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:

Język 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.

Język SQL

Rodzaj zajęć: Wszechnica Informatyczna
Tytuł: Język 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.

Język SQL

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

Streszczenie

W ramach kursu „Język SQL" uczestnicy zapoznani zostaną z MS SQL Server 2008 R2. Poznają język SQL ze szczególnym uwzględnieniem poleceń DDL (ang. *Data Definition Language*) oraz poleceń DML (ang. *Data Manipulation Language*). W ramach poleceń DML szczegółowo zaprezentowane zostanie polecenie Select – realizujące zapytania do bazy danych.

Spis treści

1 WPROWADZENIE

1.1. WSTĘP

Twórcą teorii relacyjnych baz danych jest Edgar Frank Codd. Postulaty te zostały opublikowane po raz pierwszy w roku 1970 w pracy *A Relational Model of Data for Large Shared Data Banks*. Praca ta opisuje podstawowe zależności, jakie mogą występować pomiędzy danymi trwałymi, oraz wprowadza główne założenia dotyczące modelu relacyjnego dla danych wraz z propozycją formalnych operatorów przeszukiwania danych. Burzliwy rozwój systemów opartych na relacyjnym modelu danych rozpoczął się wraz z wypuszczeniem na rynek w roku 1979 przez firmę ORACLE pierwszego komercyjnego relacyjnego systemu zarządzania bazą danych (ang. *Relational Database Management Systems*, RDBMS). Sukces tego przedsięwzięcia zapoczątkował dominację baz danych bazujących na modelu relacyjnym.

1.2. SYSTEMY ZARZĄDZANIA RELACYJNYMI BAZAMI DANYCH

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.

2 TECHNOLOGIA MS SQL SERVER 2008

2.1. ELEMENTY TECHNOLOGII MS SQL SERVER 2008

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.

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.

2.3. ĆWICZENIA

2.3.1. Ćwiczenie 1 – 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:

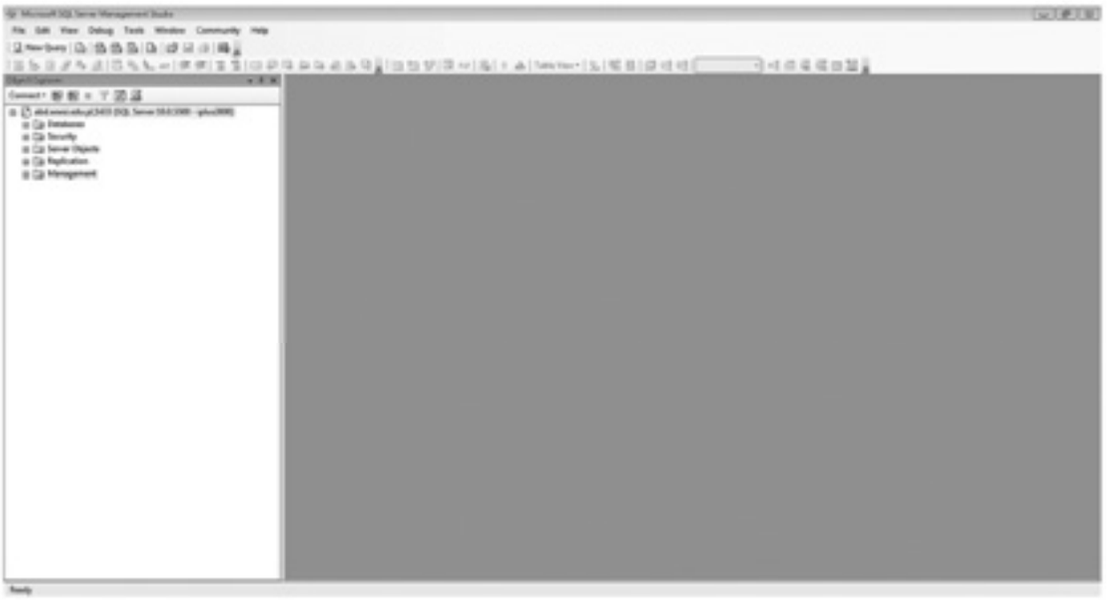
1. Uruchamiamy program SQL Server Management Studio 2008 R2
2. Po uruchomieniu program pojawi się okienko logowania

Należy podać nazwę serwera baz danych (pole Server name), wybrać tryb uwierzytelnienia (pole Autentication) oraz parametry logowania (pola Login i Password). Podane pola uzupełniamy tak jak pokazano na rysunku 1.

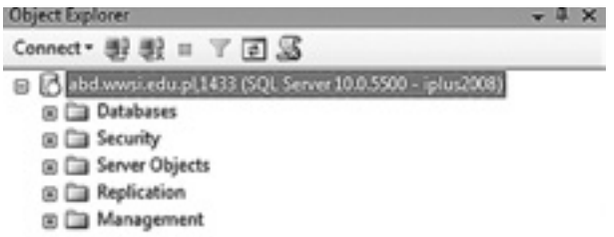
3. Po zalogowaniu uruchamia się główny panel programu MS SQL Server Management Studio

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 2. Najistotniejszym elementem tego interfejsu jest okno Object Explorer, którego ogólnąapostać pokazana została na rysunku 3.

Rysunek 1.
Okno logowania programu SQL Server Management Studio

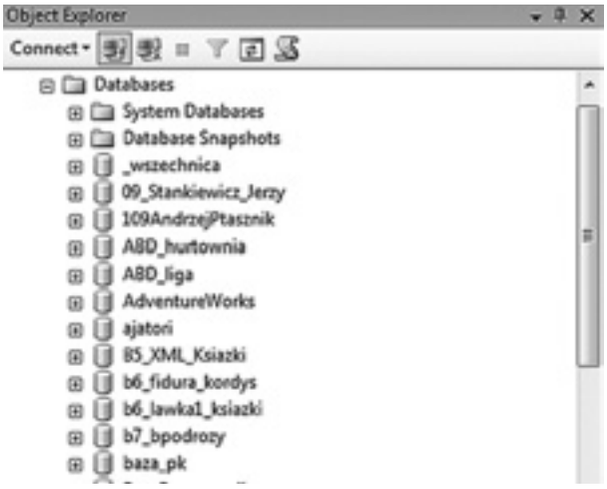


Rysunek 2.
Postać wyjściowa panelu MS SQL Server Management Studio 2008 R2



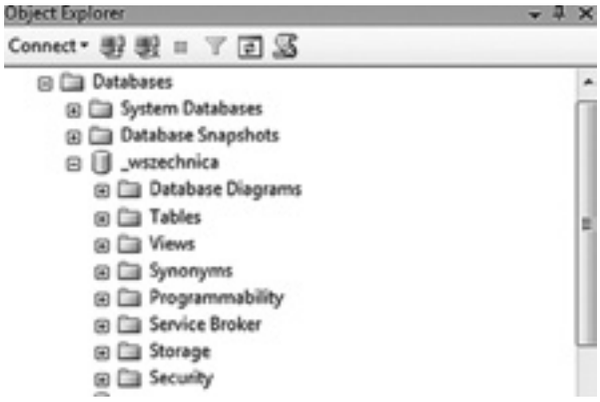
Rysunek 3.
Okno Object Explorer – kontekst ogólny

W zależności od wybranego kontekstu w oknie Object Explorer można uzyskać dostęp do różnych, hierarchicznie zdefiniowanych obiektów serwera. Kontekst pierwszy przedstawia podstawowe elementy serwera. W ramach naszego kursu istotne będzie jedynie dostęp do baz danych, które są zawarte w folderze Databases (rysunek 3). Rozwinięcie folderu Databases, jak pokazano na rysunku 4, wyświetla wszystkie bazy danych zdefiniowane na instancji serwera.



Rysunek 4.
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 5.



Rysunek 5.
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.

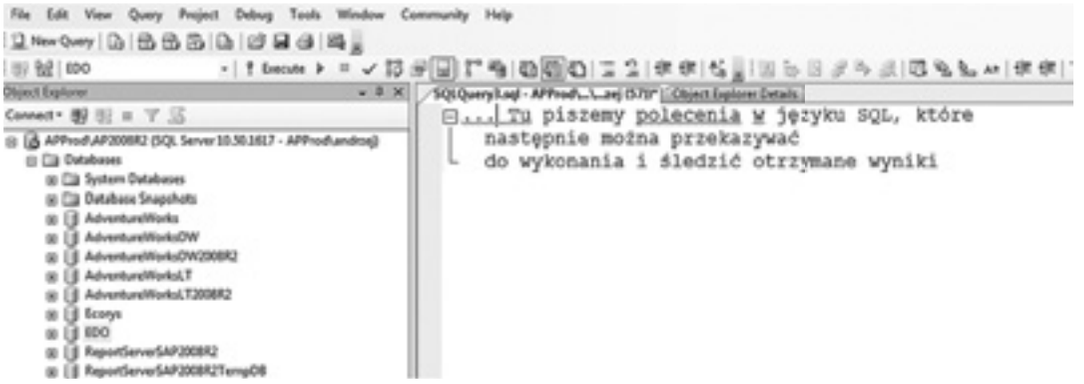
Na rysunku 6 pokazano kontekst Tables, w ramach którego widoczne są wszystkie tabele zdefiniowane w wybranej bazie danych.

Rysunek 6.
Okno Object Explorer – kontekst Tables

Dla wybranej tabeli, co pokazano na rysunku 7, 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 7.
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 8.



