Przykładowy projekt

lp.	zadanie do wykonania
1	Określenie celu i wymagań
2	Definicja DZE
3	Transformacja DZE do modelu relacyjnego Normalizacja
4	Definicja zasad poprawności danych
5	Definicja schematu bazy danych z implementacją deklaratywnych metod sprawdzania poprawności danych
6	Implementacja niedeklaratywnych mechanizmów sprawdzania poprawności danych
7	Wprowadzenie przykładowych danych
8	Definicja i implementacja zasad ochrony bazy danych

1. Cel i wymagania

Cel:

Celem systemu jest ewidencja urlopów w sposób który zapewni skuteczną organizację i zarządzanie urlopami danego pracownika, tak aby zgodnie z prawem wystawić urlopy.

Wymagania:

Wymaganie	Opis wymagania	Źródło	Waga	Miara	Uwagi
Przyznanie pracownikom puli urlopu wypoczynkowego ze względu na staż pracy	Przyznanie pracownikom urlopu wypoczynkowego proporcjonalnie do ich stażu pracy w firmie.	Kodeks pracy	10	Polityka firmy	-
Kontrola ilości dni urlopów przysługujących pracownikowi	System ma monitorować i wyświetlać ilość dni urlopowych, które przysługują pracownikowi w danym roku kalendarzowym.	Kodeks pracy	9	Liczba dni	-
Kontrola ilości dni wykorzystanych i niewykorzystanych, w tym uniemożliwienie wnioskowania ponad ustawowy limit	System ma śledzić ilość dni urlopowych wykorzystanych przez pracownika oraz ewentualny pozostały limit.	Kodeks pracy	9	Liczba dni	-
Kontrola zaległych urlopów – ostrzeganie o niewykorzystanym urlopie we wskazanym w kodeksie pracy terminie	System ma ostrzegać pracowników o niewykorzystanych urlopach i terminach, w których powinni zostać wykorzystane.	Kodeks pracy	8	Alerty	-
Prowadzenie planów urlopów (harmonogramowanie urlopu)	Możliwość tworzenia planów urlopów dla pracowników, umożliwiających zarządzanie dostępnością zasobów ludzkich w firmie.	Kodeks pracy	7	Kalendarz	-
Składowanie wniosków urlopowych z uwzględnieniem okresu urlopu (data rozpoczęcia, data zakończenia) wraz ze statusem akceptacji przełożonego	System ma przechowywać wnioski urlopowe pracowników, zawierające daty rozpoczęcia i zakończenia urlopu	Kodeks pracy	9	Baza danych	-

	oraz status akceptacji przez przełożonego.				
Składowanie struktury organizacyjnej firmy (pracownicy i przełożeni)	System ma zawierać informacje o strukturze organizacyjnej firmy, obejmującej pracowników oraz ich przełożonych.	Kodeks pracy	8	Baza danych	-
Uwzględnienie dni ustawowo wolnych od pracy, w tym weekendów	System ma uwzględniać dni ustawowo wolne od pracy, takie jak święta i weekendy, podczas planowania urlopów.	Kodeks pracy	7	Kalendarz	-
Rozliczanie należnego urlopu w przypadku rozwiązania stosunku pracy	System ma automatycznie rozliczać należny pracownikowi urlop za niewykorzystane dni w przypadku rozwiązania stosunku pracy.	Kodeks pracy	9	Liczba dni	-
Kontrola dostępu – tylko bezpośredni przełożony może akceptować wniosek pracownika	System ma zapewnić kontrolę dostępu, aby tylko bezpośredni przełożeni mogli akceptować wnioski urlopowe pracowników.	Kodeks pracy	10	Bezpieczeństwo	-
Możliwość kontroli innych rodzajów urlopów takich jak (urlop okolicznościowy, urlop bezpłatny, opieka nad zdrowym dzieckiem)	System ma umożliwiać kontrolę różnych rodzajów urlopów, takich jak urlop okolicznościowy, urlop bezpłatny czy urlop na opiekę nad zdrowym dzieckiem.	Kodeks pracy	8	Baza danych	-
Raport jest generowany raz w miesiącu	System ma generować raporty dotyczące urlopów raz w miesiącu, zawierające różne statystyki i analizy.	Kodeks pracy	7	Raport	-

Klasa: Wydział

Atrybuty:

- ID Wydziału: Unikalny identyfikator wydziału (klucz główny).
- Nazwa Wydziału: Nazwa wydziału.
- Kierownik: Informacja o kierowniku wydziału.

Metody:

- DodajKierownika(kierownik): Dodaje nowego kierownika wydziału.
- AktualizujKierownika(nowy_kierownik): Aktualizuje informacje o kierowniku wydziału.
- UsunKierownika(): Usuwa informacje o kierowniku wydziału.

Klasa: Pracownik

Atrybuty:

- ID pracownika: Unikalny identyfikator pracownika (klucz główny).
- Imię: Imię pracownika.
- Nazwisko: Nazwisko pracownika.
- PESEL: Numer PESEL pracownika.
- Adres: Adres zamieszkania pracownika.
- Data Urodzenia: Data urodzenia pracownika.
- Wykształcenie: Poziom wykształcenia pracownika.
- Data Zatrudnienia: Data zatrudnienia pracownika.
- Staż Pracy: Okres, przez który pracownik jest zatrudniony.
- Wykorzystany Urlop: Informacja o wykorzystanym urlopie.
- Data Zwolnienia: Data zakończenia zatrudnienia pracownika (jeśli występuje).
- Status Zatrudnienia: Aktualny status zatrudnienia pracownika.
- Przełożony: Referencja do przełożonego pracownika.

