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 stworzenie relacyjnej bazy danych `CompanyDB` na serwerze MS SQL. Baza służy do przechowywania danych o firmach (startupach), ich pracownikach i inwestycjach. Dane wejściowe pochodzą z plików JSON (serwis CrunchBase). Moim zadaniem było zaprojektowanie struktury tabel i napisanie skryptu, który przeniesie te dane z formatu dokumentowego (JSON) do uporządkowanej bazy SQL.

1.2. Główne założenia

Przyjąłem następujące zasady przy tworzeniu bazy:

- **Silnik:** Baza działa na Microsoft SQL Server.
- **Struktura:** Baza jest relacyjna i dąży do 3. Postaci Normalnej (3NF), aby nie powielać niepotrzebnie danych.
- **Bezpieczeństwo:** Użyłem opcji *Contained Database*. Dzięki temu użytkownicy są przypisani bezpośrednio do bazy danych, a nie do całego serwera SQL. To ułatwia przenoszenie bazy.
- **Spójność:** Zastosowałem klucze obce (Foreign Keys). Jeśli usuniemy firmę, to automatycznie usuną się jej produkty czy biura (opcja `ON DELETE CASCADE`).
- **Klucze:** Każda tabela ma swój własny, unikalny numer ID (np. `company_id`), który generuje się automatycznie. Nie używam nazw firm jako kluczy, bo mogą się powtarzać lub zmieniać.

1.3. Zakres możliwości systemu

W bazie można:

- Przechowywać informacje o firmach, ich produktach i adresach biur.
- Sprawdzać historię finansowania — kto, ile i kiedy zainwestował w daną firmę.
- Widzieć powiązania między ludźmi a firmami (kto gdzie pracuje lub pracował).
- Analizować konkurencję i przejęcia firm.
- Automatycznie importować dane z plików JSON za pomocą przygotowanego skryptu SQL.

1.4. Ograniczenia przyjęte podczas projektowania

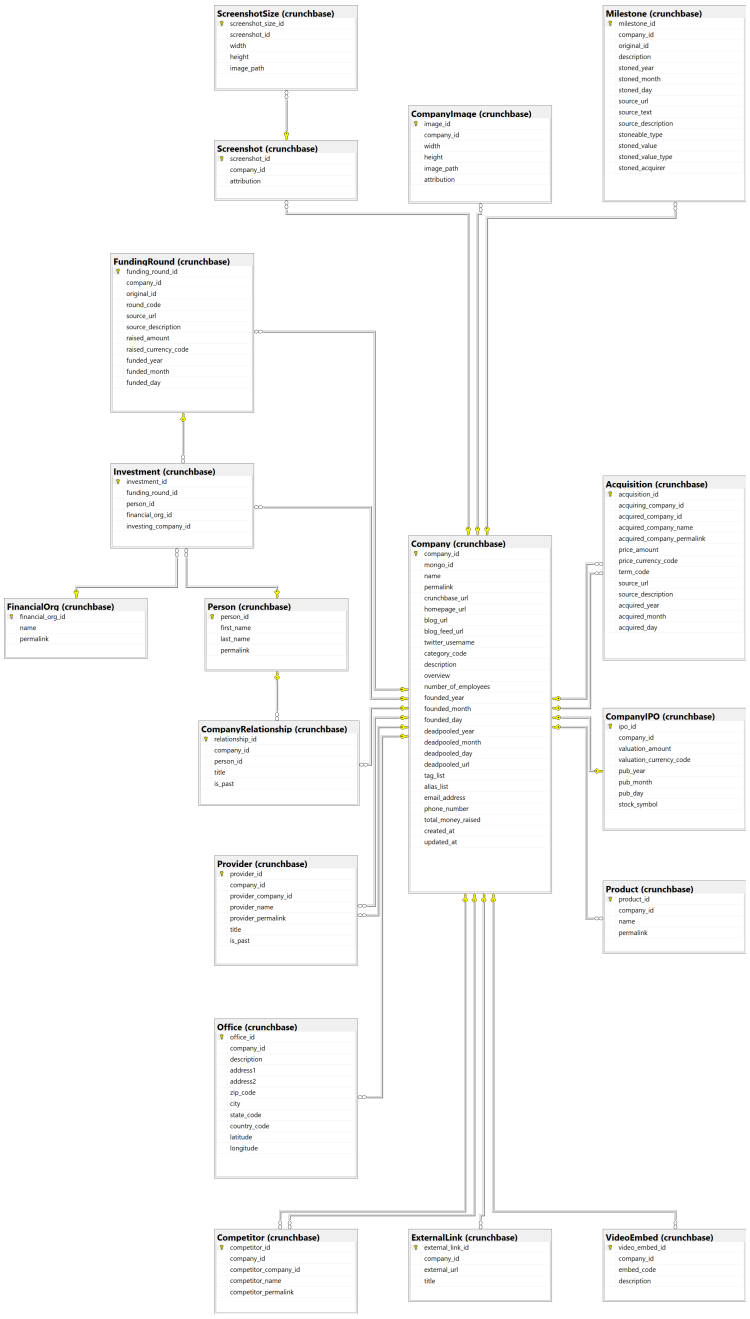
W projekcie zastosowałem kilka uproszczeń:

