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+

Kuźnia Talentów:

Zaawansowany kurs języka 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.

Zaawansowany kurs języka SQL

Rodzaj zajęć: Kuźnia Talentów
Tytuł: Zaawansowany kurs języka 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.

Zaawansowany kurs języka SQL

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

Streszczenie

W ramach kursu „Zaawansowany kurs języka SQL" uczestnikom zapoznanie zaprezentowana technologia MS SQL Server 2008 R2. Poznają język SQL ze szczególnym uwzględnieniem poleceń wprowadzonych do standardu języka SQL w ostatnich latach. Omówione zostaną przykłady zapytań rekurencyjnych z wykorzystaniem wyrażeń CTE oraz poznają nowe operatory złączenia tabel; operator tabeli przestawnej PIVOT i operator złączenia dynamicznego CROSS APPLY. W ramach kursu przewidziano wykonanie wielu ćwiczeń, w ramach których zaprezentowane zostanie wykorzystanie zaawansowanych elementów języka SQL.

Spis treści

1. **Wprowadzenie do technologii MS SQL Server 2008 R2** ... 5
   - 1.1. Wstęp ... 5
   - 1.2. Przykładowa baza danych ... 6
   - 1.3. Ćwiczenia ... 7
     - 1.3.1. Ćwiczenie 1 – Instalacja MS SQL Server 2008 R2 Express ... 7
     - 1.3.2. Ćwiczenie 2 – Zapoznanie ze środowiskiem MS SQL Server 2008 R2 ... 12

2. **Nowe elementy języka SQL** ... 15
   - 2.1. Klauzula OUTPUT poleceń modyfikacji danych(SQL–DML) ... 15
   - 2.2. Polecenie MERGE ... 17
   - 2.3. Wyrażenia CTE ... 19
   - 2.4. Ćwiczenia ... 21
     - 2.4.1. Ćwiczenie 2 – Wykorzystanie klauzuli OUTPUT w poleceniach INSERT, UPDATE i DELETE ... 21
     - 2.4.2. Ćwiczenie 3 – Modyfikacja wykazu uczniów na podstawie tabel wyników ... 21

3. **Zaawansowane techniki realizacji zapytań** ... 23
   - 3.1. Fazy przetwarzania zapytań ... 23
   - 3.2. Łączenie tabel ... 25
   - 3.3. Funkcje agregujące ... 29
   - 3.4. Operatory działania na zbiorach ... 31
   - 3.5. Zapytania złożone ... 35
   - 3.6. Funkcje szeregujące ... 36
   - 3.7. Ćwiczenia ... 41
     - 3.7.1. Ćwiczenie 5 – Wykorzystanie operatora złączenia zewnętrznego ... 41
     - 3.7.2. Ćwiczenie 6 – Zapytanie z wykorzystaniem operatorów działania na zbiorach ... 42
     - 3.7.3. Ćwiczenie 7 – Zapytania rekurencyjne ... 42
     - 3.7.4. Ćwiczenie 8 – Zapytania z wykorzystaniem funkcji szeregujących ... 42

4. **Podsumowanie** ... 43

5. **Literatura** ... 43

1 WPROWADZENIE DO TECHNOLOGII MS SQL SERVER 2008 R2

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.

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 następujących 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 następujących zastosowaniach:

- MS SQL Server 2008,
- Oracle,
- MySQL,
- Access,
- DB2,
- 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).

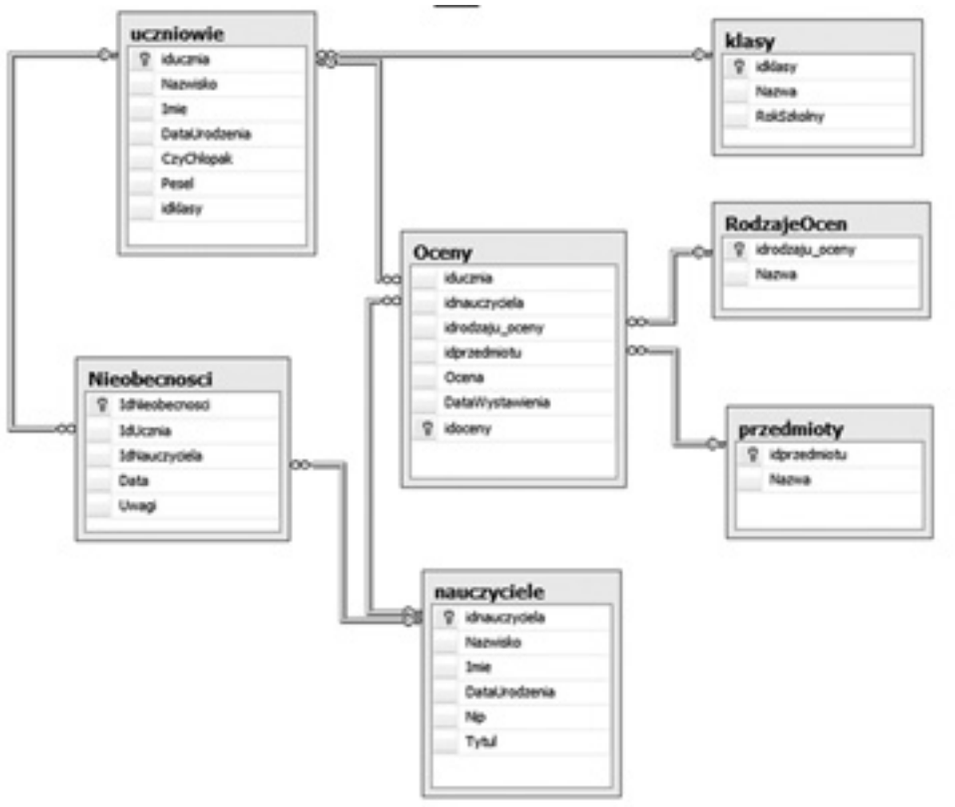
Silnik bazy danych (Database Engine) zawiera wiele różnych dodatkowych technologii:

- Usługi asynchronicznego przetwarzania (Service Broker) – umożliwiają realizację asynchronicznego przetwarzania z wykorzystaniem 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.

1.2. PRZYKŁADOWA BAZA DANYCH.

Testowa baza danych rejestrująca oceny wystawiane w pewnej szkole. Podstawowa tabela Oceny, w której zapisywane są wystawione oceny wraz z datą wystawienia. Ocena powiązana jest z uczniem, nauczycielem, przedmiotem i rodzajem ocen. Dodatkowo, w bazie danych istnieje tabela Nieobecności, w której zapisywane są dane opisujące nieobecność danego ucznia w powiązaniu z datą tej nieobecności.



Rysunek 1.
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. Jej zawartość 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***).

1.3. ĆWICZENIA

1.3.1. Ćwiczenie 1 – Instalacja MS SQL Server 2008 R2 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 2.

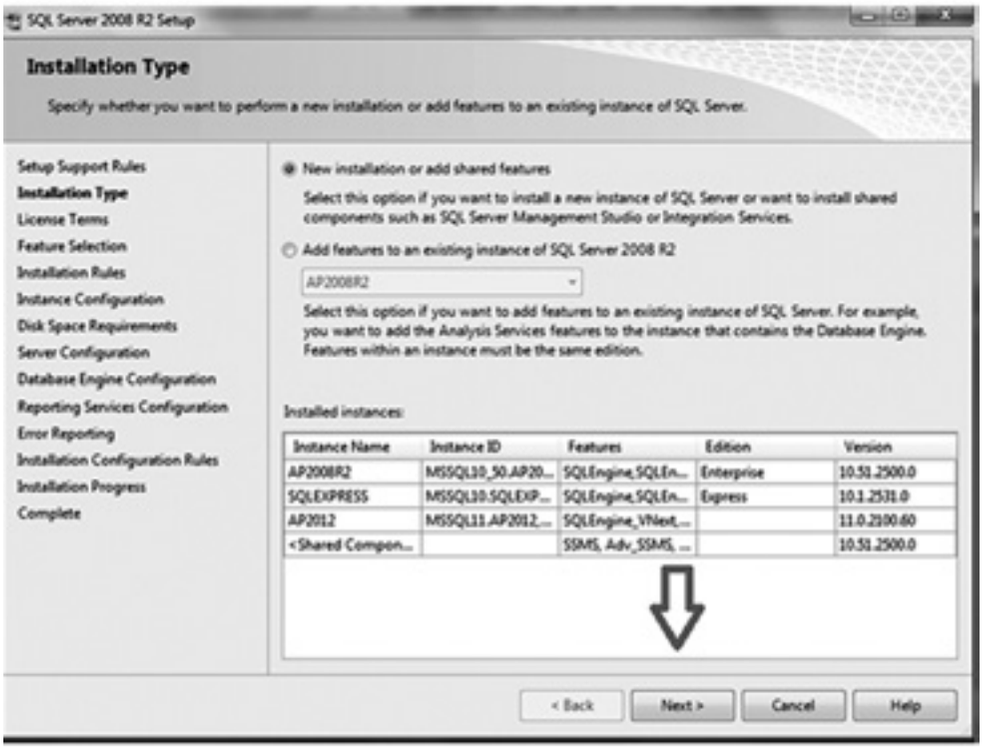
Rysunek 2.
Strona producenta SQL Server 2008 – wybór pobierania produktu

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

W oknie instalatora wybieramy opcję nowej instalacji, jak pokazano na rysunku 3.

