BAZA KPOPu

Paweł Pasternak, Tyberiusz Bobrek

Opis pracy

Nad projektem pracowaliśmy systematycznie, spotykając się po zajęciach i wspólnie implementując kolejne elementy bazy (role, tabele, typy, sekwencje, funkcje, procedury, triggery, widoki). Do zarządzania wykorzystaliśmy narzędzie **pgAdmin4**, które pozwoliło na szybszą pracę i lepszą wizualizację struktury danych niż standardowy interfejs CLI.

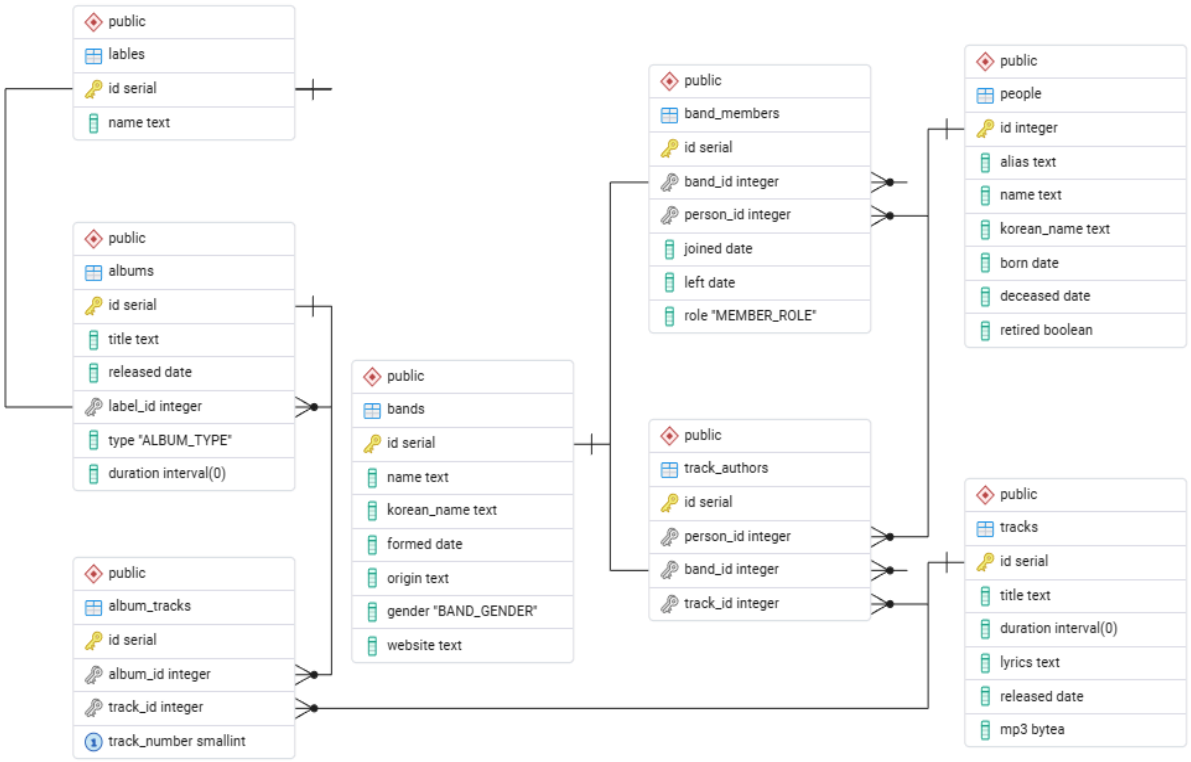
Fragmenty z kryteriów oceny:

- Schemat bazy danych - w sekcji schemat.
- Warunki integralności - w sekcjach: tabele, typy i triggery.
- Procedury, funkcje - w sekcjach: funkcje, procedury i triggery.
- Zasady bezpieczeństwa - w sekcjach: role i procedury.

