Sira Dariia – 119436

DOCUMENTATION OF THE MUSEUM'S SYSTEM FOR RECORDING ARTIFACTS

Task Description

The museum owns some of the specimens, but some specimens are on loan from other institutions (eg another museum, private collection…). Also, the museum itself may contain specimens borrowed from another institution.

Therefore, it is important for the museum to keep track of the condition of each specimen, for example, whether it is its own specimen or borrowed, as well as in what state this copy is in, for example, a borrowed copy on the way, on display, in storage, on the way back, etc.

Single instances also belong to different categories. Each specimen is always assigned a certain category, based on which the museum can conduct a search among individual specimens.

The museum also organizes various exhibitions, which include various specimens. The specimen itself can be exhibited only within one exposure, so it is impossible for one specimen to be in two different exposures at the same time.

The system should also consider the fact that if the specimen is on the way, it cannot be exhibited at this time. The museum keeps only approximate information about when a specimen is expected to arrive and, in the case of specimens, control processes are also carried out if the specimen has been loaned to another institution. Each instance requires a different control time, and these controls are also logged along with their duration.

Since the museum has several rooms/zones, the exhibition can take place within one or more rooms. However, it is impossible to have two exposures at the same time within the same zone.

The museum also keeps track of individual exhibitions that are being held, as well as those that are under preparation. The museum also maintains a record of all historical exhibits, along with a general overview of the exhibited specimens.

Processes in the Museum

1. Exhibition Planning

This process involves planning and organizing exhibitions within the museum.

Steps:

1. Create a new exhibition in the Exhibitions table.
2. Add artifacts to be displayed in the exhibition to the Displayed_Artifacts table, specifying the exhibition_id, artifact_id, and room_id.

SQL Queries:

```
-- 1. Create a new exhibition
INSERT INTO Exhibitions (name, description, start_date, end_date,
exhibition_status)
VALUES ('Exhibition Name', 'Exhibition Description', 'Start Date', 'End
Date',
'Exhibition Status');

-- 2. Add artifacts to be displayed in the exhibition
INSERT INTO Displayed_Artifacts (exhibition_id, artifact_id, room_id)
VALUES (exhibition_id_value, artifact_id_value, room_id_value);
```

Listing 1 – Exhibition Planning

2. Adding a New Artifact

This process involves adding a new artifact to the museum's collection.

Steps:

1. Insert a new record into the Artifacts table with relevant details such as name, description, control_id, and artifact_status.

SQL Queries:

```
INSERT INTO Artifacts (name, description, control_id, artifact_status)
VALUES ('Artifact Name', 'Artifact Description', control_id_value,
'Artifact
Status');
```

Listing 2 – Adding a New Artifact

3. Artifact Movement to Another Zone

This process involves relocating an artifact to a different room within the museum.

Steps:

1. Update the room_id of the artifact in the Displayed_Artifacts table.

SQL Queries:

```
UPDATE Displayed_Artifacts
SET room_id = new_room_id
WHERE artifact_id = artifact_id_value;
```

Listing 3 – Artifact Movement to Another Zone

4. Getting of Artifacts from Another Institution

This process involves receiving an artifact from another entity such as another museum or private collection.

Steps:

1. Insert a new record into the Rental table.
2. Update the rental_status of the artifact to reflect its status.
3. Insert or update a new record into the Owner table if needed.

SQL Queries:

```
-- 1. Insert a new record into Rental table
INSERT INTO Rental ("artifact_id", "owner_id", "rental_status",
"rental_date", "return_date")
VALUES ("Artifact id", "Owner id", "Rental status", "Rental date", "Return
date");

-- 2. Update the rental_status
UPDATE Rental
SET rental_status = 'borrowed',
    owner_id = owner_id_value,
    ...
WHERE artifact_id = artifact_id_value;
```

Listing 4 – Getting of Artifacts from Another Institution

5. Loaning Artifacts to Another Institution

This process involves integrating an artifact received from another entity into the museum's collection.

Steps:

1. Insert a new record into the Rental table.
2. Update the rental_status of the artifact to reflect its status.
3. Insert or update a new record into the Owner table if needed.

