Programowanie serwera baz danych – projekt zaliczeniowy

Artur Górecki – 389143

## Przetwarzania realizowane przez system

System korzysta z bazy danych *SerwisObrazow*, dołączonej w ramach projektu zaliczeniowego z przedmiotu **Projektowanie baz danych**, w wersji z dnia 13.06.2022. Wspomniana baza danych miałaby pełnić podstawy portalu, który byłby połączeniem internetowej bazy danych oraz portalu społecznościowego, dla fanów malarstwa.

Przechowuje dane obrazów (oraz dotyczące ich szczegóły) wraz z ich autorami oraz aktualnym miejscem przechowywania, dane użytkowników oraz efekty ich działań – jak nadawanie obrazom ocen, recenzji, oznaczania ich jako zobaczone; tworzenia rekomendacji innym użytkownikom czy tworzenia list.

System też przetwarza wspominane wyżej dane, pozwalając na raportowanie czy filtrowanie ich – jak choćby listę najlepiej ocenianych obrazów czy wykaz epok, które posiadają największą ilość obrazów wraz z ich najbardziej twórczymi artystami.

## Utworzone skrypty SQL

W ramach wyżej wymienionych przetwarzań wykonałem łącznie 8 skryptów SQL w programie MS SQL Server. Dostępne są one też w postaci plików .sql, zawartych w folderze wraz z dokumentacją.

### Modyfikacja danych

Pierwszym skryptem modyfikującym dane jest trigger *dbo.trg\_Ocena*, który będzie uruchamiał się zamiast polecenia INSERT na tabeli *dbo.Ocena*. Pozwala to na określenie warunków akceptacji wartości oceny danego obrazu, walidując to, czy mieści się on w przedziale <1,10>, ze skokiem 1, zgodnie z zamierzeniami zawartymi w projekcie z **Projektowania baz danych**.

Poniższy kod sprawdza, czy podana wartość mieści się w tym przedziale (w przeciwnym wypadku informując użytkownika o tym, że nie jest to poprawna wartość), a typ danych integer gwarantuje całkowitość podanej liczby, spełniając wszystkie zamierzone oczekiwania.

|  |
| --- |
| -- trigger do zapewnienia tego, że wartość oceny będzie się mieściła w założonym zakresie <1;10> ze skokiem 1.  CREATE OR ALTER TRIGGER dbo.trg\_Ocena ON dbo.Ocena  INSTEAD OF INSERT  AS  BEGIN  SET NOCOUNT ON;  DECLARE  @wartosc int  SET @wartosc = (SELECT wartosc FROM inserted);  IF @wartosc < 1 OR @wartosc > 10  THROW 50001, 'Wartosc oceny musi sie miescic w przedziale 1-10!', 1;  ELSE  INSERT INTO dbo.Ocena(id\_uzytkownik, id\_obraz, wartosc, data\_wystawienia) SELECT id\_uzytkownik, id\_obraz, wartosc, data\_wystawienia FROM inserted;  END  GO  -- sprawdzenie działania Triggera  -- spodziewany błąd - za duża wartość  INSERT INTO dbo.Ocena(id\_uzytkownik, id\_obraz, wartosc, data\_wystawienia)  VALUES  (2, 2, 11, '2022-06-17')  -- spodziewany błąd - za mała wartość  INSERT INTO dbo.Ocena(id\_uzytkownik, id\_obraz, wartosc, data\_wystawienia)  VALUES  (2, 2, 0, '2022-06-17')  -- poprawne dodanie  INSERT INTO dbo.Ocena(id\_uzytkownik, id\_obraz, wartosc, data\_wystawienia)  VALUES  (2, 2, 10, '2022-06-17') |

Drugim skryptem modyfikującym dane jest trigger *dbo.rg\_Rekomendacja*, który będzie uruchamiał się zamiast polecenia INSERT na tabeli *dbo.Rekomendacja*. W ten sposób można dodatkowo sprawdzić poprawność podanych danych oraz umożliwić to, że przedmiot rekomendacji może być tylko jeden – albo obraz, albo artysta.