Dokumentacja Techniczna Bazy Danych

Ten dokument opisuje strukturę i logikę zaimplementowaną w plikach *base.sql* i *users.sql*. Baza danych ma na celu zagregowanie danych dotyczących artystów, zespołów, albumów i utworów kpopowych. Użytkownicy w zależności od roli mogą jedynie przeglądać rekordy lub również je dodawać, zmieniać lub usuwać. Mają również dostęp do kilku funkcji pozwalających na szybsze dodawanie wpisów do bazy danych, generowania nazw zespołów, czy zliczania czasu trwania całych albumów. Dane które są wprowadzane muszą spełniać odpowiednie wymagania.

Schemat relacji w bazie danych



Role

Bezpieczeństwo danych oparliśmy na modelu kontroli dostępu opartej na rolach (RBAC - Role-Based Access Control). Zdefiniowaliśmy trzy główne role użytkowników o zróżnicowanych uprawnieniach, a także roli twórcy bazy danych, która ma pełne możliwości do wprowadzenia zmian w bazie danych.

guest

Opis: rola mająca jedynie uprawnienia do odczytu rekordów w bazie danych.

Definicja:

```
CREATE USER guest;

GRANT SELECT ON TABLE public.album_tracks TO guest;
GRANT SELECT ON TABLE public.albums TO guest;
GRANT SELECT ON TABLE public.band_members TO guest;
GRANT SELECT ON TABLE public.bands TO guest;
GRANT SELECT ON TABLE public.people TO guest;
GRANT SELECT ON TABLE public.track_authors TO guest;
GRANT SELECT ON TABLE public.tracks TO guest;
GRANT SELECT ON TABLE public.artist_band_tracks_while_member TO
guest;
GRANT SELECT ON TABLE public.band_with_albums_and_tracks TO guest;
GRANT SELECT ON TABLE public.lables TO guest;
GRANT SELECT ON TABLE public.tracks_in_album TO guest;
GRANT SELECT ON TABLE public.tracks_with_album_and_artists TO
guest;

-- lub

GRANT SELECT ON ALL TABLES IN SCHEMA public TO guest;
```

contributor

Opis: Rola mająca jedynie uprawnienia do wykorzystania procedur zdefiniowanych w bazie danych. Umożliwia to danemu użytkownikowi na dodawanie danych, w sposób kontrolowany poprzez to, jak zostały zdefiniowane procedury.

Definicja:

```
CREATE USER contributor;
GRANT USAGE ON SCHEMA public TO contributor;

GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public TO contributor;

-- lub

GRANT ALL ON PROCEDURE public.add_album(IN title text, IN released
date, IN label_id integer, IN type public."ALBUM_TYPE", IN
duration interval) TO contributor;
GRANT ALL ON PROCEDURE public.add_album_track(IN album_id integer,
IN track_id integer, IN track_number smallint) TO contributor;
GRANT ALL ON PROCEDURE public.add_band(IN name text, IN gender
public."BAND_GENDER", IN korean_name text, IN formed date, IN
origin text, IN website text) TO contributor;
GRANT ALL ON PROCEDURE public.add_band_member(IN band_id integer,
IN person_id integer, IN joined date, IN left_date date, IN role
public."MEMBER_ROLE") TO contributor;
GRANT ALL ON PROCEDURE public.add_label(IN name text) TO
contributor;
GRANT ALL ON PROCEDURE public.add_person(IN name text, IN alias
text, IN korean_name text, IN born date, IN deceased date, IN
retired boolean) TO contributor;
GRANT ALL ON PROCEDURE public.add_track(IN title text, IN duration
interval, IN released date, IN lyrics text) TO contributor;
GRANT ALL ON PROCEDURE public.add_track_author(IN track_id
integer, IN person_id integer, IN band_id integer) TO contributor;
```

editor

Opis: Rola mająca uprawnienia do odczytu, dodawania, zmieniania i usuwaniu rekordów w bazie danych.