Metody:

- DodajPracownika(dane_pracownika): Dodaje nowego pracownika do bazy danych.
- AktualizujDanePracownika(nowe_dane): Aktualizuje dane istniejącego pracownika.
- ZwolnijPracownika(data_zwolnienia): Zwalnia pracownika z pracy, ustawiając odpowiedni status zatrudnienia i datę zwolnienia.
- ZmienPrzełożonego(nowy_przełożony): Aktualizuje informacje o przełożonym pracownika.
- ZłóżWniosekUrlopowy(dane_wniosku): Rejestruje wniosek urlopowy pracownika.

- AktualizujStatusUrlopu(id_urlopu, nowy_status): Aktualizuje status akceptacji wniosku o urlop.
- UsunPracownika(): Usuwa pracownika z bazy danych.

Klasa: UrlopNależnyWDanymRoku

Atrybuty:

- ID roku: Unikalny identyfikator roku (klucz główny).
- Nominał urlopu: Określa ilość dni urlopu, która przysługuje pracownikowi w danym roku.
- Ilość wykorzystanych dni: Liczba dni urlopu, które pracownik już wykorzystał w danym roku.

Klasa: Urlop

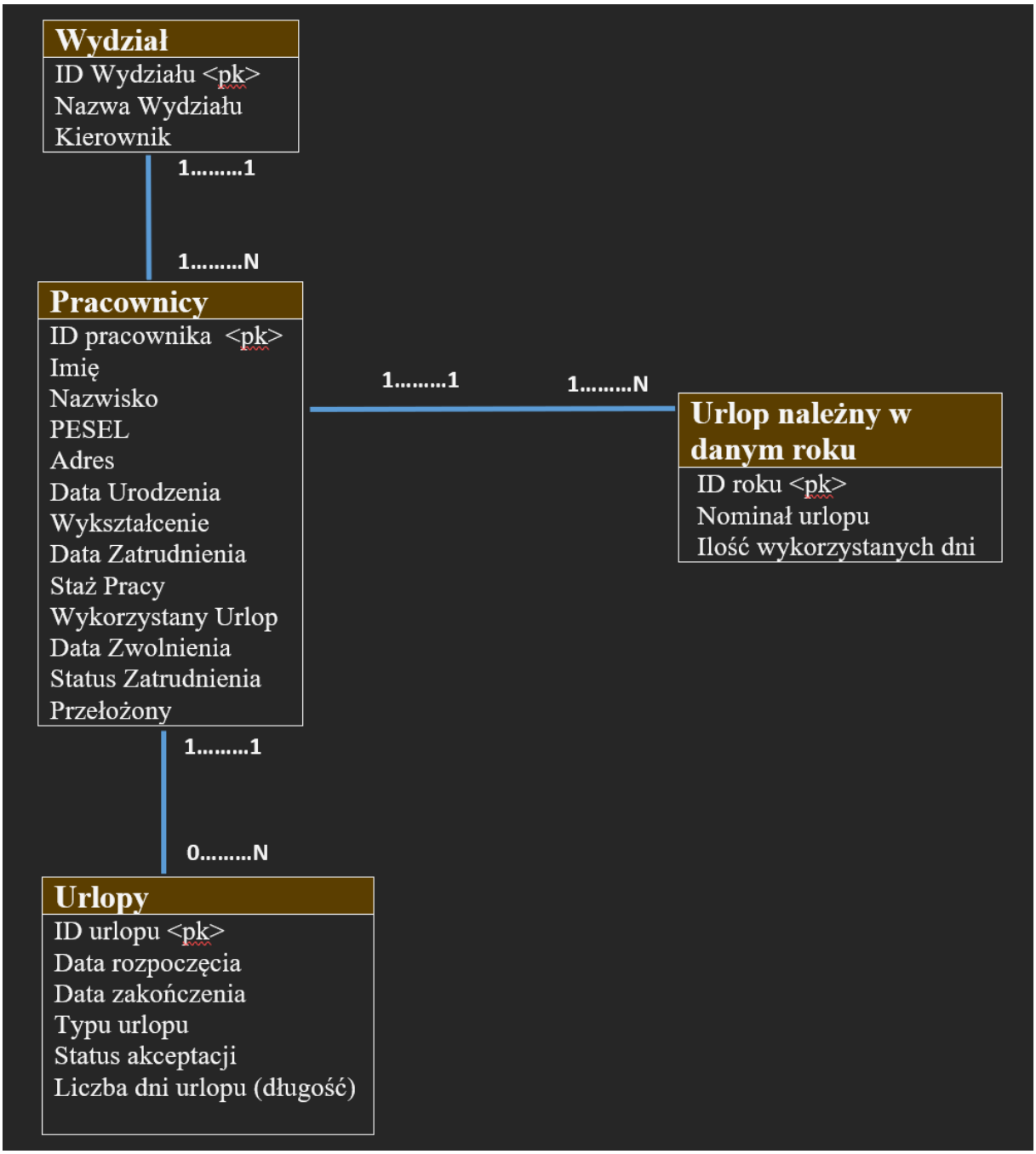
Atrybuty:

- ID urlopu: Unikalny identyfikator urlopu (klucz główny).
- Data rozpoczęcia: Data rozpoczęcia urlopu.
- Data zakończenia: Data zakończenia urlopu.
- Typ urlopu: Rodzaj urlopu (np. urlop wypoczynkowy, urlop okolicznościowy).
- Status akceptacji: Aktualny status akceptacji wniosku o urlop.
- Liczba dni urlopu (długość): Ilość dni trwania urlopu.

