

# Sprawozdanie

## Bazy danych – Projekt

Temat: Filharmonia.

Wykonujący: Kamil Przygodzki I5Y5S1

Data oddawania projektu: 2017-02-23

## Analiza problemu:

Projekt dotyczy bazy danych filharmonii. Problematyka takiej organizacji związana jest z przechowywaniem informacji o koncertach, muzykach, utworach, o zakupionych biletach, a także o zmianach zachodzących w budżecie.

Specyfiką koncerty filharmonicznych jest mnogość wykonawców – w przypadku koncertów symfonicznych dużych filharmonii może ona sięgać nawet kilkuset muzyków. W przypadku przedstawianego projektu liczba ta nie przekracza 50 muzyków w czasie wykonywania jednego koncertu. Koncerty wykonywane są na jednej z dwóch dostępnych sal: koncertowej, mieszczącej 70 osób i kameralnej, mieszczącej 32 osoby. Koncerty te są biletowane, stąd też na koncert nie może wejść więcej osób niż wskazuje na to liczba miejsc.

W ramach koncertów wykonywany jest zróżnicowany repertuar, jednakże skład wykonawczy jest najczęściej bardzo podobny, zazwyczaj jest to orkiestra symfoniczna, z którą czasami współpracuje chór, bądź soliści. Stąd też w celu zaoszczędzenia pamięci możliwym było ominięcie tworzenia tabeli gromadzącej Pesele wszystkich muzyków grających w czasie danego koncertu, a utworzone zostały tabele przejściowe, która wskazują na rodzaj składu wykonawczego. Wykonywany repertuar musi być uprzednio opanowany przez każdego z muzyków, stąd konieczność istnienia tabeli określającej repertuar każdego z wykonawców.

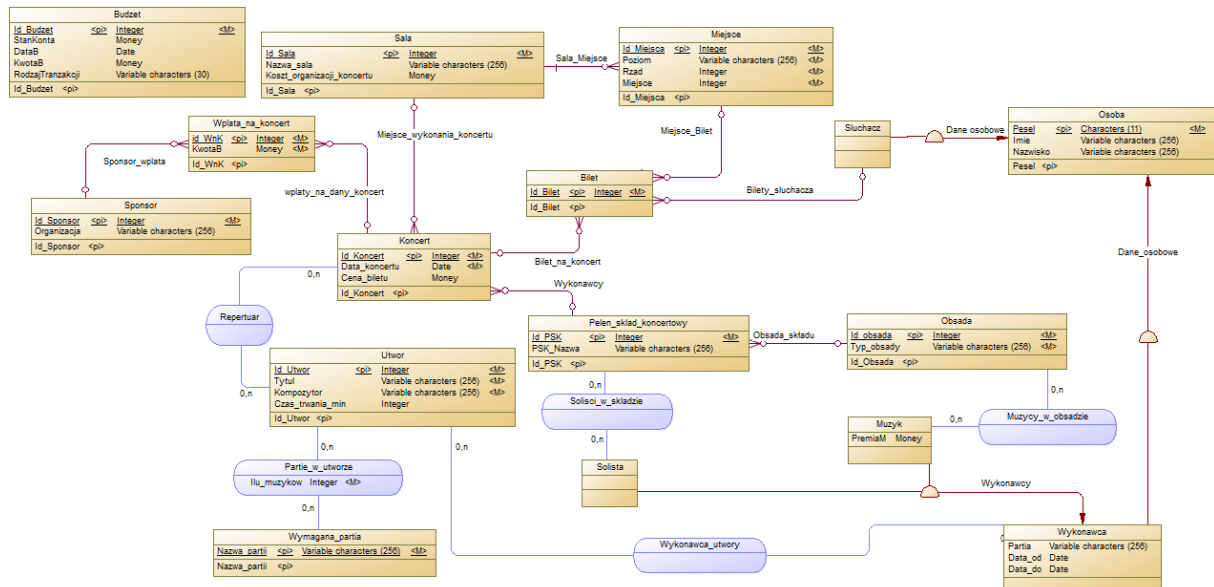
Do możliwości bazy danych zaliczają się:

- tworzenie różnych statystyk upraszczających organizację koncertów (jak np. przeliczanie średnich przychodów z podobnych koncertów)
- transakcje zakupu biletów
- tworzenie koncertów, a także ich usuwanie, razem ze wszystkimi powiązanymi z nimi rekordów
- śledzenie i automatyczne aktualizowanie stanu budżetu w związku ze zdarzeniami zachodzącymi w bazie.

Największym ograniczeniem stworzonej bazy danych jest jej przeznaczenie stałego składu wykonawczego, w przypadku organizowania koncertów na składy o zróżnicowanej liczbie muzyków należałoby utworzyć bezpośrednią asocjację pomiędzy muzykiem i koncertem.

# Projekt

## Model konceptualny:



## Opis encji i atrybutów:

- Bilet - jest potwierdzeniem dokonania opłaty przez klienta za udział w danym koncercie i jednocześnie przy koncertach biletowanych wskazuje miejsce przydzielone słuchaczowi.
  - Id\_Bilet - Identyfikator konkretnego biletu na konkretny koncert
- Budzet - Przedstawia historie stanu konta filharmonii.
  - Id\_Budzet - Identyfikator poszczególnych stanów budżetu.
  - StanKonta - Stan konta filharmonii po dokonaniu zmiany określonej w pozostałych atrybutach.
  - DataB - Data wprowadzenia nowego rekordu.
  - KwotaB - Kwota o którą zmienia się stan konta.
  - RodzajTranzakcji - Uwagi odnośnie wprowadzonej zmiany określające ich charakter.
- Koncert - Konkretnie wykonanie danego repertuaru w danej sali.
  - Id\_Koncert - Identyfikator koncertu.
  - Data\_koncertu - Termin wykonania koncertu.
  - Cena\_biletu - Cena jaką płaci słuchacz za bilet na dany koncert.
- Miejsce - jest konkretnym fotelem, miejscem przeznaczonym dla osoby kupującej bilet, wyznaczonym przez identyfikator.
  - Id\_Miejsca - Identyfikator miejsca w danej sali.
  - Poziom - Parter, bądź jeden z balkonów, jeżeli istnieje.
  - Rzad - Numer rzędu, w którym znajduje się dane miejsce.
  - Miejsce - Numer miejsca w danym rzędzie.
- Muzyk - Muzyk pracujący dla filharmonii.
  - PremiaM - Premie otrzymywane przez muzyka za poszczególne koncerty.
- Obsada - Zawarte są tu powtarzające się w niemal każdym koncercie schematy obsad muzycznych.
  - Id\_obsada - Identyfikator obsady.
  - Typ\_obsady - Zawiera nazwę typu obsady (np. orkiestra smyczkowa, czy orkiestra symfoniczna).

7. Osoba – Encja abstrakcyjna przechowująca określone dane osobowe
  - a) Pesel – Identyfikator osoby
  - b) Imie – Imię osoby.
  - c) Nazwisko – Nazwisko osoby.
8. Pelen\_sklad\_koncertowy - Zestawienie pełnych obsad, które całością mogą wykonać koncert.
  - a) Id\_PSK - Identyfikator danego składu.
  - b) PSK\_Nazwa - Słowne określenie składu wykonawczego do pewnego typu koncertów.
9. Sala - Miejsce wykonania koncertu. W danej filharmonii znajdują się dwie sale.
  - a) Id\_Sala - Numeryczny identyfikator sali.
  - b) Nazwa\_sala - Każda sala ma swoją nazwę, odzwierciedlającą przeznaczenie.
  - c) Koszt\_organizacji\_koncertu - Określa ile kosztuje zorganizowanie koncertu na danej sali.
10. Sluchacz - Klienci filharmonii.
11. Solista - Soliści pracujący dla filharmonii.
12. Sponsor - Organizacje wspierające od strony finansowej organizowanie koncertów w filharmonii.
  - a) Id\_Sponsor - Identyfikator danego sponsora.
  - b) Organizacja - Nazwa organizacji czy też firmy.
13. Utwór - Dzieło muzyczne wykonywane w ramach danego koncertu.
  - a) Id\_Utwor – Identyfikator danego utworu.
  - b) Tytuł – Tytuł utworu.
  - c) Kompozytor – Imię i nazwisko kompozytora, który napisał dany utwór.
  - d) Czas\_trwania\_min - Przybliżony czas wykonania danego utworu.
14. Wplata\_na\_koncert - Tabela określa zaangażowanie finansowe sponsorów w poszczególne koncerty.
  - a) Id\_Wnk - Klucz identyfikujący konkretną kwotę przeznaczoną na dany koncert przez sponsora.
  - b) KwotaWnK - Kwota jaką wpłacił sponsor na dany koncert.
15. Wykonawca – Encja abstrakcyjna przechowująca wspólne informacje o wykonawcach
  - a) Partia – Partia realizowana przez wykonawcę
  - b) Data\_od – Data od której wykonawca pracował dla filharmonii.
  - c) Data\_do – Data do której wykonawca pracował dla filharmonii.
16. Wymagana\_partia - Partie wokalne-instrumentalne pojawiające się w danym utworze
  - a) Nazwa\_partii - Określa nazwę partii wokalne-instrumentalnej.

#### Asocjacje:

1. Muzycy\_w\_obsadzie - Zestawienie muzyków wchodzących w skład danej obsady wykonawczej
2. Muzyk\_utwory - Utwory, które muzyk ma w swoim repertuarze.
3. Partie\_w\_utworze - Zestawienie wszystkich partii wymaganych do wykonania poszczególnych utworów.
  - a) Ilu\_muzykow - Określa ilu muzyków wykonujących daną partię jest potrzebnych do danego utworu.
4. Repertuar - Zestawienie wszystkich utworów wchodzących w skład danego koncertu.
5. Solisci\_w\_skladzie - Muzycy będący solistami w ramach danego składu
6. Solista\_utwory - Utwory, które solista ma w swoim repertuarze.

