Slovenská technická univerzita
Fakulta informatiky a informačných technológií v Bratislave

Dokumentácia k zadaniu 5

Implementácia návrhu databázy zo zadania 4

Autor: Dominik Mifkovič
Predmet: Databázové systémy
Cvičiaci: Ing. William Brach
Obdobie: Letný semester 2024

Obsah

1 Hlavné tabuľky

1.1 Opis atribútov

Categories: Tabuľka obsahuje kategórie, do ktorých sa exempláre zaraďujú.
Atribúty:

- id: Identifikátor kategórie (primárny kľúč).
- name: Názov kategórie (jedinečný).

Exemplars: Kľúčová tabuľka s informáciami o exemplároch.
Atribúty:

- id: Identifikátor exemplára (primárny kľúč).
- category_id: Referencia na kategóriu, do ktorej exemplár patrí (cudzí kľúč odkazujúci na categories.id).
- name: Názov exemplára (jedinečný).
- description: Popis exemplára.
- ownership_status: Stav vlastníctva (vlastnený/zapožičaný).
- current_status: Aktuálny stav exemplára (v sklade, na vystavení, kontrola, na ceste).
- loan_id: Referencia na záznam o zapožičaní (cudzí kľúč odkazujúci na loans.id).

Expositions: Tabuľka s informáciami o expozíciách.
Atribúty:

- id: Identifikátor expozície (primárny kľúč).
- name: Názov expozície (jedinečný).
- start_date: Dátum začatia expozície.
- end_date: Dátum ukončenia expozície.
- status: Stav expozície (plánovaná, prebiehajúca, ukončená).

Zones: Tabuľka s informáciami o priestorových zónach.
Atribúty:

- id: Identifikátor zóny (primárny kľúč).
- name: Názov zóny.

Institutions: Tabuľka s informáciami o inštitúciách zapojených do zapožičiavania.
Atribúty:

- id: Identifikátor inštitúcie (primárny kľúč).
- name: Názov inštitúcie (jedinečný).

Loans: Tabuľka s detailmi o zapožičaných exemplároch.
Atribúty:

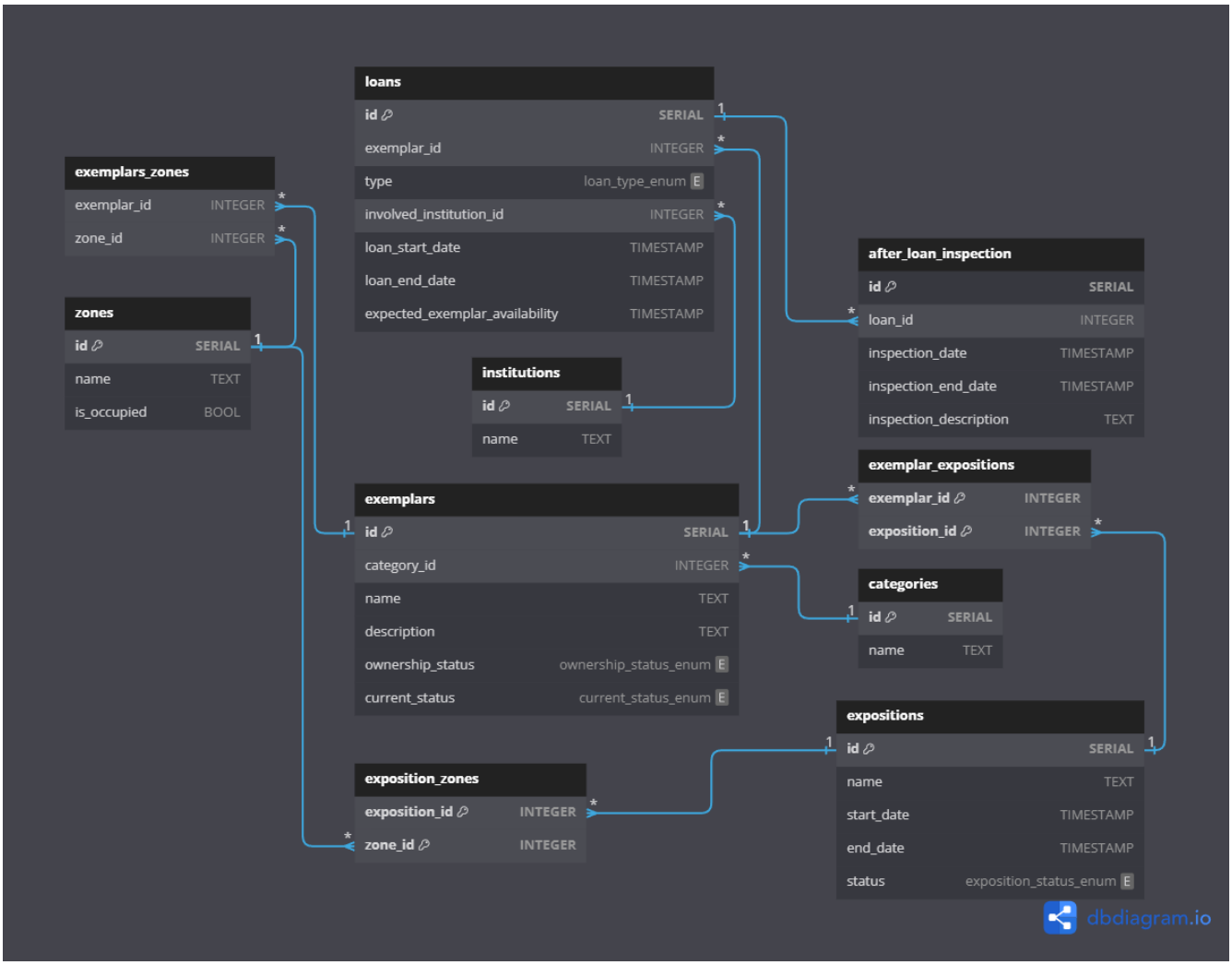
- id: Identifikátor záznamu o zapožičaní (primárny kľúč).
- exemplar_id: Referencia na exemplár, ktorý bol zapožičaný (cudzí kľúč odkazujúci na exemplars.id).
- type: Typ zapožičania (zapožičané inej inštitúcií/od inej inštitúcií).
- involved_institution_id: Referencia na zapojenú inštitúciu (cudzí kľúč odkazujúci na institutions.id).
- loan_start_date: Dátum začatia zapožičania.

- loan_end_date: Dátum ukončenia zapožičania.
- expected_exemplar_availability: Očakávaný dátum dostupnosti exemplára po návrate.

AfterLoanInspection: Tabuľka s informáciami o kontrolách po ukončení zapožičania.
Atribúty:

- id: Identifikátor záznamu o kontrole (primárny kľúč).
- loan_id: Referencia na záznam o zapožičaní (cudzí kľúč odkazujúci na loans.id).
- inspection_date: Dátum kontroly.
- inspection_end_date: Dátum ukončenia kontroly.
- inspection_description: Popis zistení z kontroly.