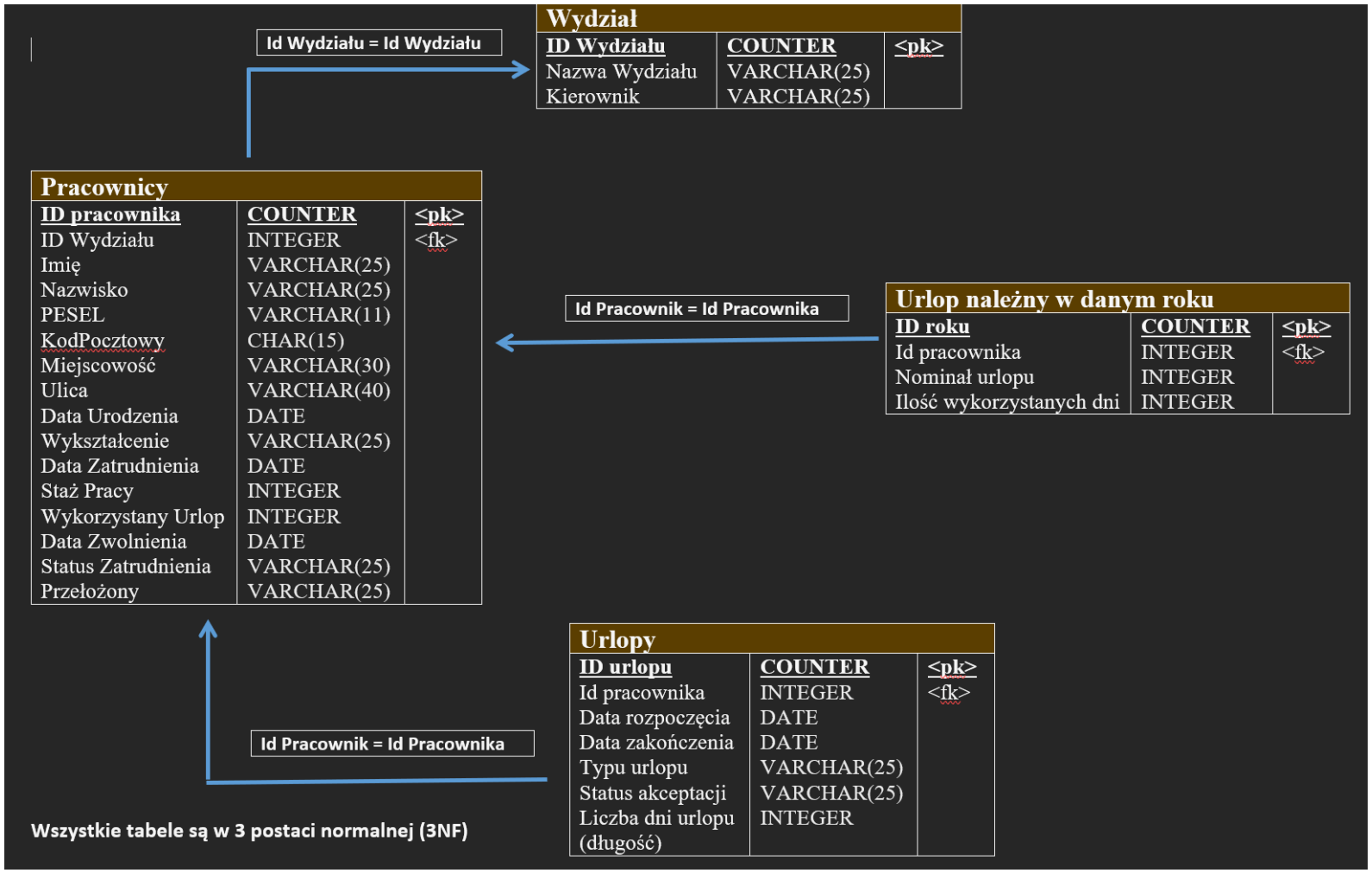
Metody:

- DodajUrlop(dane_urlopu): Dodaje nowy urlop do bazy danych.
- AktualizujDaneUrlopu(id_urlopu, nowe_dane): Aktualizuje dane istniejącego urlopu.
- UsunUrlop(): Usuwa informacje o urlopie z bazy danych.

2. Definicja DZE



3. Transformacja DZE do modelu relacyjnego - Normalizacja



4. Definicja zasad poprawności danych

Oto zakres danych, jakie muszą być wprowadzone w poszczególnych polach tabel:

Tabela "Wydział":

- ID Wydziału: liczby całkowite dodatnie (np. 1, 2, 3, ...)
- Nazwa Wydziału: tekst (np. "Wydział Księgowości")
- Kierownik: ID pracownika będącego kierownikiem tego wydziału

Tabela "Pracownicy":