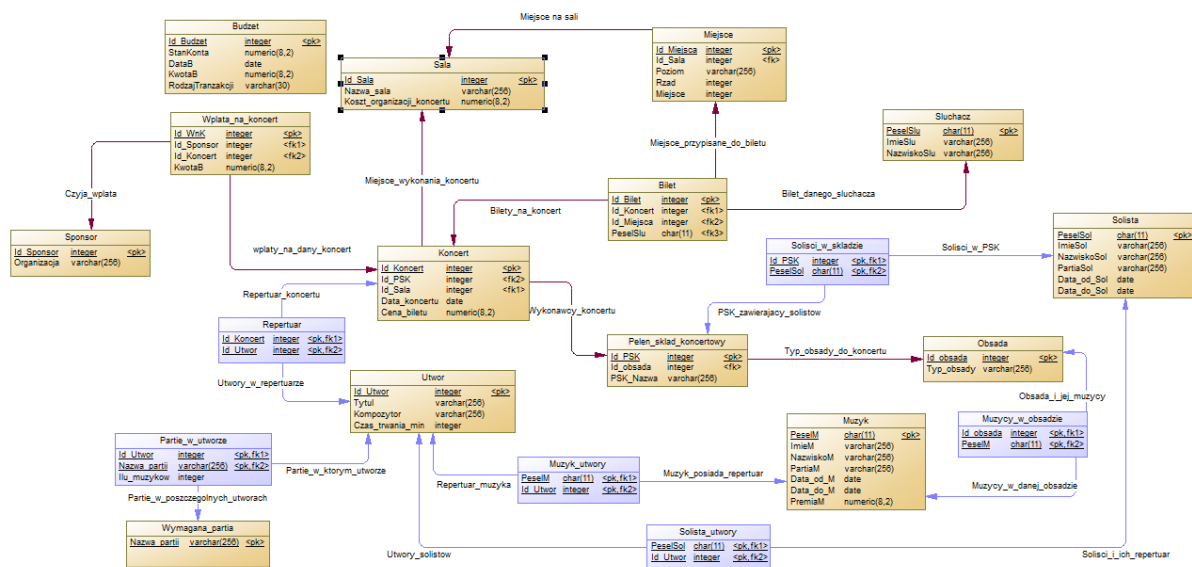
## Powiązania

1. Bilet\_na\_koncert - Dany bilet pozwala na wstęp na tylko jeden koncert, a na dany koncert sprzedawane jest wiele biletów.
2. Bilety\_sluchacza - Każdy słuchacz może kupować wiele biletów, a jeden bilet dotyczy tylko jednej osoby.
3. Miejsce\_Bilet - W ramach pojedynczego wydarzenia każdy bilet może mieć przypisane tylko jedno miejsce, natomiast w ramach różnych wydarzeń do miejsca przypisany jest zawsze inny bilet
4. Miejsce\_wykonania\_koncertu - Sala w danym momencie może być zajęta tylko przez jeden koncert - niemożliwym jest organizowanie dwóch wydarzeń jednocześnie w jednym miejscu.
5. Obsada\_składu - Dany typ obsady może się pojawiać w wielu szerszych składach koncertowych. Dany pełen skład ma w sobie jeden typ obsady.
6. Sala\_Miejsce - W każdej sali znajduje się wiele miejsc, ale każde miejsce jest przypisane tylko do jednej sali.
7. Sponsor\_wplata - Jeden sponsor może dokonać wpłaty na wiele koncertów. Dana wpłata dotyczy jednego sponsora.
8. Wpłaty\_na\_dany\_koncert - Na dany koncert mogą być przebrane kwoty przez różnych sponsorów. Kwoty te dotyczą jednego konkretnego koncertu.
9. Wykonawcy - Dany skład może wykonywać wiele koncertów. Dany koncert wykonywany jest przez jeden skład.

## Dziedziczenia:

1. Muzyk i Solista dziedziczą po Wykonawcy
2. Wykonawca i Słuchacz dziedziczą po Osobie

## Model fizyczny:



# Implementacja

## Widoki

1. Bilans\_koncertow - Widok ten obrazuje bilans każdego z koncertów - zyski, bądź straty z każdego koncertu na podstawie kosztu organizacji, dochodu z biletów, a także dofinansowania sponsorów.

```
ALTER VIEW "DBA"."Bilans_koncertow"(Id_Koncert,Data_koncert, Bilans)
```

```
AS
```

```
SELECT tab.Id_Koncert,tab.Data_koncertu, SUM(tab.Pieniadze) AS Bilans_Koncertu
```

```
FROM
```

```
(SELECT K.Id_Koncert,K.Data_koncertu, COUNT(B.Id_Bilet)*K.Cena_biletu AS Pieniadze
```

```
FROM Koncert K
```

```
INNER JOIN Bilet B
```

```
ON B.Id_Koncert=K.Id_Koncert
```

```
GROUP BY K.Id_Koncert,K.Data_koncertu,K.Cena_biletu
```

```
UNION ALL
```

```
SELECT K.Id_Koncert,K.Data_koncertu,-SUM(S.Koszt_organizacji_koncertu)
```

```
FROM Koncert K
```

```
INNER JOIN Sala S
```

```
ON K.Id_Sala=S.Id_Sala
```

```
WHERE DATEDIFF(DAY,K.Data_koncertu,NOW())>0
```

```
GROUP BY K.Id_Koncert,K.Data_koncertu
```

```
UNION ALL
```

```
SELECT K.Id_Koncert,K.Data_koncertu,-SUM(K.Wynagrodzenie_muzykow)
```

```
FROM Koncert K
```

```
WHERE DATEDIFF(DAY,K.Data_koncertu,NOW())>0
```

GROUP BY K.Id\_Koncert,K.Data\_koncertu

UNION ALL

SELECT K.Id\_Koncert,K.Data\_koncertu, SUM(WnK.Kwota)

FROM Koncert K

INNER JOIN Wplata\_na\_koncert WnK

ON WnK.Id\_Koncert=K.Id\_Koncert

WHERE DATEDIFF(DAY,K.Data\_koncertu,NOW())>0

GROUP BY K.Id\_Koncert,K.Data\_koncertu)tab

GROUP BY tab.Id\_Koncert,tab.Data\_koncertu

ORDER BY tab.Id\_Koncert

SQL Statements

1 SELECT \* FROM Bilans\_koncertow

2

<

Results

	Id_Koncert	Data_koncert	Bilans	
1	1	2016-01-01	450.00	
2	2	2016-02-05	270.00	
3	3	2016-03-09	465.00	
4	4	2016-03-25	600.00	
5	5	2016-04-24	640.00	
6	6	2016-05-20	50.00	
7	7	2016-06-09	460.00	
8	8	2016-07-29	350.00	
9	9	2016-08-20	300.00	
10	10	2016-09-17	420.00	
11	11	2016-10-16	365.00	
12	12	2016-11-10	490.00	
13	13	2016-12-06	200.00	
14	14	2017-03-10	200.00	

2. Placa\_pracownikow – Widok przedstawia płacę wszystkich muzyków i solistów pracujących obecnie dla filharmonii

```
ALTER VIEW "DBA"."Placa_pracownikow"(Pesel,Imie,Nazwisko,Partia,Placa)
```

```
AS
```

```
SELECT M.PeselM,M.ImieM,M.NazwiskoM,M.PartiaM,CONVERT(Numeric(5,2),150.00) AS  
Zarobki_w_zl
```

```
FROM Muzyk M
```

```
WHERE M.Data_do_M IS NULL
```

```
AND DATEDIFF(MONTH,M.Data_od_M, NOW()) <3
```

```
UNION
```

```
SELECT M.PeselM, M.ImieM,M.NazwiskoM,M.PartiaM,CONVERT(Numeric(5,2),200.00)
```

```
FROM Muzyk M
```

```
WHERE M.Data_do_M IS NULL
```

```
AND DATEDIFF(MONTH,M.Data_od_M, NOW()) BETWEEN 3 AND 4
```

```
UNION
```

```
SELECT M.PeselM, M.ImieM,M.NazwiskoM,M.PartiaM, CONVERT(Numeric(5,2),300.00)
```

```
FROM Muzyk M
```

```
WHERE M.Data_do_M IS NULL
```

```
AND DATEDIFF(MONTH,M.Data_od_M, NOW()) BETWEEN 5 AND 6
```

```
UNION
```

```
SELECT M.PeselM, M.ImieM,M.NazwiskoM,M.PartiaM, CONVERT(Numeric(5,2),350.00)
```

```
FROM Muzyk M
```

```
WHERE M.Data_do_M IS NULL
```

```
AND DATEDIFF(MONTH,M.Data_od_M, NOW()) BETWEEN 7 AND 9
```

```
UNION
```

```
SELECT M.PeselM, M.ImieM,M.NazwiskoM,M.PartiaM, CONVERT(Numeric(5,2),400.00)
```

```
FROM Muzyk M
```

```
WHERE M.Data_do_M IS NULL
```

AND DATEDIFF(MONTH,M.Data\_od\_M, NOW()) >= 10

UNION

SELECT S.PeselSol, S.ImieSol, S.NazwiskoSol,S.PartiaSol, CONVERT(Numeric(5,2),700.00)

