informatyka+

Algorytmika i programowanie

Bazy danych

Multimedia, grafika i technologie internetowe

Sieci komputerowe

Tendencje w rozwoju informatyki i jej zastosowań

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

informatyka+

Wszechnica Informatyczna:

Procedury, funkcje, wyzwalacze – programowanie w T-SQL

Andrzej Ptasznik

Człowiek – najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Procedury, funkcje, wyzwalacze – programowanie w T-SQL

Rodzaj zajęć: Wszechnica Informatyczna
Tytuł: Procedury, funkcje, wyzwalacze – programowanie w T-SQL
Autor: mgr inż. Andrzej Ptasznik

Zeszyt dydaktyczny opracowany w ramach projektu edukacyjnego **Informatyka+** — ponadregionalny program rozwijania kompetencji uczniów szkół ponadgimnazjalnych w zakresie technologii informacyjno-komunikacyjnych (ICT).
www.informatykaplus.edu.pl
kontakt@informatykaplus.edu.pl

Wydawca: Warszawska Wyższa Szkoła Informatyki
ul. Lewartowskiego 17, 00-169 Warszawa
www.wwsi.edu.pl
rektorat@wwsi.edu.pl

Projekt graficzny: FRYCZ I WICHA

Warszawa 2012

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Warszawska Wyższa Szkoła Informatyki

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

Procedury, funkcje, wyzwalacze – programowanie w T-SQL

Andrzej Ptasznik
Warszawska Wyższa Szkoła Informatyki
aptaszni@wwsi.edu.pl

Streszczenie

W ramach kursu „Procedury, funkcje, wyzwalacze – programowanie w T-SQL" uczestnicy zapoznani zostaną z MS SQL Server 2008 R2. Poznają podstawy projektowania relacyjnych baz danych, wykonają instalację instancji SQL Server 2008 R2 Express. Praktycznie wykonają implementację zaprojektowanej bazy danych. Nauczą się jak określać i definiować reguły poprawności dla wybranych danych. W kolejnych zadaniach wyjaśnione zostaną problemy definiowania widoków i ich wykorzystanie. Omówione zostaną podstawy języka T-SQL i wykorzystanie tego języka przy programowaniu procedur, funkcji i wyzwalaczy. Problemy omawiane w trakcie kursu oraz realizowane ćwiczenia powinny umożliwić uczestnikom samodzielne tworzenie prostych baz danych i ich programowanie.

Spis treści

1 WPROWADZENIE

Współczesne technologie baz danych są bardzo złożone i rozległe. Trudno dzisiaj mówić o specjalistach baz danych, gdyż w ramach tej dziedziny wydzielić można wiele pomniejszych specjalizacji i każda z nich jest wystarczająco duża i skomplikowana. W dziedzinie baz danych można wydzielić kilka podstawowych specjalizacji:

- Projektant baz danych – osoba zajmująca się projektowaniem struktury baz danych i logiki biznesowej,
- Administrator baz danych – osoba zajmująca się zarządzaniem serwerami baz danych i różnymi aspektami bezpieczeństwa danych,
- Programista baz danych – osoba zajmująca się programowaniem po stronie serwera (procedury, funkcje, wyzwalacze),
- Programista interfejsów do baz danych – osoba tworząca oprogramowanie użytkowe korzystające z baz danych.

Do każdej specjalizacji literatura podstawowa obejmuje pozycje zawierające tysiące stron. Ten wstęp wyjaśnia, że w ramach naszego kursu nie będziemy zajmować się całokształtem problemów związanych z bazami danych i technologią MS SQL Server 2008 R2, a jedynie pewnym wycinkiem problemów związanych z programowaniem po stronie serwera bazy danych. Musimy jednak zdawać sobie sprawę, że jest to początek drogi do tego, żeby stać się pełnoprawnym specjalistą z danej dziedziny.

2 TECHNOLOGIA MS SQL SERVER 2008 R2

2.1. ELEMENTY TECHNOLOGII MS SQL SERVER 2008

Systemem Zarządzania Bazami Danych(SZBD) nazywamy specjalistyczne oprogramowanie umożliwiające tworzenie baz danych oraz ich eksploatację.

Wydaje się oczywiste, że tworzenie i działanie baz danych musi być wspierane przez specjalistyczne oprogramowanie, które powinno umożliwiać realizacje pewnych zadań:

- definiowanie obiektów bazy danych,
- manipulowanie danymi,
- generowanie zapytań,
- zapewnienie spójności i integralności danych.

Zadania te brzmią bardzo ogólnie, obejmują jednak większość potrzeb w zakresie tworzenia i eksploatacji baz danych. Dla przybliżenia pojęcia SZBD można podać kilka nazw handlowych, pod jakimi te produkty można spotkać na rynku i w zastosowaniach: MS SQL Server 2008, Oracle, MySQL, Access, DB2 i wiele, wiele innych mniej lub bardziej popularnych.

Jednym z najważniejszych zadań stojących przed SZBD jest zapewnienie spójności i integralności danych, czyli dostarczenie mechanizmów zapewniających przestrzeganie określonych reguł przez dane. SZBD dostarczają mechanizmy służące do zapewnienia spójności i integralności danych, czyli mówiąc innymi słowami, zapewnienia logicznej poprawności danych zapisanych w bazie. Podstawowe mechanizmy, realizujące te zadania to:

- deklaracja typu,
- definicje kluczy,
- reguły poprawności dla kolumny,
- reguły poprawności dla wiersza,
- reguły integralności referencyjnej.

Systemy Zarządzania Bazami Danych, oparte na modelu relacyjnym, w dalszym ciągu burzliwie się rozwijają. Średnio co trzy lata dostarczane są nowe wersje systemów, które wprowadzają szereg nowych funkcji i technologii.

W ramach kursu będziemy wykorzystywać technologię MS SQL Server 2008 R2. Jest to jeden z najpopularniejszych serwerów baz danych. Edycja SQL Server Express jest wersją darmową z możliwością wykorzystania jej w celach komercyjnych. Technologia SQL Server 2008 zawiera następujące podsystemy:

- Serwer bazy danych (Database Engine) – podsystem odpowiedzialny za zarządzanie bazami danych (definiowanie, eksploatacja i administracja baz danych),
- Serwer raportowania (Reporting Services) – podsystem umożliwiający zarządzanie procesem tworzenia i dystrybucji raportów generowanych na podstawie danych z różnych źródeł (bazy danych, pliki Excel, pliki tekstowe, dokumenty XML),
- Serwer usług analitycznych (Analysis Services) – podsystem wspomagający organizację hurtowni danych, wielowymiarowych kostek analitycznych, tworzenie pulpitów menadżerskich oraz realizację algorytmów wyszukiwania złożonych zależności (Data Mining),
- Serwer usług integracyjnych (Integration Services) – podsystem realizujący zadania integracji danych, polegające w dużym uproszczeniu na pobieraniu danych z pewnych źródeł danych, poddanie ich procesowi przetwarzania (sprawdzanie poprawności, eliminowanie błędów itp.), a następnie zapisanie przetworzonych danych w docelowej lokalizacji. Zadania te są określane w teorii jako platforma ET&L (Extract, Transform and Load).

Silnik bazy danych (Database Engine) zawiera wiele różnych dodatkowych technologii:

- Usługi asynchronicznego przetwarzania (Service Broker) – umożliwiają realizację asynchronicznego przetwarzania z wykorzystanie kolejek
- Usługi replikacji danych – umożliwiają konfigurowanie zadań związanych z odtwarzaniem części zasobów bazy danych w innych lokalizacjach
- Usługi wyszukiwania pełno tekstowego – umożliwiają wyszukiwanie fragmentów tekstu w dużych zasobach tekstowych

Wymienione zostały niektóre elementy technologii MS SQL Server 2008, co i tak pokazuje, że jest to bardzo rozległy i złożony system umożliwiający realizacje bardzo różnych zadań związanych z bazami danych.

2.2. ĆWICZENIA

2.2.1. Ćwiczenie 1 – Instalacja instancji MS SQL Server 2008 Express

Produkt MS SQL Server 2008 R2 Express jest w pełni funkcjonalnym serwerem baz danych udostępnianym za darmo do zastosowań komercyjnych. W porównaniu z płatnymi licencjami produktu jest on pozbawiony niektórych mechanizmów związanych z obsługą bardzo dużych baz danych oraz posiada kilka ograniczeń:

- wykorzystuje jedynie jeden procesor,
- wykorzystuje tylko 1 GB pamięci operacyjnej,
- rozmiar pliku jednej bazy danych nie może przekroczyć 10 GB.

Ograniczenia te nie stanowią większego problemu przy tworzeniu małych i średnich baz danych.

W ramach ćwiczenia zainstalujemy na komputerach uczestników instancję serwera.

Produkt można pobrać ze strony producenta, czyli firmy Microsoft, z adresu internetowego: http://www.microsoft.com/download/en/details.aspx?id=26729.

Po uruchomieniu strony należy wybrać opcję pobierania odpowiedniego produktu. W naszym przypadku pobieramy pełną wersję z dodatkowymi narzędziami. Wybór opcji pokazano na rysunku 1.

Po wciśnięciu odpowiedniego przycisku DOWNLOAD, rozpocznie się proces pobierania, który potrwa, w zależności od przepustowości sieci od kilku do kilkudziesięciu minut. Po pobraniu produktu automatycznie rozpocznie się proces instalacji.

Rysunek 1.
Strona producenta SQL Server 2008 – wybór pobierania produktu

W oknie instalatora wybieramy opcję nowej instalacji, jak pokazano na rysunku 2.

Rysunek 2.
Okno podstawowe instalatora

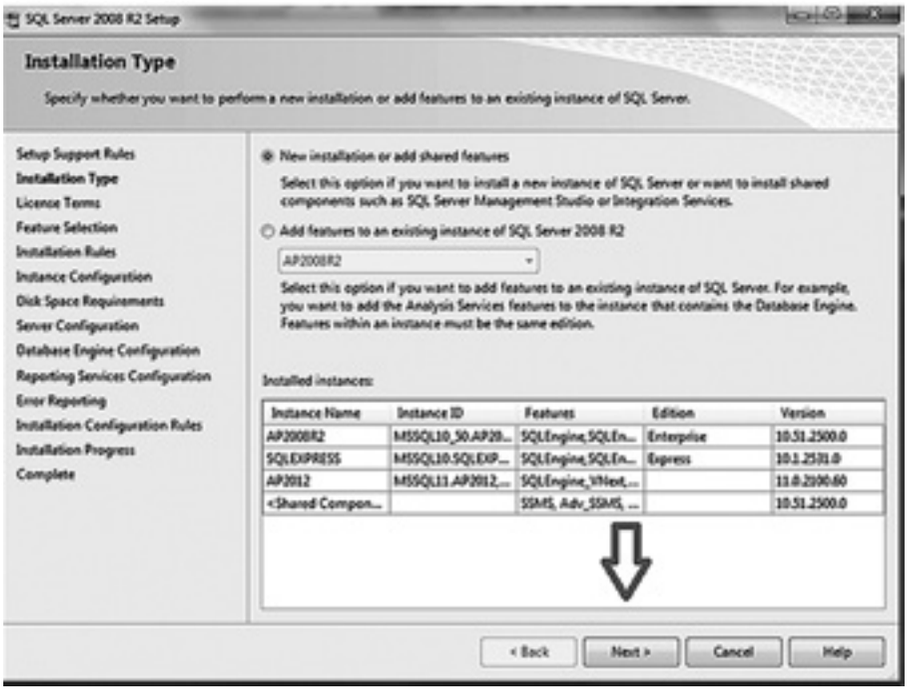
Po automatycznym wykonaniu przez instalatora czynności wstępnych, pojawi się okno (rysunek 3) procesu instalacji. W oknie klikamy przycisk Next.

W następnym kroku musimy zaakceptować warunki licencji (jest ona bezpłatna) tak jak pokazano na rysunku 4 i wcisnąć przycisk Next.