Rysunek 3.
Okno podstawowe instalatora

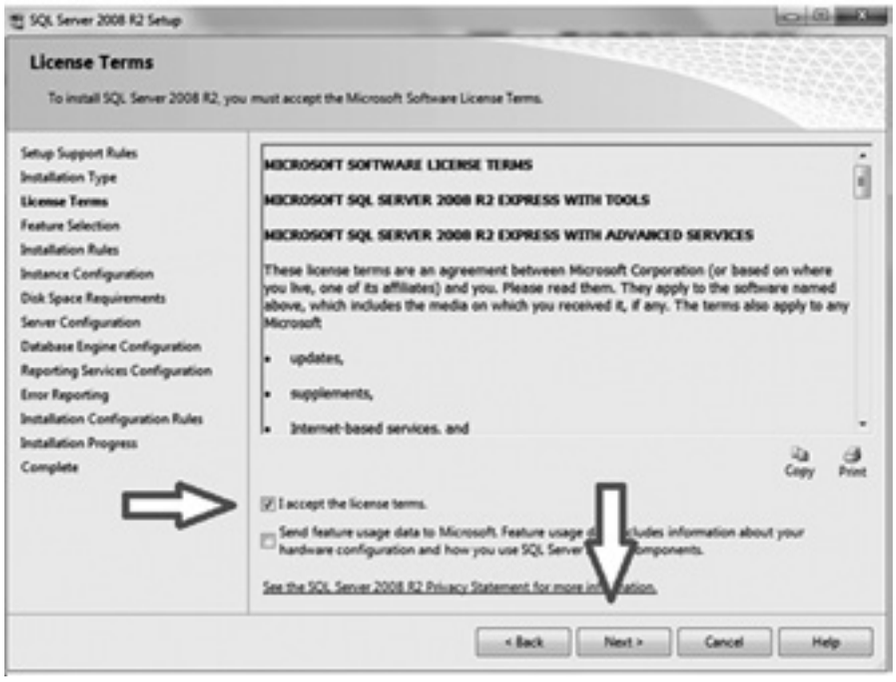
Po automatycznym wykonaniu, przez instalatora, czynności wstępnych pojawi się okno (rysunek 4) procesu instalacji. W oknie klikamy przycisk Next.



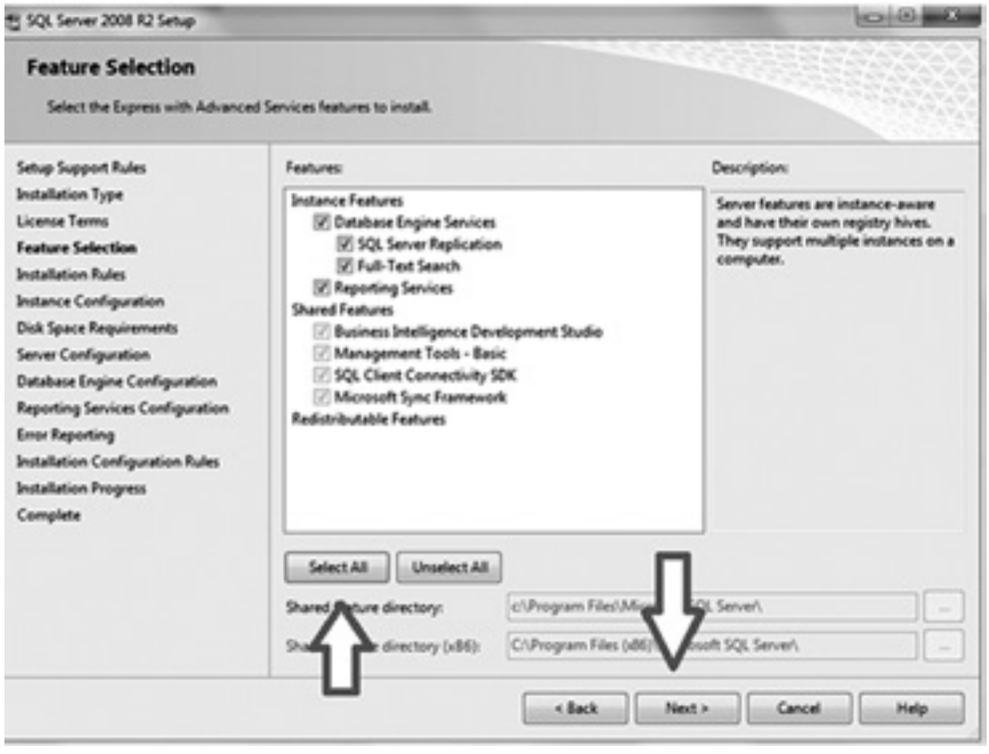
Rysunek 4.
Okno rozpoczęcia procesu instalacji

W następnym kroku musimy zaakceptować warunki licencji (jest ona bezpłatna) tak jak pokazano na rysunku 5 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 6.

