***Wojskowa Akademia Techniczna***

***im. Jarosława Dąbrowskiego***



**Wydział Cybernetyki, kierunek informatyka - inżynieria systemów**

Realizacja zadania laboratoryjnego w ramach przedmiotu:

*Systemy Baz Danych*

Temat laboratorium:

***Temporalne Bazy Danych***

**Opracował:** Radosław Relidzyński, **Grupa:** WCY23IX3S4

Spis treści

[Wstęp teoretyczny 3](#_Toc169109243)

[Środowisko 3](#_Toc169109244)

[Opis środowiska 3](#_Toc169109245)

[Wykaz informacji o wersjach środowiska: 4](#_Toc169109246)

[Instrukcja do przygotowania bazy danych 4](#_Toc169109247)

[Model logiczny 6](#_Toc169109248)

[Model relacyjny 6](#_Toc169109249)

[Wykaz tabel 7](#_Toc169109250)

[Wykaz zapytań 7](#_Toc169109251)

[Zapytanie 1: suma cen biletów dla zniżek 7](#_Toc169109252)

[Zapytanie 2: wiek lotu 7](#_Toc169109253)

[Zapytanie 3: średni czas lotu 8](#_Toc169109254)

[Zapytanie 4: liczba zmian w biletach lotu 8](#_Toc169109255)

[Zapytanie 5: bilety kupione przez klienta w danym roku 9](#_Toc169109256)

[Zapytanie 6: średnia cen biletów dla modelu 9](#_Toc169109257)

[Zapytanie 7: średnia cen biletów w zależności od zniżki historycznie 10](#_Toc169109258)

[Zapytanie 8: średnia cen biletów na podstawie miejsc z danego momentu w czasie 10](#_Toc169109259)

[Wykaz wyzwalaczy 11](#_Toc169109260)

[Wyzwalacz informacji, jeśli nowy klient założył konto w dniu urodzin 11](#_Toc169109261)

[Wyzwalacz sprawdzający czy założony użytkownik posiadał wcześniej już konto 12](#_Toc169109262)

[Wyzwalacz zwracający licznik zmian w locie 14](#_Toc169109263)

[Wykaz procedur 15](#_Toc169109264)

[Podsumowanie 15](#_Toc169109265)

# Wstęp teoretyczny

**Baza danych** – „uporządkowany zbiór danych określających wybrany fragment rzeczywistości lub problemu, które są przechowywane trwale w pamięci komputerowej do której może mieć dostęp wielu użytkowników w dowolnej chwili czasu.”

**System zarządzania bazami danych** – „zorganizowany zbiór narzędzi (programów komputerowych i bibliotek), które umożliwiają wykonanie podstawowych operacji na danych (CRUD) zawartych w jednej lub więcej bazach danych.”

System baz danych – jego definicja wyraża się wzorem:

Gdzie:

[źródło: materiały z wykładu „Temporalne bazy danych” dr inż. Jarosława Koszeli]

# Środowisko

### Opis środowiska

W ramach projektu rolę systemu zarządzania bazą danych będzie pełnić narzędzie Microsoft SQL Server, a baza danych zostanie utworzona pod nazwą „lotniska europy.” Dodatkowo wykorzystane zostanie środowisko SQL Server Management Studio.

**SQL Server Management Studio** – „zintegrowane środowisko do zarządzania wszystkimi komponentami (baza danych, usługi analityczne, usługi raportowe itd.), wchodzącymi w skład Microsoft SQL Server. Zawiera narzędzia do konfiguracji, monitorowania i administrowania instancjami SQL Server. Umożliwia budowę zapytań i skryptów, zawiera zarówno edytor skryptów jak i narzędzia graficzne.”  
[źródło: <https://pl.wikipedia.org/wiki/SQL_Server_Management_Studio>]

**Microsoft SQL Server** – „system zarządzania bazą danych, wspierany i rozpowszechniany przez korporację Microsoft. Jest to główny produkt bazodanowy tej firmy, który charakteryzuje się tym, iż jako język zapytań używany jest przede wszystkim Transact-SQL, który stanowi rozwinięcie standardu ANSI/ISO.”  
[źródło: <https://pl.wikipedia.org/wiki/Microsoft_SQL_Server>]

