POLITECHNIKA ŁÓDZKA

Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki

Projektowanie i Administracja Baz Danych

Semestr 5

Projekt Bazy Danych

Company (CrunchBase)

Autor: Mateusz Mróz

Nr indeksu: 251190

Łódź, styczeń 2026

Spis treści

1. Podstawowe założenia projektu

1.1. Cel tworzenia bazy danych

Celem projektu jest zaprojektowanie i implementacja relacyjnej bazy danych przechowującej informacje o firmach technologicznych, pochodzące z serwisu CrunchBase. Baza danych ma umożliwiać:

- przechowywanie szczegółowych informacji o firmach (nazwa, opis, kategoria, data założenia),
- śledzenie rund finansowania i inwestycji,
- zarządzanie informacjami o osobach związanych z firmami (założyciele, pracownicy, inwestorzy),
- rejestrowanie przejęć, konkurentów i kamieni milowych w historii firm,
- przechowywanie danych o produktach, biurach i mediach firmowych.

1.2. Główne założenia

1. **Normalizacja** – baza danych spełnia trzy pierwsze postacie normalne (1NF, 2NF, 3NF).
2. **Integralność danych** – wykorzystanie kluczy głównych, obcych oraz ograniczeń CHECK, UNIQUE i DEFAULT.
3. **Bezpieczeństwo** – system ról i użytkowników z różnymi poziomami uprawnień (contained users).
4. **Skalowalność** – struktura pozwalająca na łatwe dodawanie nowych firm i danych.
5. **Zgodność z danymi źródłowymi** – nazwy kolumn zachowują oryginalne nazwy pól z dokumentów JSON.

1.3. Zakres możliwości systemu

System umożliwia:

- import danych z plików JSON przy użyciu funkcji OPENJSON,
- wstawianie i aktualizację firm przez procedurę składowaną `UpsertCompany`,
- obliczanie łącznego finansowania firmy przez funkcję skalarną `GetTotalFunding`,
- przeglądanie podsumowania firm przez widok `vw_CompanyOverview`,
- zarządzanie dostępem przez role i contained users.

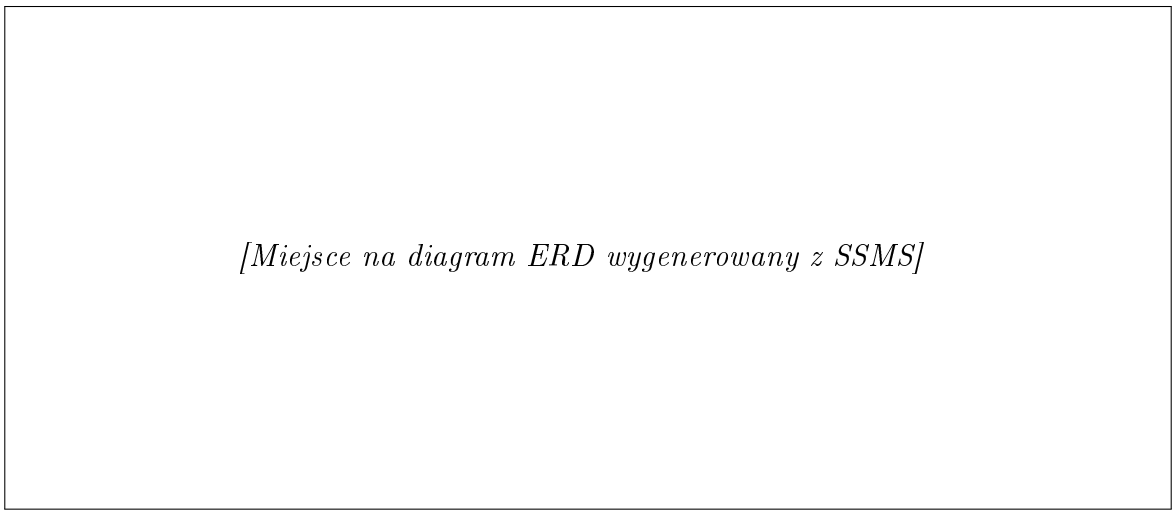
1.4. Ograniczenia przyjęte podczas projektowania

- Dane źródłowe ograniczone do 6 dokumentów JSON (6 firm: Wetpaint, Facebook, Twitter, Omnidrive, StumbleUpon, Scribd).
- Brak walidacji zewnętrznych URL-i (przechowywane jako tekst).
- Obrazy przechowywane jako ścieżki do plików, nie jako dane binarne (BLOB).
- Waluta finansowania domyślnie USD (brak automatycznego przeliczania walut).
- Baza typu `CONTAINMENT = PARTIAL` (umożliwia tworzenie contained users).