Definicja:

```
CREATE USER editor;

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.album_tracks TO
editor;
GRANT USAGE ON SEQUENCE public.album_tracks_id_seq TO editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.albums TO
editor;
GRANT USAGE ON SEQUENCE public.albums_id_seq TO editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.band_members TO
editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.bands TO editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.people TO
editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.track_authors TO
editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.tracks TO
editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE
    public.artist_band_tracks_while_member TO editor;
GRANT USAGE ON SEQUENCE public.band_members_band_id_seq TO editor;
GRANT USAGE ON SEQUENCE public.band_members_id_seq TO editor;
GRANT USAGE ON SEQUENCE public.band_members_person_id_seq TO
editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE
    public.band_with_albums_and_tracks TO editor;
GRANT USAGE ON SEQUENCE public.bands_id_seq TO editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.lables TO
editor;
GRANT USAGE ON SEQUENCE public.lables_id_seq TO editor;
GRANT USAGE ON SEQUENCE public.members_id_seq TO editor;
GRANT USAGE ON SEQUENCE public.track_authors_id_seq TO editor;
GRANT USAGE ON SEQUENCE public.tracks_id_seq TO editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public.tracks_in_album
TO editor;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE
    public.tracks_with_album_and_artists TO editor;
```

```
-- lub

GRANT SELECT ON ALL TABLES IN SCHEMA public TO editor;
GRANT INSERT ON ALL TABLES IN SCHEMA public TO editor;
GRANT UPDATE ON ALL TABLES IN SCHEMA public TO editor;
GRANT DELETE ON ALL TABLES IN SCHEMA public TO editor;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA public TO editor;
```

owner

Opis: Rola posiadająca całkowity i absolutny dostęp do bazy danych.

Definiowana automatycznie.

Typy

Dla optymalizacji spójności danych wykorzystaliśmy własne typy.

ALBUM_TYPE

Opis: Określa rodzaj albumu.

Definicja:

```
CREATE TYPE public."ALBUM_TYPE" AS ENUM (
    'Studio',
    'EP',
    'Soundtrack',
    'Live',
    'Compilation',
    'Reissue',
    'Single'
);
```

BAND_GENDER

Opis: Określa rodzaj zespołu.

Definicja:

```
CREATE TYPE public."BAND_GENDER" AS ENUM (
    'boys',
    'girls',
    'mixed'
);
```

MEMBER_ROLE

Opis: Określa funkcję pełnioną przez członka zespołu.

Definicja:

```
CREATE TYPE public."MEMBER_ROLE" AS ENUM (
    'leader',
    'vocalist',
    'dancer',
    'rapper',
    'maknae',
    'visual',
    'center'
);
```

Tabele

Zastosowaliśmy trzy poziomy zabezpieczeń integralności danych:

- W samej strukturze: Wymuszamy relacje kluczami obcymi (np. kaskadowe usuwanie utworów po usunięciu albumu), blokujemy puste pola (NOT NULL) i używamy własnych typów lub typów jak interval, które są przeznaczone do konkretnych celów.
- Sprawdzenia (CHECK): Proste reguły walidacji, np. data dołączenia do zespołu musi być wcześniejsza niż data odejścia, a numer utworu musi być dodatni.
- Automatyzacja (Triggery): Zaawansowane reguły, które automatycznie obliczają czas trwania albumu i blokują dodanie daty wydania wcześniejszej niż data urodzenia artysty lub powstania zespołu.

album_tracks

Opis: Przechowuje informacje o tym jakie utwory należą do jakiego albumu i w jakiej są kolejności w tym albumie.