FROM Solista S

WHERE S.Data\_do\_Sol IS NULL

	Pesel	Imie	Nazwisko	Partia	Placa
1	66092312485	Kinga	Kwiatkowska	solo_fortepian	700.00
2	69100912458	Anna	Mucha	solo_sopran	700.00
3	86111198900	Agnieszka	Dawidziuk	solo_alt	700.00
4	88031012485	Dawid	Pekiel	solo_tenor	700.00
5	87031545818	Jacek	Sztos	solo_bas	700.00
6	66092312469	Jan	Dudek	pskrzypce	400.00
7	88111101358	Jerzy	Krol	dskrzypce	400.00
8	66121201358	Jolanta	Zielicka	wiolonczela	400.00
9	74101212458	Marek	Pawlak	kontrabas	400.00
10	81110312469	Joanna	Jankowska	flet	400.00
11	79121401358	Magdalena	Wojciechowska	oboj	400.00
12	92122310000	Roman	Rutkowski	fagot	400.00
13	90041901346	Danuta	Mazur	fagot	400.00
14	76050301373	Henryk	Baran	trabka	400.00
15	65071712485	Kazimierz	Szewczyk	tuba	400.00
16	91100911374	Dorota	Nowicka	perkusja	400.00
17	87031911374	Marianna	Adamczyk	organy	400.00
18	78101012485	Beata	Adamczyk	dyrygent	400.00
19	90031012470	Katarzyna	Winiewska	pskrzypce	350.00
20	87031911358	Piotr	Wieczorek	pskrzypce	350.00
21	78061212458	Beata	Dabrowska	pskrzypce	350.00
22	78010111369	Krystyna	Wojcik	dskrzypce	350.00
23	77020111391	Tomasz	Majewski	dskrzypce	350.00
24	91031012470	Ewa	Kowalczyk	altowka	350.00
25	67021212469	Jan	Jaworski	altowka	350.00
26	72061412469	Marek	Malinowski	wiolonczela	350.00
27	87120312469	Janina	Malinowski	wiolonczela	350.00

3. Wykonawcy - Widok wyświetla wszystkich wykonawców, którzy zostaliby w danym momencie zatrudnieni do wykonania poszczególnych utworów.

```
ALTER VIEW "DBA"."Wykonawcy"(Id_Utwor,Tytul,Kompozytor,ImieW,NazwiskoW,PeselW,PartiaW)
```

```
AS
```

```
SELECT U.Id_Utwor,U.Tytul,U.Kompozytor,M.ImieM,M.NazwiskoM,M.PeselM,M.PartiaM
```

```
FROM Utwor U
```

```
INNER JOIN Partie_w_utworze PwU
```

```
ON U.Id_Utwor=PwU.Id_Utwor
```

```
INNER JOIN Wymagana_partia WP
```

```
ON WP.Nazwa_partii=PwU.Nazwa_partii
```

```
INNER JOIN Muzyk_utwory MU
```

```
ON MU.Id_Utwor=U.Id_Utwor
```

```
INNER JOIN Muzyk M
```

```
ON WP.Nazwa_partii=M.PartiaM
```

```
AND MU.PeselM=M.PeselM
```

```
WHERE M.Data_do_M IS NULL
```

```
UNION
```

```
SELECT U.Id_Utwor,U.Tytul,U.Kompozytor,S.ImieSol,S.NazwiskoSol,S.PeselSol,S.PartiaSol
```

```
FROM Utwor U
```

```
INNER JOIN Partie_w_utworze PwU
```

```
ON U.Id_Utwor=PwU.Id_Utwor
```

```
INNER JOIN Wymagana_partia WP
```

```
ON WP.Nazwa_partii=PwU.Nazwa_partii
```

```
INNER JOIN Solista_utwory SU
```

```
ON U.Id_Utwor=SU.Id_Utwor
```

INNER JOIN Solista S

ON SU.PeselSol=S.PeselSol

AND S.PartiaSol=WP.Nazwa\_partii

WHERE S.Data\_do\_Sol IS NULL

SQL Statements

1

SELECT \* FROM Wykonawcy WHERE Id\_Utwor=1

2

Results

	Id_Utwor	Tytul	Kompozytor	ImieW	NazwiskoW	PeselW	PartiaW
1	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Jan	Dudek	66092312469	pskrzypce
2	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Katarzyna	Winiewska	90031012470	pskrzypce
3	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Piotr	Wieczorek	87031911358	pskrzypce
4	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Beata	Dabrowska	78061212458	pskrzypce
5	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Jerzy	Krol	88111101358	dskrzypce
6	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Krystyna	Wojcik	78010111369	dskrzypce
7	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Tomasz	Majewski	77020111391	dskrzypce
8	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Ewa	Kowalczyk	91031012470	altowka
9	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Jan	Jaworski	67021212469	altowka
10	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Jolanta	Zielicka	66121201358	wiolonc...
11	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Marek	Malinowski	72061412469	wiolonc...
12	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Janina	Woniak	87120212469	wiolonc...
13	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Marek	Pawlak	74101212458	kontrabas
14	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	Beata	Adamczyk	78101012485	dyrygent

4. Aktualni\_w\_PSK – Wyświetla liczbę muzyków (i ich partię) wchodzących w skład poszczególnych PSK (pomocniczy do procedury „Utwór do repertuaru”)

```
ALTER VIEW "DBA"."Aktualni_w_PSK"(PSK,Obsada,Partia,Liczba_muzykow)
```

```
AS
```

```
SELECT PSK.Id_PSK,PSK.Id_obsada,M.PartiaM, COUNT(M.PartiaM) AS Lmuzykow
```

```
FROM Pelen_sklad_koncertowy PSK
```

```
INNER JOIN Obsada O
```

```
ON PSK.Id_obsada=O.Id_obsada
```

```
INNER JOIN Muzycy_w_obsadzie MwO
```

```
ON O.Id_obsada=MwO.Id_obsada
```

```
INNER JOIN Muzyk M
```

```
ON M.PeselM=MwO.PeselM AND M.Data_do_M IS NULL
```

```
GROUP BY PSK.Id_PSK,PSK.Id_obsada,M.PartiaM
```

```
UNION
```

```
SELECT PSK.Id_PSK,PSK.Id_obsada,S.PartiaSol, COUNT(S.PartiaSol)
```

```
FROM Pelen_sklad_koncertowy PSK
```

```
INNER JOIN Solisci_w_skladzie SwS
```

```
ON PSK.Id_PSK=SwS.Id_PSK
```

```
INNER JOIN Solista S
```

```
ON SwS.PeselSol=S.PeselSol AND S.Data_do_Sol IS NULL
```

```
GROUP BY PSK.Id_PSK,PSK.Id_obsada,S.PartiaSol
```

## SQL Statements

```
1 SELECT * FROM Aktualni_w_PSK WHERE PSK=1  
2
```

&lt;

## Results

	PSK	Obsada	Partia	Liczba_muzykow
1	1	4	pskrzypce	4
2	1	4	puzon	1
3	1	4	alt	3
4	1	4	kontrabas	1
5	1	4	flet	2
6	1	4	bas	2
7	1	4	perkusja	2
8	1	4	tenor	2
9	1	4	fagot	2
10	1	4	wiolonczela	3
11	1	4	dyrygent	1
12	1	4	altowka	2
13	1	4	waltornia	2
14	1	4	tuba	1
15	1	4	klarnet	1
16	1	4	sopran	3
17	1	4	oboj	2
18	1	4	trabka	2
19	1	4	dskrzypce	3
20	1	4	solo_sopran	1
21	1	4	solo_fort...	1
22	1	4	solo_bas	1
23	1	4	solo_tenor	1
24	1	4	solo_alt	1

## Procedury

1. Czy\_wykonywalny - Procedura ta sprawdza, czy możliwym jest wykonanie wskazanego utworu przy obecnym składzie wykonawczym. Jeżeli nie – wypisuje jakich partii brakuje i w jakiej liczbie.

```
ALTER PROCEDURE "DBA"."Czy_wykonywalny"(IN Nr_utworu INT)

BEGIN

    DECLARE L_wymaganych_partii INT;

    DECLARE L_partii_speln INT;

    DECLARE It_Nazwa_partii VARCHAR(30);

    DECLARE Tytul VARCHAR(30);

    DECLARE Kompozytor VARCHAR (30);

    DECLARE It_Liczba_muzykow INT;

    DECLARE L_dostepni_muzycy INT;

    DECLARE L_brakujacych INT;

    DECLARE Iterator CURSOR FOR

        (SELECT WP.Nazwa_partii, PwU.Ilu_muzykow

        FROM Wymagana_partia WP

        INNER JOIN Partie_w_utworze PwU

        ON WP.Nazwa_partii = PwU.Nazwa_partii

        AND PwU.Id_Utwor = Nr_utworu);

    SET L_partii_speln = 0;

    SET Tytul=(SELECT Tytul FROM Utwor WHERE Id_Utwor=Nr_utworu);

    SET Kompozytor=(SELECT Kompozytor FROM Utwor WHERE Id_Utwor=Nr_utworu);

    SET L_wymaganych_partii = (SELECT COUNT(PwU.Id_Utwor)

        FROM Partie_w_utworze PwU

        WHERE PwU.Id_Utwor=Nr_utworu);

    OPEN iterator;

    petla: LOOP

        FETCH NEXT Iterator INTO It_Nazwa_partii, It_Liczba_muzykow;
```

```

IF (SQLCODE !=0) THEN LEAVE petla ENDIF;

SET L_dostepni_muzycy = (SELECT COUNT(*)

                        FROM Wykonawcy W

                        WHERE W.Id_Utwor=Nr_utworu

                        AND W.PartiaW=It_Nazwa_partii);

IF (L_dostepni_muzycy < It_Liczba_muzykow) THEN

    SET L_brakujacych = It_Liczba_muzykow - L_dostepni_muzycy;

    MESSAGE 'Brakuje ' + CONVERT (CHAR, L_brakujacych) + ' muzykow wykonujacych partie ' +
It_Nazwa_partii TO CLIENT;

ELSE

    SET L_partii_speln = L_partii_speln + 1;

ENDIF;

END LOOP;

CLOSE iterator;

IF (L_partii_speln = L_wymaganych_partii) THEN

    MESSAGE 'Utwor ' + Tytul + ' ' + Kompozytor + ' moze zostac wykonany' TO CLIENT;

ELSE MESSAGE 'Utwor ' + Tytul + ' ' + Kompozytor + ' nie moze zostac wykonany' TO CLIENT;

ENDIF;

END

```

SQL Statements	
1	CALL Czy_wykonywalny(6)
2	
<	
Results	
Brakuje 1 muzykow wykonujacych partie kontrabas	
Utwor Symfonia IX Ludwig van Beethoven nie moze zostac wykonany	
1 row(s) affected	
Execution time: 0.071 seconds	
Procedure completed	

SQL Statements	
1	CALL Czy_wykonywalny(9)
2	
<	
Results	
Utwor Te Deum Anton Bruckner moze zostac wykonany	
1 row(s) affected	
Execution time: 0.105 seconds	
Procedure completed	

