STU FIIT

Návrh Databázy

DBS

Norbert Matuška
4-6-2024

Contents

Opis vzťahov medzi tabuľkami

Item a Category

Každý exemplár patrí do presne jednej kategórie, zatiaľ čo jedna kategória môže obsahovať viacero exemplárov. Tento vzťah je reprezentovaný foreign key CategoryID v tabuľke Item. Vzťah je preto One-to-Many.

Item a ItemInstitutionLoan

Exemplár môže byť požičaný viacerým inštitúciám v rôznych časoch, reprezentované pomocou foreign key ItemID v tabuľke ItemInstitutionLoan. One-to-Many.

Item a ItemStateLog

Pre jeden exemplár môže existovať viacero záznamov stavu v logu (ItemStateLog), čo umožňuje sledovať históriu stavov predmetu. Vzťah je reprezentovaný foreign key ItemID v ItemStateLog. One-to-Many.

Exhibition a ExhibitionSpaceZone

Jedna výstava môže zahrňovať viacero priestorových zón (SpaceZone) prostredníctvom tabuľky ExhibitionSpaceZone, kde je ExhibitionID foreign key. One-to-Many

Institution a ItemInstitutionLoan

Inštitúcia môže požičiavať viacero exemplárov, ale každé požičiavanie (ItemInstitutionLoan) je spojené s presne jednou inštitúciou. Vzťah je reprezentovaný foreign key InstitutionID v tabuľke ItemInstitutionLoan. Znova je to teda One-to-Many.

SpaceZone a ExhibitionSpaceZone

Tabuľka SpaceZone predstavuje lokality pre usporiadanie výstav. Tabuľka ExhibitionSpaceZone určuje kde sa konkrétna výstava nachádza alebo kde sa má konáť. Vzťah medzi SpaceZone a ExhibitionSpaceZone je typu One-to-Many, čo znamená, že jedna konkrétna priestorová zóna môže byť súčasťou viacerých rôznych výstav, ktoré sa nekonajú súčasne.

Exhibition a ItemStateLog

Jedna výstava môže byť asociovaná s viacerými záznamami v ItemStateLog cez foreign key ExhibitionID. Toto umožňuje sledovať, ktoré predmety boli súčasťou danej výstavy.

Item a ItemExhibition

// Oversight, má to podobný princíp ako Exhibition a ItemStateLog a to že sledovať ktoré exempláre boli súčasťou výstavy, ale tlačí ma čas a nestíham to zmeniť. Rozdiel je v tom, že ItemExhibition neposkytuje kompletnú históriu stavu exemplára, ale len jeho asociovanie s výstavami. Je to teda dôležité na plánovanie a organizáciu výstav, aj keď možno by sa to dalo aj bez tejto tabulky. (Nedostatok spánku, ospravedlňujem sa)

Exhibition a ItemExhibition

Každá výstava môže zahŕňať viacero exemplárov prostredníctvom ItemExhibition, kde ExhibitionID je foreign key. Vzťah je preto One-to-Many

Trigger funkcie

check_item_exhibition_overlap()

Táto funkcia je navrhnutá tak, aby pred vložením nového záznamu do tabuľky ItemExhibition zistila, či sa daný exemplár už neprekrýva s inou výstavou, ktorá sa časovo prelína s novou výstavou. Funkcia skontroluje, či existuje už existujúci záznam v ItemExhibition, kde ItemID je rovnaké ako ItemID v novom zázname a kde dátumy existujúcej výstavy sa prekrývajú s dátumami výstavy nového záznamu.

```
CREATE OR REPLACE FUNCTION check_item_exhibition_overlap()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM ItemExhibition ie
        INNER JOIN Exhibition e ON ie.ExhibitionID = e.ExhibitionID
        WHERE ie.ItemID = NEW.ItemID
        AND e.ExhibitionID <> NEW.ExhibitionID
        AND (
            (e.StartDate <= (SELECT EndDate FROM Exhibition WHERE ExhibitionID = NEW.ExhibitionID)
            AND
            e.EndDate >= (SELECT StartDate FROM Exhibition WHERE ExhibitionID = NEW.ExhibitionID))
        )
    ) THEN
        RAISE EXCEPTION 'Item is already part of an ongoing exhibition.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_exhibition_overlap
BEFORE INSERT ON ItemExhibition
FOR EACH ROW EXECUTE FUNCTION check_item_exhibition_overlap();
```

check_item_available_before_exhibition()