Definicja:

```
CREATE TABLE public.album_tracks (
    id integer NOT NULL,
    album_id integer NOT NULL,
    track_id integer NOT NULL,
    track_number smallint NOT NULL
);
```

Constraints:

```
ALTER TABLE ONLY public.album_tracks
    ADD CONSTRAINT album_tracks_pkey PRIMARY KEY (id);
ALTER TABLE ONLY public.album_tracks
    ADD CONSTRAINT track_number_positive CHECK ((track_number >
0));
ALTER TABLE ONLY public.album_tracks
    ADD CONSTRAINT track_num_in_album_uniq UNIQUE (track_number,
album_id);
CREATE TRIGGER set_album_duration AFTER INSERT OR DELETE OR UPDATE
ON public.album_tracks FOR EACH ROW EXECUTE FUNCTION
public.album_duration();
ALTER TABLE ONLY public.album_tracks
    ADD CONSTRAINT album_tracks_album_id_fkey FOREIGN KEY
(album_id) REFERENCES public.albums(id) ON DELETE CASCADE NOT
```

```
VALID;
ALTER TABLE ONLY public.album_tracks
    ADD CONSTRAINT album_tracks_track_id_fkey1 FOREIGN KEY
(track_id) REFERENCES public.tracks(id) ON DELETE CASCADE NOT
VALID;
```

albums

Opis: Przechowuje informacje dotyczące poszczególnych albumów takie jak ich tytuł, czy data wydania.

Definicja:

```
CREATE TABLE public.albums (
    id integer NOT NULL,
    title text NOT NULL,
    released date NOT NULL,
    label_id integer NOT NULL,
    type public."ALBUM_TYPE" NOT NULL,
    duration interval(0)
);
```

Constraints:

```
ALTER TABLE ONLY public.albums
    ADD CONSTRAINT albums_pkey PRIMARY KEY (id);
ALTER TABLE public.albums
    ADD CONSTRAINT positive_intervals CHECK ((duration >=
'00:00:00'::interval)) NOT VALID;
ALTER TABLE public.albums
    ADD CONSTRAINT title_not_empty CHECK ((btrim(title) <>
''::text)) NOT VALID;
CREATE CONSTRAINT TRIGGER album_release_date_constraint AFTER
INSERT OR UPDATE ON public.albums DEFERRABLE INITIALLY DEFERRED
FOR EACH ROW EXECUTE FUNCTION public.enforce_album_release_date();
ALTER TABLE ONLY public.albums
    ADD CONSTRAINT albums_label_id_fkey FOREIGN KEY (label_id)
REFERENCES public.lables(id);
```

band_members

Opis: Przechowuje informacje o tym którzy artyści należą do jakich zespołów, w jakim czasie i jaką w nim pełnią rolę. Artyści często zmieniają zespoły, dlatego tak ważna jest informacja o tym w jakim czasie są w jakim zespole.

Definicja:

```
CREATE TABLE public.band_members (
    id integer NOT NULL,
    band_id integer NOT NULL,
    person_id integer NOT NULL,
    joined date NOT NULL,
    "left" date,
    role public."MEMBER_ROLE"
);
```

Constraints:

```
ALTER TABLE public.band_members
    ADD CONSTRAINT band_members_check CHECK ((joined < "left"))
NOT VALID;
ALTER TABLE ONLY public.band_members
    ADD CONSTRAINT band_members_pkey PRIMARY KEY (id);
ALTER TABLE ONLY public.band_members
    ADD CONSTRAINT band_members_band_id_fkey FOREIGN KEY (band_id)
REFERENCES public.bands(id);
ALTER TABLE ONLY public.band_members
    ADD CONSTRAINT band_members_person_id_fkey FOREIGN KEY
(person_id) REFERENCES public.people(id);
```

bands

Opis: Przechowuje informacje na temat poszczególnych zespołów takie jak ich nazwa, data założenia, czy jest to *boys*-, czy *girls*- band.