- **Tagi:** Pola takie jak `tag_list` (lista tagów po przecinku) zostawiłem jako zwykły tekst. Rozbijanie tego na osobne tabele skomplikowałoby import, a w tym projekcie nie jest to kluczowe. To świadome odstępstwo od 1. Postaci Normalnej.
- **Waluty:** Zakładam, że kwoty są w dolarach (USD) lub są przeliczane, nie robiłem osobnego systemu kursów walut.
- **Braki danych:** Pliki JSON są dziurawe (np. brak roku założenia), więc baza pozwala na wpisywanie wartości pustych (`NULL`) w wielu miejscach.

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 relacyjnej

Sercem bazy jest tabela `Company` (Firmy). Wszystkie inne tabele są z nią powiązane — np. `Product` (Produkty), `Office` (Biura).

Podczas projektowania musiałem rozwiązać dwa trudniejsze problemy:

1. Relacja Wiele-do-Wielu (Ludzie i Firmy)

Jedna osoba może pracować w wielu firmach, a w firmie pracuje wiele osób. Nie mogłem wpisać pracownika bezpośrednio do tabeli firmy.

- **Rozwiązanie:** Stworzyłem tabelę łączącą `CompanyRelationship`. Łączy ona ID osoby z ID firmy i dodaje informację o stanowisku (np. "CEO").

2. Kto jest inwestorem? (Różne typy inwestorów)

W firmę może zainwestować: inna osoba, inna firma albo bank/fundusz.

- **Rozwiązanie:** W tabeli `Investment` (Inwestycje) mam trzy kolumny na ID: `person_id`, `financial_org_id` i `investing_company_id`. Wypełniam tylko jedną z nich dla danej inwestycji, a reszta zostaje pusta. To proste i skuteczne rozwiązanie problemu polimorfizmu w SQL.

2.3. Normalizacja

Starałem się trzymać zasad normalizacji:

- **1NF (Atomowość):** W każdej kratce tabeli jest jedna wartość. Wyjątkiem są wspomniane wcześniej tagi, które zostawiłem jako listę po przecinku dla wygody.
- **2NF:** Każda tabela ma swój klucz główny (`ID`), więc wszystkie dane w wierszu dotyczą konkretnego obiektu (np. konkretnej firmy). Nie ma sytuacji, że część danych zależy od czegoś innego.
- **3NF:** Usunąłem zależności przechodnie.
  Mały wyjątek: W tabeli `Acquisition` (Przejęcia) trzymam nazwę przejętej firmy ORAZ jej ID. Robię to celowo — czasem przejmowana firma nie istnieje w naszej bazie (nie ma ID), a musimy wiedzieć, jak się nazywała. To bezpieczniejsza opcja.

