project": {
                "all_comments": {
                    "$concatArrays": [f"$comments_star_{s}" for s in range(1, 6)
    ]
                }
            }
        },
        {"$unwind": "$all_comments"},
        {"$group": {"_id": None, "avg_rating": {"$avg": "$all_comments.rating"
    }}},
    ]
)
```

RD4: Calculate the average rating of the comments posted by a given customer

```
db.visitors.aggregate(
    [
        {"$match": {"_id": <customer_id>}},
        {"$unwind": "$comments"},
        {"$group": {"_id": None, "avg_rating": {"$avg": "$comments.rating"}}},
    ]
)
```

RD5: Read all the comments of a given artwork with a given rating

```
db.artworks.aggregate(
    [
        {"$match": {"_id": <artwork_id>}},
        {"$unwind": f"$comments_star_{<rating>}"},
        {"$replaceRoot": {"newRoot": f"$comments_star_{rating}"}},
    ]
)
```

RD6: Read all the surveys, relative to this museum, filled out in a specific year

```
db.visitors.find(
    {"surveys.date_of_compilation": {"$gte": <date_start>, "$lte": <date_end>}},
    {"surveys": 1},
)
```

RD7: Read all the surveys filled out by a specific customer

```
db.visitors.find_one({"_id": <customer_id>}, {"surveys": 1})
```

RD8: Read all the surveys from a specific affiliated museum

```
db.suppliers.aggregate(
    [
        {"$match": {"_id": <supplier_id>}},
        {
            "$lookup": {
                "from": "visitors",
                "localField": "survey_ids",
                "foreignField": "_id",
                "as": "surveys",
            }
        },
    ]
)
```

RD9: Read all tickets of a visitor

```
db.visitors.find_one(<visitor_id>)["tickets"]
```

RD10: Read all tickets for an event

```
db.activities.aggregate(
    [
        {"$match": {"_id": <event_id>}},
        {
            "$lookup": {
                "from": "visitors",
                "localField": "ticket_ids",
                "foreignField": "tickets._id",
                "as": "visitors",
            }
        },
    ]
)
```

RD11: Count tickets issued in a year

```
db.visitors.count_documents(
    {"tickets.date": {"$gte": <start>, "$lt": <end>}}, hint="tickets.date_1"
)
```

RD12: Read 10 messages of a given chat previous than a specific date

```
customer_id = db.visitors.find_one({"chat": {"$exists": 1}})["_id"]
before_date = datetime(2026, 8, 25)

pipeline = [
    {"$match": {"_id": customer_id}},
    {"$project": {"chat.new_messages": 1}},
    {"$unwind": "$chat.new_messages"},
    {"$match": {"chat.new_messages.sent_date": {"$lt": before_date}}},
    {"$sort": {"chat.new_messages.sent_date": -1}},
    {"$limit": 10},
    {"$replaceRoot": {"newRoot": "$chat.new_messages"}},
]
msgs = list(db.visitors.aggregate(pipeline))

if (l := len(msgs)) < 10:
    old_ids = db.visitors.find_one(customer_id)["chat"]["old_messages"]
    old_msgs = list(
        db.messages.find(
            {
                "_id": {"$in": old_ids},
                # "customer_id": customer_id, # Inconsitent
                "sent_date": {"$lt": before_date},
            },
            sort=[("sent_date", -1)],
            limit=10 - l,
        )
    )
    msgs.extend(old_msgs)
jprint(msgs[:2])
```

RD13: Read active chats for a guide

```
db.roles.aggregate(
    [
        {"$match": {"_id": <guide_id>}},
        {
            "$lookup": {
                "from": "visitors",
                "localField": "chat_ids",
                "foreignField": "chat._id",
                "as": "customers",
            }
        },
    ]
```

```
)
```

RD14: Count chats opened in a year

```
db.visitors.count_documents(
    {"chat.date_creation": {"$gte": <start>, "$lt": <end>}}, hint="chat.
    date_creation_1"
)
```

RD15: Retrieve the artwork of a trade

```
db.artworks.find_one({"trade._id": <trade_id>})
```

RD16: Count trades in a given year