Definicja:

```
CREATE TABLE public.bands (
    id integer NOT NULL,
    name text NOT NULL,
    korean_name text,
    formed date NOT NULL,
    origin text,
    gender public."BAND_GENDER" NOT NULL,
    website text
);
```

Constraints:

```
ALTER TABLE ONLY public.bands
    ADD CONSTRAINT bands_pkey PRIMARY KEY (id);
```

people

Opis: Przechowuje informacje na temat poszczególnych artystów takie jak ich synonim, prawdziwe imię, data urodzenia, data śmierci i to czy zrezygnowali z kontynuowania kariery muzycznej.

Definicja:

```
CREATE TABLE public.people (
    id integer CONSTRAINT members_id_not_null NOT NULL,
    alias text,
    name text NOT NULL,
    korean_name text,
    born date,
    deceased date,
    retired boolean DEFAULT false NOT NULL
);
```

Constraints:

```
ALTER TABLE ONLY public.people
    ADD CONSTRAINT members_pkey PRIMARY KEY (id);
```

track_authors

Opis: Przechowuje informacje na temat piosenek danych artystów lub zespołów.

Definicja:

```
CREATE TABLE public.track_authors (
    id integer NOT NULL,
    person_id integer,
    band_id integer,
    track_id integer NOT NULL
);
```

Constraints:

```
ALTER TABLE ONLY public.track_authors
    ADD CONSTRAINT track_authors_pkey PRIMARY KEY (id);
ALTER TABLE ONLY public.track_authors
    ADD CONSTRAINT track_authors_band_id_fkey FOREIGN KEY
(band_id) REFERENCES public.bands(id);
ALTER TABLE ONLY public.track_authors
    ADD CONSTRAINT track_authors_person_id_fkey FOREIGN KEY
(person_id) REFERENCES public.people(id);
ALTER TABLE ONLY public.track_authors
    ADD CONSTRAINT track_authors_person_id_fkey FOREIGN KEY
(person_id) REFERENCES public.people(id);
```

tracks

Opis: Przechowuje informacje na temat poszczególnych piosenek takich jak tytuł, ich długość, tekst, data wydania. Pozwala również na przechowywanie plików mp3 w samej bazie danych.

Definicja:

```
CREATE TABLE public.tracks (
    id integer NOT NULL,
    title text NOT NULL,
    duration interval(0) CONSTRAINT tracks_length_not_null NOT
NULL,
    lyrics text,
    released date NOT NULL,
    mp3 bytea
);
```

Constraints:

```
ALTER TABLE ONLY public.tracks
    ADD CONSTRAINT tracks_pkey PRIMARY KEY (id);
CREATE CONSTRAINT TRIGGER track_release_date_constraint AFTER
INSERT OR UPDATE OF released ON public.tracks DEFERRABLE INITIALLY
DEFERRED FOR EACH ROW EXECUTE FUNCTION
public.enforce_track_release_date();
```

lables

Opis: Przechowuje informacje na temat wydawnictw muzycznych.

Definicja:

```
CREATE TABLE public.lables (
    id integer NOT NULL,
    name text NOT NULL
);
```

Constraints:

```
ALTER TABLE ONLY public.lables
    ADD CONSTRAINT lables_pkey PRIMARY KEY (id);
```

Funkcje

Dla poprawy jakości użytkowania i zabawy.

generate_band_name

Opis: Tworzy ciąg znaków składający się z jednego lub dwóch słów (zaczynających się wielką literą) oraz losowej cyfry na końcu. Imituje to proces tworzenia niektórych nazw zespołów