Rysunek 8.
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 WPROWADZENIE DO JĘZYKA SQL

3.1. HISTORIA JĘZYKA SQL

Język **SQL** (ang. *Structured Query Language*) został stworzony do pracy z relacyjną bazą danych. Jest to język nieproceduralny, należący do grupy języków deklaratywnych. Składnia języka SQL opisuje, co ma być zrobione, a nie jak należy to wykonać. Problem „jak wykonać" przeniesiony został na poziom systemu zarządzania bazą danych. Pierwowzór języka SQL – SEQUEL (ang. *Structured English Query Language*) – został zaprojektowany przez IBM w 1974 roku. Pierwsza zaś wersja SQL została komercyjnie zastosowana przez firmę Oracle Corporation w 1979 roku. Do dnia dzisiejszego język SQL jest ciągle rozwijany i znajduje zastosowanie w większości systemów opartych na relacyjnym modelu danych. W założeniach twórców miał to być język uniwersalny, możliwie prosty i maksymalnie zbliżony do języka naturalnego. Czy to założenie zostało zrealizowane – trudno jednoznacznie ocenić, ponieważ jest to język łatwy i prosty..., ale czasami może być trudny.

Język SQL można podzielić na kilka podstawowych części:

- Język definiowania danych (ang. *Data Definition Language*, DDL),
- Język manipulowania danymi (ang. *Data Manipulation Language*, DML),
- Język zapytań (ang. *Data Query Language*, DQL),
- Język kontroli dostępu do danych (ang. *Data Control Language*, DCL),
- Polecenia administracyjne,
- Polecenia obsługi transakcji.

Przedstawione wyżej pogrupowanie poleceń języka SQL ma znaczenie jedynie umowne i przy jego stosowaniu nie istnieją odmienne reguły i zasady do każdej części języka SQL. Do dnia dzisiejszego trwa rozwój języka SQL, wielu producentów Systemów Zarządzania Relacyjnymi Bazami Danych implementuje go w swoich systemach. Ponieważ różni producenci implementowali język SQL to skutkiem ubocznym takiej sytuacji były rozbieżności wersji tego języka u różnych producentów.

3.2. STANDARDY JĘZYKA SQL

Początkowy rozwój języka SQL skutkował powstaniem różnych jego dialektów. Pierwszą próbą uporządkowania i ujednolicenia dialektów języka SQL był standard ANSI (ang. *American National Standards Institute*) z roku 1986. Rok później ISO (ang. *International Organization for Standarization*) zaakceptowała ten standard, wydając własny dokument normalizacyjny. W roku 1989 opublikowano zweryfikowany standard ANSI znany powszechnie jako SQL1. Niestety, głównie ze względu na sprzeczne interesy producentów systemów bazodanowych, standard ten nie określał wielu podstawowych cech języka, a wiele właściwości zdefiniowano jako zależne od implementacji. Kolejną próbą ujednolicenia języka było przyjęcie (w roku 1992) standardu SQL2. Niezbędnym kompromisem okazało się wprowadzenie trzech poziomów zgodności z nowym standardem:

1. Podstawowy poziom zgodności (ang. *Entry-level conformance*) był właściwie powtórzeniem standardu SQL1. Ten poziom zgodności wdrożyli w swoich produktach prawie wszyscy producenci serwerów bazodanowych.
2. Pośredni poziom zgodności (ang. *Intermediate-level conformance*) stanowił ogólnie osiągalny zbiór zasadniczych ujednoliceń języka.
3. Pełny poziom zgodności (ang. *Full conformance*) – oferował pełną zgodność z właściwościami standardu SQL2.

W roku 1999 organizacje ANSI i ISO opracowały standard SQL3. Był to pierwszy standard obejmujący zaawansowane funkcje i obszary zastosowań języka SQL, takie jak modele obiektowo-relacyjnych baz danych, mechanizmy wywołania instrukcji języka SQL, czy techniki zarządzania spójnością danych.

Nadal jednak istnieją różnice w wersjach języka SQL, biorąc pod uwagę różnych producentów – można uznać, że dysponujemy językiem SQL w jego standardowej wersji i „gwarami" opartymi o SQL u różnych producentów. Z punktu widzenia nauki języka nie ma większego znaczenia, którą jego wersję poznamy jako pierwszą, gdyż opanowanie podstaw języka SQL umożliwia szybkie przystosowanie się do dowolnej jego mutacji. W tabeli 1 przedstawione zostały podstawowe standardy języka SQL.

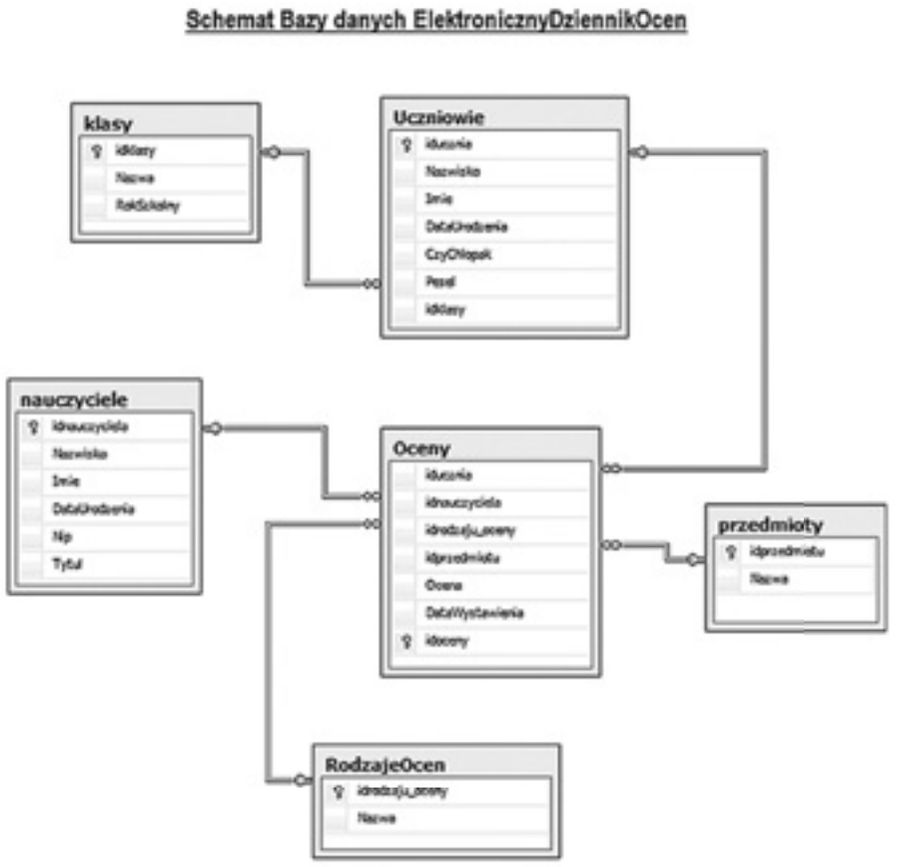
Rok wprowadzenia	Nazwa standardu	Opis standardu
1986	SQL-86/SQL-87	Pierwsza publikacja standardu organizacji ANSI i ISO
1989	SQL-89	Niewielkie (mało istotne) zmiany standardu SQL-86
1992	SQL-92/SQL2	Zasadnicze zmiany w stosunku do oryginału (najbardziej popularny standard do dziś)
1999	SQL-99/SQL3	Aktualizacja standardu z roku 1992 poprzez: nowe sposoby selekcji, nowe reguły integralności, pewne elementy struktury obiektowej
2003	SQL-2003	Obsługa formatu XML i pól z automatycznie generowanymi wartościami
2008	SQL-2008	Niewielka korekta standardu SQL-2003

Tabela 1. Standardy języka SQL

Standard języka to wytyczne dla producentów Systemów Zarządzania Bazami Danych, a nie obowiązkowe reguły i zasady. Pomimo istnienia standardów języka SQL – różne jego implementacje różnią się nieznacznie od siebie.

3.3. PRZYKŁADOWA BAZA DANYCH

W trakcie kursu będziemy definiować własne bazy danych, natomiast przy omawianiu języka zapytań, będziemy korzystać z przykładowej bazy danych o nazwie ElektronicznyDziennikOcen, której schemat został przedstawiony na rysunku 9.



Rysunek 9.
Schemat bazy danych ElektronicznyDziennikOcen

Baza ElektronicznyDziennikOcen umożliwia ewidencjowanie ocen wystawianych uczniom w ramach procesu dydaktycznego. Podstawową tabelą tej bazy danych jest tabela o nazwie „Oceny". W tabeli ***Oceny*** są przechowywane dane o pewnych zdarzeniach – wystawionych ocenach. Ta tabela jest centralnym punktem bazy danych. Zawartość tej tabeli można opisać następująco: Pewien uczeń (***iducznia***) od jakiegoś nauczyciela (***idnauczyciela***) otrzymał pewien rodzaj oceny (***idrodzaju_oceny***) z pewnego przedmiotu (***idprzedmiotu***) o wartości oceny (***ocena***) wystawionej pewnego dnia (***data***).

4 JĘZYK DEFINIOWANIA DANYCH SQL DDL (ANG. DATA DEFINITION LANGUAGE, DDL)

Część języka umożliwiająca definiowanie struktur bazy danych (tabele, widoki, procedury, indeksy i inne obiekty bazy danych) oraz ich modyfikację. W skład DDL wchodzą następujące polecenia: **CREATE** (zdefiniuj obiekt bazy danych), **DROP** (usuń obiekt z bazy danych) i **ALTER** (zmień definicję istniejącego obiektu).

4.1. TWORZENIE BAZY DANYCH I JEJ OBIEKTÓW – POLECENIE CREATE

Pierwszym krokiem, w procesie tworzenia bazy danych jest zdefiniowanie jej obiektów. Oczywistym się wydaje, że obiektami bazy danych są tabele, ale oprócz tabel w bazie danych możemy definiować:

- widoki,
- wyzwalacze,
- procedury składowane,
- funkcje,
- ograniczenia,
- indeksy,
- i wiele, wiele innych.