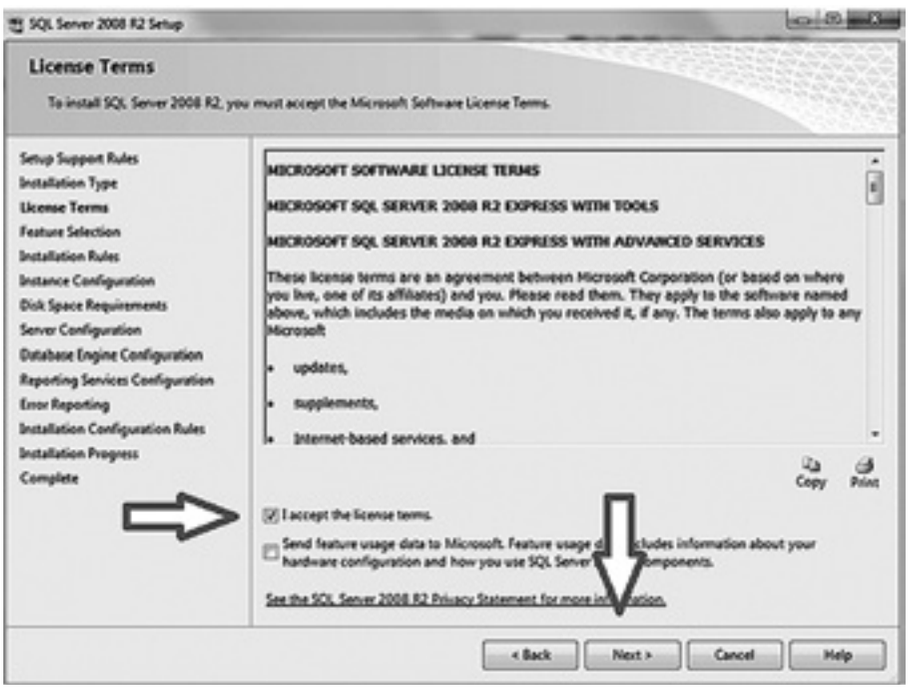
W kolejnym oknie wybieramy elementy technologii, które mają zostać zainstalowane. Wciskamy opcję Select ALL a następnie przycisk Next, tak jak pokazano na rysunku 5.

W kolejnym kroku (rysunek 6) musimy nazwać instancje naszego serwera. Po wprowadzeniu nazwy klikamy przycisk Next.

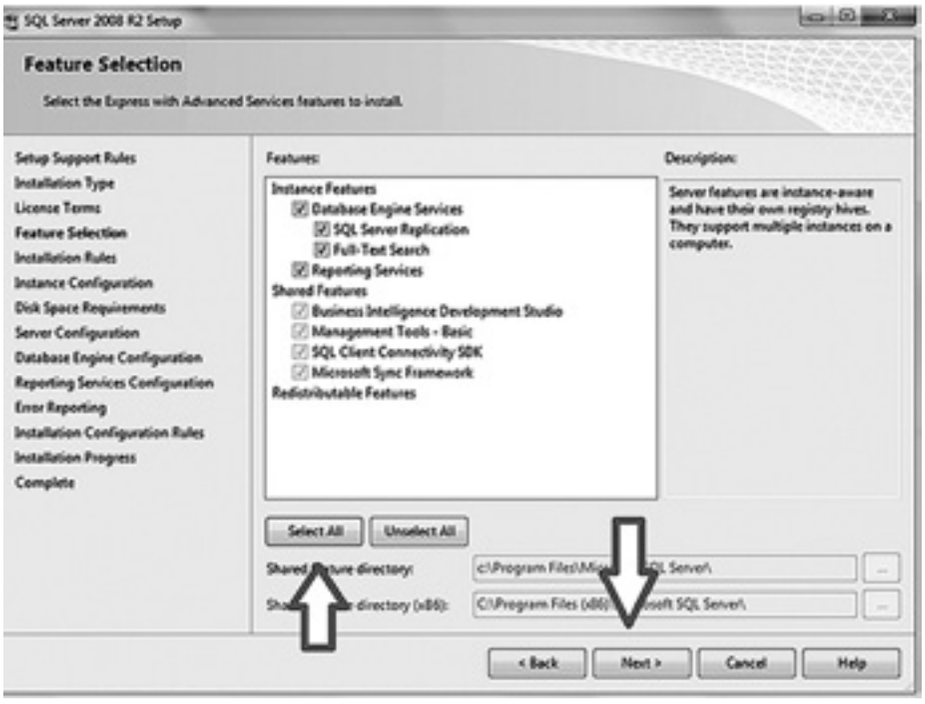
W kolejnym kroku procesu instalacji ustalamy konta, na których będą uruchomione odpowiednie usługi SQL Servera. W naszym przypadku przyjmiemy wartości domyślne, zaproponowane przez instalator i klikniemy przycisk Next (rysunek 7).



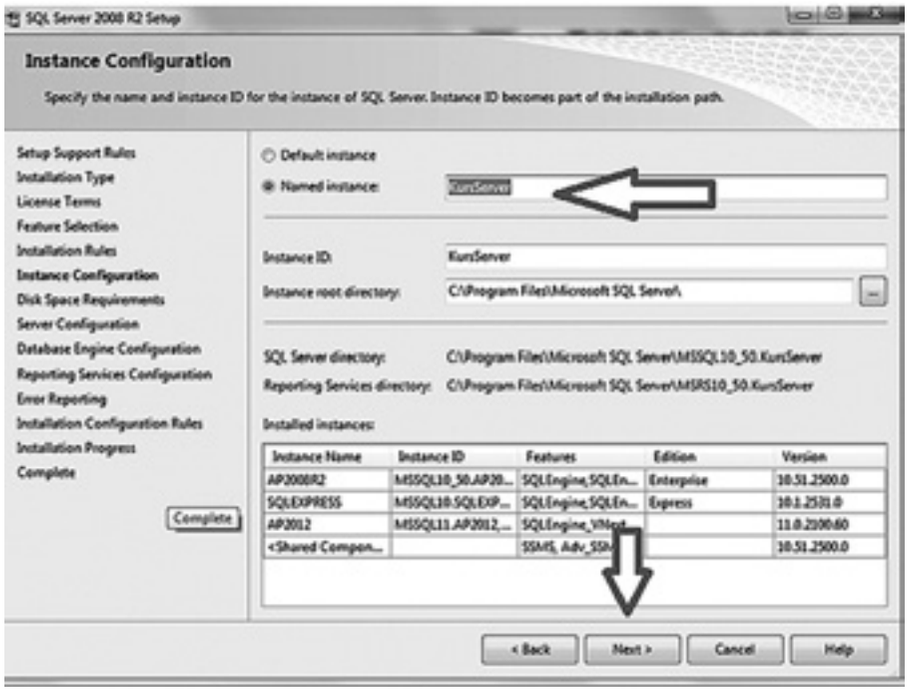
Rysunek 3.
Okno rozpoczęcia procesu instalacji



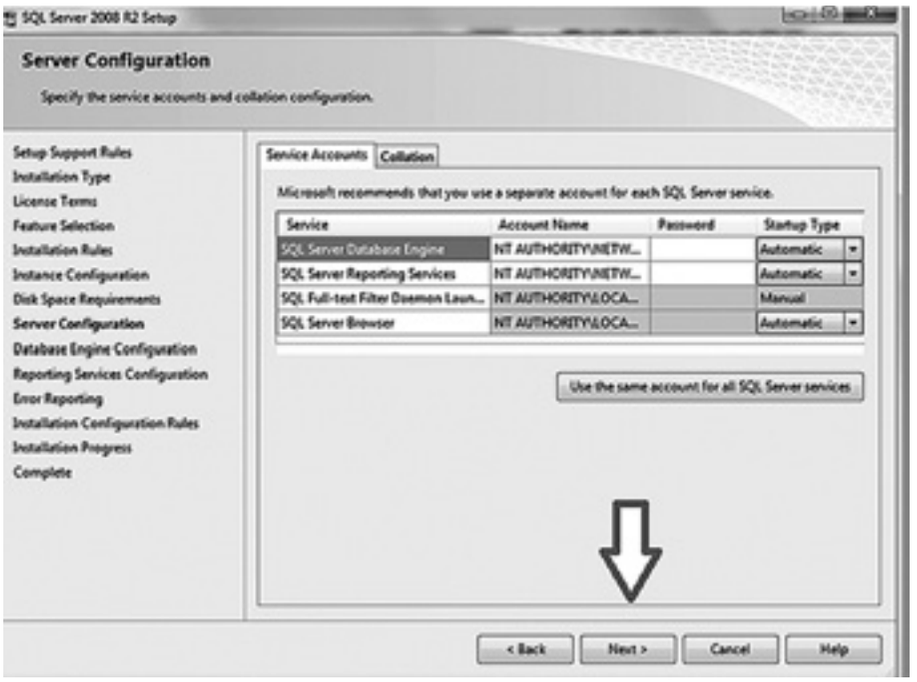
Rysunek 4.
Okno akceptowania postanowień licencji produktu.



Rysunek 5.
Okno wyboru elementów technologii do instalacji

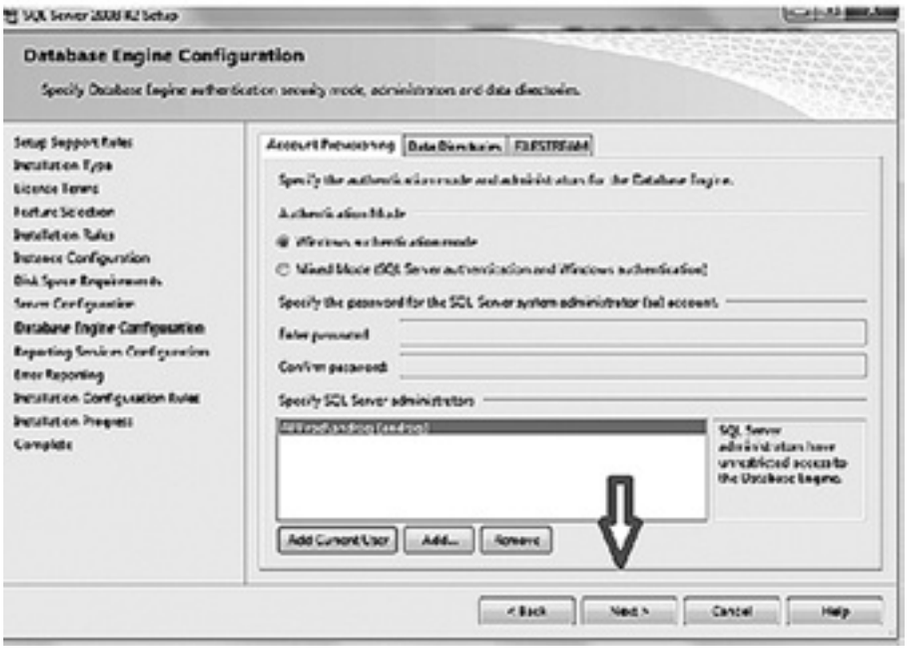


Rysunek 6.
Okno instalatora – ustalanie nazwy instancji serwera



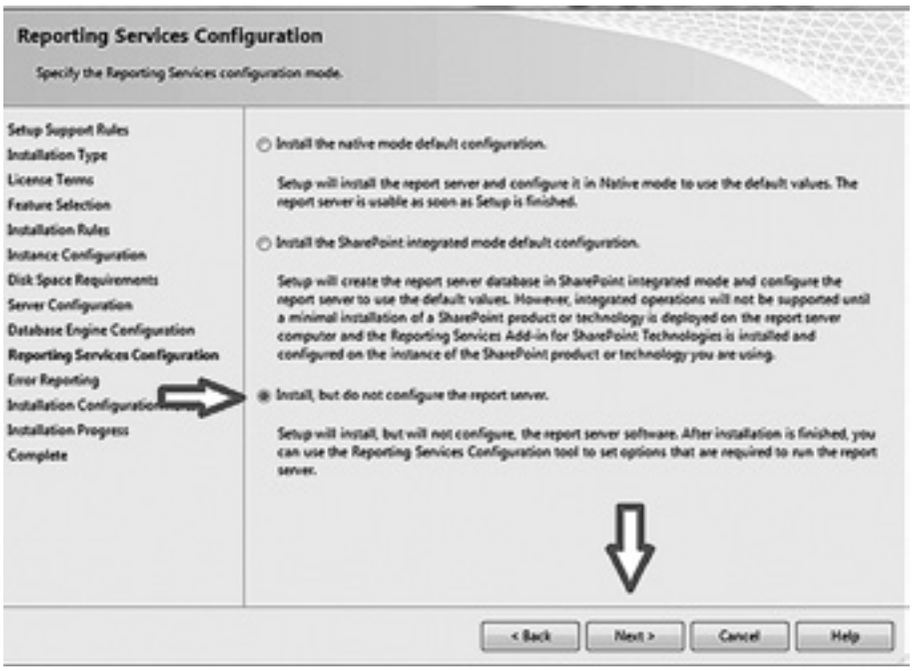
Rysunek 7.
Okno wyboru kont dla usług serwera

Następnie ustalamy tryb uwierzytelnienia SQL Server – w naszym przypadku przyjmujemy wartości domyślne i naciskamy przycisk Next (rysunek 8).