Definicja:.

```
CREATE FUNCTION public.generate_band_name(words text[] DEFAULT
NULL::text[]) RETURNS text
    LANGUAGE plpgsql
    AS $$
DECLARE
    word_count integer;
    word1 text;
    word2 text;
    use_two boolean := random() < 0.5;
    rand_num integer := floor(random() * 10);
    default_words text[] := ARRAY['star', 'dream', 'moon',
'shine', 'pulse'];
BEGIN
    IF words IS NULL OR array_length(words, 1) = 0 THEN
        words := default_words;
    END IF;

    word_count := array_length(words, 1);
    word1 := words[floor(random() * word_count + 1)];

    IF use_two AND word_count > 1 THEN
        LOOP
            word2 := words[floor(random() * word_count + 1)];
            EXIT WHEN word2 <> word1;
        END LOOP;
        RETURN initcap(word1) || initcap(word2) || rand_num;
    ELSE
        RETURN initcap(word1) || rand_num;
    END IF;
END;$$;
```

count_artist_songs

Opis: Funkcja która zlicza wszystkie utwory danego artysty niezależnie czy w zespole, czy indywidualnie.

Definicja:

```
CREATE FUNCTION public.count_artist_songs(p_artist_name text)
RETURNS integer
    LANGUAGE plpgsql
    AS $$
DECLARE
    v_person_id integer;
    v_count integer;
BEGIN
    SELECT id INTO v_person_id
    FROM public.people
    WHERE name = p_artist_name OR alias = p_artist_name
    LIMIT 1;

    IF v_person_id IS NULL THEN
        RETURN 0;
    END IF;

    SELECT COUNT(DISTINCT t.id) INTO v_count
    FROM public.tracks t
    WHERE
        EXISTS (
            SELECT 1 FROM public.track_authors ta
            WHERE ta.track_id = t.id AND ta.person_id =
v_person_id
        )
        OR
        EXISTS (
            SELECT 1
            FROM public.track_authors ta
            JOIN public.band_members bm ON ta.band_id = bm.band_id
            WHERE ta.track_id = t.id
              AND bm.person_id = v_person_id
              AND t.released >= bm.joined
              AND (bm."left" IS NULL OR t.released <= bm."left")
        );
```

```
    RETURN v_count;
END;$$;
```

Procedury

Procedury te wykorzystują klauzulę `SECURITY DEFINER`, co oznacza, że wykonują się z uprawnieniami twórcy procedury, a nie wywołującego ją użytkownika. Dzięki temu użytkownik *contributor* może wprowadzać dane jedynie w sposób przez nas zdefiniowany.

add_album

Opis: Procedura do dodawania albumów.

Definicja:

```
CREATE PROCEDURE public.add_album(IN title text, IN released date,
IN label_id integer, IN type public."ALBUM_TYPE", IN duration
interval DEFAULT NULL::interval)
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO public.albums (title, released, label_id,
type, duration)
    VALUES (title, released, label_id, type, duration);$$;
```

add_band

Opis: Procedura do dodawania zespołów.

Definicja:

```
CREATE PROCEDURE public.add_band(IN name text, IN gender
public."BAND_GENDER", IN korean_name text DEFAULT NULL::text, IN
formed date DEFAULT CURRENT_DATE, IN origin text DEFAULT
NULL::text, IN website text DEFAULT NULL::text)
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO public.bands (name, korean_name, formed,
origin, gender, website)
    VALUES (name, korean_name, formed, origin, gender,
website);$$;
```

add_band_member

Opis: Procedura do dodawania członków zespołów.

Definicja:

```
CREATE PROCEDURE public.add_band_member(IN band_id integer, IN
person_id integer, IN joined date, IN left_date date DEFAULT
NULL::date, IN role public."MEMBER_ROLE" DEFAULT
NULL::public."MEMBER_ROLE")
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO public.band_members (band_id, person_id,
joined, "left", role)
    VALUES (band_id, person_id, joined, left_date, role);$$;
```

add_label

Opis: Procedura do dodawania wydawców/producentów.

Definicja:

```
CREATE PROCEDURE public.add_label(IN name text)
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO public.lables (name)
    VALUES (name);$$;
```

add_person

Opis: Procedura do dodawania artystów.

