**POLITECHNIKA ŁÓDZKA**
Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki
Instytut Informatyki Stosowanej

## Projektowanie i Administracja Baz Danych

Semestr 5

# Projekt Bazy Danych

### Company (CrunchBase)

**Autor:** Mateusz Mróz
**Nr indeksu:** 251190
**Kierunek:** Informatyka
**Tryb studiów:** Stacjonarne

Łódź, styczeń 2026

# Spis treści

# 1 Założenia wstępne

## 1.1 Cel projektu

Celem projektu jest zaprojektowanie i implementacja relacyjnej bazy danych na podstawie danych pochodzących z serwisu CrunchBase. Baza danych ma przechowywać informacje o firmach technologicznych, ich finansowaniu, pracownikach, produktach oraz przejęciach.

## 1.2 Źródło danych

Dane źródłowe pochodzą z pliku JSON zawierającego 6 dokumentów opisujących firmy:

- Wetpaint
- Facebook
- Twitter
- Omnidrive
- StumbleUpon
- Scribd

## 1.3 Wymagania funkcjonalne

1. Przechowywanie danych o firmach (nazwa, opis, kategoria, data założenia, liczba pracowników)
2. Rejestracja rund finansowania i inwestycji
3. Śledzenie relacji między osobami a firmami
4. Ewidencja produktów i biur firm
5. Historia przejęć i konkurencji
6. Przechowywanie kamieni milowych firm

## 1.4 Wymagania niefunkcjonalne

1. Normalizacja do trzeciej postaci normalnej (3NF)
2. Integralność referencyjna poprzez klucze obce
3. Ograniczenia CHECK dla walidacji danych
4. Indeksy dla optymalizacji zapytań
5. System ról i uprawnień (contained users)

## 1.5 Technologie

- **SZBD:** Microsoft SQL Server 2019+
- **Język:** T-SQL
- **Tryb bazy:** CONTAINMENT = PARTIAL (dla contained users)

# 2 Diagram ERD

## 2.1 Opis encji

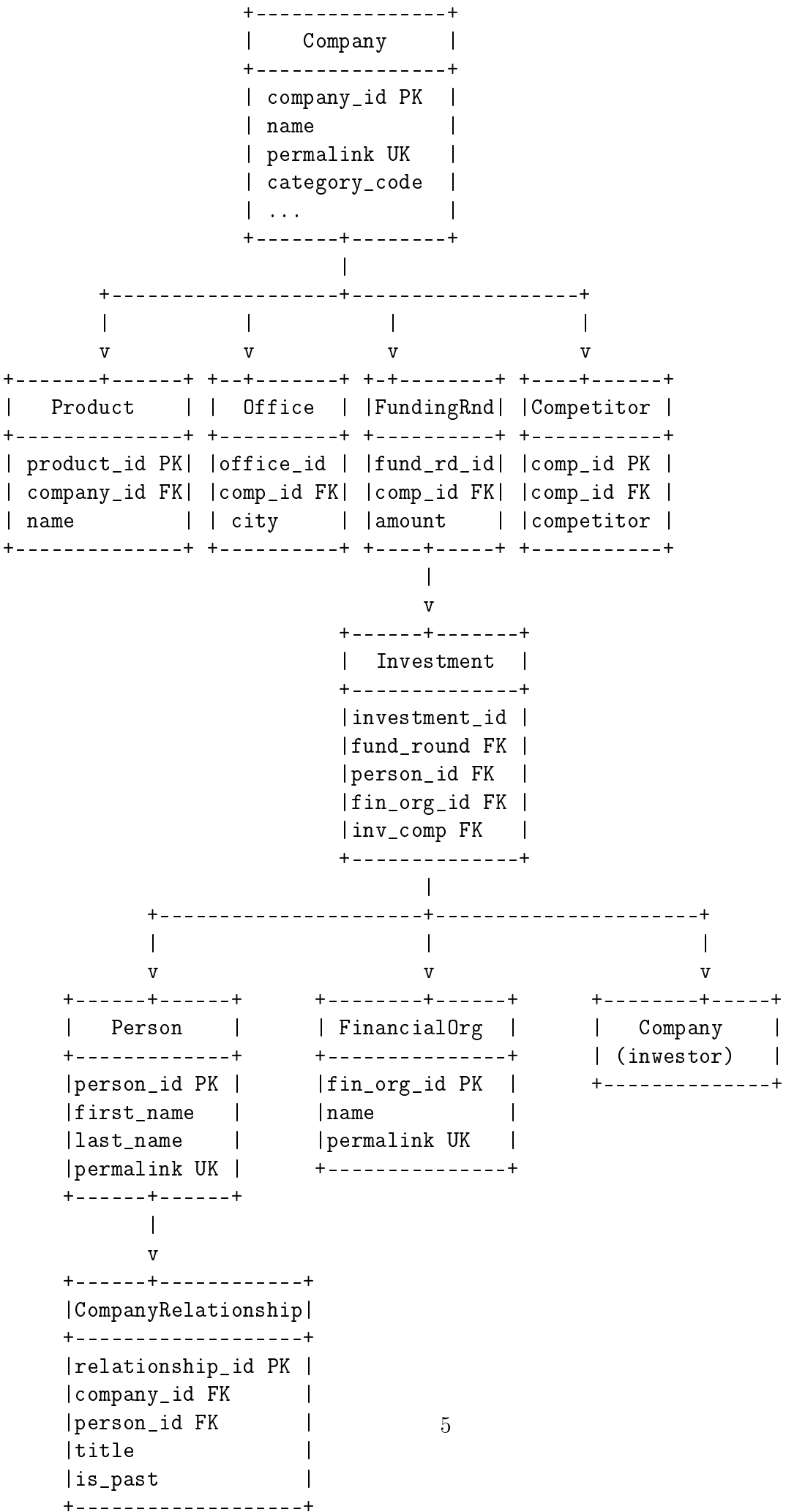
Baza danych składa się z 18 tabel reprezentujących następujące encje:

Tabela 1: Lista tabel w bazie danych

| **Tabela** | **Opis** | **Klucz główny** |
|---|---|---|
| Company | Firmy | company_id (INT IDENTITY) |
| Person | Osoby (założyciele, inwestorzy) | person_id (INT IDENTITY) |
| FinancialOrg | Organizacje finansowe | financial_org_id (INT IDENTITY) |
| Product | Produkty firm | product_id (INT IDENTITY) |
| Office | Biura firm | office_id (INT IDENTITY) |
| FundingRound | Rundy finansowania | funding_round_id (INT IDENTITY) |
| Investment | Inwestycje w rundach | investment_id (INT IDENTITY) |
| Acquisition | Przejęcia firm | acquisition_id (INT IDENTITY) |
| Milestone | Kamienie milowe | milestone_id (INT IDENTITY) |
| Competitor | Konkurenci | competitor_id (INT IDENTITY) |
| CompanyRelationship | Relacje osoba-firma | relationship_id (INT IDENTITY) |
| ExternalLink | Linki zewnętrzne | external_link_id (INT IDENTITY) |
| Screenshot | Zrzuty ekranu | screenshot_id (INT IDENTITY) |
| ScreenshotSize | Rozmiary zrzutów | screenshot_size_id (INT IDENTITY) |
| VideoEmbed | Osadzone filmy | video_embed_id (INT IDENTITY) |
| Provider | Dostawcy usług | provider_id (INT IDENTITY) |
| CompanyImage | Obrazy firm | company_image_id (INT IDENTITY) |
| CompanyIPO | Oferty publiczne (IPO) | ipo_id (INT IDENTITY) |

## 2.2 Diagram relacji

Poniżej przedstawiono uproszczony diagram relacji między głównymi encjami:


Company
company_id PK
name
permalink UK
category_code
...
Product
product_id PK
company_id FK
name
Office
office_id
comp_id FK
city
FundingRnd
fund_rd_id
comp_id FK
amount
Competitor
comp_id PK
comp_id FK
competitor
Investment
investment_id
fund_round FK
person_id FK
fin_org_id FK
inv_comp FK
Person
person_id PK
first_name
last_name
permalink UK
FinancialOrg
fin_org_id PK
name
permalink UK
Company
(inwestor)
CompanyRelationship
relationship_id PK
company_id FK
person_id FK
title
is_past

## 2.3 Relacje między tabelami

- **Company - Product**: 1:N (firma może mieć wiele produktów)
- **Company - Office**: 1:N (firma może mieć wiele biur)
- **Company - FundingRound**: 1:N (firma może mieć wiele rund finansowania)
- **FundingRound - Investment**: 1:N (runda może mieć wielu inwestorów)
- **Person - Investment**: 1:N (osoba może inwestować w wiele rund)
- **FinancialOrg - Investment**: 1:N (org. finansowa może inwestować w wiele rund)
- **Company - CompanyRelationship - Person**: M:N (osoby pracują w firmach)
- **Company - Competitor - Company**: M:N (firmy konkurują ze sobą)
- **Company - Acquisition - Company**: 1:N (firmy przejmują inne firmy)

# 3 Opis obiektów bazy danych

## 3.1 Tabele

### 3.1.1 Tabela Company

Główna tabela przechowująca informacje o firmach.