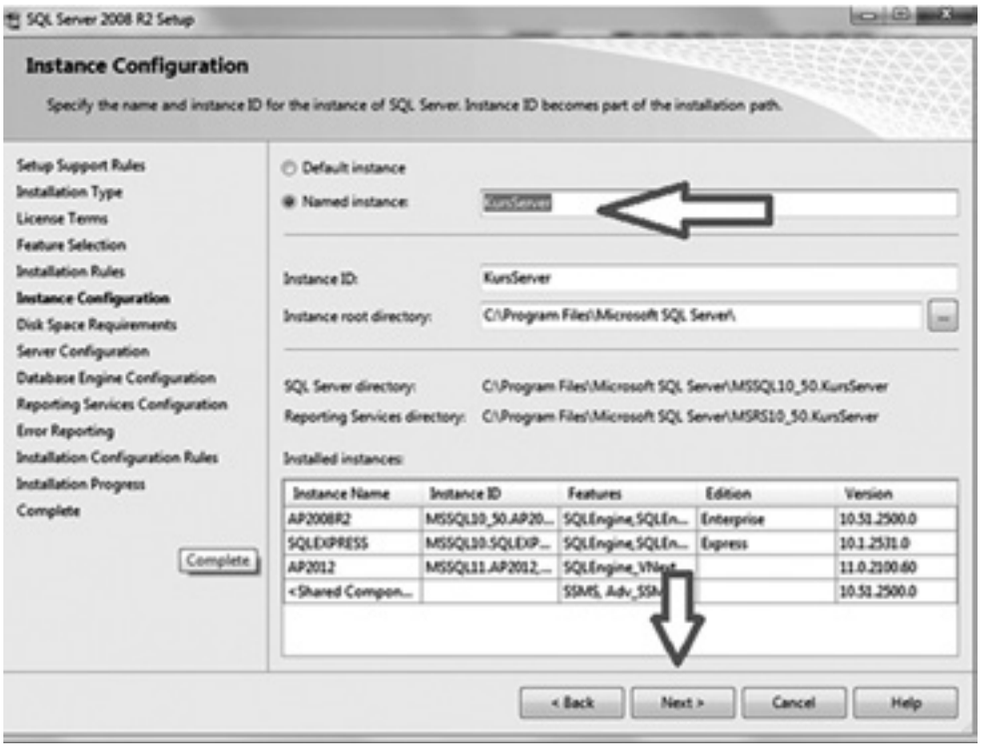


Rysunek 5.
Okno akceptowania postanowień licencji produktu



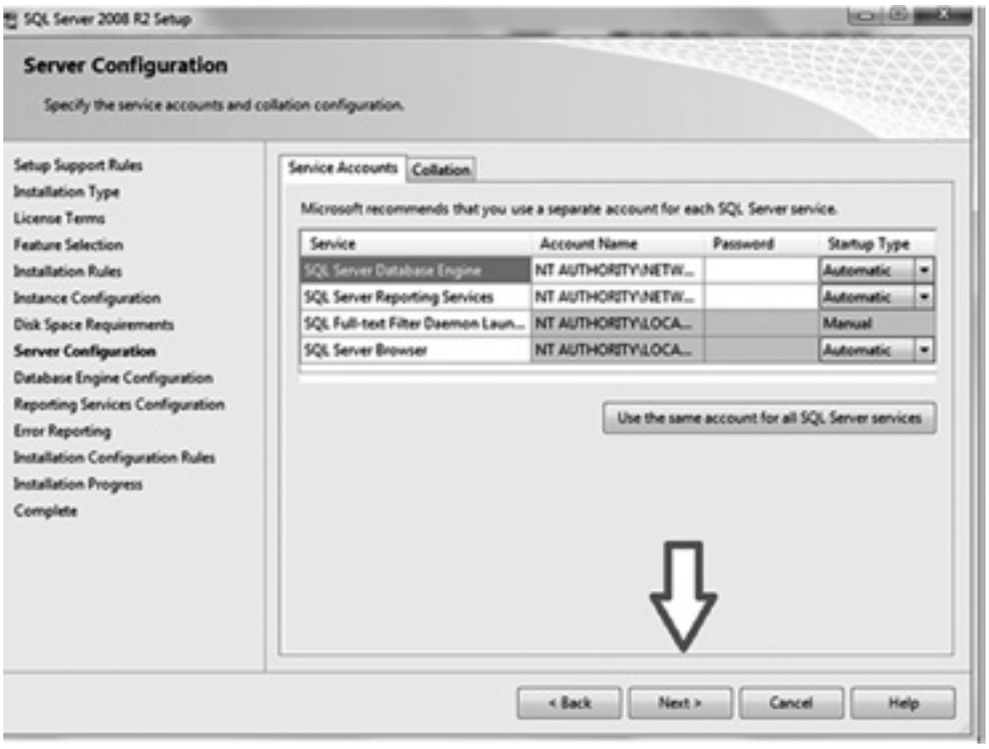
Rysunek 6.
Okno wyboru elementów technologii do instalacji

W kolejnym kroku musimy(rysunek 7) nazwać instancje naszego serwera. Po wprowadzeniu nazwy klikamy przycisk Next.



Rysunek 7.
Okno instalatora – ustalanie nazwy instancji serwera

Następnie ustalamy konta, na których będą uruchomione odpowiednie usługi SQL Servera. W naszym przypadku przyjmiemy wartości domyślne, zaproponowane przez instalator i klikniemy przycisk Next (rysunek 8).

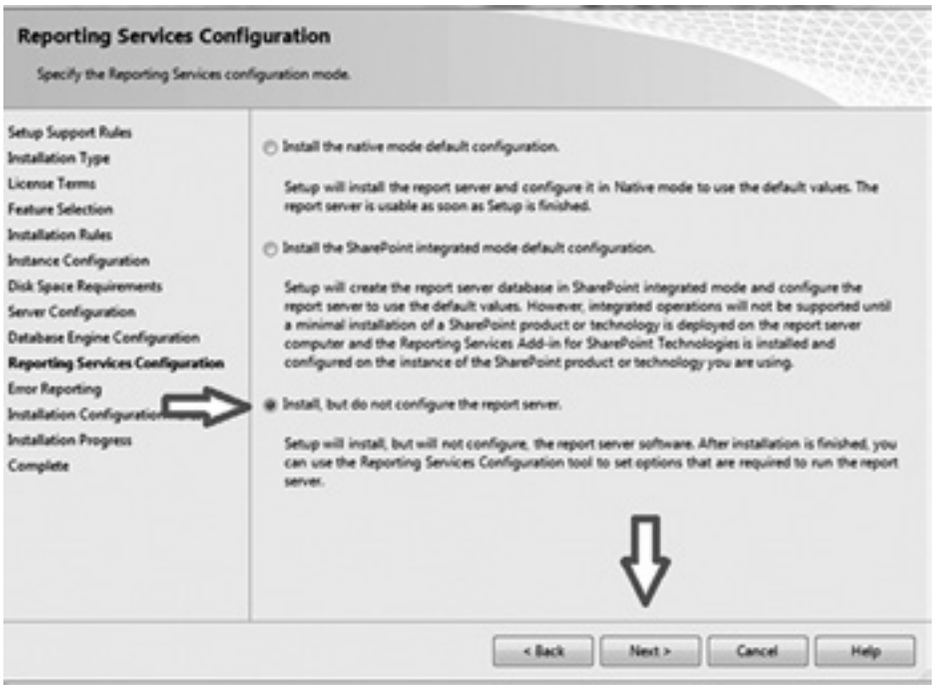


Rysunek 8.
Okno wyboru kont dla usług serwera

W kolejnym kroku (rysunek 9) ustalamy tryb uwierzytelnienia SQL Server – przyjmujemy wartości domyślne i naciskamy przycisk NEXT.

Rysunek 9.
Okno wyboru trybu uwierzytelniania

Następny etap to ustalenie sposobu konfigurowania usługi Reporting Services. Wybieramy opcję „Install, but not configure the report server" (rysunek 10) i wciskamy przycisk Next.



Rysunek 10.
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.

1.3.2. Ćwiczenie 2 – Zapoznanie ze środowiskiem MS SQL Server 2008 R2.

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

- 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,
- bezpośrednie przekazywanie do wykonania poleceń SQL,
- śledzenie planów wykonania zapytań,
- realizacja zadań administracyjnych.

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 okno logowania

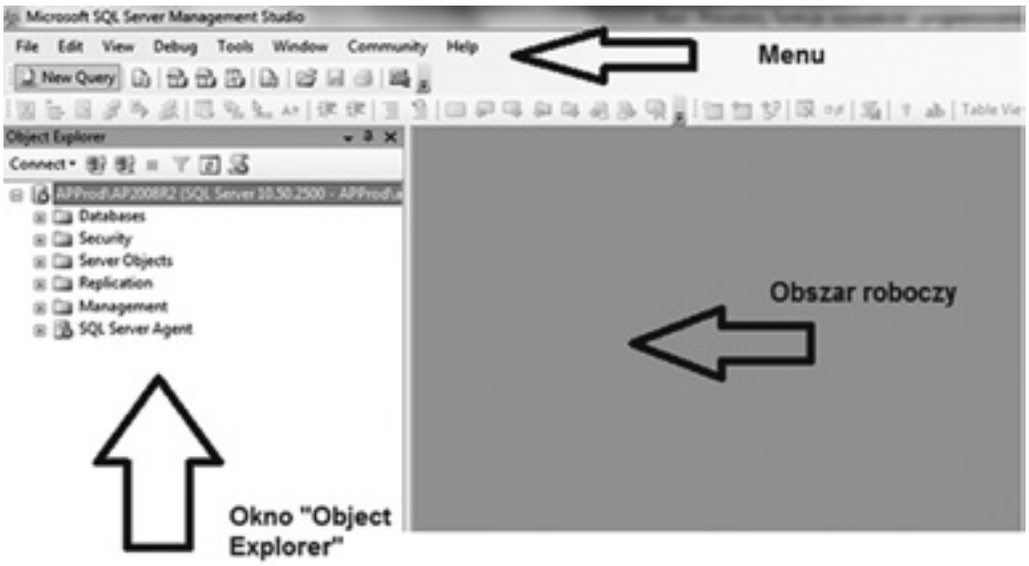
Rysunek 11.
Okno logowania programu SQL Server Management Studio

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

Uwaga: Szczegółowe parametry logowania (login i hasło) zostaną podane przez prowadzącego kurs.

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 12. Najistotniejszym elementem tego interfejsu jest okno Object Explorer, którego ogólną postać widzimy na rysunku 13.



Rysunek 12.
Postać wyjściowa panelu MS SQL Server Management Studio 2008 R2

Rysunek 13.
Okno Object Explorer – kontekst ogólny

W oknie Object Explorer można, w zależności od wybranego kontekstu, 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. Rozwinięcie folderu Databases, jak pokazano na rysunku 14, wyświetla wszystkie bazy danych zdefiniowane na instancji serwera.

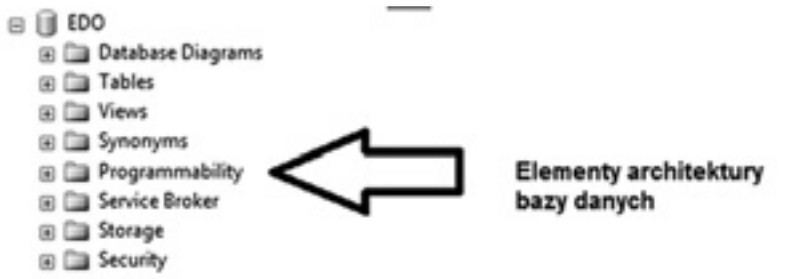
W tym kontekście możemy uzyskać dostęp do szczegółów definicji wszystkich baz danych. Dla wybranej bazy danych możemy przejść do widoku szczegółów. Kontekst wybranej bazy danych pokazano na rysunku 15.