SQL Queries: (Same as step 4, but `rental_status = 'lent'`)

Database Description

Table: Artifacts

- **artifact_id (int):** The *artifact_id* field is selected as the unique artifact identifier, restricted to primary key, and will be automatically incremented. It references the *artifact_id* field of the **Rental** table establishing a one-to-many relationship, because each artifact has only one rental relationship as well as owner.
- **name (varchar(255)):** The **name** field represents the unique name of the artifact to ensure uniqueness and prevent insertion of empty values.
- **description (text):** Description of the artifact, providing additional details or context. This field can be left empty.
- **control_id (int):** The **control_id** field is a foreign key associated with the Control table, which allows each artifact to be associated with the corresponding control information. Each artifact is associated with one control process.
- **artifact_status (int):** he **artifact_status** field indicates the status of the artifact and checks that the value belongs to the set of valid statuses: 'in_transit', 'on_display', 'in_storage'. The column contains a constraint for *artifact_status* using check to ensure that artifact status only accepts certain values.

This table stores information about various artifacts possessed by the museum, including their names, descriptions, statuses, owners, and control processes.

Table: Control

- **control_id (int):** Unique identifier for each control process. Primary key of the table. Autoincremented for each new entry.
- **control_date (date):** Date indicates the date of the control process. This column contains the check control date is withing date of shipping and delivering.
- **checked (boolean):** Indicates whether the artifact associated with the control process has been checked.
- **description (text):** Description of the condition of the artifact. This field can be left empty.
- **shipping_date (date):** Date when the artifact associated with the control process was shipped.

- **delivery_date (date):** Date when the artifact associated with the control process was delivered.

This table is linked to the Artifacts table by the *control_id* field, providing tracking of control processes for each artifact.

The museum keeps information about when the item is expected to arrive and, in the case of specimens, also the control processes in case the item has been lent to another institution. Each item requires a different control time, and these controls are also recorded along with their duration. Info about transit date is in Control table because the item can be owned by museum and placed in storage, so it is also needed to keep information about delivering as well as borrowed artifacts.

Table: Rental

- **rental_id (int):** Unique identifier for each renting. Primary key of the table. Auto-incremented for each new entry.
- **artifact_id (int):** Identifies the artifact associated with ownership, referencing the artifact_id in the Artifacts table.
- **owner_id (int):** Identifies the owner of the artifact. It references the *rental_id* field of the Owner table establishing a many-to-one relationship, because each renting record have only one owner, but owners can lend a lot of artifacts.
- **rental_status (int):** Represents the temporary exchange of an artifact, which can be borrowed, lent or owned by this museum. The column contains a constraint for *rental_status* using check to ensure that rental status only accepts certain values.
- **borrow_date (date):** Date when the artifact was borrowed by the owner.
- **return_date (date):** Date when the artifact is expected to be returned by the owner. This and previous columns constraint checks that dates conform to ISO8601 format and dates cannot be NULL, and that dates are in UTC format.

This table maintains records of rental details for artifacts, including whether they are adopted, borrowed dates, and return dates. The Rental table establishes a many-to-one relationship between artifacts and their owners, since each artifact can have only one owner, but an owner can have many artifacts.

It is possible to add new rental records with the same artifacts with another exhibition period. An artwork itself can only be exhibited within one exposition, so it is impossible for one artwork to be in two different expositions at the same time. To escape this situation, it is recommended to check data period of each exhibition.

This table is for each artifact as well as items owned by museum, which are not adopted. In this case the *owner_id* is id of directly the museum, consequently the borrow and return date can be null.

Table: Owners

- **owner_id (int):** Unique identifier for each owner. Primary key of the table. Auto-incremented for each new entry.
- **name (varchar(255)):** Name of the owner. It's a non-null field and must be unique across all owners.
- **contact_phone (varchar(20)):** Phone number of the owner. This field can be left empty.
- **contact_email (varchar(100)):** Email address of the owner. This field can be left empty.
- **address (text):** Address of the owner. This field can be left empty.

This table provides information about the owners of artifacts, including their names, contact details, and addresses.

Table: Exhibitions

- **exhibition_id (int):** Unique identifier for each exhibition. Primary key of the table. Autoincremented for each new entry.
- **name (varchar(255)):** Name of the exhibition. It's a non-null field and must be unique across all exhibitions.
- **description (text):** Description providing additional details or context about the exhibition. This field can be left empty.
- **start_date (date):** Start date of the exhibition.
- **end_date (date):** End date of the exhibition.
- **exhibition_status (int):** Indicates the status of the exhibition, which can be preparation, active or passed to manage with another triggers.

