Bazy danych

Temat projektu:

Baza danych szpitala

Prowadzący Laboratorium: **płk dr inż. Jarosław Koszela**

Autor: ***Marek Cackowski***

Grupa: ***WCY201Y1S1***

Numer albumu: ***76771***

Data wykonania: ***27.01.2022***

Treść zadania

Wykonanie bazy danych szpitala posiadającej minimum:

- 10 tabel,
- 3 widoki,
- 3 procedury,
- 3 funkcje,
- 3 wyzwalacze,
- 3 użytkowników,

Zaimplementowanej w dwóch środowiskach (komercyjnym oraz open source).

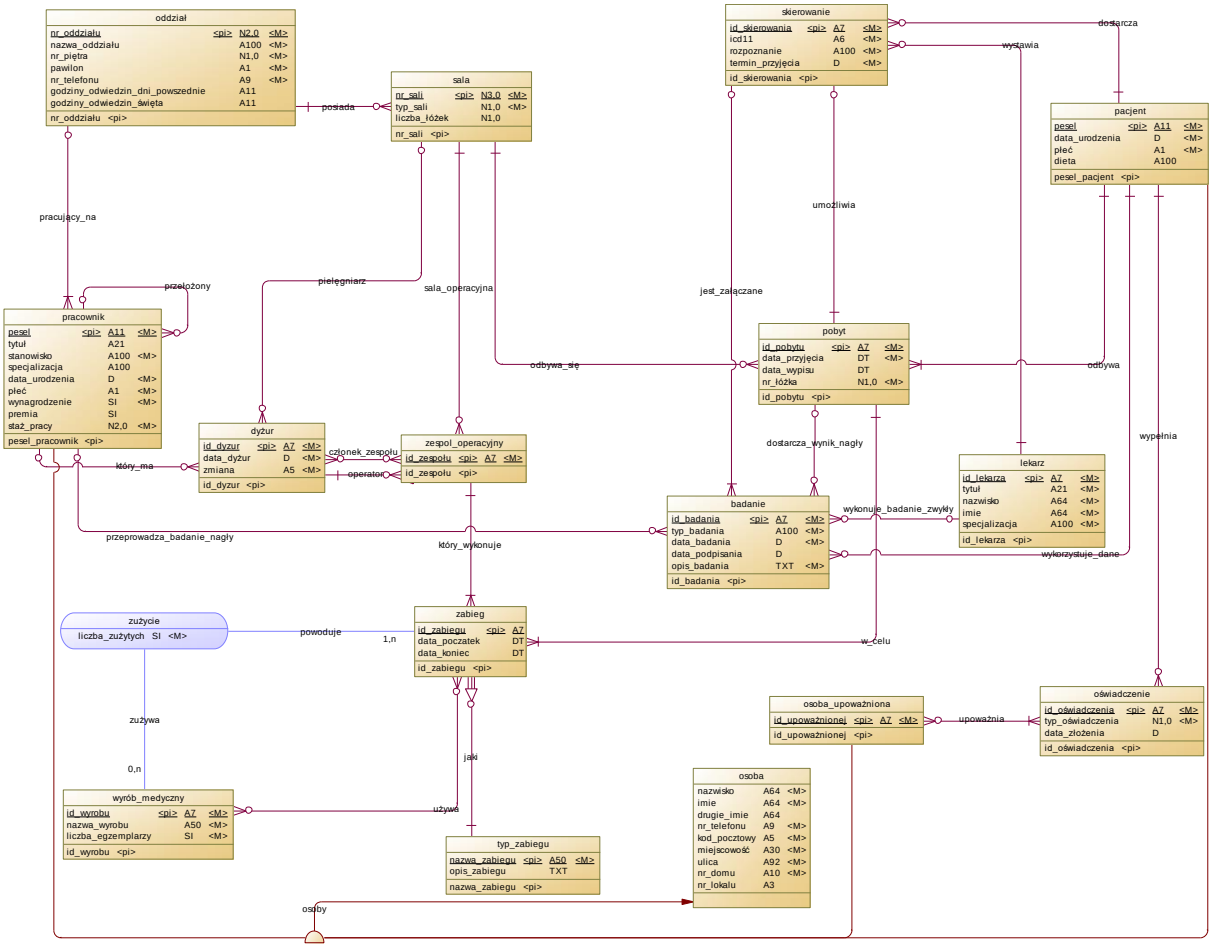
Opis

Szpital składa się z oddziałów posiadających sale (chorych i operacyjne). Na salach chorych przebywają pacjenci w trakcje swoich pobytów. Do sal tych są przypisane dyżurujące pielęgniarki. Na salach operacyjnych odbywają się zabiegi przeprowadzane przez operatora (opcjonalnie z zespołem operacyjnym składającym się z lekarzy i/lub pielęgniarek). Pacjent ma przypisane określone zabiegi do pobytu. Każdy zabieg jest określonego typu. Zabieg może używać (aparaturę) lub zużywać wyroby medyczne.