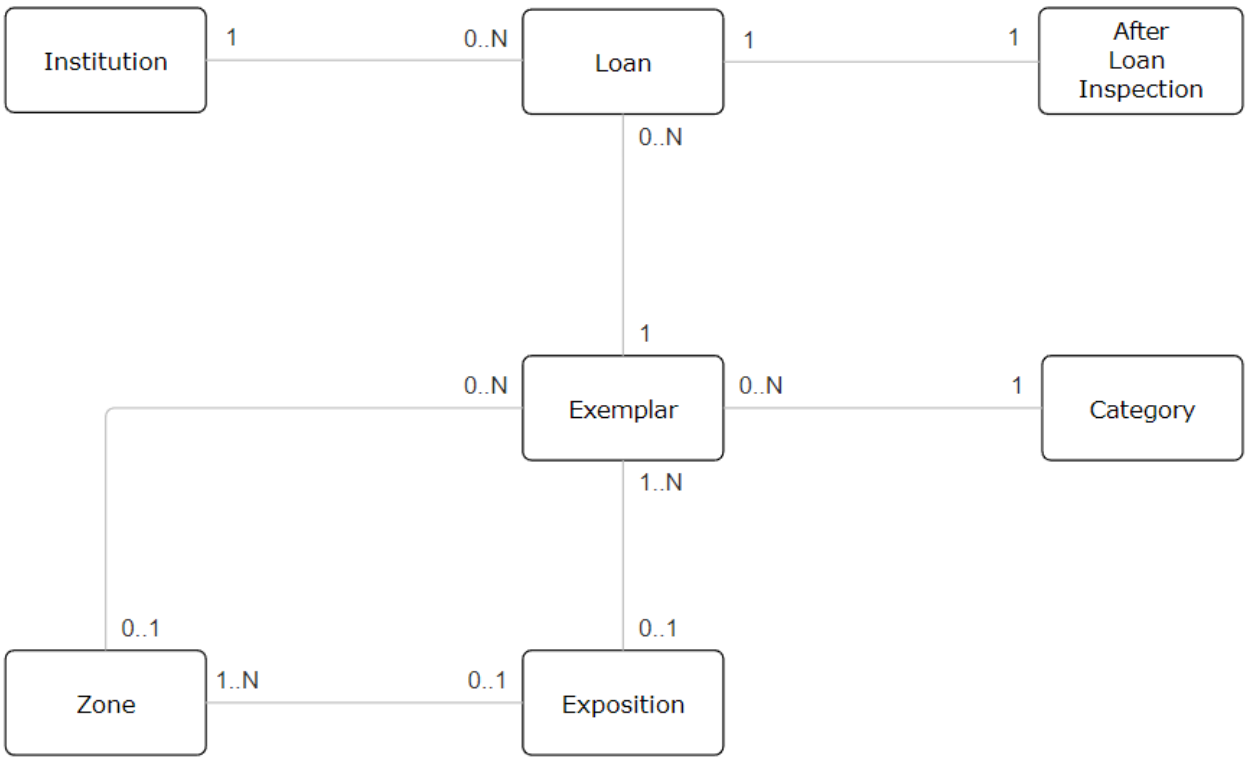
1.2 Schéma databázy



2 Entity

Category: Kategórie, do ktorých sa exempláre zaraďujú.
Exemplar: Exempláre s rôznymi vlastnosťami (stav, umiestnenie, história).
Exposition: Expozície s definovaným dátumom konania a umiestnením v zónach.
Zone: Priestorové zóny, v ktorých prebiehajú expozície.
Institution: Inštitúcie zapojené do procesu zapožičiavania exemplárov.
Loan: Záznamy o zapožičaných exemplároch s detailmi o inštitúcii a dátume.
AfterLoanInspection: Záznamy o kontrolách exemplárov po ich návrate z zapožičania.

2.1 Entitno-Relačný diagram



2.2 Opis vzťahov medzi entitami

Exemplar ⟷ Category
Každý exemplár patrí práve do jednej kategórie.
Kategória môže obsahovať viacero exemplárov.

Exemplar ⟷ Exposition
Exemplár môže byť súčasťou 0 až 1 expozície v daný čas.
Expozícia môže obsahovať 1 až N exemplárov v daný čas.

Exemplar ⟷ Zone
Exemplár môže byť v 0 až 1 zóne v daný čas.
Zóna môže obsahovať 0 až N exemplárov.

Exemplar ⟷ Loan
Exemplár môže byť viackrát zapožičaný.
Jedno zapožičanie sa týka 1 exemplára.

Exposition ⟷ Zone
Expozícia sa môže rozprestierať vo viacerých zónach.
Zóna môže hostiť maximálne 1 expozíciu v daný čas.

Loan ⟷ Institution
Každé zapožičanie je spojené s 1 inštitúciou.
Inštitúcia sa zúčastňuje na 0 až N zapožičaniach.

Loan ⟷ AfterLoanInspection
Každé zapožičanie má po skončení práve 1 kontrolu.
Kontrola súvisí len s práve 1 zapožičaním.

