System bazy danych dla Zakładu Zoologicznego na Uniwersytecie

Cel projektu

Projekt zakłada stworzenie bazy danych wspierającej działalność Zakładu Zoologicznego na Uniwersytecie. System gromadzi i zarządza informacjami dotyczącymi prowadzonych badań, finansowania, pracowników, wykorzystywanych materiałów.

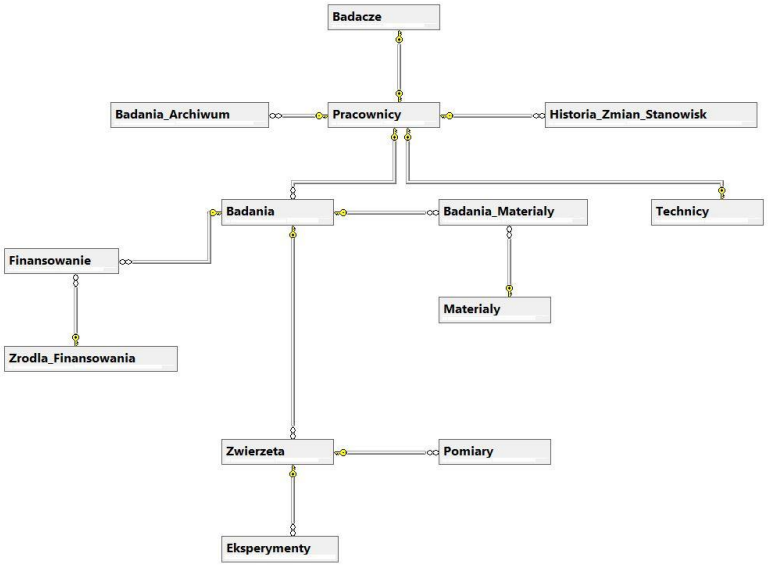
Zakres funkcjonalny

Baza danych obejmuje następujące kluczowe obszary:

1. **Zarządzanie badaniami**
    - Rejestrowanie badań prowadzonych przez pracowników zakładu.
    - Przechowywanie informacji o temacie, budżecie oraz terminach realizacji.
    - Monitorowanie kosztów zużycia materiałów.
2. **Zarządzanie finansowaniem**
    - Ewidencja źródeł finansowania badań i hodowli zwierząt.
    - Śledzenie wydatków związanych z materiałami oraz wynagrodzeniami pracowników.
3. **Obsługa pracowników**
    - Przechowywanie danych o zatrudnionych osobach, ich stanowiskach i historii zatrudnienia.
    - Mechanizmy zapobiegające duplikacji danych osobowych.
4. **Zarządzanie magazynem**
    - Rejestracja materiałów wykorzystywanych w badaniach i ich ilości.
    - Zapobieganie wprowadzaniu ujemnych stanów magazynowych.
5. **Archiwizacja i pielęgnacja danych**
    - Automatyczne przenoszenie zakończonych badań do archiwum.
    - Regularna konserwacja bazy, w tym backupy

Makar Dorohuntsev

Diagram ER



Główne encje i relacje

I. **Pracownicy** prowadzą **badania** (1:N).

II. **Badania** używają **materiałów** (N:M, tabela Badania_Materialy).

III. **Badania** są finansowane przez **źródła finansowania** (1:N).

IV. **Badania** obejmują **zwierzęta** (N:M).

V. **Zwierzęta** poddawane są **eksperymentom** i **pomiarom** (N:M).

VI. **Pracownicy** zmieniają stanowiska, co jest zapisywane w **historii zmian stanowisk**(1:N).

VII. **Badania** po zakończeniu trafiają do **archiwum** (1:1).

Wyzwalacze

Podwojne_Zatrudnienie

- Zapobiega podwójnemu zatrudnieniu tej samej osoby w Pracownicy, sprawdza, czy dana osoba już istnieje w tabeli Pracownicy.