```
db.artworks.count_documents(
    {"trade.date_start": {"$gte": <start>, "$lt": <end>}}, hint="trade.
    date_start_1"
)
```

RD18: Retrieve all the artworks of a given author

```
db.artworks.find({"author_ids": <author_id>})
```

RD19: Retrieve all activity of a given workshop type

```
db.activities.find({"workshop_title": <workshop_title>})
```

RD20: Read all the surveys filled out in a specific day

```
db.visitors.find (
    {"surveys.date_of_compilation": {"$gte": <date_start>, "$lte": <date_end>}},
    {"surveys": 1},
).hint("surveys.date_of_compilation_1")
```

RD21: Retrieve information about a given laboratory activity, including the total number of reservations

```
db.activities.aggregate(
    [
        {"$match": {"_id": <activity_id>}},
        {"$addFields": {"total_reservations": {"$size": {"$ifNull": ["
    $ticket_ids", []]}}}}
    ]
)
```

RD22: Retrieve all the information about a limited event

```
db.limited_events.find({"_id": <event_id>})
```

RD23: Retrieve all the details of a visitor

```
db.visitors.find_one({"_id": <visitor_id>})
```

RD24: Retrieve the visitor count for each land from most frequent to least frequent

```
db.visitors.aggregate(
    [
        {"$sortByCount": "$land"},
        {"$project": {"state": "$_id", "count": 1, "_id": 0}},
    ]
)
```

RD25: Retrieve the visitor count for each impairment from most frequent to least frequent

```
db.visitors.aggregate(
    [
        {"$sortByCount": "$impairment"},
        {"$project": {"impairment": "$_id", "count": 1, "_id": 0}},
    ]
)
```

RD26: Count tickets issued for each year

```
db.visitors.aggregate(
    [
        {"$unwind": "$tickets"},
        {"$project": {"year": {"$year": "$tickets.date"}}},
        {"$sortByCount": "$year"},
        {"$project": {"year": "$_id", "count": 1, "_id": 0}}
    ], hint="tickets.date_1"
)
```

RD27: Calculate the distribution of visitor ages grouped into bands based on ticket activity in a specific year

```
db.visitors.aggregate(
    [
        {
            "$match": {
                "birth_date": {"$ne": None},
                "tickets": {
                    "$elemMatch": {
```

```
                        "date": {"$gte": <start_date>, "$lt": <end_date>}
                    }
                }
            }
        },
        {
            "$addFields": {
                "valid_ticket": {
                    "$first": {
                        "$filter": {
                            "input": "$tickets",
                            "as": "t",
                            "cond": {
                                "$and": [
                                    {"$gte": ["$$t.date", <start_date>]},
                                    {"$lt": ["$$t.date", <end_date>]}
                                ]
                            }
                        }
                    }
                }
            }
        },
        {
            "$addFields": {
                "age": {
                    "$dateDiff": {
                        "startDate": "$birth_date",
                        "endDate": "$valid_ticket.date",
                        "unit": "year"
                    }
                }
            }
        },
        {
            "$addFields": {
                "age_group": {
                    "$switch": {
                        "branches": [
                            {"case": {"$lte": ["$age", 18]}, "then": "0–18"},
                            {"case": {"$and": [{"$gt": ["$age", 18]}, {"$lte": [
"$age", 30]}]}, "then": "19–30"},
                            {"case": {"$and": [{"$gt": ["$age", 30]}, {"$lte": [
"$age", 45]}]}, "then": "31–45"},
                            {"case": {"$and": [{"$gt": ["$age", 45]}, {"$lte": [
"$age", 65]}]}, "then": "46–65"},
                            {"case": {"$gt": ["$age", 65]}, "then": "65+"}
                        ],
                        "default": "unknown"
                    }
                }
            }
        },
        {
            "$group": {
```

```
                "_id": "$age_group",
                "count": {"$sum": 1}
            }
        },
        {
            "$group": {
                "_id": None,
                "total": {"$sum": "$count"},
                "groups": {
                    "$push": {
                        "age_group": "$_id",
                        "count": "$count"
                    }
                }
            }
        },
        {"$unwind": "$groups"},
        {
            "$project": {
                "_id": 0,
                "age_group": "$groups.age_group",
                "count": "$groups.count",
                "percentage": {
                    "$round": [
                        {"$multiply": [{"$divide": ["$groups.count", "$total"]}, 100]},
                        2
                    ]
                }
            }
        },
        {"$sort": {"age_group": 1}}
    ], hint="tickets.date_1"
)
```

RD28: Count material usage in all artworks