3 Vytvorenie a počiatočné naplnenie databázy

* Poznámka - zmenený enumerátor v current_status_enum: ON_DISPLAY → IN_EXPO

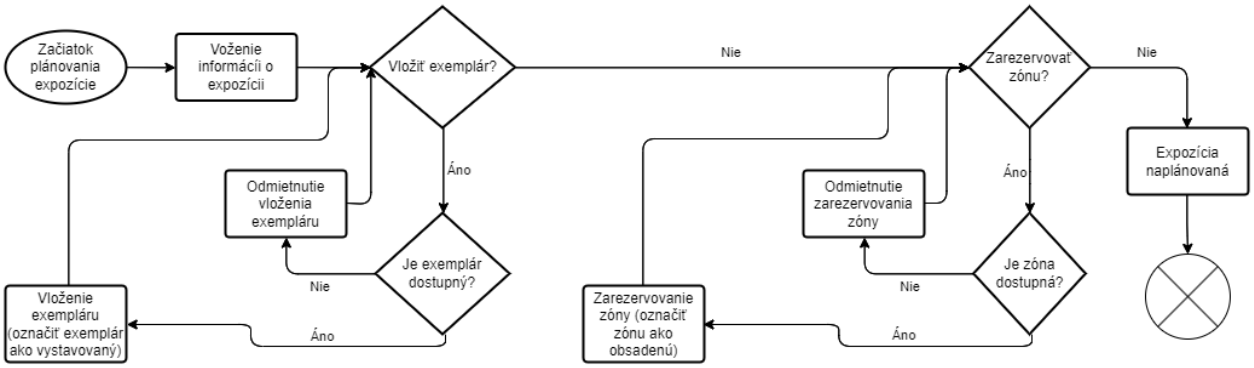
```sql
ALTER TABLE IF EXISTS exemplars_zones DROP CONSTRAINT IF EXISTS exemplars_zones_exemplar_id_fkey;
ALTER TABLE IF EXISTS exemplars_zones DROP CONSTRAINT IF EXISTS exemplars_zones_zone_id_fkey;
ALTER TABLE IF EXISTS loans DROP CONSTRAINT IF EXISTS loans_exemplar_id_fkey;
ALTER TABLE IF EXISTS loans DROP CONSTRAINT IF EXISTS loans_involved_institution_id_fkey;
ALTER TABLE IF EXISTS after_loan_inspection DROP CONSTRAINT IF EXISTS after_loan_inspection_loan_id_fkey;
ALTER TABLE IF EXISTS exemplar_expositions DROP CONSTRAINT IF EXISTS exemplar_expositions_exemplar_id_fkey;
ALTER TABLE IF EXISTS exemplar_expositions DROP CONSTRAINT IF EXISTS exemplar_expositions_exposition_id_fkey;
ALTER TABLE IF EXISTS exposition_zones DROP CONSTRAINT IF EXISTS exposition_zones_exposition_id_fkey;
ALTER TABLE IF EXISTS exposition_zones DROP CONSTRAINT IF EXISTS exposition_zones_zone_id_fkey;

DROP TABLE IF EXISTS exemplars_zones;
DROP TABLE IF EXISTS loans;
DROP TABLE IF EXISTS after_loan_inspection;
DROP TABLE IF EXISTS exemplar_expositions;
DROP TABLE IF EXISTS exposition_zones;
DROP TABLE IF EXISTS exemplars;
DROP TABLE IF EXISTS expositions;
DROP TABLE IF EXISTS zones;
DROP TABLE IF EXISTS institutions;
DROP TABLE IF EXISTS categories;

DROP TYPE IF EXISTS ownership_status_enum;
DROP TYPE IF EXISTS current_status_enum;
DROP TYPE IF EXISTS exposition_status_enum;
DROP TYPE IF EXISTS loan_type_enum;

CREATE TYPE "ownership_status_enum" AS ENUM (
'OWNED',
'LOANED'
);

CREATE TYPE "current_status_enum" AS ENUM (
'IN_STORAGE',
'IN_EXPO',
'BEING_INSPECTED',
'IN_TRANSIT',
'NONE'
);
```

```
39
40  CREATE TYPE "exposition_status_enum" AS ENUM (
41  'PLANNED',
42  'IN_PROGRESS',
43  'ENDED'
44  );
45
46  CREATE TYPE "loan_type_enum" AS ENUM (
47  'LOANED_TO',
48  'LOANED_IN'
49  );
50
51  CREATE TABLE "categories" (
52  "id" SERIAL PRIMARY KEY,
53  "name" TEXT UNIQUE
54  );
55
56  CREATE TABLE "exemplars" (
57  "id" SERIAL PRIMARY KEY,
58  "category_id" INTEGER REFERENCES "categories" ("id"),
59  "name" TEXT UNIQUE,
60  "description" TEXT,
61  "ownership_status" ownership_status_enum DEFAULT 'OWNED',
62  "current_status" current_status_enum DEFAULT 'IN_STORAGE',
63  CONSTRAINT compatible_ownership_status CHECK (
64  (ownership_status = 'OWNED' AND current_status IN ('IN_STORAGE', 'IN_EXPO', 'BEING_INSPECTED'))
65  OR (ownership_status = 'LOANED' AND current_status IN ('IN_STORAGE', 'IN_EXPO', 'BEING_INSPECTED', 'IN_TRANSIT')))
66  );
67
68  CREATE TABLE "expositions" (
69  "id" SERIAL PRIMARY KEY,
70  "name" TEXT UNIQUE,
71  "start_date" TIMESTAMP,
72  "end_date" TIMESTAMP,
73  "status" exposition_status_enum DEFAULT 'PLANNED'
74  );
75
76  CREATE TABLE "zones" (
77  "id" SERIAL PRIMARY KEY,
78  "name" TEXT,
79  "is_occupied" BOOLEAN DEFAULT FALSE
80  );
81
82
83  CREATE TABLE "exemplars_zones" (
84  "exemplar_id" INTEGER REFERENCES "exemplars" ("id") ON DELETE CASCADE ON UPDATE CASCADE UNIQUE,
85  "zone_id" INTEGER REFERENCES "zones" ("id") ON DELETE CASCADE ON UPDATE CASCADE
86  );
87
88  CREATE TABLE "institutions" (
89  "id" SERIAL PRIMARY KEY,
90  "name" TEXT UNIQUE
91  );
92
93  CREATE TABLE "loans" (
94  "id" SERIAL PRIMARY KEY,
95  "exemplar_id" INTEGER REFERENCES "exemplars" ("id"),
96  "type" loan_type_enum DEFAULT 'LOANED_TO',
97  "involved_institution_id" INTEGER REFERENCES "institutions" ("id"),
```

```
"loan_start_date" TIMESTAMP,
"loan_end_date" TIMESTAMP,
"expected_exemplar_availability" TIMESTAMP
);


CREATE TABLE "after_loan_inspection" (
"id" SERIAL PRIMARY KEY,
"loan_id" INTEGER REFERENCES "loans" ("id") ON DELETE CASCADE ON UPDATE CASCADE,
"inspection_date" TIMESTAMP,
"inspection_end_date" TIMESTAMP,
"inspection_description" TEXT
);

CREATE TABLE "exemplar_expositions" (
"exemplar_id" INTEGER REFERENCES "exemplars" ("id") ON DELETE CASCADE ON UPDATE CASCADE,
"exposition_id" INTEGER REFERENCES "expositions" ("id") ON DELETE CASCADE ON UPDATE CASCADE,
PRIMARY KEY ("exemplar_id", "exposition_id")
);

CREATE TABLE "exposition_zones" (
"exposition_id" INTEGER REFERENCES "expositions" ("id") ON DELETE CASCADE ON UPDATE CASCADE,
"zone_id" INTEGER REFERENCES "zones" ("id") ON DELETE CASCADE ON UPDATE CASCADE,
PRIMARY KEY ("exposition_id", "zone_id"),
CONSTRAINT one_exp_per_zone UNIQUE (zone_id)
);


-- Pridanie kategórií
INSERT INTO categories (name) VALUES ('Maľba');
INSERT INTO categories (name) VALUES ('Socha');
INSERT INTO categories (name) VALUES ('Fotografia');

-- Pridanie inštitúcií
INSERT INTO institutions (name) VALUES ('Inštitúcia 1');
INSERT INTO institutions (name) VALUES ('Inštitúcia 2');
INSERT INTO institutions (name) VALUES ('Inštitúcia 3');

-- Pridanie zón
INSERT INTO zones (name) VALUES ('Zóna 1');
INSERT INTO zones (name) VALUES ('Zóna 2');
INSERT INTO zones (name) VALUES ('Zóna 3');

-- Pridanie exemplárov
INSERT INTO exemplars (category_id, name, description, ownership_status, current_status)
VALUES ((SELECT id FROM categories WHERE name = 'Maľba'), 'Maľba 1', 'Popis maľby 1', 'OWNED', 'IN_STORAGE');
INSERT INTO exemplars (category_id, name, description, ownership_status, current_status)
VALUES ((SELECT id FROM categories WHERE name = 'Socha'), 'Socha 1', 'Popis sochy 1', 'OWNED', 'IN_STORAGE');
INSERT INTO exemplars (category_id, name, description, ownership_status, current_status)
VALUES ((SELECT id FROM categories WHERE name = 'Fotografia'), 'Fotografia 1', 'Popis fotografie 1', 'OWNED', 'IN_STORAGE');
```