The museum keeps records of individual exhibitions that are ongoing and those that are in preparation. The museum also keeps a record of all historical

exhibitions, along with a general overview of the exhibited items. This table stores information about exhibitions held by the museum, including their names, duration, statuses, and associated rooms.

Data columns constraint checks that dates conform to ISO8601 format and dates cannot be NULL, and that dates are in UTC format.

Table: Displayed_Artifacts

- **exhibition_id (int):** Identifies the exhibition in which an artifact is displayed, referencing the exhibition_id in the Exhibitions table. Each artifact may be exhibited in one or more exhibitions.
- **artifact_id (int):** Identifies the artifact being displayed, referencing the artifact_id in the Artifacts table. Each artifact may be exhibited in one or more exhibitions.
- **room_id (int):** Identifies the room where the exhibition takes place, referencing the room_id field of the Rooms table in a many-to-one relationship. Each exhibition takes place in one room, but one room can contain several exhibitions.

The table indicating which artifacts are displayed in which exhibitions. This table establishes a many-to-many relationship between exhibitions and artifacts because each artifact can be displayed on many exhibitions, and each exhibition can contain many artifacts.

Since the museum has several rooms, an exhibition can take place within one or more rooms. However, it is impossible for two exhibitions to take place simultaneously within the same room. To escape this situation, it is recommended to check data period of each exhibition.

The system should also consider the fact that if an item is in transit, it cannot be displayed at that time.

Table: Rooms

- **room_id (int):** Unique identifier for each room. Primary key of the table. Auto-incremented for each new entry.
- **name (varchar(100)):** Name of the room. It's a non-null field and must be unique across all rooms.
- **description (text):** Description providing additional details or context about the room. This field can be left empty.

This table contains information about the rooms within the museum where exhibitions can be held.

Table: Categories

- **category_id (int):** Unique identifier for each category. Primary key of the table. Autoincremented for each new entry.
- **name (varchar(100)):** Name of the category. It's a non-null field and must be unique across all categories.
- **description (text):** Description providing additional details or context about the category. This field can be left empty.

This table maintains different categories into which artifacts can be classified. Categories help in organizing and searching for specific artifacts. Each item always has a specific category assigned to it, which allows the museum to search among individual items.

Table: ArtifactCategories

- **artifact_id (int):** Identifies the artifact associated with a particular category, referencing the artifact_id in the Artifacts table. Each artifact can be associated with multiple categories.
- **category_id (int):** Identifies the category to which the artifact belongs, referencing the category_id in the Categories table. Each category can contain multiple artifacts.

The ArtifactCategories table establishes a many-to-many relationship (many categories can be bound to many artifacts), combining artifacts and their categories.

Triggers and Functions Description

Preventing Room from duplicates on active exhibition:

Function **check_room_insertion()** enforces that each room hosts only one active exhibition. By counting active exhibitions associated with a room, it prevents simultaneous exhibitions in the same room. If an active exhibition already exists in a room, the function raises an exception, preventing new exhibition insertion.

Trigger **trigger_check_room_insertion** invokes **check_room_insertion()** before inserting records into **Displayed_Artifacts**. It ensures execution of the function each time a new exhibition is displayed. If a room already hosts an active exhibition, the trigger halts insertion, maintaining scheduling integrity.

Preventing wrong dates settings:

Function **check_shipping_rental_dates()** ensures shipping and delivery dates align with artifact rental periods, maintaining logistics accuracy. It validates that shipping precedes rental and delivery follows return to prevent date discrepancies.

Trigger **trigger_check_dates** calls **check_shipping_rental_dates()** before Control table insertion or update. Enforcing shipping before rental and delivery after return, the trigger safeguards logistics integrity by halting incorrect date settings.

Preventing Artifact from duplicates on active exhibition:

Function **ensure_unique_display()** ensures an artifact is displayed in only one active exhibition. It halts insertion if the artifact is already displayed elsewhere, maintaining exhibition integrity.

Trigger **trigger_ensure_unique_display** invokes **ensure_unique_display()** before inserting into Displayed_Artifacts. It enforces the rule that artifacts cannot be simultaneously displayed at multiple active exhibitions.