Skrypt wpierw sprawdza, czy zarówno odbiorca jak i nadawca rekomendacji istnieją – rekomendacja nie może zostać wysłana w eter, jak i z niego pochodzić; jest wpisem od jednego do drugiego użytkownika. Toteż następnie sprawdzane jest to, czy przypadkiem ktoś nie wysyła rekomendacji do samego siebie, taka możliwość jest zablokowana. Następnym krokiem jest upewnienie się, że rekomendacja rekomenduje dokładnie jeden obiekt – artystę lub obraz, sprawdzając, czy przypadkiem nie zostały podane naraz oba, lub wcale. Po powyższej walidacji, ostatecznie dodawany jest rekord.

|  |
| --- |
| CREATE OR ALTER TRIGGER dbo.trg\_Rekomendacja ON dbo.Rekomendacja  INSTEAD OF INSERT  AS  BEGIN  SET NOCOUNT ON;  DECLARE  @nadawca int,  @odbiorca int,  @obraz int,  @artysta int;  SET @nadawca = (SELECT id\_uzytkownik\_nadawca FROM inserted);  SET @odbiorca = (SELECT id\_uzytkownik\_odbiorca FROM inserted);  SET @obraz = (SELECT id\_obraz FROM inserted);  SET @artysta = (SELECT id\_artysta FROM inserted);  IF (SELECT COUNT(1) FROM dbo.Uzytkownik WHERE id\_uzytkownik = @nadawca) = 0  THROW 50000, 'Nadawca musi istniec!', 1;  IF (SELECT COUNT(1) FROM dbo.Uzytkownik WHERE id\_uzytkownik = @odbiorca) = 0  THROW 50000, 'Odbiorca musi istniec!', 1;  IF @nadawca = @odbiorca  THROW 50001, 'Nie mozna wysylac rekomendacji samemu sobie!', 1;  IF @obraz IS NULL AND @artysta IS NULL  THROW 50002, 'Cos musi byc przedmiotem rekomendacji!', 1;  IF @obraz IS NOT NULL AND @artysta IS NOT NULL  THROW 50003, 'Nie mozna jednoczesnie polecic artysty i obrazu!', 1;  ELSE  INSERT INTO dbo.Rekomendacja(id\_uzytkownik\_nadawca, id\_uzytkownik\_odbiorca, id\_obraz, id\_artysta)  SELECT id\_uzytkownik\_nadawca, id\_uzytkownik\_odbiorca, id\_obraz, id\_artysta FROM inserted;  END  GO  -- błędne wykonanie - użytkownik poleca nieistniejącemu użytkownikowi  INSERT INTO dbo.Rekomendacja(id\_uzytkownik\_nadawca, id\_uzytkownik\_odbiorca, id\_obraz, id\_artysta)  VALUES (1, 3, 2, NULL);  -- błędne wykonanie - użytkownik poleca samemu sobie  INSERT INTO dbo.Rekomendacja(id\_uzytkownik\_nadawca, id\_uzytkownik\_odbiorca, id\_obraz, id\_artysta)  VALUES (1, 1, 2, NULL);  -- błędne wykonanie - rekomendacja niczego nie poleca  INSERT INTO dbo.Rekomendacja(id\_uzytkownik\_nadawca, id\_uzytkownik\_odbiorca, id\_obraz, id\_artysta)  VALUES (1, 2, NULL, NULL);  -- błędne wykonanie - rekomendacja zawiera oba pola  INSERT INTO dbo.Rekomendacja(id\_uzytkownik\_nadawca, id\_uzytkownik\_odbiorca, id\_obraz, id\_artysta)  VALUES (1, 2, 2, 3);  -- prawidłowe wywołanie  INSERT INTO dbo.Rekomendacja(id\_uzytkownik\_nadawca, id\_uzytkownik\_odbiorca, id\_obraz, id\_artysta)  VALUES (2, 1, NULL, 3) |

### Przeglądanie danych

Pierwszym skryptem do przeglądania danych jest skrypt umożliwiający filtrowanie obrazów ich rokiem stworzenia, jednocześnie dając możliwość wyboru sortowania (w danym roku, od danego roku, do danego roku, czy w okresie pomiędzy dwoma).

