Fakulta informatiky a informačných technológií STU, Bratislava Ilkovičova 2, 842 16 Bratislava 4

Zadanie č. 5

Implementácia procesov na dátovej úrovni

Múzeum

Databázové systémy

Frederik Duvač
4. semester
21. 4. 2024

Obsah

1. Zadanie

V tomto zadaní budete implementovať procesy a obmedzenia (constrains) z modelu, ktorý ste vypracovali v predchádzajúcom zadaní. Implementácia bude kompletne na databázovej úrovni pomocou SQL. Procesy implementujte ako SQL funkcie/procedúry. Na ostatné obmedzenia môžete používať Triggers alebo Constraints - prípadne iné pokročilejšie PostgreSQL konštrukcie.

Ak urobíte zmeny v dátovom modeli, musia byť popísané rozdiely v dokumentácii.

Napíšte aj príklady na overenie funkcionality Vašej implementácie (volania funkcií/procedúr, DML - INSERTs, UPDATEs a podobne). Príklady napíšte aj pre neplatné situácie (v prípade, že nie sú splnené podmienky príkazy a volania nemajú prebehnúť). Napríklad: ak nemá prebehnúť nejaký INSERT - pochváľte sa s ním. Zadanie, ktoré nebude obsahovať aj takéto príklady bude bodovo penalizované.

Vytvorte testovaciu databázu, ktorá bude súčasťou odovzdania, na ktorej budete prezentovať svoje zadanie. (Databáza naplnená Vašimi údajmi spolu s vytvorenými všetkými funkciami/procedúrami/trigrami) Naplnené dáta slúžia pre účely prezentovania a je teda potrebné mať naplnené dáta v takej kvalite aby bolo možne prezentovať jednotlivé scenare.

V rámci dokumentácie, je potrebné vysvetliť vaše riešenie, príklady volania. Jednotlivé funkcie/procedúry adekvátne okomentujte (ideálne priamo v SQL). Dokumentácia musí byť realizovaná ako PDF dokumentácia s tým, že sa bude nachádzať v AIS odovzdaní. Dokumentáciu môžete napísať aj v anglickom jazyku po dohode s cvičiacim.

Implementujte všetky procesy a obmedzenia z predchádzajúceho zadania. Menovite nás budú minimálne zaujímať nasledovné procesy:

- Naplánovanie expozície.
- Vkladanie nového exempláru.
- Presun exempláru do inej zóny.
- Prevziate exempláru z inej inštitúcie.
- Zapožičanie exempláru z inej inštitúcie.

2. Konceptuálny návrh databázy

2.1. Zmeny

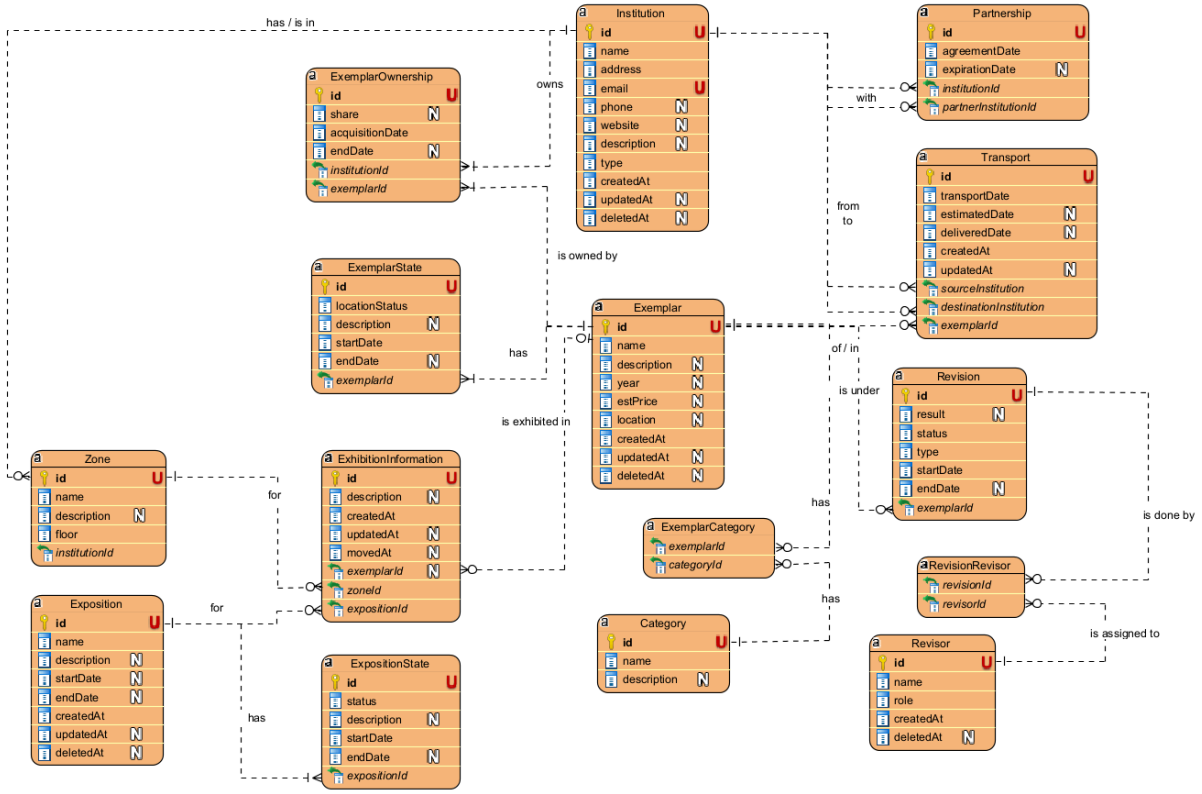
Najpodstatnejšou zmenou je osamostatnenie statusu expozície z entity **ExhibitionInformation**, kde vznikla nová entita **ExemplarState**, to prinesie menšiu redundantnosť dát a zároveň zabezpečí väčšiu flexibilitu pri analyzovaní zozbieraných záznamov o životnom cykle expozície.

Ďalšie zmeny boli vykonané len pri dátových typoch, kde sa atribúty potrebné na reprezentáciu dátumov zmenili z **date** na **timestamp.** V entite Exposition pribudli **startDate**, **endDate** a **updatedAt**. V entite **Revision** zmizol atribút **revisionDate**, v zhľadom na to, že bol redundantný.