- ID pracownika: liczby całkowite dodatnie (np. 1, 2, 3, ...)
- Imię: tekst (np. "Jan", "Anna")
- Nazwisko: tekst (np. "Kowalski", "Nowak")
- PESEL: numer PESEL pracownika, maksymalnie 11 znaków
- Data Urodzenia: data w formacie DD.MM.RRRR (np. 01.01.1990)
- Wykształcenie: tekst (np. "wyższe", "średnie")
- Data Zatrudnienia: data w formacie DD.MM.RRRR (np. 01.01.2020)
- Staż Pracy: liczba całkowita dodatnia (np. 2, 3, 4, ...)
- Wykorzystany Urlop: liczba całkowita dodatnia (np. 0, 5, 10, ...)
- Data Zwolnienia: data w formacie DD.MM.RRRR (jeśli pracownik został zwolniony)
- Status Zatrudnienia: tekst (np. "aktywny", "zwolniony")
- Przełożony: ID przełożonego (referencja do ID innego pracownika w tej samej tabeli)

Tabela "Urlop należny w danym roku":

- ID roku: unikalne ID dla każdego roku (np. 2021, 2022, ...)
- Nominał urlopu: liczba całkowita dodatnia (np. 20, 26)
- Ilość wykorzystanych dni: liczba całkowita nieujemna (np. 0, 5, 10)

Tabela "Urlopy":

- ID urlopu: unikalne ID dla każdego urlopu (np. 1, 2, 3, ...)
- Data rozpoczęcia: data w formacie DD.MM.RRRR
- Data zakończenia: data w formacie DD.MM.RRRR
- Typu urlopu: tekst (np. "urlop wypoczynkowy")
- Status akceptacji: tekst (np. "zaakceptowany")

- Liczba dni urlopu (długość): liczba całkowita dodatnia (np. 1, 5, 10)

Reguły poprawności pól

Tabela: Wydział

1. **Reguła poprawności pola dla ID Wydziału**:
    - ID Wydziału musi być unikalne dla każdego rekordu: UNIQUE
2. **Reguła poprawności pola dla Nazwa Wydziału**:
    - Pole Nazwa Wydziału nie może być puste: IS NOT NULL
    - Pole Nazwa Wydziału nie może zawierać znaków specjalnych: NOT LIKE '%[!@#$%^&*(),.?":{}|<>]%'
3. **Reguła poprawności pola dla Kierownik**:
    - Pole Kierownik nie może być puste: IS NOT NULL
    - Pole Kierownik musi być przypisane do istniejącego pracownika w tabeli Pracownicy: EXISTS(SELECT 1 FROM Pracownicy WHERE Pracownicy.ID_pracownika = Wydział.Kierownik)

Tabela: Pracownicy

1. **Reguła poprawności pola dla ID Pracownika**:
    - ID Pracownika musi być unikalne dla każdego rekordu: UNIQUE
    - ID pracownika nie może być puste: Is Not Null
2. **Reguła poprawności pola dla Imię**
    - Imię nie może być puste: Is Not Null
3. **Reguła poprawności pola dla Nazwisko**
    - Nazwisko nie może być puste: Is Not Null
4. **Reguła poprawności pola dla Miejscowość**
    - Miejscowość nie może być pusta: Is Not Null
5. **Reguła poprawności pola dla Ulica**
    - Ulica nie może być pusta: Is Not Null
6. **Reguła poprawności pola dla Data Urodzenia**:
    - Data Urodzenia nie może być pusta: IS NOT NULL

- Data Urodzenia musi być wcześniejsza niż bieżąca data: < CURRENT_DATE

7. **Reguła poprawności dla pola PESEL**:
    - Numer PESEL musi zawierać 11 cyfr: LEN([PESEL]) = 11
    - PESEL musi zawierać tylko cyfry: Is Numeric([PESEL])
8. **Reguła poprawności pola dla Data Zatrudnienia**:
    - Data Zatrudnienia nie może być pusta: IS NOT NULL
    - Data Zatrudnienia musi być wcześniejsza lub równa bieżącej dacie: <= CURRENT_DATE
9. **Reguła poprawności pola dla Data Zwolnienia**:
    - Data Zwolnienia, jeśli jest wprowadzona, musi być późniejsza niż Data Zatrudnienia: > [Data Zatrudnienia]
    - Data Zwolnienia nie może być wcześniejsza niż aktualna data: [Data Zwolnienia] > Now()
10. **Reguła poprawności pola dla Status Zatrudnienia**:
    - Status Zatrudnienia musi być jednym z ustalonych wartości, np. "Aktywny", "Nieaktywny": IN ('Aktywny', 'Nieaktywny')
11. **Reguła poprawności pola dla KodPocztowy**:
    - Kod pocztowy powinien być zgodny z formatem xx-xxx, gdzie x to cyfry: LIKE '##-###'
    - Kod pocztowy musi mieć dokładnie 5 znaków: Length([KodPocztowy]) = 5
12. **Reguła poprawności pola dla Staż Pracy**:
    - Staż pracy nie może być ujemny: >= 0
13. **Reguła poprawności pola dla Wykorzystany urlop**
    - Pole wykorzystany urlop musi być liczbą nieujemną: [Wykorzystany Urlop] >= 0
    - Pole wykorzystany urlop nie może być większe niż Nominał Urlopu
14. **Reguła poprawności dla pola Wykształcenie**:
    - Pole Wykształcenie musi zawierać jedno z ustalonych wykształceń, np. "Podstawowe", "Średnie", "Wyższe": IN ('Podstawowe', 'Średnie', 'Wyższe')