Rysunek 8.
Okno wyboru trybu uwierzytelniania

W kolejnym kroku należy ustalić sposób konfigurowania usługi Reporting Services. Wybieramy opcję „Install, but not configure the report server" (rysunek 9) i wciskamy przycisk Next.



Rysunek 9.
Okno wyboru trybu konfiguracji usługi Reporting Services

Po wykonaniu omówionych etapów rozpocznie się proces instalacji, który może potrwać kilkadziesiąt minut.

Uwaga: W trakcie wykonywania poszczególnych etapów ćwiczenia zostaną omówione szczegóły instalacji przez prowadzącego kurs.

2.2.2. Ćwiczenie 2 – Zapoznanie ze środowiskiem SQL Management Studio.

SQL Server Management Studio to podstawowe narzędzie administracji systemu SQL Server, które pojawiło się w wersji SQL Server 2005. Za jego pomocą możliwe jest:

- tworzenie, edycja i usuwanie baz danych i obiektów baz danych,
- zarządzanie zadaniami, np. wykonywanie kopii zapasowych,
- wyświetlanie informacji dotyczących bieżącej aktywności, np. zalogowanych użytkowników,
- zarządzanie bezpieczeństwem,
- zarządzanie usługami pocztowymi bazy danych,
- tworzenie katalogów wyszukiwania pełnotekstowego i zarządzanie nimi,
- tworzenie i zarządzanie bazami publikatorów i subskrybentów na potrzeby replikacji baz danych,
- i wiele, wiele innych zadań.

Uwaga! SQL Server Management Studio to tylko wygodne narzędzie do obsługi SQL Server, nie jest ono jednak niezbędne do jego działania.

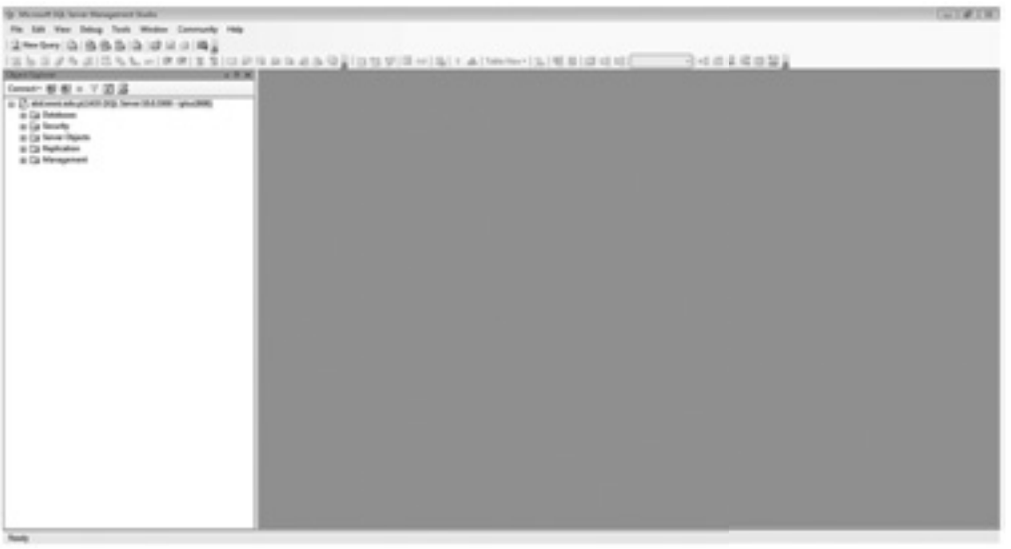
W celu zapoznania się z podstawowymi funkcjami tego środowiska realizujemy następujące czynności:

1. Uruchamiamy program SQL Server Management Studio 2008 R2
2. Po uruchomieniu programu pojawi się okienko logowania

Należy podać nazwę serwera baz danych (pole Server name), wybrać tryb uwierzytelnienia (pole Autentication) oraz parametry logowania (pola Login i Password).

Rysunek 10.
Okno logowania programu SQL Server Management Studio

3. Po zalogowaniu uruchamia się główny panel programu MS SQL Server Management Studio.

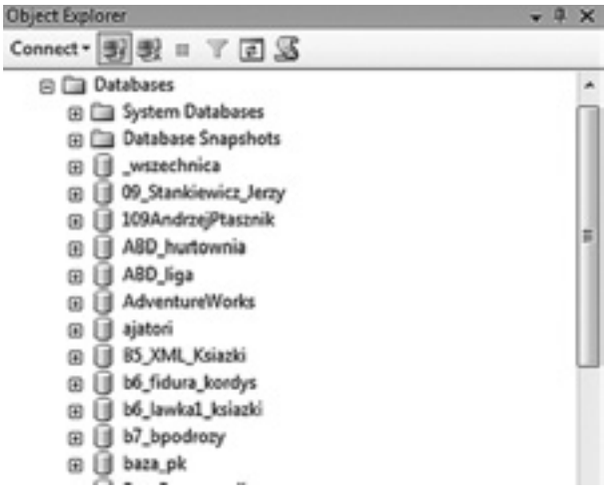


Rysunek 11.
Postać wyjściowa panelu MS SQL Server Management Studio 2008 R2

SQL Server Management Studio 2008 R2 pozwala na realizację większości zadań związanych z definiowaniem, administracją i eksploatacją baz danych. Ogólną postać interfejsu pokazano na rysunku 11. Najistotniejszym elementem tego interfejsu jest okno Object Explorer, którego ogólna postać pokazana została na rysunku 12.

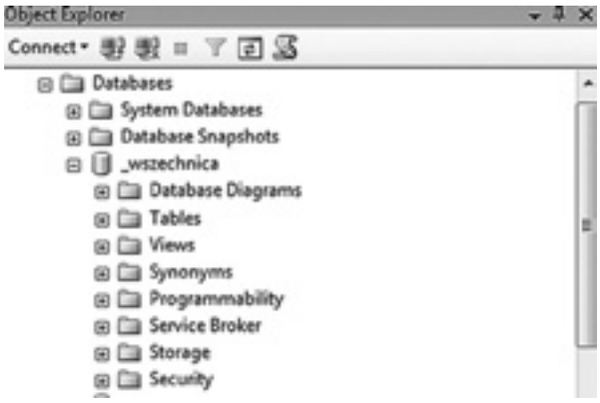
W oknie Object Explorer w zależności od wybranego kontekstu można uzyskać dostęp do różnych, hierarchicznie zdefiniowanych obiektów serwera. Kontekst pierwszy przedstawia podstawowe elementy serwera. W ramach naszego kursu istotny będzie jedynie dostęp do baz danych, które są zawarte w folderze Databases (rysunek 12). Rozwinięcie folderu Databases, jak pokazano na rysunku 13, wyświetla wszystkie bazy danych zdefiniowane na instancji serwera.

Rysunek 12.
Okno Object Explorer – kontekst ogólny



Rysunek 13.
Okno Object Explorer – kontekst Databases

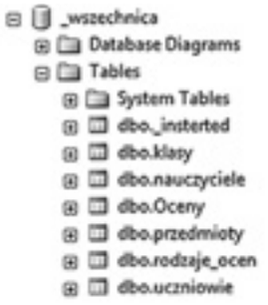
W tym kontekście możemy uzyskać dostęp do szczegółów definicji wszystkich baz danych. Dla wybranej bazy danych możemy przejść do widoku jest szczegółów. Kontekst wybranej bazy danych pokazano na rysunku 14.



Rysunek 14.
Okno Object Explorer – kontekst wybranej bazy danych

W tym kontekście uzyskujemy dostęp do tabel, widoków, elementów programowania bazy danych i innych bardziej zaawansowanych elementów takich jak Service Broker czy Security.

Na rysunku 15 pokazano kontekst Tables, w ramach którego widoczne są wszystkie tabele zdefiniowane w wybranej bazie danych.

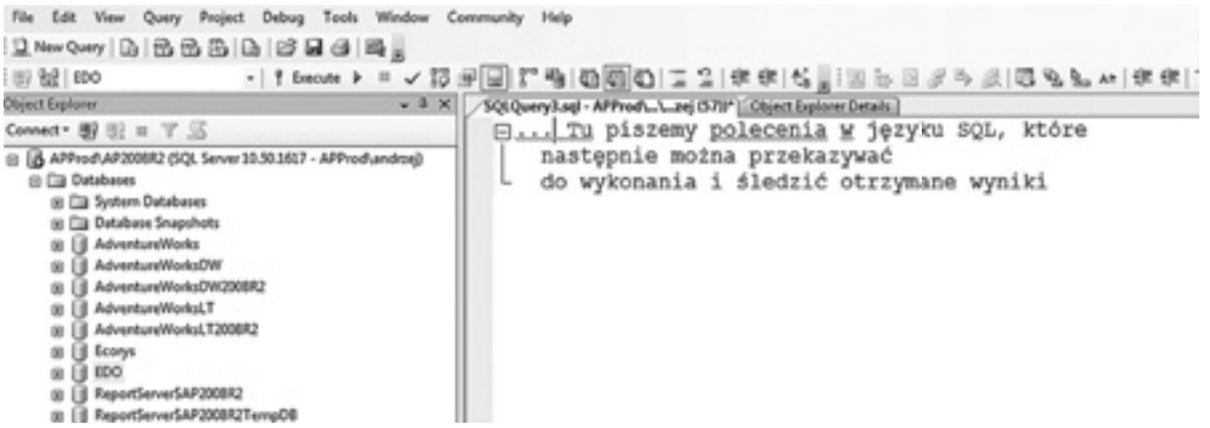


Rysunek 15.
Okno Object Explorer – kontekst Tables

Dla wybranej tabeli, co pokazano na rysunku 16, możemy otrzymać widok jej szczegółów. Uzyskujemy dostęp do definicji kolumn, kluczy i ograniczeń, a także do wyzwalaczy (ang. *triggers*), indeksów i statystyk. Ramy naszego kursu nie obejmują wielu z tych pokazanych elementów technologii, ale widać wyraźnie, że okno Object Explorer jest zorganizowane na zasadzie „od ogółu do szczegółu" i jest możliwość dostępu do każdego szczegółu bazy danych.

Rysunek 16.
Okno Object Explorer – kontekst wybranej tabeli

Oprócz dostępu do zdefiniowanych i utworzonych na serwerze baz danych i ich szczegółów, w ramach SQL Server Management Studio 2008 R2, możemy pisać i wykonywać polecenia i skrypty pisane w języku SQL. W tym celu należy otworzyć okno edycyjne (opcja New Query) i otrzymamy widok pokazany na rysunku 17.



Rysunek 17.
Uruchamianie okna New Query

Warto wiedzieć, że każde otwarte okno New Query tworzy odrębną sesję połączeniową z bazą danych, dlatego nie jest dobrym zwyczajem otwieranie niezliczonej ilości tych okien (choć teoretycznie możemy ich otworzyć bardzo dużo). Teks zapisany w oknie Query możemy, jeżeli jest taka konieczność, zapisać jako plik tekstowy (z rozszerzeniem .SQL). Problemy i wątpliwości związane z wykorzystaniem SQL Serwer Management Studio 2008 R2 można w ramach kursy wyjaśnić z prowadzącym.

3 TWORZENIE BAZY DANYCH

W ramach kolejnych ćwiczeń, każdy uczestnik kursu utworzy bazę danych, przykładowe tabele i zdefiniuje kilka wybranych ograniczeń.