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

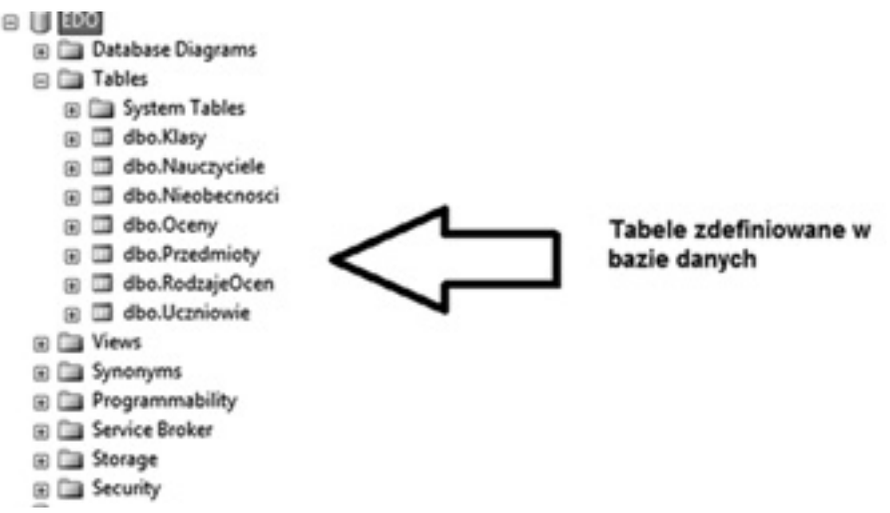
Na rysunku 16 pokazano kontekst Tables, w ramach którego widoczne są wszystkie tabele zdefiniowane w wybranej bazie danych.

Dla wybranej tabeli, co pokazano na rysunku 17, 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 14.
Okno Object Explorer – kontekst Databases



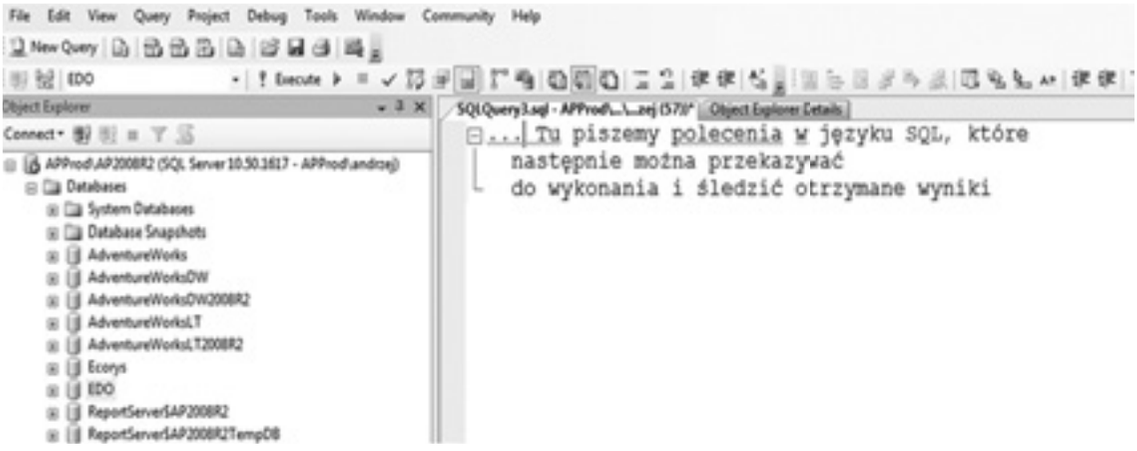
Rysunek 15.
Okno Object Explorer – kontekst wybranej bazy danych



Rysunek 16.
Okno Object Explorer – kontekst Tables

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.

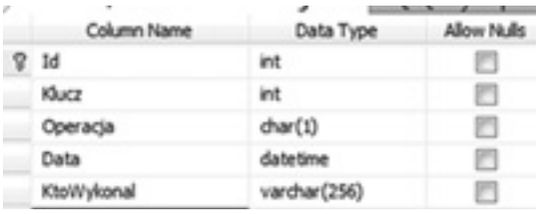
Warto wiedzieć, że każde otwarte okno New Query tworzy odrębną sesję połączeniową z bazą danych, dlatego nie jest dobrym zwyczajem otwieranie niezliczonej ilości tych okien (choć teoretycznie możemy ich otworzyć bardzo dużo). Tekst zapisany w oknie Query możemy, jeżeli jest taka konieczność, zapisać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

2 NOWE ELEMENTY JĘZYKA SQL

2.1. KLAUZULA OUTPUT POLECEŃ MODYFIKACJI DANYCH(SQL –DML).

W wersji SQL Server 2005 do poleceń modyfikacji danych (INSERT, UPDATE, DELETE) dodano nową klauzulę OUTPUT, która umożliwia zwrócenie informacji o dodawanych, usuwanych lub modyfikowanych wierszach podczas wykonywanej instrukcji DML (ang. *Data Manipulation Language*). Wyniki zwracane przez klauzulę OUTPUT można skierować do tabeli lub zmiennej tabelarycznej.

W celu zademonstrowania działania klauzuli OUTPUT, w przykładowej bazie danych ElektronicznyDziennikOcen została utworzona tabela o nazwie LogZmian, w której będziemy rejestrować wyniki działania poleceń modyfikacji danych wykonywanych dla tabeli Uczniowie. Strukturę tabeli LogZmian pokazano na rysunku 18.



	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Klucz	int	☐
	Operacja	char(1)	☐
	Data	datetime	☐
	KtoWykonal	varchar(256)	☐

Rysunek 18.
Struktura tabeli LogZmian

W poszczególnych kolumnach tabeli LogZmian będziemy zapisywać następujące wartości:
- kolumna Klucz – wartość klucza podstawowego modyfikowanego wiersza, czyli zawartość kolumny IdUcznia,
- kolumna Operacja – jednoliterowy skrót wykonywanej operacji (I-Insert, U-Update, D- Delete),
- kolumna Data – datę i czas wykonanej operacji,
- kolumna KtoWykonal – login użytkownika wykonującego operację.

W bazie danych ElektronicznyDziennikOcen wykonujemy polecenie INSERT zapisujące do tabeli Uczniowie nowy wiersz danych:

```
INSERT INTO Uczniowie
      (Nazwisko,Imie, DataUrodzenia, CzyChlopak, Pesel, idklasy)
OUTPUT Inserted.iducznia, 'I', GETDATE(), SYSTEM_USER     INTO LogZmian
  VALUES
      ('Bryda'
      ,'Ewelina'
      ,'1994-09-23'
      ,0
      ,'94092387645'
      ,3);
```

W klauzuli OUTPUT żądamy zapisania do tabeli LogZmian następujących danych:
- Inserted.iducznia – wartość klucza podstawowego nowego wiersza,
- 'I' – symbol operacji INSERT,
- GETDATE() – funkcja zwraca aktualny czas,
- SYSTEM_USER – funkcja zwraca login użytkownika.

Po wykonaniu polecenia, w tabeli LogZmian, zostanie zapisany wiersz danych, którego zawartość określiliśmy w klauzuli OUTPUT. Zawartość tabeli LogZmian, po wykonaniu polecenia INSERT, pokazano na rysunku 19.

Id	Klucz	Operacja	Data	KtoWykonal
1	611	I	2012-05-14 15:33:46.323	APProd\andrzej

Rysunek 19.
Zwartość tabeli LogZmian po wykonaniu polecenia INSERT

Wykonujemy teraz dwa kolejne polecenia, z których jedno modyfikuje wybrany wiersz w tabeli Uczniowie, a drugie usuwa wiersz o podanym numerze Pesel:

```
UPDATE Uczniowie
SET
idklasy=4
OUTPUT Inserted.iducznia, 'U', GETDATE(), SYSTEM_USER  INTO LogZmian
WHERE iducznia=7;
```

```
DELETE Uczniowie
OUTPUT Deleted.iducznia, 'D', GETDATE(), SYSTEM_USER  INTO LogZmian
WHERE Pesel='94092387645';
```

Każde z powyższych poleceń w klauzuli OUTPUT realizuje zapis do tabeli LogZmian. Zawartość tabeli LOg-Zmian, po wykonaniu powyższych poleceń, pokazano na rysunku 20.

Id	Klucz	Operacja	Data	KtoWykonał
1	611	I	2012-05-14 15:33:46.323	APProd\andrzej
2	7	U	2012-05-14 15:34:54.877	APProd\andrzej
3	611	D	2012-05-14 15:35:14.117	APProd\andrzej

Rysunek 20.
Zawartość tabeli LogZmian po wykonaniu poleceń Update i Delete

Jak można było zobaczyć w powyższych przykładach w klauzuli OUTPUT odwołujemy się do tabeli Inserted lub Deleted. SQL Server automatycznie zarządza tymi tabelami i udostępnia je podczas wykonywania wyzwalaczy oraz operacji z klauzulą OUTPUT. Zależnie od tego, dla jakiej tabeli wykonujemy zmiany danych, takie kolumny są dostępne w tabelach INSERTED i DELETED. Tabela INSERTED zawiera wiersze, które zostały dodane do tabeli, natomiast DELETED te, które zostały usunięte. W przypadku operacji zmiany danych tabela DELETED zawiera wartości przed zmianą, a tabela INSERTED wartości po zmianie.