Tabela: Urlop należny w danym roku

1. **Reguła poprawności pola dla ID roku**:
    - ID roku musi być unikalne dla każdego rekordu: UNIQUE
    - ID roku nie może być puste: Is Not Null
2. **Reguła poprawności pola dla Nominał urlopu**:
    - Nominał urlopu musi być większy lub równy zero: >= 0
    - Pole nominał urlopu nie może być puste: Is Not Null
3. **Reguła poprawności pola dla Ilość wykorzystanych dni**:
    - Ilość wykorzystanych dni nie może być większa niż Nominał urlopu: <= [Nominał urlopu]
    - Ilość wykorzystanych dni musi być większy lub równy zero: >= 0

Tabela: Urlopy

1. **Reguła poprawności pola dla ID urlopu**:
    - ID urlopu musi być unikalne dla każdego rekordu: UNIQUE
2. **Reguła poprawności pola dla Liczba dni urlopu**:
    - Liczba dni urlopu nie może być ujemna: >= 0
    - Liczba dni urlopu nie może przekraczać maksymalnego limitu urlopu dla danego pracownika (Nominał urlopu)
3. **Reguła poprawności pola dla Data zakończenia**:
    - Data zakończenia musi być późniejsza niż data rozpoczęcia: > [Data rozpoczęcia]
4. **Reguła poprawności pola dla Daty rozpoczęcia urlopu**:
    - Nie zezwalaj na wprowadzanie dat z przeszłości: >=Date()
5. **Reguła poprawności pola dla Status akceptacji**:
    - Status akceptacji musi być jednym z ustalonych wartości, np. "Zaakceptowany", "Odrzucony", "Oczekujący": IN ('Zaakceptowany', 'Odrzucony', 'Oczekujący')\
6. **Reguła poprawności pola dla Typu urlopu**:
    - Typ urlopu musi być jednym z ustalonych wartości, np. "Urlop wypoczynkowy", "Urlop okolicznościowy", "Urlop bezpłatny": IN ('Urlop wypoczynkowy', 'Urlop okolicznościowy', 'Urlop bezpłatny')

5. Definicja schematu bazy danych z implementacją deklaratywnych metod sprawdzania poprawności danych

```
-- Tworzenie tabeli Wydział
CREATE TABLE Wydział (
    [ID Wydziału] INTEGER PRIMARY KEY,
    [Nazwa Wydziału] VARCHAR(25) NOT NULL,
    [Kierownik] VARCHAR(25) NOT NULL
       CONSTRAINT [IX_Wydział_ID] UNIQUE ([ID Wydziału]),
    CONSTRAINT [CK_Wydział_Nazwa] CHECK ([Nazwa Wydziału] IS NOT NULL AND [Nazwa
Wydziału] NOT LIKE '%[!@#$%^&*(),.?":{}|<>]%'),
 CONSTRAINT [CK_Wydział_Kierownik] CHECK (
        [Kierownik] IS NOT NULL ));

-- Tworzenie tabeli Pracownicy
CREATE TABLE Pracownicy (
    [ID pracownika] INTEGER PRIMARY KEY,
    [ID Wydziału] INTEGER NOT NULL,
    [Imię] VARCHAR(25) NOT NULL,
    [Nazwisko] VARCHAR(25) NOT NULL,
    [PESEL] VARCHAR(11) NOT NULL,
    [KodPocztowy] CHAR(15) NOT NULL,
    [Miejscowość] VARCHAR(30) NOT NULL,
    [Ulica] VARCHAR(40) NOT NULL,
    [Data Urodzenia] DATE NOT NULL CHECK ([Data Urodzenia] < GETDATE()),
    [Wykształcenie] NVARCHAR(20) CHECK ([Wykształcenie] IN ('Podstawowe', 'Średnie',
'Wyższe')),
    [Data Zatrudnienia] DATE NOT NULL,
    [Staż Pracy] INT NOT NULL CHECK ([Staż Pracy] >= 0),
    [Wykorzystany Urlop] INT NOT NULL CHECK ([Wykorzystany Urlop] >= 0),
    [Data Zwolnienia] DATE,
    [Status Zatrudnienia] NVARCHAR(20) CHECK ([Status Zatrudnienia] IN ('Aktywny',
'Nieaktywny')),
    [Przełożony] VARCHAR(40),
    CONSTRAINT [IX_Pracownicy_ID] UNIQUE ([ID Pracownika]),
    CONSTRAINT [CK_PESEL_Length] CHECK (LEN([PESEL]) = 11),
    CONSTRAINT [CK_PESEL_Numeric] CHECK (ISNUMERIC([PESEL]) = 1),
    CONSTRAINT [CK_KodPocztowy_Format] CHECK ([KodPocztowy] LIKE '__-___'),
    CONSTRAINT [CK_KodPocztowy_Length] CHECK (LEN([KodPocztowy]) = 6),
    CONSTRAINT [FK_Pracownicy_Wydział] FOREIGN KEY ([ID Wydziału]) REFERENCES Wydział
([ID Wydziału]),
    FOREIGN KEY ([ID Wydziału]) REFERENCES Wydział ([ID Wydziału])
);

-- Tworzenie tabeli Urlop należny w danym roku
CREATE TABLE UrlopNależny (
    [ID roku] INTEGER PRIMARY KEY,
    [Id pracownika] INTEGER NOT NULL,
    [Nominał urlopu] INTEGER NOT NULL,
    [Ilość wykorzystanych dni] INTEGER NOT NULL,
       CONSTRAINT [CK_UrlopNależny_IDRoku_Unikalność] UNIQUE ([ID roku]),
    CONSTRAINT [CK_UrlopNależny_IDRoku_Puste] CHECK ([ID roku] IS NOT NULL),
    CONSTRAINT [CK_UrlopNależny_NominałUrlopu_Nieujemny] CHECK ([Nominał urlopu] >=
0),
    CONSTRAINT [CK_UrlopNależny_NominałUrlopu_Puste] CHECK ([Nominał urlopu] IS NOT
NULL),
    CONSTRAINT [CK_UrlopNależny_IlośćWykorzystanychDni_WiększaLubRównaZero] CHECK
([Ilość wykorzystanych dni] >= 0),
    CONSTRAINT [CK_UrlopNależny_IlośćWykorzystanychDni_MniejszaNiżNominałUrlopu] CHECK
([Ilość wykorzystanych dni] <= [Nominał urlopu]),
```

```
    CONSTRAINT [FK_UrlopNależny_Pracownicy] FOREIGN KEY ([Id pracownika]) REFERENCES
Pracownicy ([ID Pracownika]),
    FOREIGN KEY ([Id pracownika]) REFERENCES Pracownicy ([ID pracownika])
);

-- Tworzenie tabeli Urlopy
CREATE TABLE Urlopy (
    [ID urlopu] INTEGER PRIMARY KEY,
    [Id pracownika] INTEGER NOT NULL,
    [Data rozpoczęcia] DATE NOT NULL CHECK ([Data rozpoczęcia] >= GETDATE()),
    [Data zakończenia] DATE NOT NULL,
    [Typu urlopu] VARCHAR(50) CHECK ([Typu urlopu] IN ('Urlop wypoczynkowy', 'Urlop
okolicznościowy', 'Urlop bezpłatny')),
    [Status akceptacji] VARCHAR(50) CHECK ([Status akceptacji] IN ('Zaakceptowany',
'Odrzucony', 'Oczekujący')),
    [Liczba dni urlopu] INT NOT NULL CHECK ([Liczba dni urlopu] >= 0),

    CONSTRAINT [CK_Urlopy_TypUrlopu] CHECK ([Typu urlopu] IN ('Urlop wypoczynkowy',
'Urlop okolicznościowy', 'Urlop bezpłatny')),
    CONSTRAINT [FK_Urlopy_Pracownicy] FOREIGN KEY ([Id pracownika]) REFERENCES
Pracownicy ([ID Pracownika]),
    FOREIGN KEY ([Id pracownika]) REFERENCES Pracownicy ([ID pracownika])
);
```