Skrypt na początku zawiera dwie instrukcje warunkowe, których celem jest zwrócenie użytkownikowi konkretnego komunikatu w przypadku niezałączenia koniecznych parametrów – jednego roku do filtrowania oraz wybrania trybu filtrowania.

W przypadku wybrania opcji „od”, „do” i „równo” system upewnia się, czy nie ma podanej drugiej daty, a następnie wyświetla wszystkie obrazy spełniające warunek.

W przypadku wybrania opcji „pomiędzy” system upewnia się, że obie daty zostały uwzględnione, a jednocześnie są podane w kolejności chronologicznej oraz jedna nie jest równa drugiej; następnie wyświetlając pasujące obrazy.

|  |
| --- |
| -- procedura do przeglądania wszystkich obrazów z ich najwazniejszymi szczegółami, stworzonymi między wybranymi latami  CREATE OR ALTER PROCEDURE up\_sortuj\_obrazy\_rokiem (@rok\_1 int, @rok\_2 int = NULL, @decyzja nvarchar(8))  AS  BEGIN  SET NOCOUNT ON;  IF @rok\_1 IS NULL  THROW 50001, 'Pierwszy rok nie moze byc pusty!', 1;  IF @decyzja IS NULL  THROW 50002, 'Trzeba podac tryb sortowania!', 1;  IF @decyzja = 'rowno'  BEGIN  IF @rok\_2 IS NOT NULL  THROW 50011, 'W przypadku zakresu "rowno" nie mozna podac pary lat', 1;  ELSE  select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  WHERE YEAR(data\_powstania) = @rok\_1  END  IF @decyzja = 'od'  BEGIN  IF @rok\_2 IS NOT NULL  THROW 50011, 'W przypadku zakresu "od" nie mozna podac pary lat', 1;  ELSE  select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  WHERE YEAR(data\_powstania) >= @rok\_1  END  IF @decyzja = 'do'  BEGIN  IF @rok\_2 IS NOT NULL  THROW 50012, 'W przypadku zakresu "do" nie mozna podac pary lat', 1;  ELSE  select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  WHERE YEAR(data\_powstania) <= @rok\_1  END  IF @decyzja = 'pomiedzy'  BEGIN  IF @rok\_2 IS NULL  THROW 50013, 'W przypadku zakresu "pomiedzy" trzeba podac pare lat!', 1;  ELSE  IF @rok\_2 < @rok\_1  THROW 50031, 'Druga data nie moze byc mniejsza od pierwszej!', 1;  IF @rok\_1 = @rok\_2  THROW 50032, 'Daty nie moga byc rowne sobie', 1;  ELSE  select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  WHERE YEAR(data\_powstania) BETWEEN @rok\_1 AND @rok\_2  END  IF @decyzja != 'od' AND @decyzja != 'do' AND @decyzja != 'pomiedzy' AND @decyzja != 'rowno'  THROW 50021, 'Podaj parametr w postaci "od"/"do"/"pomiedzy"!', 1;  RETURN  END  -- wywołanie nieprawidłowe - brak drugiej daty  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1505, @decyzja = 'pomiedzy'  -- wywołanie nieprawidłowe - zła decyzja  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1800, @decyzja = 'dx'  -- wywołanie nieprawidłowe - brak pierwszego, koniecznego roku  EXEC up\_sortuj\_obrazy\_rokiem @rok\_2 = 1800, @decyzja = 'do'  -- wywołanie nieprawidłowe - za dużo lat  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1800, @rok\_2 = 1999, @decyzja = 'do'  -- wywołanie nieprawidłowe - daty nie sa sobie równe  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1807, @rok\_2 = 1807, @decyzja = 'pomiedzy'  -- wywołanie nieprawidłowe - daty są w złej kolejności  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1807, @rok\_2 = 1800, @decyzja = 'pomiedzy'  -- wywołanie prawidłowe  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1500, @rok\_2 = 1950, @decyzja = 'pomiedzy'  -- wywołanie prawidłowe  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1600, @decyzja = 'do'  -- wywołanie prawidłowe  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1500, @decyzja = 'od'  -- wywołanie prawidłowe  EXEC up\_sortuj\_obrazy\_rokiem @rok\_1 = 1507, @decyzja = 'rowno' |