Cieľom tejto funkcie je overiť, že exemplár môže byť súčasťou výstavy len vtedy, ak je jeho posledný známy stav available. Funkcia vyhľadá posledný záznam v ItemStateLog pre daný exemplár podľa ItemID a skontroluje, či je stav predmetu 'available'.

```
CREATE OR REPLACE FUNCTION check_item_exhibition_overlap()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM ItemExhibition ie
        INNER JOIN Exhibition e ON ie.ExhibitionID = e.ExhibitionID
        WHERE ie.ItemID = NEW.ItemID
        AND e.ExhibitionID <> NEW.ExhibitionID
        AND (
            (e.StartDate <= (SELECT EndDate FROM Exhibition WHERE ExhibitionID = NEW.ExhibitionID)
            AND
            e.EndDate >= (SELECT StartDate FROM Exhibition WHERE ExhibitionID = NEW.ExhibitionID))
        )
    ) THEN
        RAISE EXCEPTION 'Item is already part of an ongoing exhibition.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_exhibition_overlap
BEFORE INSERT ON ItemExhibition
FOR EACH ROW EXECUTE FUNCTION check_item_exhibition_overlap();
```

check_spacezone_exhibition_overlap()

Táto funkcia kontroluje pred vložením alebo aktualizáciou záznamu v ExhibitionSpaceZone, či už neexistuje iná výstava, ktorá by sa časovo prekrývala v rovnakej SpaceZone. Funkcia vyhľadá v tabuľke ExhibitionSpaceZone pre danú SpaceZone (identifikovanú pomocou SpaceID) a skontroluje, či sú dátumy novej výstavy kompatibilné s už existujúcimi výstavami v tejto zóne.

```
CREATE OR REPLACE FUNCTION check_spacezone_exhibition_overlap()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM ExhibitionSpaceZone esz
        INNER JOIN Exhibition e ON esz.ExhibitionID = e.ExhibitionID
        WHERE esz.SpaceID = NEW.SpaceID
        AND e.ExhibitionID != NEW.ExhibitionID
        AND NOT (
            NEW.StartDate > e.EndDate OR
            NEW.EndDate < e.StartDate
        )
    ) THEN
        RAISE EXCEPTION 'Another exhibition is already scheduled.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_spacezone_exhibition_overlap
BEFORE INSERT OR UPDATE ON ExhibitionSpaceZone
FOR EACH ROW EXECUTE FUNCTION check_spacezone_exhibition_overlap();
```

check_exhibition_dates()

Táto funkcia kontroluje len nech sa nedá vytvoriť výstava ktorá končí skorej ako začína.

```
CREATE OR REPLACE FUNCTION check_exhibition_dates()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.EndDate < NEW.StartDate THEN
        RAISE EXCEPTION 'End date cannot be before start date.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_exhibition_date_check
BEFORE INSERT ON Exhibition
FOR EACH ROW EXECUTE FUNCTION check_exhibition_dates();
```

check_item_state_before_loan

Funkcia kontroluje, či je možné požičať exemplár inej inštitúcii a teda či je exemplár available.

```sql
CREATE OR REPLACE FUNCTION check_item_state_before_loan()
RETURNS TRIGGER AS $$
DECLARE
    latestState VARCHAR(100);
BEGIN
    SELECT State INTO latestState
    FROM ItemStateLog
    WHERE ItemID = NEW.ItemID
    ORDER BY Date DESC, LogID DESC
    LIMIT 1;

    IF latestState != 'available' THEN
        RAISE EXCEPTION 'Cannot loan item because it is currently in state: %', latestState;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_item_state_before_loan
BEFORE INSERT ON ItemInstitutionLoan
FOR EACH ROW EXECUTE FUNCTION check_item_state_before_loan();
```

Základné procesy múzea

Naplánovanie výstavy

Toto záhrňa vytvorenie záznamu do tabuľky Exhibition a asociovanie tohto záznamu so SpaceZone a Item tabuľkami. V SQL by to vyzeralo nejak takto:
INSERT INTO Exhibition (Name, Description, StartDate, EndDate) VALUES ('Exhibition Name', 'Description', 'StartDate', 'EndDate')

INSERT INTO ExhibitionSpaceZone (ExhibitionID, SpaceID) VALUES ('ExhibitionID', 'SpaceID')

Vkladanie nového exempláru

Vytvoríme nový záznam v tabuľke Item a taktiež aj logneme počiatočný stav:
INSERT INTO Item (Name, Description, State, CategoryID) VALUES ('Item Name', 'Description', 'State', 'CategoryID')