2. Schemat bazy danych

2.1. Diagram ERD

Poniżej przedstawiono diagram relacji bazy danych. Diagram można wygenerować w SQL Server Management Studio poprzez: Database Diagrams → New Database Diagram.



Rysunek 1: Diagram relacji bazy danych CompanyDB

2.2. Opis struktury

Baza danych składa się z **19 tabel** zorganizowanych w schemacie `crunchbase`. Tabele można podzielić na następujące grupy logiczne:

Tabela 1: Podział tabel na grupy logiczne

Grupa	**Tabele**
Główna	Company
Podmioty	Person, FinancialOrg
Finansowanie	FundingRound, Investment, CompanyIPO
Struktura firmy	Product, Office, CompanyRelationship
Relacje biznesowe	Competitor, Acquisition, Provider, Milestone
Media	ExternalLink, VideoEmbed, CompanyImage, Screenshot, ScreenshotSize

2.3. Główne relacje między tabelami

- **Company 1:N Product** – firma może mieć wiele produktów
- **Company 1:N Office** – firma może mieć wiele biur/siedzib
- **Company 1:N FundingRound** – firma może mieć wiele rund finansowania
- **FundingRound 1:N Investment** – runda może mieć wiele inwestycji
- **Person M:N Company** – osoba może być związana z wieloma firmami (przez CompanyRelationship)
- **Company 1:N Acquisition** – firma może przejmować wiele innych firm
- **Screenshot 1:N ScreenshotSize** – zrzut ekranu może mieć wiele rozmiarów

3. Obiekty bazy danych i ich opis

3.1. Tabele

3.1.1. Company – Tabela główna firm

Opis funkcjonalny: Przechowuje podstawowe informacje o firmach technologicznych – nazwę, opis, kategorię działalności, daty założenia i zamknięcia, dane kontaktowe oraz informacje o pozyskanym finansowaniu.

Tabela 2: Struktura tabeli Company (wybrane kolumny)

Kolumna	Typ danych	Ograniczenia
company_id	INT IDENTITY(1,1)	PRIMARY KEY
mongo_id	NVARCHAR(50)	NOT NULL, UNIQUE
name	NVARCHAR(255)	NOT NULL
permalink	NVARCHAR(255)	NOT NULL, UNIQUE
category_code	NVARCHAR(100)	NULL
number_of_employees	INT	CHECK (≥ 0)
founded_year	INT	NULL
founded_month	INT	CHECK (1-12)
founded_day	INT	CHECK (1-31)
created_at	DATETIME2	DEFAULT GETDATE()
updated_at	DATETIME2	DEFAULT GETDATE()

Klucz główny: `company_id` (auto-inkrementacja)

Ograniczenia UNIQUE: `mongo_id`, `permalink`

Ograniczenia CHECK:

- `number_of_employees >= 0`
- `founded_month BETWEEN 1 AND 12`
- `founded_day BETWEEN 1 AND 31`

Ograniczenia DEFAULT:

- `created_at = GETDATE()`
- `updated_at = GETDATE()`

3.1.2. Person – Osoby

Opis funkcjonalny: Przechowuje informacje o osobach związanych z firmami – założycielach, pracownikach, inwestorach.

Tabela 3: Struktura tabeli Person

Kolumna	Typ danych	Ograniczenia
person_id	INT IDENTITY(1,1)	PRIMARY KEY
first_name	NVARCHAR(100)	NOT NULL
last_name	NVARCHAR(100)	NOT NULL
permalink	NVARCHAR(255)	NOT NULL, UNIQUE

3.1.3. FinancialOrg – Organizacje finansowe

Opis funkcjonalny: Przechowuje informacje o funduszach VC, funduszach inwestycyjnych i innych organizacjach finansowych inwestujących w firmy.

Tabela 4: Struktura tabeli FinancialOrg

Kolumna	Typ danych	Ograniczenia
financial_org_id	INT IDENTITY(1,1)	PRIMARY KEY
name	NVARCHAR(255)	NOT NULL
permalink	NVARCHAR(255)	NOT NULL, UNIQUE

3.1.4. FundingRound – Rundy finansowania

Opis funkcjonalny: Przechowuje informacje o rundach finansowania firm – typ rundy (seed, Series A, B, C...), kwotę, walutę, datę.