2. Utwór\_do\_repertuaru(Id\_Utwor,Id\_Koncert) - Procedura sprawdza, czy dany utwór może zostać wykonany przez skład przeznaczony do wykonania danego utworu. Najpierw sprawdza czy koncert się jeszcze nie odbył (wtedy nie można zmienić repertuaru), następnie porównuje wymagane partie w danym utworze z partiami będącymi w określonym dla danego koncertu składzie.

```
ALTER PROCEDURE "DBA"."Utwór_do_repertuaru"(IN Nr_utworu INTEGER , IN Nr_koncertu  
INTEGER)
```

```
BEGIN
```

```
    DECLARE PSK_koncertu INT;
```

```
    DECLARE flag INT;
```

```
    DECLARE It_Nazwa_partii VARCHAR (20);
```

```
    DECLARE Czas INT;
```

```
    DECLARE Tytuł VARCHAR(30);
```

```
    DECLARE Kompozytor VARCHAR(30);
```

```
    DECLARE Powtorka INT;
```

```
    DECLARE Czas_trwania INT;
```

```
    DECLARE Iterator CURSOR FOR
```

```
        (SELECT PwU.Nazwa_partii
```

```
            FROM Utwor U
```

```
            INNER JOIN Partie_w_utworze PwU
```

```
            ON U.Id_Utwor=PwU.Id_Utwor
```

```
            WHERE U.Id_Utwor=Nr_utworu);
```

```
////////Warunek: Zgodność czasu wykonania
```

```
    SET Czas = (SELECT DATEDIFF(DAY,NOW(),K.Data_koncertu)
```

```
        FROM Koncert K
```

```
        WHERE K.Id_Koncert=Nr_koncertu);
```

```
    IF(Czas < 0) THEN MESSAGE 'Koncert juz sie odbyl, nie mozesz zmienic jego repertuaru.' TO CLIENT;
```

```
    ELSE    //koncert się jeszcze nie odbył - można zmienić repertuar
```

```

SET Powtorka = (SELECT COUNT(*)

FROM Repertuar R

WHERE R.Id_Koncert=Nr_koncertu

AND R.Id_Utwor=Nr_utworu);

SET Czas_trwania = (SELECT SUM(U.Czas_trwania_min)

FROM Utwor U

INNER JOIN Repertuar R

ON U.Id_Utwor=R.Id_Utwor

INNER JOIN Koncert K

ON K.Id_Koncert=R.Id_Koncert

WHERE K.Id_Koncert=Nr_koncertu) +

(SELECT Czas_trwania_min

FROM Utwor

WHERE Id_Utwor = Nr_utworu);

IF(Powtorka = 1) THEN

MESSAGE 'Utwor juz jest w repertuarze.' TO CLIENT;

ELSEIF(Czas_trwania > 150) THEN

MESSAGE 'Nie mozna wstawic utworu - koncert bylby zbyt dlugi' TO CLIENT;

ELSE

SET PSK_koncertu = (SELECT K.Id_PSK

FROM Koncert K

WHERE K.Id_Koncert=Nr_koncertu);

SET Tytul = (SELECT U.Tytul

FROM Utwor U

WHERE U.Id_Utwor=Nr_utworu);

SET Kompozytor = (SELECT U.Kompozytor

FROM Utwor U

```

```

        WHERE U.Id_Utwor=Nr_utworu);

SET flag = 1;

//początek transakcji

BEGIN

    INSERT INTO Repertuar

        VALUES(Nr_koncertu,Nr_utworu);

    OPEN iterator;

    petla: LOOP

        FETCH NEXT Iterator INTO It_Nazwa_partii;

        IF (SQLCODE != 0 OR flag = 0) THEN LEAVE petla ENDIF;

        SET flag = ( SELECT COUNT(*)

            FROM Aktualni_w_PSK A

                WHERE A.PSK=PSK_koncertu

                    AND A.Partia=It_Nazwa_partii);

    END LOOP;

    CLOSE iterator;

    IF (flag=0) THEN

        MESSAGE 'Utwor przeznaczony jest na inna obsade niz wybrana do wykonania tego
koncertu.' TO CLIENT;

        ROLLBACK;

    ELSE

        MESSAGE 'Dodano utwor: ' + Tytul + ' - ' + Kompozytor + ' do repertuaru' TO CLIENT;

        COMMIT;

    ENDIF;

END;

ENDIF;

ENDIF;

END

```

SQL Statements	
1	CALL Utwor_do_repertuaru(1,2)
2	
<	
▲▼	
Results	
Koncert juz sie odbyl, nie mozesz zmienic jego repertuaru.	
Execution time: 0.053 seconds	
Procedure completed	

SQL Statements	
1	CALL Utwor_do_repertuaru(1,23)
	<
▲▼	
Results	
Utwor przeznaczony jest na inna obsade niz wybrana do wykonania tego koncertu.	
1 row(s) affected	
Execution time: 0.028 seconds	
Procedure completed	

SQL Statements	
1	CALL Utwor_do_repertuaru(1,14)
	<
▲▼	
Results	
Utwor juz jest w repertuarze.	
Execution time: 0.006 seconds	
Procedure completed	

SQL Statements	
1	CALL Utwor_do_repertuaru(4,23)
	<
▲▼	
Results	
Dodano utwor: Nokturny - Fryderyk Chopin do repertuaru	
1 row(s) affected	
Execution time: 0.025 seconds	
Procedure completed	

SQL Statements	
1	CALL Utwor_do_repertuaru(1,18)
	<
▲▼	
Results	
Nie mozna wstawic utworu - koncert bylby zbyt dlugi	
Execution time: 0.146 seconds	
Procedure completed	

Zawartość tabel po wykonaniu procedur

Results				
	Id_Utwor	Tytul	Kompozytor	Czas_trwania_min
1	1	Eine Kleine Nachtmusik	Wolfgang Amadeusz Mozart	20
2	2	Der Tod und das Madchen	Franz Schubert	40
3	3	Koncert fortepianowy e-moll	Fryderyk Chopin	40
4	4	Nokturny	Fryderyk Chopin	120
5	5	Symfonia V	Ludwig van Beethoven	35
6	6	Symfonia IX	Ludwig van Beethoven	65
7	7	Koncert fortepianowy b-moll	Piotr Czajkowski	40
8	8	Te Deum	Anoine Charpentier	25
9	9	Te Deum	Anton Bruckner	25
10	10	Christus factus est	Anton Bruckner	5
11	11	Locus iste	Anton Bruckner	4
12	12	Crucem Tuam	Pawel Lukaszewski	4
13	13	Nunc dimittis	Pawel Lukaszewski	4
14	14	Vexilla regis	Anton Bruckner	5
15	15	Sicut cervus	Giovanni da Palestrina	3
16	16	O Crux ave	Giovanni da Palestrina	3
17	17	Koncert na chór e-moll	Siergiej Rachmaninov	9
18	18	Msza Berlinska	Arvo Part	28

SQL Statements

2 SELECT \* FROM Repertuar WHERE Id\_Koncert>13

<

Results

	Id_Koncert	Id_Utwor
1	14	2
2	14	5
3	18	6
4	14	1
5	18	7
6	18	18
7	23	4

Results				
	Id_Koncert	Id_PSK	Id_Sala	Data_koncertu
1	1	1	1	2016-01-01
2	2	2	2	2016-02-05
3	3	3	3	2016-03-09
4	4	4	2	2016-03-25
5	5	5	1	2016-04-24
6	6	3	1	2016-05-20
7	7	1	1	2016-06-09
8	8	2	1	2016-07-29
9	9	3	1	2016-08-20
10	10	4	2	2016-09-17
11	11	3	1	2016-10-16
12	12	5	1	2016-11-10
13	13	2	1	2016-12-06
14	14	2	2	2017-03-10
15	18	1	1	2017-04-01
16	23	4	1	2017-06-01

3. Wynagrodzenie - Procedura aktualizuje rekord w tabeli z koncertami o wartość wynagrodzenia wypłaconego wykonawcom za dany koncert.

```
ALTER PROCEDURE "DBA"."Wynagrodzenie"(IN Nr_Koncertu INT)
```

```
BEGIN
```

```
    DECLARE Suma Numeric (8,2);
```

```
    DECLARE W_Prog1 Numeric (8,2);
```

```
    DECLARE W_Prog2 Numeric (8,2);
```

```
    DECLARE W_Prog3 Numeric (8,2);
```

```
    DECLARE W_Prog4 Numeric (8,2);
```

```
    DECLARE W_Prog5 Numeric (8,2);
```

```
    DECLARE W_Solisci Numeric (8,2);
```

```
    SET W_Prog1 = 150 * (SELECT COUNT(*)
```

```
        FROM Koncert K
```

```
        INNER JOIN Pelen_sklad_koncertowy PSK
```

```
            ON K.Id_PSK=PSK.Id_PSK
```

```
        INNER JOIN Obsada O
```

```
            ON PSK.Id_obsada=O.Id_obsada
```

```
        INNER JOIN Muzycy_w_obsadzie MwO
```

```
            ON MwO.Id_obsada=O.Id_obsada
```

```
        INNER JOIN Muzyk M
```

```
            ON MwO.PeselM=M.PeselM
```

```
        AND DATEDIFF(MONTH,M.Data_od_M,K.Data_koncertu) < 3
```

```
        AND (M.Data_do_M IS NULL
```

```
            OR DATEDIFF(DAY,K.Data_koncertu,M.Data_do_M) > 0)
```

```
        WHERE K.Id_Koncert=Nr_Koncertu);
```

```
    SET W_Prog2 = 250 * (SELECT COUNT(*)
```

```

FROM Koncert K

INNER JOIN Pelen_sklad_koncertowy PSK

    ON K.Id_PSK=PSK.Id_PSK

INNER JOIN Obsada O

    ON PSK.Id_obsada=O.Id_obsada

INNER JOIN Muzycy_w_obsadzie MwO

    ON MwO.Id_obsada=O.Id_obsada

INNER JOIN Muzyk M

    ON MwO.PeselM=M.PeselM

    AND DATEDIFF(MONTH,M.Data_od_M,K.Data_koncertu) BETWEEN 3 AND 4

    AND (M.Data_do_M IS NULL

        OR DATEDIFF(DAY,K.Data_koncertu,M.Data_do_M) > 0)

WHERE K.Id_Koncert=Nr_Koncertu);