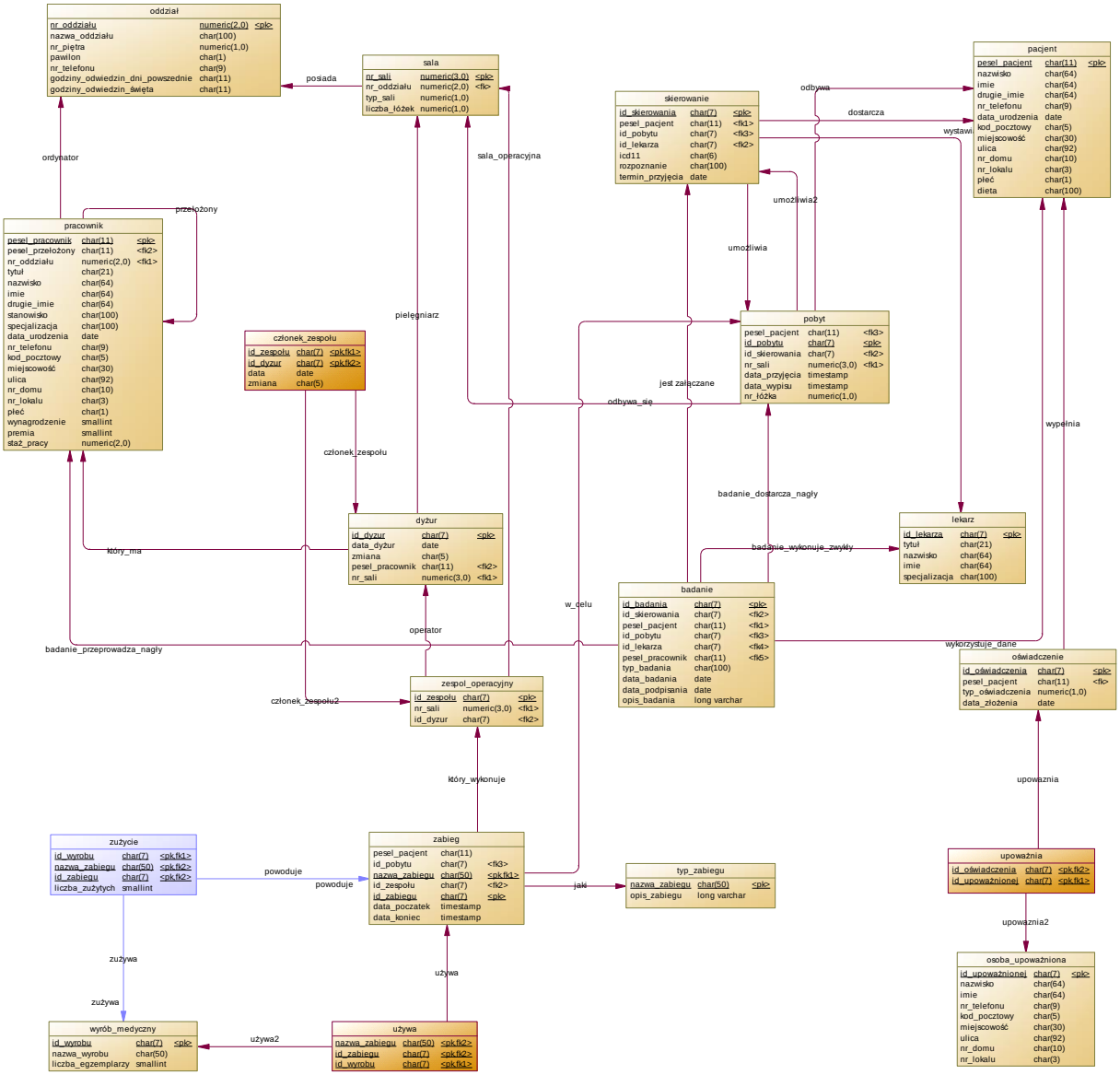
Pacjent jest przyjmowany do szpitala na podstawie skierowania (poza nagłymi przypadkami, w których skierowanie nie jest wystawiane, a badania są przeprowadzane przez lekarzy ze szpitala i przypisywane do konkretnego pobytu), które może zawierać badania (wystawiane przez określonego lekarza, który nie jest pracownikiem szpitala).

Pacjent ma możliwość złożenia dwóch oświadczeń, z których każde może upoważnić do 2 osób.

Model konceptualny



Model fizyczny



Wykaz tabel

- badanie - jest to badanie przeprowadzone osobiście przez lekarza w czasie postępowania z pacjentem.
- byly_pracownik - jest to tabela, do której zapisywani są wszyscy pracownicy usunięci z tabeli pracownik.
- czlonek_zespolu - jest to lekarz asystent lub pielęgniarka, która jest częścią zespołu operacyjnego.
- dyzur - jest to pełnienie obowiązków społecznych lub zawodowych w określonym przedziale czasu poza podstawowym czasem pracy.
- historia_pacjenta - jest to tabela, do której dodawane są wszystkie zmiany wprowadzone w tabeli pacjent.
- historia_upowaznionej - jest to tabela, do której dodawane są wszystkie zmiany wprowadzone w tabeli osoba_upowazniona.
- lekarz - jest to osoba posiadająca wiedzę i uprawnienia do leczenia ludzi. Zajmuje się przeprowadzaniem badań, konsultacji oraz wydawaniem skierowań. Nie jest on pracownikiem naszego szpitala.
- oddzial - jest to jednostka organizacyjna szpitala, świadcząca całodobową opiekę lekarsko-pielęgniarską nad pacjentami.
- osoba_upowazniona - jest to osoba, która otrzymała upoważnienie do uzyskiwania informacji o stanie zdrowia pacjenta i udzielonych świadczeń zdrowotnych lub do uzyskania dokumentacji medycznej.
- oswiadczenie – jest to formularz składany przez pacjenta w celu upoważnienia do uzyskiwania informacji o stanie zdrowia pacjenta i udzielonych świadczeń zdrowotnych lub uzyskania dokumentacji zdrowotnej.
- pacjent – jest to osoba korzystająca ze świadczeń opieki zdrowotnej, niezależnie od tego czy jest zdrowa, czy chora.
- pobyt – jest to okres między przyjęciem, a wypisaniem ze szpitala, podczas którego pacjent jest poddawany zabiegom.
- pracownik – jest to osoba fizyczna wykonująca określonego rodzaju pracę na rzecz pracodawcy, pod jego kierownictwem, w wyznaczonym przez niego miejscu i czasie, za co przysługuje mu wynagrodzenie.
- sala – jest to pomieszczenie znajdujące się w budynku szpitala, w którym mogą przebywać pacjenci w oczekiwaniu na zabieg.
- skierowanie – jest to z reguły pisemna dyspozycja lekarska kierująca pacjenta na leczenie szpitalne.
- typ_zabiegu – jest to określony zabieg medyczny, który służy diagnozowaniu, profilaktyce, a przede wszystkim leczeniu pacjenta.
- upowaznia – osoby podane na oświadczeniu zostają upoważnione do odbioru dokumentacji medycznej lub dostępu do informacji o stanie zdrowia (w zależności od wybranego typu oświadczenia).
- uzywa – użycie wyrobów medycznych, które w wyniku zabiegu nie ulegają zużyciu.
- wyrob_medyczny – jest to narzędzie, przyrząd, aparat, sprzęt, materiał lub inny artykuł, stosowany samodzielnie lub w połączeniu, włączając oprogramowanie niezbędne do właściwego stosowania wyrobu, przeznaczone przez wytwórcę do stosowania u ludzi w celu:

- diagnozowania, zapobiegania, monitorowania, leczenia lub łagodzenia przebiegu chorób,

- diagnozowania, monitorowania, leczenia, łagodzenia lub kompensowania urazów lub upośledzeń,

- badania, zastępowania lub modyfikowania budowy anatomicznej lub prowadzenia procesu fizjologicznego,

- regulacji poczęć.

- zabieg – jest to rodzaj czynności medycznej służącej diagnozowaniu, profilaktyce, a przede wszystkim leczeniu pacjenta. Czynność taka może być wykonywana zarówno ręcznie, jak i przy pomocy skomplikowanej aparatury medycznej i narzędzi.
- zespol_operacyjny – jest to zespół składający się z operatora, lekarzy asystentów i pielęgniarek. W zespole nie muszą znajdować się pielęgniarki, ani lekarze asystenci.
- zuzycie - jest to zużycie konkretnego wyrobu medycznego podczas jednego zabiegu.

Widoki

grupy_wiekowe_pobyty

Widok pokazuje ilość zakończonych pobytów w tym roku według podziału na grupy wiekowe (1- od 0 do 9, 2- od 10 do 19,...).

```
CREATE VIEW [dbo].[grupy_wiekowe_pobyty] AS
SELECT
	TOP 12 FLOOR (
		DATEDIFF (
			YEAR,
			p.data_urodzenia,
			GETDATE ()
		) / 10
	) + 1 AS grupa_wiekowa,
	COUNT (pobyt.id_pobytu) AS liczba_pobytow
FROM
	pacjent p
	INNER JOIN pobyt ON p.pesel_pacjent = pobyt.pesel_pacjent
WHERE
	YEAR (data_przyjecia)= YEAR (GETDATE())
GROUP BY
	FLOOR (
		DATEDIFF (
			YEAR,
			p.data_urodzenia,
			GETDATE ()
		)/ 10
	) + 1
ORDER BY
	grupa_wiekowa ASC
```

wizyty_ze_skierowaniem

Widok pokazuje listę pacjentów razem z liczbą wizyt, które odbyli na podstawie okazania skierowania. Przedstawia tylko pacjentów, którzy odbyli przynajmniej jedno badanie poza szpitalem.

```
CREATE VIEW [dbo].[wizyty_ze_skierowaniem] AS
SELECT
	TOP 100 p.nazwisko,
	p.imie,
	(
		SELECT
			COUNT(s1.pesel_pacjent)
		FROM
			skierowanie s1
		WHERE
			s1.pesel_pacjent = s.pesel_pacjent
			AND s1.id_lekarza IS NOT NULL
	) AS wizyty_ze_skierowaniem
FROM
	pacjent p
	INNER JOIN skierowanie s ON p.pesel_pacjent = s.pesel_pacjent
	INNER JOIN badanie b ON s.id_skierowania = b.id_skierowania
WHERE
	b.pesel_pracownik IS NULL
GROUP BY
	p.pesel_pacjent,
	p.nazwisko,
	p.imie,
	s.pesel_pacjent
ORDER BY
	3 DESC,
	COUNT(b.id_badania) DESC
```

zabiegi_wymagajace_najwiecej_zasobow

Widok pokazuje 3 zabiegi, podczas których średnia suma zużytych egzemplarzy oraz użytych wyrobów medycznych jest największa.

```
CREATE [dbo].[zabiegi_wymagajace_najwiecej_wyrobow] AS
SELECT
	TOP 3 tz.nazwa_zabiegu,
	SUM (x.suma) AS srednia_liczba_wyrobow
FROM
	(
		SELECT
			z1.nazwa_zabiegu AS nazwa,
			ROUND(
				CAST(
					AVG(zuzycie.liczba_zuzytych) AS float
				),
				2
			) AS suma
		FROM
			zabieg z1
			INNER JOIN zuzycie ON z1.id_zabiegu = zuzycie.id_zabiegu
		GROUP BY
			z1.nazwa_zabiegu
		UNION ALL
		SELECT
			z2.nazwa_zabiegu AS nazwa,
			ROUND(
				CAST(
					COUNT(uzywa.id_wyrobu) AS float
				)/ (                                SELECT
						COUNT(DISTINCT id_zabiegu)
					FROM
						uzywa
					WHERE
						nazwa_zabiegu = z2.nazwa_zabiegu
				),
				2
			) AS suma
		FROM
			zabieg z2
			INNER JOIN uzywa ON z2.id_zabiegu = uzywa.id_zabiegu
		GROUP BY
			z2.nazwa_zabiegu
	) X
	INNER JOIN typ_zabiegu tz ON tz.nazwa_zabiegu = x.nazwa
GROUP BY
	tz.nazwa_zabiegu
ORDER BY
	2 DESC
```