Tabela 5: Struktura tabeli FundingRound

Kolumna	Typ danych	Ograniczenia
funding_round_id	INT IDENTITY(1,1)	PRIMARY KEY
company_id	INT	FK → Company, ON DELETE CASCADE
round_code	NVARCHAR(50)	NULL
raised_amount	DECIMAL(18,2)	CHECK (≥ 0)
raised_currency_code	NVARCHAR(10)	DEFAULT 'USD'
funded_year	INT	NULL
funded_month	INT	CHECK (1-12)
funded_day	INT	CHECK (1-31)

3.1.5. Investment – Inwestycje

Opis funkcjonalny: Tabela łącząca inwestorów z rundami finansowania. Inwestorem może być osoba, organizacja finansowa lub inna firma.

Tabela 6: Struktura tabeli Investment

Kolumna	Typ danych	Ograniczenia
investment_id	INT IDENTITY(1,1)	PRIMARY KEY
funding_round_id	INT	FK → FundingRound, ON DELETE CASCADE
person_id	INT	FK → Person (opcjonalny)
financial_org_id	INT	FK → FinancialOrg (opcjonalny)
investing_company_id	INT	FK → Company (opcjonalny)

Uwaga: Inwestorem może być osoba, organizacja finansowa lub firma – stąd trzy opcjonalne klucze obce.

3.1.6. Pozostałe tabele

Product – produkty oferowane przez firmy (FK do Company z ON DELETE CASCADE).

Office – lokalizacje biur i siedzib firm (FK do Company z ON DELETE CASCADE).

CompanyRelationship – tabela asocjacyjna łącząca osoby z firmami, zawiera stanowisko i flagę `is_past`.

Competitor – informacje o konkurentach firm.

Acquisition – informacje o przejęciach firm (firma przejmująca i przejmowana).

Milestone – kamienie milowe w historii firmy.

Provider – dostawcy usług dla firm.

ExternalLink – linki zewnętrzne do artykułów o firmie.

VideoEmbed – osadzone filmy (kod embed).

CompanyImage – obrazy i logo firm.

Screenshot – zrzuty ekranu produktów.

ScreenshotSize – różne rozmiary zrzutów ekranu.

CompanyIPO – informacje o IPO firmy.

3.1.7. Uzasadnienie normalizacji

Baza danych spełnia wszystkie trzy postacie normalne:

1NF (Pierwsza postać normalna):

- Każda kolumna zawiera wartości atomowe (niepodzielne).
- Tablice z JSON (np. products, offices, funding_rounds) zostały rozdzielone do osobnych tabel.
- Każda tabela ma unikalny klucz główny (IDENTITY).

2NF (Druga postać normalna):

- Baza jest w 1NF.
- Wszystkie atrybuty niekluczowe zależą od całego klucza głównego.
- Nie występują częściowe zależności funkcyjne.

3NF (Trzecia postać normalna):

- Baza jest w 2NF.
- Nie występują przechodnie zależności funkcyjne.
- Każdy atrybut niekluczowy zależy bezpośrednio od klucza głównego.

3.2. Indeksy

W bazie danych utworzono następujące indeksy nieklastrowe (poza automatycznymi dla PK i UNIQUE):

Tabela 7: Indeksy utworzone w bazie danych

Nazwa indeksu	Tabela	Kolumna
IX_Company_Name	Company	name
IX_Company_CategoryCode	Company	category_code
IX_FundingRound_RoundCode	FundingRound	round_code

Uzasadnienie utworzenia indeksów:

1. **IX_Company_Name** – przyspiesza wyszukiwanie firm po nazwie, często używane w zapytaniach.
2. **IX_Company_CategoryCode** – przyspiesza filtrowanie firm według kategorii działalności.
3. **IX_FundingRound_RoundCode** – przyspiesza grupowanie i filtrowanie rund finansowania według typu (seed, series-a, series-b, etc.).

3.3. Triggery

W obecnej wersji bazy danych **nie zdefiniowano triggerów**. Integralność danych jest zapewniona przez:

- więzy klucza obcego (FK) z opcją ON DELETE CASCADE,
- ograniczenia CHECK walidujące wartości,
- ograniczenia DEFAULT dla wartości domyślnych,
- procedurę składowaną do bezpiecznego wstawiania/aktualizacji danych.