```
db.artworks.aggregate(
    [
        {"$unwind": "$materials"},
        {"$sortByCount": "$materials"},
        {"$project": {"material": "$_id", "count": 1, "_id": 0}},
    ]
)
```

RD29: Count top 5 visit reasons

```
db.visitors.aggregate(
    [
        {"$unwind": "$surveys"},
        {"$sortByCount": "$surveys.reason_for_visit"},
        {"$limit": 5},
```

```
        {"$project": {"reason": "$_id", "count": 1, "_id": 0}},
    ]
)
```

RD30: Count how many artworks are original vs. not

```
db.artworks.aggregate(
    [
        {
            "$group": {
                "_id": "$is_original",
                "count": {"$sum": 1} # True = Original
            }
        },
        {"$sort": {"_id": -1}},
        {"$project": {"Original": "$_id", "count": 1, "_id": 0}},
    ]
)
```

RD31: Count how many visitors came back

```
db.visitors.aggregate(
    [
        {"$project": {
            "num_tickets": {"$size": {"$ifNull": ["$tickets", []]}}
        }},
        {"$match": {"num_tickets": {"$gt": 1}}},
        {"$count": "count"},
        {"$project": {"visitors returned": "$count", "_id": 0}}
    ]
)
```

RD32: Count top 5 days with most ticket activity in a given year

```
db.visitors.aggregate(
    [
        {"$match": {"tickets.date": {"$gte": <start>, "$lt":<end>}}},
        {"$unwind": "$tickets"},
        {"$project": {
            "day": {
                "$dateTrunc": {
                    "date": "$tickets.date",
                    "unit": "day"
                }
            }
        }},
        {"$group": {
            "_id": "$day",
            "count": {"$sum": 1}
        }},
```

```
        {"$sort": {"count": -1}},
        {"$limit": 5},
        {"$project": {"day": "$_id", "count": 1, "_id": 0}}
    ], hint="tickets.date_1"
)
```

RD33: Average enrollment rate by workshop title

```
db.activities.aggregate(
    [
        {"$project": {
            "workshop_title": 1,
            "enrollment_rate": {
                "$cond": [
                    {"$gt": ["$capacity", 0]},
                    {"$divide": ["$enrolled", "$capacity"]},
                    None
                ]
            }
        }},
        {"$group": {
            "_id": "$workshop_title",
            "avg_enrollment_rate": {"$avg": "$enrollment_rate"}
        }},
        {"$project": {
            "workshop_title": "$_id",
            "avg_enrollment_rate": 1,
            "_id": 0
        }}
    ]
)
```

RD34: Count visitor education level breakdown from surveys in a given period

```
db.visitors.aggregate(
    [
        {"$match": {
            "surveys.date_of_compilation": {"$gte": <start>, "$lt": <end>}
        }},
        {"$project": {"titles": "$surveys.title_of_studies"}},
        {"$unwind": "$titles"},
        {"$group": {"_id": "$titles", "count": {"$sum": 1}}},
        {"$sort": {"count": -1}},
        {"$project": {"title": "$_id", "count": 1, "_id": 0}},
    ], hint="surveys.date_of_compilation_1"
)
```

RD35: Show workshop duration distribution

```
db.workshops.aggregate(
```

```
    [
        {"$sortByCount": "$duration"},
        {"$project": {
            "_id": 0,
            "duration": {
                "$dateToString": {
                    "format": "%H:%M:%S",
                    "date": "$_id"
                }
            },
            "count": 1
        }}
    ]
)
```

RD36: Count the number of museums vs external suppliers

```
db.suppliers.aggregate(
    [
        {
            "$group": {
                "_id": "$is_museum",
                "count": {"$sum": 1}  # True = Museum
            }
        },
        {"$sort": {"_id": -1}},
        {"$project": {"Museum": "$_id", "count": 1, "_id": 0}},
    ]
)
```

4.3.3 Update operations

UP1: Update the description of a material