Ujemna_Ilosc_Materialow

- Zapewnia, że ilość materiałów w tabeli Materialy nie może spaść poniżej zera. W przypadku próby zmiany na wartość ujemną transakcja jest wycofywana

Zmien_Historie_Stanowisk

- Automatycznie zapisuje historię zmian stanowisk pracowników w tabeli Historia_Zmian_Stanowisk.

Max_Budzet_Badania

- Kontroluje wydatki na badania i jeśli suma kosztów materiałów dla badania przekracza jego budżet, zmiana zostaje zablokowana.

Dodaj_Date_Konca_Stanowiska

- Automatycznie uzupełnia datę zakończenia poprzedniego stanowiska w Historia_Zmian_Stanowisk.

Procedury składowane

Dodaj_Nowego_Pracownika

- Dodaje nowego pracownika do tabeli Pracownicy i sprawdza, czy nie występuje podwójne zatrudnienie.

Zmien_Stanowisko_Pracownika

- Służy do zmiany stanowiska pracowników

Zwieksz_Budzet_Badania

- Pozwala na zwiększenie budżetu dla danego badania.

Raport_Wydatkow_Materialow

- Generuje raport zawierający temat i łączną kwotę wydaną badanie.

Usun_Nieaktywnego_Pracownika

- Sprawdza, czy pracownik istnieje w bazie.
- Sprawdza, czy nie ma aktywnych badań, do których jest przypisany.
- Usuwa pracownika tylko wtedy, gdy nie ma aktywnych badań.

Typowe zapytania do bazy danych:

- Ranking najdroższych badań

```
SELECT b.Nazwa, SUM(m.Cena * bm.Ilosc) AS Koszt_Badania
FROM Badania b
JOIN Badania_Materialy bm ON b.Badanie_Id = bm.Badanie_Id
JOIN Materialy m ON bm.Material_Id = m.Material_Id
GROUP BY b.Nazwa
ORDER BY Koszt_Badania DESC;
```

- Najnowsze badania

```
SELECT TOP 5 Nazwa, Data_Rozpoczecia, Data_Zakonczenia
FROM Badania
ORDER BY Data_Rozpoczecia DESC;
```

- Eksperymenty przeprowadzane na zwierzętach

```
SELECT z.Nazwa_Gatunku, e.Nazwa_Eksperymentu, e.Data_Wykonania
FROM Eksperymenty e
JOIN Zwierzeta z ON e.Zwierze_Id = z.Zwierze_Id
ORDER BY e.Data_Wykonania DESC;
```

- Całkowity budżet badań według roku i źródła finansowania

```
SELECT YEAR(f.Data_Przyznania) AS Rok,
    z.Nazwa_Zrodla,
    SUM(f.Kwota) AS Suma_Finansowania
FROM Finansowanie f
JOIN Zrodla_Finansowania z ON f.Zrodlo_Id = z.Zrodlo_Id
GROUP BY YEAR(f.Data_Przyznania), z.Nazwa_Zrodla
ORDER BY Rok DESC, Suma_Finansowania DESC;
```

- Lista aktywnych badań i ich kierowników

```
SELECT b.Badanie_Id, b.Nazwa, p.Imie, p.Nazwisko, b.Data_Rozpoczecia,
b.Data_Zakonczenia
FROM Badania b
JOIN Pracownicy p ON b. Pracownik_Prowadzacy_Id = p.Pracownik_Id
WHERE b.Data_Zakonczenia IS NULL;
```