3.4. Procedury składowane

3.4.1. Procedura: UpsertCompany

Opis: Procedura służąca do wstawiania nowej firmy lub aktualizacji istniejącej na podstawie unikalnego identyfikatora `mongo_id`. Implementuje wzorzec UPSERT (UPDATE or INSERT).

Parametry wejściowe:

- `@mongo_id` (NVARCHAR(50)) – unikalny identyfikator z MongoDB (wymagany)
- `@name` (NVARCHAR(255)) – nazwa firmy (wymagany)
- `@permalink` (NVARCHAR(255)) – unikalny identyfikator URL (wymagany)
- `@category_code` (NVARCHAR(100)) – kategoria działalności (opcjonalny)
- `@description` (NVARCHAR(MAX)) – opis firmy (opcjonalny)
- `@number_of_employees` (INT) – liczba pracowników (opcjonalny)
- `@founded_year` (INT) – rok założenia (opcjonalny)

Działanie:

1. Sprawdza, czy firma o podanym `mongo_id` już istnieje w bazie.
2. Jeśli istnieje – aktualizuje dane (UPDATE) i ustawia `updated_at = GETDATE()`.
3. Jeśli nie istnieje – wstawia nowy rekord (INSERT).
4. Wyświetla komunikat o wykonanej operacji.

```sql
CREATE OR ALTER PROCEDURE crunchbase.UpsertCompany
    @mongo_id NVARCHAR(50),
    @name NVARCHAR(255),
    @permalink NVARCHAR(255),
    @category_code NVARCHAR(100) = NULL,
    @description NVARCHAR(MAX) = NULL,
    @number_of_employees INT = NULL,
    @founded_year INT = NULL
AS
BEGIN
    SET NOCOUNT ON;

    IF EXISTS (SELECT 1 FROM crunchbase.Company
               WHERE mongo_id = @mongo_id)
    BEGIN
        UPDATE crunchbase.Company
        SET name = @name,
            category_code = @category_code,
            description = @description,
            number_of_employees = @number_of_employees,
            founded_year = @founded_year,
            updated_at = GETDATE()
        WHERE mongo_id = @mongo_id;

        PRINT 'Firma zaktualizowana: ' + @name;
    END
    ELSE
    BEGIN
        INSERT INTO crunchbase.Company
            (mongo_id, name, permalink, category_code,
             description, number_of_employees, founded_year)
        VALUES
            (@mongo_id, @name, @permalink, @category_code,
             @description, @number_of_employees, @founded_year);

        PRINT 'Firma dodana: ' + @name;
    END
END
```

Listing 1: Procedura UpsertCompany

Zastosowanie: Procedura umożliwia bezpieczne dodawanie i aktualizowanie firm bez ryzyka duplikacji danych. Jest wykorzystywana przez użytkownika `Emp` do zarządzania danymi.

3.5. Funkcje użytkownika

3.5.1. Funkcja skalarna: GetTotalFunding

Opis: Funkcja skalarna zwracająca łączną sumę finansowania pozyskanego przez firmę we wszystkich rundach finansowania.

Parametry wejściowe:

- `@company_id` (INT) – identyfikator firmy

Zwracana wartość: `DECIMAL(18,2)` – suma finansowania (0 jeśli brak danych)

```
CREATE OR ALTER FUNCTION crunchbase.GetTotalFunding(
    @company_id INT
)
RETURNS DECIMAL(18,2)
AS
BEGIN
    DECLARE @total DECIMAL(18,2);

    SELECT @total = ISNULL(SUM(raised_amount), 0)
    FROM crunchbase.FundingRound
    WHERE company_id = @company_id;

    RETURN @total;
END
```

Listing 2: Funkcja GetTotalFunding

Zastosowanie: Funkcja jest wykorzystywana w widoku `vw_CompanyOverview` do prezentacji podsumowania finansowania każdej firmy. Może być również używana bezpośrednio w zapytaniach SELECT.

