

|  |
| --- |
| Business Template  **Subject areas** |
| **Logo / Image** |

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

## Business background

Overview of the museum, including its purpose (e.g., preserving cultural heritage, scientific collections, or artistic exhibits).Information about the scale of the museum: the number of artifacts, visitors, staff, and exhibitions annually.

## Problems. Current Situation

The museum struggles with tracking the location, condition, and details of its artifacts. Artifacts moved between storage and exhibition areas are often not updated in a centralized system, leading to confusion and delays.

**Exhibition Planning Challenges:**  
Planning exhibitions, especially temporary ones, is labor-intensive and time-consuming due to fragmented and incomplete information. This affects the coordination between teams and limits the museum's ability to expand online exhibitions.

## the Benefits of implementing a database. Project Vision

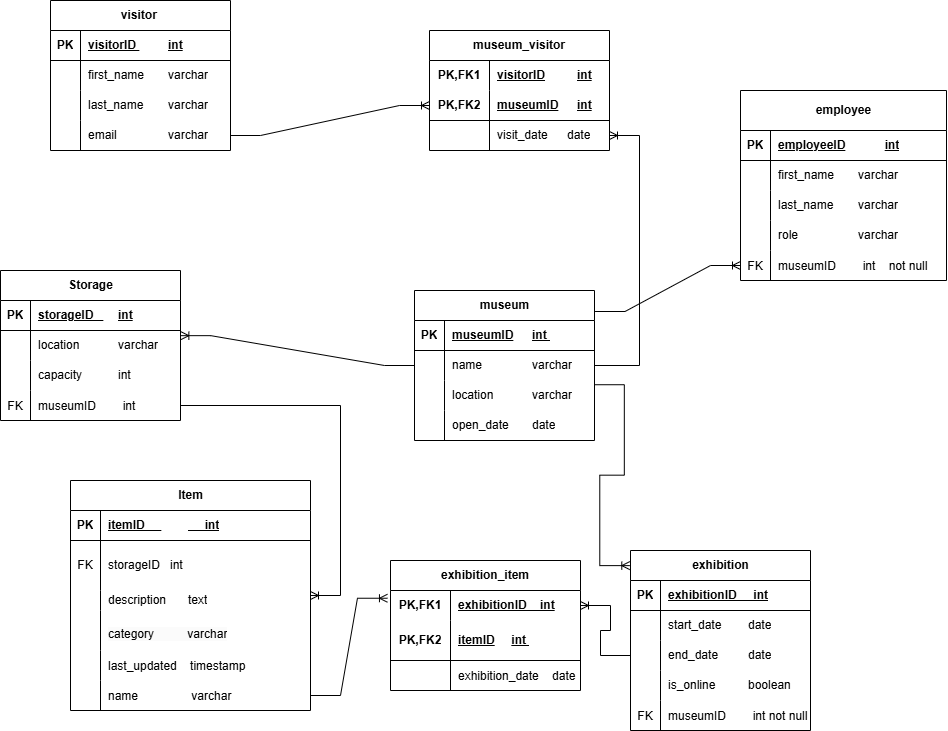
**Centralized Artifact Management:**  
A unified platform will provide up-to-date information on all items, including their location, condition, and exhibition history. This will reduce duplication of effort and make it easier to access critical details about the collection.

**Streamlined Exhibition Planning:**  
The database will simplify the planning and coordination of exhibitions, ensuring all relevant information is easily accessible. Integration with online platforms will also make it easier to promote exhibitions and engage with a global audience.

# Model description

## Definitions & Acronyms

## Logical Scheme



## Objects

Table Description

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| item | itemID | PK | Int |
| storageID | FK | Int |
| category |  | Varchar |
|  | last\_updated |  | timestamp |
|  | description |  | text |
|  | Name |  | Varchar(not null) |

Table item has many-to-many relationship with table exhibition since there can be many items exhibited at one exhibition and one item can be exhibited at many exhibitions. Also table item has one-to many relationship with table Storage and because of that we need storageID as FK in table item.

We have also non-prime attributes which are:category- to know in which category an item belongs,for example is it artwork, artifact or even historical object. Also we have last\_updated to keep track when we lastly updated some information about an item and description tells us the history of an item.

Now I want to talk about storageID part. Some items can be at the museum and not in storage because there are exhibitions. storageID here shows where is item saved when it’s not any exhibition.We can can information about item with tables museum->exhibition->exhibition\_item->item.