Drugą procedurą do przeglądania danych jest procedura *up\_filtruj\_nazwy\_obrazow*, pozwalająca na filtrowanie artystów i ich dzieł, wyświetlając poza tytułem dzieła wszystkie jego dodatkowe informacje.

Procedura wpierw sprawdza, czy podano jakiekolwiek parametry (które domyślnie mają możliwość bycia pustymi, co pozwala na sytuację, w której użytkownik może wybrać którąkolwiek z opcji), informując użytkownika o ewentualnym braku.

W przypadku podania tylko tytułu pracy, procedura sprawdza, czy faktycznie istnieją prace zawierające w nazwie podany ciąg znaków – w przypadku braku wyników użytkownik jest adekwatnie informowany. Jeżeli natomiast są wyniki, zostaną one wyświetlone.  
W przypadku podania tylko imienia lub nazwiska artysty, procedura sprawdza, czy faktycznie istnieją prace zawierające podany ciąg znaków – w przypadku braku wyników użytkownik jest adekwatnie informowany. Jeżeli natomiast są wyniki, zostaną one wyświetlone.

Adekwatnie wygląda filtrowanie w przypadku podania obu parametrów, gdzie system wpierw sprawdza, czy istnieją rekordy spełniające oba wymagania. W przypadku braku – użytkownik się o tym braku dowie, a jeżeli istnieją – zostaną wyświetlone.