Polecenie CREATE języka SQL umożliwia definiowanie obiektów bazy danych. Dla każdego definiowanego obiektu bazy danych składnia polecenia CREATE jest praktycznie inna. Wynika to z faktu, że dla różnych obiektów bazy danych trzeba określić inne parametry. Definiowanie tabeli wymaga opisania jej budowy, czyli z jakich kolumn będzie się składała, dla kolumn należy określić ich typ oraz ewentualne ograniczenia. Tabela musi posiadać klucz podstawowy, czyli w poleceniu definiującym tabelę powinno się wskazać kolumnę (lub kolumny), która będzie kluczem podstawowym. Polecenie CREATE definiujące tabelę może mieć następującą postać:

```
CREATE TABLE [dbo].[Nauczyciele]
(
        [IdNauczyciela] [int] IDENTITY(1,1) NOT NULL,
        [Nazwisko] [varchar](50) NOT NULL,
        [Imie] [varchar](50) NOT NULL,
        [DataUrodzenia] [date] NULL,
        [Pesel] [char](13) NOT NULL,
        [Tytul] [varchar](32) NOT NULL,
 CONSTRAINT [PK_nauczyciele] PRIMARY KEY CLUSTERED
(
        [IdNauczyciela] ASC
)
) ON [PRIMARY]
```

Aby zrozumieć to polecenie, sformułujemy zdanie w języku potocznym opisujące czynności, jakie serwer baz danych ma wykonać w odpowiedzi na to polecenie.

Zdefiniować nową tabelę o nazwie Nauczyciele, która składa się z następujących kolumn:

- *IdNauczyciela – kolumna typu integer z automatycznym ustalaniem wartości i niedopuszczalną wartością null,*
- *Nazwisko – kolumna typu varchar(50) z niedopuszczalną wartością null,*
- *Imie – kolumna typu varchar(50) z niedopuszczalną wartością null,*
- *DataUrodzenia – kolumna typu date z niedopuszczalną wartością null,*
- *Pesel – kolumna typu varchar(11) z niedopuszczalną wartością null,*
- *Tytul – kolumna typu varchar(32).*

Dodatkowo definiujemy ograniczenie uznające kolumnę IdNauczyciela za klucz podstawowy tabeli. Porównując postać polecenia i jego interpretację wyrażoną w języku potocznym, można zauważyć, że polecenie języka SQL w precyzyjny sposób określiło, jak ma wyglądać tabela, którą chcemy zdefiniować, brak jest w nim natomiast jakichkolwiek wskazówek, jak to ma być wykonane.

Zupełnie inaczej będzie wyglądało polecenie CREATE definiujące widok (widok jest tabelą wirtualną powstającą jako wynik polecenia SELECT) z tego powodu, że definiując widok należy podać polecenie SELECT, które go tworzy;

```
CREATE VIEW WidokTestowy
AS
SELECT Naazwisko, Imie, Pesel, DataUrodzenia
FROM Uczniowie
WHERE IdKlasy=1
WITH CHECK OPTION
```

Umieszczona na końcu polecenia dyrektywa WITH CHECK OPTION zapewnia, że wprowadzając dane lub je modyfikując poprzez widok będzie przestrzegany warunek opisany w klauzuli WHERE. Działanie tej dyrektywy umożliwi wprowadzenie danych ucznia z klasy o IdKlasy=1, a także uniemożliwi modyfikację tej kolumny poprzez widok.

Z podanych dwóch przykładów poleceń CREATE widać, że składania ich zależy od rodzaju tworzonego obiektu (co wydaje się całkowicie logiczne).

4.2. MODYFIKACJA OBIEKTÓW BAZY DANYCH – POLECENIE ALTER

Polecenie ALTER umożliwia modyfikację istniejącego (wcześniej zdefiniowanego) obiektu bazy danych. Podobnie jak w przypadku polecenia CREATE, szczegółowa składnia zależy od rodzaju modyfikowanego obiektu. Jako pierwszy przykład podamy polecenie ALTER modyfikujące tabelę Nauczyciele, dodając do tej tabeli ograniczenie typu UNIQUE (wymusza unikalne wartości w danej kolumnie) dla kolumny Pesel:

```
ALTER TABLE Nauczyciele
ADD CONSTRAINT UniqPesel UNIQUE NONCLUSTERED (Pesel)
```

I w kolejnym przykładzie zmodyfikujemy definicję widoku o nazwie WidokTestowy, dodając do widoku nową kolumnę Tytul:

```
ALTER VIEW WidokTestowy
WITH SCHEMABINDIG
AS
SELECT Nazwisko, Imie, Pesel, DataUrodzenia,Tytul
FROM Uczniowie
WHERE IdKlasy=1
WITH CHECK OPTION
```

Dodatkowo zamieściliśmy w zmodyfikowanym widoku dyrektywę WITH SCHEMABINDING, która zapewnia poprawność definicji widoki, czyli nie można zmodyfikować tabeli Uczniowie w taki sposób, który spowodowałby błąd zapytania dowiązanego do widoku (np. usunięcie kolumny Tytul). W ramach kursu nie jesteśmy w stanie zapoznać się z wszystkimi możliwymi postaciami polecenia ALTER, ponieważ są one inne dla każdego typu obiektu definiowanego w bazie danych.

4.3. USUWANIE OBIEKTÓW BAZY DANYCH – POLECENIE DROP

Polecenie DROP języka SQL, w przeciwieństwie do CREATE i ALTER, jest bardzo proste, ponieważ wystarczy wskazać typ obiektu, który ma być usunięty, oraz jego nazwę. Ogólna składnia polecenia DROP przedstawia się następująco:

DROP *TypObiektu NazwaObiektu*.

Usunięcie z bazy danych tabeli o nazwie Osoby będzie miało postać:

```
DROP TABLE Osoby,
```

a usunięcie widoku o nazwie WidokTestowy;

```
DROP VIEW WidokTestowy.
```

Polecenie DROP nie wymaga innych elementów składniowych, a jedynie określenie typu usuwanego obiektu i jego nazwę.

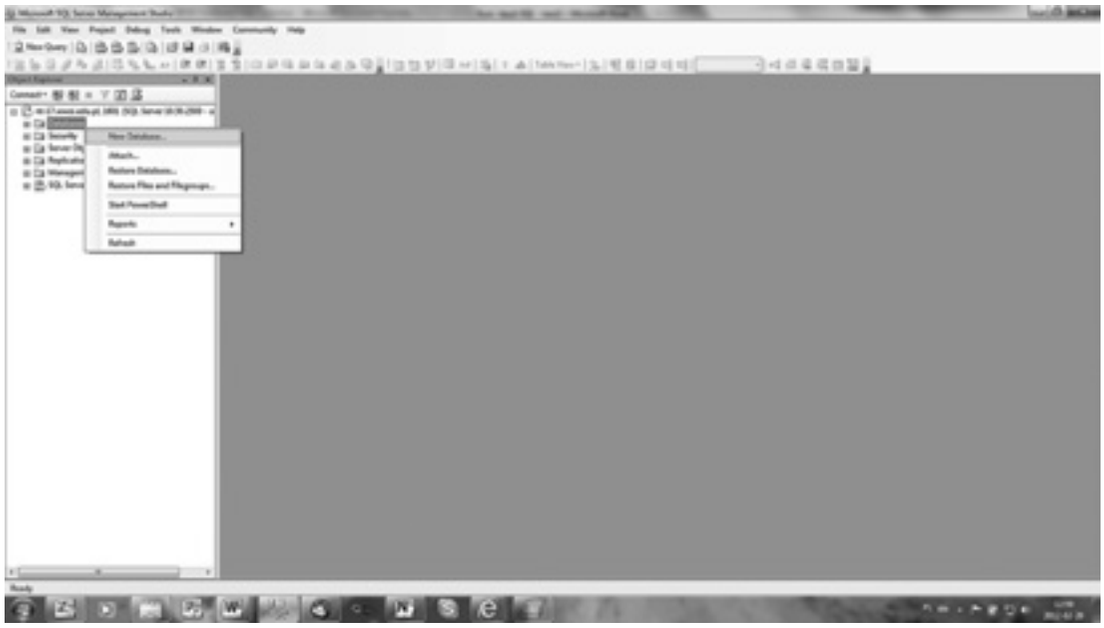
4.4. ĆWICZENIA

4.4.1. Ćwiczenie 2 – Tworzenie bazy danych

Każdy uczestnik kursu utworzy bazę danych na dostępnym serwerze o nazwie KURS*NazwiskoImie* (prefix KURS oraz nazwisko i imię uczestnika kursu).

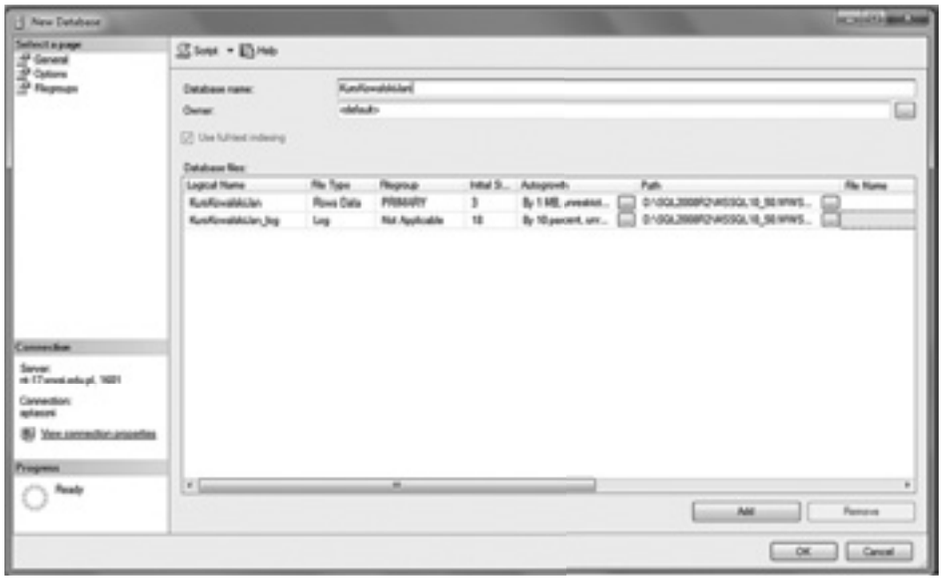
Tworzenie bazy danych realizujemy, w SQL Server Management Studio, według następującego schematu:

1. Wybieramy opcję New Database (klikamy prawym klawiszem myszy na folder Databases w oknie Object Explorer) – jak pokazano na rysunku 10.