Definicja:

```
CREATE PROCEDURE public.add_person(IN name text, IN alias text
DEFAULT NULL::text, IN korean_name text DEFAULT NULL::text, IN
born date DEFAULT NULL::date, IN deceased date DEFAULT NULL::date,
IN retired boolean DEFAULT false)
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO public.people (name, alias, korean_name,
born, deceased, retired)
    VALUES (name, alias, korean_name, born, deceased, retired);$$;
```

add_album_track

Opis: Procedura do dodawania utworów do albumów.

Definicja:

```
CREATE PROCEDURE public.add_album_track(IN album_id integer, IN
track_id integer, IN track_number smallint)
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO public.album_tracks (album_id, track_id,
track_number)
    VALUES (album_id, track_id, track_number);$$;
```

add_track

Opis: Procedura do dodawania utworów.

Definicja:

```
CREATE PROCEDURE public.add_track(IN title text, IN duration
interval, IN released date DEFAULT CURRENT_DATE, IN lyrics text
DEFAULT NULL::text)
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO tracks (title, duration, lyrics, released)
VALUES (title, duration, lyrics, released);$$;
```

add_track_author

Opis: Procedura do dodawania autora do utworu.

Definicja:

```
CREATE PROCEDURE public.add_track_author(IN track_id integer, IN
person_id integer DEFAULT NULL::integer, IN band_id integer
DEFAULT NULL::integer)
    LANGUAGE sql SECURITY DEFINER
    SET search_path TO 'public', 'pg_temp'
    AS $$INSERT INTO public.track_authors (person_id, band_id,
track_id)
    VALUES (person_id, band_id, track_id);$$;
```

Triggery

Do automatyzacji niektórych operacji i zapewnienia integralności danych wprowadziliśmy kilka triggerów.

album_duration

Opis: Na bieżąco oblicza długość trwania całego albumu na podstawie należących do niego ścieżek.

Definicja:

```
CREATE FUNCTION public.album_duration() RETURNS trigger
    LANGUAGE plpgsql
    AS $$DECLARE
    affected_album integer;
BEGIN
     IF TG_OP = 'DELETE' THEN
        affected_album := OLD.album_id;
    ELSE
        affected_album := NEW.album_id;
    END IF;

    UPDATE public.albums
    SET duration = (
        SELECT SUM(t.duration)
        FROM public.album_tracks at
        JOIN public.tracks t ON at.track_id = t.id
        WHERE at.album_id = affected_album
    )
    WHERE id = affected_album;
    RETURN NULL;
END;$$;
```

enforce_album_release_date

Opis: Zapewnia poprawność daty wydania albumu przy próbie jej dodania bądź zmiany (data wydania powinna być później względem daty utworzenia zespołu oraz dat urodzeń jego członków).

Definicja:

```
CREATE FUNCTION public.enforce_album_release_date() RETURNS
trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    IF EXISTS (
        SELECT 1
           FROM public.album_tracks at
        JOIN public.track_authors ta ON ta.track_id = at.track_id
        JOIN public.bands b ON b.id = ta.band_id
        WHERE at.album_id = NEW.id
          AND NEW.released < b.formed
    ) THEN
        RAISE EXCEPTION
            'Album "%" release date (%) is before its band''s
formation date',
            NEW.title, NEW.released;
    END IF;

    IF EXISTS (
        SELECT 1
        FROM track_authors ta
        JOIN people p ON p.id = ta.person_id
        WHERE ta.track_id = NEW.id
          AND NEW.released < p.born
    ) THEN
        RAISE EXCEPTION
            'Album % release date (%) is before artist birth
date',
            NEW.title, NEW.released;
    END IF;

    RETURN NEW;
END;$$;
```

enforce_track_release_date

Opis: Zapewnia poprawność daty wydania ścieżki przy próbie jej dodania bądź zmiany (data wydania powinna być później względem daty utworzenia zespołu oraz dat urodzeń jego członków).