6. Implementacja niedeklaratywnych mechanizmów sprawdzania poprawności danych

Procedura Składowana: Sprawdzenie Wykorzystanego Urlopu

```
CREATE PROCEDURE SprawdzWykorzystanyUrlop
    @ID_pracownika INT,
    @Wykorzystany_Urlop INT
AS
BEGIN
    DECLARE @Nominał_Urlopu INT;

    SELECT @Nominał_Urlopu = UrlopNależny.[Nominał urlopu]
    FROM UrlopNależny
    WHERE UrlopNależny.[Id pracownika] = @ID_pracownika;

    -- Sprawdź, czy Wykorzystany Urlop nie jest większy niż Nominał Urlopu
    IF @Wykorzystany_Urlop > @Nominał_Urlopu
    BEGIN
        RAISERROR('Wykorzystany Urlop nie może być większy niż Nominał Urlopu', 16,
1);
        RETURN;
    END

    UPDATE Pracownicy
    SET [Wykorzystany Urlop] = @Wykorzystany_Urlop
    WHERE [ID pracownika] = @ID_pracownika;
END
```

Procedura Składowana: Sprawdzenie i Wstawienie Nowego Urlopu

```
CREATE PROCEDURE SprawdzIWSstawUrlop
    @ID_urlopu INT,
    @Id_pracownika INT,
    @Data_rozpoczecia DATE,
    @Data_zakonczenia DATE,
    @Typu_urlopu VARCHAR(50),
    @Status_akceptacji VARCHAR(50),
    @Liczba_dni_urlopu INT
AS
BEGIN
    DECLARE @Nominał_Urlopu INT;

    SELECT @Nominał_Urlopu = [Nominał urlopu]
    FROM UrlopNależny
    WHERE [Id pracownika] = @Id_pracownika;

    -- Sprawdź, czy Liczba dni urlopu nie przekracza maksymalnego limitu urlopu
    IF @Liczba_dni_urlopu > @Nominał_Urlopu
    BEGIN
        RAISERROR('Liczba dni urlopu nie może przekraczać maksymalnego limitu urlopu',
16, 1);
        RETURN;
    END

    -- Sprawdź, czy Data zakończenia jest późniejsza niż Data rozpoczęcia
    IF @Data_zakonczenia <= @Data_rozpoczecia
    BEGIN
        RAISERROR('Data zakończenia musi być późniejsza niż data rozpoczęcia', 16, 1);
        RETURN;
    END
```

```
    INSERT INTO Urlopy ([ID urlopu], [Id pracownika], [Data rozpoczęcia], [Data
zakończenia], [Typu urlopu], [Status akceptacji], [Liczba dni urlopu])
    VALUES (@ID_urlopu, @Id_pracownika, @Data_rozpoczecia, @Data_zakonczenia,
@Typu_urlopu, @Status_akceptacji, @Liczba_dni_urlopu);
END
```