Przykład użycia:

```
SELECT name, crunchbase.GetTotalFunding(company_id) AS funding
FROM crunchbase.Company
ORDER BY funding DESC;
```

3.6. Widoki

3.6.1. Widok: vw_CompanyOverview

Opis: Widok prezentujący podsumowanie informacji o firmach wraz z obliczonym łącznym finansowaniem i liczbą produktów oraz rund finansowania.

```
CREATE OR ALTER VIEW crunchbase.vw_CompanyOverview
AS
SELECT
    c.company_id,
    c.name,
    c.category_code,
    c.founded_year,
    c.number_of_employees,
    c.homepage_url,
    crunchbase.GetTotalFunding(c.company_id) AS total_funding,
    (SELECT COUNT(*) FROM crunchbase.FundingRound fr
     WHERE fr.company_id = c.company_id) AS funding_rounds_count,
    (SELECT COUNT(*) FROM crunchbase.Product p
     WHERE p.company_id = c.company_id) AS products_count
```

```
15 FROM crunchbase.Company c;
```

Listing 3: Widok vw_CompanyOverview

Kolumny widoku:

- `company_id` – identyfikator firmy
- `name` – nazwa firmy
- `category_code` – kategoria działalności
- `founded_year` – rok założenia
- `number_of_employees` – liczba pracowników
- `homepage_url` – strona główna firmy
- `total_funding` – łączne finansowanie (obliczone przez funkcję)
- `funding_rounds_count` – liczba rund finansowania
- `products_count` – liczba produktów

Zastosowanie: Widok umożliwia szybki przegląd najważniejszych informacji o firmach bez konieczności pisania złożonych zapytań. Jest udostępniony użytkownikowi `Guest` do przeglądania danych.

Przykład użycia:

```
1 SELECT * FROM crunchbase.vw_CompanyOverview
2 ORDER BY total_funding DESC;
```

4. Role, uprawnienia i użytkownicy

4.1. Konfiguracja bazy dla Contained Users

Baza danych została skonfigurowana z opcją `CONTAINMENT = PARTIAL`, co umożliwia tworzenie użytkowników przechowywanych lokalnie w bazie (contained users). Użytkownicy ci nie wymagają loginów na poziomie serwera.

```
-- Wlaczenie zaawansowanych opcji
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;

-- Wlaczenie contained database authentication
EXEC sp_configure 'contained database authentication', 1;
RECONFIGURE;

-- Utworzenie bazy z CONTAINMENT = PARTIAL
CREATE DATABASE CompanyDB CONTAINMENT = PARTIAL;
```

Listing 4: Konfiguracja Contained Database Authentication

4.2. Role w bazie danych

W bazie zdefiniowano trzy role odpowiadające różnym poziomom dostępu:

Tabela 8: Role zdefiniowane w bazie danych

Rola	Poziom	Uprawnienia
AdminRole	Pełny	Członek roli `db_owner` – pełne uprawnienia do wszystkich obiektów bazy
EmpRole	Ograniczony	Uprawnienie EXECUTE na schemacie `crunchbase` – może wykonywać procedury składowane
GuestRole	Minimalny	Uprawnienie SELECT na widoku `vw_CompanyOverview` – tylko odczyt przez widok

```
-- Rola administratora
CREATE ROLE AdminRole;
ALTER ROLE db_owner ADD MEMBER AdminRole;

-- Rola pracownika
CREATE ROLE EmpRole;
GRANT EXECUTE ON SCHEMA::crunchbase TO EmpRole;

-- Rola goscia
CREATE ROLE GuestRole;
GRANT SELECT ON crunchbase.vw_CompanyOverview TO GuestRole;
```

Listing 5: Tworzenie ról i nadawanie uprawnień

4.3. Użytkownicy (Contained Users)

W bazie utworzono trzech użytkowników lokalnych (contained users):

Tabela 9: Użytkownicy w bazie danych

Użytkownik	Hasło	Rola	Możliwości
Admin	Admin	AdminRole	Pełny dostęp do bazy, tworzenie/modyfikacja/usuwanie obiektów
Emp	Emp	EmpRole	Wykonywanie procedur składowanych (np. UpsertCompany)
Guest	Guest	GuestRole	Przeglądanie danych tylko przez widok

```
-- Administrator
CREATE USER Admin WITH PASSWORD = 'Admin';
ALTER ROLE AdminRole ADD MEMBER Admin;

-- Pracownik
CREATE USER Emp WITH PASSWORD = 'Emp';
ALTER ROLE EmpRole ADD MEMBER Emp;

-- Gosc
CREATE USER Guest WITH PASSWORD = 'Guest';
ALTER ROLE GuestRole ADD MEMBER Guest;
```