4 Opis základných procesov

4.1 Plánovanie expozície podľa zadania 4



Expozícia sa bude plánovať tak, že sa vytvorí záznam v tabuľke expositions. Následne sa pomocou spojovacej tabuľky budú pridávať do expozície exempláre, ale iba tie, ktoré majú current_status = 'IN_STORAGE'. Po pridaní sa exemplára nastaví current_status na 'IN_EXPO'. Následne sa bude môcť taktiež pomocou spojovacej tabuľky obsadiť zóna. Ak je dostupná tak sa v spojovacej tabuľke priradí k expozícii a nastaví sa jej hodnota is_occupied = TRUE.

4.1.1 Funkcie a triggery

```
CREATE OR REPLACE FUNCTION create_exposition(exp_name TEXT, exemplare TEXT[], zony TEXT[], start_date TIMESTAMP, end_date TIMESTAMP)
DECLARE
expo_id INTEGER;
exemplar_name TEXT;
zone_name TEXT;
BEGIN
-- Vytvorenie novej expozície
INSERT INTO expositions (name, start_date, end_date)
VALUES (exp_name, start_date, end_date);

-- Získanie ID novej expozície
SELECT id INTO expo_id FROM expositions WHERE name = exp_name;

-- Pridanie exemplárov do expozície
FOR i IN 1..array_length(exemplare, 1) LOOP
exemplar_name := exemplare[i];
INSERT INTO exemplar_expositions (exemplar_id, exposition_id)
VALUES ((SELECT id FROM exemplars WHERE name = exemplar_name), expo_id);
END LOOP;

-- Pridávanie zón
FOR j IN 1..array_length(zony, 1) LOOP
zone_name := zony[j];
INSERT INTO exposition_zones (exposition_id, zone_id)
VALUES (expo_id, (SELECT id FROM zones WHERE name = zone_name));
END LOOP;
END;
$$ LANGUAGE plpgsql;
```

```
-- Pridanie exemplára do expozície

CREATE OR REPLACE FUNCTION add_exemplar_to_exposition() RETURNS TRIGGER AS $$
DECLARE
exemplar_status current_status_enum;
BEGIN
SELECT current_status INTO exemplar_status FROM exemplars WHERE id = NEW.exemplar_id;
IF exemplar_status <> 'IN_STORAGE' THEN
RAISE EXCEPTION 'Exemplár nie je dostupný pre pridanie do expozície.';
-- Ak exemplár nemá status IN_STORAGE, znamená to, že nie je dostupný
ELSE
UPDATE exemplars SET current_status = 'IN_EXPO' WHERE id = NEW.exemplar_id;
RETURN NEW;
-- Exemplára sa nastaví status IN_EXPO
END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER exemplar_status_check BEFORE INSERT OR UPDATE ON exemplar_expositions
FOR EACH ROW EXECUTE PROCEDURE add_exemplar_to_exposition();









-- Obsadenie zóny pre expozíciu

CREATE OR REPLACE FUNCTION occupy_zone() RETURNS TRIGGER AS $$
DECLARE
zone_status BOOLEAN;
first_zone INTEGER;
BEGIN
SELECT is_occupied INTO zone_status FROM zones WHERE id = NEW.zone_id;
SELECT COUNT(*) INTO first_zone FROM exposition_zones WHERE exposition_id = NEW.exposition_id;
IF zone_status = TRUE THEN
RAISE EXCEPTION 'Zóna je už obsadená.';
-- Zóna nie je dostupná pre pridanie
ELSE
UPDATE zones SET is_occupied = TRUE WHERE id = NEW.zone_id;

-- Pridanie exemplárov expozície do zóny len ak je to prvá pridávaná zóna
IF first_zone = 0 THEN
INSERT INTO exemplars_zones (exemplar_id, zone_id)
SELECT exemplar_id, NEW.zone_id FROM exemplar_expositions WHERE exposition_id = NEW.exposition_id;
END IF;

RETURN NEW;
END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER zone_occupancy_check BEFORE INSERT ON exposition_zones
FOR EACH ROW EXECUTE PROCEDURE occupy_zone();
```

```sql
-- Kontrola a nastavenie statusu expozície pri vkladaní alebo úprave času

CREATE OR REPLACE FUNCTION set_exposition_status() RETURNS TRIGGER AS $$
BEGIN
IF (NEW.start_date <= NOW() AND NEW.end_date >= NOW()) THEN
NEW.status := 'IN_PROGRESS';
UPDATE exemplars SET current_status = 'IN_EXPO'
WHERE id IN (SELECT exemplar_id FROM exemplar_expositions WHERE exposition_id = NEW.id);
UPDATE zones SET is_occupied = TRUE WHERE id IN (SELECT zone_id FROM exposition_zones WHERE exposition_id = NEW.id);
-- Ak expozícia prebieha
ELSIF (NEW.end_date < NOW()) THEN
NEW.status := 'ENDED'; -- Ak expozícia skončila, správanie je rovnaké ako pri vymazaní expozície
DELETE FROM exemplars_zones WHERE exemplar_id IN (SELECT exemplar_id FROM exemplar_expositions WHERE exposition_id = NEW.id);
DELETE FROM exemplar_expositions WHERE exposition_id = NEW.id;
DELETE FROM exposition_zones WHERE exposition_id = NEW.id;
UPDATE exemplars SET current_status = 'IN_STORAGE' WHERE id IN (
SELECT id FROM exemplars WHERE current_status = 'IN_EXPO' AND id NOT IN (
SELECT exemplar_id FROM exemplar_expositions
)
);
UPDATE zones SET is_occupied = FALSE WHERE id IN (
SELECT id FROM zones WHERE is_occupied = TRUE AND id NOT IN (
SELECT zone_id FROM exposition_zones
)
);
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER set_exposition_status BEFORE UPDATE ON expositions
FOR EACH ROW EXECUTE PROCEDURE set_exposition_status();
CREATE OR REPLACE TRIGGER set_exposition_status_insert BEFORE INSERT ON expositions
FOR EACH ROW EXECUTE PROCEDURE set_exposition_status();

-- Vymazanie expozície

CREATE OR REPLACE FUNCTION delete_exposition() RETURNS TRIGGER AS $$
BEGIN
-- Nastavenie exemplárov späť na 'IN_STORAGE'
DELETE FROM exemplar_expositions WHERE exposition_id = OLD.id;
DELETE FROM exposition_zones WHERE exposition_id = OLD.id;
UPDATE exemplars SET current_status = 'IN_STORAGE' WHERE id IN (
SELECT id FROM exemplars WHERE current_status = 'IN_EXPO' AND id NOT IN (
SELECT exemplar_id FROM exemplar_expositions
)
);

-- Nastavenie zón späť na neobsadené
UPDATE zones SET is_occupied = FALSE WHERE id IN (
SELECT id FROM zones WHERE is_occupied = TRUE AND id NOT IN (
SELECT zone_id FROM exposition_zones
)
);

RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER exposition_deletion AFTER DELETE ON expositions FOR EACH ROW EXECUTE PROCEDURE delete_exposition();
```