3.3. ĆWICZENIA

3.3.1. Ćwiczenie 3 – Tworzenie bazy danych.

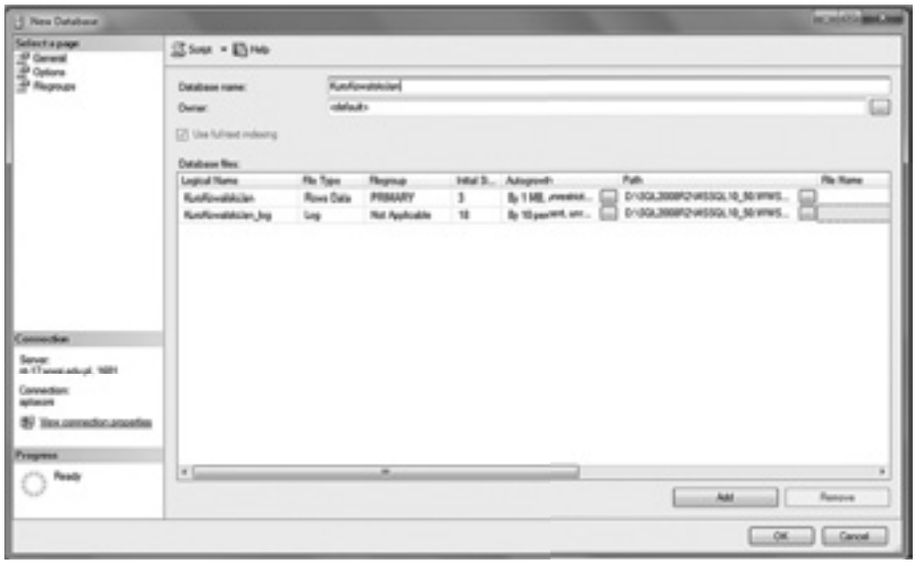
Każdy uczestnik kursu utworzy bazę danych na dostępnym serwerze o nazwie KURS*NazwiskoImie* (prefix KURS oraz nazwisko i imię uczestnika kursu). Dalsze ćwiczeniach będą realizowane w ramach utworzonej bazy danych.

Tworzenie bazy danych realizujemy w SQL Server Management Studio według następującego schematu:

1. Wybieramy opcję New Database (klikamy prawym klawiszem myszy na folder Databases w oknie Object Explorer) – jak pokazano na rysunku 18.

Rysunek 18.
Wybór opcji New Database

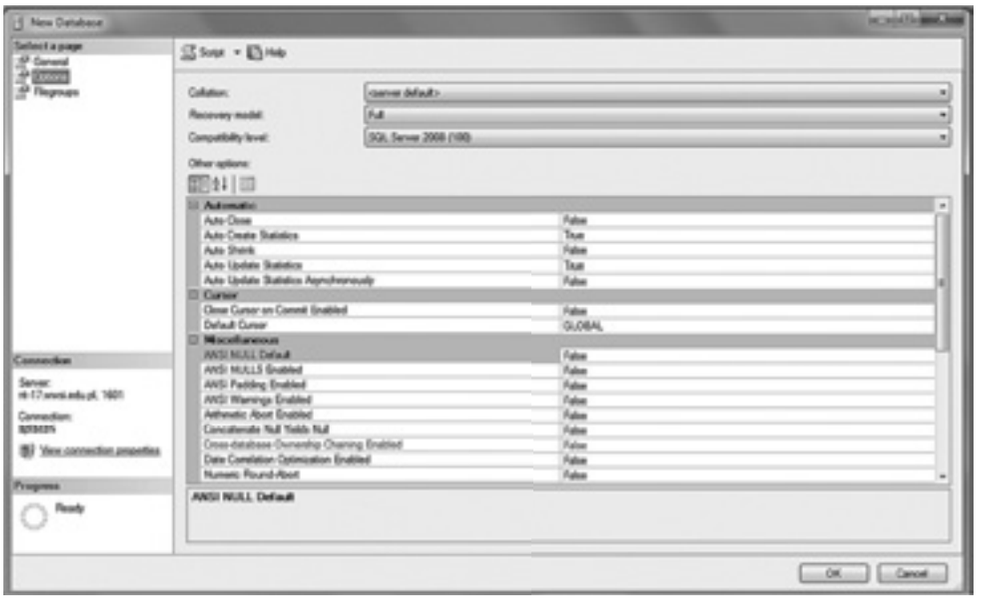
2. Po uruchomieniu opcji New Database, pojawi się kreator nowej bazy danych.



Rysunek 19.
Kreator tworzenia bazy danych

W polu „Database name” wpisujemy nazwę bazy danych – KURSNazwiskoImie (prefix KURS oraz nazwisko i imię uczestnika kursu). Po zapisaniu nazwy bazy danych przechodzimy na stronę Options.

3. Przejście na stronę Options.



Rysunek 20.
Strona Options

W polu „Recovery model” wybieramy wartość Simple (domyślnie jest wybrana wartość Full). Przestawienie tej opcji jest konieczne, ponieważ wartość Full wymaga uruchomienia automatycznego wykonywania kopii dziennika transakcyjnego (jest to konieczne w systemach produkcyjnych). Po przestawieniu opcji w oknie Recovery model, wracamy na stronę General i wybieramy przycisk OK. Baza danych o podanej nazwie zostanie utworzona.

W oknie Object Explorer uzyskamy dostęp do utworzonej bazy danych, jak pokazano na rysunku 21.

Baza danych jest gotowa do eksploatacji, w pierwszym kroku zdefiniujemy w bazie danych dwie przykładowe tabele.

3.3.2. Ćwiczenie 4 – Definiowanie tabel.

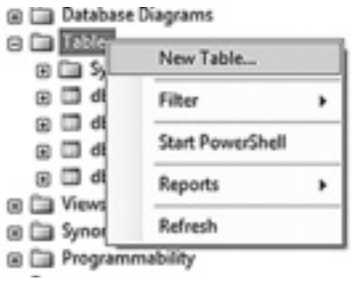
W ramach ćwiczenia zostaną zdefiniowane dwie przykładowe tabele. Tworzenie nowej tabeli polega na zdefiniowaniu kolumn, z których tabela będzie złożona. W tym celu, po wciśnięciu prawego klawisza myszy na folderze Table, wybieramy opcję New Table, jak pokazano na rysunku 22.

Po wybraniu opcji New Table pojawi się tabela, w której określamy nazwy kolumn ich typ oraz dodatkowe własności. W ramach ćwiczenia należy zdefiniować dwie tabele. Pierwszą tabelę o nazwie Osoby, definiujemy zgodnie z wzorem pokazanym na rysunku 23.

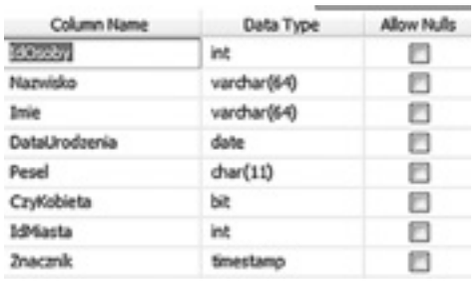
Dodatkowo dla kolumny IdOsoby, ustawiamy własność Identity Specification, tak jak pokazano na rysunku 24, co spowoduje automatyczną numerację kolumny IdOsoby.

Po zdefiniowaniu tabeli, zgodnie ze wzorem, zamykamy okno projektanta i w oknie, które się pojawi, określamy jej nazwę jak pokazano na rysunku 25.

Rysunek 21.
Widok okna Object Explorer po utworzeniu nowej bazy danych



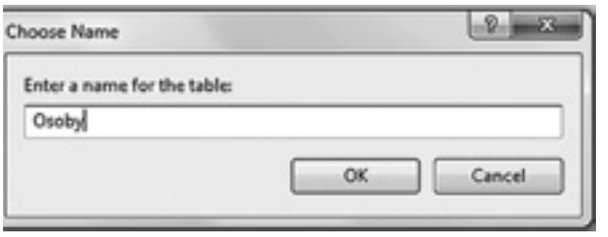
Rysunek 22.
Wybór zadania New Table



Column Name	Data Type	Allow Nulls
IdOsoby	int	☐
Nazwisko	varchar(64)	☐
Imie	varchar(64)	☐
DataUrodzenia	date	☐
Pesel	char(11)	☐
CzyKobieta	bit	☐
IdMiasta	int	☐
Znacznik	timestamp	☐

Rysunek 23.
Struktura tabeli Osoby

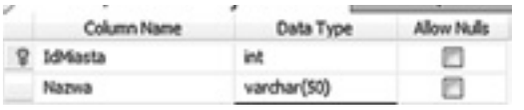
Rysunek 24.
Ustawianie własności Identity Specification



Rysunek 25.
Ustalenie nazwy tworzonej tabeli

Kolumna o nazwie Znacznik.

Drugą tabelę, o nazwie Miasta, definiujemy zgodnie ze wzorem przedstawionym na rysunku 26.



Rysunek 26.
Struktura tabeli Miasta

Podobnie, jak przy definiowaniu tabeli Osoby, ustawiamy dla kolumny IdMiasta własność Identity Specification i po zamknięciu okna projektanta nadajemy tabeli nazwę Miasta. Po wykonaniu ćwiczenia nasza baza danych składa się z dwóch tabel jak pokazano na rysunku 27.

Rysunek 27.
Struktura bazy danych po wykonaniu ćwiczenia

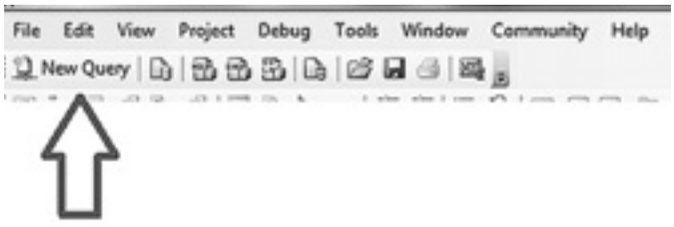
W kolejnych ćwiczeniach, dla utworzonych tabel, zdefiniujemy reguły poprawności danych.

3.3.3. Ćwiczenie 5 – Definiowanie reguł poprawności CHECK.

Tabele utworzone w ramach ćwiczenia 4 zapewniają poprawność danych jedynie w zakresie zdefiniowanych typów danych (typ danych jest określeniem dziedziny dopuszczalnych wartości). Bardzo często jest to ograniczenie niewystarczające, dlatego należy definiować dodatkowe ograniczenia. Dla tabel zdefiniowanych w ramach ćwiczenia 4 zdefiniujemy następujące ograniczenia:

- zapewnienie, żeby w kolumnie Nazwa tabeli Miasta zapisywane były tylko unikalne wartości,
- zapewnienie, żeby numer Pesel składał się dokładnie z 11 cyfr, a dodatkowo był zgodny z datą urodzenia i płcią.

Regułę poprawności definiujemy poprzez wykonanie odpowiedniego polecenia w języku SQL. Edycje polecenia i jego wykonanie wykonujemy w oknie New Query, które otwieramy poprzez wybranie opcji New Query jak pokazano na rysunku 28.