2.2. Pridané entity a vzťahy

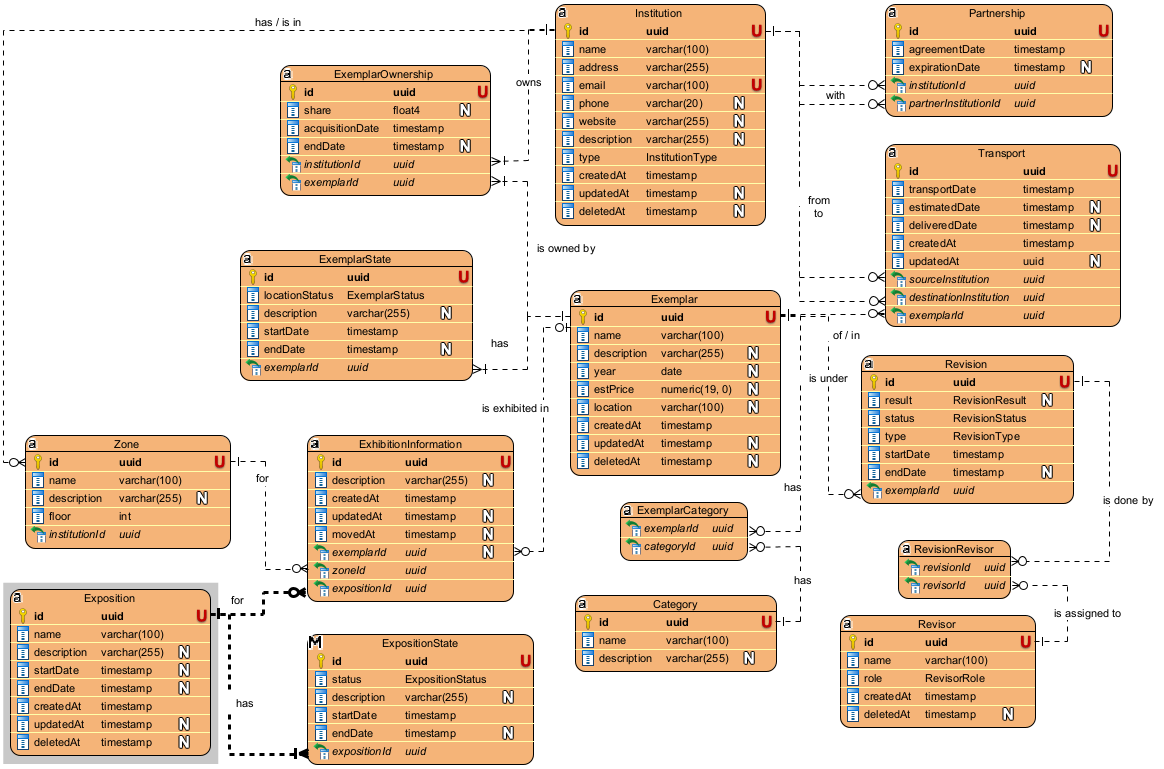
ExpositionState	
Tabuľka **ExpositionState** obsahuje záznamy o statusoch všetkých expozícií.	
Relácia	**Kardinalita**
Exposition	Many to one
Atribúty	**Popis**
id status description startDate endDate expositionId	status - enum hodnota vyjadrujúca, či je daná expozícia plánovaná, prebiehajúca alebo archivovaná ('PLANNED', ONGOING', 'ARCHIVED')

2.3. Konceptuálny diagram



Obr. 1. - Konceptuálny návrh databázy múzea

3. Fyzický návrh databázy



Obr. 2. - ER fyzický diagram

SQL formát data definition language (DDL) sa nachádza v prílohe.

4. Inicializácia databázy a testovacie scenáre

Potrebné procedúry na inicializáciu databázy sú v súbore museum.sql, kde k DDL pribudli inicializačné procedúry **drop_tables()** a **drop_types()**, ktoré zabezpečia premazanie vytvorených dát a vlastných dátových typov.

Ďalej sa vytvoria nasimulované statické dáta entít **Institution, Zone, Category** a dáta entít **Exemplar, ExemplarOwnership, ExemplarState, Revision, Revisor, RevisionRevisor** a **Transport** potrebné pre simuláciu procesu prevzatia exemplára z inej inštitúcie. Tieto dáta reprezentujú proces, ktorý musel prebehnúť predtým ako sa môže exemplár prevziať z inej inštitúcie.

Pri inicializáci sú vytvorené aj hlavné procedúry pre procesy **plan_exposition(), insert_new_exemplar(), move_exemplar_to_zone(), acquire_back_exemplar(), acquire_new_exemplar_from_institution()** a ich pomocné funkcie.

Celý inicializačný kód sa nachádza v prílohe Inicializácia databázy - museum.sql.

Volania jednotlivých vytvorených procedúr pre všetky procesy a ich rôzne testovacie scenáre sa nachádzajú v súbore scenarios.sql. Každý proces je oddelený sekciou s komentárom a obsahuje okomentované, tak ako aj úspešné scenáre tak aj neúspešné.

Kód k testovacím scenárom sa nachádza v prílohe Simulácia procesov - scenarios.sql.

5. Procesy

5.1. Naplánovanie expozície

Pri naplánovaní expozície sa vytvorý nový záznam v tabuľke **Exposition.** Zvalidujú sa dátumy ak sú. Vytvorí sa nový **ExpositionState.** V tabuľke **ExhibitedInformation** sa vytvoria nové záznamy s pridelenými ID zóny a expozície. Databáza umožňuje vytvoriť expozíciu bez vopred známych dátumov. Pri plánovaní sa validujú všetky scenáre pomocou funkcie **check_exposition_overlap()**, ktoré sú v kóde okomentované. Nastaví sa status expozície na PLANNED.

5.2. Vkladanie nového exemplára

Pri vytvorení nového exemplára sa inicializujú všetky povinné prípadne aj nepovinné atribúty entity **Exemplar**. Vytvorí sa záznam o vlastníctve v tabuľke **ExemplarOwnership** s priradeným ID našej inštitúcie (predpokladajme že záznam o našom múzeu je staticky vytvorený pri inicializácií databázy) a exemplára v prípade, že exemplár je vlastnený viacerými inštitúciami takýchto záznamov sa vytvorí príslušné množstvo.

V prípade že exemplár má kategóriu, vytvorí sa záznam o kategórií, v prípade ak neexistuje inak sa použije existujúca kategória. Vytvorí sa relácia medzi kategóriou a exemplárom v tabuľke **ExemplarCategory**.

V tabuľke **ExemplarState** sa vytvorí záznam s **locationStatus** IN_STORAGE a priradí sa ID exemplára.

5.3. Presun exemplára do inej zóny

Presun exemplára do inej zóny sprevádzajú nasledujúce kroky:

Na začiatok sa uistite či existujú expozície ak nie vytvorte si ich postupnostou krokov pre naplánovanie expozície.

Najskôr prebehne kontrola, či existuje zóna, exemplár a expozícia. V prípade, ak exemplár je v sklade alebo v trezore, uzavrie sa predchádzajúci status exemplára a vytvorí sa nový záznam v tabuľke **ExemplarState** s **locationStatusom** WITHIN_MUSEUM. Keď exemplár dorazí na svoje miesto v zóne predchádzajúci status sa ukončí a vytvorí sa nový status pre daný exemplár EXHIBITED. V prípade že záznam **ExhibitedInfomation** neobsahuje ešte exemplár tak sa priradí ID exemplára inak sa vytvorí nový záznam.

V prípade ak exemplár je už vystavený exemplár dostane status WITHIN_MUSEUM, a v príslušnom zozname **ExhibitedInformation** nastaví dátum presunu a vytvorí sa nový záznam **ExhibitedInformation** s novou zónou a status sa nastavi na EXHIBITED.

5.4. Prevzatie exempláru z inej inštitúcie

Pri prevzatí exempláru z inej inštitúcie máme vytvorené nasimulované dáta s celým historickým procesom presunu exempláru do inej institucie a naspat do našej inštitúcie. Posledný status exempláru je IN_TRANSIT. Prevzatie prebieha tak že sa skontroluje, či exemplár vôbec existuje. Uzavrie sa predchádzajúci stav a vytvorí sa nový záznam s **locationStatus** IN_STORAGE a priradí sa ID exempláru.

5.5. Zapožičanie exempláru z inej inštitúcie

Tento proces vyžaduje informácie o exemplári, ktoré inicializujú atribúty pri vytvorení záznamu entity **Exemplar**. Na základe unikátneho emailu inštitúcie sa zvaliduje vstup, ak institucia neexistuje tak sa vytvorí. Ďalšie kroky, ktoré sa vykonajú sú rovnaké ako pri prevzatí a vytváraní exempláru. Zmena nastane pri vytváraní záznamu v tabuľke **ExemplarState**, kde sa vytvorí záznam s **locationStatus** BORROWED a priradí sa ID exempláru.

6. Záver

Návrh a dizajn databázového modelu s jemnou zmenou, pomohol k flexibilite riešenia tohto zadania. Navrhnutý databázový model je nadmieru postačujúci pre splnenie požiadaviek stanovených v zadání.