```
db.artworks.update_many(
    {"materials": <old_name>},
    {"$set": {"materials.$[m]": <new_name>}},
    array_filters=[{"m": old_name}],
)
```

UP2: Check-in a ticket

```
db.visitors.update_one(
    {"_id": <visitor_id>},
    {"$set": {"tickets.$[t].check_in": True}},
    array_filters=[{"t._id": <ticket_id>}],
)
```

UP3: Update trade information of an artwork

```
db.artworks.update_one(
    {"_id": "La Notte Stellata"},
    {"$set": {
        "trade.win_price": 5000,
        "trade.winner_id": <customer_id>,
    }}
)
```

UP4: Update information of an artwork

```
db.artworks.update_one(
    {"_id": "La Notte Stellata"},
    {"$set": {
        "isOriginal": False,
        "tecniques": ["painting"],
        "materials": ["Glass", "Bronze"],
    }}
)
```

UP5: Update curriculum of an employee

```
db.employees.update_one(
    {"_id": <employee_id>}, {"$set": {"curriculum": <new_curriculum>}}
)
```

UP6: Update employee's department

```
db.employees.update_one({"_id": <employee_id>}, {"$set": {"department": <
    department_id>}})
```

UP7: Update the shift of an given employee

```
db.roles.update_one(
    {"_id": <employee_id>},
    {
        "$set": {
            "shifts.$[s].time_start": <new_time_start>,
            "shifts.$[s].time_end": <new_time_end>,
        }
    },
    array_filters=[{"s.day_of_week": <day_of_week>}],
)
```

UP8: Change the date of a limited event

```
db.limited_events.update_one(
    {"_id": <limited_event_id>},
    {
        "$set": {
            "startDate": datetime(2025, 10, 2, 10, 0),
            "endDate": datetime(2025, 10, 2, 18, 0),
        }
    },
)
```

UP9: Update a message in a given chat

```
db.messages.update_one({"_id": <message_id>}, {"$set": {"content": <new_content
    >}})
```

UP10: Update prices of a workshop

```
db.workshops.update_one(
    {"_id": <workshop_id>}, {"$set": {"price_person": 30.00, "price_class":
    90.00}}
)
```

4.3.4 Delete operations

DL1: Delete a posted comment

```
db.artworks.update_many(
    {},
    {
        "$pull": {
            "comments_star_1": {"_id": <comment_id>},
            "comments_star_2": {"_id": <comment_id>},
            "comments_star_3": {"_id": <comment_id>},
            "comments_star_4": {"_id": <comment_id>},
            "comments_star_5": {"_id": <comment_id>},
        }
    },
)
```

DL2: Delete an artwork

```
db.artworks.delete_one({"_id": "La Notte Stellata"})
```

DL3: Delete a shift of an employee

```
db.roles.update_one(
    {"_id": <employee_id>},
    {"$pull": {"shifts": {"day_of_week": <day_of_week>}}},
)
```

DL4: Delete a message