4.1.2 Príklad použitia

Vytvoríme si expozíciu, ktorá je naplánovaná a ďalšiu, ktorá prebieha

```
SELECT create_exposition('PLANNED EXPO'::text, ARRAY['Maľba 1'],  ARRAY['Zóna 1'],
 '2025-01-01'::timestamp, '2025-01-02'::timestamp);
SELECT create_exposition('IN PROGRESS EXPO'::text, ARRAY['Fotografia 1'],
 ARRAY['Zóna 2'], '2023-01-01'::timestamp, '2024-10-10'::timestamp);
```

	id [PK] integer	name text	start_date timestamp without time zone	end_date timestamp without time zone	status exposition_status_enum
1	1	PLANNED EXPO	2025-01-01 00:00:00	2025-01-02 00:00:00	PLANNED
2	2	IN PROGRESS EXPO	2023-01-01 00:00:00	2024-10-10 00:00:00	IN_PROGRESS

Pokúsime sa pridať Zónu 1 do expozície IN PROGRESS EXPO, čo nepôjde, lebo zóna už je obsadená expozíciou PLANNED EXPO

```
DO $$
DECLARE
expo_id INTEGER;
BEGIN
SELECT id INTO expo_id FROM expositions WHERE name = 'IN PROGRESS EXPO';
INSERT INTO exposition_zones (exposition_id, zone_id)
VALUES (expo_id, (SELECT id FROM zones WHERE name = 'Zóna 1'));
END $$;
```

```
ERROR:  Zóna je už obsadená.
CONTEXT:  PL/pgSQL function occupy_zone() line 9 at RAISE
SQL statement "INSERT INTO exposition_zones (exposition_id, zone_id)
    VALUES (expo_id, (SELECT id FROM zones WHERE name = 'Zóna 1'))"
PL/pgSQL function inline_code_block line 6 at SQL statement

SQL state: P0001
```

Pridáme teda Zónu 3, čo už zbehne v poriadku lebo tá nie je obsadená.

```
DO $$
DECLARE
expo_id INTEGER;
BEGIN
SELECT id INTO expo_id FROM expositions WHERE name = 'IN PROGRESS EXPO';
INSERT INTO exposition_zones (exposition_id, zone_id)
VALUES (expo_id, (SELECT id FROM zones WHERE name = 'Zóna 3'));
END $$;
```

Stav zón po týchto procesoch:

	id [PK] integer	name text	is_occupied boolean
1	1	Zóna 1	true
2	2	Zóna 2	true
3	3	Zóna 3	true

Stav exemplárov pred zmenou dátumu expozícií:

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_EXPO

Upravíme dátumy, tak aby sa automaticky zmenili stavy expozícií, PLANNED EXPO začne a IN PROGRESS EXPO skončí:

```
1 UPDATE expositions SET start_date = '2024-01-01' WHERE name = 'PLANNED EXPO';
2 UPDATE expositions SET end_date = '2024-03-01' WHERE name = 'IN PROGRESS EXPO';
```

	id [PK] integer	name text	start_date timestamp without time zone	end_date timestamp without time zone	status exposition_status_enum
1	1	PLANNED EXPO	2024-01-01 00:00:00	2025-01-02 00:00:00	IN_PROGRESS
2	2	IN PROGRESS EXPO	2023-01-01 00:00:00	2024-03-01 00:00:00	ENDED

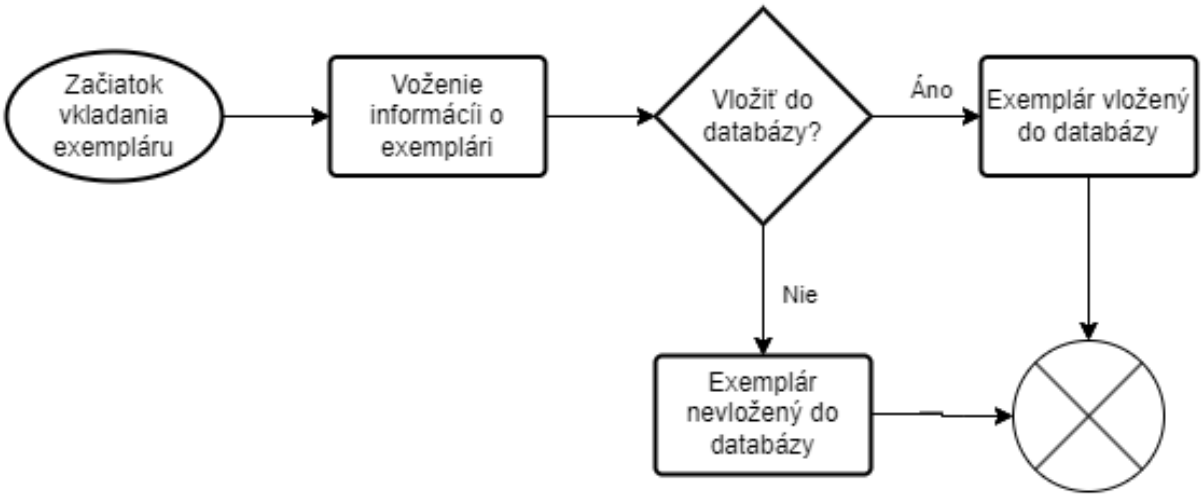
Stav exemplárov a zón po zmene dátumu expozícií (exempláre a zóny expozície IN PROGRESS EXPO sa uvoľnia):

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_STORAGE

	id [PK] integer	name text	is_occupied boolean
1	1	Zóna 1	true
2	2	Zóna 2	false
3	3	Zóna 3	false

Správanie pri vymazaní expozície je rovnaké ako pri jej skončení. Taktiež sa uvoľnia všetky zóny a exempláre, ktoré mala zóna zabraté.

4.2 Vkladanie exemplára (do databázy) podľa zadania 4



Klasické vkladanie nového údaju do databázy.

