

|  |
| --- |
| BUSINESS DESCRIPTION  **museum** |

Contents

[1 Business Description 3](#_Toc62212630)

[1.1 Business background 3](#_Toc62212631)

[1.2 Problems. Current Situation 3](#_Toc62212632)

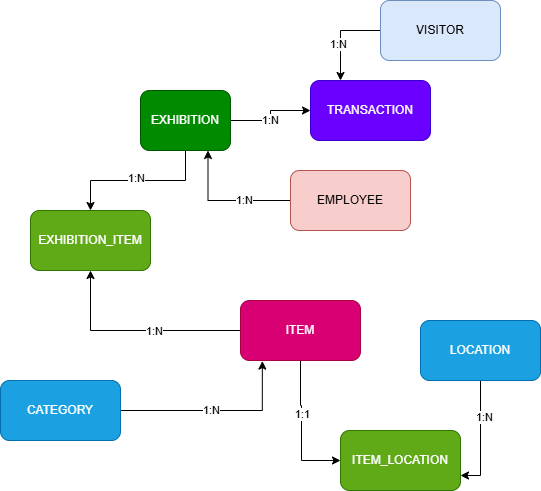
[1.3 The benefits of implementing a database. Project Vision 3](#_Toc62212633)

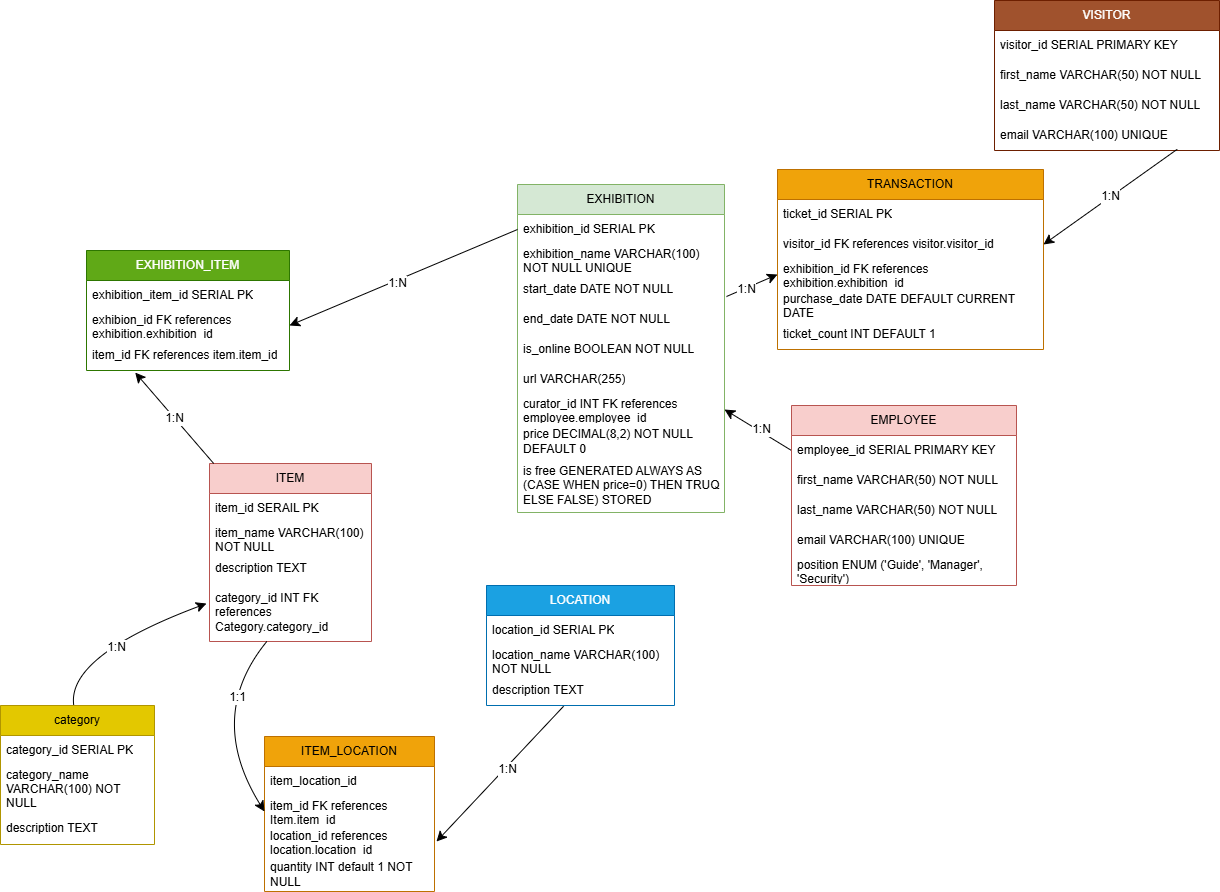
[2 Model description 3](#_Toc62212634)