```
db.visitors.update_many(
    {"_id": <customer_id>},
    {"$pull": {"chat.old_messages": {"_id": <message_id>}}},
)
db.messages.delete_one({"_id": <message_id>})
```

4.4 Conclusion

This project presents a reimplementation of a relational database for the Omero Ancona Museum. Due to evolving requirements in recent years, the current RDBMS has become obsolete.

The standard literature design workflow is followed. First, the conceptual schema is revisited in light of the new requirements. Then, it is translated into the NoAM representation. Lastly, the schema is implemented in MongoDB and the database is seeded to test the implementation of operations.

Thanks to the output of this project, existing mobile and desktop applications can easily transition to MongoDB because the interface offered by the defined operations is coherent. This brings several advantages, especially in terms of performance, scalability, and flexibility.

Appendix A offers a qualitative comparison of alternative designs produced by LLMs.

APPENDIX A

A comparison of alternative solutions designed by LLMs

This experiment compares how GPT-5, Gemini 2.5 Pro, and Claude Opus 4.1 design a NoSQL database using Prompt Chaining and Chain of Thought. The analysis reveals that the three models have distinct strategies for balancing embedding and referencing. GPT is more quantitative and cost-driven, Gemini prioritizes consistency, and Claude offers a balanced approach.

A.1 Experiment overview

In general, the design of NoSQL systems is a poorly systematized topic. Various inconsistent strategies can be found online. For this reason, within the scope of this project, it was deemed interesting to examine how Large Language Models (LLMs) address this issue. Since they were trained in a self-supervised manner on the web, they could produce unexpected results given the particular nature of the topic.

Due to the complexity of the problem, the prompts were written by combining two techniques: *Prompt Chaining* and *Chain of Thought*.

- *Prompt Chaining* (see Figure A.1) consists in **dividing the problem into several sequential tasks, with the advantage of greater control** at each stage, although it requires a greater number of interactions, unlike *Stepwise Prompting*, which condenses requests into a single prompt and all responses into a single generation, leading to lower overall quality (Sun *et al.* [2024]).

- *Chain of Thought*, instead of simply providing input and expecting a response, it involves showing the model examples that demonstrate the reasoning process. This can significantly improve the performance of the LLM in arithmetic and symbolic reasoning tasks. In the context of this experiment, this is very useful for comparing different models not only in terms of results but also in terms of analytical ability (Wei *et al.* [2023]). The Figure A.2 shows how this technique enables the LLM to mimic the intuitive process of human thought when dealing with problems that are divided into several stages.

In more pratical terms, the idea behind this experiment is to give the chosen LLMs the ER diagram, the volume and the operation tables as inputs. They will first be prompted to design the aggregates, and then the collections represented as JSON. The result of the first prompt will

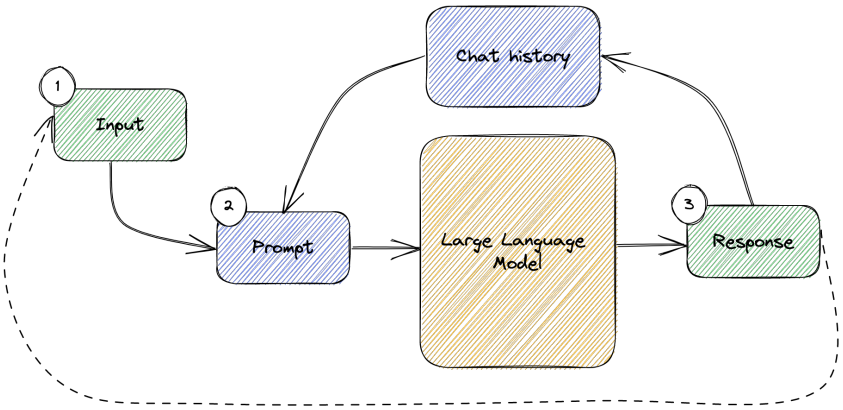


Figure A.1: The flowchart of the *prompt chaining* scenario

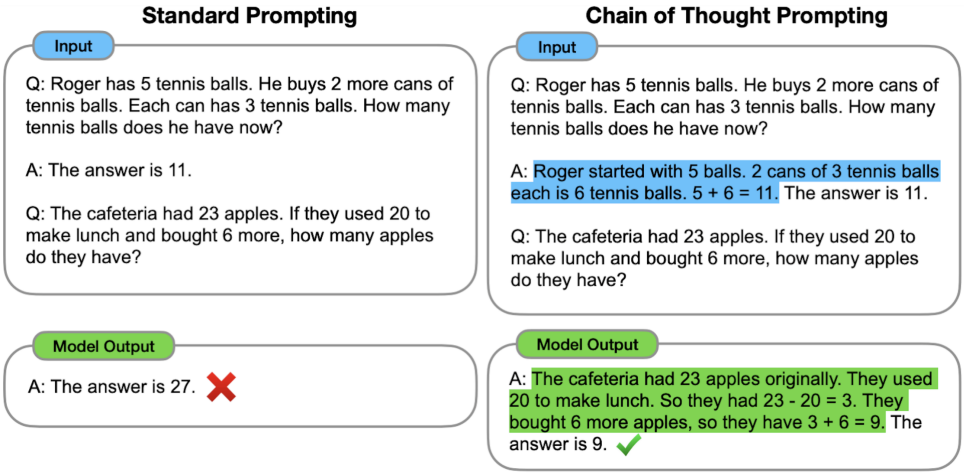


Figure A.2: Output differences between a simple prompt and one with a CoT approach

serve as input for the next, forming a pipeline. The LLMs chosen for comparison are *GPT-5*, *Gemini 2.5 Pro*, and *Claude Opus 4.1*.

The full results are available at the links below, while the following paragraphs constitute the analysis of their content.

- **GPT**: *https://chatgpt.com/share/68b808c1-40bc-800f-b0d0-bd1c39aba1d4*
- **Gemini**: *https://g.co/gemini/share/966756590d84*
- **Claude**: *https://claude.ai/share/efdc40db-ae64-40b9-adfb-44bd2ad14d3c*

A.2 Results and comparison