```

```

SET W_Prog3 = 300 * (SELECT COUNT(*)

FROM Koncert K

INNER JOIN Pelen_sklad_koncertowy PSK

    ON K.Id_PSK=PSK.Id_PSK

INNER JOIN Obsada O

    ON PSK.Id_obsada=O.Id_obsada

INNER JOIN Muzycy_w_obsadzie MwO

    ON MwO.Id_obsada=O.Id_obsada

INNER JOIN Muzyk M

    ON MwO.PeselM=M.PeselM

    AND DATEDIFF(MONTH,M.Data_od_M,K.Data_koncertu) BETWEEN 5 AND 6

    AND (M.Data_do_M IS NULL

        OR DATEDIFF(DAY,K.Data_koncertu,M.Data_do_M) > 0)

```

WHERE K.Id\_Koncert=Nr\_Koncertu);

SET W\_Prog4 = 350 \* (SELECT COUNT(\*)

FROM Koncert K

INNER JOIN Pelen\_sklad\_koncertowy PSK

ON K.Id\_PSK=PSK.Id\_PSK

INNER JOIN Obsada O

ON PSK.Id\_obsada=O.Id\_obsada

INNER JOIN Muzycy\_w\_obsadzie MwO

ON MwO.Id\_obsada=O.Id\_obsada

INNER JOIN Muzyk M

ON MwO.PeselM=M.PeselM

AND DATEDIFF(MONTH,M.Data\_od\_M,K.Data\_koncertu) BETWEEN 7 AND 9

AND (M.Data\_do\_M IS NULL

OR DATEDIFF(DAY,K.Data\_koncertu,M.Data\_do\_M) > 0)

WHERE K.Id\_Koncert=Nr\_Koncertu);

SET W\_Prog5 = 400 \* (SELECT COUNT(\*)

FROM Koncert K

INNER JOIN Pelen\_sklad\_koncertowy PSK

ON K.Id\_PSK=PSK.Id\_PSK

INNER JOIN Obsada O

ON PSK.Id\_obsada=O.Id\_obsada

INNER JOIN Muzycy\_w\_obsadzie MwO

ON MwO.Id\_obsada=O.Id\_obsada

INNER JOIN Muzyk M

ON MwO.PeselM=M.PeselM

```

AND DATEDIFF(MONTH,M.Data_od_M,K.Data_koncertu) >= 10

AND (M.Data_do_M IS NULL

OR DATEDIFF(DAY,K.Data_koncertu,M.Data_do_M) > 0)

WHERE K.Id_Koncert=Nr_Koncertu);

```

```

SET W_Solisci = 700 * (SELECT COUNT(*)

FROM Koncert K

INNER JOIN Pelen_sklad_koncertowy PSK

ON K.Id_PSK=PSK.Id_PSK

INNER JOIN Solisci_w_skladzie SwS

ON PSK.Id_PSK=SwS.Id_PSK

WHERE K.Id_Koncert=Nr_Koncertu);

```

```

SET Suma = W_Prog1 + W_Prog2 + W_Prog3 + W_Prog4 + W_Prog5 + W_Solisci;

UPDATE Koncert

SET Koncert.Wynagrodzenie_muzykow = Suma

WHERE Koncert.Id_Koncert = Nr_Koncertu;

END

```

Zmiana w tabeli koncert po wywołaniu procedury dla koncertu nr 14:

14	14	2	2017-03-10	10,750.00	50.00
----	----	---	------------	-----------	-------

## Funkcje

1. Dofinansowanie\_utworu - Funkcja oblicza ile średnio pieniędzy przeznaczają sponsorzy na koncerty, w których wykonywany jest utwór o indeksie przekazanym jako parametr.

```
ALTER FUNCTION "DBA"."Dofinansowanie_utworu"(IN Nr_utworu INT)
```

```
RETURNS NUMERIC(8,2)
```

```
NOT DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE "Kwota" NUMERIC(8,2);
```

```
    DECLARE "Suma_wplat" INT;
```

```
    DECLARE "Liczba_koncertow" INT;
```

```
    SET Suma_wplat = (SELECT SUM(WnK.Kwota)
```

```
        FROM Utwor U
```

```
        INNER JOIN Repertuar R
```

```
        ON U.Id_Utwor=R.Id_Utwor
```

```
        INNER JOIN Koncert K
```

```
        ON K.Id_Koncert=R.Id_Koncert
```

```
        INNER JOIN Wplata_na_koncert WnK
```

```
        ON WnK.Id_Koncert=K.Id_Koncert
```

```
        WHERE U.Id_Utwor = Nr_utworu);
```

```
    SET Liczba_koncertow = (SELECT COUNT(R.Id_Koncert)
```

```
        FROM Utwor U
```

```
        INNER JOIN Repertuar R
```

```
        ON U.Id_Utwor=R.Id_Utwor
```

```
        WHERE U.Id_Utwor = Nr_utworu);
```

```
    SET Kwota = Suma_wplat/Liczba_koncertow;
```

```
    RETURN "Kwota";
```

```
END
```

# SQL Statements

```
1 SELECT Dofinansowanie_utworu(Id_Utwor) FROM Utwor
```

## Results

	Dofinansowanie_utworu(Utwor.Id_Utwor)
1	7,360.00
2	5,875.00
3	9,550.00
4	1,200.00
5	7,360.00
6	8,925.00
7	9,020.00
8	13,300.00
9	13,000.00
10	11,900.00
11	11,900.00
12	11,900.00
13	11,900.00
14	11,900.00
15	11,900.00
16	11,900.00
17	11,900.00
18	10,283.00

2. Premia - Funkcja oblicza, na podstawie przychodów z poszczególnych koncertów oraz liczby muzyków je wykonujących, premię należną dla wykonawcy od daty ostatniego jej zliczenia, po czym przelewa wyliczoną ilość pieniędzy na konto muzyka.

```
ALTER FUNCTION "DBA"."Premia"(IN Pesel CHAR(11))

RETURNS NUMERIC(8,2)

NOT DETERMINISTIC

BEGIN

    DECLARE "Kwota" NUMERIC(8,2);

    DECLARE "Data_pp" DATE; //Data od której koncerty bierzemy pod uwagę

    DECLARE "Roznica_dat" INT;

    DECLARE "Licznik" INT;

    DECLARE "Czy_pobierał_premie" INT;

    DECLARE "Brak_nowych_koncertow" INT;

    DECLARE "Liczba_muzykow" INT;

    DECLARE "Piniondz" NUMERIC(8,2);

    DECLARE "Stara_premia" NUMERIC(8,2);

    DECLARE "Premia_koncert" NUMERIC(8,2);

    DECLARE "Stary_stan_k" NUMERIC(8,2);

    DECLARE "Nowy_stan_k" NUMERIC(8,2);

    DECLARE "It_Nr_koncertu" INT;

    DECLARE "It_Data_koncertu" DATE;

    DECLARE "Iterator" CURSOR FOR //Wszystkie koncerty w których muzyk brał udział

        (SELECT K.Id_Koncert,K.Data_koncertu

        FROM Muzyk M

        INNER JOIN Muzycy_w_obsadzie MwO

        ON M.PeselM=MwO.PeselM

        AND M.PeselM=Pesel
```

```

INNER JOIN Obsada O

    ON O.Id_obsada=MwO.Id_obsada

INNER JOIN Pelen_sklad_koncertowy PSK

    ON PSK.Id_obsada=O.Id_obsada

INNER JOIN Koncert K

    ON K.Id_PSK=PSK.Id_PSK

    AND DATEDIFF(DAY,M.Data_od_M,K.Data_koncertu)>0

    AND DATEDIFF(DAY,K.Data_koncertu,ISNULL(M.Data_do_M,NOW())) > 0

    ORDER BY K.Id_Koncert DESC);

SET Kwota = 0;

SET Licznik=0;

SET Stara_premia = ISNULL((SELECT M.PremiaM FROM Muzyk M WHERE M.PeselM=Pesel),0);


SET Czy_pobieral_premie = (SELECT COUNT(*)

    FROM Budzet B

    WHERE B.Rodzaj_tranzakcji=Pesel);

IF(Czy_pobieral_premie = 0) THEN

    SET Data_pp = (SELECT M.Data_od_M

        FROM Muzyk M

        WHERE M.PeselM = Pesel);

ELSE

    SET Data_pp = (SELECT TOP 1 B.DataB

        FROM Budzet B

        WHERE B.Rodzaj_tranzakcji = Pesel

        ORDER BY B.DataB DESC);

ENDIF;

```

```

SET Brak_nowych_koncertow = 0;

SET Roznica_dat = 0;

//Pętla

OPEN iterator;

petla: LOOP

    FETCH NEXT Iterator INTO It_Nr_koncertu, It_Data_koncertu;

    IF (SQLCODE != 0 OR Roznica_dat < 0 OR Brak_nowych_koncertow = 1) THEN LEAVE petla ENDIF;

    SET Roznica_dat = (SELECT DATEDIFF(DAY, Data_pp, It_Data_koncertu));

    IF (Roznica_dat > 0) THEN

        SET Liczba_muzykow =      //Liczba muzyków wykonujących poszczególne koncerty
        (SELECT COUNT(M.PeselM)

            FROM Koncert K

            INNER JOIN Pelen_sklad_koncertowy PSK

            ON PSK.Id_PSK=K.Id_PSK

            AND K.Id_Koncert = It_Nr_koncertu

            INNER JOIN Obsada O

            ON O.Id_obsada=PSK.Id_obsada

            INNER JOIN Muzycy_w_obsadzie MwO

            ON MwO.Id_obsada=O.Id_obsada

            INNER JOIN Muzyk M

            ON M.PeselM=MwO.PeselM

            AND DATEDIFF(DAY, K.Data_koncertu, M.Data_od_M) <= 0

            AND DATEDIFF(DAY, K.Data_koncertu, ISNULL(M.Data_do_M, NOW())) >= 0);

        SET Piniondz = (SELECT BK.Bilans      //zysk filharmonii z danego koncertu

            FROM Bilans_koncertow BK