2.2. POLECENIE MERGE.

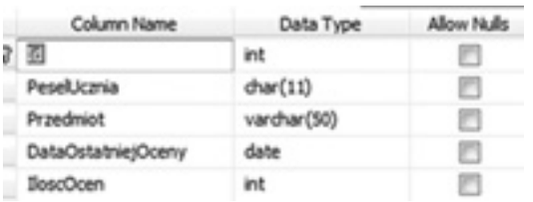
Polecenie MERGE jest nowym poleceniem języka SQL wprowadzonym w standardzie SQL – 2003. Polecenie MERGE można nazwać zbiorczą modyfikacją tabeli na podstawie zawartości innej tabeli. Ogólną postać składni polecenia można przedstawić następująco:

```
MERGE NazwaTabeliModyfikowanej
      USING NazwaTabeliŹrodlowej
      ON WarunekPolaczeniaTabel
WHEN MATCHED  (Ewentualne inne warunki) THEN (INSERT,UPDATE LUB DELETE)
WHEN NOT MATCHED BY TARGET (Ewentualne inne warunki) THEN (INSERT,UPDATE LUB DELETE)
WHEN NOT MATCHED BY SOURCE (Ewentualne inne warunki) THEN (INSERT,UPDATE LUB DELETE)
```

Na pierwszy rzut oka składnia ta wygląda na dość złożoną, ale po przeanalizowaniu konkretnego przykładu staje się jasna i dość czytelna. Działanie instrukcji MERGE można opisać w następujący sposób:

- Określenie tabeli, która podlega modyfikacji
- Określenie tabeli, na podstawie której wykonywana jest modyfikacja
- Podanie warunku połączenia tabel
- Dowolną ilość razy podajemy warunek, dla którego podejmowana jest odpowiednia akcja (INSERT, UPDATE lub DELETE) – warunek składa się z jednego z trzech określeń wyjściowych:
  - WHEN MATCHED (dla tych wierszy, dla których został spełniony warunek połączenia tabel)
  - WHEN NOT MATCHED (BY TARGET lub BY SOURCE) – dla wierszy, które nie spełniły warunku połączenia, bo brak odpowiednika w tabeli modyfikowanej (BY TARGET) lub tabeli źródłowej (BY SOURCE)
  - Określenie wyjściowe możemy łączyć z dowolnymi innymi wyrażeniami, dzięki czemu można, w zależności od logiki zadania, uzależniać wykonywaną akcję od różnych warunków logicznych.

W celu lepszego zrozumienia istoty polecenia MERGE przeanalizujemy przykład jego wykorzystania. W bazie danych jest tabela o nazwie KartaUcznia, której strukturę pokazano na rysunku 21. W tabeli tej przechowywane są dane opisujące liczbę ocen otrzymanych z konkretnych przedmiotów przez uczniów (kolumna PeselUcznia).



Column Name	Data Type	Allow Nulls
Id	int	
PeselUcznia	char(11)	
Przedmiot	varchar(50)	
DataOstatniejOceny	date	
IloscOcen	int	

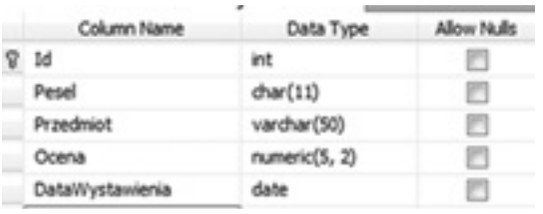
Rysunek 21.
Struktura tabeli KartaUcznia

Przykładową zawartość tabeli KartaUcznia pokazano na rysunku 22.

Id	PeselUcznia	Przedmiot	DataOstatniejOceny	IloscOcen
1	91031199123	Fizyka	2009-01-13	10
2	92022277654	Fizyka	2009-02-10	19
3	92031275446	Fizyka	2009-02-07	19
17	91031199123	Geografia	2009-02-08	17
18	92022277654	Geografia	2009-02-08	11
19	92031275446	Geografia	2009-02-07	15
20	92051577646	Geografia	2009-01-28	6

Rysunek 22.
Przykładowa zawartość tabeli KartaUcznia

Zakładamy, że okresowo otrzymujemy dane w tabeli o nazwie WynikZaliczenia. Strukturę tabeli WynikZaliczenia pokazano na rysunku 23.



Column Name	Data Type	Allow Nulls
Id	int	
Pesel	char(11)	
Przedmiot	varchar(50)	
Ocena	numeric(5, 2)	
DataWystawienia	date	

Rysunek 23.
Struktura tabeli WynikZaliczenia

Dane z tabeli WynikZaliczenia będą służyły do aktualizacji tabeli KartaUcznia. Modyfikacja danych w tabeli KartaUcznia ma być realizowana według następujących zasad:

- Dla każdego wiersza w tabeli WynikZaliczenia realizujemy następujące zadanie:
  - Jeżeli dla danego ucznia jest w tabeli KartaUcznia wiersz opisujący ilość ocen otrzymanych z danego przedmiotu to zwiększamy zawartość kolumny IloscOcen o 1 oraz modyfikujemy kolumnę DataOstatniejOceny
  - Jeżeli dla danego ucznia nie istnieje w tabeli KartaUcznia wiersz opisujący ilości ocen otrzymanych z danego przedmiotu to należy dopisać nowy wiersz do tabeli KartaUcznia.

Z analizy zadania wynika, że na podstawie zawartości tabeli WynikZaliczenia, dla każdego wiersza tej tabeli musimy wykonać polecenie UPDATE lub INSERT w tabeli KartaUcznia. Przykładową zawartość tabeli WynikZaliczenia pokazano na rysunku 24.

Id	Pesel	Przedmiot	Ocena	DataWystawienia
1	91031199123	Literatura	4.00	2011-11-23
2	91031199123	Fizyka	5.00	2011-11-24
3	91071777123	Matematyka	4.00	2011-11-24
4	92022277654	Geografia	5.00	2011-11-25

Rysunek 24.
Przykładowa zawartość tabeli WynikZaliczenia

Opisane zadanie możemy zrealizować za pomocą następującego polecenia MERGE:

```
MERGE KartaOcen
USING WynikZaliczenia AS WZ
ON KartaOcen.PeselUcznia=WZ.Pesel
    AND KartaOcen.Przedmiot=WZ.Przedmiot
WHEN MATCHED     THEN UPDATE
                        SET DataOstatniejOceny=WZ.DataWystawienia,
                            IloscOcen+=1
WHEN  NOT MATCHED BY TARGET THEN
      INSERT(PeselUcznia,Przedmiot, DataOstatniejOceny,IloscOcen)
      VALUES(  WZ.Pesel, WZ.Przedmiot,WZ.DataWystawienia, 1)  ;
```

Wykonanie podanego polecenia wprowadzi niezbędne zmiany w tabeli KartaUcznia w zależności od zawartości tabeli WynikZaliczenia. Przykładową zawartość tabeli KartaUcznia, po wykonaniu polecenia MERGE pokazano na rysunku 25.

Id	PeselUcznia	Przedmiot	DataOstatniejOceny	IloscOcen
1	91031199123	Fizyka	2011-11-24	11
2	92022277654	Fizyka	2009-02-10	19
3	92031275446	Fizyka	2009-02-07	19
17	91031199123	Geografia	2009-02-08	17
18	92022277654	Geografia	2011-11-25	12
19	92031275446	Geografia	2009-02-07	15
20	92051577646	Geografia	2009-01-28	6
21	91031199123	Literatura	2011-11-23	1
22	91071777123	Matematyka	2011-11-24	1

Rysunek 25.
Zawartość tabeli KartaUcznia po wykonaniu polecenia MERGE

Analizując zawartość tabeli KartaUcznia, po wykonaniu polecenia możemy zauważyć, że zostały dopisane dwa wiersze i dwa zostały zmodyfikowane.