pacenci_informacje_dla_upowaznionych

Widok przedstawia informacje, które są udostępniane osobom upoważnionym do informacji medycznej na temat pacjenta.

```
CREATE VIEW [dbo].[pacjenci_informacje_dla_upowaznionych] AS
SELECT
	p.nazwisko,
	p.imie,
	ski.nr_sali,
	czy_posiada_skierowanie,
	COUNT (DISTINCT b.id_badania) AS liczba_wykonanych_badan,
	STUFF (
		(
			SELECT
				', ' + TRIM(tz.nazwa_zabiegu)
			FROM
				typ_zabiegu tz
				INNER JOIN zabieg z ON z.nazwa_zabiegu = tz.nazwa_zabiegu
			WHERE
				z.pesel_pacjent = p.pesel_pacjent FOR XML PATH('')
		),
		1,
		1,
		''
	) AS zabiegi,
	(
		SELECT
			TOP 1 TRIM (nazwisko) + ' ' + TRIM(imie)+ ', ' +
				LEAD(TRIM (nazwisko) + ' ' + TRIM(imie)) OVER(ORDER BY
p.pesel_pacjent)
		FROM
			osoba_upowazniona osu1
			INNER JOIN upowaznia up1 ON osu1.id_upowaznionej = up1.id_upowaznionej
			INNER JOIN oswiadczenie os1 ON up1.id_oswiadczenia = os1.id_oswiadczenia
		WHERE
			os1.pesel_pacjent = p.pesel_pacjent
			AND typ_oswiadczenia = 1
		ORDER BY
			p.pesel_pacjent
	) AS osoby_upowaznione
FROM
	(		SELECT
			ou.id_upowaznionej,
			ou.nazwisko,
			ou.imie,
			o.pesel_pacjent
		FROM
			osoba_upowazniona ou
			INNER JOIN upowaznia u ON ou.id_upowaznionej = u.id_upowaznionej
			INNER JOIN oswiadczenie o ON o.id_oswiadczenia = u.id_oswiadczenia
		WHERE
			typ_oswiadczenia = 1
	) AS osw
	INNER JOIN pacjent p ON p.pesel_pacjent = osw.pesel_pacjent
```

```
        INNER JOIN (
              SELECT
                    pobyt.pesel_pacjent,
                    pobyt.id_pobytu,
                    nr_lozka,
                    sala.nr_sali,
                    CASE
                          WHEN pobyt.id_skierowania IS NULL THEN 'NIE'
                          ELSE 'TAK'
                    END AS czy_posiada_skierowanie,
                    skierowanie.id_skierowania
              FROM
                    pobyt
                    INNER JOIN sala ON sala.nr_sali = pobyt.nr_sali
                    LEFT JOIN skierowanie ON skierowanie.id_skierowania = pobyt.id_skierowania
              WHERE
                    DATEDIFF(
                          DAY,
                          data_przyjecia,
                          GETDATE()
                    ) >= 0
                    AND data_wypisu IS NULL
       ) AS ski ON ski.pesel_pacjent = p.pesel_pacjent
       INNER JOIN badanie b ON b.id_pobytu = ski.id_pobytu
       OR b.id_skierowania = ski.id_skierowania
WHERE
       data_badania <= GETDATE()
GROUP BY
       p.pesel_pacjent,
       p.nazwisko,
       p.imie,
       ski.nr_sali,
       czy_posiada_skierowanie
```

doktorzy_najdluzszy_staz

Widok przedstawia lekarzy, których staż znajduje się w 3 najdłuższych na ich oddziale oraz posiadają stopień doktora.

```
CREATE VIEW [dbo].[doktorzy_najdluzszy_staz] AS
SELECT
	TOP 100 CONCAT (
		TRIM (tytul),
		' ',
		TRIM (nazwisko),
		' ',
		TRIM (imie)
	) AS doktor,
	nazwa_oddzialu
FROM
	(
		SELECT
			p1.pesel_pracownik,
			RANK() OVER(
				PARTITION BY o1.nazwa_oddzialu
				ORDER BY
					p1.staz DESC
			) AS pozycja
		FROM
			pracownik p1
			INNER JOIN oddzial o1 ON p1.nr_oddzialu = o1.nr_oddzialu
		WHERE
			stanowisko = 'lekarz'
	) X
	INNER JOIN pracownik p ON x.pesel_pracownik = p.pesel_pracownik
	INNER JOIN oddzial o ON p.nr_oddzialu = o.nr_oddzialu
WHERE
	pozycja < 4
	AND tytul LIKE '%dr%'
ORDER BY
	nazwa_oddzialu ASC
```

Procedury

premia_covid()