|  |
| --- |
| -- procedura do przeglądania wszystkich obrazów z ich najwazniejszymi szczegółami, filtrując po tytule/artyscie  CREATE OR ALTER PROCEDURE up\_filtruj\_nazwy\_obrazow (@tytul nvarchar(250) = NULL, @artysta nvarchar(355) = NULL)  AS  BEGIN  SET NOCOUNT ON;  IF @tytul IS NULL AND @artysta is NULL  THROW 50001, 'Nalezy podac warunki filtrowania!', 1;  IF @artysta IS NULL  BEGIN  --BEGIN TRY  IF (SELECT COUNT(1) FROM  (select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, ROUND(avg(CAST(ocena.wartosc AS FLOAT)),2) as srednia\_ocena,  YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa as epoka, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  inner join dbo.ocena on ocena.id\_obraz = Obraz.id\_obraz  WHERE obraz.nazwa LIKE ('%'+@tytul+'%')  group by obraz.nazwa, osoba.imie, osoba.nazwisko, ocena.id\_obraz, obraz.data\_powstania, epoka.nazwa, obraz.rozmiar) as filtrowanie ) = 0  THROW 50201, 'Brak wyników!', 1;  ELSE  select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, ROUND(avg(CAST(ocena.wartosc AS FLOAT)),2) as srednia\_ocena,  YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa as epoka, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  inner join dbo.ocena on ocena.id\_obraz = Obraz.id\_obraz  WHERE obraz.nazwa LIKE ('%'+@tytul+'%')  group by obraz.nazwa, osoba.imie, osoba.nazwisko, ocena.id\_obraz, obraz.data\_powstania, epoka.nazwa, obraz.rozmiar    /\*END TRY  BEGIN CATCH  THROW 50101, 'Błąd filtrowania!', 1;  END CATCH\*/  END    IF @tytul IS NULL  BEGIN  --BEGIN TRY  IF (SELECT COUNT(1) FROM  (select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, ROUND(avg(CAST(ocena.wartosc AS FLOAT)),2) as srednia\_ocena,  YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa as epoka, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  inner join dbo.ocena on ocena.id\_obraz = Obraz.id\_obraz  WHERE (osoba.imie + ' ' + osoba.nazwisko) LIKE ('%'+@artysta+'%')  group by obraz.nazwa, osoba.imie, osoba.nazwisko, ocena.id\_obraz, obraz.data\_powstania, epoka.nazwa, obraz.rozmiar) as filtrowanie ) = 0  THROW 50201, 'Brak wyników!', 1;  ELSE  select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, ROUND(avg(CAST(ocena.wartosc AS FLOAT)),2) as srednia\_ocena,  YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa as epoka, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  inner join dbo.ocena on ocena.id\_obraz = Obraz.id\_obraz  WHERE (osoba.imie + ' ' + osoba.nazwisko) LIKE ('%'+@artysta+'%')  group by obraz.nazwa, osoba.imie, osoba.nazwisko, ocena.id\_obraz, obraz.data\_powstania, epoka.nazwa, obraz.rozmiar    /\*END TRY  BEGIN CATCH  THROW 50101, 'Błąd filtrowania!', 1;  END CATCH\*/  END  IF @artysta IS NOT NULL AND @tytul IS NOT NULL  BEGIN  --BEGIN TRY  IF (SELECT COUNT(1) FROM  (select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, ROUND(avg(CAST(ocena.wartosc AS FLOAT)),2) as srednia\_ocena,  YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa as epoka, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  inner join dbo.ocena on ocena.id\_obraz = Obraz.id\_obraz  WHERE (osoba.imie + ' ' + osoba.nazwisko) LIKE ('%'+@artysta+'%') AND obraz.nazwa LIKE ('%'+@tytul+'%')  group by obraz.nazwa, osoba.imie, osoba.nazwisko, ocena.id\_obraz, obraz.data\_powstania, epoka.nazwa, obraz.rozmiar) as filtrowanie ) = 0  THROW 50201, 'Brak wyników!', 1;  ELSE  select obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, ROUND(avg(CAST(ocena.wartosc AS FLOAT)),2) as srednia\_ocena,  YEAR(obraz.data\_powstania) as rok\_powstania, epoka.nazwa as epoka, obraz.rozmiar  from dbo.obraz inner join dbo.Artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.osoba on osoba.id\_osoba = artysta.id\_osoba  inner join dbo.epoka on epoka.id\_epoka = Obraz.id\_epoka  inner join dbo.ocena on ocena.id\_obraz = Obraz.id\_obraz  WHERE (osoba.imie + ' ' + osoba.nazwisko) LIKE ('%'+@artysta+'%') AND obraz.nazwa LIKE ('%'+@tytul+'%')  group by obraz.nazwa, osoba.imie, osoba.nazwisko, ocena.id\_obraz, obraz.data\_powstania, epoka.nazwa, obraz.rozmiar    /\*END TRY  BEGIN CATCH  THROW 50101, 'Błąd filtrowania!', 1;  END CATCH\*/  END  RETURN  END  -- prawidlowe wywołanie  EXEC up\_filtruj\_nazwy\_obrazow @artysta = 'Leo', @tytul = 'Da'  -- prawidlowe wywołanie  EXEC up\_filtruj\_nazwy\_obrazow @artysta = '', @tytul = 'Mo'  -- prawidlowe wywołanie  EXEC up\_filtruj\_nazwy\_obrazow @artysta = 'C', @tytul = ''  -- prawidlowe wywołanie  EXEC up\_filtruj\_nazwy\_obrazow @artysta = 'Dal'  -- prawidlowe wywołanie  EXEC up\_filtruj\_nazwy\_obrazow @tytul = 'Ost'  -- prawidlowe wywołanie  EXEC up\_filtruj\_nazwy\_obrazow @artysta = '', @tytul = ''  -- prawidłowe wywołanie - ale brak wyników  EXEC up\_filtruj\_nazwy\_obrazow @artysta = 'Matejko', @tytul = 'Bitwa'  -- prawidłowe wywołanie - ale brak wyników  EXEC up\_filtruj\_nazwy\_obrazow @artysta = 'Matejko'  -- prawidłowe wywołanie - ale brak wyników  EXEC up\_filtruj\_nazwy\_obrazow @tytul = 'Bitwa'  -- nieprawidlowe wywołanie - brak filtrów  EXEC up\_filtruj\_nazwy\_obrazow |