Rysunek 10.
Wybór opcji New Database

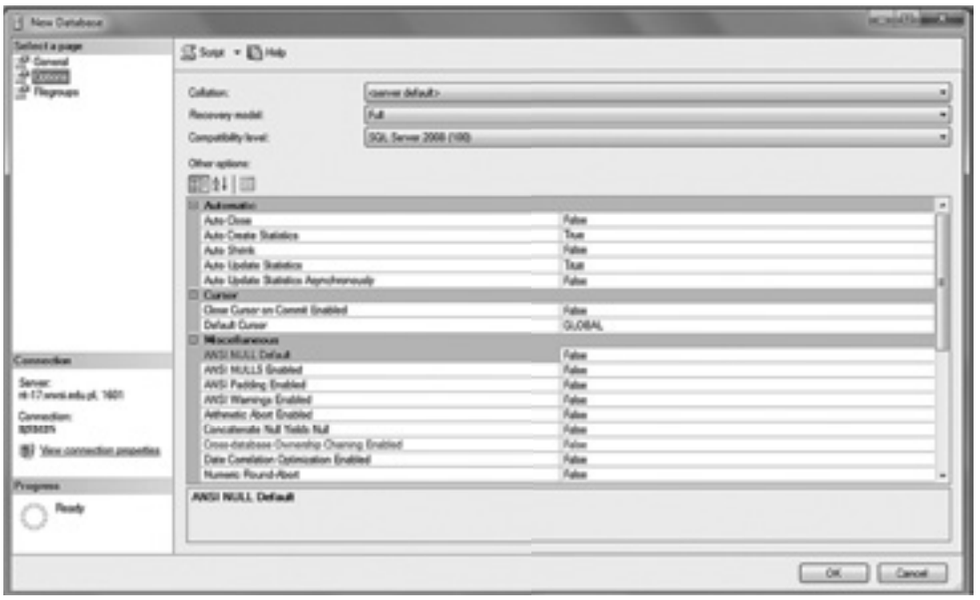
2. Po uruchomieniu opcji New Database, pojawi się kreator nowej bazy danych.



Rysunek 11.
Kreator tworzenia bazy danych

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

3. Przejście na stronę Options



Rysunek 12.
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 13.

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

4.4.2.Ćwiczenie 3 – Definiowanie tabel

W ramach ćwiczenie 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 14.

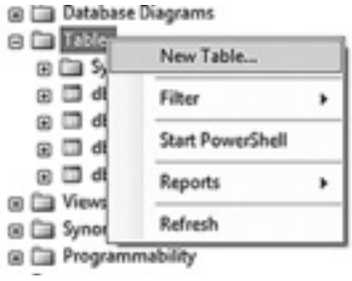
Po wybraniu opcji New Table pojawi się tabela, w której określamy nazwy kolumn, ich typ oraz dodatkowe własności. W ramach ćwiczenia należy zdefiniować dwie tabele. Pierwszą tabelę, o nazwie Osoby, definiujemy zgodnie ze wzorem pokazanym na rysunku 15.

Dodatkowo, dla kolumny IdOsoby, ustawiamy własność Identity Specyfication, tak jak pokazano na rysunku 16, spowoduje to automatyczną numerację kolumny IdOsoby.

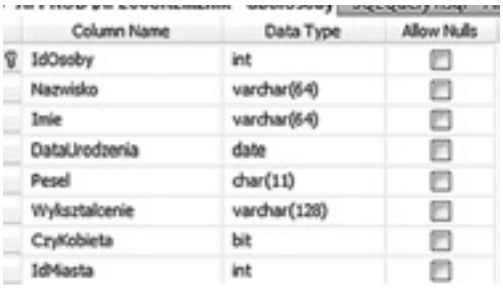
Po zdefiniowaniu tabeli, zgodnie ze wzorem, zamykamy okno projektanta i w oknie, które się pojawi, określamy jej nazwę, jak pokazano na rysunku 17.

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 19.

Rysunek 13.
Widok okna Object Explorer po utworzeniu nowej bazy danych



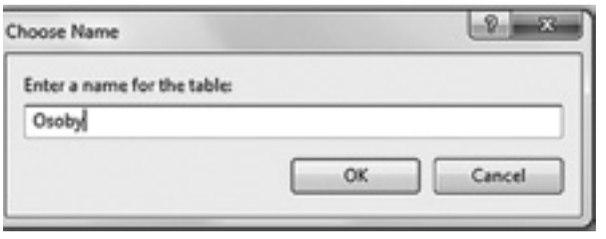
Rysunek 14.
Wybór zadania New Table



Column Name	Data Type	Allow Nulls
IdOsoby	int	
Nazwisko	varchar(64)	
Imie	varchar(64)	
DataUrodzenia	date	
Pesel	char(11)	
Wyksztalcenie	varchar(128)	
CzyKobieta	bit	
IdMiasta	int	

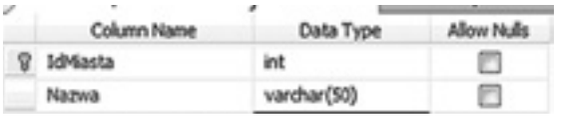
Rysunek 15.
Struktura tabeli Osoby

Rysunek 16.
Ustawianie własności Identity Specification



Rysunek 17.
Ustalenie nazwy tworzonej tabeli

Drugą tabelę, o nazwie Miasta, definiujemy zgodnie ze wzorem przedstawionym na rysunku 18.



Rysunek 18.
Struktura tabeli Miasta

Rysunek 19.
Struktura bazy danych po wykonaniu ćwiczenia

W kolejnych ćwiczeniach dla utworzonych tabel zdefiniujemy reguły poprawności danych.

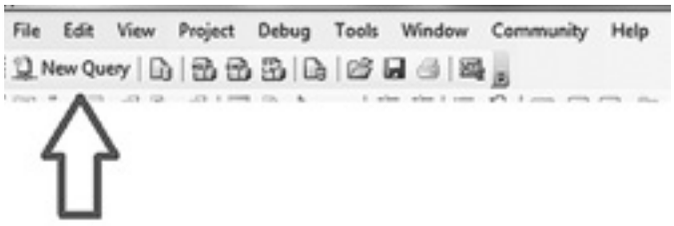
4.4.3.Ćwiczenie 4 – Definiowanie reguł poprawności CHECK i UNIQUE

Tabele utworzone w ramach ćwiczenia 3 zapewniają poprawność danych jedynie w zakresie zdefiniowanych typów danych (typ danych jest określeniem dziedziny dopuszczalnych wartości). Bardzo często jest to ograni-

czenie niewystarczające, dlatego należy definiować dodatkowe ograniczenia. Dla tabel zdefiniowanych w ramach ćwiczenia 3 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,
- zapewnienie, żeby w kolumnie Wykształcenie można było wprowadzać tylko zbiór ustalonych wartości.

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 20.



Rysunek 20.
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 21.

Rysunek 21.
Błąd wygenerowany przy próbie zapisania nazwy miasta już zapisanej w tabeli

Kolejne polecenie, zdefiniuje ograniczenie typu CHEECK (wyrażenie logiczne), które ma zapewnić, że w kolumnie Wykształcenie w tabeli Osoby będzie można zapisać tylko ustalone wartości. Zdefiniujemy także wartość domyślną dla kolumny Wykształcenie. W oknie New Query piszemy następujące polecenie:

ALTER TABLE **Osoby**
ADD CONSTRAINT **Ogr2** DEFAULT ('Brak danych') FOR **Wykształcenie**

Wykonujemy polecenie poprzez wciśnięcie klawisza F5. W kolejnym poleceniu zdefiniujemy ograniczenie zapewniające poprawność numeru Pesel. W skład 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 wieku XXI.

Ostatnim ograniczeniem w ramach tego ćwiczenia będzie ograniczenie dla kolumny Wyksztalcenie w tabeli Osoby, które ma zapewnić, że wartości w tej kolumnie są ograniczone tylko do określonego zbioru.

W oknie New Query piszemy następujące polecenie:

```
ALTER TABLE Osoby
ADD  CONSTRAINT Ogr4 CHECK
 (
    Wyksztalcenie  IN ('Podstawowe' ,'Średnie' ,'Wyższe' ,'Brak danych')
 )
```

Wykonujemy polecenie wciskając klawisz F5.

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.

4.4.4.Ćwiczenie 5 – Definiowanie reguł integralności referencyjnej

Pojęcie integralność referencyjna określa zależności występujące między 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.

5 JĘZYK MANIPULACJI DANYMI SQL DML

5.1 WPROWADZANIE DANYCH DO TABEL

– POLECENIE INSERT

Wprowadzanie danych do tabel realizowane jest poprzez polecenie INSERT języka SQL. Polecenie INSERT umożliwia wstawianie wierszy tylko do jednej wyspecyfikowanej tabeli. Przykładowe polecenia będą odnosiły się do tabel zdefiniowanych w rozdziale 4.