Procedura przyznaje roczną premię pielęgniarzom, którzy pracują na oddziale chorób zakaźnych. Premia jest zależna od średniej płacy pielęgniarzy.

```
CREATE PROCEDURE [dbo].[premia_covid]
AS
BEGIN
	DECLARE @pesel CHAR(11)
	DECLARE @liczba_dni INT
	DECLARE @srednie_wynagrodzenie INT
	DECLARE kursor CURSOR LOCAL FOR
		SELECT
			p.pesel_pracownik,
			COUNT(d.id_dyzur)
		FROM
			pracownik p
			INNER JOIN dyzur d ON p.pesel_pracownik = d.pesel_pracownik
			INNER JOIN sala s ON d.nr_sali = s.nr_sali
			INNER JOIN oddzial o ON s.nr_oddzialu = o.nr_oddzialu
		WHERE
			stanowisko = 'pielegniarz'
			AND nazwa_oddzialu = 'Chorób zakaźnych'
			AND DATEDIFF(
				YEAR,
				data_dyzur,
				GETDATE ()
			)= 0
		GROUP BY
			p.pesel_pracownik
	SET TRAN ISOLATION LEVEL SERIALIZABLE
	BEGIN TRY
		BEGIN TRAN
		OPEN kursor
		FETCH NEXT FROM kursor INTO @pesel, @liczba_dni
		WHILE @@FETCH_STATUS=0
		BEGIN
			IF @liczba_dni>=2
			BEGIN
				SET @srednie_wynagrodzenie=[dbo].[srednie_wynagrodzenie_pielegniarze]()
				UPDATE pracownik
				SET premia=premia+@liczba_dni*@srednie_wynagrodzenie/50
				WHERE pesel_pracownik=@pesel
			END
			FETCH NEXT FROM kursor INTO @pesel, @liczba_dni
		END
		COMMIT TRAN
	END TRY
	BEGIN CATCH
		IF @@TRANCOUNT>0
		BEGIN
			PRINT 'TRANSACTION ERROR'
			ROLLBACK TRAN
		END
		ELSE
		BEGIN
			PRINT 'ERROR'
		END
	END CATCH
	SET TRAN ISOLATION LEVEL READ COMMITTED
	CLOSE kursor
	DEALLOCATE kursor
END
```

przyznaj_premie()

Procedura przyznaje premię dla lekarza (lub lekarzy jeżeli oboje wykonali tyle samo zabiegów), który brał udział w największej liczbie zabiegów w tym miesiącu (jako operator lub członek zespołu).

```
CREATE PROCEDURE [dbo].[przyznaj_premie]
AS
BEGIN
	DECLARE @premia INT
	DECLARE @pesel CHAR(11)
	DECLARE kursor CURSOR LOCAL FOR
		SELECT
			TOP 1 WITH TIES p.pesel_pracownik
		FROM
			(		SELECT
					ze1.id_zespolu,
					d1.id_dyzur
				FROM
					dyzur d1
					INNER JOIN czlonek_zespolu c1 ON d1.id_dyzur = c1.id_dyzur
					INNER JOIN zespol_operacyjny ze1 ON c1.id_zespolu =
ze1.id_zespolu
				UNION ALL		SELECT
					ze2.id_zespolu,
					d2.id_dyzur
				FROM
					dyzur d2
					INNER JOIN zespol_operacyjny ze2 ON d2.id_dyzur = ze2.id_dyzur
			) AS X
			INNER JOIN zabieg za ON x.id_zespolu = za.id_zespolu
			INNER JOIN dyzur d ON x.id_dyzur = d.id_dyzur
			INNER JOIN pracownik p ON d.pesel_pracownik = p.pesel_pracownik
		WHERE
			stanowisko = 'lekarz'
			AND DATEDIFF (
				MONTH,
				data_poczatek,
				GETDATE ()
			)= 0
		GROUP BY
			p.pesel_pracownik
		HAVING
			COUNT(za.id_zabiegu) > 10
		ORDER BY
			COUNT (za.id_zabiegu) DESC
	BEGIN TRY
		BEGIN TRAN
		OPEN kursor
		FETCH NEXT FROM kursor INTO @pesel
		WHILE @@FETCH_STATUS=0
		BEGIN
			SET @premia=(SELECT premia
					FROM pracownik
					WHERE pesel_pracownik=@pesel)
			IF @premia IS NULL
			BEGIN
				UPDATE pracownik
				SET premia=1000
				WHERE pesel_pracownik=@pesel
			END
```

```
                    ELSE
                    BEGIN
                           UPDATE pracownik
                           SET premia=premia+1000
                           WHERE pesel_pracownik=@pesel
                    END
                    FETCH NEXT FROM kursor INTO @pesel
             END
             COMMIT TRAN
      END TRY
      BEGIN CATCH
             IF @@TRANCOUNT>0
             BEGIN
                    PRINT 'TRANSACTION ERROR'
                    ROLLBACK TRAN
             END
             ELSE
             BEGIN
                    PRINT 'ERROR'
             END
      END CATCH
      CLOSE kursor
      DEALLOCATE kursor
END
```

zmien_typ_sali()