2.3. WYRAŻENIA CTE.

Wyrażenia CTE (ang. *Common Table Expression*) są nowym elementem wprowadzonym do standardu języka SQL. Podstawowym celem wprowadzenia wyrażeń CTE było umożliwienie pisania zapytań rekurencyjnych. Dodatkowym zastosowaniem tych wyrażeń jest tworzenie nazwanych zbiorów danych, które mogą być wykorzystywane do realizacji jednego z poleceń manipulacji danymi (SELECT, INSERT, UPDATE, DELETE, MERGE). Nazwane zbiory tworzone są dla konkretnego polecenia i przestają istnieć po jego wykonaniu. Można uznać definiowanie wyrażeń CTE jako prolog do konkretnego polecenia języka SQL. Podstawowa składnia tworzenia wyrażeń CTE jest następująca:

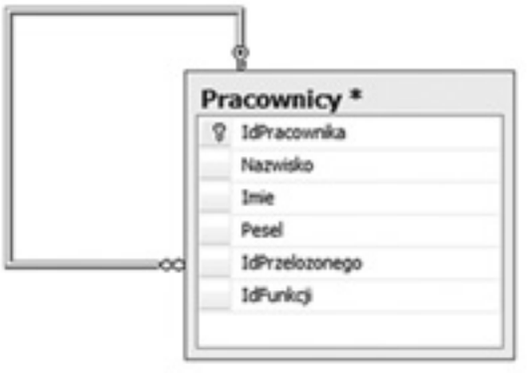
```
WITH Nazwa_zbioru AS
( Polecenie_Select )
Polecenie SQL
```

Należy dodać, że w jednym prologu polecenia można definiować dowolną liczbę nazwanych zbiorów, dzięki czemu możemy odpowiednio przygotować i przetworzyć dane przed wykonaniem konkretnego zadania. Wyrażenia CTE stosujemy w celu:

- czytelnego zapisu złożonych zadań w ramach jednego polecenia SQL,
- wykonania złożonych schematów przetwarzania danych,
- realizacji zapytań rekurencyjnych.

Podstawowym celem wprowadzenia wyrażeń CTE do standardu języka SQL jest umożliwienie realizacji zapytań rekurencyjnych, ponieważ wcześniej takiej możliwości nie było. Tworzenie zapytania rekurencyjnego omówimy na bazie przykładu.

W celu odwzorowania hierarchii, w relacyjnych bazach danych stosuje się klucze obce, które odwołują się do innego wiersza w tej samej tabeli. Przykład takiego rozwiązania pokazano na rysunku 26. W tabeli Pracownicy umieszczony jest klucz obcy o nazwie idprzelozonego, który wskazuje na inny wiersz w tej samej tabeli (przełożony jest tez pracownikiem i ma swojego przełożonego).Takie konstrukcje tabel stwarzają problemy przy zapytaniach, których istotą jest wybranie danych powiązanych z konkretnym wierszem na wszystkich poziomach hierarchii (wszyscy podwładni konkretnej osoby na wszystkich poziomach podległości). Ponieważ problem przeglądania hierarchii jest w swej istocie problemem rekurencyjnym, to jego rozwiązanie wymaga również rekurencyjnego podejścia. Klasyczne polecenie SELECT języka SQL nie daje możliwości realizowania zapytań rekurencyjnych. Wyrażenia CTE pozwalają na definiowanie takiego zbioru nazwanego, który zapewni nam rozwiązanie zadań rekurencyjnych.



Rysunek 26.
Struktura tabeli Pracownicy

Chcemy wybrać wszystkich podwładnych (na dowolnym poziomie) konkretnego pracownika.

Odwołując się do tabel pokazanych na rysunku 26, możemy przystąpić do realizacji zadania. Dla lepszego zrozumienia przykładu możemy założyć zawartość tabeli Pracownicy, jak pokazano na rysunku 27.

IdPracownika	Nazwisko	Imie	Pesel	IdPrzelozonego	IdFunkcji
1	Kot	Jan	87090812321	1	1
2	Pies	Janina	81110987654	1	2
3	Okoń	Stanisław	76120785609	1	2
4	Płotka	Zofia	59122188723	2	3
5	Leszcz	Piotr	76120976781	2	3
6	Szczupak	Maria	82061209834	4	4
7	Lin	Karol	87062187654	4	4
8	Myszka	Szara	44091187654	6	5
9	Szczurek	Paweł	69121189876	6	5
10	Królik	Violetta	90111298723	8	6

Rysunek 27.
Przykładowa zawartość tabeli Pracownicy

Polecenie, które dla przykładowych danych zwróci dane wszystkich podwładnych (na wszystkich poziomach) pracownika o IdPracownika=1 może mieć następującą postać:

```
WITH Podwladni AS
(
```

```
  SELECT Imie+' '+Nazwisko as Pracownik,
         0 as Poziom,
         idpracownika
  FROM Pracownicy
  WHERE IdPracownika=1
  UNION ALL
  SELECT  Pr.Imie+' '+Pr.Nazwisko as Pracownik ,
          Po.Poziom+1 as Poziom,
          Pr.idpracownika
  FROM Pracownicy Pr JOIN Podwladni Po
     ON Pr.IdPrzelozonego=Po.idpracownika
  WHERE Pr.idPrzelozonego !=Pr.IdPracownika
)
SELECT * FROM Podwladni
```

Wynik tego zapytania dla przyładowych danych z rysunku 27 pokazono na rysunku 28.

Pracownik	Poziom	idpracownika
Jan Kot	0	1
Janina Pies	1	2
Stanisław Okoń	1	3
Zofia Płotka	2	4
Piotr Leszcz	2	5
Maria Szczupak	3	6
Karol Lin	3	7
Szara Myszka	4	8
Paweł Szczurek	4	9
Violetta Królik	5	10

Rysunek 28.
Wynik zapytania rekurencyjnego

W omawianym zapytaniu utworzyliśmy kolumnę o nazwie Poziom, która określa „odległość" w hierarchii danego pracownika od tego, który jest początkowym elementem hierarchii.

2.4. ĆWICZENIA

2.4.1. Ćwiczenie 2 – Wykorzystanie klauzuli OUTPUT w poleceniach INSERT, UPDATE i DELETE.

W ramach ćwiczenia należy zaprojektować tabelę, w której będą zapisywane wyniki modyfikacji danych w tabeli Nauczyciele w bazie danych ElektronicznyDziennikOcen. W zaprojektowanej tabeli powinna być zapisana:

- zawartość modyfikowanego wiersza,
- data tej modyfikacji,
- rodzaj wykonanego polecenia,
- login użytkownika wykonującego modyfikację.

Po utworzeniu tabeli należy napisać i wykonać kilka poleceń INSERT, UPDATE i DELETE, które w klauzuli OUTPUT będą realizowały zapisywanie wyników modyfikacji w zaprojektowanej tabeli.

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

2.4.2. Ćwiczenie 3 – Modyfikacja wykazu uczniów na podstawie tabeli aktualizującej dane.

W ramach ćwiczenia należy wykonać modyfikację tabeli o nazwie ListyUczniow (przykładową zawartość tabeli pokazano na rysunku 29) na podstawie danych w tabeli Aktualizacja.

Nazwisko	imie	Pesel	DataUrodzenia	Klasa	OstAktualizacja
Piesek	Jan	92051587746	1992-05-15	Ia	2012-05-15 08:26:24.357
Gąska	Wacek	91031199123	1991-03-11	Ia	2012-05-15 08:26:24.357
Krówka	Rysio	92051577646	1992-05-15	Ia	2012-05-15 08:26:24.357
Sarenka	Rysio	92121278766	1992-12-12	Ia	2012-05-15 08:26:24.357
Miś	Wacek	93031199123	1993-03-11	Ia	2012-05-15 08:26:24.357
Nowak	Piotr	92091298795	1992-09-12	Ia	2012-05-15 08:26:24.357
Kotek	Katarzyna	92031275446	1992-03-12	IIa	2012-05-15 08:26:24.357
Lisek	Kasia	92022277654	1992-02-22	IIa	2012-05-15 08:26:24.357
Kurka	Jola	92060288788	1992-06-02	IIa	2012-05-15 08:26:24.357
Gazela	Basia	92111177446	1992-11-11	IIa	2012-05-15 08:26:24.357
Konik	Kasia	93031275446	1993-03-12	IIa	2012-05-15 08:26:24.357
Ryba	Jan	93051587746	1993-05-15	IIa	2012-05-15 08:26:24.357
Kura	Kasia	93022277654	1993-02-22	IIa	2012-05-15 08:26:24.357
Łoś	Jola	93060288788	1993-06-02	IIa	2012-05-15 08:26:24.357
Okoń	Rysio	93051577646	1993-05-15	IIa	2012-05-15 08:26:24.357

Rysunek 29.
Przykładowa zawartość tabeli ListyUczniow

Na rysunku 30 pokazano przykładową zawartość tabeli Aktualizacja.