Procedura Składowana: Sprawdzenie i Wstawienie Nowego Pracownika

```
CREATE PROCEDURE SprawdzIWStawPracownika
    @ID_pracownika INT,
    @ID_Wydziału INT,
    @Imie VARCHAR(25),
    @Nazwisko VARCHAR(25),
    @PESEL VARCHAR(11),
    @KodPocztowy CHAR(6),
    @Miejscowosc VARCHAR(30),
    @Ulica VARCHAR(40),
    @Data_Urodzenia DATE,
    @Wyksztalcenie NVARCHAR(20),
    @Data_Zatrudnienia DATE,
    @Staz_Pracy INT,
    @Wykorzystany_Urlop INT,
    @Data_Zwolnienia DATE = NULL,
    @Status_Zatrudnienia NVARCHAR(20),
    @Przelozony VARCHAR(40) = NULL
AS
BEGIN
        IF @Data_Zwolnienia <= @Data_Zatrudnienia
        BEGIN
            RAISERROR('Data Zwolnienia musi być późniejsza niż Data Zatrudnienia', 16,
1);
            RETURN;
        END

INSERT INTO Pracownicy ([ID pracownika], [ID Wydziału], [Imię], [Nazwisko], [PESEL],
[KodPocztowy], [Miejscowość], [Ulica], [Data Urodzenia], [Wykształcenie], [Data
Zatrudnienia], [Staż Pracy], [Wykorzystany Urlop], [Data Zwolnienia], [Status
Zatrudnienia], [Przełożony])
    VALUES (@ID_pracownika, @ID_Wydziału, @Imie, @Nazwisko, @PESEL, @KodPocztowy,
@Miejscowosc, @Ulica, @Data_Urodzenia, @Wyksztalcenie, @Data_Zatrudnienia,
@Staz_Pracy, @Wykorzystany_Urlop, @Data_Zwolnienia, @Status_Zatrudnienia,
@Przelozony);
END
```

7. Wprowadzenie przykładowych danych

```
INSERT INTO Wydział ([ID Wydziału], [Nazwa Wydziału], [Kierownik])
VALUES (1, 'Wydział Księgowości', 'Jan Kowalski'),
       (2, 'Wydział Projektowy', 'Anna Nowak'),
       (3, 'Wydział Projektowy', 'Piotr Wiśniewski');

INSERT INTO Pracownicy ([ID pracownika], [ID Wydziału], [Imię], [Nazwisko], [PESEL],
[KodPocztowy], [Miejscowość], [Ulica], [Data Urodzenia], [Wykształcenie], [Data
Zatrudnienia], [Staż Pracy], [Wykorzystany Urlop], [Data Zwolnienia], [Status
Zatrudnienia], [Przełożony])
VALUES (1, 1, 'Marcin', 'Zieliński', '85010112345', '00-001', 'Warszawa', 'Ulica 1',
'1985-01-01', 'Wyższe', '2010-06-15', 14, 10, NULL, 'Aktywny', 'Jan Kowalski'),
       (2, 2, 'Marta', 'Kowalczyk', '87030567890', '00-002', 'Kraków', 'Ulica 2',
'1987-03-05', 'Wyższe', '2015-03-01', 9, 5, NULL, 'Aktywny', 'Anna Nowak'),
       (3, 3, 'Paweł', 'Wiśniewski', '90041234567', '00-003', 'Łódź', 'Ulica 3',
'1990-04-12', 'Średnie', '2018-07-20', 6, 2, NULL, 'Aktywny', 'Piotr Wiśniewski');

INSERT INTO UrlopNależny ([ID roku], [Id pracownika], [Nominał urlopu], [Ilość
wykorzystanych dni])
VALUES (2024, 1, 26, 10),
       (2024, 2, 26, 5),
       (2024, 3, 20, 2);

INSERT INTO Urlopy ([ID urlopu], [Id pracownika], [Data rozpoczęcia], [Data
zakończenia], [Typu urlopu], [Status akceptacji], [Liczba dni urlopu])
VALUES (1, 1, '2024-07-01', '2024-07-15', 'Urlop wypoczynkowy', 'Zaakceptowany', 14),
       (2, 2, '2024-08-01', '2024-08-10', 'Urlop wypoczynkowy', 'Zaakceptowany', 10),
       (3, 3, '2024-09-01', '2024-09-05', 'Urlop okolicznościowy', 'Zaakceptowany',
5);
```

8. Definicja i implementacja zasad ochrony bazy danych

Instrukcja tworzenia kopii zapasowej w bazie danych SQL

Krok 1: Przygotowanie do tworzenia kopii zapasowej

1. **Zalogowanie się do serwera SQL**
   - Użyj narzędzia takiego jak SQL Server Management Studio (SSMS) lub połącz się z serwerem SQL za pomocą skryptu.
2. **Sprawdzenie uprawnień**
   - Upewnij się, że masz odpowiednie uprawnienia do tworzenia kopii zapasowych bazy danych. Wymagane są uprawnienia `BACKUP DATABASE`.