Listing 1: Struktura tabeli Company

```
CREATE TABLE crunchbase.Company (
    company_id INT IDENTITY(1,1) PRIMARY KEY,
    mongo_id VARCHAR(50) NULL,
    name NVARCHAR(255) NOT NULL,
    permalink NVARCHAR(255) NOT NULL UNIQUE,
    crunchbase_url NVARCHAR(500) NULL,
    homepage_url NVARCHAR(500) NULL,
    category_code VARCHAR(100) NULL,
    description NVARCHAR(MAX) NULL,
    number_of_employees INT NULL CHECK (number_of_employees >= 0),
    founded_year INT NULL,
    founded_month INT NULL CHECK (founded_month BETWEEN 1 AND 12),
    founded_day INT NULL CHECK (founded_day BETWEEN 1 AND 31),
    total_money_raised VARCHAR(100) NULL,
    created_at DATETIME DEFAULT GETDATE(),
    updated_at DATETIME DEFAULT GETDATE()
);
```

**Ograniczenia:**

- PRIMARY KEY na company_id
- UNIQUE na permalink
- CHECK na number_of_employees >= 0
- CHECK na founded_month (1-12)
- CHECK na founded_day (1-31)
- DEFAULT GETDATE() dla created_at i updated_at

### 3.1.2 Tabela FundingRound

Przechowuje informacje o rundach finansowania.

Listing 2: Struktura tabeli FundingRound

```
CREATE TABLE crunchbase.FundingRound (
    funding_round_id INT IDENTITY(1,1) PRIMARY KEY,
    company_id INT NOT NULL FOREIGN KEY
        REFERENCES crunchbase.Company(company_id) ON DELETE CASCADE
        ,
    original_id INT NULL,
    round_code VARCHAR(50) NULL,
```

```
    raised_amount DECIMAL(18,2) NULL,
    raised_currency_code VARCHAR(10) DEFAULT 'USD',
    funded_year INT NULL,
    funded_month INT NULL,
    funded_day INT NULL
);
```

### 3.1.3 Tabela Investment

Tabela asocjacyjna dla inwestycji w rundach finansowania.

Listing 3: Struktura tabeli Investment

```
CREATE TABLE crunchbase.Investment (
    investment_id INT IDENTITY(1,1) PRIMARY KEY,
    funding_round_id INT NOT NULL FOREIGN KEY
        REFERENCES crunchbase.FundingRound(funding_round_id) ON
           DELETE CASCADE,
    person_id INT NULL FOREIGN KEY
        REFERENCES crunchbase.Person(person_id),
    financial_org_id INT NULL FOREIGN KEY
        REFERENCES crunchbase.FinancialOrg(financial_org_id),
    investing_company_id INT NULL FOREIGN KEY
        REFERENCES crunchbase.Company(company_id)
);
```

## 3.2 Indeksy

Listing 4: Definicje indeksów

```
CREATE NONCLUSTERED INDEX IX_Company_Name
    ON crunchbase.Company(name);
CREATE NONCLUSTERED INDEX IX_Company_CategoryCode
    ON crunchbase.Company(category_code);
CREATE NONCLUSTERED INDEX IX_Company_FoundedYear
    ON crunchbase.Company(founded_year);
CREATE NONCLUSTERED INDEX IX_Person_LastName
    ON crunchbase.Person(last_name);
CREATE NONCLUSTERED INDEX IX_FundingRound_RoundCode
    ON crunchbase.FundingRound(round_code);
CREATE NONCLUSTERED INDEX IX_FundingRound_RaisedAmount
    ON crunchbase.FundingRound(raised_amount);
CREATE NONCLUSTERED INDEX IX_Office_City
    ON crunchbase.Office(city);
```

## 3.3 Procedury składowane

### 3.3.1 AddCompany

Procedura dodająca nową firmę do bazy danych.