INSERT INTO ItemStateLog (ItemID, State, Date, Location) VALUES ('ItemID', 'Initial State', CURRENT_TIMESTAMP, 'Location')

Presun exempláru do inej zóny

Keď sa exemplar presunie do inej zóny, updateneme v ItemStateLog jeho location:

INSERT INTO ItemStateLog (ItemID, State, Date, Location) VALUES ('ItemID', 'State', CURRENT_TIMESTAMP, 'New Zone')

Prevzatie exempláru z inej inštitúcie

Keďže sme prevzali exemplar od inej inštitúcie, a nebolo to zapožičané, zoberieme to rovnako ako vkladanie nového exempláru.

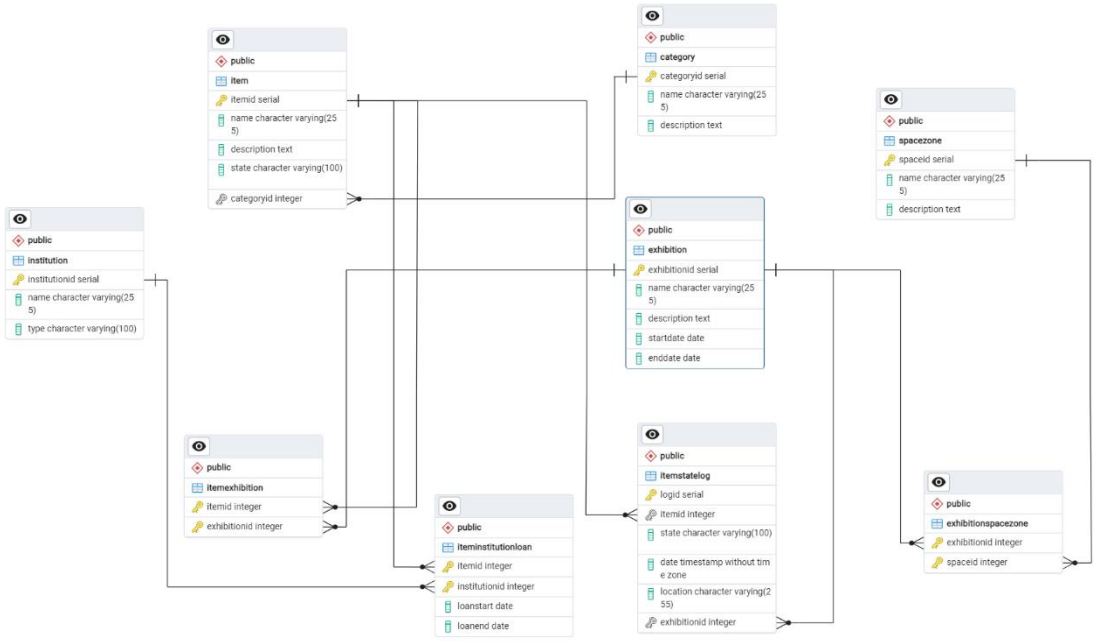
Zapožičanie exempláru z inej inštitúcie

Spravíme záznam o exemplári, vytvoríme aj záznam o zapožičaní. Vhodné by možno aj bolo vytvoriť trigger funkciu ktorá kontroluje, ak by sme chceli vložiť exemplár do výstavy, či nieje náhodou po ubehnutej lehote požičania a teda ho musíme vrátiť.
INSERT INTO ItemInstitutionLoan (ItemID, InstitutionID, LoanStart, LoanEnd) VALUES ('ItemID', 'InstitutionID', 'LoanStart', 'LoanEnd');