Część skryptowa

Tabele:

```
CREATE TABLE Pracownicy (
    Pracownik_Id INT IDENTITY(1,1) PRIMARY KEY,
    Imie NVARCHAR(50) NOT NULL,
    Nazwisko NVARCHAR(50) NOT NULL,
    Data_Zatrudnienia DATE NOT NULL,
    Email NVARCHAR(100) UNIQUE NOT NULL,
    Typ_Pracownika NVARCHAR(50) NOT NULL CHECK (Typ_Pracownika IN ('Badacz', 'Technik'))
);
CREATE TABLE Badacze (
    Pracownik_Id INT PRIMARY KEY,
    Specjalizacja NVARCHAR(100) NOT NULL,
    FOREIGN KEY (Pracownik_Id) REFERENCES Pracownicy(Pracownik_Id) ON DELETE CASCADE
);
CREATE TABLE Technicy (
    Pracownik_Id INT PRIMARY KEY,
    Obszar_Pracy NVARCHAR(100) NOT NULL,
```

```
    FOREIGN KEY (Pracownik_Id) REFERENCES Pracownicy(Pracownik_Id) ON DELETE CASCADE
);
CREATE TABLE Badania (
    Badanie_Id INT IDENTITY(1,1) PRIMARY KEY,
    Temat NVARCHAR(200) NOT NULL,
    Data_Rozpoczecia DATE NOT NULL,
    Data_Zakonczenia DATE,
    Pracownik_Prowadzacy_Id INT NOT NULL,
    Opis TEXT,
    Budzet_Calkowity DECIMAL(10,2),
    Koszt_Materialow DECIMAL(10,2),
    FOREIGN KEY (Pracownik_Prowadzacy_Id) REFERENCES Pracownicy(Pracownik_Id)
);
CREATE TABLE Zwierzeta (
    Zwierze_Id INT IDENTITY(1,1) PRIMARY KEY,
    Gatunek NVARCHAR(100) NOT NULL,
    Imie NVARCHAR(50),
    Data_Urodzenia DATE,
    Waga DECIMAL(5,2),
    Badanie_Id INT,
    FOREIGN KEY (Badanie_Id) REFERENCES Badania(Badanie_Id)
);
CREATE TABLE Pomiary (
    Pomiar_Id INT IDENTITY(1,1) PRIMARY KEY,
    Zwierze_Id INT NOT NULL,
    Data_Pomiaru DATE NOT NULL,
    Waga DECIMAL(5,2),
    Temperatura DECIMAL(4,2),
    FOREIGN KEY (Zwierze_Id) REFERENCES Zwierzeta(Zwierze_Id) ON DELETE CASCADE
);
CREATE TABLE Eksperymenty (
    Eksperyment_Id INT IDENTITY(1,1) PRIMARY KEY,
    Nazwa NVARCHAR(200) NOT NULL,
    Opis TEXT,
```

```
    Data_Rozpoczecia DATE NOT NULL,
    Data_Zakonczenia DATE,
    Zwierze_Id INT,
    FOREIGN KEY (Zwierze_Id) REFERENCES Zwierzeta(Zwierze_Id) ON DELETE CASCADE
);
CREATE TABLE Materialy (
    Material_Id INT IDENTITY(1,1) PRIMARY KEY,
    Nazwa NVARCHAR(100) NOT NULL,
    Typ NVARCHAR(50),
    Ilosc INT NOT NULL,
    Data_Zakupu DATE,
    Cena DECIMAL(10,2)
);

CREATE TABLE Badania_Materialy (
    Badanie_Id INT,
    Material_Id INT,
    Ilosc_Wykorzystana INT NOT NULL,
    PRIMARY KEY (Badanie_Id, Material_Id),
    FOREIGN KEY (Badanie_Id) REFERENCES Badania(Badanie_Id),
    FOREIGN KEY (Material_Id) REFERENCES Materialy(Material_Id)
);
CREATE TABLE Zrodla_Finansowania (
    Zrodlo_Id INT IDENTITY(1,1) PRIMARY KEY,
    Nazwa NVARCHAR(100) NOT NULL,
    Typ_Zrodla NVARCHAR(50) CHECK (Typ_Zrodla IN ('Grant', 'Dotacja', 'Inwestycja', 'Budzet
Wewnetrzny'))
);

CREATE TABLE Finansowanie (
    Finansowanie_Id INT IDENTITY(1,1) PRIMARY KEY,
    Zrodlo_Id INT FOREIGN KEY REFERENCES Zrodla_Finansowania(Zrodlo_Id),
    Kwota DECIMAL(10,2),
    Data_Przyznania DATE,
    Badanie_Id INT,
```

```
    FOREIGN KEY (Badanie_Id) REFERENCES Badania(Badanie_Id)
);

CREATE TABLE Historia_Zmian_Stanowisk (
    Historia_Id INT IDENTITY(1,1) PRIMARY KEY,
    Pracownik_Id INT,
    Stare_Stanowisko NVARCHAR(50),
    Nowe_Stanowisko NVARCHAR(50),
    Data_Zmiany DATE NOT NULL,
        Data_Konca_Stanowiska DATE NULL,
    FOREIGN KEY (Pracownik_Id) REFERENCES Pracownicy(Pracownik_Id)
);

CREATE TABLE Badania_Archiwum (
    Badania_Id INT PRIMARY KEY,
    Temat_Badania NVARCHAR(255) NOT NULL,
    Data_Rozpoczecia DATE NOT NULL,
    Data_Zakonczenia DATE NOT NULL,
    Pracownik_Prowadzacy_Id INT,
    Opis_Badania NVARCHAR(MAX),
    Budzet_Calkowity DECIMAL(18,2),
    Koszt_Materialow DECIMAL(18,2),
    Data_Archiwizacji DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (Pracownik_Prowadzacy_Id) REFERENCES Pracownicy(Pracownik_Id)
);
```