```

```
WHERE BK.Id_Koncert = It_Nr_koncertu);
```

```
SET Premia_koncert = Piniondz/(2*Liczba_muzykow);
```

```
SET Kwota = Kwota + Premia_koncert;
```

```
SET Licznik=Licznik+1;
```

```
ENDIF;
```

```
IF(Licznik=0 AND Roznica_dat<0) THEN SET Brak_nowych_koncertow = 1;
```

```
ENDIF;
```

```
END LOOP;
```

```
CLOSE iterator;
```

```
IF(Brak_nowych_koncertow = 1) THEN MESSAGE 'Od ostatniego przeliczania premii muzyk nie bral  
udzialu w koncercie' TO CLIENT;
```

```
ELSE
```

```
SET Stary_stan_k = (SELECT TOP 1 Stan_konta
```

```
FROM Budzet
```

```
ORDER BY DataB DESC);
```

```
//transakcja:
```

```
BEGIN
```

```
UPDATE Muzyk
```

```
SET Muzyk.PremiaM = Stara_premia + Kwota
```

```
WHERE Muzyk.PeselM=Pesel;
```

```
IF(Stary_stan_k < Kwota) THEN
```

```
MESSAGE 'Operacja nie powiodla sie, zbyt malo srodkow na koncie' TO CLIENT;
```

```
ROLLBACK;
```

```
ELSE
```

```
MESSAGE 'Operacja powiodla sie' TO CLIENT;
```

```

SET Nowy_stan_k = Stary_stan_k - Kwota;

INSERT INTO Budzet(Stan_konta,DataB,KwotaB,Rodzaj_tranzakcji)

VALUES(Nowy_stan_k,NOW(),-Kwota,Pesel);

COMMIT;

ENDIF;

END;

ENDIF;

RETURN "Kwota";

END

```

Stan przed wykonaniem:

SQL Statements					
<pre> 5 SELECT TOP 5 * FROM Budzet ORDER BY Id_Budzet DESC </pre>					
Results					
	Id_Budzet	Stan_konta	DataB	KwotaB	Rodzaj_tranzakcji
1	55	5,424.92	2017-02-19	-280.00	Wycofanie koncertu 20
2	54	5,704.92	2017-02-19	40.00	Kupno biletu przez 79031466663 na koncert nr 20
3	53	5,664.92	2017-02-19	40.00	Kupno biletu przez 58120311348 na koncert nr 20
4	51	5,624.92	2017-02-19	40.00	Kupno biletu przez 76050301309 na koncert nr 20
5	49	5,624.92	2017-02-19	40.00	Kupno biletu

SQL Statements		
<pre> 1 SELECT PeselM, PremiaM 2 FROM Muzyk 3 WHERE Data_do_M IS NOT NULL 4 AND PartiaM = 'altowka' </pre>		
Results		
	PeselM	PremiaM
1	89101009946	0.00
2	69123012469	0.00

Stan po jednym wykonaniu:

Stan po powtórny wykonaniu:

SQL Statements		SQL Statements	
1	SELECT PeselM, PremiaM, Premia (PeselM)	1	SELECT PeselM, PremiaM, Premia (PeselM)
2	FROM Muzyk	2	FROM Muzyk
3	WHERE Data_do_M IS NOT NULL	3	WHERE Data_do_M IS NOT NULL
4	AND PartiaM = 'altowka'	4	AND PartiaM = 'altowka'
<		<	
Results		Results	
PeselM	PremiaM	PesiaM	Premia(Muzyk.PesiaM)
1 89101009946	0.00	1 89101009946	26.56
2 69123012469	0.00	2 69123012469	26.56

SQL Statements	
1	SELECT TOP 5 * FROM Budzet ORDER BY Id_Budzet DESC
2	
<	
Results	
Id_Budzet	Stan_konta
1 57	5,611.80
2 56	5,638.36
3 55	5,424.92
4 54	5,704.92
5 53	5,664.92

3. Procent\_widowni\_na\_utworze - Zwraca jaki procent widowni jest średnio zajęty w czasie koncertów, gdy wykonywany jest wskazany w parametrze utwor

```
ALTER FUNCTION "DBA"."Procent_zajetej_widowni_na_utworze"( IN Nr_Utworu INT )
```

```
RETURNS CHAR(6)
```

```
NOT DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE "Procent" NUMERIC (4,2);
```

```
    SET Procent = (SELECT CONVERT(NUMERIC(4,2),CONVERT(FLOAT,LB.Lbiletow)/LM.LMiejsc*100)
```

```
        FROM
```

```
        (SELECT U.Id_Utwor,COUNT(B.Id_Bilet) AS Lbiletow
```

```
            FROM Utwor U
```

```
            INNER JOIN REPERTUAR R
```

```
            ON U.Id_Utwor=R.Id_Utwor
```

```
            INNER JOIN Koncert K
```

```
            ON K.Id_Koncert=R.Id_Koncert
```

```
            INNER JOIN Bilet B
```

```
            ON B.Id_Koncert=K.Id_Koncert
```

```
            GROUP BY U.Id_Utwor)LB,
```

```
        (SELECT U.Id_Utwor, COUNT(M.Id_Miejsc) AS LMiejsc
```

```
            FROM Utwor U
```

```
            INNER JOIN REPERTUAR R
```

```
            ON U.Id_Utwor=R.Id_Utwor
```

```
            INNER JOIN Koncert K
```

```
            ON K.Id_Koncert=R.Id_Koncert
```

```
            INNER JOIN Sala S
```

```
            ON K.Id_Sala=S.Id_Sala
```

```
            INNER JOIN Miejsce M
```

```
            ON M.Id_Sala=S.Id_Sala
```

```

GROUP BY U.Id_Utwor)LM

WHERE LM.Id_Utwor = LB.Id_Utwor

AND LM.Id_Utwor = Nr_Utworu);

MESSAGE 'Przy wykonaniu danego utworu ' + CONVERT(VARCHAR,Procent) + '% miejsc jest
zajetych' TO CLIENT;

RETURN "Procent";

END

```

Wykonanie dla wszystkich utworow:

	Procent zajętej widowni
1	72.76
2	70.66
3	90.00
4	41.79
5	72.76
6	58.93
7	66.29
8	80.00
9	75.71
10	78.57
11	78.57
12	78.57
13	78.57
14	78.57
15	78.57
16	78.57
17	78.57
18	64.52

## Wyzwalacze

1. Usun\_koncert - Usuwa nieodbyty koncert, a więc i wszystkie inne rekordy z nim związane: repertuar, wpłaty sponsorów, bilety a także cofa przychody w budżecie związane ze sponsorami i klientami.

```
ALTER TRIGGER "Usun_koncert" BEFORE DELETE
ORDER 1 ON "DBA"."Koncert"
REFERENCING OLD AS stary
FOR EACH ROW
BEGIN
    DECLARE Kwota_bilety NUMERIC(8,2);
    DECLARE Koszt_biletu NUMERIC(8,2);
    DECLARE Liczba_biletow INT;
    DECLARE Sponsoring NUMERIC(8,2);
    DECLARE Stary_budzet NUMERIC(8,2);
    DECLARE Nowy_budzet NUMERIC(8,2);
    DECLARE KwotaBudzet NUMERIC(8,2);
    DECLARE R_tranzakcji VARCHAR(100);
    IF(DATEDIFF(DAY,stary.Data_koncertu,NOW())<0) THEN
        SET Liczba_biletow = (SELECT COUNT(*)
                                FROM Bilet B
                                WHERE B.Id_Koncert=stary.Id_Koncert);
        SET Koszt_biletu = (SELECT K.Cena_biletu
                            FROM Koncert K
                            WHERE K.Id_Koncert=stary.Id_Koncert);
        SET Kwota_bilety = Liczba_biletow*Koszt_biletu;
        SET Sponsoring = (SELECT SUM(Kwota)
                            FROM Wplata_na_koncert
```

```

WHERE Id_Koncert=stary.Id_Koncert);

SET R_tranzakcji = 'Wycofanie koncertu ' + CONVERT(CHAR,stary.Id_Koncert);

SET Stary_budzet = (SELECT TOP 1 B.Stan_konta

FROM Budzet B

ORDER BY B.Id_Budzet DESC);

SET KwotaBudzet = Kwota_bilety + Sponsoring;

SET Nowy_budzet = Stary_budzet - Kwota_Budzet;

INSERT INTO Budzet(Stan_konta,DataB,KwotaB,Rodzaj_tranzakcji)

VALUES (Nowy_budzet,NOW(),-KwotaBudzet,R_tranzakcji);

DELETE FROM Bilet

WHERE Id_Koncert=stary.Id_Koncert;

DELETE FROM Repertuar

WHERE DBA.Repertuar.Id_Koncert=stary.Id_Koncert;

DELETE FROM Wplata_na_koncert

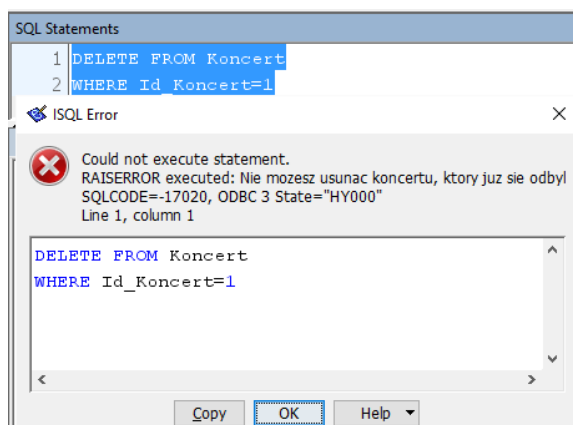
WHERE Id_Koncert=stary.Id_Koncert

ELSE RAISERROR 17020 'Nie mozesz usunac koncertu, ktory juz sie odbyl';

ENDIF;

END