3. Obiekty bazy danych i ich opis

3.1. Tabele

Baza danych składa się z 18 tabel. Poniżej znajduje się szczegółowy opis każdej z nich.

3.1.1. 1. Tabela crunchbase.Company

Opis: Główna tabela przechowująca podstawowe informacje o firmach (startupach), takie jak nazwa, kategoria, opis, daty założenia i zamknięcia.

Klucz główny (PK): `company_id` (INT, IDENTITY).

Klucze obce (FK): Brak.

Ograniczenia:

- `UNIQUE` na kolumnach `mongo_id` oraz `permalink`.
- `CHECK (number_of_employees >= 0)` — liczba pracowników nie może być ujemna.
- `CHECK (founded_month BETWEEN 1 AND 12)` — walidacja miesiąca.
- `CHECK (founded_day BETWEEN 1 AND 31)` — walidacja dnia.
- `DEFAULT GETDATE()` dla dat utworzenia rekordu.

Uwagi: Pole `tag_list` jest przechowywane jako tekst (odstępstwo od 1NF dla uproszczenia).

3.1.2. 2. Tabela crunchbase.Person

Opis: Słownik osób (pracowników, założycieli, inwestorów).

Klucz główny (PK): `person_id` (INT, IDENTITY).

Klucze obce (FK): Brak.

Ograniczenia: `UNIQUE` na kolumnie `permalink`.

3.1.3. 3. Tabela crunchbase.FinancialOrg

Opis: Słownik organizacji finansowych (fundusze VC, banki).

Klucz główny (PK): `financial_org_id` (INT, IDENTITY).

Klucze obce (FK): Brak.

Ograniczenia: `UNIQUE` na kolumnie `permalink`.

3.1.4. 4. Tabela crunchbase.Product

Opis: Produkty oferowane przez firmy.

Klucz główny (PK): `product_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.5. 5. Tabela crunchbase.Office

Opis: Lokalizacje biur i siedzib firm.

Klucz główny (PK): `office_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.6. 6. Tabela crunchbase.FundingRound

Opis: Rundy finansowania firm — typ rundy (seed, Series A, B, C, etc.), kwota, waluta, data.

Klucz główny (PK): `funding_round_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia:

- `CHECK (raised_amount >= 0)` — kwota nie może być ujemna.
- `DEFAULT 'USD'` dla `raised_currency_code`.
- `CHECK (funded_month BETWEEN 1 AND 12)` — walidacja miesiąca.
- `CHECK (funded_day BETWEEN 1 AND 31)` — walidacja dnia.

3.1.7. 7. Tabela crunchbase.Investment

Opis: Łączy rundę finansowania z inwestorem. Inwestorem może być osoba, organizacja finansowa lub inna firma.

Klucz główny (PK): `investment_id` (INT, IDENTITY).

Klucze obce (FK):

- `funding_round_id` → `FundingRound` (ON DELETE CASCADE) — w co zainwestowano.
- `person_id` → `Person` (opcjonalny) — jeśli inwestorem jest osoba.
- `financial_org_id` → `FinancialOrg` (opcjonalny) — jeśli inwestorem jest organizacja.
- `investing_company_id` → `Company` (opcjonalny) — jeśli inwestorem jest inna firma.

Uwagi: W danym rekordzie wypełniony jest zazwyczaj tylko jeden z trzech kluczy inwestorów.

3.1.8. 8. Tabela crunchbase.Acquisition

Opis: Informacje o zakupie jednej firmy przez drugą.

Klucz główny (PK): `acquisition_id` (INT, IDENTITY).

Klucze obce (FK):

- `acquiring_company_id` → `Company` — firma kupująca.
- `acquired_company_id` → `Company` (opcjonalny) — firma kupowana.

Uwagi do normalizacji: Przechowuję nazwę przejmowanej firmy (`acquired_company_name`) jako tekst, nawet jeśli mam do niej ID. To zabezpieczenie na wypadek, gdyby przejmowana firma nie istniała w naszej bazie jako osobny rekord (świadoma redundancja).