Widoki i funkcje:

```
GO
CREATE VIEW Aktywni_Pracownicy AS
SELECT Pracownik_Id, Imie, Nazwisko, Typ_Pracownika, Data_Zatrudnienia, Email
FROM Pracownicy
WHERE Data_Zatrudnienia IS NOT NULL;

GO
CREATE VIEW Widok_Historii_Zmian_Stanowisk AS
```

```
SELECT h.Pracownik_Id, p.Imie, p.Nazwisko, h.Stare_Stanowisko, h.Nowe_Stanowisko,
h.Data_Zmiany
FROM Historia_Zmian_Stanowisk h
JOIN Pracownicy p ON h.Pracownik_Id = p.Pracownik_Id;

GO
CREATE VIEW Widok_Zuzycia_Materialow AS
SELECT m.Nazwa, SUM(bm.Ilosc_Wykorzystana) AS Ilosc_Zuzyta
FROM Materialy m
JOIN Badania_Materialy bm ON m.Material_Id = bm.Material_Id
GROUP BY m.Nazwa;

-- Zaawansowane badanie czyli badanie z budżetem wykorzystanym co najmniej na 85%
GO
CREATE VIEW Widok_Badan_Zaawansowanych AS
SELECT b.Badanie_Id, b.Temat,
    SUM(m.Cena * bm.Ilosc_Wykorzystana) AS Wydatki,
    b.Budzet_Calkowity,
    (SUM(m.Cena * bm.Ilosc_Wykorzystana) / b.Budzet_Calkowity) * 100 AS Procent_Wykorzystania
FROM Badania b
JOIN Badania_Materialy bm ON b.Badanie_Id = bm.Badanie_Id
JOIN Materialy m ON bm.Material_Id = m.Material_Id
GROUP BY b.Badanie_Id, b.Temat, b.Budzet_Calkowity
HAVING (SUM(m.Cena * bm.Ilosc_Wykorzystana) / b.Budzet_Calkowity) * 100 > 84;

GO
CREATE VIEW Widok_Najdrozszych_Badan AS
SELECT TOP 10 Badanie_Id, Temat, Budzet_Calkowity
FROM Badania
ORDER BY Budzet_Calkowity DESC;

GO
CREATE VIEW Widok_Pracownikow_Bez_Badan AS
SELECT p.Pracownik_Id, p.Imie, p.Nazwisko, p.Email
FROM Pracownicy p
```

```
LEFT JOIN Badania b ON p.Pracownik_Id = b.Pracownik_Prowadzacy_Id
WHERE b.Badanie_Id IS NULL;

GO
CREATE FUNCTION Suma_Wydatkow_Na_Badanie (@Badanie_Id INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @Suma DECIMAL(10,2);
    SELECT @Suma = SUM(m.Cena * bm.Ilosc_Wykorzystana)
    FROM Materialy m
    JOIN Badania_Materialy bm ON m.Material_Id = bm.Material_Id
    WHERE bm.Badanie_Id = @Badanie_Id;