```



Stan bazy przed:

SQL Statements

1

2

SELECT\* FROM Koncert ORDER BY Id\_Koncert DESC

Results

	Id_Koncert	Id_PSK	Id_Sala	Data_koncertu	Wynagrodzenie_muzykow	Cena_biletu
1	23	4	1	2017-06-01	(NULL)	30.00
2	18	1	1	2017-04-01	(NULL)	55.00
3	14	2	2	2017-03-10	10,750.00	50.00
4	13	2	1	2016-12-06	10,300.00	50.00

SQL Statements

1

2

SELECT\* FROM Repertuar WHERE Id\_Koncert =14

<

Results

	Id_Koncert	Id_Utwor
1	14	2
2	14	5
3	14	1

SQL Statements

1

2

SELECT \* FROM Wplata\_na\_koncert ORDER BY id\_WnK DESC

<

Results

	id_WnK	Id_Sponsor	Id_Koncert	Kwota	
1	46	1	14	2,000.00	
2	45	5	13	2,800.00	

Stan bazy po:

SQL Statements

1

2

-

<

SELECT \* FROM Wplata\_na\_koncert ORDER BY id\_WnK DESC

Results

	id_WnK	Id_Sponsor	Id_Koncert	Kwota
1	45	5	13	2,800.00
2	44	4	13	2,500.00
3	43	3	13	2,800.00

SQL Statements	
1	SELECT* FROM Repertuar WHERE Id_Koncert =14
2	
<	
Results	
Id_Koncert	Id_Utwor

SQL Statements

1

SELECT\* FROM Bilet ORDER BY Id\_Bilet DESC

2

<

Results

	Id_Bilet	Id_Koncert	Id_Miejsca	PeselSlu	
1	676	13	66	88121212383	
2	675	13	65	87120212420	
3	674	13	64	66111601336	
4	673	13	63	77020111306	
5	672	13	62	66111601235	

SQL Statements

1

SELECT\* FROM Budzet ORDER BY Id\_Budzet DESC

2

<

Results

	Id_Budzet	Stan_konta	DataB	KwotaB	Rodzaj_tranzakcji
1	58	3,411.80	2017-02-20	-2200.00	Wycofanie koncertu 14
2	57	5,611.80	2017-02-20	-26.56	69123012469
3	56	5,638.36	2017-02-20	-26.56	89101009946

SQL Statements

1 SELECT \* FROM Koncert ORDER BY Id\_Koncert DESC

<

Results

	Id_Koncert	Id_PSK	Id_Sala	Data_koncertu	Wynagrodzenie_muzykow	Cena_biletu
1	23	4	1	2017-06-01	(NULL)	30.00
2	18	1	1	2017-04-01	(NULL)	55.00
3	13	2	1	2016-12-06	10,300.00	50.00

2. Zakup\_biletu - Wyzwalacz sprawdza czy możliwym jest zakupienie biletu na dane miejsce na danym koncercie. Jeżeli nic nie stoi na przeszkodzie dokonuje zakupu i aktualizuje stan budżetu filharmonii.

```
ALTER TRIGGER "Zakup_biletu" BEFORE INSERT
```

```
ORDER 1 ON "DBA"."Bilet"
```

```
REFERENCING NEW AS Nowa
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE I_koncertu INT;
```

```
    DECLARE I_miejsca INT;
```

```
    DECLARE I_sali INT;
```

```
    DECLARE Pesel CHAR(11);
```

```
    DECLARE Czy_zajete INT;
```

```
    DECLARE Czy_sie_odbyl INT;
```

```
    DECLARE Czy_dobra_sala INT;
```

```
    DECLARE Koszt_biletu NUMERIC(6,2);
```

```
    DECLARE Stary_budzet Numeric(8,2);
```

```
    DECLARE Nowy_budzet Numeric(8,2);
```

```
    DECLARE R_tranzakcji VARCHAR(100);
```

```
    SET Pesel=Nowa.PeselSlu;
```

```
    SET I_koncertu = Nowa.Id_Koncert;
```

```
    SET I_miejsca = Nowa.Id_Miejsca;
```

```
    SET I_sali = (SELECT K.Id_Sala
```

```
        FROM Koncert K
```

```
        WHERE K.Id_Koncert=I_koncertu);
```

```
SET Czy_sie_odbyl = (SELECT DATEDIFF(DAY,K.Data_koncertu,NOW()))
```

```
FROM Koncert K
```

```
WHERE K.Id_Koncert=I_koncertu);
```

```
SET Czy_dobra_sala = (SELECT COUNT(*)
```

```
FROM Miejsce M
```

```
INNER JOIN Sala S
```

```
ON S.Id_Sala=I_sali
```

```
AND S.Id_Sala=M.Id_Sala
```

```
AND M.Id_Miejsca=I_miejsca);
```

```
SET Czy_zajete = (SELECT COUNT(*)
```

```
FROM Bilet
```

```
WHERE Id_Koncert=I_koncertu
```

```
AND Id_Miejsca=I_miejsca);
```

```
IF(Czy_sie_odbyl>0) THEN
```

```
RAISERROR 17001 'Operacja wstrzymana - koncert juz sie odbyl!';
```

```
ELSEIF(Czy_sie_odbyl=0) THEN
```

```
RAISERROR 17002 'Operacja wstrzymana - internetowa sprzedaz biletow zakonczona';
```

```
ELSEIF(I_miejsca<1 OR I_miejsca>103 OR I_miejsca=72) THEN
```

```
RAISERROR 17003 'Operacja wstrzymana - nie ma takiego miejsca';
```

```
ELSEIF(Czy_dobra_sala = 0) THEN
```

```
RAISERROR 17004 'Operacja wstrzymana - podane miejsce znajduje sie na zlej sali';
```

```
ELSEIF(Czy_zajete=1) THEN
```

```
RAISERROR 17005 'Operacja wstrzymana - wskazane miejsce jest zajete!';
```

```
ENDIF;
```

```
SET Koszt_biletu = (SELECT Cena_biletu
```

```
FROM Koncert K
```

```

WHERE K.Id_Koncert=I_koncertu);

SET Stary_budzet = (SELECT TOP 1 B.Stan_konta

FROM Budzet B

ORDER BY B.Id_Budzet DESC);

SET Nowy_budzet = Stary_budzet+Koszt_biletu;

SET R_tranzakcji = 'Kupno biletu przez ' + Pesel + ' na koncert nr ' +
CONVERT(VARCHAR,I_koncertu);

INSERT INTO Budzet(Stan_konta,DataB,KwotaB,Rodzaj_tranzakcji)

VALUES(Nowy_budzet,NOW(),Koszt_biletu,R_tranzakcji);

END

```

Stan bazy przed

projekt (DBA) on projekt - Interactive SQL

File Edit SQL Data Favorites Tools Window Help

SQL Statements

```
1 SELECT * From Bilet ORDER BY Id_Bilet DESC
```

Results

	Id_Bilet	Id_Koncert	Id_Miejsca	PeselSlu
1	676	13	66	88121212383
2	675	13	65	87120212420
3	674	13	64	66111601336

Stan bazy po:

SQL Statements

```
1 INSERT INTO Bilet(Id_Koncert,Id_Miejsca,PeselSlu)
2 VALUES (18,45, '67101712439')
```

Results

	Id_Bilet	Id_Koncert	Id_Miejsca	PeselSlu
1	720	18	45	67101712439
2	719	18	40	67101712418
3	676	13	66	88121212383

3

SELECT \* FROM Budzet ORDER BY Id\_Budzet DESC

Results

	Id_Budzet	Stan_konta	DataB	KwotaB	Rodzaj_tranzakcji
1	60	3,521.80	2017-02-20	55.00	Kupno biletu przez 67101712439 na koncert nr 18
2	59	3,466.80	2017-02-20	55.00	Kupno biletu przez 67101712418 na koncert nr 18

Wykonania powodujące błąd:

 ISQL Error
 ✕



Could not execute statement.  
 RAISERROR executed: Operacja wstrzymana - nie ma takiego miejsca  
 SQLCODE=-17003, ODBC 3 State="HY000"  
 Line 1, column 1

```
INSERT INTO Bilet(Id_Koncert,Id_Miej_sca,Pese
VALUES (18,120,'67101712439')
```

 ISQL Error
 ✕



Could not execute statement.  
 RAISERROR executed: Operacja wstrzymana - koncert juz sie odbyl  
 SQLCODE=-17001, ODBC 3 State="HY000"  
 Line 1, column 1

```
INSERT INTO Bilet(Id_Koncert,Id_Miej_sca,Pes
VALUES (1,45,'67101712439')
```

 ISQL Error
 ✕



Could not execute statement.  
 RAISERROR executed: Operacja wstrzymana - wskazane miejsce jest zajete!  
 SQLCODE=-17005, ODBC 3 State="HY000"  
 Line 1, column 1

```
INSERT INTO Bilet(Id_Koncert,Id_Miej_sca,Peselslu
VALUES (18,45,'67101712439')
```

 ISQL Error
 ✕



Could not execute statement.  
 RAISERROR executed: Operacja wstrzymana - podane miejsce znajduje sie na zlej sali  
 SQLCODE=-17004, ODBC 3 State="HY000"  
 Line 1, column 1

```
INSERT INTO Bilet(Id_Koncert,Id_Miej_sca,Peselslu)
VALUES (18,100,'67101712439')
```