Nazwisko	imie	Pesel	DataUrodzenia	Klasa
Czwartek	Viktor	87021276598	1987-02-12	IIc
Rak	Wojciech	91010123432	1991-01-01	IIc
Sum	Wacek	91031134565	1991-03-11	IIc
Gąska	Wacek	91031199123	1991-03-11	Ia
Lisek	Kasia	92022277654	1992-02-22	IIa
Orka	Kasia	92022288776	1992-02-22	IIc
Antylopa	Kasia	92031254567	1992-03-12	IIc
Kotek	Katarzyna	92031275446	1992-03-12	IIa
Piesek	Jan	92051587746	1992-05-15	Ia
Krówka	Rysio	92051577646	1992-05-15	Ia
Rekin	Jan	92051555432	1992-05-15	IIc
Foka	Jola	92060223454	1992-06-02	Ib
Kurka	Jola	92060288788	1992-06-02	IIa
Nowak	Piotr	92091298795	1992-09-12	Ia
Gazela	Basia	92111177446	1992-11-11	IIa

Rysunek 30.
Przykładowa zawartość tabeli Aktualizacja

Modyfikację tabeli ListyUczniow należy zrealizować według następujących zasad:

- Jeżeli dane ucznia znajdują się w tabeli Aktualizacja, a nie ma ich w tabeli ListyUczniow to należy dopisać te dane (wyróżnikiem danych ucznia jest numer Pesel). W kolumnie OstAktualizacja należy zapisać czas wykonania operacji,
- Jeżeli dane ucznia znajdują się u obu tabelach, to należy zmodyfikować odpowiedni wiersz w tabeli ListyUczniow na podstawie danych tego ucznia z tabeli Aktualizacja. W kolumnie OstAktualizacja należy zapisać czas wykonania modyfikacji,
- Jeżeli dane ucznia znajdują się w tabeli ListyUczniow, a nie ma ich w tabeli Aktualizacja to należy usunąć dane ucznia z tabeli ListyUczniow.

Zadanie zrealizować za pomocą polecenia MERGE.

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

3 ZAAWANSOWANE TECHNIKI REALIZACJI ZAPYTAŃ

3.1. FAZY PRZETWARZANIA ZAPYTAŃ.

W celu zrozumienia niektórych aspektów działania polecenia SELECT warto się chwilę zastanowić nad sposobem przetwarzania zapytań. Przedstawiona poniżej sekwencja działania jest dość umowna, ale pozwala wyjaśnić niektóre problemy związane z oceną działania polecenia SELECT.

Fazy procesu logicznego przetwarzania zapytań:

- Wyznaczanie iloczynu kartezjańskiego (połączenia skrośnego) – faza realizacji połączenia tabel
- Zastosowanie filtru klauzuli ON (warunku łączenia) – wydzielenie wierszy spełniających warunek połączenia
- Dodawanie wierszy zewnętrznych – w przypadku połączenia zewnętrznego (OUTER JOIN z opcją LEFT, RIGHT lub FULL)
- Zastosowanie filtru klauzuli WHERE – wybór wierszy spełniających warunek filtru WHERE
- Grupowanie – operacja grupowania danych przy wykorzystaniu funkcji agregujacych
- Zastosowanie opcji CUBE lub ROLLUP – operatory CUBE, ROLLUP lub GROUPING SET dodają do wyniku zapytania dodatkowe agregacje
- Zastosowanie filtru klauzuli HAVING – opóźniony warunek selekcji stosowany dla warunków odnoszących się do funkcji agregujących
- Przetwarzanie listy SELECT – obliczanie wyrażeń i nadawanie kolumną nazw zastępczych
- Zastosowanie klauzuli DISTINCT – eliminacja powtarzających się wierszy w wyniku zapytania
- Zastosowanie klauzuli ORDER BY – porządkowanie wyniku
- Zastosowanie klauzuli TOP – przekazanie do wyniku zapytania określonej ilości wierszy

Podstawowe wnioski wynikające z faz logicznego przetwarzania zapytań dotyczą stosowania filtrów klauzuli ON i WHERE. Jak widać z powyższego wyliczenia, w przypadku złączenia wewnętrznego, filtry ON i WHERE stosowane są praktycznie w tej samej fazie, a to znaczy, że nie ma większego znaczenia, w którym z tych filtrów umieścimy warunki selekcji wierszy. W przypadku złączenia wewnętrznego pomiędzy stosowaniem filtrów występuje faza dodawania wierszy zewnętrznych i w tym przypadku każdy filtr działa na innym zestawie danych. Problemy, które mogą wystąpić zilustrujemy przykładem. Wykonujemy zapytanie skierowane do bazy danych ElektronicznyDziennikOcen, w którym chcemy otrzymać nazwę klasy oraz kolumnę zawierającą liczbę uczniów z danej klasy, u których w nazwisku występuje litera a.

Zapytanie może mieć następującą postać:

```
SELECT Klasy.Nazwa, COUNT(*) as IluUczniow
FROM Klasy  JOIN Uczniowie
    ON Klasy.idklasy=Uczniowie.idklasy
WHERE Nazwisko like ‚%a’
GROUP BY Klasy.Nazwa
```

Wynik tego zapytania, dla przykładowej bazy danych pokazano na rysunku 31.

Nazwa	IluUczniow
Ia	4
IIa	7
IIc	3

Rysunek 31.
Wynik zapytania – złaczenie wewnętrzne

Rozszerzamy logikę naszego zadania, dodając warunek, w wyniku którego chcemy mieć dane wszystkich klas (w niektórych klasach może nie być uczniów z literą a w nazwisku lub brak uczniów dowiązanych do klasy)

z informacją o liczbie uczniów z literą a w nazwisku. Zastosujemy operator złączenia zewnętrznego LEFT JOIN, zapytanie po zmianie będzie miało postać:

```
SELECT Klasy.Nazwa, COUNT(*) as IluUczniow
FROM Klasy LEFT JOIN Uczniowie
    ON Klasy.idklasy=Uczniowie.idklasy
WHERE Nazwisko like ‚%a'
GROUP BY Klasy.Nazwa
```

W takiej postaci zapytania, jego wynik nie powinien ulec zmianie co pokazano na rysunku 32.

Nazwa	IluUczniow
Ia	4
IIa	7
IIc	3

Rysunek 32.
Wynik zapytania – złączenie zewnetrzne z filtrem WHERE

Na pierwszy rzut oka nie został zrealizowany operator złączenia LEFT JOIN, który miał zapewnić, że w wyniku zapytania znajdą się wszystkie klasy niezależnie od spełnienia warunku złączenia. Odpowiedzialna za taki wynik jest kolejność przetwarzania zapytań, ponieważ po zadziałaniu filtru klauzuli ON zostały dołączone wiersze z tabeli klasy, które nie spełniły warunku złączenia, ale po tym zastosowany został filtr klauzuli WHERE i zostały one usunięte z wyniku zapytania. W tej sytuacji warunek filtru WHERE należało dołączyć do filtru ON, tak jak zaprezentowano poniżej:

```
SELECT Klasy.Nazwa, COUNT(*) as IluUczniow
FROM Klasy LEFT JOIN Uczniowie
    ON Klasy.idklasy=Uczniowie.idklasy
      AND Nazwisko like ‚%a'
GROUP BY Klasy.Nazwa
```

Wynik tego zapytania, skierowany do przykładowej bazy danych, pokazano na rysunku 33.

Nazwa	IluUczniow
Ia	5
Ib	1
IIa	7
IIb	6
IIc	3
IIIa	1
IX D	1
V F	1
VII B	1
VIIC	1
VIII G	1

Rysunek 33.
Wynik zapytania – złączenie zewnętrzene z rozszerzenie klauzuli ON

Jest tu jeszcze pewna niedokładność, gdyż klasy do których brak dowiązanych uczniów wykazują w zestawieniu w kolumnie IluUczniow wartość 1. Wynika to z faktu, że istnieje jeden wiersz, w wyniku zapytania, dla klas bez dowiązanych uczniów. Problem można rozwiązać poprzez dodanie argumentu funkcji agregującej COUNT. Po zmianie zapytanie będzie miało następującą postać:

```
SELECT Klasy.Nazwa, COUNT(nazwisko) as IluUczniow
FROM Klasy LEFT JOIN Uczniowie
    ON Klasy.idklasy=Uczniowie.idklasy
      AND Nazwisko like ‚%a'
GROUP BY Klasy.Nazwa
ORDER BY IluUczniow DESC
```

Wynik zapytania skierowany do przykładowej bazy danych pokazano na rysunku 34.

Nazwa	IluUczniow
IIa	7
Ia	4
IIc	3
IIIa	0
IX D	0
V F	0
VII B	0
VIIC	0
VIII G	0
Ib	0
IIb	0

Rysunek 34.
Wynik zapytania po modyfikacji funkcji COUNT()

Funkcja agregująca COUNT(nazwisko) – liczy wiersze, które w kolumnie nazwisko nie mają wartości null.

Omówiony przykład pokazuje, że przy stosowaniu złączenia zewnętrznego należy zastanowić się, w zależności od tego, jaki wynik chcemy uzyskać, na odpowiednim umieszczeniem warunków w klauzulach ON i WHERE.