Podstawową składnią polecenia INSERT jest:

```
INSERT (lista kolumn) VALUES (Lista wartości)
```

Przykład:

```
INSERT INTO Osoby (Nazwisko,Imie,DataUrodzenia,Pesel,Wyksztalcenie,CzyKobieta,IdMiasta)
VALUES('Nowak', 'Jan', '1991-03-22', '91032276537','Wyższe',0,1)
```

Zawartość tabeli Osoby, po wykonaniu polecenia, przedstawiono na rysunku 22.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	Wyksztalcenie	CzyKobieta	IdMiasta
4	Nowak	Jan	1991-03-22	91032276537	Wyższe	0	1

Rysunek 22.
Zawartość tabeli Osoby po wykonaniu pierwszego polecenia INSERT

Uwaga: Nie wstawialiśmy żadnych wartości do kolumny IdOsoby, ponieważ dla tej kolumny jest ustawiona właściwość IDENTITY (autonumerowanie). Od wersji SQL Server 2008, składnia polecenia INSERT została rozszerzona o możliwość podawania w klauzuli VALUES danych dla wielu wierszy tak, jak pokazano na poniższym przykładzie:

```
INSERT INTO Osoby (Nazwisko,Imie,DataUrodzenia,Pesel,Wyksztalcenie,CzyKobieta,IdMiasta)
VALUES('Rybak', 'Zofia', '1990-07-21', '90072175686','Średnie',1,2) ,
      ('Kowal', 'Piotr', '1993-02-11', '93021148172','Podstawowe',0,1)
```

Zawartość tabeli Osoby, po wykonaniu polecenia, przedstawiono na rysunku 23.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	Wyksztalcenie	CzyKobieta	IdMiasta
4	Nowak	Jan	1991-03-22	91032276537	Wyższe	0	1
6	Rybak	Zofia	1990-07-21	90072175686	Średnie	1	2
7	Kowal	Piotr	1993-02-11	93021148172	Podstawowe	0	1

Rysunek 23.
Zawartość tabeli Osoby po wykonaniu drugiego polecenia INSERT

W kolejnym przykładzie skorzystamy z wartości domyślnej:
INSERT INTO **Osoby**(Nazwisko,Imie,DataUrodzenia,Pesel,CzyKobieta,IdMiasta)
VALUES('Lis', 'Beata', '1990-09-11', '90091175686', 1,2)

Ponieważ nie została wyspecyfikowana w poleceniu INSERT kolumna Wyksztalcenie to została tam umieszczona wartość określona w ograniczeniu DEFAULT. Nasza tabela po wykonaniu tego polecenia będzie miała następującą zawartość przedstawioną na rysunku 24.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	Wyksztalcenie	CzyKobieta	IdMiasta
1	Nowak	Jan	1991-03-22	91032276537	Wyższe	0	1
2	Rybak	Zofia	1990-07-21	90072175686	Średnie	1	2
3	Kowal	Piotr	1993-02-11	93021148172	Podstawowe	0	1
4	Lis	Beata	1990-09-11	90091175686	Brak danych	1	2

Rysunek 24.
Zawartość tabeli Osoby po wykonaniu trzeciego polecenia INSERT

Dodatkowo składnia polecenia INSERT umożliwia przygotowanie danych, które zamierzamy wstawić do tabeli przez polecenie SELECT lub przez procedurę składowaną uruchamianą poleceniem EXECUTE.
W ujęciu ogólnym polecenia takie będą miały następującą postać:

```
INSERT INTO NazwaTabeli(ListaKolumn)
SELECT ... PostacZapytania
Lub
INSERT INTO NazwaTabeli(ListaKolumn)
EXECUTE NazwaProcedury
```

Warunkiem poprawnego wykonania tak napisanych poleceń jest to, żeby polecenie SELECT lub procedura składowana zwracały tabelę o odpowiedniej strukturze zgodnej z listą kolumn wyspecyfikowaną w poleceniu INSERT.

5.2. MODYFIKACJA DANYCH – POLECENIE UPDATE.

Podstawową składnię polecenia UPDATE można przedstawić następująco:

```
UPDATE NazwaTabeli SET
  ListaZmian
WHERE WarunekSelekcji
```

W przedstawionej postaci składni określmy tabelę, której wiersze będą modyfikowane, listę zmian (jedno wyrażenia dla zmienianej kolumny) oraz warunek określający, które wiersze należy modyfikować.

W celu zademonstrowania działania polecenia UPDATE pokażemy kilka przykładów, korzystając z tabeli wykorzystanej w rozdziale 5.1. Aktualną zawartość tabeli pokazano na rysunku 25.

Zmienić imię osobie o IdOsoby=2 na Joanna oraz zapisać do kolumny Wyksztalcenie wartość domyślną:

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	Wyksztalcenie	CzyKobieta	IdMiasta
1	Nowak	Jan	1991-03-22	91032276537	Wyższe	0	1
2	Rybak	Zofia	1990-07-21	90072175686	Średnie	1	2
3	Kowal	Piotr	1993-02-11	93021148172	Podstawowe	0	1
4	Lis	Beata	1990-09-11	90091175686	Brak danych	1	2

Rysunek 25.
Zawartość tabeli Osoby

```
UPDATE Osoby SET
Imie='Joanna' ,
Wyksztalcenie=DEFAULT
WHERE IdOsoby=2
```

Tabela po wykonaniu modyfikacji będzie miała postać pokazaną na rysunku 26.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	Wyksztalcenie	CzyKobieta	IdMiasta
1	Nowak	Jan	1991-03-22	91032276537	Wyższe	0	1
2	Rybak	Joanna	1990-07-21	90072175686	Brak danych	1	2
3	Kowal	Piotr	1993-02-11	93021148172	Podstawowe	0	1
4	Lis	Beata	1990-09-11	90091175686	Brak danych	1	2

Rysunek 26.
Postać tabeli Osoby po wykonaniu polecenia

W ostatnim przykładzie wykonamy modyfikację danych dla wszystkich wierszy, wstawiając do kolumny Wyksztalcenie wartość 'Średnie'. Polecenie będzie miało postać:

```
UPDATE Osoby SET
Wyksztalcenie='Średnie'
```

Po wykonaniu polecenia tabela będzie miała postać przedstawioną na rysunku 27.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	Wyksztalcenie	CzyKobieta	IdMiasta
1	Nowak	Jan	1991-03-22	91032276537	Średnie	0	1
2	Rybak	Joanna	1990-07-21	90072175686	Średnie	1	2
3	Kowal	Piotr	1993-02-11	93021148172	Średnie	0	1
4	Lis	Beata	1990-09-11	90091175686	Średnie	1	2

Rysunek 27.
Postać tabeli po wykonaniu polecenia

Polecenie UPDATE może zawierać także klauzulę FROM, w której można określić tabele, które chcemy wykorzystać do realizacji modyfikacji (oczywiście musimy dodać także warunek złączenia).

5.3. USUWANIE DANYCH – POLECENIE DELETE

Polecenie DELETE realizuje usuwanie wierszy z tabeli i podstawowa składnia jest następująca:

```
DELETE FROM NazwaTabeli WHERE WarunekSelekcji
```

Brak klauzuli WHERE powoduje usunięcie wszystkich wierszy i może to być często przyczyną incydentalnego usunięcia z tabeli wszystkich wierszy. Przykład poleceń DELETE:

DELETE FROM Osoby WHERE IdOsoby BETWEEN 2 AND 3 – usuwa z tabeli wiersze o wartościach IdOsoby z przedziału ‹2,3›.

Zawartość tabeli Osoby, po wykonaniu tego polecenia, pokazana została na rysunku 28.

IdOsoby	Nazwisko	Imie	DataUrodzenia	Pesel	Wyksztalcenie	CzyKobieta	IdMiasta
1	Nowak	Jan	1991-03-22	91032276537	Średnie	0	1
4	Lis	Beata	1990-09-11	90091175686	Średnie	1	2

Rysunek 28.
Zawartość tabeli Osoby po wykonaniu polecenia

W ostatnim przykładzie usuniemy z tabeli Osoby wszystkie pozostałe wiersze. Polecenie będzie miało postać:

DELETE FROM Osoby.

Po wykonaniu tego polecenia tabela Osoby nie będzie zawierała żadnych wierszy.

W technologii MS SQL Server 2008 dostępne jest jeszcze jedno polecenie usuwające wszystkie dane z tabeli. Poleceniem tym jest TRUNCATE TABLE NazwaTabeli. Różnica pomiędzy poleceniem DELETE bez klauzuli WHERE a poleceniem TRUNCATE jest duża:

- po wykonaniu polecenia DELETE (bez klauzuli WHERE) zostaje zachowana wartość właściwości IDENTITY, czyli automatyczna numeracja wierszy będzie kontynuowana, a w przypadku TRUNCATE nastąpi rozpoczęcie automatycznego numerowania od początku,
- jeżeli w tabeli będzie wyzwalacz dla polecenia DELETE – to po wykonaniu TRUCATE nie będzie on uruchomiony,
- wydajność polecenia TRUNCATE nie jest związana z ilością wierszy zapisanych w tabeli,
- stan tabeli po wykonaniu polecenia TRUNCATE jest taki jak po utworzeniu tabeli.

5.4. ĆWICZENIA

5.4.1. Ćwiczenie 6 – Wprowadzanie danych do przykładowej tabeli

W ramach ćwiczenia należy wprowadzić przykładowe dane do tabel Miasta i Osoby. Każdy uczestnik ma zadanie:

- Wprowadzić do tabeli Miasta 10 przykładowych wierszy
- Wprowadzić do tabeli Osoby 20 przykładowych wierszy

Wprowadzanie danych należy zrealizować poprzez napisanie i wykonanie odpowiednich poleceń INSERT języka SQL. W trakcie wprowadzania danych należy sprawdzić działanie ograniczeń zdefiniowanych w ramach ćwiczeń 4 i 5.
Efekty ćwiczenia zostaną omówione z prowadzącym kurs.

5.4.2. Ćwiczenie 7 – Modyfikacja danych według zadanych warunków

W ramach tego ćwiczenia należy wykonać następujące modyfikacje danych w tabeli Osoby:

- Wprowadzić do kolumny Wyksztalcenie wartość ‘Wyższe’ dla osób z miasta o IdMiasta=1
- Zwiększyć datę urodzenia o 1 osobom urodzonym w marcu (Uwaga: zmiana daty urodzenia wiąże się z koniecznością zmiany numeru Pesel)
- Zmienić imię wszystkim kobietom na ‘Elwira’

Modyfikację danych należy zrealizować poprzez napisanie i wykonanie odpowiednich poleceń UPDATE języka SQL. W trakcie wprowadzania danych należy sprawdzić działanie ograniczeń zdefiniowanych w ramach ćwiczeń 4 i 5.