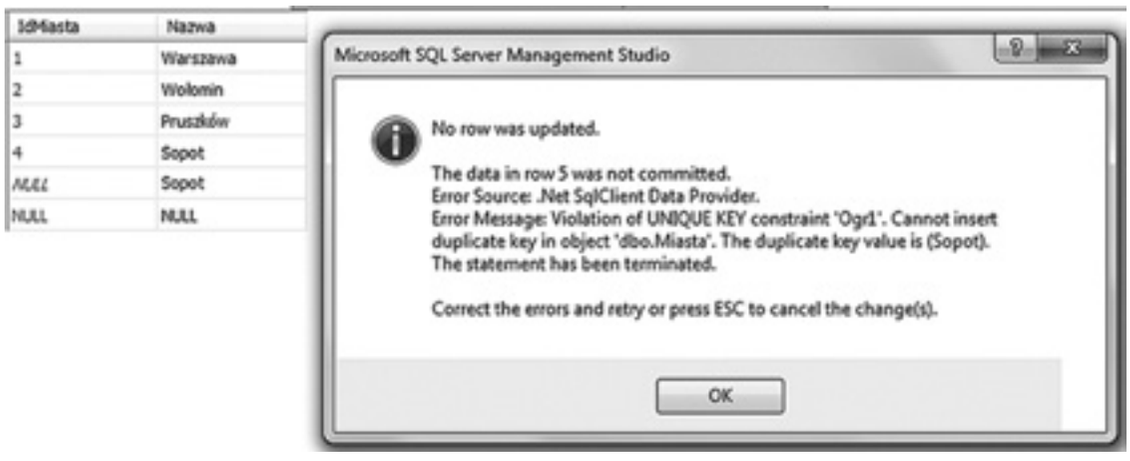


Rysunek 28.
Wybór opcji New Query

Pierwsze polecenie definiuje ograniczenie typu UNIQUE dla kolumny Nazwa w tabeli Miasta. W oknie New Query wpisujemy następujące polecenie:

```
ALTER TABLE Miasta
ADD CONSTRAINT Ogr1 UNIQUE(Nazwa)
```

Wykonujemy polecenie zapisane w oknie wciskając klawisz F5. Po wykonaniu polecenia, każda próba wprowadzenia do tabeli nazwy miasta, która już jest zapisana, wygeneruje błąd jak pokazano na rysunku 29.



Rysunek 29.
Błąd wygenerowany przy próbie zapisania nazwy miasta już zapisanej w tabeli

W kolejnym poleceniu zdefiniujemy ograniczenie zapewniajace poprawność numeru Pesel. W sklad tej reguły wchodzą trzy warunki:

- numer Pesel musi składać się dokładnie z 11 cyfr,

- sześć pierwszych cyfr numeru Pesel musi być zgodne z datą urodzenia,
- przedostatnia cyfra numeru Pesel musi być zgodna z płcią (kobiety – wartość parzysta, mężczyźni – wartość nieparzysta).

W celu zdefiniowania opisanego ograniczenia, w oknie New Query piszemy następujące polecenie:

```
ALTER TABLE Osoby
ADD  CONSTRAINT Ogr3 CHECK
(
  Pesel LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' AND
  DataUrodzenia='19'+SUBSTRING(Pesel, 1, 6) AND
  CASE
    WHEN CzyKobieta=1 AND SUBSTRING(Pesel, 10, 1) % 2=0 THEN 1
    WHEN CzyKobieta=0 AND SUBSTRING(Pesel, 10, 1) % 2=1 THEN 1
    ELSE 0
  END = 1
)
```

Wykonujemy polecenie wciskając klawisz F5.

Szczegóły polecenia omówić z prowadzącym kurs. Powyższe polecenie sprawdza poprawność numeru Pesel dla osób urodzonych w XX wieku. Proszę się zastanowić i zmodyfikować polecenie tak, żeby uwzględniało osoby urodzone w weku XXI.

Po wykonaniu zadań (poleceń) w ramach tego ćwiczenia, ustalone reguły będą przestrzegane przy każdym zapisywaniu nowego wiersza i przy każdej modyfikacji i jeżeli dane nie będą zgodne z regułami, to zostanie wygenerowany odpowiedni błąd.

Sprawdzenie działania zdefiniowanych reguł dokonamy w tracie wprowadzania danych do tabel.

3.3.4. Ćwiczenie 6 – Definiowanie reguł integralności referencyjnej.

Pojęcie integralność referencyjna określa zależności występujące miedzy danymi zapisanymi w wielu tabelach. Dla naszych przykładowych tabel powinniśmy zdefiniować regułę klucza obcego, czyli kolumny IdMiasta w tabeli Osoby. Zapewnienie poprawnych wartości kolumny klucza obcego wymaga spełnienia następujących zasad:

- nie można wprowadzić do kolumny IdMiasta w tabeli Osoby wartości klucza, który nie występuje w tabeli Miasta (nie można powiązać osoby z miastem, które nie istnieje),
- nie można zmodyfikować wartości w kolumnie IdMiasta w tabeli Osoby na wartość, która nie występuje w tabeli Miasta,
- nie można usunąć z tabeli Miasta wiersza, jeżeli są w tabeli Osoby wiersze powiązane(poprzez wartość klucza).

Wymuszenie tych zasad możemy osiągnąć poprzez definiowanie ograniczenia typu FOREIGN KEY, w tym celu w oknie New Query piszemy następujące polecenie:

```
ALTER TABLE OSOBY
ADD CONSTRAINT Ogr5 FOREIGN KEY(IdMiasta)
REFERENCES Miasta(IdMiasta)
```

Wykonujemy polecenie wciskając klawisz F5.

Po wykonaniu polecenia opisane wyżej reguły będą wymuszane przy każdej operacji modyfikacji danych w tabeli Osoby lub Miasta. Sprawdzenie działania zdefiniowanych reguł dokonamy w tracie wprowadzania danych do tabel.

3.3.5. Ćwiczenie 7 – Definiowanie kolumn obliczanych.

W tabelach można definiować dodatkowe kolumny wirtualne. Istotą definicji kolumny wyliczanej jest zawarcie w definicji kolumny wyrażenia, które oblicza wartość tej kolumny dla każdego wiersza. Zawartość tych kolumn nie jest przechowywana w bazie danych, lecz obliczana w trakcie odczytu wierszy. W ramach ćwiczenia dodamy do tabeli Osoby dwie kolumny obliczane:

- kolumna o nazwie Wiek, w której zawarty będzie wiek danej osoby wyrażony w latach,
- kolumna Płeć, w której będą wartości Kobieta lub Mężczyzna w zależności od wartość zapisanej w kolumnie CzyKobieta.

Załóżmy, że w tabeli Osoby znajduje się kilka przykładowych wierszy tak jak pokazano na rysunku 30.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	CzyKobieta	IdMiasta	Znacznik
1	Nowak	Jan	1991-03-22	91032276537	0	1	0x0000000000001772
2	Lis	Beata	1990-09-11	90091175686	1	2	0x0000000000001775
3	Kowal	ryszard	1978-12-23	78122345676	0	1	0x0000000000001776

Rysunek 30.
Przykładowa zawartość tabeli Osoby

Wykonamy dwa polecenia języka SQL dodające do tabeli Osoby kolumny obliczane. W tym celu w oknie edycyjnym piszemy następujące polecenia:

```
ALTER TABLE Osoby
ADD Wiek AS
YEAR( GetDate()) – YEAR(DataUrodzenia)
GO
ALTER TABLE Osoby
ADD Płeć AS
CASE CzyKobieta
    WHEN 1 THEN ‘Kobieta’
    ELSE ‘Mężczyzna’
END
```

Wykonujemy napisane wcześniej polecenia (poprzez wciśnięcie klawisza F5). Wykonując teraz polecenie odczytujące zawartość tabeli Osoby (SELECT * FROM Osoby), otrzymamy postać tabeli pokazana na rysunku 31.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	CzyKobieta	IdMiasta	Znacznik	Wiek	Płeć
1	Nowak	Jan	1991-03-22	91032276537	0	1	0x0000000000001772	21	Mężczyzna
2	Lis	Beata	1990-09-11	90091175686	1	2	0x0000000000001775	22	Kobieta
3	Kowal	ryszard	1978-12-23	78122345676	0	1	0x0000000000001776	34	Mężczyzna

Rysunek 31.
Postać tabeli Osoby po zdefiniowaniu kolumn obliczanych.

Kolumny obliczane są dostępne w zapytaniach tak jak każda z kolumn rzeczywistych, są natomiast niedostępne dla operacji modyfikacji danych, gdyż ich wartość jest wynikiem dowiązanego wyrażenia.

Dodatkowo zademonstrujemy jeszcze specyfikę kolumny o nazwie Znacznik, która ma typ danych Timestamp. Jak widać na rysunku 31, zawartością kolumny Znacznik są wartości binarne. Na rysunku 32 pokazana została zawartość tabeli Osoby, w kolumnie o nazwie Czas widzimy zawartość kolumny Znacznik skonwertowaną na typ datetime.

Jak widać na rysunku 32, zawartość kolumny znacznik można przekształcić do postaci opisującej moment czasu, chociaż ten czas nie jest związany z rzeczywistością (daty z roku 1900). Istotą typu timestamp jest to,

że wartości w tej kolumnie są automatycznie zmieniane przy każdej modyfikacji wiersza. Na rysunku 33 pokazana została postać tej samej tabeli, w której wykonano modyfikację danych w wierszu o wartości IdOdoby=1 (zmieniono imię z Jan na Piotr).

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	CzyKobieta	IdMiasta	Znacznik	Wiek	Płeć	Nowa
1	Nowak	Jan	1991-03-22	91032276537	0	1	0x0000000000001772	21	Mężczyzna	1900-01-01 00:00:20.007
2	Lis	Beata	1990-09-11	90091175686	1	2	0x0000000000001775	22	Kobieta	1900-01-01 00:00:20.017
3	Kowal	Ryszard	1978-12-23	78122345676	0	1	0x0000000000001777	34	Mężczyzna	1900-01-01 00:00:20.023

Rysunek 32.
Wynik zapytania tworzącego kolumnę o nazwie Nowa

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	CzyKobieta	IdMiasta	Znacznik	Wiek	Płeć	Nowa
1	Nowak	Piotr	1991-03-22	91032276537	0	1	0x0000000000001778	21	Mężczyzna	1900-01-01 00:00:20.027
2	Lis	Beata	1990-09-11	90091175686	1	2	0x0000000000001775	22	Kobieta	1900-01-01 00:00:20.017
3	Kowal	Ryszard	1978-12-23	78122345676	0	1	0x0000000000001777	34	Mężczyzna	1900-01-01 00:00:20.023

Rysunek 33.
Zawartość tabeli Osoby po wykonanej modyfikacji

Proszę zwrócić uwagę na zmienioną zawartość kolumny Znacznik i tym samym kolumny Nowa.

Istotę zastosowania kolumn typu timestamp można opisać za pomocą następującego scenariusza:

- odczytujemy dane wybrane wiersza (do aplikacji interfejsu) w celu jego modyfikacji,
- wykonujemy modyfikacje danych w odczytanym wierszu,
- przed zapisaniem zmian w bazie danych sprawdzamy, czy wartość kolumny Znacznik nie została zmieniona,
- jeżeli wartość kolumny Znacznik nie uległa zmianie – modyfikacja danych w bazie może zostać przeprowadzona,
- jeżeli wartość kolumny Znacznik uległa zmianie – to znaczy, że w czasie trwania naszej modyfikacji inny użytkownik zmienił dane w tym wierszy. Modyfikacja powinna zostać odrzucona.

Szczegóły realizowanych zadań i ewentualne problemy z wykonanie ćwiczeń zostaną omówione z prowadzącym kurs.

4 JĘZYK T-SQL.

4.1. PODSTAWY JĘZYKA T-SQL.

Język SQL nie jest klasycznym językiem programowania, ponieważ nie zawiera konstrukcji sterujących umożliwiających sterowanie przebiegiem programu. W ujęciu podstawowym w języku SQL można jedynie pisać sekwencje poleceń. Dostępny w technologii SQL Server język Transact-SQL (skr. T-SQL) jest zmodyfikowaną i uzupełnioną o elementy typowe dla proceduralnych języków programowania jak zmienne i instrukcje sterujące wykonaniem programu. Za pomocą języka T-SQL możemy :

- wyszukiwać dane w bazie danych,
- manipulować danymi – wstawiać, modyfikować i usuwać dane z baz danych.
- definiować strukturę danych – dodawać, modyfikować oraz usuwać tabele, widoki oraz inne obiekty bazy danych,
- zarządzać dostępem do danych – tworzyć i usuwać konta użytkowników oraz nadawać uprawnienia do wykonania określonych operacji
- automatyzować wykonywanie typowych zadań administracyjnych – tworzyć kopie zapasowe baz, importować i eksportować dane czy porządkować indeksy,
- programować procedury i funkcje składowane,
- programować wyzwalacze.

Nowością w środowisku MS SQL Server 2008 R2 jest możliwość programowania po stronie serwera w środowisku .Net. Wprowadza to nowe możliwości do zagadnień programowania baz danych.