4.2.1 Príklad použitia

Najskôr sa pokúsime pridať exemplár, ktorý už existuje. To nepôjde a vypíše sa chyba.

```
1 INSERT INTO exemplars (category_id, name, description, ownership_status, current_status)
2 VALUES ((SELECT id FROM categories WHERE name = 'Socha'), 'Socha 1', 'Popis', 'OWNED', 'IN_STORAGE');
```

```
ERROR:  Key (name)=(Socha 1) already exists.duplicate key value violates unique constraint "exemplars_name_key"

ERROR:  duplicate key value violates unique constraint "exemplars_name_key"
SQL state: 23505
Detail: Key (name)=(Socha 1) already exists.
```

Potom sa pokúsime pridať exemplár, ktorý ešte neexistuje. To zbehne úspešne.
Pridaný exemplár sa použije neskôr v príkladoch zapožičiavania.

```sql
INSERT INTO exemplars (category_id, name, description, ownership_status, current_status)
VALUES ((SELECT id FROM categories WHERE name = 'Socha'), 'POZICANIE', 'Popis', 'OWNED', 'IN_STORAGE');
```

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_STORAGE
4	4	2	POZICANIE	Popis	OWNED	IN_STORAGE

4.3 Zmena zóny exempláru (neopísané v zadaní 4)

4.3.1 Funkcie a triggery

```sql
-- Presunutie exemplára do inej zóny
CREATE OR REPLACE FUNCTION move_exemplar_to_zone() RETURNS TRIGGER AS $$
DECLARE
exemplar_status current_status_enum;
zone_exposition INTEGER;
BEGIN
SELECT current_status INTO exemplar_status FROM exemplars WHERE id = NEW.exemplar_id;
SELECT exposition_id INTO zone_exposition FROM exposition_zones WHERE zone_id = NEW.zone_id;

IF zone_exposition IS NULL OR zone_exposition <>
(SELECT exposition_id FROM exemplar_expositions WHERE exemplar_id = NEW.exemplar_id) THEN
RAISE EXCEPTION 'Zóna nepatrí do expozície, v ktorej je exemplár.';
-- Ak zóna nepatrí do expozície kde je daný exemplár, vypíše sa chyba
ELSE
RETURN NEW;
END IF;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER exemplar_movement BEFORE UPDATE ON exemplars_zones
FOR EACH ROW EXECUTE PROCEDURE move_exemplar_to_zone();
```

4.3.2 Príklad použitia

Použijeme rovnaké expozície ako v ukážke vkladania expozícií a pridáme Zónu 3 do expozície IN PROGRESS EXPO

```
SELECT create_exposition('PLANNED EXPO'::text, ARRAY['Maľba 1'],  ARRAY['Zóna 1'],
 '2025-01-01'::timestamp, '2025-01-02'::timestamp);
SELECT create_exposition('IN PROGRESS EXPO'::text, ARRAY['Fotografia 1'],
 ARRAY['Zóna 2'], '2023-01-01'::timestamp, '2024-10-10'::timestamp);

DO $$
DECLARE
expo_id INTEGER;
BEGIN
SELECT id INTO expo_id FROM expositions WHERE name = 'IN PROGRESS EXPO';
INSERT INTO exposition_zones (exposition_id, zone_id)
VALUES (expo_id, (SELECT id FROM zones WHERE name = 'Zóna 3'));
END $$;
```

Najskôr sa pokúsime presunúť exemplár do zóny, ktorú expozícia, obsahujúca daný exemplár, nevlastní. Skúsime presunúť exemplár zo zóny expozície PLANNED EXPO do zóny expozície IN PROGRESS EXPO. To vypíše chybu.

```
UPDATE exemplars_zones SET zone_id = 3 WHERE exemplar_id IN (SELECT id FROM exemplars WHERE name = 'Maľba 1');
```

```
ERROR:  Zóna nepatrí do expozície, v ktorej je exemplár.
CONTEXT:  PL/pgSQL function move_exemplar_to_zone() line 9 at RAISE

SQL state: P0001
```

Potom sa pokúsime presunúť exemplár v rámci zón expozície.

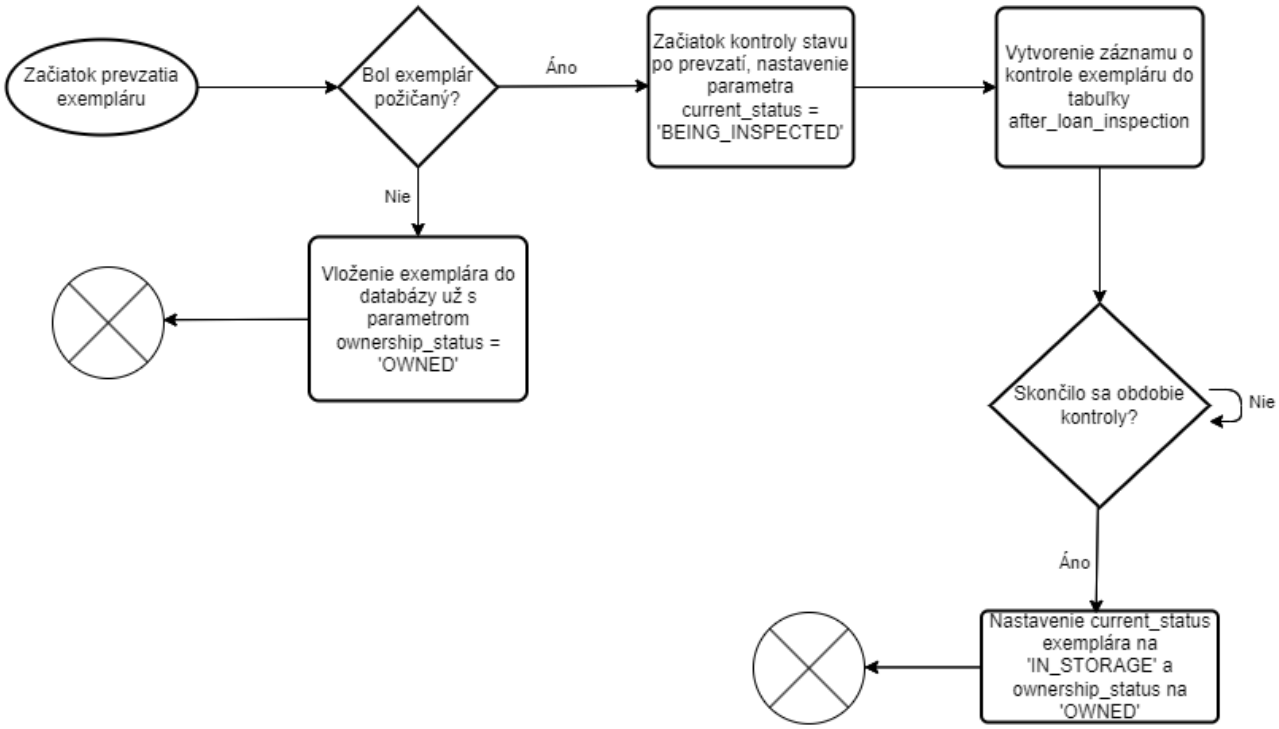
```
UPDATE exemplars_zones SET zone_id = 3 WHERE exemplar_id IN (SELECT id FROM exemplars WHERE name = 'Fotografia 1');
```