Efekty ćwiczenia zostaną omówione z prowadzącym kurs.

5.4.3. Ćwiczenie 8 – Usuwanie wybranych wierszy tabeli

W ramach tego ćwiczenia należy usunąć wiersze z tabeli osoby, według następujących zasad:

- Usunąć wiersze z tabeli Osoby dla tych osób, które są powiązane z miastem o IDMiasta=1
- Usunąć wiersze z tabeli Osoby dla tych osób, które urodziły się przed 12-04-1990
- Usunąć wiersze z tabeli Osoby dla tych osób, których nazwisko rozpoczyna się na literę mniejszą niż 'K'

Usuwanie danych należy zrealizować poprzez napisanie i wykonanie odpowiednich poleceń DELETE języka SQL.

Efekty ćwiczenia zostaną omówione z prowadzącym kurs.

6. ZAPYTANIA DO BAZ DANYCH – POLECENIE SELECT SQL

6.1. PODSTAWY SKŁADNI POLECENIA SELECT

Istotą zapytań do baz danych jest wybranie i przetworzenie części danych przechowywanych w bazie tych danych, które są potrzebne z punktu widzenia pewnej potrzeby. Zanim przystąpimy do omawiania polecenia SELECT, wyjaśnijmy istotę zapytań do relacyjnej bazy danych. Dane w bazie relacyjnej są zapisane w postaci dwuwymiarowych tabel i wynik zapytania też jest dwuwymiarową tabelą. Wynika stąd, że zapytanie polega na wybraniu z całej bazy danych takiej tabeli, która spełnia wymagania realizowanego zadania. Realizacja zapytań opiera się na trzech podstawowych operacjach wykonywanych na modelu relacyjnym:

- ***Operacja projekcji*** (zwana także rzutowaniem) – polega na wyborze podzbioru kolumn ze zbioru wszystkich dostępnych. Wynikiem tej operacji dla danej tabeli jest więc inna tabela, w której są dostępne tylko niektóre kolumny z tabeli wyjściowej. Operacja projekcji zmniejsza rozmiar tabeli wyjściowej poprzez wyeliminowanie kolumn, które w danym momencie uznajemy za nieistotne.
- ***Operacja selekcji*** – polega na wyborze podzbioru wierszy ze zbioru wszystkich wierszy dostępnych w danej tabeli. Podstawą operacji selekcji jest wyrażenie logiczne, które decyduje, czy dany wiersz powinien się znaleźć w zbiorze wynikowym. Operacja selekcji zmniejsza rozmiar tabeli wyjściowej poprzez wyeliminowanie wierszy.
- ***Operacja łączenia*** – polega na dołączeniu do tabeli wyjściowej kolumn z innej tabeli na podstawie wartości odpowiedniego wyrażenia logicznego. W relacyjnych bazach danych operację łączenia wykonujemy najczęściej w oparciu o klucz obcy w tabeli wyjściowej i klucz podstawowy w tabeli dołączanej.

Praktycznie wszystkie zapytania są realizowane w oparciu o trzy operacje: projekcję, selekcję i łączenie.

Do realizacji zapytań, język SQL udostępnia polecenie **SELECT**. Polecenie to ma dość złożoną składnię – poniżej przedstawiamy jego uproszczoną postać:

```
SELECT [TOP n] lista_kolumn
    FROM lista_tabel
    WHERE warunki_selekcji
    GROUP BY lista_kolumn_grupowania
    HAVING warunek_selekcji
    ORDER BY lista_kolumn_porzadkowania
```

gdzie:

SELECT – polecenie języka SQL używane do realizacji zapytań do bazy danych,
TOP n – ogranicza liczbę wierszy zapytania do n wierszy,
lista_kolumn – określenie, jakie kolumny i w jakiej postaci mają się znaleźć w wyniku zapytania,

FROM – klauzula polecenia **SELECT**, w której określamy, jakie tabele i w jaki sposób połączone biorą udział w realizacji zapytania,
lista_tabel – określenie, które tabele i jak połączone biorą udział w realizacji zapytania,
WHERE – klauzula polecenia **SELECT**, służąca do określenia warunków selekcji,
warunek_selekcji – wyrażenie logiczne określające, jakie wiersze powinny znaleźć się w tabeli wynikowej,
GROUP BY – klauzula polecenia **SELECT**, definiująca sposób grupowania (wykorzystywana z funkcjami agregującymi, które będą omawiane w dalszej części wykładu),
lista_kolumn_grupowania – określenie kolumn, według których jest realizowana operacja grupowania,
HAVING – klauzula polecenia **SELECT**, tak zwany opóźniony warunek selekcji (wykorzystywany najczęściej z funkcjami agregującymi),
ORDER BY – klauzula polecenia **SELECT**, w której określamy sposób uporządkowania wyników zapytania,
lista_kolumn_porzadkowania – określenie kolumn, według których należy uporządkować wynik zapytania.

Jak widać z powyższego opisu, polecenie **SELECT** nie ma zbyt wielu dodatkowych elementów składni, ale jak zobaczymy w dalszej części wykładu, można przy pomocy pozornie niewielu elementów wyrazić bardzo złożone zapytania.

6.2. OPERACJA ŁĄCZENIA TABEL

6.2.1. Złączenia wewnętrzne

Dane w bazach danych są zapisywane w wielu tabelach, co wynika z zasad projektowania relacyjnych baz danych, a logiczne powiązanie danych zapewniają nam klucze obce. Tworząc zapytania, które muszą korzystać z danych zapisanych w wielu tabelach, należy dokonać odpowiedniego połączenia tabel. W środowisku MS SQL Server 2008 R2, język SQL, udostępnia następujące operatory złączeń: Operator złączenia wewnętrznego – INNER JOIN (klauzula INNER jest domyślna, dlatego można używać tylko klauzuli JOIN) – w wyniku zapytania będą tylko te wiersze, które spełniają warunek podany w klauzuli ON. Przykład zapytania:

```
SELECT      Klasy.Nazwa,
                Uczniowie.Nazwisko,
                Uczniowie.Imie,
                Uczniowie.Pesel
FROM Klasy LEFT JOIN Uczniowie
    ON Klasy.idklasy=Uczniowie.idklasy
WHERE Uczniowie.nazwisko  LIKE 'K%'
```

Przykladowy wynik tego zapytania pokazano na rysunku 29.

Nazwa	Nazwisko	Imie	Pesel
IIa	Kotek	Katarzyna	92031275446
IIa	Kurka	Jola	92060288788
Ia	Krówka	Rysio	92051577646
IIa	Konik	Kasia	93031275446
IIa	Kura	Kasia	93022277654

Rysunek 29.
Wynik zapytania z zastosowaniem złączenia wewnętrznego

6.2.2. Złączenia zewnętrzne

Operator złączenia zewnętrznego – OUTER JOIN z dodatkowym modyfikatorem LEFT, RIGHT lub FULL. Modyfikator określa, z której tabeli do wyniku zapytania mają być dodane wiersze, dla których warunek połączenia nie został spełniony. Przykład zapytania wykorzystującego złączenie zewnętrzne:

```
SELECT  Klasy.Nazwa,
                Uczniowie.Nazwisko,
                Uczniowie.Imie,
                Uczniowie.Pesel
FROM Klasy LEFT JOIN Uczniowie
    ON Klasy.idklasy=Uczniowie.idklasy
        AND Uczniowie.Nazwisko  LIKE 'K%'
```

Przykładowy wynik tego zapytania (w dołączonych wierszach kolumny odpowiadające tabeli Uczniowie, przyjmują wartość null) pokazano na rysunku 30.

Nazwa	Nazwisko	Imie	Pesel
Ia	Krówka	Rysio	92051577646
IIa	Kotek	Katarzyna	92031275446
IIa	Kurka	Jola	92060288788
IIa	Konik	Kasia	93031275446
IIa	Kura	Kasia	93022277654
Ib	NULL	NULL	NULL
IIb	NULL	NULL	NULL
IIc	NULL	NULL	NULL
IIIa	NULL	NULL	NULL
Ia	NULL	NULL	NULL
IIb	NULL	NULL	NULL
IIb	NULL	NULL	NULL
IIb	NULL	NULL	NULL
IIb	NULL	NULL	NULL

Rysunek 30.
Wynik zapytania z zastosowaniem złączenia zewnętrznego LEFT JOIN

6.3. FUNKCJE AGREGUJĄCE

Funkcje agregujące są bardzo ważnym elementem wykorzystywanym w pobieraniu danych, ponieważ umożliwiają obliczenia dla całego wyniku zapytania. W standardzie języka SQL zdefiniowany jest zbiór funkcji agregujących. W poniższej tabeli umieszczono opis podstawowego zbioru funkcji agregujących:

Funkcja agregująca	Działanie
COUNT	Oblicza ilość wierszy
AVG	Oblicza średnią arytmetyczną
SUM	Oblicza sumę wyrażenia dla wyniku zapytania
MIN	Określa minimalną wartość wyrażenia
MAX	Określa maksymalną wartość wyrażenia

Wykorzystanie funkcji agregującej w zapytaniu powoduje, że wynik zapytania zawiera jeden wiersz. Zapytanie, w którym chcemy policzyć, ilu jest uczniów w klasie IIa będzie miało postać:

```
SELECT  COUNT(*) as IluUczniow
FROM Klasy JOIN Uczniowie ON Klasy.idklasy=Uczniowie.idklasy
WHERE klasy.Nazwa='IIa'
```

Wynik zapytania, dla przykładowej bazy danych, zwróci jeden wiersz składający się z jednej kolumny, w której przekazany zostanie wynik funkcji COUNT. Postać wyniku tego zapytania pokazano na rysunku 31.