I think this is the best way to do it.

Example with data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| itemID | Name | Description | category | storageID | Last\_updated |
| 1 | “Mona-Lisa” | A Renaissance painting by Leonardo da Vinci, depicting a woman with a mysterious smile | “artwork” | 1 | 2024-12-01  10:30:00 |
| 2 | “Rosetta Stone” | “An ancient Egyptian stone inscribed with three scripts that enabled the decoding of hieroglyphs.” | “historical object” | 1 | 2024-11-30  12:00:00 |
| 3 | “Ancient Greek Vase” | “A clay vase from the 5th century BCE featuring black-figure pottery artwork.” | “artifact” | 2 | 2024-12-01  11:00:00 |
| 4 | Medieval Sword | “ A 12th-century knight's sword, crafted with ornate engravings.” | “historical object” | 5 | 2024-12-01  14:45:00 |
| 5 | “Egyptian Sculpture” | “An ancient Egyptian stone sculpture of a pharaoh” | “artifact” | 3 | 2024-12-02  09:30:00 |
| 6 | “Renaissance Vase” | “A beautifully painted vase from the Renaissance period with floral motifs” | “artifact” | 4 | 2024-12-02  10:00:00 |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Exhibition | exhibitionID | PK | Int |
| start\_date |  | Date (constraint-start\_date<end\_date) |
| End\_date |  | date |
| Is\_online |  | boolean |
|  | museumID | FK | Int (not null) |

Table exhibition has many-to-many relationship with table item, because at one exhibition there can be many items exhibited and one item can also be exhibited at many exhibitions.it also has one-to-many relationship with table museum since one exact exhibition can be held at one museum and in one museum there can be many exhibitions helded. It has 3 non-prime attributes. First one is start\_date of the exhibition. with help of that attribute we save information when exhibition started and with end\_date we get information when it ended. Since exhibition can be helded online we have is\_online attribute to know is it helded online or offline.

Example with data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| exhibitioID | Start\_Date | End\_date | isOnline | museumID |
| 1 | 2024-11-10 | 2024-11-12 | false | 1 |
| 2 | 2024-10-01 | 2024-10-04 | True | 1 |
| 3 | 2024-11-01 | 2024-11-05 | False | 2 |
| 4 | 2024-10-15 | 2024-10-18 | True | 3 |
| 5 | 2024-11-25 | 2024-11-30 | False | 1 |
| 6 | 2024-10-20 | 2024-10-25 | True | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| museum | museumID | PK | Int |
| name |  | Varchar (not null) |
| location |  | varchar |
| Open\_date |  | date |

Since database is about the museum this table is the main table in this logical scheme. I created this table because this logical scheme is not for one museum, it also includes other museums data too.

museumID is its primary key and it uniquely describes each museum. Museum has its name that’s why I created attribute name and it can’t be empty. Location tells us where museum is located and open\_Date gives us information about when museum firstly opened.

It has one-to-many relationship with table storage since one museum can have many storages and one storage is for one museum.I will explain the logic of this. Big supermarkets have many storages and they take product from there so I thought it would be the same for museums too.

Table museum also has one-to-many relationship with table visitor because one visitor can visit only one museum at one time and museum can be visited by many visitors.

It also has one-to-many relationship with table employee since at one museum there are many employees and one employee works at only one museum.

And lastly it has one-to-many relationship with table exhibition since there can be multiple exhibitions at one museum and one exhibition can be helded at one museum.

Example with data:

|  |  |  |  |
| --- | --- | --- | --- |
| museumID | Name | Location | Open\_date |
| 1 | “National art museum” | “new-york” | 1950-06-01 |
| 2 | “National history  Museum” | “London” | 1881-04-18 |
| 3 | “Museum of modern art” | “Paris” | 1929-10-07 |
| 4 | “Science and technology museum” | “Tokyo” | 1985-03-12 |
| 5 | “Ancient Civilization Museum” | “Cairo” | 1902-11-15 |
| 6 | “Space Exploration Museum” | “Houston” | 1976-09-20 |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| visitor | visitorID | PK | Int |
| First\_name |  | Varchar |
| Last\_name |  | Varchar |
| email |  | varchar |

We needed information about visitors so that’s why I created table visitor.it has visitorID as PK and it uniquely describes each visitor. Also visitor has his/her first\_name, last\_name and email and I added that attributes in the table.