### Wykaz informacji o wersjach środowiska:

System operacyjny: Windows 11 Pro 23H2

Informacja o wersjach komponentów SQL Server Management Studio:

|  |  |
| --- | --- |
| SQL Server Management Studio | 20.1.10.0 |
| SQL Server Management Objects (SMO) | 17.100.31.0+ccc8de5aee492d8b603963ce96e9f987  7a21afa4 |
| Microsoft T-SQL Parser | 17.2.1.1+f4dfdb798c9aaa90bbe9a0cbfd92e100200a  d26c.f4dfdb798c9aaa90bbe9a0cbfd92e100200ad26c |
| Microsoft Analysis Services Client Tools | 20.0.3.0 |
| Microsoft Data SqlClient (MDS) | 5.1.5 |
| Microsoft SQL Server Data-Tier Application Framework (DacFX) | 162.2.111.2+1a4d708fee9d82fe4e01e02f3962d1e8  3d374807.1a4d708fee9d82fe4e01e02f3962d1e83d  374807 |
| Microsoft .NET Framework | 4.0.30319.42000 |
| Operating System | 10.0.22631 |

# Instrukcja do przygotowania bazy danych

1. **Inicjalizacja bazy – uruchomienie skryptu „create.sql”.**
2. **Wypełnienie bazy danymi – uruchomienie skryptu „fill\_data.sql”.**
3. **Sprawdzenie czy baza jest wypełniona – uruchomienie skryptu „check.sql”.**
4. **Wywołanie przykładowych zapytań – uruchomienie skryptu „query.sql”.**
5. **Dodanie i sprawdzenie wyzwalaczy:**
   1. Dodanie pierwszego – uruchomienie skryptu „trg\_urodziny.sql”.
   2. Sprawdzenie pierwszego – uruchomienie skryptu „test\_trg\_ urodziny.sql”.
   3. Dodanie drugiego – uruchomienie skryptu „trg\_powrot\_klienta.sql”.
   4. Sprawdzenie drugiego – uruchomienie skryptu „test\_trg\_powrot\_klienta.sql”.
   5. Dodanie trzeciego – uruchomienie skryptu „trg\_licznik\_lotow.sql”.
   6. Sprawdzenie trzeciego – uruchomienie skryptu „test\_trg\_licznik\_lotow.sql”.
   7. Ustawienie priorytetu wyzwalaczy – uruchomienie skryptu „set\_trg\_priority.sql”.
6. **Dodanie procedury:**
   1. Stworzenie procedury – uruchomienie skryptu „prd\_usun\_stare\_bilety.sql”.
   2. Uruchomienie agenta
      1. W eksploratorze rozwinąć opcje dla serwera.
      2. Kliknąć prawym przyciskiem myszy na „SQL Server Agent” i wybrać opcję „start” (może być wymagane potwierdzenie jako administratora).
   3. Dodanie zadania wykonującego prodedurę:
      1. Kliknąć ponownie prawym przyciskiem myszy na „SQL Server Agent”, wybrać opcję „New”, następnie „Job…”.
      2. Podać nazwę, na przykład „usun\_stare\_bilety”.
      3. Przejść do zakładki „Steps” i wykonać następujące działania:
         1. Podać nazwę, na przykład „usun\_stare\_bilety”.
         2. W polu „Database” wybrać z listy rozwijanej „lotniska\_europy”.
         3. W polu „Command” wpisać instrukcję wywołania procedury, czyli „EXEC prd\_usun\_stare\_bilety”.
         4. Kliknąć „OK”.
      4. Przejść do zakładki „Schedules” i wykonać następujące działania:
         1. Wybrać opcję „New…”.
         2. Podać nazwę, na przykład „usun\_stare\_bilety”.
         3. W ramach segmentu „Frequency” dla opcji „Occurs” wybrać z lisy rozwijanej opcję „Daily”.
         4. W ramach segmentu „Daily frequency” dla opcji „Occurs once at” zmienić, żeby uruchamiał się o godzinie „12:00:00 PM”.
         5. Kliknąć „OK”.
      5. Zakończyć dodawanie procedury klikając „OK”.
   4. Sprawdzić, czy procedura jest dodana
      1. Rozwinąć opcje dla „SQL Server Agent”, a następnie „Jobs”,
      2. Sprawdzić, czy lista zawiera „usun\_stare\_bilety”, poniżej przykład: A screenshot of a computer