4.1.1. Deklaracja i inicjacja zmiennych

Język T-SQL wprowadza możliwość deklarowania i wykorzystywania zmiennych, co jest niedostępne w klasycznym języku SQL. Deklarowanie zmiennych realizowane jest za pomocą polecenia DECLARE. Deklaracja zmiennej zawiera:

- nazwę zmiennej (nazwa zmiennej musi rozpoczynać się od znaku @),
- nazwę typu danych (możemy wykorzystywać wszystkie typy danych dostępne w SQL Server),
- opcjonalną inicjację wartości zmiennej.

Przykład deklaracji zmiennych bez inicjowania ich wartości:

```
DECLARE @Liczba int,  @Dzien date.
Przykład deklaracji zmiennych z inicjowaniem wartości:
DECLARE @Dane XML='‹Dane›
                        ‹Liczba›5‹/Liczba›
                        ‹Liczba›12‹/Liczba›
                    ‹Dane› ,
        @@Parametr numeric(6,3)=165.234'
```

Uwaga: Zmienna, której nazwa rozpoczyna się od znaku @ jest zmienną lokalną (dla danego połączenia z SQL Server), a zmienna, której nazwa rozpoczyna się od dwóch znaków @ jest zmienną globalną.

4.1.2. Instrukcje podstawiania.

Przypisanie zmiennej wartości w języku T-SQL możemy realizować za pomocą dwóch poleceń:

- SET – przypisanie wartości zmiennej bez wysyłania wyniku do klienta,
- SELECT – przypisanie wartości zmiennej z wysłanie wyniku do klienta.

Przykład zastosowania instrukcji podstawiania SET;

```
DECLARE @NumerPesel char(11) , @Liczba int;
SET @NumerPesel=(SELECT Pesel FROM Osoby WHERE IdOsoby=1);
SET @Liczba=RAND()*200
```

Przykład zastosowania instrukcji podstawiania SELECT:

```
DECLARE @Dzien date;
SELECT @Dzien='1976-09-12';
SELECT @Dzien=DataUrodzenia
FROM Osoby
WHERE IdOsoby=23;
```

4.1.3. Iteracje w języku T-SQL.

W języku T-SQL dostępna jest jedna konstrukcja pętli – instrukcja WHILE. Podstawowa składnia instrukcji WHILE:

```
WHILE WyrazenieLogiczne
BEGIN
 {Instrukcje T-SQL}
END
```

Zawarte pomiędzy BEGIN a END instrukcje T-SQL będą wykonywane, gdy wyrażenie logiczne będzie prawdziwe (zwraca wartość TRUE).

W bloku instrukcji może być użyte jedno z dwóch poleceń sterujących przebiegiem pętli:

- CONTINUE – przerywa wykonywanie bloku instrukcji i przechodzi do sprawdzania wyrażenia logicznego (gdy wyrażenie jest prawdziwe wykonywana jest kolejna iteracja),
- BREAK – bezwarunkowo przerywa działanie pętli.

Przykład zastosowania pętli WHILE:

```
DECLARE @Licznik int=0 , @Suma bigint=0
WHILE @Licznik‹100
BEGIN
   SET @Suma=@Suma+@Licznik
   SET @Licznik=@Licznik+1
END
```

4.1.4. Instrukcja warunkowa.

Język T-SQL udostępnia instrukcję warunkową. Podstawowa składnia instrukcji warunkowej:

```
IF WyrazenieLogiczne
BEGIN
{Instrukcje T-SQL – wykonywane gdy wyrażenie logiczne jest prawdziwe}
END
      ELSE
       BEGIN
{Instrukcje T-SQL – wykonywane gdy wyrażenie logiczne jest nieprawdziwe}
       END
```

Przykład zastosowanie instrukcji warunkowej:

```
DECLARE  @Status varchar(50), @IdOsoby int
IF (SELECT YEAR(GetDate()) – YEAR(DAtaUrodzenia)
   FROM Osoby
   WHERE IdOsoby=@IdOsoby )›18
BEGIN
  SET @Status='Osoba pełnoletnia'
END
     ELSE
      BEGIN
            SET @Status='Osoba niepełnoletnia'
      END
```

Uwaga: Można wykorzystać skróconą postać instrukcji warunkowej bez bloku instrukcji wykonywanego przy nieprawdziwym warunku logicznym.

4.1.5. Obsługa wyjątków.

Bardzo istotnym elementem każdego programu jest obsługa wyjątków, czyli sytuacji gdy pewne działania programu generują błąd. Język T-SQL udostępnia dwie formy obsługi wyjątków:

- testowanie po każdym poleceniu zmiennej systemowe @@ERROR i podejmowanie odpowiednich działań w zależności od wartości tej zmiennej,
- podział bloku instrukcji na blok podstawowy (ograniczony przez słowa kluczowe BEGI TRY … END TRY) i blok obsługi wyjątków (ograniczony przez słowa kluczowe BEGIN CATCH … END CATCH).

Przykład obsługi wyjątków z wykorzystaniem zmiennej @@ERROR:

```
DECLARE @Komunikat varchar(200)
UPDATE OSOBY SET
DataUrodzenia='1997-09-12'
WHERE IdOsoby=5
IF @@ERROR =0
BEGIN
        SET @Komunikat='Modyfikacja wykonana poprawnie'
END
        ELSE
            BEGIN
                SET @Komunikat='Błąd modyfikacji'
            END
```

Przykład obsługi wyjątków z wykorzystanie bloków TRY I CATCH:

```
DECLARE @Komunikat varchar(50)
BEGIN TRY
    UPDATE OSOBY SET
DataUrodzenia='1997-09-12'
   WHERE IdOsoby=5
SET @Komunikat='Modyfikacja wykonana poprawnie'
END TRY
BEGIN CATCH
        SET @Komunikat='Błąd modyfikacji - ' +ERROR_MESSAGE()
END CATCH
```

Uwaga: Poprawnie napisany program powinien obsługiwać wszystkie możliwe wyjątki.

4.2. ĆWICZENIA

4.2.1. Ćwiczenie 8 – Pisanie skryptów.

W ramach ćwiczenia należy napisać skrypt realizujący następujące zadania:

- Zadeklarować dwie zmienne – jedna o nazwie @IleOsob typu int, druga o nazwie @Komunikat typu varchar(200),
- Do zmiennej @IleOsob przypisać(poprzez odpowiednie zapytanie SELECT) ilość osób zapisanych w tabeli Osoby,
- Wstawić do tabeli Miasta nowy wiersz (miasto o nazwie Kłodawa),
- Po każdym poleceniu sprawdzić wartość zmiennej @@ERROR i przypisać do zmiennej @Komunikat odpowiednią wartość.

Wykonać skrypt i sprawdzić jego działanie.

Problemy, które powstaną podczas pisania skryptu omówić z prowadzącym kurs.

4.2.2. Ćwiczenie 9 – Skrypt wykorzystujący pętle.

W ramach ćwiczenia należy napisać skrypt realizujący następujące zadania:

- Utworzyć tabelę o nazwie Sekwencja składającą się z jednej kolumny o nazwie Liczba (należy wykonać odpowiednie polecenie CREATE TABLE),
- Wstawić do tabeli 100 wierszy tak, żeby w kolumnie o nawie Liczba były kolejne liczby całkowite z zakresu 50 do 150,
- Obsługę wyjątków zaprogramować z wykorzystaniem bloków TRY CATCH.

Wykonać skrypt i sprawdzić jego działanie.

Problemy, które powstaną podczas pisania skryptu omówić z prowadzącym kurs.

5 PROGRAMOWANIE PROCEDUR, FUNKCJI I WYZWALACZY

5.1. PROCEDURY SKŁADOWANE

Procedurą składowaną nazywamy specjalną strukturę w bazie danych, która przechowuje program napisany w języku T-SQL. Każda procedura może zawierać parametry i zwracać określone wartości, może również wykonywać pewne operacje. Procedury składowane są odpowiednikiem metod w programowaniu obiektowym. Stosowanie procedur składowanych pozwala na zmniejszenie liczby kroków wymiany danych pomiędzy klientem a systemem zarządzania bazą danych, co może przyczynić się do wzrostu wydajności systemu. Zastosowanie procedur składowanych pozwala również wprowadzić bardziej przejrzysty interfejs pomiędzy bazą danych a aplikacjami z niej korzystającymi.

Zastosowanie procedur składowanych:

- uporządkowanie i centralizacja operacji na bazie danych,
- wprowadzanie reguł bezpieczeństwa (klient ma prawo wykonać procedurę, a nie wykonać dowolne zapytanie),
- zmniejszenie liczby interakcji z bazą danych.

Tworzenie procedur składowanych omówimy na przykładzie procedury realizującej zapisanie nowego wiersza w tabeli Osoby (utworzonej w ramach ćwiczenia 4) według następujących założeń:

- Procedura ma przyjmować następujące parametry:
  - Nazwisko
  - Imię
  - Datę urodzenia
  - Numer Pesel
  - Płeć (1 – kobieta, 2 – mężczyzna)
  - Nazwę miasta
- Procedura powinna mieć parametr wyjściowy, w którym będzie przekazywana informacja o realizacji,
- Należy sprawdzić, czy nazwa miasta, przekazana jako parametr, jest już zapisana w tabeli Miasta (utworzona w ramach ćwiczenia 4) – jeżeli nie jest zapisana, to należy dopisać nowy wiersz do tabeli Miasta,
- Zapisać nowy wiersz do tabeli Osoby na podstawie danych przekazanych jako parametry,
- W przypadku wystąpienia błędu należy przekazać w parametrze wyjściowy odpowiedni komunikat.

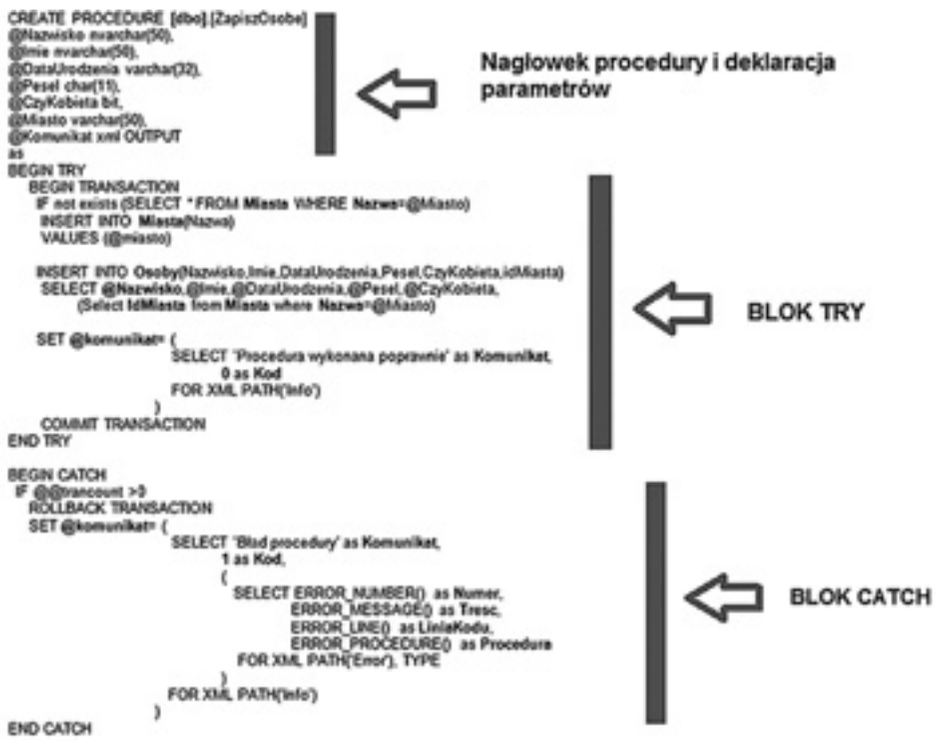
Na rysunku 34 pokazano postać polecenia CREATE PROCEDURE, które definiuje w bazie danych procedurę składowaną realizującą opisany schemat działania.