    RETURN @Suma;
END;

GO
CREATE FUNCTION Czy_Pracownik_Prowadzi_Badanie (@Pracownik_Id INT)
RETURNS BIT
AS
BEGIN
    DECLARE @Czy_Prowadzi BIT;
    IF EXISTS (SELECT 1 FROM Badania WHERE Pracownik_Prowadzacy_Id = @Pracownik_Id)
        SET @Czy_Prowadzi = 1;
    ELSE
        SET @Czy_Prowadzi = 0;
    RETURN @Czy_Prowadzi;
END;

GO
CREATE FUNCTION Oblicz_Procent_Wykorzystania_Budzetu (@Badanie_Id INT)
RETURNS DECIMAL(5,2)
AS
BEGIN
```

```
    DECLARE @Budzet DECIMAL(10,2), @Wydane DECIMAL(10,2);
    SELECT @Budzet = Budzet_Calkowity FROM Badania WHERE Badanie_Id = @Badanie_Id;
    SET @Wydane = dbo.Suma_Wydatkow_Na_Badanie(@Badanie_Id);
    RETURN (@Wydane / @Budzet) * 100;
END;

GO
CREATE FUNCTION Najdrozszy_Material_W_Badaniu (@Badanie_Id INT)
RETURNS NVARCHAR(100)
AS
BEGIN
    DECLARE @Material NVARCHAR(100);
    SELECT TOP 1 @Material = m.Nazwa
    FROM Materialy m
    JOIN Badania_Materialy bm ON m.Material_Id = bm.Material_Id
    WHERE bm.Badanie_Id = @Badanie_Id
    ORDER BY (m.Cena * bm.Ilosc_Wykorzystana) DESC;
    RETURN @Material;
END;

GO
CREATE FUNCTION Procent_Zuzycia_Materialu (@Material_Id INT)
RETURNS DECIMAL(5,2)
AS
BEGIN
    DECLARE @Zuzyto INT, @Dostepne INT;
    SELECT @Zuzyto = COALESCE(SUM(Ilosc_Wykorzystana), 0) FROM Badania_Materialy WHERE
Material_Id = @Material_Id;
    SELECT @Dostepne = Ilosc FROM Materialy WHERE Material_Id = @Material_Id;
    RETURN (@Zuzyto * 100.0 / @Dostepne);
END;

GO
CREATE FUNCTION Oblicz_Calkowite_Wydatki_Pracownika (@Pracownik_Id INT)
RETURNS DECIMAL(10,2)
```

```
AS
BEGIN
    DECLARE @Suma DECIMAL(10,2);
    SELECT @Suma = COALESCE(SUM(m.Cena * bm.Ilosc_Wykorzystana), 0)
    FROM Badania b
    JOIN Badania_Materialy bm ON b.Badanie_Id = bm.Badanie_Id
    JOIN Materialy m ON bm.Material_Id = m.Material_Id
    WHERE b.Pracownik_Prowadzacy_Id = @Pracownik_Id;
    RETURN @Suma;
END;
```