Procedura zmienia typ sali na wybrany. Jeżeli sala została zmieniona z sali dla chorych na inną łóżka z niej są przekazywane do wybranej sali (@przeniesienie). W przypadku, gdy sala została zmieniona na salę dla chorych przypisuje jej liczbę łóżek (@lozka).

```
CREATE PROOCEDURE [dbo].[zmien_typ_sali]( @nr_sali numeric(3,0), @zmiana numeric(1,0), @lozka
numeric(1,0), @przeniesienie numeric(3,0) )
AS
BEGIN
	DECLARE @typ_sali INT
	DECLARE @liczba_lozek INT
	BEGIN TRY
		BEGIN TRAN
		SET @typ_sali=(SELECT typ_sali
					FROM sala
					WHERE nr_sali=@nr_sali)
		IF @typ_sali=@zmiana
		BEGIN
			PRINT 'PODANY TYP SALI TO JEST AKTUALNY TYP'
			ROLLBACK TRAN
		END
		ELSE IF @typ_sali>0 AND @zmiana=0
		BEGIN
			UPDATE sala
			SET typ_sali=@zmiana, liczba_lozek=@lozka
			WHERE nr_sali=@nr_sali
			COMMIT TRAN
		END
		ELSE IF @typ_sali>0 AND @zmiana>0
		BEGIN
			UPDATE sala
			SET typ_sali=@zmiana
			WHERE nr_sali=@nr_sali
			COMMIT TRAN
		END
		ELSE
		BEGIN
			SET @liczba_lozek=(SELECT liczba_lozek
								FROM sala
								WHERE nr_sali=@nr_sali)
			UPDATE sala
			SET liczba_lozek=liczba_lozek+@liczba_lozek
			WHERE nr_sali=@przeniesienie
			UPDATE sala
			SET typ_sali=@zmiana, liczba_lozek=NULL
			WHERE nr_sali=@nr_sali
			IF 0<>(SELECT typ_sali
				FROM sala
				WHERE nr_sali=@przeniesienie)
			BEGIN
				PRINT 'SALA DO KTOREJ PROBUJESZ PRZENIESC LOZKA NIE JEST SALA CHORYCH'
				ROLLBACK TRAN
			END
			ELSE IF 8<=(SELECT liczba_lozek
					FROM sala
					WHERE nr_sali=@przeniesienie)
			BEGIN
				PRINT 'SALA DO KTOREJ PROBUJESZ PRZENIESC LOZKA JEST ZA MALA'
				ROLLBACK TRAN
			END
			ELSE
				COMMIT TRAN
		END
	END TRY
```

```
	BEGIN CATCH
		IF @@TRANCOUNT>0
		BEGIN
			PRINT 'TRANSACTION ERROR'
			ROLLBACK TRAN
		END
		ELSE
		BEGIN
			PRINT 'ERROR'
		END
	END CATCH
END
```

zuzycie_wyrobu( )

Procedura zmniejsza liczbę wyrobów medycznych z tabeli wyrob_medyczny o podaną w tabeli zuzycie egzemplarzy.

```
CREATE PROCEDURE [dbo].[zuzycie_wyrobu]( @zId char(7) )
AS
BEGIN
    DECLARE @licznik int
    DECLARE @zuzyte int
    DECLARE @liczba int
    DECLARE @w_id char(7)
       DECLARE kursor CURSOR LOCAL FOR
        SELECT
                    id_wyrobu, liczba_zuzytych
        FROM
                    zuzycie
        WHERE
                    id_zabiegu=@zId
    OPEN kursor
       BEGIN TRY
             BEGIN TRAN
        SET @licznik=0
        FETCH NEXT FROM kursor INTO @w_id, @zuzyte
        WHILE @@FETCH_STATUS=0
            BEGIN
                SET @liczba=(SELECT liczba_egzemplarzy
                             FROM wyrob_medyczny
                             WHERE id_wyrobu=@w_id)
                IF @liczba>=@zuzyte
                    BEGIN
                        UPDATE wyrob_medyczny
                        SET liczba_egzemplarzy=@liczba-@zuzyte
                        WHERE id_wyrobu=@w_id
                    END
                ELSE
                    BEGIN
                        SET @licznik=1
                    END
                FETCH NEXT FROM kursor INTO @w_id, @zuzyte
            END
        IF @licznik=0
            BEGIN
                DELETE zuzycie
                WHERE id_zabiegu=@zId
                COMMIT TRAN
            END
        ELSE
            BEGIN
                ROLLBACK TRAN
            END
       END TRY
       BEGIN CATCH
              IF @@TRANCOUNT > 0
              BEGIN
                     ROLLBACK TRAN
                     PRINT 'ERROR IN TRANSACTION'
              END
              ELSE
              BEGIN
                     PRINT 'ERROR'
              END
       END CATCH
       CLOSE kursor
       DEALLOCATE kursor
END
```

zastepstwo( )