Uruchomienie procedury realizowane jest za pomocą polecenia EXECUTE. Przykładowe uruchomienie procedury, której kod źródłowy pokazano na rysunku 34, może mieć następującą postać:

```
DECLARE @Odpowiedz XML

EXEC  ZapiszOsobe
            @Nazwisko = 'Nowakowski',
            @Imie = 'Zbigniew',
            @DataUrodzenia = '1993-09-12',
            @Pesel = '93091298732',
            @CzyKobieta = 0,
            @Miasto = 'Nowy Tomyśl',
            @Komunikat = @Odpowiedz OUTPUT

SELECT   @Odpowiedz
```




```
CREATE PROCEDURE [dbo].[ZapiszOsobe]
@Nazwisko nvarchar(50),
@Imie nvarchar(50),
@DataUrodzenia varchar(32),
@Pesel char(11),
@CzyKobieta bit,
@Miasto varchar(50),
@Komunikat xml OUTPUT
as
BEGIN TRY
   BEGIN TRANSACTION
    IF not exists (SELECT * FROM Miasta WHERE Nazwa=@Miasto)
     INSERT INTO Miasta(Nazwa)
     VALUES (@miasto)

    INSERT INTO Osoby(Nazwisko,Imie,DataUrodzenia,Pesel,CzyKobieta,idMiasta)
     SELECT @Nazwisko,@Imie,@DataUrodzenia,@Pesel,@CzyKobieta,
          (Select IdMiasta from Miasta where Nazwa=@Miasto)

    SET @komunikat= (
                    SELECT 'Procedura wykonana poprawnie' as Komunikat,
                           0 as Kod
                    FOR XML PATH('Info')
                   )
    COMMIT TRANSACTION
END TRY

BEGIN CATCH
 IF @@trancount >0
   ROLLBACK TRANSACTION
   SET @komunikat= (
                    SELECT 'Blad procedury' as Komunikat,
                           1 as Kod,
                           (
                             SELECT ERROR_NUMBER() as Numer,
                                    ERROR_MESSAGE() as Tresc,
                                    ERROR_LINE() as LiniaKodu,
                                    ERROR_PROCEDURE() as Procedura
                             FOR XML PATH('Error'), TYPE
                           )
                    FOR XML PATH('Info')
                   )
END CATCH
```

Rysunek 34.
Kod źródłowy procedury ZapiszOsobe

Zadeklarowana zmienna, o nazwie @Odpowiedz, została przypisana w poleceniu uruchamiającym procedurę do parametru o nazwie @komunikat (parametr zadeklarowany w procedurze jak OUTPUT). Po wykonaniu procedury odczutujemy wartość tej zmiennej (SELECT @Odpowiedz). W przypadku poprawnego wykonania procedury, wartość tego parametru będzie miała postać:

```
‹Info›
 ‹Komunikat›Procedura wykonana poprawnie‹/Komunikat›
 ‹Kod›0‹/Kod›
‹/Info›
```

Jeżeli w trakcie wykonywania procedury wystąpi błąd, komunikat będzie zawierał informacje o szczegółach tego błędu. W celu wygenerowania sytuacji błędnej, wykonamy następujące polecenie:

```
DECLARE @Odpowiedz XML

EXEC  ZapiszOsobe
            @Nazwisko = 'Nowak',
            @Imie = 'Halina',
            @DataUrodzenia = '1993-09-12',
            @Pesel = '93091244732',
            @CzyKobieta = 1,
            @Miasto = 'Kluczbork,
            @Komunikat = @Odpowiedz OUTPUT

SELECT   @Odpowiedz
```

Dane przekazane procedurze zawierają merytoryczny błąd (płeć niezgodna z numerem Pesel) i w tej sytuacji komunikat wyjściowy z procedury będzie miał postać:

```
‹Info›
 ‹Komunikat›Blad procedury‹/Komunikat›
 ‹Kod›1‹/Kod›
 ‹Error›
  ‹Numer›547‹/Numer›
  ‹Tresc›The INSERT statement conflicted with the CHECK constraint „Ogr3”.
  The conflict occurred in database „KURSAndrzejPtasznik”, table „dbo.Osoby”.‹/Tresc›
  ‹LiniaKodu›16‹/LiniaKodu›
  ‹Procedura›ZapiszOsobe‹/Procedura›
 ‹/Error›
‹/Info›
```

Uwaga: Oprogramowanie interfejsu użytkownika powinno korzystać z bazy danych jedynie poprzez procedury składowane.

5.2. FUNKCJE TABELARYCZNE.

W ramach technologii MS SQL Server 2008 R2 może definiować funkcje tabelaryczne, które tworzą w bazie danych tabele wirtualne. Istotę funkcji tabelarycznych omówimy na przykładzie. Załóżmy, że chcemy utworzyć tabelę wirtualną o nazwie OsobyZMiasta, która zawiera listę osób z pewnego miasta. Funkcję tabelaryczną zdefiniujemy w bazie danych utworzonej w ramach ćwiczenia. Postać polecenia definiującego taka tabelę będzie miało następującą postać:

```
CREATE FUNCTION OsobyZMiasta(@Idmiasta int)
RETURNS TABLE
AS
RETURN
    (
      SELECT Nazwisko, Imie, Pesel, Płeć,Wiek
      FROM Osoby
      WHERE  IdMiasta=@IdMiasta

    )
```

W nagłówku polecenia CREATE FUNCTION zadeklarowany został parametr o nazwie @idmiasta, a w dowiązanym do polecenia zapytaniu, w klauzuli WHERE, jest odwołanie do tego parametru.

Po wykonaniu tego polecenia w bazie danych została utworzona wirtualna tabela o nazwie OsobyZMiasta, do której możemy się odwoływać tak jak do innych tabel (przekazując zdefiniowane parametry).

Załóżmy, że chcemy wybrać z bazy danych dane mężczyzn z miasta o wartości idmiasta=1. Zapytanie skierowanego do wirtualnej tabeli OsobyZMiasta będzie miało następującą postać:

```
SELECT *
FROM OsobyZMiasta(1)
WHERE Płeć='Mężczyzna'
```

Wynik tego zapytania, dla przykładowej bazy danych, pokazano na rysunku 35.

Nazwisko	Imie	Pesel	Płeć	Wiek
Nowak	Piotr	91032276537	Mężczyzna	21
Kowal	Ryszard	78122345676	Mężczyzna	34

Rysunek 35.
Wynik zapytania z wykorzystaniem odwołania do funkcji tabelarycznej (@IdMiasta=1)

Definiowanie funkcji tabelarycznych może uprościć tworzenie zapytać do bazy danych.

5.3. FUNKCJE SKALARNE.

Funkcje użytkownika, które możemy tworzyć w bazie danych, podobnie jak procedury są fragmentami programów w języku T-SQL. Istotą funkcji jest to, że zwracają wartość określonego typu i dlatego możemy funkcji używać w zapytaniach wszędzie tam, gdzie ma sens wartość zwracana przez funkcję. Jako przykład funkcji skalarnej pokażemy funkcję, która według algorytmu dla numeru Pesel liczy sumę kontrolną. Wynik algorytmu wykonanego na dziesięciu pierwszych cyfrach numeru Pesel powinien w wyniku zwrócic cyfrę jedenastą. Poniżej kod funkcji obliczającej sumę kontrolną:

```
CREATE FUNCTION SumaKontrolnaPesel(@dane varchar(10))
returns char(1)
Begin
  declare @suma char(1),
        @liczba int

  set @liczba= 1*SUBSTRING(@dane,1,1)+
           3*SUBSTRING(@dane,2,1)+
           7*SUBSTRING(@dane,3,1)+
           9*SUBSTRING(@dane,4,1)+
           1*SUBSTRING(@dane,5,1)+
           3*SUBSTRING(@dane,6,1)+
           7*SUBSTRING(@dane,7,1)+
           9*SUBSTRING(@dane,8,1)+
           1*SUBSTRING(@dane,9,1)+
           3*SUBSTRING(@dane,10,1)
   set @suma=
           case
             when  (@liczba % 10)   >0 then   10-(@liczba % 10)
             else 0
           end
  return @suma
End
```

Zaprezentowane poniżej zapytanie sprawdza poprawnośc numerów Pesel zapisanych w tabeli Osoby (utworzona w ramach ćwiczenia 4):

```
SELECT  Pesel,
          CASE
             WHEN dbo.SumaKontrolnaPesel(SUBSTRING(Pesel,1,10)) = SUBSTRING (Pesel,11,1)
               THEN 'Pesel poprawny'
             ELSE 'Pesel niepoprawny'
          END as Status
FROM Osoby
```

Wynik tego zapytania dla przykładowej bazy danych (utworzona w ramach ćwiczenia 3) pokazano na rysunku 36.

Pesel	Status
91032276537	Pesel niepoprawny
90091175686	Pesel niepoprawny
78122345676	Pesel niepoprawny
93091298732	Pesel niepoprawny

Rysunek 36.
Przykładowy wynik zapytania wykorzystującego funkcję skalarną

Uwaga: W konkretnej bazie danych funkcje użytkownika tworzymy w zależności od występujących potrzeb. Trudno, jednoznacznie określić, w jakich przypadkach i dla jakich potrzeb należy definiować funkcje użytkownika.

5.4. WYZWALACZE DML

Kolejnym ciekawym mechanizmem stosowanym w bazach danych są wyzwalacze. Wyzwalacze można porównać do procedur obsługi zdarzeń w programowaniu obiektowym. W ramach wykładu zapoznamy się z poszczególnymi rodzajami wyzwalaczy oraz z typowymi scenariuszami korzystania z nich.

Wyzwalacze działające dla zdarzeń związanych z poleceniami języka DML można podzielić na:

- wyzwalacze AFTER
  - kod w nich zawarty jest wykonywany po operacji uruchamiającej wyzwalacz, ale co istotne – w ramach tej samej transakcji
- wyzwalacze INSTEAD OF
  - ich kod jest wykonywany zamiast operacji uruchamiającej wyzwalacz. Właściwa operacja nie dochodzi do skutku.

Dla jednego zdarzenia na jednej tabeli można tworzyć wiele wyzwalaczy. W takiej sytuacji ważne jest, aby pamiętać, że kolejność ich wykonania jest nieokreślona. Jedyne co możemy w tej sprawie zrobić, to określić, który z nich ma się wykonać jako pierwszy, a który jako ostatni. Realizuje się to za pomocą procedury składowanej sp_settriggerorder.

SQL Server 2008 R2 udostępnia dla kodu wyzwalaczy dwie specjalne tabele. Są to tabele **inserted** i **deleted**. Są one w pełni obsługiwane przez serwer i utrzymywane w pamięci, więc dostęp do nich jest szybki. Gdy rozpoczyna się wykonanie kodu wyzwalacza, tabele te są napełniane odpowiednimi danymi. W przypadku wyzwalacza skojarzonego z poleceniem INSERT – tabela inserted będzie zawierała dodawane wiersze. W przypadku wyzwalacza skojarzonego z poleceniem DELETE – tabela deleted będzie zawierała usuwane dane. W przypadku wyzwalacza skojarzonego z poleceniem UPDATE – tabela deleted będzie zawierała oryginalne wartości modyfikowanych danych, a tabela inserted ich nowe wartości.

Istotną cechą wyzwalaczy jest fakt, że mogą one być uruchomione nie tylko dla operacji modyfikującej pojedynczy wiersz. Wyzwalacz jest uruchamiany w odpowiedzi na wykonanie jednego polecenia, ale to polecenie może na przykład dodać lub usunąć kilkaset wierszy jednocześnie. Dlatego właśnie budując kod wyzwalacza powinniśmy uwzględniać, że może on wykonywać się dla wielu wierszy modyfikowanych jednym poleceniem uruchamiającym wyzwalacz.

Typowe zastosowanie wyzwalaczy to:

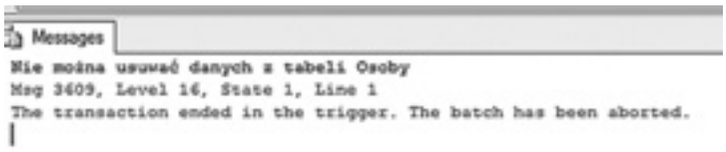
- implementacja złożonych reguł integralności,
- rejestrowanie działań użytkowników,
- zabezpieczenie przed niepożądanymi operacjami,
- wyliczanie wartości liczbowych na podstawie zmian w tabelach.