         Description automatically generated

# Model logiczny

A diagram of a computer

Description automatically generated

# Model relacyjny

A screenshot of a computer

Description automatically generated

# Wykaz tabel

Baza danych stworzona w ramach projektu pełni zadanie zbierania i zarządzania informacjami dotyczącymi wybranych lotnisk Europy oraz lotów między nimi. W ramach tego powołane zostają poniższe tabele:

* Samolot – informacje o samolocie
* Miejsce – informacje o miejscu w samolocie
* Model – informacje o modelach samolotów
* Pracownik – Informacje o pracownikach
* Licencja – intersekcja między pracownikiem i modelem, mówi, jakim
* samolotem może lecieć dany pracownik
* Lotnisko – informacje o lotniskach
* Lot (*temporalna*) – informacje o locie na podstawie samolotu i lotnisk
* Rola w locie – intersekcja między pracownikiem i lotem, mówi, jaką
* rolę pełni dany pracownik w danym locie
* Klient (*temporalna*) – informacje o kliencie
* Zniżki – informacje o zniżkach klientów
* Bilet (*temporalna*) – informacje o bilecie na podstawie lotu, miejsca w samolocie oraz klienta

# Wykaz zapytań

### Zapytanie 1: suma cen biletów dla zniżek

#### Kod zapytania

-- suma cen biletów zakupionych w ramach różnych rodzajów zniżek przez klientów w 2023 roku licząc tylko te o sumie większej niż 500

SELECT k.rodzaj\_znizki, SUM(b.cena) AS suma\_cen\_biletow

FROM bilet FOR SYSTEM\_TIME BETWEEN '2023-01-01' AND '2024-01-01' b

JOIN klient k ON b.nr\_klienta = k.nr\_klienta

GROUP BY k.rodzaj\_znizki

HAVING SUM(b.cena) > 500

ORDER BY suma\_cen\_biletow DESC;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

### Zapytanie 2: wiek lotu

#### Kod zapytania

-- wiek informacji o locie

SELECT l.nr\_lotu, MIN(l.SysStartTime) AS data\_dodania, DATEDIFF(DAY, MIN(l.SysStartTime), GETDATE()) AS wiek\_w\_dniach

FROM lot FOR SYSTEM\_TIME ALL l

GROUP BY l.nr\_lotu;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

### Zapytanie 3: średni czas lotu

#### Kod zapytania

-- średni czas dla lotu na podstawie wszystkich wartości historycznych

SELECT l.nr\_lotu, AVG(l.czas\_lotu\_min) AS sredni\_czas\_lotu

FROM lot FOR SYSTEM\_TIME ALL l

GROUP BY l.nr\_lotu

ORDER BY sredni\_czas\_lotu DESC;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

### Zapytanie 4: liczba zmian w biletach lotu

#### Kod zapytania

-- zliczanie zmian w biletach w 2020 roku w zależności od lotu

SELECT b.nr\_lotu, COUNT(\*) AS liczba\_zmian

FROM bilet FOR SYSTEM\_TIME BETWEEN '2020-01-01' AND '2020-12-31' b

WHERE b.SysEndTime != '9999-12-31 23:59:59.9999999'

GROUP BY b.nr\_lotu

ORDER BY liczba\_zmian DESC;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

### Zapytanie 5: bilety kupione przez klienta w danym roku

#### Kod zapytania

-- zliczanie biletów kupionych przez klienta w zależności od roku

SELECT k.nr\_klienta, CONCAT(k.imie, ' ', k.nazwisko) AS klient, YEAR(b.SysStartTime) AS Rok, COUNT(DISTINCT b.kod\_biletu) AS liczba\_biletow

FROM bilet FOR SYSTEM\_TIME ALL b

JOIN klient k ON b.nr\_klienta = k.nr\_klienta

GROUP BY k.nr\_klienta, k.imie, k.nazwisko, YEAR(b.SysStartTime)

ORDER BY k.nr\_klienta, Rok;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

### Zapytanie 6: średnia cen biletów dla modelu

#### Kod zapytania

-- średnia cen biletów uwzględniając wszystkie aktualizacje w zależności od modelu samolotu realizującego lot

SELECT m.nazwa AS ModelSamolotu, AVG(b.cena) AS srednia\_cen\_biletow

FROM bilet FOR SYSTEM\_TIME ALL b

JOIN lot l ON b.nr\_lotu = l.nr\_lotu

JOIN samolot s ON l.nr\_samolotu = s.nr\_samolotu

JOIN model m ON s.nr\_modelu = m.nr\_modelu

GROUP BY m.nazwa

ORDER BY srednia\_cen\_biletow DESC;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