3. Zmiana sali - Sprawdza czy można zmienić salę - jest to możliwe, jeżeli sala docelowa ma więcej miejsc niż sprzedano biletów na dany koncert. Wtedy wyzwalacz przesadza ludzi.

```
ALTER TRIGGER "Zmiana_sali" BEFORE UPDATE
```

```
ORDER 1 ON "DBA"."Koncert"
```

```
REFERENCING OLD AS stara NEW AS nowa
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    DECLARE Stare_Id INT;
```

```
    DECLARE Nowe_Id INT;
```

```
    DECLARE Koncert INT;
```

```
    DECLARE It_Miejsce INT;
```

```
    DECLARE Czy_sie_odbyl INT;
```

```
    DECLARE Ile_biletow INT;
```

```
    DECLARE Przypisz_miejsce INT;
```

```
    DECLARE Iterator CURSOR FOR
```

```
        (SELECT M.Id_Miejsca
```

```
          FROM Miejsce M
```

```
          INNER JOIN Bilet B
```

```
            ON B.Id_Miejsca=M.Id_Miejsca
```

```
          INNER JOIN Koncert K
```

```
            ON K.Id_Koncert=B.Id_Koncert
```

```
          WHERE K.Id_Koncert=nowa.Id_Koncert);
```

```
SET Koncert = nowa.Id_Koncert;
```

```
SET Stare_Id = stara.Id_Sala;
```

```
SET Nowe_Id = nowa.Id_Sala;
```

```
IF(Nowe_Id!=Stare_Id) THEN
```

```
    SET Czy_sie_odbyl = (SELECT DATEDIFF(DAY,K.Data_koncertu,NOW()))
```

```
        FROM Koncert K
```

```
        WHERE K.Id_Koncert=Koncert);
```

```
IF(Czy_sie_odbyl > 0) THEN RAISERROR 17010 'Operacja przerwana - koncert juz sie odbyl';
```

```
ENDIF;
```

```
IF(Nowe_Id=1) THEN
```

```
    OPEN iterator;
```

```
    petla: LOOP
```

```
        FETCH NEXT Iterator INTO It_Miejsce;
```

```
        IF (SQLCODE != 0) THEN LEAVE petla ENDIF;
```

```
        UPDATE Bilet
```

```
        SET Id_Miejsca=It_Miejsce-50
```

```
        WHERE Id_Miejsca=It_Miejsce
```

```
        AND Id_Koncert=Koncert;
```

```
    END LOOP;
```

```
    CLOSE iterator;
```

```
ELSEIF(Nowe_Id=2) THEN
```

```
    SET Ile_biletow = (SELECT COUNT(B.Id_Bilet)
```

```
        FROM Bilet B
```

```
        WHERE B.Id_Koncert=Koncert);
```

```
IF(Ile_biletow>32) THEN
```

```
    RAISERROR 17011 'Operacja przerwana - sprzedano wiecej biletow niz jest miejsc na docelowej sali';
```

```
ELSE
```

```
    SET Przypisz_miejsce=86-Ile_biletow/2+1;
```

```

OPEN iterator;

petla: LOOP

    FETCH NEXT Iterator INTO It_Miejsce;

    IF (SQLCODE != 0) THEN LEAVE petla ENDIF;

    IF(Przypisz_miejsce=72) THEN

        UPDATE Bilet

        SET Id_Miejsca=103

        WHERE Id_Miejsca=It_Miejsce

        AND Id_Koncert=Koncert;

    ELSE

        UPDATE Bilet

        SET Id_Miejsca=Przypisz_miejsce

        WHERE Id_Miejsca=It_Miejsce

        AND Id_Koncert=Koncert;

    ENDIF;

    SET Przypisz_miejsce=Przypisz_miejsce+1;

END LOOP;

CLOSE iterator;

ENDIF;

ELSE RAISERROR 17015 'Nie ma takiej sali.';

ENDIF;

ENDIF;

```

END2-krotne zmienienie Sali koncertu nr 18

Results				
	Id_Bilet	Id_Koncert	Id_Miejsca	PeselSlu
1	720	18	45	67101712439
2	719	18	40	67101712418
3	676	13	66	88121212383

Results				
	Id_Bilet	Id_Koncert	Id_Miejsca	PeselSlu
1	720	18	86	67101712439
2	719	18	85	67101712418
3	676	13	66	88121212383

Results				
	Id_Bilet	Id_Koncert	Id_Miejsca	PeselSlu
1	720	18	36	67101712439
2	719	18	35	67101712418
3	676	13	66	88121212383

Po dodaniu liczby biletów przekraczającej liczbę miejsc sali2 i próbie jej zmiany:

Results				
	Id_Bilet	Id_Koncert	Id_Miejsca	PeselSlu
1	719	18	30	67101712418
2	720	18	31	67101712439
3	726	18	32	67101712439
4	727	18	33	67101712439
5	728	18	34	67101712439
6	729	18	35	67101712439
7	730	18	36	69123012392
8	731	18	37	70091212383
9	732	18	38	71100912368
10	733	18	39	71100912431
11	734	18	40	72061412349
12	735	18	41	72061412453
13	736	18	42	72061412453
14	758	18	1	72061412349
15	759	18	2	72061412349
16	760	18	3	72061412349
17	761	18	4	72061412349
18	762	18	5	72061412349
19	763	18	6	72061412349
20	764	18	7	72061412349
21	765	18	8	72061412349
22	766	18	9	72061412349
23	767	18	10	72061412349
24	768	18	11	72061412349
25	769	18	12	72061412349
26	770	18	13	72061412349
27	771	18	14	72061412349
28	772	18	15	72061412349

SQL Statements	
1	SELECT COUNT(*) FROM Bilet WHERE Id_Koncert=18
2	
3	
<	
Results	
	COUNT()
1	44

ISQL Error



Could not execute statement.

RAISERROR executed: Operacja przerwana - sprzedano więcej biletów niż jest miejsc na docelowej sali

SQLCODE=-17011, ODBC 3 State="HY000"

Line 1, column 1

```
UPDATE Koncert  
SET Id_Sala=2  
WHERE Id_Koncert=18
```

Copy

OK

Help



## Użytkownicy

1. Klient – Klient może wyszukiwać koncerty i informacje z nimi związane, a także kupować na nie bilety.

Tabele:

Table Permissions									
	Name ▲	Owner	Grantors	S	I	D	U	A	R
1	Bilet	DBA	DBA		✓				
2	Koncert	DBA	DBA	✓,Ⓜ					
3	Miejsce	DBA	DBA	✓					
4	Obsada	DBA	DBA	✓					
5	Pelen_sklad_koncertowy	DBA	DBA	✓					
6	Repertuar	DBA	DBA	✓					
7	Sala	DBA	DBA	✓					
8	Utwor	DBA	DBA	✓					

2. Muzyk - użytkownik będący pracownikiem filharmonii, ma dostęp do wielu tabel (tych niezwiązanych z finansami oraz solistami), a także części funkcji, które umożliwiają mu przeliczenie swojej premii, a także obliczenie zainteresowania poszczególnymi utworami.

Tabele:

Table Permissions									
	Name ▲	Owner	Grantors	S	I	D	U	A	R
1	Koncert	DBA	DBA	✓					
2	Miejsce	DBA	DBA	✓					
3	Muzycy_w_obsadzie	DBA	DBA	✓					
4	Muzyk	DBA	DBA	✓			✓		
5	Muzyk_utwory	Muzycy_w_obsadzie	DBA	✓	✓	✓	✓		
6	Obsada	DBA	DBA	✓					
7	Partie_w_utworze	DBA	DBA	✓					
8	Pelen_sklad_koncertowy	DBA	DBA	✓					
9	Repertuar	DBA	DBA	✓					
10	Sala	DBA	DBA	✓					
11	Utwor	DBA	DBA	✓					
12	Wymagana_partia	DBA	DBA	✓					

Widoki:

	Name ▲	Owner	Grantors	S	I	D	U
1	Placa_praco...	DBA	DBA	✓			
2	Wykonawcy	DBA	DBA	✓			

Funkcje:

	Name ▲	Owner	Execute
1	Premia	DBA	✓
2	Procent_zaj...	DBA	✓

3. Prezes - jest osobą mającą kontrolę nad bazą danych filharmonii.

Tabele:

	Name ▲	Owner	Grantors	S	I	D	U	A	R
1	Bilet	DBA	DBA	✓	✓	✓	✓		
2	Budzet	DBA	DBA	✓	✓	✓			
3	Koncert	DBA	DBA	✓	✓	✓	✓		
4	Miejsce	DBA	DBA	✓	✓	✓	✓		
5	Muzycy_w_obsadzie	DBA	DBA	✓	✓	✓	✓		
6	Muzyk	DBA	DBA	✓	✓	✓	✓		
7	Muzyk_utwory	DBA	DBA	✓	✓	✓	✓		
8	Obsada	DBA	DBA	✓	✓	✓	✓		
9	Partie_w_utworze	DBA	DBA	✓	✓	✓	✓		
10	Pelen_sklad_koncertowy	DBA	DBA	✓	✓	✓	✓		
11	Repertuar	DBA	DBA	✓	✓	✓	✓		
12	Sala	DBA	DBA	✓	✓	✓	✓		
13	Sluchacz	DBA	DBA	✓	✓	✓	✓		
14	Solisci_w_skladzie	DBA	DBA	✓	✓	✓	✓		
15	Solista	DBA	DBA	✓	✓	✓	✓		
16	Solista_utwory	DBA	DBA	✓	✓	✓	✓		
17	Sponsor	DBA	DBA	✓	✓	✓	✓		
18	Utwor	DBA	DBA	✓	✓	✓	✓		
19	Wplata_na_koncert	DBA	DBA	✓	✓	✓	✓		
20	Wyznaczona_partia	DBA	DBA	✓	✓	✓	✓		

Widoki:

	Name ▲	Owner	Grantors	S	I	D	U
1	Aktualni_w...	DBA	DBA	✓	✓	✓	✓
2	Bilans_konc...	DBA	DBA	✓	✓	✓	✓
3	Placa_praco...	DBA	DBA	✓	✓	✓	✓
4	Wykonawcy	DBA	DBA	✓	✓	✓	✓

Procedury i funkcje:

	Name ▲	Owner	Execute
1	Czy_wykon...	DBA	✓
2	Dofinansow...	DBA	✓
3	Ilu_muzyko...	DBA	✓
4	Premia	DBA	✓
5	Procent_zaj...	DBA	✓
6	Utwor_do_...	DBA	✓
7	Wynagrodz...	DBA	✓