Listing 5: Procedura AddCompany

```
CREATE PROCEDURE crunchbase.AddCompany
    @name NVARCHAR(255),
    @permalink NVARCHAR(255),
    @category_code VARCHAR(100) = NULL,
    @description NVARCHAR(MAX) = NULL,
    @founded_year INT = NULL,
    @new_company_id INT OUTPUT
AS
BEGIN
    -- Walidacja uniqueness permalink
    IF EXISTS (SELECT 1 FROM crunchbase.Company
               WHERE permalink = @permalink)
    BEGIN
        RAISERROR('Firma o podanym permalink juz istnieje!', 16, 1)
           ;
        RETURN;
    END

    INSERT INTO crunchbase.Company (name, permalink, category_code,
                                    description, founded_year)
    VALUES (@name, @permalink, @category_code,
            @description, @founded_year);

    SET @new_company_id = SCOPE_IDENTITY();
END
```

### 3.3.2 SearchCompanies

Procedura wyszukująca firmy według różnych kryteriów.

Listing 6: Procedura SearchCompanies

```
CREATE PROCEDURE crunchbase.SearchCompanies
    @search_name NVARCHAR(255) = NULL,
    @category_code VARCHAR(100) = NULL,
    @min_employees INT = NULL,
    @founded_year_from INT = NULL,
    @has_funding BIT = NULL
AS
BEGIN
    SELECT c.company_id, c.name, c.permalink, c.category_code,
           c.number_of_employees, c.founded_year,
           (SELECT COUNT(*) FROM crunchbase.FundingRound fr
            WHERE fr.company_id = c.company_id) AS funding_rounds
    FROM crunchbase.Company c
    WHERE (@search_name IS NULL OR c.name LIKE '%' + @search_name +
        '%')
      AND (@category_code IS NULL OR c.category_code =
          @category_code)
      AND (@min_employees IS NULL OR c.number_of_employees >=
          @min_employees)
```

```
17        AND (@founded_year_from IS NULL OR c.founded_year >=
              @founded_year_from)
18    ORDER BY c.name;
19 END
```

### 3.3.3 GetCompanyFundingReport

Procedura generująca raport finansowania firmy.

Listing 7: Procedura GetCompanyFundingReport

```
1  CREATE PROCEDURE crunchbase.GetCompanyFundingReport
2      @company_id INT
3  AS
4  BEGIN
5      -- Informacje o firmie
6      SELECT c.name, c.category_code, c.founded_year,
7             c.number_of_employees, c.total_money_raised
8      FROM crunchbase.Company c WHERE c.company_id = @company_id;
9
10     -- Rundy finansowania
11     SELECT fr.round_code, fr.raised_amount, fr.raised_currency_code
           ,
12             fr.funded_year, fr.funded_month
13     FROM crunchbase.FundingRound fr
14     WHERE fr.company_id = @company_id
15     ORDER BY fr.funded_year DESC;
16
17     -- Podsumowanie
18     SELECT COUNT(*) AS LiczbaRund, SUM(raised_amount) AS
          SumaFinansowania,
19             AVG(raised_amount) AS SredniaKwota
20     FROM crunchbase.FundingRound WHERE company_id = @company_id;
21 END
```

## 3.4 Funkcje

### 3.4.1 GetTotalFunding (skalarna)

Oblicza całkowite finansowanie firmy.

Listing 8: Funkcja GetTotalFunding

```
1  CREATE FUNCTION crunchbase.GetTotalFunding(@company_id INT)
2  RETURNS DECIMAL(18,2)
3  AS
4  BEGIN
5      DECLARE @total DECIMAL(18,2);
6      SELECT @total = ISNULL(SUM(raised_amount), 0)
7      FROM crunchbase.FundingRound
8      WHERE company_id = @company_id;
9      RETURN @total;
```

```
END
```

### 3.4.2 GetCompanyAge (skalarna)

Oblicza wiek firmy w latach.

Listing 9: Funkcja GetCompanyAge

```
CREATE FUNCTION crunchbase.GetCompanyAge(@company_id INT)
RETURNS INT
AS
BEGIN
    DECLARE @age INT, @founded_year INT;
    SELECT @founded_year = founded_year
    FROM crunchbase.Company WHERE company_id = @company_id;

    IF @founded_year IS NULL SET @age = NULL;
    ELSE SET @age = YEAR(GETDATE()) - @founded_year;
    RETURN @age;
END
```

### 3.4.3 GetCompanyPeople (tabelaryczna)

Zwraca listę osób powiązanych z firmą.