Vďaka dobre navrhnutému ER modelu je možné jednoducho validovať pomocou funkcií a procedúr jednotlivé entity a ich informácie. Niektoré funkcie a procedúry ponúkajú variabilný vstup a tak sú využiteľné vo viacerých prípadoch.

Prílohy

1. Inicializácia databázy - museum.sql

```
/*
* DBS Assignment 5: Implementation of processes on database layer
* Author: Frederik Duvač
*/

/*
* Initial setup
*
* Remove all tables
*/
create or replace function drop_tables() returns void as
$$
declare
   table_name     varchar;
   tables_to_drop varchar[] := array [
       'Partnership',
       'Zone',
       'ExemplarOwnership',
       'Exemplar',
       'ExemplarState',
       'ExemplarCategory',
       'Transport',
       'Institution',
       'Revision',
       'RevisionRevisor',
       'Revisor',
       'Category',
       'ExhibitionInformation',
       'Exposition',
       'ExpositionState'
       ];
begin
   foreach table_name in array tables_to_drop
       loop
           execute 'drop table if exists ' || table_name || '
cascade';
       end loop;
end;
$$ language plpgsql;

do
$$
```

```
    begin
        perform drop_tables();
    end;
$$;

/*
* Remove all custom types
*/
create or replace function drop_types() returns void as
$$
declare
    type_name     varchar;
    types_to_drop varchar[] := array [
        'InstitutionType',
        'ExemplarStatus',
        'RevisionResult',
        'RevisionStatus',
        'RevisionType',
        'RevisorRole',
        'ExpositionStatus'
        ];
begin
    foreach type_name in array types_to_drop
        loop
            execute 'drop type if exists ' || type_name || '
cascade';
        end loop;
end;
$$ language plpgsql;

do
$$
    begin
        perform drop_types();
    end;
$$;

/*
* Database schema creation
*/
create type InstitutionType as enum (
    'MUSEUM',
    'GALLERY',
    'PRIVATE_COLLECTOR',
    'LIBRARY',
    'UNIVERSITY',
    'GOVERNMENT_AGENCY',
    'CULTURAL_CENTER',
```

```
    'HISTORICAL_SOCIETY',
    'FOUNDATION',
    'AUCTION_HOUSE',
    'ART_DEALER',
    'RESEARCH_INSTITUTE',
    'NON_PROFIT_ORGANIZATION',
    'CORPORATE_COLLECTION',
    'RELIGIOUS_INSTITUTION',
    'OTHER'
    );

create type ExemplarStatus as enum (
    'WITHIN_MUSEUM',
    'SENT_TO_OTHER_INSTITUTION',
    'IN_TRANSIT',
    'BORROWED',
    'EXHIBITED',
    'IN_STORAGE',
    'IN_VAULT',
    'RETURNING'
    );

create type RevisionResult as enum (
    'OK',
    'GOOD_CONDITION',
    'NEEDS_REPAIR',
    'MISSING_PARTS',
    'DAMAGED'
    );

create type RevisionStatus as enum (
    'PENDING',
    'IN_PROGRESS',
    'COMPLETED',
    'CANCELED'
    );

create type RevisionType as enum (
    'LOAN_CONDITION_CHECK',
    'CONDITION_ASSESSMENT',
    'CONSERVATION_TREATMENT',
    'AUTHENTICITY_VERIFICATION',
    'DOCUMENTATION_REVIEW',
    'PRESERVATION_SURVEY',
    'DAMAGE_REPAIR',
    'CLEANING_PROCESS',
    'TECHNICAL_ANALYSIS',
    'PRESERVATION_CHECK',
```

```
    'INVENTORY_AUDIT',
    'RESTORATION_EVALUATION',
    'SECURITY_AUDIT',
    'COMPLIANCE_INSPECTION',
    'OTHER'
    );

create type RevisorRole as enum (
    'CURATOR',
    'CONSERVATOR',
    'HISTORIAN',
    'ART_CRITIC',
    'ARCHAEOLOGIST',
    'SCIENTIST',
    'EXPERT',
    'ART_CONSERVATOR',
    'ART_HISTORIAN',
    'MUSEUM_TECHNICIAN',
    'ARCHIVIST',
    'SCIENTIFIC_RESEARCHER',
    'COLLECTIONS_MANAGER',
    'SECURITY',
    'OTHER'
    );

create type ExpositionStatus as enum (
    'PLANNED',
    'ONGOING',
    'ARCHIVED'
    );
create table Institution
(
    id          uuid            not null,
    name        varchar(100)    not null,
    address     varchar(255)    not null,
    email       varchar(100)    not null unique,
    phone       varchar(20),
    website     varchar(255),
    description varchar(255),
    type        InstitutionType not null,
    createdAt   timestamp       not null,
    updatedAt   timestamp,
    deletedAt   timestamp,
    primary key (id)
);
create table Partnership
(
    id                  uuid      not null,
```

```
    agreementDate         timestamp not null,
    expirationDate        timestamp,
    institutionId         uuid      not null,
    partnerInstitutionId uuid       not null,
    primary key (id)
);
create table Zone
(
    id             uuid         not null,
    name           varchar(100) not null,
    description    varchar(255),
    floor          int          not null,
    institutionId uuid          not null,
    primary key (id)
);
create table ExemplarOwnership
(
    id              uuid      not null,
    share           float4,
    acquisitionDate timestamp not null,
    endDate         timestamp,
    institutionId   uuid      not null,
    exemplarId      uuid      not null,
    primary key (id)
);
create table Exemplar
(
    id          uuid         not null,
    name        varchar(100) not null,
    description varchar(255),
    year        date,
    estPrice    numeric(19, 0),
    location    varchar(100),
    createdAt   timestamp    not null,
    updatedAt   timestamp,
    deletedAt   timestamp,
    primary key (id)
);
create table ExemplarState
(
    id             uuid           not null,
    locationStatus ExemplarStatus not null,
    description    varchar(255),
    startDate      timestamp      not null,
    endDate        timestamp,
    exemplarId     uuid           not null,
    primary key (id)
);
```

```sql
create table ExemplarCategory
(
   exemplarId uuid not null,
   categoryId uuid not null
);
create table Transport
(
   id                    uuid      not null,
   transportDate         timestamp not null,
   estimatedDate         timestamp,
   deliveredDate         timestamp,
   createdAt             timestamp not null,
   updatedAt             timestamp,
   sourceInstitution     uuid      not null,
   destinationInstitution uuid     not null,
   exemplarId            uuid      not null,
   primary key (id)
);
create table Revision
(
   id           uuid                          not null,
   revisionDate timestamp                     not null,
   result       RevisionResult,
   status       RevisionStatus                not null,
   type         RevisionType default 'OTHER' not null,
   startDate    timestamp                     not null,
   endDate      timestamp,
   exemplarId   uuid                          not null,
   primary key (id)
);
create table RevisionRevisor
(
   revisionId uuid not null,
   revisorId  uuid not null
);
create table Revisor
(
   id        uuid                         not null,
   name      varchar(100)                 not null,
   role      RevisorRole default 'OTHER' not null,
   createdAt timestamp                    not null,
   deletedAt timestamp,
   primary key (id)
);
create table Category
(
   id          uuid         not null,
   name        varchar(100) not null,
```

```
   description varchar(255),
   primary key (id)
);
create table ExhibitionInformation
(
   id           uuid      not null,
   description  varchar(255),
   createdAt    timestamp not null,
   updatedAt    timestamp,
   movedAt      timestamp,
   exemplarId   uuid,
   zoneId       uuid      not null,
   expositionId uuid      not null,
   primary key (id)
);
create table Exposition
(
   id          uuid         not null,
   name        varchar(100) not null,
   description varchar(255),
   startDate   timestamp,
   endDate     timestamp,
   createdAt   timestamp    not null,
   updatedAt   timestamp,
   deletedAt   timestamp,
   primary key (id)
);
create table ExpositionState
(
   id           uuid                                  not null,
   status       ExpositionStatus default 'PLANNED' not null,
   description  varchar(255),
   startDate    timestamp                             not null,
   endDate      timestamp,
   expositionId uuid                                  not null,