[2.1 Definitions & Acronyms 3](#_Toc62212635)

[2.2 Logical Scheme 3](#_Toc62212636)

[2.3 Objects 3](#_Toc62212637)





Business Description

### 1. ****Business Description****

The **museum database** is being implemented to address operational inefficiencies and improve overall management of exhibitions, items, transactions, and visitor interactions. It aims to streamline the entire process of tracking and managing items within exhibitions, recording transactions, and maintaining comprehensive records of visitors, employees, and locations. This project will support decision-making, enhance visitor experience, and facilitate the museum's growth and outreach.

#### 1.1 **Business Background**

The museum is a cultural institution dedicated to preserving and displaying art and historical artifacts. It regularly hosts exhibitions, organizes events, and attracts visitors from various regions. The museum's collections include valuable items, each of which requires careful tracking, proper categorization, and location management.

Currently, the museum relies on outdated manual processes and disconnected systems to manage exhibitions, item records, and transactions. The museum staff spends a significant amount of time organizing, searching for, and manually updating records, leading to inefficiencies and potential errors.

The lack of a centralized database results in difficulties when tracking items across exhibitions, managing visitor data, and generating reports for management and curatorial teams. There is a strong need for a unified system that can automate these tasks, integrate various departments, and provide real-time data access.

#### 1.2 **Problems and Current Situation**

The museum is facing several challenges with its current manual systems:

* **Data Fragmentation**: Information about items, exhibitions, and visitors is stored in multiple places, including paper records, spreadsheets, and disparate software, making it difficult to access and manage.
* **Operational Inefficiency**: Museum staff must manually update multiple systems, leading to wasted time and potential errors. Tracking the movement of items between locations is especially cumbersome.
* **Inconsistent Data**: The lack of a structured database has led to data inconsistencies, making it hard to verify the current status of an exhibition or track the history of an item.
* **Visitor Management**: The museum struggles to track visitor data efficiently, which impacts marketing, membership programs, and the ability to provide personalized services.

#### 1.3 **The Benefits of Implementing a Database and Project Vision**

Implementing a centralized database will address the current challenges and provide several key benefits:

* **Improved Operational Efficiency**: A relational database will automate manual processes such as item tracking, ticket sales, and visitor management. This will free up staff time for more important tasks and reduce the risk of human error.
* **Centralized Data Management**: All records related to exhibitions, items, locations, and transactions will be housed in one place, providing staff with real-time, accurate data that is easy to access and update.
* **Enhanced Reporting and Analytics**: The database will enable the generation of detailed reports on visitor numbers, ticket sales, exhibition performance, and inventory, allowing the museum to make data-driven decisions.
* **Streamlined Item and Location Tracking**: With the ITEM\_LOCATION table and the EXHIBITION\_ITEM table, the museum will be able to easily track where each item is located and how it moves between exhibitions or locations.
* **Scalability and Growth**: As the museum expands its collections and exhibitions, the database will scale with it, handling increasing amounts of data without compromising performance. The database will be able to support new data types and processes as the museum grows.
* **Improved Visitor Engagement**: By managing visitor data more efficiently, the museum will be able to offer tailored experiences, such as personalized exhibition recommendations or membership offers.
* **Project Vision**: The long-term vision for the project is to create a fully integrated, data-driven environment that supports the museum’s mission of promoting education, culture, and engagement with the public. This will allow the museum to modernize its operations, enhance its service offerings, and increase overall efficiency.