### Masowa aktualizacja danych

Pierwszym skryptem aktualizującym masowo dane, jest procedura *up\_zmiana\_galerii*, przyjmująca szereg argumentów – docelowo zmieniająca lokalizację przechowywania obrazów, na wypadek chociażby przeniesienia ich grupowo w nowe miejsce. Wynikiem tego jest możliwość przeprowadzenia prostej zmiany galerii z istniejącej na istniejącą, lub zrobienie tego zmieniając istniejącą na nową.

Procedura najpierw sprawdza, czy istnieje potrzeba dodania nowej galerii. Galerie są bezpośrednio powiązane z państwami, więc następuje walidacja, czy państwo istnieje – jeżeli tak, to po prostu zapamięta na potem odwołanie do danego państwa, w przeciwnym razie najpierw je doda, potem zapamięta odwołanie. Następnie przechodzi do faktycznej próby stworzenia nowej galerii, podając jej dane i odwołanie do państwa. Gdy ten fakt się uda, pobiera identyfikator nowo-stworzonej galerii, i zamienia istniejącą już wartość, wskazującą w której galerii znajdują się obrazy.

Jeżeli nie ma potrzeby dodania nowej galerii, system dokonuje po prostu tylko podmianę identyfikatora istniejącej galerii na inną, też istniejącą.