Preventing Artifact from inserting in in_transit status:

Function **prevent_artifact_in_transit_from_display()** blocks artifacts in transit from exhibition inclusion. If an artifact is in transit, insertion is halted, ensuring artifacts reach their destination before display.

Trigger **trigger_prevent_artifact_in_transit_from_display** calls **prevent_artifact_in_transit_from_display()** before Displayed_Artifacts insertion, maintaining exhibition management integrity.

Automatic exhibition status settings:

Function **update_exhibition_status()** dynamically updates exhibition status based on current and exhibition dates, streamlining exhibition management.

Trigger **trigger_update_exhibition_status** triggers **update_exhibition_status()** before inserting or updating Exhibitions records, ensuring real-time status updates without manual intervention.

Database Implementation

Artifacts Table

```
INSERT INTO "Artifacts" ("name", "description", "control_id",
"artifact_status")
VALUES
  ('Golden Mask', 'An ancient golden mask.', 6, 'unknown'),
  ('Still Life', 'A classic still life painting.', 7, 'in_transit'),
  ('Ceramic Vase', 'A modern ceramic vase.', 8, 'in_storage');
```

Listing 5 - Insertion to Artifacts Table example

	artifact_id	name	description	control_id	artifact_status
1	16	Golden Mask	An ancient golden mask.	6	on_display
2	17	Still Life	A classic still life painting.	7	in_transit
3	18	Ceramic Vase	A modern ceramic vase.	8	in_storage

Screenshot 1 – Result of Insertion to Artifacts Table

Scenario 1:

Constraint ensures that artifact status only accepts certain values: ('in_transit', 'on_display', 'in_storage'). On the example the artifact status is ‘unknown’, what is impossible to set.

```
INSERT INTO "Artifacts" ("name", "description", "control_id",
"artifact_status")
VALUES
  ('Golden Mask', 'An ancient golden mask.', 6, 'unknown');
```

Listing 6 - Insertion to Artifacts Table example

```
[23514] ERROR: new row for relation "Artifacts" violates check constraint "chk_artifactstatus"
Detail: Failing row contains (23, Golden Mask, An ancient golden mask., 6, unknown).
```

Screenshot 2 – Result of wrong Insertion to Artifacts Table

Artifact Categories Table

```
INSERT INTO "ArtifactCategories" ("artifact_id", "category_id")
VALUES
  (16, 4),
  (17, 5),
  (18, 6);
```

Listing 7 - Insertion to Artifact Categories Table example

	artifact_id	category_id
1	16	4
2	17	5
3	18	6

Screenshot 3 – Result of Insertion to Artifact Categories Table

Control Table

```
INSERT INTO "Control" ("control_date", "checked", "description", "shipping_date",
"delivery_date")
VALUES
  ('2024-04-20T09:00:00Z', false, 'Pending check before shipping.', '2024-04-
10T09:00:00Z', '2024-04-20T12:00:00Z'),
  ('2024-04-16T11:00:00Z', true, 'Checked before shipping.', '2024-04-
10T08:00:00Z', '2024-04-20T12:00:00Z'),
  ('2024-04-19T10:00:00Z', true, 'Checked before shipping.', '2024-04-
12T08:00:00Z', '2024-04-30T12:00:00Z');
```

Listing 8 - Insertion to Control Table example

	contro...	control_date	checked	description	shipping_date	delivery_date
1	8	2024-04-19 10:00:00.000000 +00:00	true	Checked before shipping.	2024-04-12 08:00:…	2024-04-30 12:00:…
2	6	2024-04-20 09:00:00.000000 +00:00	false	Pending check before sh…	2024-04-10 09:00:…	2024-04-20 12:00:…
3	7	2024-04-16 11:00:00.000000 +00:00	true	Checked before shipping.	2024-04-10 08:00:…	2024-04-20 12:00:…

Screenshot 3 – Result of Insertion to Control Table

Scenario 2:

The trigger rows an exception because of wrong shipping and delivering values.

```
INSERT INTO "Artifacts" ("name", "description", "control_id", "artifact_status")
VALUES
  ('Mona Lisa', 'A piece of art by Leonardo da Vinchi.', null, 'on_display');
```