   primary key (id)
);
alter table Partnership
   add constraint FKPartnershi227370 foreign key (institutionId)
references Institution (id);
alter table Partnership
   add constraint "with" foreign key (partnerInstitutionId)
references Institution (id);
alter table Zone
   add constraint "has / is in" foreign key (institutionId)
references Institution (id);
alter table ExemplarState
```

```
   add constraint "has " foreign key (exemplarId) references
Exemplar (id);
alter table ExemplarCategory
   add constraint "has  " foreign key (exemplarId) references
Exemplar (id);
alter table ExemplarCategory
   add constraint "has   " foreign key (categoryId) references
Category (id);
alter table Transport
   add constraint "from" foreign key (sourceInstitution)
references Institution (id);
alter table Transport
   add constraint "to" foreign key (destinationInstitution)
references Institution (id);
alter table Transport
   add constraint "of / in" foreign key (exemplarId) references
Exemplar (id);
alter table Revision
   add constraint "is under " foreign key (exemplarId) references
Exemplar (id);
alter table RevisionRevisor
   add constraint "is done by" foreign key (revisionId) references
Revision (id);
alter table RevisionRevisor
   add constraint "is assigned to" foreign key (revisorId)
references Revisor (id);
alter table ExhibitionInformation
   add constraint "is exhibited in" foreign key (exemplarId)
references Exemplar (id);
alter table ExhibitionInformation
   add constraint "for" foreign key (zoneId) references Zone (id);
alter table ExhibitionInformation
   add constraint "for " foreign key (expositionId) references
Exposition (id);
alter table ExemplarOwnership
   add constraint owns foreign key (institutionId) references
Institution (id);
alter table ExemplarOwnership
   add constraint "is owned by" foreign key (exemplarId)
references Exemplar (id);
alter table ExpositionState
   add constraint "has    " foreign key (expositionId) references
Exposition (id);

/*
* Seeding some initial data
*/
```

```
insert into Institution (id, name, address, email, phone, website,
description, type, createdAt)
values ('f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b', 'Museum of
Technology', '1234 Museum Street, City',
       'technology@museum.com', '123456789', 'www.museum.com',
'This is the best Museum of Technology', 'MUSEUM',
       '2022-01-01 21:16:17.293281'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3c', 'Art Gallery',
'5678 Gallery Street, City', 'art@galery.com',
       '987654321', 'www.gallery.com', 'This is the best Art
Gallery', 'GALLERY', '2022-01-01 21:16:17.293281'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3d', 'Private
Collector', '91011 Collector Street, City',
       'collector@ferrari.com', '123123123', 'www.collector.com',
'This is the best Private Collector',
       'PRIVATE_COLLECTOR', '2022-01-01 21:16:17.293281');

insert into Zone (id, name, description, floor, institutionId)
values ('f0b3b3b3-3b3b-3b3b-3b3b-000000000001', 'Main Hall', 'Main
hall of the museum', 1,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000002', 'Second Hall',
'Second hall of the museum', 1,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000003', 'Third Hall',
'Third hall of the museum', 2,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000004', 'Fourth Hall',
'Fourth hall of the museum', 2,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000005', 'Fifth Hall',
'Fifth hall of the museum', 3,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000006', 'Sixth Hall',
'Sixth hall of the museum', 3,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000007', 'Seventh Hall',
'Seventh hall of the museum', 4,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000008', 'Eighth Hall',
'Eighth hall of the museum', 4,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000009', 'Ninth Hall',
'Ninth hall of the museum', 5,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b'),
      ('f0b3b3b3-3b3b-3b3b-3b3b-000000000010', 'Tenth Hall',
'Tenth hall of the museum', 5,
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b');
```

```
insert into category (id, name, description)
values (uuid_generate_v4(), 'artificial intelligence', 'Category
related to AI technologies'),
       (uuid_generate_v4(), 'machine learning', 'Category related to
Machine Learning technologies'),
       (uuid_generate_v4(), 'data science', 'Category related to
Data Science technologies'),
       (uuid_generate_v4(), 'cybersecurity', 'Category related to
Cybersecurity technologies'),
       (uuid_generate_v4(), 'cloud computing', 'Category related to
Cloud Computing technologies'),
       (uuid_generate_v4(), 'blockchain', 'Category related to
Blockchain technologies'),
       (uuid_generate_v4(), 'internet of things', 'Category related
to IoT technologies'),
       (uuid_generate_v4(), 'virtual reality', 'Category related to
VR technologies'),
       (uuid_generate_v4(), 'augmented reality', 'Category related
to AR technologies'),
       (uuid_generate_v4(), 'quantum computing', 'Category related
to Quantum Computing technologies');

-- Seeder for process of acquire back borrowed exemplar
insert into exemplar (id, name, description, year, estPrice,
location, createdAt)
values ('f0e3e3e3-3e3e-3e3e-3e3e-000000000004', 'exemplar 4',
'exemplar 4 description', '1950-01-01', 10000,
        'location 4', '2023-01-01 21:16:17.293281');

insert into exemplarownership (id, share, acquisitionDate,
institutionId, exemplarId)
values (uuid_generate_v4(), 1, '2022-01-01 21:16:17.293281',
'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b',
        'f0e3e3e3-3e3e-3e3e-3e3e-000000000004');

insert into exemplarstate (id, locationStatus, description,
startDate, endDate, exemplarId)
values (uuid_generate_v4(), 'IN_STORAGE', 'exemplar 4 is in
storage', '2023-01-01 21:16:17.293281',
        '2023-01-02 21:16:17.293281',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004'),
       (uuid_generate_v4(), 'SENT_TO_OTHER_INSTITUTION', 'exemplar 4
is sent to other institution',
        '2023-01-02 21:16:17.293281', '2023-01-03 21:16:17.293281',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004'),
       (uuid_generate_v4(), 'IN_TRANSIT', 'exemplar 4 is in
transit', '2023-01-03 21:16:17.293281',
```

```
       '2023-01-04 21:16:17.293281',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004'),
      (uuid_generate_v4(), 'BORROWED', 'exemplar 4 is borrowed',
'2023-01-04 21:16:17.293281',
       '2023-01-05 21:16:17.293281',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004'),
      (uuid_generate_v4(), 'RETURNING', 'exemplar 4 is returning',
'2023-01-05 21:16:17.293281',
       '2023-01-06 21:16:17.293281',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004'),
      (uuid_generate_v4(), 'IN_TRANSIT', 'exemplar 4 is in
transit', '2023-01-06 21:16:17.293281',
       '2023-01-07 21:16:17.293281',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004');

insert into Revision (id, revisionDate, result, status, type,
startDate, endDate, exemplarId)
values ('f0a1a1a1-1a1a-1a1a-1a1a-000000000001', '2023-01-01
00:00:00.000000', 'OK', 'COMPLETED',
       'LOAN_CONDITION_CHECK',
       '2023-01-01 00:00:00.000000', '2023-01-01 12:00:00.000000',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004');

insert into Revisor (id, name, role, createdAt)
values ('f0a2a2a2-2a2a-2a2a-2a2a-2a2a2a2a2a2a', 'Revisor 1',
'SECURITY', '2022-01-01 21:16:17.293281'),
      ('f0a2a2a2-2a2a-2a2a-2a2a-2a2a2a2a2a2b', 'Revisor 2',
'CURATOR', '2022-01-01 21:16:17.293281'),
      ('f0a2a2a2-2a2a-2a2a-2a2a-2a2a2a2a2a2c', 'Revisor 3',
'HISTORIAN', '2022-01-01 21:16:17.293281');

insert into RevisionRevisor (revisionId, revisorId)
values ('f0a1a1a1-1a1a-1a1a-1a1a-000000000001',