Table visitor has many-to-many relationship with table museum since one visitor can visit many museums and one museum can be visited by many visitors.

Example with data:

|  |  |  |  |
| --- | --- | --- | --- |
| visitorID | First\_name | Last\_name | Email |
| 1 | “Alice” | “Johnson” | alice.johnson@example.com |
| 2 | “Bob” | “Smith” | bob.smith@example.com |
| 3 | “Charlie” | “Brown” | charlie.brown@example.com |
| 4 | “Diana” | “Evans” | diana.evans@example.com |
| 5 | “Ethan” | “Martinez” | ethan.martinez@example.com |
| 6 | “Fiona” | “Taylor” | fiona.taylor@example.com |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| storage | storageID | PK | Int |
| location |  | Varchar |
| capacity |  | Int(capacity>0) |
| museumID |  | Int(not null) |

Table storage has storageID as PK and uniquely identifies each storage. Storage also has its location

And capacity. Also we need to track of which museums storage is that so we need to add museumID

As FK in this table.

Table storage has one-to-many relationship with table museum because one storage belongs to only one museum and one museum can have many storages.

It also has one-to-many relationship with table item since there can be many items at one storage

And one item is exactly at one storage. But If the item is in museum its storageID can be null since it’s in the museum and not in storage.

Example with data:

|  |  |  |  |
| --- | --- | --- | --- |
| StorageID | Location | Capacity | museumID |
| 1 | “Basement, Wing A” | 500 | 1 |
| 2 | “Annex Building, Room 3” | 300 | 1 |
| 3 | “Underground Vault” | 800 | 2 |
| 4 | “Main Warehouse” | 1000 | 3 |
| 5 | “Off-Site Facility 1” | 600 | 4 |
| 6 | “Secure Archive Room” | 200 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| employee | employeeID | PK | Int |
| First\_name |  | varchar |
| Last\_name |  | Varchar |
| Role |  | Varchar |
|  | museumID | FK | Int (not null) |

Table employee has employeeID as PK since it uniquely identifies each employee. Employee also has first\_name,last\_name and role. We need to track which employee works at which museum so we need

museumID as FK in the table.

table employee has one-to-many relationship with table museum because one employee can work at one museum and one museum can have many employees.

Example with data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EmployeeID | First\_name | Last\_name | Role | museumID |
| 1 | “Emma” | “Wilson” | “curator” | 1 |
| 2 | “James” | “Anderson” | “archivist” | 1 |
| 3 | “Sophia” | “Thomas” | “Guide” | 2 |
| 4 | “Liam” | “Roberts” | “Security Officer” | 3 |
| 5 | “Olivia” | “Clark” | “conservator” | 4 |
| 6 | “Noah” | “Harris” | “researcher” | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Museum\_visitor | visitorID | PK,FK | Int |
| museumID | PK,FK | Int |
| Visit\_date |  | Date(visit\_date>July 1  2024) |

Museum\_visitor is bridge table between tables museum and visitor since they have many-to-many relationships. I added one more attribute visit\_date to store information about the when visitor visited museum.

Example with data:

|  |  |  |
| --- | --- | --- |
| visitorID | museumID | Visit\_date |
| 1 | 1 | 2024-11-25 |
| 2 | 1 | 2024-10-01 |
| 3 | 2 | 2024-10-16 |
| 4 | 3 | 2024-11-26 |
| 5 | 4 | 2024-10-22 |
| 6 | 2 | 2024-10-23 |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Field name | Field Description | Data Type |
| Exhibition\_item | exhibitionID | PK,FK | Int |
| itemID | PK,FK | Int |
| exhibition\_date |  | Date(<=Current\_date) |

This is a bridge table between two tables exhibition and item since they have many-to-many relationship. I added one more attribute exhibition\_date to track when the exhibitions were held for these items.

Example with data:

|  |  |  |
| --- | --- | --- |
| exhibitioID | itemID | Exhibition\_date |
| 5 | 1 | 2024-11-25 |
| 2 | 2 | 2024-10-01 |
| 4 | 3 | 2024-10-16 |
| 5 | 4 | 2024-11-26 |
| 6 | 5 | 2024-10-22 |
| 6 | 4 | 2024-10-23 |