Listing 9 - Insertion to Artifacts Table example

```
INSERT INTO "Rental" ("artifact_id", "owner_id", "rental_status", "rental_date",
"return_date")
VALUES
  (22, 5, 'borrowed', '2024-04-11T09:00:00Z', '2024-04-22T12:00:00Z');
```

Listing 10 - Insertion to Rental Table example

```
UPDATE "Control"
    SET shipping_date = '2024-04-12T08:00:00Z',
        delivery_date = '2024-04-23T08:00:00Z'
WHERE control_id = 7;
```

Listing 11 - Insertion to Control Table example

```
[P0001] ERROR: Shipping and delivery dates must be within rental period.
Where: PL/pgSQL function check_shipping_rental_dates() line 12 at RAISE
```

Screenshot 4 – Result of wrong Insertion to Control Table

Rental Table

```
INSERT INTO "Rental" ("artifact_id", "owner_id", "rental_status",
"rental_date", "return_date")
VALUES
  (16, 5, 'borrowed', '2024-04-11T09:00:00Z', '2024-04-22T12:00:00Z'),
  (17, 6, 'lent', '2024-04-10T08:00:00Z', '2024-05-20T12:00:00Z'),
  (18, 4, 'owned', NULL, NULL);
```

Listing 12 - Insertion to Rental Table example

	rental_id	artifact_id	owner_id	rental_status	rental_date	return_date
1	15	16	5	borrowed	2024-04-11 09:00…	2024-04-22 12:00…
2	16	17	6	lent	2024-04-10 08:00…	2024-05-20 12:00…
3	17	18	4	owned	<null>	<null>

Screenshot 5 – Result of Insertion to Rental Table

Owners Table

```
INSERT INTO "Owners" ("name", "contact_phone", "contact_email", "address")
VALUES
 ('Jane Smith', '+9876543210', 'jane.smith@example.com', '456 Elm Street, Othertown'),
 ('Michael Johnson', '+1122334455', 'michael.johnson@example.com', '789 Oak Avenue, Anycity'),
 ('Emily Davis', '+9988776655', 'emily.davis@example.com', '101 Pine Road, Anothercity');
```

Listing 13 - Insertion to Owners Table example

	exhibition_id	name	description	exhibition_status	start_date	end_date
1	4	Modern Art Showcase	Displaying contemporary artworks.	preparation	2024-05-01 10:00…	2024-06-01 18:00:…
2	5	Photography Exhibition	Showcasing captivating photograp…	preparation	2024-05-10 10:00…	2024-06-10 18:00:…
3	6	Pottery Collection	Featuring a variety of pottery p…	preparation	2024-05-15 10:00…	2024-06-15 18:00:…
4	12	Modern Art Showcase2	Displaying contemporary artworks.	active	2024-04-01 10:00…	2024-05-01 18:00:…

Screenshot 6 – Result of Insertion to Owners Table

Exhibition Table

```
INSERT INTO "Exhibitions" ("name", "description", "exhibition_status",
"start_date", "end_date")
VALUES
  ('Modern Art Showcase', 'Displaying contemporary artworks.', NULL, '2024-05-
01T10:00:00Z', '2024-06-01T18:00:00Z'),
  ('Photography Exhibition', 'Showcasing captivating photographs.', NULL, '2024-05-
10T10:00:00Z', '2024-06-10T18:00:00Z'),
  ('Pottery Collection', 'Featuring a variety of pottery pieces.', NULL, '2024-05-
15T10:00:00Z', '2024-06-15T18:00:00Z'),
  ('Modern Art Showcase2', 'Displaying contemporary artworks.', NULL, '2024-04-
01T10:00:00Z', '2024-05-01T18:00:00Z');
```

Listing 14 - Insertion to Exhibition Table example

	exhibition_id	name	description	exhibition_status	start_date	end_date
1	4	Modern Art Showcase	Displaying contemporary artworks.	preparation	2024-05-01 10:00…	2024-06-01 18:00…
2	5	Photography Exhibition	Showcasing captivating photographs.	preparation	2024-05-10 10:00…	2024-06-10 18:00…
3	6	Pottery Collection	Featuring a variety of pottery pie…	preparation	2024-05-15 10:00…	2024-06-15 18:00…