This write-up outlines the **museum database** project, highlighting the **background**, **current issues**, and the **benefits** of moving to a more modern, efficient, and scalable solution. If you'd like more detailed content on a specific section or additional information, feel free to ask!

Model description

Definitions & Acronyms

**PK**: Primary Key - A unique identifier for each record in a table

**FK**: Foreign Key - A field that refers to the primary key in another table

**1:1**: One-to-One relationship - Each record in Table A relates to exactly one record in Table B

**1:M**: One-to-Many relationship - One record in Table A can relate to multiple records in Table B

**M:N**: Many-to-Many relationship - Multiple records in Table A can relate to multiple records in Table B

**AUTO\_INCREMENT**: A property that automatically generates a unique number for new records

**NOT NULL**: Constraint that ensures a column cannot have a NULL value

**UNIQUE**: Constraint that ensures all values in a column are different

**DEFAULT**: Provides a default value for a column when none is specified

**ENUM**: A data type that allows a column to have one value from a predefined list

### ****Step By step approach****

**Identify Entities**:  
Start by identifying the key entities such as **Item**, **Exhibition**, **Visitor**, and **Employee**. These entities represent the core components of the museum database.

**Define Relationships**:  
Establish relationships between the tables. For instance:

* **Item** and **Exhibition** have a many-to-many relationship, which is resolved through the **Exhibition\_Item** junction table.
* **Visitor** and **Transaction** have a one-to-many relationship, where one visitor can make multiple transactions.

**Create Junction Tables**:  
Use junction tables like **Exhibition\_Item** to resolve many-to-many relationships between entities. This table links **Exhibition** and **Item** to track which items are part of which exhibitions.

**Define Attributes**:  
For each table, define the necessary fields (attributes) such as item\_id, exhibition\_name, and ticket\_count. Also, ensure the correct data types and constraints are set, such as SERIAL, VARCHAR, INT, and BOOLEAN.

**Normalization**:  
Apply database normalization rules (1NF, 2NF, 3NF) to avoid data redundancy and ensure data integrity. Each table should contain atomic values, and relationships should be properly represented using foreign keys. **Implement Business Rules**:  
Define business rules using constraints. For example, ensure that **Exhibition** cannot end before it starts, or that an item can only be in one category.

**Translate to SQL**:  
Create tables with the appropriate fields, data types, constraints, and foreign keys. Use SQL commands to set up the schema and enforce relationships.

**Test and Refine**:  
Add sample data to test the schema. Ensure queries perform as expected and refine the structure based on the testing.

# LOCATION

Tracks where each item is located within the museum or exhibition space.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| location\_id | Unique identifier for each location. | SERIAL, PK | Not Null |
| location\_name | Name of the location (e.g., ‘Hall 1’). | VARCHAR(100) | Not Null |
| description | Description of the location (e.g., ‘Main exhibition hall’). | VARCHAR(255) |  |

**Relationships:**

**1:N relationship** between **LOCATION** and **ITEM\_LOCATION**:

* A single location can store many items, as multiple items can be placed in the same location.