W sytuacji, gdy jeden z pracowników nie może przyjść do pracy dochodzi do zastępstwa (inny pracownik przychodzi za niego w danym dniu, a on zastępuje go w najbliższym możliwym terminie na tej samej zmianie).

```
CREATE PROCEDURE [dbo].[zastepstwo]( @dyzur date, @zmiana char(5), @pesel1 char(11), @pesel2
char(11) )
AS
BEGIN
       DECLARE @nowa_data date
    DECLARE @stanowisko1 char(50)
    DECLARE @stanowisko2 char(50)
    DECLARE @id_d1 char(7)
       DECLARE @id_d2 char(7)
    DECLARE @sala1 numeric(3,0)
    DECLARE @sala2 numeric(3,0)
       BEGIN TRY
              BEGIN TRAN
              SET @id_d2=(SELECT id_dyzur
                                  FROM dyzur
                                  WHERE pesel_pracownik=@pesel2 AND data_dyzur>GETDATE())
              SET @stanowisko1=(SELECT stanowisko
                                         FROM pracownik
                                         WHERE pesel_pracownik=@pesel1)
              SET @stanowisko2=(SELECT stanowisko
                           FROM pracownik
                           WHERE pesel_pracownik=@pesel2)
              SET @id_d1=(SELECT id_dyzur
                                  FROM dyzur
                                  WHERE DATEDIFF(DAY, data_dyzur, @dyzur)=0
                                         AND pesel_pracownik=@pesel1 AND zmiana=@zmiana)
              SET @sala1=(SELECT nr_sali
                                  FROM dyzur
                                  WHERE id_dyzur=@id_d1)
              SET @sala2=(SELECT nr_sali
                                  FROM dyzur
                                  WHERE id_dyzur=@id_d2)
              IF @id_d2 IS NULL OR @stanowisko1<>@stanowisko2
              BEGIN
                     PRINT 'NIE MOZNA ZAMIENIC DYZUROW DLA TYCH PRACOWNIKOW'
                     ROLLBACK TRAN
              END
              ELSE IF @id_d1 IS NULL
              BEGIN
                     PRINT 'PODANY PRACOWNIK NIE MA DYZURU W TYM TERMINIE'
                     ROLLBACK TRAN
              END
              SET @nowa_data=(SELECT data_dyzur
                           FROM dyzur
                           WHERE id_dyzur=@id_d2)

              UPDATE dyzur
        SET data_dyzur=@nowa_data, nr_sali=@sala2
        WHERE id_dyzur=@id_d1
        UPDATE dyzur
        SET data_dyzur=@dyzur, nr_sali=@sala1
        WHERE id_dyzur=@id_d2

        COMMIT TRAN
       END TRY
```

```
	BEGIN CATCH
		IF @@TRANCOUNT>0
		BEGIN
			PRINT 'TRANSACTION ERROR'
			ROLLBACK TRAN
		END
		ELSE
		BEGIN
			PRINT 'ERROR'
		END
	END CATCH
END
```

Funkcje

srednie_wynagrodznie_lekarze() i srednie_wynagrodzenie_pielegniarze()

Funkcja zwraca średnie wynagrodzenie lekarzy/pielęgniarzy z całego szpitala.

```
CREATE FUNCTION [dbo].[srednie_wynagrodzenie_lekarze]()
RETURNS FLOAT
AS
BEGIN
	DECLARE @srednia INT
	DECLARE @suma FLOAT
	DECLARE @licznik FLOAT
	SET @suma=(SELECT SUM(wynagrodzenie)
			FROM pracownik
			WHERE stanowisko='lekarz')
	SET @licznik=(SELECT COUNT(*)
				FROM pracownik
				WHERE stanowisko='lekarz')
	IF @licznik<>0
	BEGIN
		SET @srednia=@suma/@licznik
	END
	ELSE
	BEGIN
		SET @srednia=0
	END
	RETURN CAST(@srednia AS INT)
END
```

podwladni()

Funkcja zwraca tablicę wszystkich pracowników, których przełożonym jest osoba o podanym peselu.

```
CREATE FUNCTION [dbo].[podwladni] (
	@pesel CHAR (11)
)
RETURNS TABLE
AS
RETURN
	WITH X AS(
	SELECT
		pesel_pracownik,
		nazwisko,
		imie,
		pesel_przelozony
	FROM
		pracownik
	WHERE
		pesel_przelozony = @pesel
	UNION ALL	SELECT
		p.pesel_pracownik,
		p.nazwisko,
		p.imie,
		p.pesel_przelozony
	FROM
		pracownik p
		INNER JOIN X ON x.pesel_pracownik = p.pesel_przelozony
	)
	SELECT
		pesel_pracownik,
		nazwisko,
		imie
	FROM
		x
```

Wyzwalacze

trg_byly_pracownik

Wyzwalacz dodaje każdego pracownika usuniętego z tabeli do tabeli byli_pracownicy.

```
CREATE TRIGGER [dbo].[trg_byly_pracownik]
ON [dbo].[pracownik]
FOR DELETE
AS
	INSERT INTO byly_pracownik (pesel_pracownik, nr_oddzialu, tytul, nazwisko, imie,
drugie_imie, stanowisko, specjalizacja, nr_telefonu, data_urodzenia, kod_pocztowy, miejscowosc,
ulica, nr_domu, nr_lokalu, plec, wynagrodzenie, staz)
	SELECT pesel_pracownik, nr_oddzialu, tytul, nazwisko, imie, drugie_imie,
stanowisko, specjalizacja, nr_telefonu, data_urodzenia, kod_pocztowy,
miejscowosc, ulica, nr_domu, nr_lokalu, plec, wynagrodzenie, staz
	FROM deleted
```

trg_dyzur

Wyzwalacz nie pozwala na wprowadzenie nr_sali jeżeli pełnioną przez pracownika funkcją jest lekarza oraz wymaga wprowadzenia nr_sali od pracowników z innych stanowisk.

```
CREATE TRIGGER [dbo].[trg_dyzur]
ON [dbo].[dyzur]
FOR INSERT, UPDATE
AS
DECLARE @nr_sali INT
DECLARE @pesel CHAR(11)
DECLARE @stanowisko CHAR(50)
DECLARE kursor CURSOR LOCAL FOR
	SELECT pesel_pracownik, nr_sali
	FROM inserted
OPEN kursor
FETCH NEXT FROM kursor INTO @pesel, @nr_sali
WHILE @@FETCH_STATUS=0
BEGIN
	SET @stanowisko=(SELECT stanowisko
					FROM pracownik
					WHERE pesel_pracownik=@pesel)
	IF @nr_sali IS NOT NULL AND @stanowisko='lekarz'
		THROW 51000, 'Lekarz nie ma okreslonej sali podczas dyzuru', 1
	ELSE IF @nr_sali IS NULL AND @stanowisko<>'lekarz'
		THROW 51000, 'Pielegniarz odbywa dyzur na okreslonej sali', 1
	FETCH NEXT FROM kursor INTO @pesel, @nr_sali
END
```