Listing 10: Funkcja GetCompanyPeople

```
CREATE FUNCTION crunchbase.GetCompanyPeople(@company_id INT)
RETURNS TABLE
AS
RETURN (
    SELECT p.person_id, p.first_name, p.last_name, p.permalink,
           cr.title AS Stanowisko,
           CASE WHEN cr.is_past = 1 THEN 'Tak' ELSE 'Nie' END AS
              Byly
    FROM crunchbase.CompanyRelationship cr
    INNER JOIN crunchbase.Person p ON p.person_id = cr.person_id
    WHERE cr.company_id = @company_id
);
```

## 3.5 Widoki

### 3.5.1 vw_CompanyOverview

Widok pokazujący przegląd firm z podstawowymi statystykami.

Listing 11: Widok vw_CompanyOverview

```
CREATE VIEW crunchbase.vw_CompanyOverview AS
SELECT
    c.company_id, c.name AS Nazwa, c.permalink, c.category_code AS
       Kategoria,
    c.number_of_employees AS LiczbaPracownikow,
```

```
    crunchbase.GetCompanyAge(c.company_id) AS WiekFirmy,
    crunchbase.GetTotalFunding(c.company_id) AS
       CalkowiteFinansowanie,
    (SELECT COUNT(*) FROM crunchbase.FundingRound fr
     WHERE fr.company_id = c.company_id) AS LiczbaRund,
    (SELECT COUNT(*) FROM crunchbase.Product p
     WHERE p.company_id = c.company_id) AS LiczbaProdukow
FROM crunchbase.Company c;
```

### 3.5.2 vw_FundingByCategory

Widok pokazujący statystyki finansowania według kategorii.

Listing 12: Widok vw_FundingByCategory

```
CREATE VIEW crunchbase.vw_FundingByCategory AS
SELECT
    c.category_code AS Kategoria,
    COUNT(DISTINCT c.company_id) AS LiczbaFirm,
    COUNT(fr.funding_round_id) AS LiczbaRund,
    SUM(fr.raised_amount) AS SumaFinansowania,
    AVG(fr.raised_amount) AS SredniaKwota
FROM crunchbase.Company c
LEFT JOIN crunchbase.FundingRound fr ON fr.company_id = c.
   company_id
WHERE c.category_code IS NOT NULL
GROUP BY c.category_code;
```

### 3.5.3 vw_TopInvestors

Widok pokazujący najbardziej aktywnych inwestorów.

Listing 13: Widok vw_TopInvestors

```
CREATE VIEW crunchbase.vw_TopInvestors AS
SELECT InvestorType, InvestorName, InvestorPermalink,
       InvestmentCount, TotalCompaniesInvested
FROM (
    -- Osoby
    SELECT 'Osoba' AS InvestorType,
           CONCAT(p.first_name, ' ', p.last_name) AS InvestorName,
           p.permalink AS InvestorPermalink, COUNT(*) AS
              InvestmentCount,
           COUNT(DISTINCT fr.company_id) AS TotalCompaniesInvested
    FROM crunchbase.Investment inv
    INNER JOIN crunchbase.Person p ON p.person_id = inv.person_id
    INNER JOIN crunchbase.FundingRound fr
        ON fr.funding_round_id = inv.funding_round_id
    GROUP BY p.person_id, p.first_name, p.last_name, p.permalink
    UNION ALL
    -- Organizacje finansowe
    SELECT 'Organizacja' AS InvestorType, fo.name, fo.permalink,
```

```
            COUNT(*), COUNT(DISTINCT fr.company_id)
    FROM crunchbase.Investment inv
    INNER JOIN crunchbase.FinancialOrg fo
        ON fo.financial_org_id = inv.financial_org_id
    INNER JOIN crunchbase.FundingRound fr
        ON fr.funding_round_id = inv.funding_round_id
    GROUP BY fo.financial_org_id, fo.name, fo.permalink
) AS AllInvestors;
```

## 3.6 Triggery

### 3.6.1 trg_Company_UpdateTimestamp

Automatyczna aktualizacja znacznika czasu przy modyfikacji firmy.