# Example Data for LOCATION

|  |  |  |
| --- | --- | --- |
| location\_id | location\_name | description |
| 1 | Hall 1 | Main exhibition hall |
| 2 | Hall 2 | Special art exhibition |
| 3 | Storage | Storage for archival items |

# CATEGORY

Defines different categories of items, such as paintings, sculptures, and photographs.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| category\_id | Unique identifier for each category. | SERIAL, PK | Not Null |
| category\_name | Name of the category (e.g., 'Paintings'). | VARCHAR(100) | Not Null |
| description | Description of the category (e.g., 'Works of art'). | TEXT |  |

**Relationships:**

* One CATEGORY has many ITEM (1:N relationship)

|  |  |  |
| --- | --- | --- |
| category\_id | category\_name | description |
| 1 | Paintings | Artworks created by applying pigment on a surface |
| 2 | Sculptures | Three-dimensional works of art |
| 3 | Photographs | Images captured by cameras |

# ITEM

Contains details about the items (artifacts, artworks) in the museum, including their category.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| item\_id | Unique identifier for each item. | SERIAL, PK | Not Null |
| item\_name | Name of the item (e.g., 'Mona Lisa'). | VARCHAR(100) | Not Null |
| Description | Description of the item. | TEXT |  |
| category\_id | Foreign key referencing CATEGORY.category\_id. | INT, FK → CATEGORY.category\_id | Not Null |

**Relationships:**

* Many ITEM belong to one CATEGORY (N:1)
* Many ITEM can be in many EXHIBITION via EXHIBITION\_ITEM (M:N)
* Each ITEM can have one ITEM\_LOCATION records (1:1)
* Between item and exhibitoin\_item is 1:N relationship.

|  |  |  |  |
| --- | --- | --- | --- |
| item\_id | item\_name | description | category\_id |
| 1 | Mona Lisa | Famous painting by Leonardo da Vinci | 1 |
| 2 | The Thinker | Sculpture by Auguste Rodin | 2 |
| 3 | Sunset View | Photograph by Ansel Adams | 3 |

# EXHIBITION

Stores information about exhibitions, including dates, curator, and whether it's online or in person.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| exhibition\_id | Unique identifier for each exhibition. | SERIAL, PK | Not Null |
| exhibition\_name | Name of the exhibition (e.g., 'Modern Art Showcase'). | VARCHAR(100) | Not Null |
| start\_date | Start date of the exhibition. | DATE | Not Null |
| end\_date | End date of the exhibition. | DATE | Not Null |
| is\_online | Whether the exhibition is online. | BOOLEAN | Not Null |
| url | URL to the online exhibition (if applicable). | VARCHAR(255) |  |
| curator\_id | Foreign key referencing EMPLOYEE.employee\_id (curator). | INT, FK → EMPLOYEE.employee\_id | Not Null |
| price | Price of the exhibition. | DECIMAL(8,2) | Default 0 |
| is\_free | Whether the exhibition is free (calculated as price = 0). | BOOLEAN | Generated Always (CASE WHEN price = 0 THEN TRUE ELSE FALSE END) |

**Relationships:**

* One EXHIBITION is curated by one EMPLOYEE (1:N) but employee can curate multipe Exhibition
* One EXHIBITION has many ITEM through EXHIBITION\_ITEM (M:N)
* Bettween Exhibition and Exhibition\_item is 1:N relatinship.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| exhibition\_id | exhibition\_name | start\_date | end\_date | is\_online | url | curator\_id | price | is\_free |
| 1 | Renaissance Art | 2025-01-01 | 2025-06-30 | True | https://renaissanceart.museum | 1 | 20.0 | False |
| 2 | Modern Sculptures | 2025-03-15 | 2025-07-15 | False | https://sculptures.museum | 2 | 15.0 | False |
| 3 | Nature Photography | 2025-02-01 | 2025-05-01 | True | https://naturephoto.museum | 3 | 0.0 | True |

# EXHIBITION\_ITEM

Links items and exhibitions to manage the many-to-many relationship between them.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| exhibition\_id | Foreign key referencing EXHIBITION.exhibition\_id. | INT, PK, FK → EXHIBITION.exhibition\_id | Not Null |
| item\_id | Foreign key referencing ITEM.item\_id. | INT, PK, FK → ITEM.item\_id | Not Null |