Screenshot 7 – Result of Insertion to Exhibition Table

```
UPDATE "Exhibitions"
SET "start_date" = '2024-03-20T10:00:00Z',
    "end_date" = '2024-04-20T18:00:00Z'
WHERE "name" = 'Pottery Collection';
```

Listing 15 - Update to Exhibition Table example

	exhibition_id	name	description	exhibition_status	start_date	end_date
1	4	Modern Art Showcase	Displaying contemporary artworks.	preparation	2024-05-01 10:00…	2024-06-01 18:00…
2	5	Photography Exhibition	Showcasing captivating photographs.	preparation	2024-05-10 10:00…	2024-06-10 18:00…
3	12	Modern Art Showcase2	Displaying contemporary artworks.	active	2024-04-01 10:00…	2024-05-01 18:00…
4	6	Pottery Collection	Featuring a variety of pottery pie…	active	2024-03-20 10:00…	2024-04-20 18:00…

Screenshot 8 – Result of Insertion to Exhibition Table

Rooms Table

```
INSERT INTO "Rooms" ("name", "description")
VALUES
  ('Gallery Room 2', 'Another spacious room for art exhibitions.'),
  ('Lobby', 'The entrance area of the gallery.'),
  ('Small Gallery', 'A cozy space for intimate art displays.');
```

Listing 16 - Insertion to Rooms Table example

	room_id	name	description
1	4	Gallery Room 2	Another spacious room for art exhibitions.
2	5	Lobby	The entrance area of the gallery.
3	6	Small Gallery	A cozy space for intimate art displays.

Screenshot 9 – Result of Insertion to Rooms Table

Displayed Artifacts Table

```
INSERT INTO "Displayed_Artifacts" ("exhibition_id", "artifact_id",
"room_id")
VALUES
  (4, 16, 4),
  (6, 18, 5);
```

Listing 17 - Insertion to Displayed Artifacts Table example

	exhibition_id	artifact_id	room_id
1	4	16	4
2	6	18	5

Screenshot 10 – Result of Insertion to Displayed Artifacts Table

Scenario 3:

The trigger rows an exception because the artifact is included in the active exhibition is in the "in_transit" state;

```
INSERT INTO "Displayed_Artifacts" ("exhibition_id", "artifact_id",
"room_id")
VALUES
  (4, 17, 4);
```

Listing 18 - Insertion to Displayed Artifacts Table example

```
[P0001] ERROR: The artifact is in transit and cannot be displayed at this time.
Where: PL/pgSQL function prevent_artifact_in_transit_from_display() line 10 at RAISE
```

Screenshot 11 – Result of wrong Insertion to Displayed Artifacts Table

Scenario 4:

The trigger rows an exception because inserted room already have had an active exhibition in the sane room.

```
INSERT INTO "Displayed_Artifacts" ("exhibition_id", "artifact_id",
"room_id")
VALUES
  (6, 17, 4);
```

Listing 19 - Insertion to Displayed Artifacts Table example

```
[P0001] ERROR: The room already has an active exhibition.
Where: PL/pgSQL function check_room_insertion() line 12 at RAISE
```

Screenshot 12 – Result of wrong Insertion to Displayed Artifacts Table

Scenario 5:

The trigger rows an exception because the inserted artifact is already on display in another exhibition

```
INSERT INTO "Displayed_Artifacts" ("exhibition_id", "artifact_id",
"room_id")
VALUES
  (12, 18, 6);
```

Listing 20 - Insertion to Displayed Artifacts Table example

```
[P0001] ERROR: The artifact is already displayed at another exhibition.
Where: PL/pgSQL function ensure_unique_display() line 14 at RAISE
```

Screenshot 13 – Result of wrong Insertion to Displayed Artifacts Table

Categories Table

```
INSERT INTO "Categories" ("name", "description")
VALUES
('Painting', 'Artworks created by applying pigment to a surface.'),
('Pottery', 'Artworks made of clay and other materials, typically shaped
while wet and then hardened by firing.'),
('Photography', 'Art of creating durable images by recording light.');
```

Listing 21 - Insertion to Categories Table example

	category_id	name	description
1	4	Painting	Artworks created by applying pigment to a surfa…
2	5	Pottery	Artworks made of clay and other materials, typi…
3	6	Photography	Art of creating durable images by recording lig…

Screenshot 14 – Result of wrong Insertion to Categories Table