Listing 14: Trigger trg_Company_UpdateTimestamp

```
CREATE TRIGGER crunchbase.trg_Company_UpdateTimestamp
ON crunchbase.Company
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE crunchbase.Company
    SET updated_at = GETDATE()
    FROM crunchbase.Company c
    INNER JOIN inserted i ON c.company_id = i.company_id
    WHERE c.updated_at = i.updated_at;
END
```

### 3.6.2 trg_FundingRound_Audit

Trigger audytowy rejestrujący zmiany w rundach finansowania.

Listing 15: Trigger trg_FundingRound_Audit

```
CREATE TRIGGER crunchbase.trg_FundingRound_Audit
ON crunchbase.FundingRound
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- INSERT
    IF EXISTS (SELECT 1 FROM inserted) AND NOT EXISTS (SELECT 1
       FROM deleted)
        INSERT INTO crunchbase.FundingRound_Audit
            (funding_round_id, company_id, action_type,
             new_raised_amount, new_round_code)
        SELECT funding_round_id, company_id, 'INSERT',
               raised_amount, round_code FROM inserted;

```

```
    -- UPDATE
    IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM
       deleted)
        INSERT INTO crunchbase.FundingRound_Audit
            (funding_round_id, company_id, action_type,
             old_raised_amount, new_raised_amount, old_round_code,
                new_round_code)
        SELECT i.funding_round_id, i.company_id, 'UPDATE',
               d.raised_amount, i.raised_amount, d.round_code, i.
                  round_code
        FROM inserted i INNER JOIN deleted d
            ON i.funding_round_id = d.funding_round_id;

    -- DELETE
    IF NOT EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1
       FROM deleted)
        INSERT INTO crunchbase.FundingRound_Audit
            (funding_round_id, company_id, action_type,
             old_raised_amount, old_round_code)
        SELECT funding_round_id, company_id, 'DELETE',
               raised_amount, round_code FROM deleted;
END
```

### 3.6.3 trg_Investment_Validate

Trigger walidacyjny sprawdzający poprawność inwestycji.

Listing 16: Trigger trg_Investment_Validate

```
CREATE TRIGGER crunchbase.trg_Investment_Validate
ON crunchbase.Investment
INSTEAD OF INSERT
AS
BEGIN
    SET NOCOUNT ON;

    -- Sprawdzenie czy inwestor jest podany
    IF EXISTS (SELECT 1 FROM inserted
               WHERE person_id IS NULL
                 AND financial_org_id IS NULL
                 AND investing_company_id IS NULL)
    BEGIN
        RAISERROR('Inwestycja musi miec inwestora!', 16, 1);
        RETURN;
    END

    -- Wstawienie zatwierdzonych rekordow
    INSERT INTO crunchbase.Investment
        (funding_round_id, person_id, financial_org_id,
           investing_company_id)
    SELECT funding_round_id, person_id, financial_org_id,
       investing_company_id
```

```
22      FROM inserted;
23  END
```

# 4 Role, uprawnienia i użytkownicy

## 4.1 Architektura bezpieczeństwa

Baza danych wykorzystuje mechanizm **contained users** (użytkownicy zawierci), co umożliwia:

- Logowanie bezpośrednio do bazy danych bez potrzeby konta na serwerze
- Łatwiejszą przenośność bazy danych między serwerami
- Izolację uprawnień na poziomie bazy danych

## 4.2 Role

W bazie danych zdefiniowano 5 ról:

Tabela 2: Zdefiniowane role i ich uprawnienia

| Rola | Uprawnienia |
|---|---|
| role_Admin | Pełna kontrola (db_owner) - wszystkie operacje |
| role_Employee | EXECUTE + SELECT na schemacie crunchbase |
| role_Guest | SELECT tylko na widokach |
| role_DataAnalyst | SELECT + EXECUTE (bez modyfikacji danych) |
| role_ReportViewer | SELECT na widokach + wybrane procedury raportowe |

## 4.3 Użytkownicy

Tabela 3: Użytkownicy contained i ich role

| Użytkownik | Rola | Opis |
|---|---|---|
| admin_user | role_Admin | Administrator z pełnym dostępem |
| emp_user | role_Employee | Pracownik - procedury i odczyt |
| guest_user | role_Guest | Gość - tylko widoki |
| analyst_user | role_DataAnalyst | Analityk danych |
| report_user | role_ReportViewer | Przeglądanie raportów |

## 4.4 Kod tworzący role i użytkowników

Listing 17: Tworzenie ról

```
-- Rola Admin
CREATE ROLE role_Admin;
ALTER ROLE db_owner ADD MEMBER role_Admin;