### Zapytanie 7: średnia cen biletów w zależności od zniżki historycznie

#### Kod zapytania

-- średnia cena biletów dla klienta w zależności od zniżki jaką ma lub miał

WITH ZmianyZnizek AS (

SELECT k.nr\_klienta, CONCAT(k.imie, ' ', k.nazwisko) AS klient, k.rodzaj\_znizki, k.SysStartTime, k.SysEndTime

FROM klient FOR SYSTEM\_TIME BETWEEN '2021-01-01' AND '2023-01-01' k

),

BiletyCeny AS (

SELECT b.nr\_klienta, b.cena, b.SysStartTime

FROM bilet FOR SYSTEM\_TIME BETWEEN '2021-01-01' AND '2023-01-01' b

)

SELECT zz.klient klient, zz.rodzaj\_znizki, AVG(bc.cena) AS srednia\_cen\_biletow

FROM ZmianyZnizek zz

JOIN BiletyCeny bc ON zz.nr\_klienta = bc.nr\_klienta

GROUP BY zz.klient, zz.rodzaj\_znizki

ORDER BY klient, rodzaj\_znizki DESC;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

### Zapytanie 8: średnia cen biletów na podstawie miejsc z danego momentu w czasie

#### Kod zapytania

-- średnia cena biletu na podstawie klas miejsc

SELECT m.klasa, AVG(b.cena) AS srednia\_cena

FROM bilet FOR SYSTEM\_TIME AS OF '2021-01-01 12:00:00.0000000' b

JOIN miejsce m ON b.nr\_miejsca = m.nr\_miejsca AND b.nr\_samolotu = m.nr\_samolotu

GROUP BY m.klasa

ORDER BY srednia\_cena DESC;

#### Rezultat zapytania

A screenshot of a computer

Description automatically generated

# Wykaz wyzwalaczy

### Wyzwalacz informacji, jeśli nowy klient założył konto w dniu urodzin

Uzasadnienie biznesowe: W ramach akcji promocyjnej klienci, którzy założą konto w dniu swoich urodzin mogą otrzymać dodatkową zniżkę.

-- jesli klient zalozy konto w dniu urodzin, to dostanie znizke

CREATE TRIGGER trg\_urodziny

ON klient

AFTER INSERT

AS

BEGIN

DECLARE @Message NVARCHAR(1000);

DECLARE @nr\_klienta INT;

DECLARE @imie NVARCHAR(50);

DECLARE @nazwisko NVARCHAR(50);

-- DECLARE @data\_dodania DATE;

DECLARE @data\_urodzenia DATE;

DECLARE @plec NVARCHAR(1);

DECLARE @rodzaj\_znizki NVARCHAR(20);

DECLARE @miesiac\_dodania INT;

DECLARE @dzien\_dodania INT;

DECLARE @miesiac\_urodzenia INT;

DECLARE @dzien\_urodzenia INT;

DECLARE @klient NVARCHAR(40);

DECLARE client\_birthday\_cursor CURSOR FOR

SELECT imie, nazwisko, data\_urodzenia

FROM inserted;

OPEN client\_birthday\_cursor;

FETCH NEXT FROM client\_birthday\_cursor INTO @imie, @nazwisko, @data\_urodzenia;

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @miesiac\_dodania = MONTH(GETDATE());

SET @dzien\_dodania = DAY(GETDATE());

SET @miesiac\_urodzenia = MONTH(@data\_urodzenia);

SET @dzien\_urodzenia = DAY(@data\_urodzenia);

SET @klient = CONCAT(@imie, ' ', @nazwisko);

IF @miesiac\_dodania = @miesiac\_urodzenia AND @dzien\_dodania = @dzien\_urodzenia

BEGIN

SET @Message = FORMATMESSAGE('%s zalozyl wlasnie konto w dniu urodzin, przeslij mu znizki', @klient);

RAISERROR(@Message, 0, 1) WITH NOWAIT;

END

FETCH NEXT FROM client\_birthday\_cursor INTO @imie, @nazwisko, @data\_urodzenia;

END;

CLOSE client\_birthday\_cursor;

DEALLOCATE client\_birthday\_cursor;

END;