IluUczniow
14

Rysunek 31.
Wynik zapytania z wykorzystaniem funkcji agregującej
W wyniku zapytania można umieścić wiele funkcji agregujących i wartości stałych:

```
SELECT  COUNT(*) as IluUczniow,
        'To jest wartość stała ' as Opis,
        MAX(DataUrodzenia) as NajstarszyUczen
```

```
FROM Klasy JOIN Uczniowie ON Klasy.idklasy=Uczniowie.idklasy
WHERE klasy.Nazwa='IIa'
```

Wynik tego zapytania dla przykładowej bazy danych pokazano na rysunku 32.

IluUczniow	Opis	NajstarszyUczen
14	To jest wartość stała	1993-12-12

Rysunek 32.
Wynik zapytania z zastosowaniem wielu funkcji agregujących

Funkcje agregujące najczęściej wykorzystuje się w powiązaniu z klauzulą grupującą GROUP BY. Działanie klauzuli GROUP BY zademonstrujemy na przykładzie zapytania, które zwraca dane uczniów z klasy IIa oraz ich średnią ocenę z fizyki. Zapytanie to, skierowane do bazy danych ElektronicznyDziennikOcen, będzie miało następującą postać:

```
SELECT Nazwisko,
       Imie,
       Pesel ,
      AVG(Ocena) as Srednia
FROM Klasy JOIN Uczniowie ON Klasy.idklasy=Uczniowie.idklasy
           JOIN Oceny ON Oceny.iducznia=Uczniowie.iducznia
           JOIN Przedmioty ON Przedmioty.idprzedmiotu=Oceny.idprzedmiotu
WHERE   Klasy.Nazwa='IIa' AND Przedmioty.Nazwa='Fizyka'
GROUP BY   Nazwisko,
           Imie,
           Pesel
```

Wynik tego zapytania dla przykładowej bazy danych pokazano na rysunku 33.

Nazwisko	Imie	Pesel	Srednia
Gazela	Basia	92111177446	3.666666
Konik	Kasia	93031275446	2.615384
Kotek	Katarzyna	92031275446	2.578947
Kura	Kasia	93022277654	3.571428
Kurka	Jola	92060288788	2.769230
Lisek	Kasia	92022277654	3.473684
Łoś	Jola	93060288788	3.642857
Miś	Wacek	93031199123	2.615384
Okoń	Rysio	93051577646	3.312500
Plotka	Wojtek	93030399846	2.428571
Różyczka	Basia	93111177446	2.687500
Ryba	Jan	93051587746	2.875000

Rysunek 33.
Wynik zapytania z zastosowanie klauzuli GROUP BY.

Bez klauzuli GROUP BY próba wykonania zapytania zakończyłaby się błędem:

```
Messages
Msg 8120, Level 16, State 1, Line 1
Column 'Uczniowie.Nazwisko' is invalid in the select list because it
is not contained in either an aggregate function or the GROUP BY clause.
```

Uwaga: Wszystkie kolumny (które nie są wynikiem funkcji agregującej lub wartościami stałymi) należy przenieść do klauzuli GROUP BY.

Kolejną klauzulą stosowaną w połączeniu z funkcjami agregującymi jest klauzula HAVING, którą stosujemy jako filtr, gdy warunek selekcji odnosi się do wyniku funkcji agregującej. Do wcześniejszego zapytania dodamy warunek, żeby w wyniku byli tylko ci uczniowie, którzy osiągnęli średnią ocenę z fizyki większą niż 3.00. Zapytanie będzie miało następująca postać:

```
SELECT Nazwisko,
       Imie,
       Pesel ,
       AVG(Ocena) as Srednia
FROM Klasy JOIN Uczniowie ON Klasy.idklasy=Uczniowie.idklasy
            JOIN Oceny ON Oceny.iducznia=Uczniowie.iducznia
            JOIN Przedmioty ON Przedmioty.idprzedmiotu=Oceny.idprzedmiotu
WHERE   Klasy.Nazwa='IIa' AND Przedmioty.Nazwa='Fizyka'
GROUP BY   Nazwisko,
           Imie,
           Pesel
HAVING AVG(Ocena) ›3.00
```

Wynik tego zapytania dla przykładowej bazy danych pokazano na rysunku 34.

Nazwisko	Imie	Pesel	Srednia
Gazela	Basia	92111177446	3.666666
Kura	Kasia	93022277654	3.571428
Lisek	Kasia	92022277654	3.473684
Łoś	Jola	93060288788	3.642857
Okoń	Rysio	93051577646	3.312500

Rysunek 34.
Wynik zapytania z zastosowaniem klauzuli HAVING.

Klauzuli HAVING nie powinno się używać do określania innych warunków selekcji (nieodnoszących się do wyniku funkcji agregującej).

6.4. ZAPYTANIA ZŁOŻONE

Kolejny mechanizm, dzięki któremu wzrastają znacząco możliwości języka SQL to podzapytania. Dają one możliwość użycia zapytania SELECT wewnątrz innego zapytania SELECT, stąd jedno z nich będziemy nazywać zapytaniem zewnętrznym, a drugie zapytaniem wewnętrznym. Oczywiście poziomów zagnieżdżeń może być więcej i dane zapytanie SELECT może być zarówno zewnętrzne, jak i wewnętrzne. Istniej jedna ogólna zasada określająca, gdzie można umieścić podzapytanie: wszędzie tam, gdzie wynik podzapytania ma sens. Rozróżniamy dwa typy podzapytań:

- Podzapytania skorelowane
- Podzapytania nieskorelowane

Różnicę pomiędzy typami podzapytań pokażemy na przykładach:

Zapytanie złożone nieskorelowane

Chcemy wybrać dane uczniów za wyjątkiem tych, którzy należą do klasy, którzy w swojej nazwie mają literę 'a'.

```
Select Nazwisko,
       Imie,
       Pesel
FROM Uczniowie
WHERE idklasy NOT IN  (
```

```
    SELECT Idklasy
        FROM Klasy
    WHERE Nazwa LIKE '%a%'
        )
```

Dla przykładowej bazy danych otrzymujemy wynik pokazany na rysunku 34.

Nazwisko	Imie	Pesel
Wilczek	Jasio	99121209876
Antylopa	Kasia	92031254567
Rekin	Jan	92051555432
Orka	Kasia	92022288776
Foka	Jola	92060223454
Sum	Wacek	91031134565
Rak	Wojciech	91010123432

Rysunek 34.
Przykład zapytania złożonego (nieskorelowanego)

Uwaga: Podzapytanie nieskorelowane może być wykonane samodzielnie – nie jest związane z zapytaniem nadrzędnym.

Zapytanie złożone skorelowane
Chcemy pobrać dane uczniów z klasy o wartości idklasy=1, wynik ma zawierać nazwisko, imię, numer Pesel oraz średnią ocen danego ucznia.

```
Select Nazwisko,
       Imie,
       Pesel,
       (
         SELECT  AVG(Ocena)
          FROM Oceny
          WHERE iducznia=Uczniowie.iducznia
       ) AS Srednia
FROM Uczniowie
WHERE idklasy=1
```

Dla przykładowej bazy danych otrzymujemy wynik pokazany na rysunku 35.

Nazwisko	Imie	Pesel	Srednia
Piesek	Jan	92051587746	2.980000
Gąska	Wacek	91031199123	2.857142
Krówka	Rysio	92051577646	2.781818
Zebra	Wojtek	93030399846	3.075757
Sarenka	Rysio	92121278766	2.563636

Rysunek 35.
Wynik zapytania złożonego skorelowanego

Podzapytanie z naszego przykładu nie może być wykonane samodzielnie, ponieważ odwołuje się do tabeli Uczniowie. Zapytania złożone pozwalają w jednym poleceniu wykonać bardzo złożone operacje.

6.5. ĆWICZENIA

6.5.1. Ćwiczenie 9. Zapytania do jednej tabeli

W ramach ćwiczenia należy samodzielnie wykonać następujące zapytania:

- Wybrać dane uczniów (nazwisko, imię i numer Pesel), którzy są dowiązani do klasy o wartości IdKlasy=1. Wynik zapytania uporządkować alfabetycznie według nazwiska.
- Wybrać dane klas (nazwa klasy i rok szkolny) z roku szkolnego 2011/2012. Wynik zapytania uporządkować malejąco według nazwy klasy.
- Wybrać dane nauczycieli (nazwisko, imię i tytuł), którzy mają więcej niż 40 lat. Wynik zapytania uporządkować alfabetycznie według nazwiska.

Problemy przy realizacji zadania oraz ich wyniki omówić z prowadzącym kurs.

6.5.2. Ćwiczenie 10. Zapytania z wykorzystaniem łączenia tabel

W ramach ćwiczenia należy samodzielnie wykonać następujące zapytania:

- Wybrać dane uczniów (nazwisko, imię, data urodzenia oraz nazwa klasy) dla uczniów, którzy urodzili się w roku 1993. Wynik zapytania uporządkować malejąco według daty urodzenia.
- Wybrać listę ocen (nawa przedmiotu, rodzaj oceny, wartość oceny oraz data jej wystawienia), które otrzymał uczeń o wartości IdUcznia=2. Wynik uporządkować malejąco według daty wystawienia oceny.
- Wybrać dane nauczycieli (nazwisko, imię oraz tytuł), którzy wystawili ocenę uczniom z klasy Ia. Wynik uporządkować alfabetycznie według nazwiska nauczyciela.

Problemy przy realizacji zadania oraz ich wyniki omówić z prowadzącym kurs.