'f0a2a2a2-2a2a-2a2a-2a2a-2a2a2a2a2a2a'),
      ('f0a1a1a1-1a1a-1a1a-1a1a-000000000001',
'f0a2a2a2-2a2a-2a2a-2a2a-2a2a2a2a2a2b'),
      ('f0a1a1a1-1a1a-1a1a-1a1a-000000000001',
'f0a2a2a2-2a2a-2a2a-2a2a-2a2a2a2a2a2c');

insert into transport (id, transportDate, estimatedDate,
deliveredDate, createdAt, updatedAt, sourceInstitution,
                       destinationInstitution, exemplarId)
values ('f0c4c4c4-4c4c-4c4c-4c4c-000000000001', '2023-01-03
21:16:17.293281', '2023-01-04 21:16:17.293281',
       '2023-01-04 21:16:17.293281', now(), now(),
'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b',
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3d',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004'),
```

```
      ('f0c4c4c4-4c4c-4c4c-4c4c-000000000002', '2023-01-05
21:16:17.293281', '2023-01-06 21:16:17.293281',
       '2023-01-06 21:16:17.293281', now(), now(),
'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3d',
       'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b',
'f0e3e3e3-3e3e-3e3e-3e3e-000000000004');

----------------------------
-- 1. Plan new exposition --
----------------------------
create or replace function zone_exists(
   zone_id uuid
)
   returns boolean as
$$
begin
   return exists(select 1 from zone where id = zone_id);
end;
$$ language plpgsql;

create or replace function check_exposition_dates(
   exposition_id uuid
)
   returns boolean as
$$
declare
   exposition_record record;
begin
   select id, startdate, enddate
   into exposition_record
   from exposition
   where id = exposition_id;

   if exposition_record.startdate >= now() and
exposition_record.enddate > exposition_record.startdate then
       return true;
   else
       return false;
   end if;

end ;
$$ language plpgsql;

create or replace function
get_latest_exposition_state(exposition_id uuid)
   returns table
           (
               id           uuid,
```

```
                status       ExpositionStatus,
                description  varchar(255),
                startDate    timestamp,
                endDate      timestamp,
                expositionId uuid
            )
as
$$
begin
    return query
        select ExpositionState.id,
               ExpositionState.status,
               ExpositionState.description,
               ExpositionState.startDate,
               ExpositionState.endDate,
               ExpositionState.expositionId
        from ExpositionState
        where ExpositionState.expositionId = exposition_id
        order by startDate desc
        limit 1;
end;
$$ language plpgsql;

create or replace function check_exposition_overlap(
    new_zone_id uuid,
    new_start_date timestamp,
    new_end_date timestamp
)
    returns void as
$$
declare
    overlapping_exposition_count integer;
begin
    select count(*)
    into overlapping_exposition_count
    from exhibitioninformation ei
             join exposition e on ei.expositionId = e.id
             join expositionstate es on e.id = es.expositionId
             join lateral (select status from
get_latest_exposition_state(e.id)) as latest_exposition_state on
true
    where ei.zoneId = new_zone_id
      and (
        -- Check if the new exposition overlaps with any other
ONGOING / PLANED exposition in the same zone with certain time
frame
        (
            (
```

```
                latest_exposition_state.status != 'ARCHIVED'
                    and
                e.startDate is not null and e.endDate is not null
                )
                and
            (
                (e.startDate <= new_start_date and e.endDate >=
new_start_date)
                    or
                (e.startDate >= new_start_date and e.startDate <=
new_end_date)
                )
            )
            or
            -- If the ONGOING exposition has no end date, it is not
possible to plan a new exposition in the same zone
       (latest_exposition_state.status = 'ONGOING' and e.endDate
is null)
            or
            -- If the PLANNED exposition has no end date and the
new exposition starts before or within the PLANNED exposition,
            -- it is not possible to plan a new exposition in the
same zone if the new exposition ends after the PLANNED exposition
starts
       (latest_exposition_state.status = 'PLANNED' and e.startDate
is not null and e.endDate is null and
        e.startDate <= new_end_date)
            or
       (
            -- If there is a planned exposition in the future with
start and end date, and the new exposition without end date starts
before or within the planned one
            latest_exposition_state.status = 'PLANNED'
                and
            e.startDate is not null and e.endDate is not null
                and
            e.endDate >= new_start_date and new_end_date is null
            )
       );

   if overlapping_exposition_count > 0 then
       raise exception 'Another exposition is already in the same
zone during the same time period';
   end if;
end;
$$ language plpgsql;

create or replace procedure plan_exposition(
```

```
    exposition_name varchar(100),
    zone_ids uuid[],
    exposition_id uuid = uuid_generate_v4(),
    exposition_state ExpositionStatus = 'PLANNED',
    exposition_description varchar(255) = null,
    expositionstate_description varchar(255) = null,
    exposition_start_date timestamp = null,
    exposition_end_date timestamp = null
)
    language plpgsql
as
$$
declare
    new_exposition_id uuid;
    zone_id           uuid;
begin
    -- Start a transaction
    begin
        -- Insert new exposition
        new_exposition_id := exposition_id;
        insert into exposition (id, name, description, startdate,
endDate, createdat)
        values (new_exposition_id, exposition_name,
exposition_description, exposition_start_date,
exposition_end_date,
                now());

        -- Check if the new exposition dates are valid
        if exposition_start_date is not null and
exposition_end_date is not null then
            if not check_exposition_dates(new_exposition_id) then
                raise exception 'Exposition start date must be in
the future and end date must be after the start date';
            end if;
        end if;

        -- Create new exposition state
        insert into expositionstate (id, status, description,
startdate, expositionid)
        values (uuid_generate_v4(), exposition_state,
expositionstate_description, now(), new_exposition_id);

        foreach zone_id in array zone_ids
            loop
                -- Check if zone exists
                if not zone_exists(zone_id) then
                    raise exception 'Zone with id % does not
exist', zone_id;
```

```
                end if;

                -- Check if the new exposition overlaps with any
other exposition in the same zone
                perform check_exposition_overlap(zone_id,
exposition_start_date, exposition_end_date);

                -- Create new exhibited information
                insert into exhibitioninformation (id, createdAt,
zoneid, expositionid)
                values (uuid_generate_v4(), now(), zone_id,
new_exposition_id);
            end loop;

    exception
        when others then
            -- If any operation fails, roll back the transaction
            rollback;
            raise;
    end;
end;
$$;

---------------------------------
-- 2. Insert of a new exemplar --
---------------------------------
create or replace function exemplar_exists(
    exemplar_id uuid
)
    returns boolean as
$$
begin
    return exists(select 1 from exemplar where id = exemplar_id);
end;
$$ language plpgsql;

create or replace function institution_exists(institution_id uuid)
    returns boolean as
$$
begin
    return exists(select 1 from institution where id =
institution_id);
end;