Spojovacia tabuľka exemplars_zones pred a po zmene zóny:

	exemplar_id integer	zone_id integer
1	1	1
2	3	2

	exemplar_id integer	zone_id integer
1	1	1
2	3	3

4.4 Prevzatie exemplára z inej inštitúcie podľa zadania 4



Najprv sa zistí či sa údaj o exemplári nachádza v tabuľke loans a teda či bol inštitúcii zapožičaný alebo nie. Ak nie tak sa jednoducho vytvorí záznam o exemplári v databáze s parametrom ownership_status = 'OWNED'. Ak áno nastaví sa exemplár current_status na 'BEING_INSPECTED' až dovtedy dokiaľ sa nedokončí kontrola. Zároveň sa o kontrole vytvorí záznam v tabuľke after_loan_inspection. Ak sa kontrola skončí, nastaví sa exemplár current_status na 'IN_STORAGE' a ownership_status na 'OWNED'.

4.4.1 Funkcie a triggery

```
1  CREATE OR REPLACE FUNCTION receive_exemplar() RETURNS TRIGGER AS $$
2  DECLARE
3  loan_record INTEGER;
4  loan_type TEXT;
5  BEGIN
6  SELECT COUNT(*) INTO loan_record FROM loans WHERE exemplar_id = NEW.id AND type = 'LOANED_TO';
7  SELECT type INTO loan_type FROM loans WHERE exemplar_id = NEW.id;
8  IF (loan_record = 0 AND loan_type <> 'LOANED_IN') OR (OLD.ownership_status = 'LOANED' AND NEW.ownership_status <> 'LOANED') THEN
9  -- Exemplár nebol zapožičaný, takže sa vytvorí záznam s ownership_status = 'OWNED'
10 NEW.ownership_status := 'OWNED';
11 ELSIF loan_type <> 'LOANED_IN' THEN
12 NEW.ownership_status := 'LOANED';
13 END IF;
14
15
16
17
18
19
20
21
```

```sql
-- Vytvorenie záznamu o inšpekcii
INSERT INTO after_loan_inspection (loan_id, inspection_date, inspection_end_date, inspection_description)
VALUES ((SELECT id FROM loans WHERE exemplar_id = NEW.id), NOW(), NOW() + INTERVAL '5 days', 'FINE');
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER exemplar_reception BEFORE UPDATE ON exemplars
FOR EACH ROW WHEN (OLD.ownership_status = 'LOANED') EXECUTE PROCEDURE receive_exemplar();


-- Požičanie exemplára inštitúcii
CREATE OR REPLACE FUNCTION loan_to(exemplar_name TEXT, institution_name TEXT, start_date TIMESTAMP,
end_date TIMESTAMP, available TIMESTAMP) RETURNS VOID AS $$
DECLARE own_status ownership_status_enum;
BEGIN
SELECT ownership_status INTO own_status FROM exemplars WHERE name = exemplar_name;
IF own_status <> 'OWNED' THEN
RAISE EXCEPTION 'Nie je možné zapožičať nevlastnený alebo momentálne zapožičaný exemplár.';
END IF;
UPDATE exemplars
SET ownership_status = 'LOANED', current_status = 'IN_TRANSIT'
WHERE name = exemplar_name;

INSERT INTO loans (exemplar_id, type, involved_institution_id, loan_start_date, loan_end_date, expected_exemplar_availability)
VALUES (
(SELECT id FROM exemplars WHERE name = exemplar_name),
'LOANED_TO',
(SELECT id FROM institutions WHERE name = institution_name),
start_date,
end_date,
available
);
END;
$$ LANGUAGE plpgsql;


-- Prevzatie požičaného exemplára od inštitúcie
CREATE OR REPLACE FUNCTION loan_to_returned(exemplar_name TEXT) RETURNS VOID AS $$
BEGIN
UPDATE exemplars
SET ownership_status = 'OWNED', current_status = 'BEING_INSPECTED'
WHERE name = exemplar_name;
END;
$$ LANGUAGE plpgsql;

-- Manuálne ukončenie kontroly pre ukážku
CREATE OR REPLACE FUNCTION end_inspection(exemplar_name TEXT) RETURNS VOID AS $$
BEGIN
UPDATE exemplars
SET ownership_status = 'OWNED', current_status = 'IN_STORAGE'
WHERE name = exemplar_name;
END;
$$ LANGUAGE plpgsql;
```

4.4.2 Príklad použitia

Ak exemplár nebol požičaný ale jednalo sa o ”dar”, jednoducho sa vytvorí nový záznam v tabuľke exemplars. Ak ale bol exemplár zapožičaný, proces prebehne inak. Ukážka zapožičania exemplára inej inštitúcii:

```
SELECT loan_to('POZICANIE'::text, 'Inštitúcia 1'::text, NOW()::TIMESTAMP,
 NOW()::TIMESTAMP + INTERVAL '5 days' , NOW()::TIMESTAMP + INTERVAL '5 days');
```

Stav exemplára POZICANIE pred zapožičaním:

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_EXPO
4	4	2	POZICANIE	Popis	OWNED	IN_STORAGE

Vytvorenie nového záznamu v tabuľke loans:

	id [PK] integer	exemplar_id integer	type loan_type_enum	involved_institution_id integer	loan_start_date timestamp without time zone	loan_end_date timestamp without time zone	expected_exemplar_availability timestamp without time zone
1	1	4	LOANED_TO	1	2024-04-22 10:02:21.367749	2024-04-27 10:02:21.367749	2024-04-27 10:02:21.367749

Stav exemplára POZICANIE po zapožičaní:

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_EXPO
4	4	2	POZICANIE	Popis	LOANED	IN_TRANSIT

Ak by sa chcel zapožičať exemplár ktorý nie je vlastnený alebo je momentálne zapožičaný, vypíše sa error:

```
ERROR:  Nie je možné zapožičať nevlastnený alebo momentálne zapožičaný exemplár.
CONTEXT:  PL/pgSQL function loan_to(text,text,timestamp without time zone,timestamp without time zone,timestamp without time zone) line 6 at RAISE

SQL state: P0001
```

Po ukončení zapožičania inej inštitúcii sa exemplár prevezme späť.

```
SELECT loan_to_returned('POZICANIE'::text);
```

Stav exemplára POZICANIE po prevzatí:

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_EXPO
4	4	2	POZICANIE	Popis	OWNED	BEING_INSPECTED