|  |
| --- |
| CREATE OR ALTER PROCEDURE up\_zmiana\_galerii (  @galeria\_przed int = NULL,  @galeria\_po int = NULL,  @nazwa\_galerii nvarchar(100) = NULL,  @nazwa\_panstwa nvarchar(100) = NULL,  @nazwa\_miasta nvarchar(100) = NULL,  @adres\_galerii nvarchar(255) = NULL  )  AS  BEGIN  SET NOCOUNT ON;  IF @galeria\_po IS NULL AND @galeria\_przed IS NOT NULL AND @nazwa\_panstwa IS NOT NULL AND @nazwa\_galerii IS NOT NULL AND @nazwa\_miasta IS NOT NULL AND @adres\_galerii IS NOT NULL  -- gdy trzeba dodać nową galerię  BEGIN TRY  -- sprawdzenie, czy panstwo istnieje  DECLARE @temp\_id\_panstwa int = '';  BEGIN  -- panstwo nie istnieje -> trzeba dodac  BEGIN TRY  IF (SELECT COUNT(1) FROM dbo.Panstwo WHERE Panstwo.nazwa = @nazwa\_panstwa) = 0  BEGIN  INSERT INTO dbo.Panstwo(nazwa) VALUES (@nazwa\_panstwa)  SET @temp\_id\_panstwa = (SELECT Panstwo.id\_panstwo FROM Dbo.Panstwo WHERE nazwa = @nazwa\_panstwa)  END  -- panstwo istnieje  ELSE  SET @temp\_id\_panstwa = (SELECT Panstwo.id\_panstwo FROM Dbo.Panstwo WHERE nazwa = @nazwa\_panstwa)  END TRY  BEGIN CATCH  THROW 50011, 'Błąd w pobieraniu ID Państwa', 1;  END CATCH  END  -- dodanie galerii  BEGIN TRY  INSERT INTO dbo.Galeria(nazwa, id\_panstwo, miasto, adres)  VALUES(@nazwa\_galerii, @temp\_id\_panstwa, @nazwa\_miasta, @adres\_galerii)  END TRY  BEGIN CATCH  THROW 50021, 'Błąd dodania galerii', 1;  END CATCH  -- pobranie adresu nowo-powstałej galerii  DECLARE @temp\_id\_galerii int = '';  BEGIN TRY  SET @temp\_id\_galerii = (SELECT id\_galeria FROM DBO.Galeria  WHERE nazwa = @nazwa\_galerii);  END TRY  BEGIN CATCH  THROW 50031, 'Błąd pobrania ID nowo-dodanej galerii', 1;  END CATCH  -- zmiana id galerii, do której odwołuje się obraz  BEGIN TRY  UPDATE dbo.Obraz  SET id\_galeria = @temp\_id\_galerii  WHERE id\_galeria = @galeria\_przed  END TRY  BEGIN CATCH  THROW 50041, 'Błąd w aktualizacji ID Galerii', 1;  END CATCH  END TRY  BEGIN CATCH  THROW 51001, 'Błąd w procesie aktualizacji galerii', 1;  END CATCH    -- po prostu update  IF @galeria\_po IS NOT NULL AND @galeria\_przed IS NOT NULL AND @nazwa\_galerii IS NULL AND @nazwa\_miasta IS NULL AND @nazwa\_panstwa IS NULL AND @adres\_galerii IS NULL  -- sprawdzenie, czy pierwotna galeria istnieje  IF (SELECT COUNT(1) FROM dbo.Galeria WHERE id\_galeria = @galeria\_przed) = 1  -- sprawdzenie, czy docelowa galeria istnieje  IF (SELECT COUNT(1) FROM dbo.Galeria WHERE id\_galeria = @galeria\_po) = 1  UPDATE dbo.Obraz  SET id\_galeria = @galeria\_po  WHERE id\_galeria = @galeria\_przed  ELSE  THROW 50092, 'Docelowa galeria nie istnieje!', 1;  ELSE  THROW 50091, 'Podana pierwotna galeria nie istnieje', 1;  RETURN  END  -- zmiana galerii na istniejącą  EXEC up\_zmiana\_galerii @galeria\_przed = 5, @galeria\_po = 3  -- zamiana galerii dodając nową w istniejącym mieście  EXEC up\_zmiana\_galerii @galeria\_przed = 1, @nazwa\_galerii = 'Galeria Wydziału Ekonomiczno-Socjologicznego UŁ',  @adres\_galerii = 'POW 3/5', @nazwa\_miasta = 'Łódź', @nazwa\_panstwa = 'Polska'  -- zmiana galerii dodając nowe Państwo  EXEC up\_zmiana\_galerii @galeria\_przed = 1, @nazwa\_galerii = 'University of Cambridge Gallery',  @adres\_galerii = 'The Old Schools, Trinity Ln, Cambridge CB2 1TN', @nazwa\_miasta = 'Cambridge', @nazwa\_panstwa = 'Anglia'  -- nieprawidłowe wykonanie - brak pierwotnej galerii  EXEC up\_zmiana\_galerii @galeria\_przed = 6, @galeria\_po = 1  -- nieprawidłowe wykonanie - brak galerii docelowej  EXEC up\_zmiana\_galerii @galeria\_przed = 1, @galeria\_po = 6  -- select podglądowy  select obraz.id\_galeria, Galeria.nazwa, Galeria.adres, galeria.id\_panstwo, Panstwo.nazwa, galeria.miasto, obraz.nazwa from dbo.Galeria inner join dbo.Panstwo on galeria.id\_panstwo=panstwo.id\_panstwo inner join dbo.Obraz on obraz.id\_galeria = galeria.id\_galeria |

Drugim skryptem jest procedura, która dokonuje zmiany nazwy miasta galerii na inną, podaną przez użytkownika, co może służyć w przypadkach potrzeby translacji na polski lub z języka polskiego.