$$ language plpgsql;

create or replace function
check_ownership_array_lengths(institution_ids uuid[],
acquisition_dates timestamp[])
```

```
    returns boolean as
$$
begin
    return array_length(institution_ids, 1) =
array_length(acquisition_dates, 1);
end;
$$ language plpgsql;

create or replace procedure insert_new_exemplar(
    exemplar_name varchar(100),
    owner_institution_ids uuid[],
    owner_acquisition_date timestamp[],
    exemplar_id uuid = uuid_generate_v4(),
    exemplar_description varchar(255) = null,
    exemplar_year timestamp = null,
    exemplar_estprice numeric = null,
    exemplar_location varchar(100) = null,
    exemplarstate_location_status exemplarstatus = 'IN_STORAGE',
    exemplarstate_desc varchar(255) = null,
    category_names varchar(100)[] = null,
    category_descriptions varchar(255)[] = null,
    owner_share float4[] = null
)
    language plpgsql
as
$$
declare
    new_exemplar_id      uuid;
    new_category_id      uuid;
    owner_inst_id        uuid;
    category_name        varchar(100);
    category_description varchar(255);
    counter              integer := 1;
begin
    -- Start a transaction
    begin
        if not check_ownership_array_lengths(owner_institution_ids,
owner_acquisition_date) then
            raise exception 'Length of owner_institution_ids and
owner_acquisition_date arrays must be equal';
        end if;

        -- Insert new exemplar
        new_exemplar_id := exemplar_id;
        if exemplar_exists(new_exemplar_id) then
            raise exception 'Exemplar with id % already exists',
new_exemplar_id;
        end if;
```

```
        insert into exemplar (id, name, description, year,
estprice, location, createdat)
        values (new_exemplar_id, exemplar_name,
exemplar_description, exemplar_year, exemplar_estprice,
                exemplar_location,
                now());

        foreach owner_inst_id in array owner_institution_ids
            loop
                if not institution_exists(owner_inst_id) then
                    raise exception 'Institution with id % does not
exist', owner_inst_id;
                end if;
                -- Insert new ownership record
                insert into exemplarownership (id, share,
acquisitiondate, enddate, institutionid, exemplarid)
                values (uuid_generate_v4(), owner_share[counter],
owner_acquisition_date[counter], null,
                        owner_inst_id, new_exemplar_id);
                counter := counter + 1;
            end loop;

        counter := 1;

        -- Insert new categories if exists
        if category_names is not null then
            if array_length(category_names, 1) !=
array_length(category_descriptions, 1) then
                raise exception 'Length of category_names and
category_descriptions arrays must be equal';
            end if;

            foreach category_name in array category_names
                loop
                    category_description :=
category_descriptions[counter];
                    -- Check if category exists, if not create new
one
                    select id into new_category_id from category
where name = category_name;
                    if new_category_id is null then
                        new_category_id := uuid_generate_v4();
                        insert into category (id, name,
description)
                        values (new_category_id, category_name,
category_description);
                    end if;
```

```
                    -- Assign category to the new exemplar
                    insert into exemplarcategory (exemplarid,
categoryid)
                    values (new_exemplar_id, new_category_id);
                    counter := counter + 1;
                end loop;
        end if;

        -- Set the initial state for the new exemplar
        insert into exemplarstate (id, locationstatus, description,
startdate, exemplarid)
        values (uuid_generate_v4(), exemplarstate_location_status,
exemplarstate_desc, now(), new_exemplar_id);

    exception
        when others then
            -- If any operation fails, roll back the transaction
            rollback;
            raise;
    end;
end;
$$;

-------------------------------------------
-- 3. Move of the exemplar to other zone --
-------------------------------------------
create or replace function exposition_exists(
    exposition_id uuid
)
    returns boolean as
$$
begin
    return exists(select 1 from exposition where id =
exposition_id);
end;
$$ language plpgsql;

create or replace function close_latest_exemplar_state(
    exemplar_id uuid
)
    returns void as
$$
declare
    exemplar_state record;
begin
    -- Get the current state of the exemplar
    select into exemplar_state *
```

```
    from exemplarstate
    where exemplarid = exemplar_id
      and endDate is null
    order by startdate desc
    limit 1;

    -- Close the latest state of the exemplar
    update exemplarstate
    set enddate = now()
    where id = exemplar_state.id;
end;
$$ language plpgsql;

create or replace procedure move_exemplar_to_zone(
    exemplar_id uuid,
    zone_id uuid,
    exposition_id uuid
)
    language plpgsql
as
$$
declare
    exemplar_state  record;
    exhibition_info record;
begin
    -- Start a transaction
    begin
        -- Check if the exemplar exists
        if not exemplar_exists(exemplar_id) then
            raise exception 'Exemplar with id % does not exist',
exemplar_id;
        end if;

        -- Check if the zone exists
        if not zone_exists(zone_id) then
            raise exception 'Zone with id % does not exist',
zone_id;
        end if;

        -- Check if the exposition exists
        if not exposition_exists(exposition_id) then
            raise exception 'Exposition with id % does not exist',
exposition_id;
        end if;

        select into exemplar_state *
        from exemplarstate
        where exemplarid = exemplar_id
```

```
          and endDate is null
        order by startdate desc
        limit 1;

        -- If the exemplar is in storage, move it to the exhibition
        if (exemplar_state.locationstatus = 'IN_STORAGE' or
exemplar_state.locationStatus = 'IN_VAULT') then
            -- Close the latest state of the exemplar
            perform close_latest_exemplar_state(exemplar_id);

            -- Create a new state for the exemplar with
locationStatus set to WITHIN_MUSEUM
            insert into exemplarstate (id, locationstatus,
startdate, exemplarid)
            values (uuid_generate_v4(), 'WITHIN_MUSEUM', now(),
exemplar_id);

            perform close_latest_exemplar_state(exemplar_id);

            -- Find the current exhibition information where to put
the exemplar
            select into exhibition_info *
            from exhibitioninformation
            where zoneid = zone_id
              and expositionid = exposition_id
            order by createdat
            limit 1;

            if exhibition_info is null then
                raise exception 'Exposition with id % does not
exist in the zone with id %', exposition_id, zone_id;
            end if;

            if exhibition_info.exemplarId is not null then
                -- Create a new exhibition information record with
the new zone
                insert into exhibitioninformation (id, createdat,
zoneid, exemplarid, expositionid)
                values (uuid_generate_v4(), now(), zone_id,
exemplar_id, exposition_id);
            else
                -- Update the exhibition information with the new
exemplar
                update exhibitioninformation
                set exemplarid = exemplar_id
                where id = exhibition_info.id;
            end if;
        elseif exemplar_state.locationstatus = 'EXHIBITED' then
```

```
            -- Close the latest state of the exemplar