Wyzwalacze:

```
CREATE TRIGGER Podwojne_Zatrudnienie
ON Pracownicy INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1 FROM Pracownicy
        WHERE Imie IN (SELECT Imie FROM inserted)
        AND Nazwisko IN (SELECT Nazwisko FROM inserted)
        AND Data_Zatrudnienia IN (SELECT Data_Zatrudnienia FROM inserted)
    )
    BEGIN
        RAISERROR('Pracownik o takich danych już istnieje!', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO Pracownicy (Imie, Nazwisko, Data_Zatrudnienia, Email, Typ_Pracownika)
        SELECT Imie, Nazwisko, Data_Zatrudnienia, Email, Typ_Pracownika FROM inserted;
    END
END;

GO
CREATE TRIGGER Ujemna_Ilosc_Materialow
```

```
ON Materialy
INSTEAD OF INSERT, UPDATE
AS
BEGIN
    IF EXISTS (SELECT 1 FROM inserted WHERE Ilosc < 0)
    BEGIN
        RAISERROR('Ilość materiałów nie może być ujemna!', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO Materialy (Material_Id, Nazwa, Ilosc, Cena)
        SELECT Material_Id, Nazwa, Ilosc, Cena FROM inserted;
    END
END;



GO
CREATE TRIGGER Zmien_Historie_Stanowisk
ON Pracownicy AFTER UPDATE
AS
BEGIN
    INSERT INTO Historia_Zmian_Stanowisk (Pracownik_Id, Stare_Stanowisko, Nowe_Stanowisko,
Data_Zmiany)
    SELECT d.Pracownik_Id, d.Typ_Pracownika, i.Typ_Pracownika, GETDATE()
    FROM deleted d INNER JOIN inserted i ON d.Pracownik_Id = i.Pracownik_Id
    WHERE d.Typ_Pracownika <> i.Typ_Pracownika;
END;


GO
CREATE TRIGGER Max_Budzet_Badania
ON Badania_Materialy AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
```

```
        SELECT 1 FROM Badania B
        JOIN (SELECT Badanie_Id, SUM(Ilosc_Wykorzystana * Cena) AS Koszt
            FROM Badania_Materialy BM
            JOIN Materialy M ON BM.Material_Id = M.Material_Id
            GROUP BY Badanie_Id) Koszty
        ON B.Badanie_Id = Koszty.Badanie_Id
        WHERE Koszty.Koszt > B.Budzet_Calkowity
    )
    BEGIN
        RAISERROR('Koszt materialów przekracza budżet badania!', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;

GO
CREATE TRIGGER Dodaj_Date_Konca_Stanowiska
ON Pracownicy AFTER UPDATE
AS
BEGIN
    UPDATE Historia_Zmian_Stanowisk
    SET Data_Konca_Stanowiska = GETDATE()
    WHERE Pracownik_Id IN (SELECT Pracownik_Id FROM deleted)
    AND Data_Konca_Stanowiska IS NULL;
END;
```