The analysis below compares the chosen models when making decisions about embedding versus referencing. Each scenario has different logic and motivations, and the goal is to provide a qualitative analysis of the strategies employed, not to create a benchmark. To this purpose, three of the collections where the models showed the greatest discrepancies were selected: `Artwork`, `Visitor`, and `Trade`.

Artwork collection

Starting with *GPT*, with regard to COMMENT, it argues that embedding recent comments is beneficial as it saves approximately 10,500 reading equivalents per day in this model, since readings of ARTWORK are much more numerous than COMMENT writings. With regard to MATERIAL and AUTHOR, it argues that it is convenient to embed very frequent reads, referring to the RD1 operation, but at the same time suggests opting for denormalization: the canonical collections of `Material`, `Technique`, and `Author` are maintained for updates that can be performed lazily or with background work.

Gemini decides to use embedding for MATERIAL, TECHNIQUE, and ROOM because there are few of them, they are stable, and they are read often. For AUTHOR and COMMENT, it has chosen referencing, justifying this with the high number of entries.

Claude reaches the same conclusion of GPT by advocating embedding for the same entities, justifying it with the fact that the 6.5-fold reduction in read operations far outweighs the minimal increase in update complexity, and that the low cardinality (on average 2-3 related entities) and stable nature of these relationships make embedding ideal. As for COMMENT, it decides to keep them as a separate aggregate with references to the CUSTOMER and ARTWORK. The justification provided is that the high frequency of writing (2,000 entries per day) and the need to query COMMENT for both customers and ARTWORK a separate aggregate with references the optimal choice.

An interesting aspect is the management of the ROOM, which is referenced by *GPT* and *Claude* but not by *Gemini*, with the justification that they are stable entities for which it would not make sense, while others argue that the reference is preferable since they are shared among many instances.

Visitor collection

As far as TICKET is concerned, according to *GPT*, `Visitor` must contain references to them, but not complete data. The justification provided is that embedding would reduce joins when TICKET is always accessed from a VISITOR, but most operations are ticket-centric, and visitors have less than one ticket on average, so embedding complicates searches and can make targeted updates more complicated. Its proposed solution is a collection of high-level `Ticket` with a small denormalized `Visitor` snapshot, so the need for joins with `Visitor` is avoided for the most common operations (check-in, scanning, refund).

Gemini has a different opinion, suggesting that references to `Visitor` be included directly in `Ticket`, together with the `Event` one. However, its justification is similar to that of *GPT*, explaining that the TICKET entity has a high volume (800,000) and its own life cycle: this is precisely why it decides to reference it in `Ticket`.

Claude agrees, arguing that with 25,000 daily transactions (the highest number in the system), `Ticket` need independent access and that the nature of the bridge between `Visitor` and `Event`, in addition to frequent status updates (check-ins), makes it an autonomous aggregate that is optimal for performance and consistency.

Trade collection

As for TRADE, *GPT* suggests directly incorporating a small snapshot of the `Artwork` into the `Trade` document, justified by the low volume of transactions and the need for quick access to the work's data when consulting transactions.

Gemini also argues that embedding the entire `Artwork` in `Trade` is inefficient and opts to embed a limited set of easily accessible fields such as *Title* and *Description* to support list views without additional queries. It also argues that embedding `Bid` in `Trade` is very efficient, as reading the `AUCTION` is often accompanied by reading all elements of `BID`.