            perform close_latest_exemplar_state(exemplar_id);

            -- Create a new state for the exemplar with
locationStatus set to WITHIN_MUSEUM
            insert into exemplarstate (id, locationstatus,
startdate, exemplarid)
            values (uuid_generate_v4(), 'WITHIN_MUSEUM', now(),
exemplar_id);

            perform close_latest_exemplar_state(exemplar_id);

            -- Get the current exhibition information of the
exemplar
            select into exhibition_info *
            from exhibitioninformation
            where exemplarid = exemplar_id
            order by createdat desc
            limit 1;

            -- Update the exhibition information with the date of
the move
            update exhibitioninformation
            set movedat   = now(),
                updatedAt = now()
            where id = exhibition_info.id;

            -- Find the current exhibition information where to put
the exemplar
            select into exhibition_info *
            from exhibitioninformation
            where zoneid = zone_id
              and expositionid = exposition_id
            order by createdat
            limit 1;

            if exhibition_info is null then
                raise exception 'Exposition with id % does not
exist in the zone with id %', exposition_id, zone_id;
            end if;

            if exhibition_info.exemplarId is not null then
                -- Create a new exhibition information record with
the new zone
                insert into exhibitioninformation (id, createdat,
zoneid, exemplarid, expositionid)
                values (uuid_generate_v4(), now(), zone_id,
exemplar_id, exposition_id);
```

```
            else
                -- Update the exhibition information with the new
exemplar
                update exhibitioninformation
                set exemplarid = exemplar_id
                where id = exhibition_info.id;
            end if;
        else
            -- If the exemplar is not in museum, it cannot be moved
            raise exception 'Exemplar with id % is not in storage,
vault or exhibited', exemplar_id;
        end if;

        -- Create a new state for the exemplar with locationStatus
set to EXHIBITED
        insert into exemplarstate (id, locationstatus, startdate,
exemplarid)
        values (uuid_generate_v4(), 'EXHIBITED', now(),
exemplar_id);

    exception
        when others then
            -- If any operation fails, roll back the transaction
            rollback;
            raise;
    end;
end;
$$;

-----------------------------------------------------------------
-- 4. Acquire back borrowed exemplar from another institution --
-----------------------------------------------------------------
create or replace procedure acquire_back_exemplar(exemplar_id uuid)
    language plpgsql
as
$$
declare
    exemplar_state record;
begin
    -- Start a transaction
    begin
        -- check if the exemplar exists
        if not exemplar_exists(exemplar_id) then
            raise exception 'Exemplar with id % does not exist',
exemplar_id;
        end if;

        -- get the current state of the exemplar
```

```
        select into exemplar_state *
        from exemplarstate
        where exemplarid = exemplar_id
          and enddate is null
        order by startdate desc
        limit 1;

        if exemplar_state.locationstatus != 'IN_TRANSIT' then
            raise exception 'Exemplar with id % is not in transit
so it cannot be acquired back to storage', exemplar_id;
        end if;

        -- close the latest state of the exemplar
        update exemplarstate
        set enddate = now()
        where id = exemplar_state.id;

        -- create a new state for the exemplar with locationStatus
set to IN_STORAGE
        insert into exemplarstate (id, locationstatus, startdate,
exemplarid)
        values (uuid_generate_v4(), 'IN_STORAGE', now(),
exemplar_id);

    exception
        when others then
            -- If any operation fails, roll back the transaction
            rollback;
            raise;
    end;
end;
$$;

-----------------------------------------------------
-- 5. Borrow the exemplar from other institution --
-----------------------------------------------------
create or replace function
get_institution_ids_from_emails(institution_emails varchar(100)[])
    returns uuid[] as
$$
declare
    institution_ids uuid[];
begin
    select array_agg(id)
    into institution_ids
    from institution
    where email = any (institution_emails);
```

```
    return institution_ids;
end;
$$ language plpgsql;

create or replace function
institution_email_exists(institution_email varchar(100))
    returns boolean as
$$
begin
    return exists(select 1 from institution where email =
institution_email);
end;
$$ language plpgsql;

create or replace procedure acquire_new_exemplar_from_institution(
    exemplar_name varchar(100),
    institution_name varchar(100),
    institution_address varchar(255),
    institution_email varchar(100),
    institution_type InstitutionType,
    owner_institution_emails varchar(100)[],
    owner_acquisition_date timestamp[],
    owner_share float4[],
    institution_phone varchar(20) = null,
    institution_website varchar(255) = null,
    institution_description varchar(255) = null,
    exemplar_description varchar(255) = null,
    exemplar_year timestamp = null,
    exemplar_estprice numeric = null,
    exemplar_location varchar(100) = null,
    category_names varchar(100)[] = null,
    category_descriptions varchar(255)[] = null
)
    language plpgsql
as
$$
declare
    new_institution_id    uuid;
    new_exemplar_id       uuid;
    owner_institution_ids uuid[];
    email                 varchar(100);
begin
    -- Start a transaction
    begin
        -- Check if the institution email exists, if not, create a
new one institution
        if not institution_email_exists(institution_email) then
            new_institution_id := uuid_generate_v4();
```

```
            insert into institution (id, name, address, email,
phone, website, description, type, createdat)
            values (new_institution_id, institution_name,
institution_address, institution_email, institution_phone,
                    institution_website, institution_description,
institution_type, now());
        end if;

        foreach email in array owner_institution_emails
            loop
                if not institution_email_exists(email) then
                    raise exception 'Institution with email % does
not exist', email;
                end if;
            end loop;

        owner_institution_ids :=
get_institution_ids_from_emails(owner_institution_emails);
        new_exemplar_id := uuid_generate_v4();

        -- Create new exemplar and set the initial state for the
new exemplar to borrowed
        call insert_new_exemplar(
                exemplar_id := new_exemplar_id,
                exemplar_name := exemplar_name,
                exemplar_description := exemplar_description,
                exemplar_year := exemplar_year,
                exemplar_estprice := exemplar_estprice,
                exemplar_location := exemplar_location,
                exemplarstate_location_status := 'BORROWED',
                exemplarstate_desc := 'This is new exemplar we are
looking forward to acquire it into our museum collection',
                category_names := category_names,
                category_descriptions := category_descriptions,
                owner_institution_ids := owner_institution_ids,
                owner_acquisition_date := owner_acquisition_date,
                owner_share := owner_share
            );

    exception
        when others then
            -- If any operation fails, roll back the transaction
            rollback;
            raise;
    end;
end;
$$;
```