Procedury:

```
CREATE PROCEDURE Dodaj_Nowego_Pracownika
    @Imie NVARCHAR(50),
    @Nazwisko NVARCHAR(50),
    @Data_Zatrudnienia DATE,
    @Email NVARCHAR(100),
    @Typ_Pracownika NVARCHAR(50),
    @Specjalizacja NVARCHAR(100) = NULL
AS
BEGIN
```

```
    DECLARE @NowyId INT;

    INSERT INTO Pracownicy (Imie, Nazwisko, Data_Zatrudnienia, Email, Typ_Pracownika)
    VALUES (@Imie, @Nazwisko, @Data_Zatrudnienia, @Email, @Typ_Pracownika);

    SET @NowyId = SCOPE_IDENTITY();

    IF @Typ_Pracownika = 'Badacz'
        INSERT INTO Badacze (Pracownik_Id, Specjalizacja) VALUES (@NowyId, @Specjalizacja);
    ELSE IF @Typ_Pracownika = 'Technik'
        INSERT INTO Technicy (Pracownik_Id, Obszar_Pracy) VALUES (@NowyId, @Specjalizacja);
END;

GO
CREATE PROCEDURE Zmien_Stanowisko_Pracownika
    @Pracownik_Id INT,
    @NoweStanowisko NVARCHAR(50)
AS
BEGIN
    UPDATE Pracownicy SET Typ_Pracownika = @NoweStanowisko WHERE Pracownik_Id =
@Pracownik_Id;
END;

GO
CREATE PROCEDURE Zwieksz_Budzet_Badania
    @Badanie_Id INT,
    @Kwota DECIMAL(10,2)
AS
BEGIN
    UPDATE Badania SET Budzet_Calkowity = Budzet_Calkowity + @Kwota WHERE Badanie_Id =
@Badanie_Id;
END;

GO
CREATE PROCEDURE RaportWydatkowNaMaterialy
```

```
AS
BEGIN
    SELECT B.Temat, SUM(BM.Ilosc_Wykorzystana * M.Cena) AS Koszt_Calkowity
    FROM Badania_Materialy BM
    JOIN Materialy M ON BM.Material_Id = M.Material_Id
    JOIN Badania B ON BM.Badanie_Id = B.Badanie_Id
    GROUP BY B.Temat;
END;

GO
CREATE PROCEDURE Usun_Nieaktywnego_Pracownika
    @Pracownik_Id INT
AS
BEGIN
    DECLARE @Liczba_Aktywnych_Badan INT;

    -- Sprawdzenie, czy pracownik istnieje w bazie
    IF NOT EXISTS (SELECT 1 FROM Pracownicy WHERE Pracownik_Id = @Pracownik_Id)
    BEGIN
        RAISERROR('Pracownik o podanym ID nie istnieje!', 16, 1);
        RETURN;
    END;

    -- Sprawdzenie, czy pracownik jest przypisany do aktywnych badań
    SELECT @Liczba_Aktywnych_Badan = COUNT(*)
    FROM Badania
    WHERE Pracownik_Prowadzacy_Id = @Pracownik_Id AND Data_Zakonczenia IS NULL;

    IF @Liczba_Aktywnych_Badan > 0
    BEGIN
        RAISERROR('Nie można usunąć pracownika, ponieważ jest przypisany do aktywnych badań!',
16, 1);
        RETURN;
    END;
```

```
    -- Usunięcie pracownika z tabeli Pracownicy
    DELETE FROM Pracownicy WHERE Pracownik_Id = @Pracownik_Id;

    PRINT 'Pracownik został pomyślnie usunięty i zarchiwizowany.';
END;
```

Pielęgnacja:

```
-- Pielęgnacja bazy danych
-- Archiwacja badań do archiwum starszych od 3 lat
GO
CREATE PROCEDURE Archiwizuj_Badania
AS
BEGIN
    INSERT INTO Badania_Archiwum (Badania_Id, Temat_Badania, Data_Rozpoczecia,
Data_Zakonczenia)
    SELECT Badanie_Id, Temat, Data_Rozpoczecia, Data_Zakonczenia
    FROM Badania
    WHERE Data_Zakonczenia < DATEADD(YEAR, -3, GETDATE());

    DELETE FROM Badania
    WHERE Data_Zakonczenia < DATEADD(YEAR, -3, GETDATE());
END;

-- Backup do zachowania danych zautomatyzowany za pomocą SQL Server Agent

BACKUP DATABASE Test_Zaklad
TO DISK = 'C:\Backups\Test_Zaklad_Full.bak'
WITH FORMAT, MEDIANAME = 'SQLServerBackups', NAME = 'Full Backup Test_Zaklad';

-- Backup dziennika bazy danych
BACKUP LOG Test_Zaklad
TO DISK = 'C:\Backups\Test_Zaklad_Log.trn';
```