3.1.9. 9. Tabela crunchbase.Milestone

Opis: Kamienie milowe w historii firmy (ważne wydarzenia).

Klucz główny (PK): `milestone_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.10. 10. Tabela crunchbase.Competitor

Opis: Informacje o konkurentach firm.

Klucz główny (PK): `competitor_id` (INT, IDENTITY).

Klucze obce (FK):

- `company_id` → `Company` (ON DELETE CASCADE) — firma.
- `competitor_company_id` → `Company` (opcjonalny) — firma konkurencyjna.

Ograniczenia: `UNIQUE (company_id, competitor_permalink)` — unikalna para firma-konkurent.

3.1.11. 11. Tabela crunchbase.CompanyRelationship

Opis: Tabela łącznikowa dla relacji wiele-do-wielu między osobami a firmami. Przechowuje historię zatrudnienia.

Klucz główny (PK): `relationship_id` (INT, IDENTITY).

Klucze obce (FK):

- `company_id` → `Company` (ON DELETE CASCADE).
- `person_id` → `Person` (ON DELETE CASCADE).

Dodatkowe pola: `title` (stanowisko), `is_past` (czy to była praca w przeszłości).

3.1.12. 12. Tabela crunchbase.ExternalLink

Opis: Linki zewnętrzne do artykułów o firmie.

Klucz główny (PK): `external_link_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.13. 13. Tabela crunchbase.Screenshot

Opis: Zrzuty ekranu produktów.

Klucz główny (PK): `screenshot_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.14. 14. Tabela crunchbase.ScreenshotSize

Opis: Różne rozmiary zrzutów ekranu (thumbnails).

Klucz główny (PK): `screenshot_size_id` (INT, IDENTITY).

Klucz obcy (FK): `screenshot_id` → `Screenshot` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.15. 15. Tabela crunchbase.VideoEmbed

Opis: Osadzone filmy (kod embed) dotyczące firm.

Klucz główny (PK): `video_embed_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.16. 16. Tabela crunchbase.Provider

Opis: Dostawcy usług dla firm.

Klucz główny (PK): `provider_id` (INT, IDENTITY).

Klucze obce (FK):

- `company_id` → `Company` (ON DELETE CASCADE) — firma.
- `provider_company_id` → `Company` (opcjonalny) — firma dostawcy.

Ograniczenia: Brak dodatkowych.

3.1.17. 17. Tabela crunchbase.CompanyImage

Opis: Obrazy i logo firm.

Klucz główny (PK): `image_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE).

Ograniczenia: Brak dodatkowych.

3.1.18. 18. Tabela crunchbase.CompanyIPO

Opis: Informacje o IPO (Initial Public Offering) firmy.

Klucz główny (PK): `ipo_id` (INT, IDENTITY).

Klucz obcy (FK): `company_id` → `Company` (ON DELETE CASCADE, UNIQUE).

Ograniczenia: Firma może mieć tylko jedno IPO (relacja 1:1).

3.2. Indeksy

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

Tabela 1: 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**. Zrezygnowano z triggerów na rzecz więzów integralności (FOREIGN KEY CASCADE) oraz procedur składowanych, co zapewnia lepszą wydajność i przejrzystość logiki bazy.

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.

```
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
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:

```
SELECT * FROM crunchbase.vw_CompanyOverview
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 2: 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 3: 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

```sql
-- 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:

```sql
-- 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

- ✓ 18 tabel w schemacie `crunchbase` 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 4: Pliki wchodzące w skład projektu

Plik	Opis
01_struktura.sql	Tworzenie bazy danych, schematu i wszystkich 18 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
05_testy.sql	Zapytania testowe i demonstracyjne (SELECT, JOIN, agregacje)
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
5. `05_testy.sql` — zapytania testowe (opcjonalnie, do demonstracji)

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

Koniec dokumentu