trg_pracownik

Wyzwalacz nie pozwala na wprowadzenie lekarza bez tytułu i specjalizacji oraz nie pozwala na wprowadzenie pracownika z innego stanowiska, który je posiada.

```
CREATE TRIGGER [dbo].[trg_pracownik]
ON [dbo].[pracownik]
FOR INSERT, UPDATE
AS
DECLARE @stanowisko CHAR(50)
DECLARE @tytul BIT
DECLARE @specjalizacja BIT
DECLARE @oddzial BIT
DECLARE kursor CURSOR LOCAL FOR
	SELECT stanowisko,
		CASE WHEN tytul IS NULL THEN 0 ELSE 1 END,
		CASE WHEN specjalizacja IS NULL THEN 0 ELSE 1 END,
		CASE WHEN nr_oddzialu IS NULL THEN 0 ELSE 1 END
	FROM inserted
OPEN kursor
FETCH NEXT FROM kursor INTO @stanowisko, @tytul, @specjalizacja, @oddzial
WHILE @@FETCH_STATUS=0
BEGIN
	IF @stanowisko<>'lekarz' AND (@tytul=1 OR @oddzial=1)
		THROW 51000, 'Pielegniarz nie moze miec tytulu, ani byc ordynatorem', 1
	ELSE IF @stanowisko='lekarz' AND (@tytul=0 OR @specjalizacja=0)
		THROW 51000, 'Lekarz musi miec tytul oraz specjalizacji', 1
	FETCH NEXT FROM kursor INTO @stanowisko, @tytul, @specjalizacja, @oddzial
END
```

trg_pobyt

Wyzwalacz nie pozwala na wprowadzenie pobytu na łóżku, które ma wyższy numer niż liczba łóżek na danej sali, na wprowadzenie daty_wypisu późniejszej od obecnej daty, ani daty_wypisu mniejszej od daty_przyjecia.

```
CREATE TRIGGER [dbo].[trg_pobyt]
ON [dbo].[pobyt]
FOR INSERT, UPDATE
AS
DECLARE @liczba_lozek INT
DECLARE @nr_lozka INT
DECLARE kursor CURSOR LOCAL FOR
	SELECT liczba_lozek, nr_lozka
	FROM sala INNER JOIN inserted ON sala.nr_sali=inserted.nr_sali
OPEN kursor
FETCH NEXT FROM kursor INTO @liczba_lozek, @nr_lozka
WHILE @@FETCH_STATUS=0
BEGIN
	IF @liczba_lozek<@nr_lozka
		THROW 51000, 'Nie ma lozka o takim numerze w tej sali', 1
	FETCH NEXT FROM kursor INTO @liczba_lozek, @nr_lozka
END
```

trg_sala

Wyzwalacz nie pozwala na wprowadzenie sali chorych bez łóżek, ani innego typu sali z łóżkami.

```
CREATE TRIGGER [dbo].[trg_sala]
ON [dbo].[sala]
FOR INSERT, UPDATE
AS
DECLARE @typ_sali INT
DECLARE @liczba_lozek BIT
DECLARE kursor CURSOR LOCAL FOR
	SELECT typ_sali,
		CASE WHEN liczba_lozek IS NULL THEN 0 ELSE 1 END
	FROM inserted
OPEN kursor
FETCH NEXT FROM kursor INTO @typ_sali, @liczba_lozek
WHILE @@FETCH_STATUS=0
BEGIN
	IF @typ_sali=0 AND @liczba_lozek=0
		THROW 51000, 'Na sali chorych musza byc lozka', 1
	ELSE IF @typ_sali>0 AND @liczba_lozek=1
		THROW 51000, 'Lozka moga byc tylko na sali chorych', 1
	FETCH NEXT FROM kursor INTO @typ_sali, @liczba_lozek
END
```

trg_zmiana_oswiadczenia

Wyzwalacz powoduje, że jeżeli pacjent posiada już oświadczenie danego typu, a chce wprowadzić kolejne to poprzednie zostaje usunięte.

```
CREATE TRIGGER [dbo].[trg_zmiana_oswiadczenia]
ON [dbo].[oswiadczenie]
FOR INSERT, UPDATE
AS
DECLARE @typ_oswiadczenia TINYINT
DECLARE @pesel CHAR(11)
DECLARE kursor CURSOR LOCAL FOR
	SELECT typ_oswiadczenia, pesel_pacjent
	FROM inserted
BEGIN TRY
	OPEN kursor
	FETCH NEXT FROM kursor INTO @typ_oswiadczenia, @pesel
	WHILE @@FETCH_STATUS=0
	BEGIN
		IF @typ_oswiadczenia=1
		BEGIN
			BEGIN TRAN
			DELETE
			FROM oswiadczenie
			WHERE typ_oswiadczenia=1 AND pesel_pacjent=@pesel
			COMMIT TRAN
		END
		ELSE
		BEGIN
			BEGIN TRAN
			DELETE
			FROM oswiadczenie
			WHERE typ_oswiadczenia=2 AND pesel_pacjent=@pesel
			COMMIT TRAN
		END
		FETCH NEXT FROM kursor INTO @typ_oswiadczenia, @pesel
	END
END TRY
BEGIN CATCH
	IF @@TRANCOUNT>0
	BEGIN
		PRINT 'TRANSACTION ERROR IN TRG_ZMIANA_OSWIADCZENIA'
		ROLLBACK TRAN
	END
	ELSE
	BEGIN
		PRINT 'ERROR'
	 END
END CATCH
```