Vytvorenie nového záznamu v tabuľke after_loan_inspection:

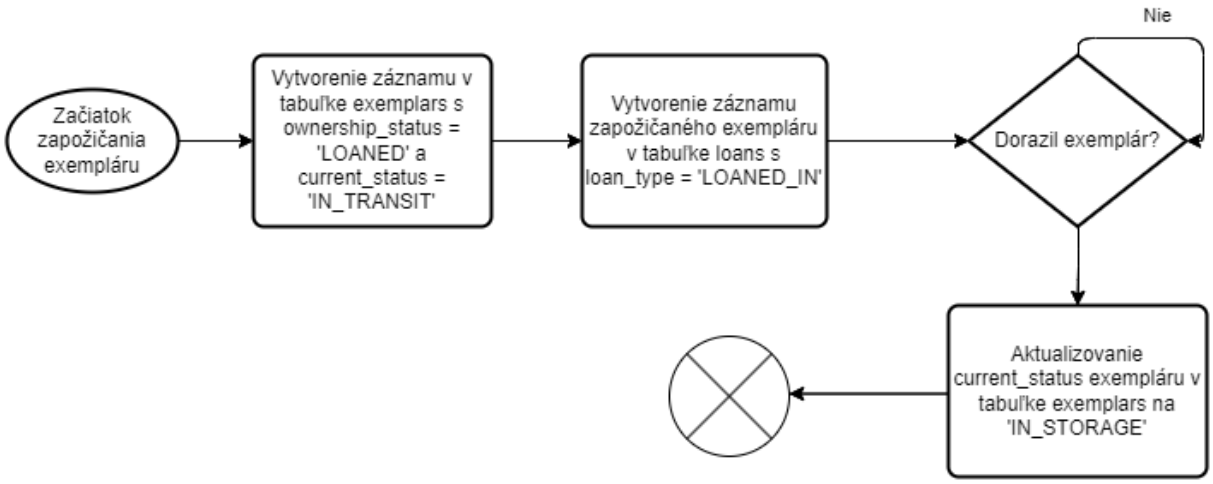
	id [PK] integer	loan_id integer	inspection_date timestamp without time zone	inspection_end_date timestamp without time zone	inspection_description text
1	1	1	2024-04-22 10:05:03.936629	2024-04-27 10:05:03.936629	FINE

Stav exemplára POZICANIE po ukončení kontroly (manuálne ukončenie pre ukážku):

```
SELECT end_inspection('POZICANIE'::text);
```

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_EXPO
4	4	2	POZICANIE	Popis	OWNED	IN_STORAGE

4.5 Zapožičanie exemplára z inej inštitúcie podľa zadania 4



Najprv sa vytvorí nový záznam v tabuľke exemplars. Ownership_status sa nastaví na 'LOANED' a current_status na 'IN_TRANSIT'. Taktiež sa o zapožičaní vytvorí záznam v tabuľke loans typu 'LOANED_IN'. Potom čo dorazí exemplár sa mu nastaví current_state na 'IN_STORAGE'.

4.5.1 Funkcie a triggery

```sql
CREATE OR REPLACE FUNCTION loan_from(
in_category_name TEXT,
in_exemplar_name TEXT,
in_description TEXT,
in_involved_institution_name TEXT,
in_loan_start_date TIMESTAMP,
in_loan_end_date TIMESTAMP,
in_expected_availability TIMESTAMP
) RETURNS VOID AS $$
DECLARE
v_category_id INTEGER;
v_involved_institution_id INTEGER;
v_exemplar_id INTEGER;
BEGIN
-- Vloženie nového exemplára
INSERT INTO exemplars (category_id, name, description, ownership_status, current_status)
VALUES ((SELECT id FROM categories WHERE name = in_category_name),
in_exemplar_name,
in_description,
'LOANED',
'IN_TRANSIT');

-- Získanie ID vloženého exemplára
SELECT id INTO v_exemplar_id FROM exemplars WHERE name = in_exemplar_name;

-- Vloženie nového záznamu o pôžičke
INSERT INTO loans (exemplar_id, type, involved_institution_id, loan_start_date, loan_end_date, expected_exemplar_availability)
VALUES (v_exemplar_id,
'LOANED_TO',
(SELECT id FROM institutions WHERE name = in_involved_institution_name),
in_loan_start_date,
in_loan_end_date,
in_expected_availability);
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION loan_arrived(exemplar_name TEXT) RETURNS VOID AS $$
BEGIN
UPDATE exemplars
SET ownership_status = 'LOANED', current_status = 'IN_STORAGE'
WHERE name = exemplar_name;
END;
$$ LANGUAGE plpgsql;
```

4.5.2 Príklad použitia

```
SELECT loan_from(
'Socha'::text,
'ZAPOZICANE'::text,
'Popis'::text,
'Inštitúcia 1'::text,
NOW()::timestamp,
NOW()::timestamp + INTERVAL '5 days',
NOW()::timestamp
);
```

Pri vytváraní pôžičky od inej inštitúcie sa najskôr vytvorí záznam v tabuľke exemplars:

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_EXPO
4	4	2	POZICANIE	Popis	OWNED	IN_STORAGE
5	5	2	ZAPOZICANE	Popis	LOANED	IN_TRANSIT

Taktiež sa vytvorí záznam v tabuľke loans:

	id [PK] integer	exemplar_id integer	type loan_type_enum	involved_institution_id integer	loan_start_date timestamp without time zone	loan_end_date timestamp without time zone	expected_exemplar_availability timestamp without time zone
1	1	4	LOANED_TO	1	2024-04-22 10:18:34.376951	2024-04-27 10:18:34.376951	2024-04-27 10:18:34.376951
2	2	5	LOANED_IN	1	2024-04-22 10:18:34.376951	2024-04-27 10:18:34.376951	2024-04-22 10:18:34.376951

Po dorazení sa exemplár presunie do skladu a je pripravený na použitie:

```
SELECT loan_arrived('ZAPOZICANE'::text);
```

	id [PK] integer	category_id integer	name text	description text	ownership_status ownership_status_enum	current_status current_status_enum
1	2	2	Socha 1	Popis sochy 1	OWNED	IN_STORAGE
2	1	1	Maľba 1	Popis maľby 1	OWNED	IN_EXPO
3	3	3	Fotografia 1	Popis fotografie 1	OWNED	IN_EXPO
4	4	2	POZICANIE	Popis	OWNED	IN_STORAGE
5	5	2	ZAPOZICANE	Popis	LOANED	IN_STORAGE

5 Poznámka ku demoštrácii príkladov

Každá ukážka je spravená nasledovným spôsobom:
1: Ak je potrebné, SELECTni tabuľku pred úpravou

```
                SELECT * FROM meno_tabulky
```

2: Vykonaj príslušnú akciu
3: Zobraz zmeny po akcii v potrebných tabuľkách

```
                SELECT * FROM meno_tabulky
```