Krok 2: Tworzenie kopii zapasowej bazy danych

1. **Wybór bazy danych**

```
USE ZarzadzanieUrlopami;
GO;
```

2. **Tworzenie pełnej kopii zapasowej bazy danych**

```
BACKUP DATABASE ZarzadzanieUrlopami
TO DISK = 'C:\Backup\ZarzadzanieUrlopami_Full.bak'
WITH FORMAT,
     MEDIANAME = 'ZarzadzanieUrlopamiBackup',
     NAME = 'Pelna kopia zapasowa bazy ZarzadzanieUrlopami';
GO;
```

3. **Tworzenie różnicowej kopii zapasowej bazy danych**
   - W celu zaoszczędzenia miejsca, można również tworzyć różnicowe kopie zapasowe po wykonaniu pełnej kopii zapasowej.

```
BACKUP DATABASE ZarzadzanieUrlopami
TO DISK = 'C:\Backup\ZarzadzanieUrlopami_Diff.bak'
WITH DIFFERENTIAL,
     MEDIANAME = 'ZarzadzanieUrlopamiBackup',
     NAME = 'Roznicowa kopia zapasowa bazy ZarzadzanieUrlopami';
GO;
```

4. **Tworzenie kopii zapasowej dziennika transakcji**
   - Kopie zapasowe dziennika transakcji pomagają w odzyskiwaniu danych do konkretnego punktu w czasie.

```
BACKUP LOG ZarzadzanieUrlopami
TO DISK = 'C:\Backup\ZarzadzanieUrlopami_Log.bak'
WITH NOFORMAT,
     MEDIANAME = 'ZarzadzanieUrlopamiBackup',
     NAME = 'Kopia zapasowa dziennika transakcji bazy ZarzadzanieUrlopami';
GO;
```

Krok 3: Automatyzacja kopii zapasowych

1. **Utworzenie harmonogramu kopii zapasowych za pomocą SQL Server Agent**
   - Użyj SQL Server Agent, aby utworzyć zadania kopii zapasowych, które będą automatycznie uruchamiane w określonych interwałach.
2. **Przykładowy skrypt tworzenia zadania kopii zapasowej**

```
EXEC msdb.dbo.sp_add_job
    @job_name = N'Zadanie pełnej kopii zapasowej';

EXEC msdb.dbo.sp_add_jobstep
    @job_name = N'Zadanie pełnej kopii zapasowej',
    @step_name = N'Krok kopii zapasowej',
    @subsystem = N'TSQL',
    @command = N'BACKUP DATABASE ZarzadzanieUrlopami TO DISK =
''C:\Backup\ZarzadzanieUrlopami_Full.bak'' WITH FORMAT;',
    @on_success_action = 1,
    @on_fail_action = 2;

EXEC msdb.dbo.sp_add_schedule
    @schedule_name = N'Harmonogram pełnej kopii zapasowej',
    @freq_type = 4,
    @freq_interval = 1,
    @active_start_time = 010000;

EXEC msdb.dbo.sp_attach_schedule
    @job_name = N'Zadanie pełnej kopii zapasowej',
    @schedule_name = N'Harmonogram pełnej kopii zapasowej';

EXEC msdb.dbo.sp_add_jobserver
    @job_name = N'Zadanie pełnej kopii zapasowej';
```

Zasady Ochrony Bazy Danych

Zasady ogólne

1. **Regularne tworzenie kopii zapasowych**
   - Pełne kopie zapasowe co najmniej raz na tydzień.
   - Różnicowe kopie zapasowe codziennie.
   - Kopie zapasowe dziennika transakcji co godzinę.
2. **Bezpieczne przechowywanie kopii zapasowych**
   - Kopie zapasowe powinny być przechowywane na oddzielnych, zabezpieczonych serwerach lub nośnikach.
   - Kopie zapasowe powinny być szyfrowane.
3. **Ograniczenie dostępu**
   - Dostęp do operacji backupu i przywracania bazy danych powinien być ograniczony do uprawnionych administratorów.
   - Użytkownicy powinni mieć tylko niezbędne uprawnienia do wykonywania swoich obowiązków.
4. **Regularne testowanie procedur przywracania**
   - Przeprowadzanie regularnych testów przywracania danych, aby upewnić się, że kopie zapasowe są sprawne i mogą być wykorzystane w razie potrzeby.

5. **Monitorowanie i logowanie**
    - Wszystkie operacje backupu i przywracania powinny być monitorowane i logowane.
    - Alarmowanie w przypadku niepowodzenia operacji backupu.

Kontrola dostępu i bezpieczeństwo danych

1. **Role i uprawnienia**
    - Definiowanie ról i przypisywanie im odpowiednich uprawnień.
    - Regularne przeglądanie i aktualizacja uprawnień użytkowników.
2. **Szyfrowanie danych**
    - Szyfrowanie danych w spoczynku i w trakcie przesyłania.
    - Wykorzystanie SSL/TLS do zabezpieczenia połączeń z bazą danych.
3. **Audyt i zgodność z przepisami**
    - Regularne audyty bezpieczeństwa bazy danych.
    - Zapewnienie zgodności z obowiązującymi przepisami dotyczącymi ochrony danych.

Przykład implementacji kontroli dostępu

1. **Tworzenie ról i przypisywanie uprawnień**

```
CREATE ROLE Rola_MenedzeraUrlopow;
CREATE ROLE Rola_MenedzeraHR;

GRANT SELECT, UPDATE, INSERT, DELETE ON Pracownicy TO Rola_MenedzeraUrlopow;
GRANT SELECT, UPDATE, INSERT, DELETE ON Urlopy TO Rola_MenedzeraUrlopow;
GRANT SELECT, UPDATE, INSERT, DELETE ON HR TO Rola_MenedzeraHR;
GRANT SELECT ON SzczegolyPracownikow TO Rola_MenedzeraHR;
DENY UPDATE, INSERT, DELETE ON Urlopy TO Rola_MenedzeraHR;
```

2. **Przypisywanie użytkowników do ról**

```
ALTER ROLE Rola_MenedzeraUrlopow ADD MEMBER Uzytkownik_Admin;
ALTER ROLE Rola_MenedzeraHR ADD MEMBER Uzytkownik_HR;
```