W każdym przypadku zastosowania wyzwalaczy jest to decyzja wynikająca z chęci automatyzacji pewnych działań w zależności od wystąpienia odpowiedniego zdarzenia.

Zdefiniujemy wyzwalacz, którego zadaniem będzie uniemożliwienie usuwania danych z tabeli Osoby. Kod polecenia definiującego taki wyzwalacz będzie miał postać:

```
CREATE TRIGGER Blokada
ON Osoby
AFTER DELETE
AS
ROLLBACK
PRINT ‘Nie można usuwać danych z tabeli Osoby’
```

Po wykonaniu tego polecenia każda próba usunięcia danych z tabeli Osoby będzie skutkowało komunikatem systemu pokazanym na rysunku 37:



Rysunek 37.
Komunikat systemu przy próbie usunięcia danych z tabeli Osoby

5.5. WYZWALACZE DDL.

Osobną grupą wyzwalaczy są wyzwalacze DDL. Jak sama nazwa wskazuje reagują one na zdarzenia związane z poleceniami języka DDL (Data Definition Language). Takimi poleceniami są: CREATE, ALTER, DROP, GRANT, DENY, REVOKE, UPDATE STATISTICS. Przeznaczenie wyzwalaczy DDL jest nieco inne niż wcześniej omawianych wyzwalaczy DML. Podstawową rolą wyzwalaczy DDL jest wspomaganie mechanizmów audytu i śledzenia zmian w strukturze bazy danych oraz ograniczanie możliwości manipulowania tą strukturą. W kodzie wyzwalacza DDL dostępna jest specjalna funkcja EVENTDATA(), która zwraca szczegółowe informacje o zdarzeniu w formie XML Wyzwalacze DDL można tworzyć na poziomie bazy danych lub całego serwera. W dokumencie XML, zwracanym przez funkcję EVENTDATA(), zwarte są szczegółowe informacje o szczegółach polecenia, które uruchomiło działanie wyzwalacza. Przykładową postać wyniku funkcji EVENTDATA() pokazano na rysunku 38.

```
<EVENT_INSTANCE>
 <EventType>CREATE_VIEW</EventType>
 <PostTime>2012-04-25T07:52:38.177</PostTime>
 <SPID>57</SPID>
 <ServerName>APPROD\AP2008R2</ServerName>
 <LoginName>APProd\andrzej</LoginName>
 <UserName>dbo</UserName>
 <DatabaseName>KURSAndrzejPtasznik</DatabaseName>
 <SchemaName>dbo</SchemaName>
 <ObjectName>Test</ObjectName>
 <ObjectType>VIEW</ObjectType>
 <TSQLCommand>
  <SetOptions ANSI_NULLS="ON"
              ANSI_NULL_DEFAULT="ON"
              ANSI_PADDING="ON"
  <CommandText>create view Test  As Select * FROM Miasta</CommandText>
 </TSQLCommand>
</EVENT_INSTANCE>
```

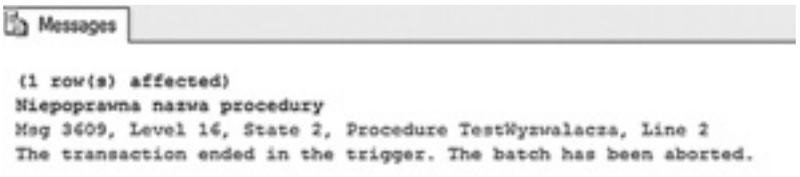
Rysunek 38.
Przykładowa postać dokumentu XML zwracanego przez funkcje EVENTDATA()

Załóżmy, że chcemy wymusić w naszej bazie danych zasadę, że każda procedura składowana definiowana przez użytkownika musi mieć nazwę rozpoczynającą się od prefixu ‘PS’. W celu realizacji takiego założenia utworzymy odpowiedni wyzwalacz:

```
CREATE TRIGGER  NazwaOK
ON DATABASE
AFTER CREATE_PROCEDURE
```

```
AS
IF (EVENTDATA()).value('EVENT_INSTANCE[1]/ObjectName[1]' , 'sysname')
 NOT LIKE 'PS%'
BEGIN
  ROLLBACK
 PRINT 'Niepoprawna nazwa procedury'
END
```

Po wykonaniu tego polecenia, po każdej próbie utworzenia procedury składowanej, której nazwa nie będzie rozpoczynała się od 'PS', zostanie wygenerowany komunikat, pokazany na rysunku 39 i procedura nie zostanie utworzona.



Rysunek 39.
Postać komunikatu

Uwaga: Wyzwalacze DDL mogą działać w kontekście konkretnej bazy danych (ON DATABASE) lub w kontekście całego serwera (ON ALL SERVER). Przy pomocy wyzwalaczy DDL można wymusić przestrzeganie reguł poprawności definiowania obiektów w bazie danych.

5.6. ĆWICZENIA

5.6.1. Ćwiczenie 10 – Procedura składowana.

W ramach ćwiczenia należy utworzyć w bazie danych procedurę składowaną o nazwie ModyfikacjaOsoby, która realizuje następujące zadania:

- Procedura ma przyjmować następujące parametry:
  - Nazwisko
  - Imię
  - Datę urodzenia
  - Numer Pesel
  - Płeć (1 – kobieta, 2 – mężczyzna)
  - Nazwę miasta
  - Identyfikator osoby
- Procedura powinna mieć parametr wyjściowy, w którym będzie przekazywana informacja o realizacji,
- Należy sprawdzić, czy nazwa miasta przekazana jako parametr jest już zapisana w tabeli Miasta (utworzona w ramach ćwiczenia 4) – jeżeli nie jest zapisana, to należy dopisać nowy wiersz do tabeli Miasta,
- Zmodyfikować wiersz w tabeli Osoby na podstawie danych przekazanych jako parametry,
- W przypadku wystąpienia błędu należy przekazać w parametrze wyjściowy odpowiedni komunikat.

Po utworzeniu procedury należy przetestować jej działanie.

Omówić z prowadzącym kurs problemy powstałe w trakcie definiowania procedury.

5.6.2. Ćwiczenie 11 – Funkcja tabelaryczna.

W ramach ćwiczenia należy zdefiniować w bazie danych funkcje tabelaryczną, która przyjmuje dwa parametry (nazwę miasta oraz liczbę całkowitą) i zwraca dane osób (nazwisko, imię, numer Pesel i płeć), które są do-

wiązane do miasta o podanej nazwie i których wiek (w latach) jest większy od podanej, jako parametr, liczby całkowitej. Funkcję definiujemy w bazie danych utworzonej w ramach ćwiczenia 3 i wykorzystujemy tabele Osoby i Miasta utworzone w ramach ćwiczenia 4.

Po utworzeniu funkcji należy przetestować jej działanie dla różnych wartości parametrów i omówić z prowadzącym kurs problemy powstałe przy definiowaniu i testowaniu funkcji.

5.6.3. Ćwiczenie 12 – Funkcja skalarna.

Należy zdefiniować w bazie danych funkcje skalarną, która przyjmuje jako parametr nazwę miasta i zwraca liczbę całkowitą określającą ilość osób dowiązanych do danego miasta. Po zdefiniowaniu funkcji, należy przetestować jej działanie wykorzystując ją w zapytaniach. Funkcję definiujemy w bazie danych utworzonej w ramach ćwiczenia 3 i wykorzystujemy tabele Osoby i Miasta utworzone w ramach ćwiczenia 4.

Po utworzeniu funkcji należy przetestować jej działanie dla różnych wartości parametrów i omówić z prowadzącym kurs problemy powstałe przy definiowaniu i testowaniu funkcji.

5.6.4. Ćwiczenie 13 – Wyzwalacz DML.

W ramach ćwiczenia, należy zdefiniować w bazie danych wyzwalacz, który będzie dowiązany do tabeli Osoby dla poleceń INSERT, UPDATE i DELETE. Wyzwalacz powinien rejestrować operacje modyfikacji wykonane w tabeli. Przed definiowaniem wyzwalacza należy w bazie danych utworzyć tabelę o nazwie LogZmian, której schemat pokazano na rysunku 40.

Column Name	Data Type	Allow Nulls
ID	int	☐
Kto	varchar(100)	☐
Kiedy	datetime	☐
Komputer	varchar(100)	☐
Operacja	char(1)	☐

Rysunek 40.
Schemat tabeli LogZmian

Po każdym poleceniu modyfikującym dane (INSERT, UPADTE i DELETE) wyzwalacz powinien zapisywać w tabeli LogZmian odpowiedni wiersz, w którym zawarte są następujące dane:

- login wykonującego polecenie (kolumna KTO),
- dokładny czas wykonania operacji (kolumna KIEDY),
- nazwę komputera. z którego przesłano polecenie do bazy danych (kolumna KOMPUTER),
- symbol operacji (I – insert, D- delete, U- update) – (kolumna OPERACJA).

Przykładową zawartość tabeli LogZmian pokazano na rysunku 41.

Results | Messages

ID	Kto	Kiedy	Komputer	Operacja
5	APProd\andrzej	2012-04-25 08:50:09.720	APPROD	I
6	APProd\andrzej	2012-04-25 08:50:15.727	APPROD	U
7	APProd\andrzej	2012-04-25 08:50:20.550	APPROD	D
8	APProd\andrzej	2012-04-25 08:50:26.760	APPROD	U

Rysunek 41.
Przykładowa zawartość tabeli LogZmian

Po zdefiniowaniu wyzwalacza należy przetestować jego działanie i sprawdzić zapisy powstające w tabeli LogZmian.

Omówić z prowadzącym kurs problemy powstałe przy definiowaniu i testowaniu wyzwalacza.

Uwaga: Potrzebne wartość dotyczące zalogowanej osoby oraz nazwę komputera, z którego przesłano polecenie można uzyskać za pomocą funkcji systemowych SYSTEM_USER i HOST_NAME.

5.6.5. Ćwiczenie 14 – Wyzwalacz DDL.

W ramach ćwiczenia należy zdefiniować w bazie danych wyzwalacz, który uniemożliwi usunięcie i modyfikacje struktury tabel.

Po utworzeniu wyzwalacza należy przetestować jej działanie dla poleceń DROP TABLE i ALTER TABLE oraz omówić z prowadzącym kurs problemy powstałe przy definiowaniu i testowaniu wyzwalacza.

6 PODSUMOWANIE.

Zagadnienia związane z programowaniem w języku T-SQL są bardzo istotnym elementem implementacji baz danych. W ramach naszego kursu pokazane zostały postawy programowanie i zrealizowane zostały proste ćwiczenia definiujące w bazie danych, procedury, funkcje i wyzwalacze. Wielu aspektach programowania w języku T-SQL, ze względu na ograniczony czas kursu, nie omawialiśmy. Widza zdobyta w trakcie ćwiczeń powinna umożliwić dalsze jej pogłębianie w ramach samodzielnych zadań.

7 LITERATURA.

1. Ben-Gan I., Kollar L., Sarka D., *MS SQL Server 2005 od środka: Zapytania w języku T-SQL,* APN PROMISE, Warszawa 2006
2. Coburn R., *SQL dla każdego*, Helion, Gliwice 2001
3. Rizzo T., Machanic A., Dewson R., Walters R., Sack J., Skin J., *SQL Server 2005*, WNT, Warszawa 2008
4. Szeliga M., ABC języka SQL, Helion, Gliwice 2002
5. Vieira R., *SQL Server 2005. Programowanie. Od Podstaw,* Helion, Gliwice 2007

W projekcie **Informatyka +**, poza wykładami i warsztatami, przewidziano następujące działania:

- 24-godzinne kursy dla uczniów w ramach modułów tematycznych
- 24-godzinne kursy metodyczne dla nauczycieli, przygotowujące do pracy z uczniem zdolnym
- nagrania 60 wykładów informatycznych, prowadzonych przez wybitnych specjalistów i nauczycieli akademickich
- konkursy dla uczniów, trzy w ciągu roku
- udział uczniów w pracach kół naukowych
- udział uczniów w konferencjach naukowych
- obozy wypoczynkowo-naukowe.

Szczegółowe informacje znajdują się na stronie projektu

www.informatykaplus.edu.pl