Listing 6: Tworzenie użytkowników contained

4.4. Sposób logowania jako Contained User

Aby zalogować się jako contained user w SQL Server Management Studio:

1. Server name: `localhost` (lub nazwa serwera)
2. Authentication: `SQL Server Authentication`
3. Login: `Admin` / `Emp` / `Guest`
4. Password: `Admin` / `Emp` / `Guest`
5. Options → Connect to database: `CompanyDB`

4.5. Testowanie uprawnień

Test użytkownika Admin:

```
-- Logowanie jako Admin - moze wszystko
SELECT * FROM crunchbase.Company;
INSERT INTO crunchbase.Company (mongo_id, name, permalink)
```

```
VALUES ('test123', 'Test', 'test');
EXEC crunchbase.UpsertCompany 'test456', 'Test2', 'test2';
```

Test użytkownika Emp:

```
-- Logowanie jako Emp - moze wykonac procedure
EXEC crunchbase.UpsertCompany 'test789', 'Test3', 'test3';

-- Nie moze bezposrednio modyfikowac tabel
SELECT * FROM crunchbase.Company; -- BLAD!
```

Test użytkownika Guest:

```
-- Logowanie jako Guest - moze tylko przegladac widok
SELECT * FROM crunchbase.vw_CompanyOverview;

-- Nie moze nic wiecej
SELECT * FROM crunchbase.Company; -- BLAD!
EXEC crunchbase.UpsertCompany 'x', 'X', 'x'; -- BLAD!
```

5. Uwagi końcowe

5.1. Napotkane problemy

Podczas realizacji projektu napotkano następujące wyzwania:

1. **Struktura zagnieżdżona JSON** – dokumenty JSON zawierały głęboko zagnieżdżone obiekty (np. `investments` wewnątrz `funding_rounds`). Wymagało to użycia wielokrotnego `CROSS APPLY OPENJSON`.
2. **Polimorficzne inwestycje** – inwestorem może być osoba (`person`), organizacja finansowa (`financial_org`) lub firma (`company`). Rozwiązano przez trzy opcjonalne klucze obce w tabeli `Investment`.
3. **Referencje do nieistniejących firm** – niektórzy konkurenci i przejęte firmy nie istnieją w danych źródłowych (6 firm). Zachowano ich nazwy i permalinki jako tekst, z opcjonalnym kluczem obcym jeśli firma istnieje w bazie.
4. **Contained users** – wymagana była dodatkowa konfiguracja serwera (`sp_configure`) przed utworzeniem użytkowników lokalnych.

5.2. Elementy zrealizowane

- ✓ 19 tabel z pełną strukturą więzów integralności
- ✓ Klucze główne (PK) i obce (FK) dla wszystkich tabel
- ✓ Ograniczenia CHECK, UNIQUE i DEFAULT
- ✓ 3 indeksy nieklastrowe na często używanych kolumnach
- ✓ 1 procedura składowana (UpsertCompany)
- ✓ 1 funkcja skalarna (GetTotalFunding)
- ✓ 1 widok (vw_CompanyOverview)
- ✓ 3 role z różnymi uprawnieniami
- ✓ 3 contained users (Admin, Emp, Guest)
- ✓ Skrypt importu danych z JSON (OPENJSON)
- ✓ Normalizacja do 3NF

5.3. Pliki projektu

Tabela 10: Pliki wchodzące w skład projektu

Plik	Opis
01_struktura.sql	Tworzenie bazy danych, schematu i wszystkich 19 tabel z ograniczeniami
02_obiekty.sql	Procedura składowana, funkcja skalarna i widok
03_uzytkownicy.sql	Role i contained users z uprawnieniami
04_import.sql	Import danych z pliku JSON do wszystkich tabel
Raport_CompanyDB.pdf	Niniejszy dokument

5.4. Kolejność uruchamiania skryptów

1. `01_struktura.sql` – tworzenie bazy i tabel

2. `02_obiekty.sql` – procedura, funkcja, widok
3. `03_uzytkownicy.sql` – role i użytkownicy
4. `04_import.sql` – import danych z JSON

Uwaga: W pliku `04_import.sql` należy zmodyfikować ścieżkę do pliku JSON odpowiednio do lokalizacji na dysku.

Koniec dokumentu