2. Simulácia procesov - scenarios.sql

```
create extension if not exists "uuid-ossp";

----------------------------
-- 1. Plan new exposition --
----------------------------
select *
from exposition;

select *
from expositionstate;

select *
from exhibitioninformation;

-- Test 1 - pass: Plan an exposition without conflicts
call plan_exposition(
       exposition_name := 'Exposition 1',
       zone_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-000000000001']::uuid[],
       exposition_description := 'This is a description for
Exposition 1',
       expositionstate_description := 'This is a state description
for Exposition 1',
       exposition_start_date := '2024-05-15 00:00:00.000000',
       exposition_end_date := '2024-05-31 23:59:59.999999'
    );

-- Test 2 - fail: Plan a new exposition with a planning zone
conflict
-- First, we plan an exposition in the same zone - pass
call plan_exposition(
       exposition_name := 'Exposition 2',
       zone_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-000000000001']::uuid[], -- Assuming this
zone ID exists in database
       exposition_description := 'This is a description for
Exposition 2',
       expositionstate_description := 'This is a state description
for Exposition 2',
       exposition_start_date := '2024-06-01 00:00:00.000000',
       exposition_end_date := '2024-06-30 23:59:59.999999'
    );

-- Then, we try to plan another exposition in the same zone and
the same time period, which should raise an exception - fail
call plan_exposition(
```

```
        exposition_name := 'Exposition 3',
        zone_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-000000000001']::uuid[],
        exposition_description := 'This is a description for
Exposition 3',
        expositionstate_description := 'This is a state description
for Exposition 3',
        exposition_start_date := '2024-06-15 00:00:00.000000',
        exposition_end_date := '2024-07-15 23:59:59.999999'
    );

-- Test 3 - fail: Plan a new exposition with a planning zone
conflict time collision - planning without end date
call plan_exposition(
        exposition_name := 'Exposition 4',
        zone_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-000000000001']::uuid[],
        exposition_description := 'This is a description for
Exposition 4',
        expositionstate_description := 'This is a state description
for Exposition 4',
        exposition_start_date := '2024-05-01 00:00:00.000000'
    );

---------------------------------
-- 2. Insert of a new exemplar --
---------------------------------
select *
from exemplar;

select *
from exemplarstate;

select *
from category;

select *
from exemplarownership;

select *
from institution;

-- Test 1 - pass: Insert a new exemplar with a non-existing
categories
call insert_new_exemplar(
        exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000001',
        exemplar_name := 'exemplar 1',
        exemplar_description := 'exemplar 1 description',
```

```
        exemplar_year := '0100-01-01',
        exemplar_estprice := 100000,
        exemplar_location := 'location 1',
        category_names := array ['category1', 'category2', 'machine
learning']::varchar(100)[], -- creates new categories 1 and 2
        category_descriptions := array ['description1',
'description2', null]::varchar(255)[],
        owner_institution_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b',
'f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3c']::uuid[],
        owner_acquisition_date := array ['2022-01-01
21:16:17.293281', '2022-01-01 21:16:17.293281']::timestamp[],
        owner_share := array [0.5]::float4[]
    );

-- Test 2 - fail: Insert a new exemplar with a non-existing
institution
call insert_new_exemplar(
        exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000002',
        exemplar_name := 'exemplar 2',
        exemplar_description := 'exemplar 2 description',
        exemplar_year := '0100-01-01',
        exemplar_estprice := 100000,
        exemplar_location := 'location 2',
        category_names := array ['data science']::varchar(100)[],
        category_descriptions := array ['description1',
'description2', null]::varchar(255)[],
        owner_institution_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3f']::uuid[],
        owner_acquisition_date := array ['2022-01-01
21:16:17.293281']::timestamp[],
        owner_share := array [0.5]::float4[]
    );

-- Test 3 - pass: Insert another new exemplar
call insert_new_exemplar(
        exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000003',
        exemplar_name := 'exemplar 3',
        exemplar_description := 'exemplar 3 description',
        exemplar_year := '0100-01-01',
        exemplar_estprice := 100000,
        exemplar_location := 'location 3',
        exemplarstate_location_status := 'IN_TRANSIT',
        category_names := array ['blockchain']::varchar(100)[],
        category_descriptions := array
['description1']::varchar(255)[],
        owner_institution_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-3b3b3b3b3b3b']::uuid[],
```

```
       owner_acquisition_date := array ['2022-01-01
21:16:17.293281']::timestamp[],
       owner_share := array [1]::float4[]
    );

-------------------------------------------
-- 3. Move of the exemplar to other zone --
-------------------------------------------
select *
from exposition;

select *
from expositionstate;

select *
from exhibitioninformation;

select *
from exemplarstate;

-- pass - First, we create some ongoing expositions and make sure
to crete exemplars
call plan_exposition(
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000001',
       exposition_name := 'Exposition ONGOING 1',
       zone_ids := array ['f0b3b3b3-3b3b-3b3b-3b3b-000000000001',
'f0b3b3b3-3b3b-3b3b-3b3b-000000000002']::uuid[],
       exposition_state := 'ONGOING',
       exposition_description := 'This is a description for
Exposition ONGOING 1',
       expositionstate_description := 'This is a state description
for Exposition ONGOING 1',
       exposition_start_date := now()::timestamp,
       exposition_end_date := '2024-05-1 00:00:00.000000'
    );

call plan_exposition(
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000002',
       exposition_name := 'Exposition ONGOING 2',
       zone_ids := array
['f0b3b3b3-3b3b-3b3b-3b3b-000000000003']::uuid[],
       exposition_state := 'ONGOING',
       exposition_description := 'This is a description for
Exposition ONGOING 2',
       expositionstate_description := 'This is a state description
for Exposition ONGOING 2',
       exposition_start_date := now()::timestamp,
       exposition_end_date := '2024-05-1 00:00:00.000000'
```

```
    );

-- Test 1 - fail: Move an exemplar to a zone that does not exist
call move_exemplar_to_zone(
       exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000001',
       zone_id := 'f0b3b3b3-3b3b-3b3b-3b3b-000000000000',
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000001'
    );

-- Test 2 - fail: Move an exemplar which does not exist
call move_exemplar_to_zone(
       exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000000',
       zone_id := 'f0b3b3b3-3b3b-3b3b-3b3b-000000000001',
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000002'
    );

-- Test 3 - fail: Move an exemplar to exposition which does not
exist
call move_exemplar_to_zone(
       exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000000',
       zone_id := 'f0b3b3b3-3b3b-3b3b-3b3b-000000000001',
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000002'
    );

-- Test 4 - fail: Move an exemplar which is in transit
call move_exemplar_to_zone(
       exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000003',
       zone_id := 'f0b3b3b3-3b3b-3b3b-3b3b-000000000001',
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000002'
    );

-- Test 5 - fail: Move an exemplar 1 to exposition 2 which is not
in zone 1
call move_exemplar_to_zone(
       exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000001',
       zone_id := 'f0b3b3b3-3b3b-3b3b-3b3b-000000000001',
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000002'
    );

-- Test 6 - pass: Move an exemplar 1 to exposition 1 into zone 2
call move_exemplar_to_zone(
       exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000001',
       zone_id := 'f0b3b3b3-3b3b-3b3b-3b3b-000000000002',
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000001'
    );

-- Test 7 - pass: Move an exemplar 1 to exposition 2 into zone 3
call move_exemplar_to_zone(
```

```
       exemplar_id := 'f0e3e3e3-3e3e-3e3e-3e3e-000000000001',
       zone_id := 'f0b3b3b3-3b3b-3b3b-3b3b-000000000003',
       exposition_id := 'f0c3c3c3-3c3c-3c3c-3c3c-000000000002'
    );

-----------------------------------------------------------------
-- 4. Acquire back borrowed exemplar from another institution --
-----------------------------------------------------------------
select *
from exemplarstate e
where e.exemplarid = 'f0e3e3e3-3e3e-3e3e-3e3e-000000000003';

-- Test 1 - pass: Acquire back exemplar 4
call acquire_back_exemplar('f0e3e3e3-3e3e-3e3e-3e3e-000000000004');

-- Test 2 - fail: Acquire back exemplar 4 again which is not in
transit
call acquire_back_exemplar('f0e3e3e3-3e3e-3e3e-3e3e-000000000004');

-------------------------------------------------
-- 5. Borrow the exemplar from other institution --
-------------------------------------------------
select *
from exemplar;

select *
from exemplarstate;

select *
from exemplarownership;

select *
from institution;

-- Test 1 - pass: Acquire exemplar 5
call acquire_new_exemplar_from_institution(
       exemplar_name := 'exemplar 5',
       exemplar_description := 'exemplar 5 description',
       exemplar_year := '0100-01-01',
       exemplar_estprice := 100000,
       exemplar_location := 'location 5',
       category_names := array ['blockchain', 'artificial
intelligence']::varchar(100)[],
       category_descriptions := array ['description1',
'description2']::varchar(255)[],
       owner_institution_emails := array ['technology@museum.com',
'inst4@seznam.cz']::varchar(100)[],
```

```
        owner_acquisition_date := array ['2022-01-01
21:16:17.293281', '2022-01-01 21:16:17.293281']::timestamp[],
        owner_share := array [0.7, 0.3]::float4[],
        institution_name := 'Institution 4',
        institution_address := 'Address 4',
        institution_email := 'inst4@seznam.cz',
        institution_type := 'UNIVERSITY',
        institution_phone := '1234512345',
        institution_website := 'www.inst4.cz',
        institution_description := 'This is a description for
Institution 4'
    );

-- Test 2 - fail: Acquire exemplar 6 from non existing institution
call acquire_new_exemplar_from_institution(
        exemplar_name := 'exemplar 6',
        exemplar_description := 'exemplar 6 description',
        exemplar_year := '0100-01-01',
        exemplar_estprice := 100000,
        exemplar_location := 'location 6',
        category_names := array ['blockchain', 'artificial
intelligence']::varchar(100)[],
        category_descriptions := array ['description1',
'description2']::varchar(255)[],
        owner_institution_emails := array ['random@museum.com',
'inst4@seznam.cz']::varchar(100)[],
        owner_acquisition_date := array ['2022-01-01
21:16:17.293281', '2022-01-01 21:16:17.293281']::timestamp[],
        owner_share := array [0.7, 0.3]::float4[],
        institution_name := 'Institution 4',
        institution_address := 'Address 4',
        institution_email := 'inst4@seznam.cz',
        institution_type := 'UNIVERSITY',
        institution_phone := '1234512345',
        institution_website := 'www.inst4.cz',
        institution_description := 'This is a description for
Institution 4'
    );
```