**Relationships:**

* Links many EXHIBITION and many ITEM (resolves M:N relationship)
* Between Item and Exhibition\_Item is 1:N relationship
* Between Exhibition and Exhibition\_Item is 1:N relationship

|  |  |
| --- | --- |
| exhibition\_id | item\_id |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

# ITEM\_LOCATION

Records the movement and quantity of items in different locations within the museum.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| item\_location\_id | Unique identifier for each item-location relationship. | SERIAL, PK | Not Null |
| item\_id | Foreign key referencing ITEM.item\_id. | INT, FK → ITEM.item\_id | Not Null |
| location\_id | Foreign key referencing LOCATION.location\_id. | INT, FK → LOCATION.location\_id | Not Null |
| quantity | Number of items at the specific location. | INT DEFAULT 1 | Not Null |

**Relationships:**

* In one location can be multiple items. So its 1 to many relationship between item\_location and location and between location and item M:N.

|  |  |  |  |
| --- | --- | --- | --- |
| item\_location\_id | item\_id | location\_id | quantity |
| 1 | 1 | 1 | 3 |
| 2 | 2 | 2 | 5 |
| 3 | 3 | 3 | 2 |

# VISITOR

Holds details about visitors, including their personal information and email.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| visitor\_id | Unique identifier for each visitor. | SERIAL, PK | Not Null |
| first\_name | First name of the visitor. | VARCHAR(50) | Not Null |
| last\_name | Last name of the visitor. | VARCHAR(50) | Not Null |
| email | Email address of the visitor (unique). | VARCHAR(100) | Unique |

Relationships:

* One VISITOR can have many TRANSACTION entries (1:N)

|  |  |  |  |
| --- | --- | --- | --- |
| visitor\_id | first\_name | last\_name | email |
| 1 | John | Doe | john.doe@email.com |
| 2 | Jane | Smith | jane.smith@email.com |
| 3 | Alice | Johnson | alice.johnson@email.com |

# EMPLOYEE

Contains information about museum employees, including their roles (e.g., Guide, Manager).

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| employee\_id | Unique identifier for each employee. | SERIAL, PK | Not Null |
| first\_name | First name of the employee. | VARCHAR(50) | Not Null |
| last\_name | Last name of the employee. | VARCHAR(50) | Not Null |
| email | Email address of the employee (unique). | VARCHAR(100) | Unique |
| position | Position of the employee (e.g., 'Guide', 'Security', 'Manager'). | ENUM('Guide', 'Security', 'Manager') | Not Null |

**Relationships:**

* One EMPLOYEE can curate many EXHIBITION (1:N)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| employee\_id | first\_name | last\_name | email | position |
| 1 | Michael | Green | michael.green@email.com | Manager |
| 2 | Sarah | Brown | sarah.brown@email.com | Guide |
| 3 | David | White | david.white@email.com | Security |

# TRANSACTION

Tracks the transactions made by visitors for exhibitions, including ticket purchases.

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Description | Data Type | Constraints |
| transaction\_id | Unique identifier for each transaction. | SERIAL, PK | Not Null |
| visitor\_id | Foreign key referencing VISITOR.visitor\_id. | INT, FK → VISITOR.visitor\_id | Not Null |
| exhibition\_id | Foreign key referencing EXHIBITION.exhibition\_id. | INT, FK → EXHIBITION.exhibition\_id | Not Null |
| ticket\_count | Number of tickets purchased. | INT DEFAULT 1 | Not Null |
| purchase\_date | Date of the transaction. | DATE | Not Null |

**Relationships:**

* One VISITOR can have many TICKET\_PURCHASE entries (1:N)
* One EXHIBITION can have many TICKET\_PURCHASE entries (1:N)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| transaction\_id | visitor\_id | exhibition\_id | ticket\_count | purchase\_date |
| 1 | 1 | 1 | 2 | 2025-01-10 |
| 2 | 2 | 2 | 1 | 2025-03-01 |
| 3 | 3 | 3 | 3 | 2025-02-15 |