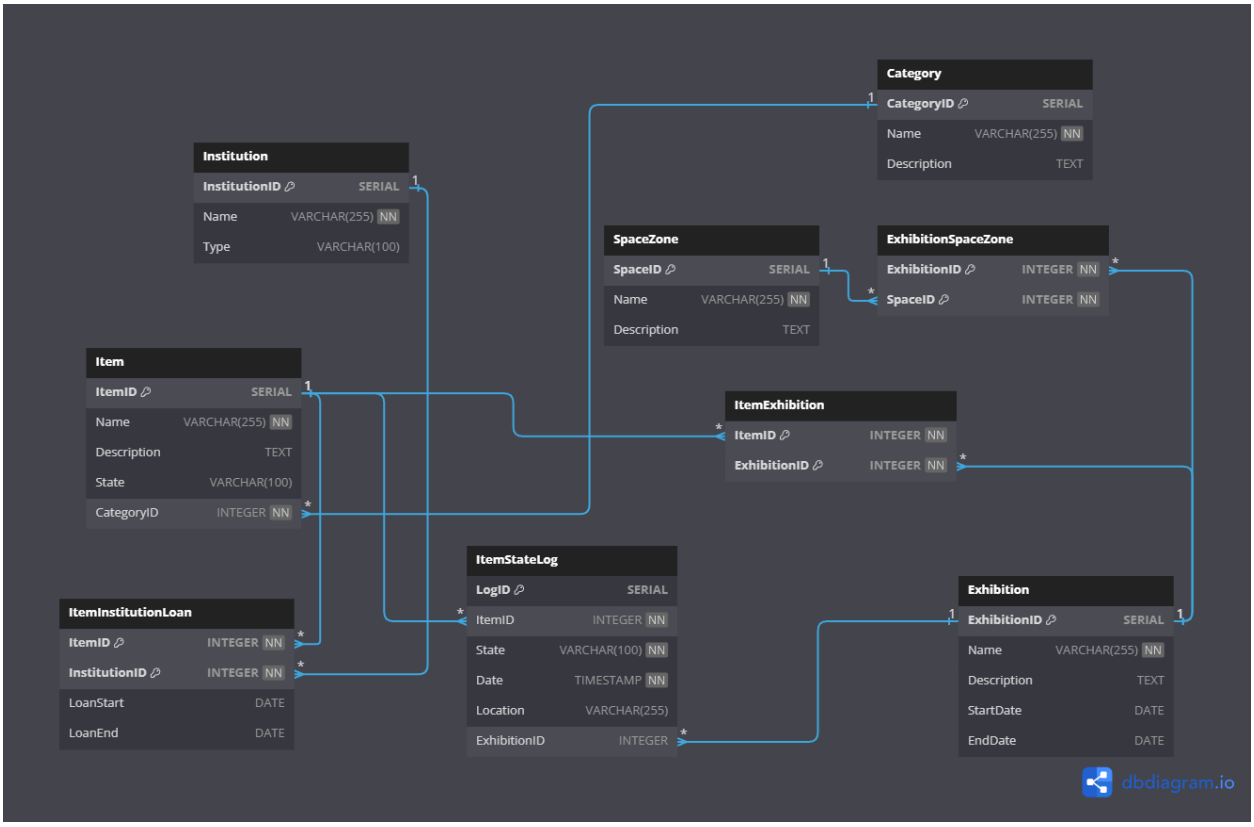
Prílohy

V zip súbore vedľa tejto dokumentácie nájdete aj ER diagram aj relačný diagram spolu s .sql súborom. Prilepím to ale taktiež aj sem

ERD



Relačný Diagram



Zmeny

- Location variable v itemstatelog je teraz prepojená so spacezone, aby sme mohli sledovať, kde sa exemplár nachádza

Testovacie scenáre

Plánovanie výstavy

Valídny insert:

INSERT INTO Exhibition (Name, Description, StartDate, EndDate)

VALUES ('Spring Art Collection', 'A collection of modern art pieces for spring.', '2024-05-01', '2024-05-31');

Zlý príklad:

INSERT INTO Exhibition (Name, Description, StartDate, EndDate)

VALUES ('Impossible Exhibition', 'This exhibition has conflicting dates.', '2024-06-01', '2024-05-01');

Vkladanie nového exempláru

INSERT INTO Item (Name, Description, State, CategoryID)

VALUES ('Ancient Vase', 'A decorative vase from ancient Greece.', 'available', 1);

Presúvanie exempláru do inej zóny:

UPDATE ItemStateLog

SET SpaceID = 2

WHERE ItemID = 1;

Prevzatie exempláru od inej inštitúcie

To isté ako vloženie exempláru

Zapožičanie exempláru

INSERT INTO ItemInstitutionLoan (ItemID, InstitutionID, LoanStart, LoanEnd)

VALUES (1, 2, '2024-05-01', '2024-05-15');

Nevalídne zapožičanie:

```
UPDATE ItemStateLog

SET State = 'maintenance'

WHERE ItemID = 1;


INSERT INTO ItemInstitutionLoan (ItemID, InstitutionID, LoanStart, LoanEnd)

VALUES (1, 2, '2024-05-01', '2024-05-15');
```