Definicja:

```
CREATE FUNCTION public.enforce_track_release_date() RETURNS
trigger
    LANGUAGE plpgsql
    AS $$BEGIN
    -- Band formation date check
    IF EXISTS (
        SELECT 1
        FROM public.track_authors ta
        JOIN public.bands b ON b.id = ta.band_id
        WHERE ta.track_id = NEW.id
          AND NEW.released < b.formed
    ) THEN
        RAISE EXCEPTION
            'Track "%" release date (%) is before its band''s
formation date',
            NEW.title, NEW.released;
    END IF;

    -- Artist birth date check
    IF EXISTS (
        SELECT 1
        FROM public.track_authors ta
        JOIN public.people p ON p.id = ta.person_id
        WHERE ta.track_id = NEW.id
          AND NEW.released < p.born
    ) THEN
        RAISE EXCEPTION
            'Track "%" release date (%) is before its artist''s
birth date',
            NEW.title, NEW.released;
    END IF;

    RETURN NEW;
END;$$;
```

Widoki

Aby usprawnić oraz ujednolicić możliwości przeglądania bazy danych wprowadziliśmy kilka najprzydatniejszych widoków.

artist_band_tracks_while_member

Opis: Zestawienie każdego artysty z utworami, które współtworzył należąc do danego zespołu.

Definicja:

```
CREATE VIEW public.artist_band_tracks_while_member AS
 SELECT p.name AS artist_name,
    b.name AS band_name,
    t.title AS track_title,
    a.title AS album_title,
    t.released AS track_released,
    bm.joined,
    bm."left"
   FROM ((((((public.band_members bm
     JOIN public.people p ON ((p.id = bm.person_id)))
     JOIN public.bands b ON ((b.id = bm.band_id)))
     JOIN public.track_authors ta ON ((ta.band_id = b.id)))
     JOIN public.tracks t ON ((t.id = ta.track_id)))
     LEFT JOIN public.album_tracks at ON ((at.track_id = t.id)))
     LEFT JOIN public.albums a ON ((a.id = at.album_id)))
  WHERE ((t.released >= bm.joined) AND ((bm."left" IS NULL) OR
(t.released <= bm."left")));
```

band_with_albums_and_tracks

Opis: Zestawienie każdego zespołu ze wszystkimi jego albumami oraz utworami.

Definicja:

```
CREATE VIEW public.band_with_albums_and_tracks AS
 SELECT b.name AS band_name,
    a.title AS album_title,
    t.title AS track_title,
    at.track_number,
    t.duration AS track_duration
   FROM ((((public.bands b
     JOIN public.track_authors ta ON ((ta.band_id = b.id)))
     JOIN public.tracks t ON ((t.id = ta.track_id)))
     JOIN public.album_tracks at ON ((at.track_id = t.id)))
     JOIN public.albums a ON ((a.id = at.album_id)));
```

tracks_with_album_and_artist

Opis: Proste zestawienie wszystkich ścieżek z konkretnymi albumami oraz ich autorami.

Definicja:

```
CREATE VIEW public.tracks_with_album_and_artists AS
 SELECT title AS track,
    ( SELECT string_agg(a.title, ', '::text) AS string_agg
           FROM ((public.tracks t2
             LEFT JOIN public.album_tracks ats ON ((t.id =
ats.track_id)))
             LEFT JOIN public.albums a ON ((a.id = ats.album_id)))
          WHERE (t2.id = t.id)
          GROUP BY t.id) AS albums,
    ( SELECT string_agg(COALESCE(b.name, p.alias, p.name), ',
'::text) AS string_agg
           FROM (((public.tracks t2
             LEFT JOIN public.track_authors ta ON ((t.id =
ta.track_id)))
             LEFT JOIN public.bands b ON ((b.id = ta.band_id)))
             LEFT JOIN public.people p ON ((p.id = ta.person_id)))
          WHERE (t2.id = t.id)
          GROUP BY t.id) AS artists,
    duration,
    released,
    lyrics
   FROM public.tracks t
  GROUP BY id;
```