6.5.3. Ćwiczenie 11. Zapytania z wykorzystaniem funkcji agregujących

W ramach ćwiczenia należy samodzielnie wykonać następujące zapytania:

- Wybrać dane uczniów (nazwisko, imię, data urodzenia oraz średnią ocen z fizyki) dla uczniów, którzy mają mniej niż 18 lat. Wynik zapytania uporządkować malejąco według daty urodzenia.
- Wybrać dane uczniów (nazwisko, imię, płeć oraz średnia ocen z matematyki) dla tych uczniów, których średnia ocen z matematyki jest większa niż 3.50. Wynik uporządkować malejącego według średniej ocen

Problemy przy realizacji zadania oraz ich wyniki omówić z prowadzącym kurs.

6.5.4. Ćwiczenie 13. Zapytania złożone

W ramach ćwiczenia należy samodzielnie wykonać następujące zapytania:

- Wybrać dane uczniów (nazwisko, imię, data urodzenia oraz numer Pesel) dla tych uczniów, którzy w roku 2010 nie otrzymali oceny niedostatecznej z matematyki. Wynik zapytania uporządkować alfabetycznie według nazwiska ucznia.
- Wybrać dane nauczycieli (nazwisko, imię i tytuł), którzy nie wystawiali ocen w klasie Ia.

Problemy przy realizacji zadania oraz ich wyniki omówić z prowadzącym kurs.

7 INNE OBIEKTY BAZY DANYCH

7.1. WIDOKI

W zależności od potrzeb w bazach danych można definiować tabele wirtualne jako wynik pewnego zapytania. Obiekty takie nazywamy widokami lub perspektywami (ang. *View*). Tworzenie widoku w bazie danych realizujemy za pomocą polecenia CREATE VIEW. W poleceniu tym należy umieścić zapytanie SELECT. Istotę tworzenia widoków omówimy na przykładzie, w którym utworzymy widok o nazwie UczniowiePelnoletni. Polecenie tworzące taki widok będzie miało następującą postać.

```
CREATE VIEW UczniowiePelnoletni
AS
SELECT Nazwisko, Imie, Pesel, DataUrodzenia, Klasy.Nazwa as Klasa,
        CASE CzyChlopak
          WHEN 1 THEN ‘Mężczyzna’
```

```
            ELSE 'Kobieta
        END Płeć
FROM Uczniowie JOIN Klasy ON Uczniowie.IdKlasy=Klasy.IdKlasy
WHERE YEAR(GetDate() ) – Year(DataUrodzenia) >17
```

Wykonanie tego polecenie spowoduje utworzenie w bazie danych wirtualnej tabeli, której zawartość będzie tworzona dynamicznie, z załączonego do polecenia zapytania, w momencie odwołania się do widoku. W zapytaniach odwołujemy się do widoku tak jak do każdej standardowej tabeli. Załóżmy, że chcemy uzyskać dane pełnoletnich uczennic z klasy Ia. Zamiast tworzyć zapytanie odwołujące się do tabel bazy danych możemy skorzystać ze zdefiniowanego widoku. W tym celu należy napisać następujące zapytanie:

```
SELECT Nazwisko, Imie, Pesel, Płeć, DataUrodzenia
FROM UczniowiePelnoletni
WHERE Płeć='Kobieta'
```

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

Nazwisko	Imie	Pesel	Płeć	DataUrodzenia
Kotek	Katarzyna	92031275446	Kobieta	1992-03-12
Lisek	Kasia	92022277654	Kobieta	1992-02-22
Kurka	Jola	92060288788	Kobieta	1992-06-02
Gazela	Basia	92111177446	Kobieta	1992-11-11
Sarenka	Rysio	92121278766	Kobieta	1992-12-12
Konik	Kasia	93031275446	Kobieta	1993-03-12
Kura	Kasia	93022277654	Kobieta	1993-02-22
Łoś	Jola	93060288788	Kobieta	1993-06-02
Różyczka	Basia	93111177446	Kobieta	1993-11-11
Stokrotka	Rysio	93121278766	Kobieta	1993-12-12
Antylopa	Kasia	92031254567	Kobieta	1992-03-12
Orka	Kasia	92022288776	Kobieta	1992-02-22
Foka	Jola	92060223454	Kobieta	1992-06-02

Rysunek 35.
Wynik zapytania skierowanego do widoku UczniowiePelnoletni

Tworząc widoki w bazach danych możemy, w zależności od problemu, osiągnąć różne korzyści:

- Uproszczenie schematu bazy danych – dane przechowujemy w kilku tabelach, które bardzo często muszą być łączone. Tworzymy widok, który udostępnia dane z kilku połączonych tabel,
- Sterowanie bezpieczeństwem dostępu do danych – widok zawiera tylko wybrane kolumny z pewnej tabeli i niektórzy użytkownicy uzyskują prawa do widoku, a nie do tabeli macierzystej, tym samym nie mają dostępu do kolumn nieumieszczonych w tym widoku.

Zagadnienia związane z wykorzystaniem widoków są zdecydowanie bardziej złożone, ale problemy z tym związane wybiegają poza zakres tego kursu.

7.2. FUNKCJE TABELARYCZNE

W ramach technologii MS SQL Server 2008 R2 może definiować funkcje tabelaryczne, które tworzą w bazie danych tabele wirtualne. Tabelami wirtualnymi są także widoki, omówione w rozdziale 7.1. Podstawową różnicą pomiędzy tymi dwoma typami tabel wirtualnych jest możliwość wykorzystania parametrów przez funkcje tabelaryczne. Istotę funkcji tabelarycznych omówimy na przykładzie. Załóżmy, że chcemy utworzyć tabelę wirtualną o nazwie UczniowieKlasy, która zawiera listę uczniów z pewnej klasy. Postać polecenia definiującego taką tabelę będzie miało następującą postać:

```
CREATE FUNCTION UczniowieKlasy (@Idklasy int)
RETURNS Table
```

```
AS
RETURN
(
 SELECT Nazwisko, Imie, Pesel, DataUrodzenia,
          CASE CzyChlopak
             WHEN 1 THEN ‘Mężczyzna’
             ELSE ‘Kobieta
          END Płeć
FROM Uczniowie
WHERE IdKlasy=@IdKlasy
)
```

W nagłówku polecenia CREATE FUNCTION zadeklarowany został parametr o nazwie @idklasy, 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 UczniowieKlasy, do której możemy się odwoływać tak jak do innych tabel (przekazując zdefiniowane parametry).

Załóżmy, że chcemy wybrać z bazy danych dane mężczyzn z klasy o wartości idklasy=1. Zapytanie skierowane do wirtualnej tabeli UczniowieKlasy będzie miało następującą postać:

```
SELECT *
FROM UczniowieKlasy(1)
WHERE Płeć=’Mężczyzna’
```

Wynik tego zapytania dla przykładowej bazy danych pokazano na rysunku 36.

Nazwisko	Imie	Pesel	DataUrodzenia	Płeć
Piesek	Jan	92051587746	1992-05-15	Mężczyzna
Gąska	Wacek	91031199123	1991-03-11	Mężczyzna
Krówka	Rysio	92051577646	1992-05-15	Mężczyzna
Zebra	Wojtek	93030399846	1993-03-13	Mężczyzna

Rysunek 36.
Wynik zapytania z wykorzystaniem odwołania do funkcji tabelarycznej (@Idklasy=1)

To samo zapytanie, przy innej wartości parametru, przekaże inny wynik.

```
SELECT *
FROM UczniowieKlasy(2)
WHERE Płeć=’Mężczyzna’
```

Wynik tego zapytania dla przykładowej bazy danych pokazano na rysunku 37.

Nazwisko	Imie	Pesel	DataUrodzenia	Płeć
Ryba	Jan	93051587746	1993-05-15	Mężczyzna
Miś	Wacek	93031199123	1993-03-11	Mężczyzna
Okoń	Rysio	93051577646	1993-05-15	Mężczyzna
Płotka	Wojtek	93030399846	1993-03-13	Mężczyzna
Nowak	Piotr	92091298795	1992-09-12	Mężczyzna

Rysunek 37.
Wynik zapytania z wykorzystaniem odwołania do funkcji tabelarycznej (@Idklasy=2)

Definiowanie funkcji tabelarycznych może uprościć tworzenie zapytań do bazy danych.

7.3. ĆWICZENIA

7.3.1. Ćwiczenie 15 – Definiowanie widoków

W ramach ćwiczenia należy zdefiniować dwa widoki według następujących założeń:

- Zdefiniować widok zawierający nazwę klasy, nazwę przedmiotu oraz średnie ocen danej klasy z danego przedmiotu.
- Zdefiniować widok zawierający dane uczniów (nazwisko, imię, pesel, idklasy) dla tych uczniów, którzy urodzili się po 12-09-1993.

Problemy przy realizacji zadania oraz ich wyniki omówić z prowadzącym kurs. Po wykonaniu polecenia definiującego widok należy napisać zapytanie odczytujące dane z tego widoku

7.3.2. Ćwiczenie 16 – Definiowanie funkcji tabelarycznych

W ramach ćwiczenia należy zdefiniować dwie funkcje tabelaryczne według nastepujących założeń:

- Zdefiniować funkcję tabelaryczną, która tworzy tabele zawierającą nazwisko, imię i numer Pesel tych uczniów, którzy osiągnęli średnią ocen większą od zadanej wartości (zadana wartość powinna być parametrem tej funkcji)
- Zdefiniować funkcję tabelaryczną, która tworzy tabelę zawierającą nazwę przedmiotu oraz średnią ocen z danego przedmiotu dla zadanej klasy w zadanym roku.

Problemy przy realizacji zadania oraz ich wyniki omówić z prowadzącym kurs. Po wykonaniu polecenia definiującego funkcję tabelaryczną należy napisać zapytanie odczytujące dane z wykorzystaniem tej funkcji dla różnych wartości parametrów.

8 PODSUMOWANIE

W ramach naszego kursu dokonaliśmy zapoznania z podstawami języka SQL. Należy zdawać sobie sprawę, że język ten jest bardziej złożony niż pokazane w trakcie kursu przykłady. Wielu możliwości języka SQL nie omówiliśmy ze względu na ograniczone ramy czasowe kursu.

9 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