Wpierw procedura upewnia się, że faktycznie miasto, które chcemy zmienić – istnieje. Jeżeli tak, dokonuje zmiany nazwy miasta na podaną w drugim parametrze.

|  |
| --- |
| CREATE OR ALTER PROCEDURE up\_zmiana\_nazwy\_miasta(@nazwa\_przed nvarchar(100), @nazwa\_po nvarchar(100))  AS  BEGIN  SET NOCOUNT ON;  -- sprawdzenie, czy pierwotne miasto istnieje  IF (SELECT COUNT(1) FROM dbo.Galeria WHERE miasto = @nazwa\_przed) > 0  UPDATE dbo.Galeria  SET miasto = @nazwa\_po  WHERE miasto = @nazwa\_przed  ELSE  THROW 50001, 'Brak podanego miasta!',1;  RETURN  END  -- select  Select DISTINCT miasto from dbo.Galeria  -- poprawne wywołanie  EXEC up\_zmiana\_nazwy\_miasta @nazwa\_przed = 'Nowy Jork', @nazwa\_po = 'New York'  -- niepoprawne wywołanie - brak miasta na liscie  EXEC up\_zmiana\_nazwy\_miasta @nazwa\_przed = 'Rzym', @nazwa\_po = 'Rome' |

### Raporty

Pierwszym skryptem do generowania raportów jest funkcja, której zadaniem jest podanie określonej liczby najlepiej ocenianych obrazów, razem z ich szczegółami.

Funkcja sumuje wszystkie oceny, pokazując, jaką średnią ocenę ma dany obraz, filtruje listę średnią oceną w sposób malejący, a następnie umożliwia określenie, ile pozycji ma pokazać, tworząc narzędzie do szybkiego ukazania określonej liczby „topowych” obrazów w danej chwili.

|  |
| --- |
| -- funkcja do wyświetlania raportu najlepiej ocenianych obrazów, w postaci ich średniej oceny, nazwy i autora  CREATE OR ALTER FUNCTION pokaz\_top\_obrazy (@ilosc int)  RETURNS TABLE  AS  RETURN  (select top(@ilosc) ROUND(avg(CAST(ocena.wartosc AS FLOAT)),2) as srednia\_ocena, obraz.nazwa, osoba.imie + ' ' + osoba.nazwisko as artysta, ocena.id\_obraz  from dbo.ocena inner join dbo.obraz on ocena.id\_obraz=obraz.id\_obraz  inner join dbo.artysta on obraz.id\_artysta = artysta.id\_artysta  inner join dbo.Osoba on artysta.id\_osoba = osoba.id\_osoba  group by ocena.id\_obraz, obraz.nazwa, osoba.imie, osoba.nazwisko  order by srednia\_ocena desc)  -- Wywołanie  SELECT \* from pokaz\_top\_obrazy(5) |

Drugim skryptem do raportowania jest funkcja, która pokazuje listę najpopularniejszych epok, wraz z ich najbardziej twórczymi przedstawicielami. Sortuje same epoki malejąco po fakcie posiadanych obrazów, a obok jednocześnie podaje, ile obrazów stworzył dany artysta.

|  |
| --- |
| -- funkcja do wyświetlania raportu najpopularniejszych epok, wraz z ich przedstawicielami i iloscia obrazow przez nich stworzonych  CREATE OR ALTER FUNCTION pokaz\_najpopularniejsze\_epoki (@ilosc int)  RETURNS TABLE  AS  RETURN  (select top(@ilosc) epoka.nazwa, count(epoka.nazwa) as ilosc\_obrazow, osoba.imie + ' ' + osoba.nazwisko as artysta  from dbo.Epoka inner join dbo.Obraz on obraz.id\_epoka = epoka.id\_epoka  inner join dbo.Artysta on artysta.id\_artysta = obraz.id\_artysta  inner join dbo.Osoba on osoba.id\_osoba = Artysta.id\_osoba  group by epoka.nazwa, osoba.imie, osoba.nazwisko  order by ilosc\_obrazow desc)  -- Wywołanie  SELECT \* from pokaz\_najpopularniejsze\_epoki(2) |