GO

Przykład zapytania aktywującego wyzwalacz:

-- TEST trg\_urodziny

INSERT INTO KLIENT (IMIE, NAZWISKO, DATA\_URODZENIA, RODZAJ\_ZNIZKI, PLEC)

VALUES ('Dzisiaj', 'Urodzony', CAST(DATEADD(YEAR, -18, GETDATE()) AS DATETIME), NULL, 'M');

INSERT INTO KLIENT (IMIE, NAZWISKO, DATA\_URODZENIA, RODZAJ\_ZNIZKI, PLEC)

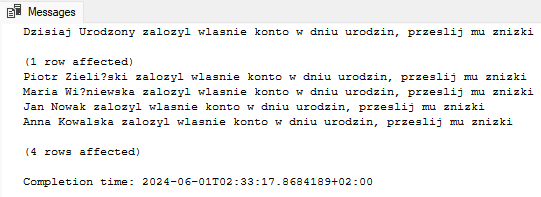
VALUES

('Anna', 'Kowalska', CAST(DATEADD(YEAR, -40, GETDATE()) AS DATETIME), NULL, 'M'),

('Jan', 'Nowak', CAST(DATEADD(YEAR, -30, GETDATE()) AS DATETIME), NULL, 'M'),

('Maria', 'Wi?niewska', CAST(DATEADD(YEAR, -20, GETDATE()) AS DATETIME), NULL, 'M'),

('Piotr', 'Zieli?ski', CAST(DATEADD(YEAR, -30, GETDATE()) AS DATETIME), NULL, 'M');



### Wyzwalacz sprawdzający czy założony użytkownik posiadał wcześniej już konto

Uzasadnienie biznesowe: Aby zoptymalizować nr\_klienta tak, żeby mniej rosnął, po dodaniu ponownie tego samego klienta baza danych na podstawie tabeli historycznej przywraca jego stary numer.

-- jesli klient znajduje sie w tabeli historycznej, uzyj starego nr\_klienta

CREATE TRIGGER trg\_powrot\_klienta

ON klient

AFTER INSERT

AS

BEGIN

DECLARE @Message NVARCHAR(1000);

DECLARE @nr\_klienta INT;

DECLARE @imie NVARCHAR(50);

DECLARE @nazwisko NVARCHAR(50);

DECLARE @data\_urodzenia DATE;

DECLARE @plec NVARCHAR(1);

DECLARE @rodzaj\_znizki NVARCHAR(20);

DECLARE client\_comeback\_cursor CURSOR FOR

SELECT imie, nazwisko, data\_urodzenia, plec, rodzaj\_znizki

FROM inserted;

OPEN client\_comeback\_cursor;

FETCH NEXT FROM client\_comeback\_cursor INTO @imie, @nazwisko, @data\_urodzenia, @plec, @rodzaj\_znizki;

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @nr\_klienta = nr\_klienta

FROM klientHistory

WHERE imie = @imie AND nazwisko = @nazwisko AND data\_urodzenia = @data\_urodzenia;

IF @nr\_klienta IS NOT NULL

BEGIN

DELETE FROM klient

WHERE imie = @imie AND nazwisko = @nazwisko AND data\_urodzenia = @data\_urodzenia;

SET IDENTITY\_INSERT klient ON;

INSERT INTO klient (nr\_klienta, imie, nazwisko, data\_urodzenia, plec, rodzaj\_znizki)

VALUES (@nr\_klienta, @imie, @nazwisko, @data\_urodzenia, @plec, @rodzaj\_znizki);

SET @Message = FORMATMESSAGE('Klient %s %s istnieje w historii, uzyto starego nr\_klienta %d.', @imie, @nazwisko, @nr\_klienta);

RAISERROR(@Message, 0, 1) WITH NOWAIT;

END

ELSE

BEGIN

SET @Message = FORMATMESSAGE('Klient %s %s zostal dodany jako nowy klient.', @imie, @nazwisko);

RAISERROR(@Message, 0, 1) WITH NOWAIT;

END

FETCH NEXT FROM client\_comeback\_cursor INTO @imie, @nazwisko, @data\_urodzenia, @plec, @rodzaj\_znizki;

END;

CLOSE client\_comeback\_cursor;

DEALLOCATE client\_comeback\_cursor;

END;

GO

Przykład sytuacji wywołania:

-- TEST trg\_powrot\_klienta

INSERT INTO KLIENT (IMIE, NAZWISKO, DATA\_URODZENIA, RODZAJ\_ZNIZKI, PLEC)

VALUES ('Karol', 'Karolewski', CAST('03-08-2002' AS DATETIME), NULL, 'M');

-- pobierz nr\_klienta nowo dodanego klienta

DECLARE @nr\_klienta INT;

SELECT @nr\_klienta = nr\_klienta

FROM KLIENT

WHERE IMIE = 'Karol' AND NAZWISKO = 'Karolewski' AND DATA\_URODZENIA = CAST('03-08-2002' AS DATETIME);

-- usun klienta

DELETE FROM KLIENT

WHERE nr\_klienta = @nr\_klienta;

-- dodaj ponownie klienta z nowymi danymi

INSERT INTO KLIENT (IMIE, NAZWISKO, DATA\_URODZENIA, RODZAJ\_ZNIZKI, PLEC)

VALUES ('Karol', 'Karolewski', CAST('03-08-2002' AS DATETIME), NULL, 'M');

A screenshot of a computer

Description automatically generated

### Wyzwalacz zwracający licznik zmian w locie

Uzasadnienie biznesowe: W celu weryfikacji kolejnych zmian wyzwalacz rejestruje kolejne zmiany, może on potencjalnie również informować administratora o tym, że zaszły takie zmiany.

-- nie mozna edytowa? informacji o locie, który istnieje juz od miesiaca

CREATE TRIGGER trg\_licznik\_lotow

ON lot

AFTER UPDATE

AS

BEGIN

DECLARE @Message NVARCHAR(1000);

DECLARE @nr\_lotu INT;

DECLARE @licznik INT;

SELECT @nr\_lotu = INSERTED.nr\_lotu

FROM INSERTED;

SELECT @licznik = COUNT(\*)

FROM dbo.lotHistory

WHERE nr\_lotu = @nr\_lotu;

SET @Message = FORMATMESSAGE('To jest %s zmiana lotu numer %s ', CAST(@licznik AS VARCHAR), CAST(@nr\_lotu AS VARCHAR));

RAISERROR(@Message, 0, 1) WITH NOWAIT;

END;

GO

Przykład sytuacji wywołania:

-- TEST trg\_licznik\_lotow

INSERT INTO lot (data\_lotu, czas\_lotu\_min, lotnisko\_startowe, lotnisko\_docelowe, nr\_samolotu)

VALUES (DATEADD(MONTH, 1, CONVERT(DATE, GETDATE())), 120, 1, 2, 1);

-- pobierz nr\_lotu nowo dodanego lotu

DECLARE @nr\_lotu INT;

SELECT @nr\_lotu = nr\_lotu

FROM lot

WHERE data\_lotu = DATEADD(MONTH, 1, CONVERT(DATE, GETDATE())) AND nr\_samolotu = 1;

UPDATE lot

SET czas\_lotu\_min = 130

WHERE nr\_lotu = @nr\_lotu;

UPDATE lot

SET czas\_lotu\_min = 140

WHERE nr\_lotu = @nr\_lotu;

UPDATE lot

SET czas\_lotu\_min = 150

WHERE nr\_lotu = @nr\_lotu;

DELETE FROM lot

WHERE data\_lotu = DATEADD(MONTH, 1, CONVERT(DATE, GETDATE())) AND nr\_samolotu = 1;

A screenshot of a computer

Description automatically generated

# Wykaz procedur

Procedura usuwająca bilety na loty, które już się odbyły

CREATE PROCEDURE prd\_usun\_stare\_bilety

AS

BEGIN

DELETE b

FROM bilet b

JOIN lot l ON b.nr\_lotu = l.nr\_lotu

WHERE l.data\_lotu < CAST(GETDATE() AS DATE);

END;

# Podsumowanie

W ramach realizacji zadania laboratoryjnego udało się stworzyć system zarządzania informacjami w zakresie lotnisk Europy.

Udało się skutecznie wykorzystać narzędzia Microsoft SQL Server oraz SQL Server Management Studio. W projekcie wykorzystano zaawansowane funkcje takie jak wyzwalacze i procedury

Wprowadzenie temporalności niektórych tabel umożliwia przechowywanie i analizę danych w kontekście ich zmian w czasie, co pozwala na większą kontrolę w zakresie integralności i kontroli zmian w bazie.