Finally, *Claude* agrees with Gemini on embedding `Bid` but argues that there should only be referencing for `Artwork`, justifying this with less redundancy.

A.3 Conclusions

Some conclusions can be drawn about the behavior of the models from the analysis of the design of the three previous collections. First, despite probably being trained on common NoSQL design principles, it is clear that each LLM has developed its own strategy for balancing embedding and referencing, which fully reflects the assumption of low topic homogeneity.

GPT, for example, is characterized by a highly quantitative and cost-oriented approach based on an explicit model that translates operations into read and write units. **This strategy often causes GPT to adopt aggressive embedding in situations involving intensive reading and infrequent updates**. GPT accepts a certain amount of redundancy as the price for optimal performance, thus adopting an overall pragmatic approach based on the evidence of its own calculations.

Gemini, on the other hand, favors a more cautious approach, minimizing redundancy and maintaining the separation of entities with high cardinality or independent life cycles. This results in a model that is closer to a "classic" one, in which data consistency and long-term maintainability are prioritized over immediate performance gains.

Finally, Claude strikes a balance: it opts for embedding in the presence of stable, low-cardinality relationships but prefers references when there is a high volume of writes or bridge entities. This reduces the risk of duplication and maintains the scalability of the system.

In conclusion, this experiment clearly shows that all three models agree on the general principle of performing embeddings when the data is stable, closely related, and of low cardinality, and referencing when the objects are dynamic, of high cardinality, or functionally autonomous. The distinction lies in the weight attributed to each factor. Cost and performance are crucial for GPT; Gemini prefers consistency, even at the cost of greater redundancy; and Claude is a middle ground between the two approaches.

Bibliography

ATZENI, P., BUGIOTTI, F., CABIBBO, L. and TORLONE, R. (2020), «Data modeling in the NoSQL world», *Computer Standards and Interfaces*, vol. 67, p. 103 149, URL https://www.sciencedirect.com/science/article/pii/S0920548916301180. (Cited at page 27)

ATZENI, P., CERI, S., FRATERNALI, P., PARABOSCHI, S. and TORLONE, R. (2023), *Database Systems*, McGraw-Hill. (Cited at pages 8, 12, 16 e 37)

MINISTRYOFCULTURE (2016), «Omero State Tactile Museum», http://musei.beniculturali.it/musei?mid=201. (Cited at page 1)

MONGODB (2025a), «JSON and BSON», URL https://www.mongodb.com/resources/basics/json-and-bson. (Cited at page 37)

MONGODB (2025b), «MongoDB Extended JSON (v2)», URL https://www.mongodb.com/docs/manual/reference/mongodb-extended-json/. (Cited at page 37)

MONGODB (2025c), «MySQL to MongoDB Migration Guide», URL https://www.mongodb.com/resources/solutions/use-cases/mysql-to-mongodb.

MORELLI, B. (2023a), «Database for a museum managment», https://github.com/MrPio/MuseoOmero_Database/files/11069638/MuseoOmero_Database.pdf. (Cited at pages iii, 5, 8, 10, 15 e 17)

MORELLI, V. (2023b), «Design and implementation of an app in MAUI technology to support visitors to a tactile museum», https://hdl.handle.net/20.500.12075/15349. (Cited at pages iii, 3 e 7)

NETWORKMUSEUM (2017), «With your eyes and hands», https://web.archive.org/web/20180817102603/http://networkmuseum.com/OLTRE%20IDEA%20DI%20ACCESSIBILITA/170216-GRASSINI%20ALDO/GRASSINI%20ALDO.htm. (Cited at page 1)

SUN, S., YUAN, R., CAO, Z., LI, W. and LIU, P. (2024), «Prompt Chaining or Stepwise Prompt? Refinement in Text Summarization», URL https://arxiv.org/abs/2406.00507. (Cited at page 63)

WEI, J., WANG, X., SCHUURMANS, D., BOSMA, M., ICHTER, B., XIA, F., CHI, E., LE, Q. and ZHOU, D. (2023), «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models», URL `https://arxiv.org/abs/2201.11903`. (Cited at page 63)