```

```sql
-- Rola Employee
CREATE ROLE role_Employee;
GRANT EXECUTE ON SCHEMA::crunchbase TO role_Employee;
GRANT SELECT ON SCHEMA::crunchbase TO role_Employee;

-- Rola Guest
CREATE ROLE role_Guest;
GRANT SELECT ON crunchbase.vw_CompanyOverview TO role_Guest;
GRANT SELECT ON crunchbase.vw_FundingByCategory TO role_Guest;
GRANT SELECT ON crunchbase.vw_TopInvestors TO role_Guest;
DENY SELECT ON crunchbase.Company TO role_Guest;
DENY INSERT, UPDATE, DELETE ON SCHEMA::crunchbase TO role_Guest;
```

Listing 18: Tworzenie użytkowników contained

```sql
-- Administrator
CREATE USER admin_user WITH PASSWORD = 'Admin123!@#Strong';
ALTER ROLE role_Admin ADD MEMBER admin_user;

-- Pracownik
CREATE USER emp_user WITH PASSWORD = 'Emp456!@#Strong';
ALTER ROLE role_Employee ADD MEMBER emp_user;

-- Go
CREATE USER guest_user WITH PASSWORD = 'Guest789!@#Strong';
ALTER ROLE role_Guest ADD MEMBER guest_user;
```

## 4.5 Instrukcja logowania

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

1. Server: localhost (lub nazwa serwera)
2. Authentication: SQL Server Authentication
3. Login: nazwa użytkownika (np. admin_user)
4. Password: hasło użytkownika
5. Options → Connection Properties → Connect to database: CompanyDB

# 5 Uwagi końcowe

## 5.1 Normalizacja

Baza danych została znormalizowana do trzeciej postaci normalnej (3NF):

- **1NF**: Wszystkie atrybuty są atomowe, każda tabela ma klucz główny
- **2NF**: Usunięto częściowe zależności funkcyjne - wszystkie atrybuty nie będące kluczem zależą od całego klucza głównego
- **3NF**: Usunięto zależności przechodnie - atrybuty nie zależą od innych atrybutów nie będących kluczem

**Wyjątek od 3NF:** Pole `tag_list` w tabeli Company zostało celowo pozostawione jako VARCHAR z listą tagów oddzielonych przecinkami. Decyzja ta wynika z:

- Rzadkiego użycia do wyszukiwania
- Uproszczenia struktury bazy
- Zgodności z formatem danych źródłowych

## 5.2 Decyzje projektowe

### 5.2.1 Daty jako osobne kolumny

Daty (np. founded_year, founded_month, founded_day) zostały rozdzielone na osobne kolumny INT zamiast typu DATE, ponieważ:

- Dane źródłowe często zawierają niepełne daty (tylko rok lub rok i miesiąc)
- Typ DATE wymaga kompletnej daty
- Rozdzielenie umożliwia elastyczne zapytania (np. firmy założone w danym roku)

### 5.2.2 Nullable foreign keys

Niektóre klucze obce dopuszczają wartości NULL, ponieważ:

- Inwestor może być osobą, organizacją finansową LUB firmą
- Konkurent może nie istnieć jako osobna firma w bazie
- Firma przejmująca może nie być w naszym zbiorze danych

### 5.2.3 ON DELETE CASCADE

Kaskadowe usuwanie zostało włączone dla relacji:

- FundingRound → Investment
- Company → Product, Office, FundingRound, Milestone, etc.

## 5.3 Możliwe rozszerzenia

1. Dodanie pełnotekstowego wyszukiwania (Full-Text Search) dla opisu i overview
2. Implementacja historii zmian dla wszystkich tabel (temporal tables)
3. Partycjonowanie dużych tabel według roku
4. Dodanie warstwy API (REST API) do komunikacji z bazą
5. Integracja z narzędziami BI (Power BI, Tableau)

## 5.4 Podsumowanie

Projekt obejmuje:

- 18 tabel relacyjnych
- 7 indeksów niesklastrowanych
- 5 procedur składowanych
- 6 funkcji (3 skalarne, 3 tabelaryczne)
- 6 widoków
- 4 triggery
- 5 ról i 5 użytkowników contained
- Tabela audytu dla śledzenia zmian

Baza danych jest gotowa do użycia produkcyjnego i może być rozszerzana według potrzeb biznesowych.