Zrozumienie faz logicznego przetwarzania wyjaśnia, dlaczego do nazw zastępczych kolumn (aliasów) można się odwoływać w obrębie danego zapytania jedynie w klauzuli ORDER BY.

3.2. ŁĄCZENIE TABEL.

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%'
```

Wynik tego zapytania skierowany do bazy danych ElektronicznyDziennikOcen pokazano na rysunku 35.

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 35.
Wynik zapytania z wykorzystaniem operatora INNER JOIN

- 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%’
```

Wynik tego zapytania (w dołączonych wierszach kolumny odpowiadające tabeli Uczniowie, przyjmują wartość null) pokazano na rysunku 36.

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 36.
Wynik zapytania z wykorzystaniem operatora LEFT JOIN

- Operator złączenia tabeli przestawnej – PIVOT – nowy rodzaj operatora złączeń wprowadzony w MS SQL Server 2005, umożliwiający tworzenie tabeli przestawnej. Przykład zapytania tworzącego tabelę przestawną:

```
WITH TabelaPlaska AS
(
  SELECT Klasy.Nazwa as Klasa,
         Przedmioty.Nazwa as Przedmiot,
         Oceny.Ocena
```

```
        FROM Klasy JOIN Uczniowie ON Klasy.idklasy=Uczniowie.idklasy
                JOIN Oceny ON Oceny.iducznia=Uczniowie.iducznia
                JOIN Przedmioty ON Przedmioty.idprzedmiotu=Oceny.idprzedmiotu
)
SELECT *
FROM TabelaPlaska
PIVOT (AVG(Ocena) FOR Przedmiot in ([Fizyka],[Matematyka],[ Informatyka],[Geografia] ))   as TMP
```

Wynik tego zapytania dla przykładowych danych pokazano na rysunku 37.

Klasa	Fizyka	Matematyka	Informatyka	Geografia
Ia	2.746987	2.919354	2.881578	2.918032
Ib	NULL	NULL	4.000000	NULL
IIa	2.960227	3.060439	2.848314	3.021978
IIc	NULL	NULL	NULL	4.333333

Rysunek 37.
Wynik zapytania wykorzystującego operator PIVOT

W powyższym przykładzie wyrażenie CTE buduje tabelę, która zostaje przekształcona w tabelę przestawną. Fragment tabeli zwracanej przez wyrażenie CTE ma postać pokazana na rysunku 38.

Klasa	Przedmiot	Ocena
IIa	Fizyka	1.00
Ia	Geografia	1.00
IIa	Geografia	2.00
Ia	Matematyka	2.00
IIa	Informatyka	2.00
Ia	Matematyka	5.00
IIa	Fizyka	3.00
Ia	Matematyka	2.00
IIa	Matematyka	3.00
IIa	Geografia	2.00
IIa	Geografia	2.00
IIa	Geografia	2.00
Ia	Informatyka	5.00
IIa	Geografia	5.00
IIa	Matematyka	3.00

Rysunek 38.
Wynik zwracany przez wyrażenie CTE.

Teraz można lepiej zrozumieć działanie operatora PIVOT, czyli wyrażenia:

```
(AVG(Ocena) FOR Przedmiot in ([Fizyka],[Matematyka],[ Informatyka],[Geografia] ))
```

Dla kolumny Ocena zastosowana zostaje funkcja agregująca AVG, wyspecyfikowane wartości z kolumny przedmiot zostają zamienione na kolumny, kolumny pozostałe tworzą zawartość każdego wiersza wyniku zapytania.

- Operator złączenia dynamicznego – APPLY – występuje z modyfikatorem OUTER lub CROSS. Operator złączenia APPLY wprowadzony został w MS SQL Server 2005. Ogólną postać operatora APPLY można przedstawić następująco:
  TabelaWejsciowa CROSS APPLY TabelaDynamiczna

Istotą działania operatora APPLY jest to, że dla każdego wiersza tabeli TabelaWejsciowa tworzona jest dynamicznie inna postać tabeli TabelaDynmiczna i do wyniku zapytania dołączana jest kombinacja wiersza z tabeli TabelaWejsciowa z każdym wierszem tabeli TabelaDynamiczna.

W sytuacji gdy dla danego wiersza tabeli TabelaWejsciowa, Tabela Dynamiczna nie zwraca żadnego wiersza, to wynik zależy od modyfikatora; dla CROSS APPLY – do wyniku zapytania nie jest dołączany żaden wiersz, a w przypadku OUTER APPLY (odpowiednik złączenia zewnętrznego) dołączany jest jeden wiersz, który w kolumnach odpowiadających tabeli TabelaDynamiczna ma wartości null.

Przykład zapytania wykorzystującego operator APPLY;

```
SELECT Uczniowie.Nazwisko +' ,+Uczniowie.Imie as Uczen,
        Dyn.Przedmiot,
        Dyn.Srednia
FROM Uczniowie CROSS APPLY
                (
                        SELECT Przedmioty.Nazwa as Przedmiot,
                                AVG(Ocena) as Srednia
                        FROM Oceny JOIN Przedmioty
                                ON Oceny.idprzedmiotu=Przedmioty.idprzedmiotu
                        WHERE iducznia=Uczniowie.iducznia
                        GROUP BY Przedmioty.Nazwa
                ) AS Dyn
        ORDER BY Uczen
```

Wynik tego zapytania dla przykładowych danych pokazano na rysunku 39.

Uczen	Przedmiot	Srednia
Foka Jola	Geografia	4.333333
Gazela Basia	Geografia	2.714285
Gazela Basia	Fizyka	3.666666
Gazela Basia	Informatyka	2.611111
Gazela Basia	Literatura	3.083333
Gazela Basia	Matematyka	3.071428
Gąska Wacek	Matematyka	2.750000
Gąska Wacek	Informatyka	2.538461
Gąska Wacek	Fizyka	3.400000
Gąska Wacek	Geografia	2.882352
Konik Kasia	Geografia	3.250000
Konik Kasia	Fizyka	2.615384
Konik Kasia	Informatyka	2.181818
Konik Kasia	Matematyka	2.789473
Kotek Katarzyna	Matematyka	3.166666

Rysunek 39.
Wynik zapytania z wykorzystaniem operatora CROSS APPLY

To samo zapytanie z operatorem OUTER APPLY:

```
SELECT Uczniowie.Nazwisko +' ,+Uczniowie.Imie as Uczen,
        Dyn.Przedmiot,
        Dyn.Srednia
FROM Uczniowie OUTER APPLY
                (
                        SELECT Przedmioty.Nazwa as Przedmiot,
                                AVG(Ocena) as Srednia
```

```
                    FROM Oceny JOIN Przedmioty
                        ON Oceny.idprzedmiotu=Przedmioty.idprzedmiotu
                    WHERE iducznia=Uczniowie.iducznia
                    GROUP BY Przedmioty.Nazwa
                ) AS Dyn
        ORDER BY Uczen
```

Wynik zawiera dane także tych uczniów, którzy nie mają wystawionej żadnej oceny, przykładowy wynik zapytania pokazano na rysunku 40.

Uczen	Przedmiot	Srednia
Sum Wacek	NULL	NULL
Rak Wojciech	NULL	NULL
Stokrotka Rysio	NULL	NULL
Orka Kasia	NULL	NULL
Nowak Piotr	NULL	NULL
Antylopa Kasia	NULL	NULL
Rekin Jan	NULL	NULL
Kotek Katarzyna	Fizyka	2.578947
Piesek Jan	Fizyka	2.600000
Lisek Kasia	Fizyka	3.473684
Kurka Jola	Fizyka	2.769230
Gąska Wacek	Fizyka	3.400000
Krówka Rysio	Fizyka	2.636363
Zebra Wojtek	Fizyka	2.888888
Gazela Basia	Fizyka	3.666666

Rysunek 40.
Wynik zapytania z wykorzystaniem operatora OUTER APPLY

Operator APPLY jest często wykorzystywany w zapytaniach łączących dane relacyjne z danymi zapisanymi w kolumnach typu XML.

3.3. 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 liczbę wierszy
AVG	Oblicza średnią arytmetyczną
SUM	Oblicza sumę wyrażenia dla wyniku zapytania
MIN	Określa minimalną wartość wyrażenia
MAX	Określa maksymalną wartość wyrażenia
STDEV	Oblicza odchylenie standardowe
VAR	Oblicza wariancję

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 pokazano na rysunku 41.

IluUczniow
14

Rysunek 41.
Wynik zapytania wykorzystującego funkcję COUNT().

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 42.

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

Rysunek 42.
Wynik zapytania z wykorzystaniem 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 43.

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
Płotka	Wojtek	93030399846	2.428571
Różyczka	Basia	93111177446	2.687500
Ryba	Jan	93051587746	2.875000

Rysunek 43.
Wynik zapytania z wykorzystaniem klauzuli GROUP BY

Bez klauzuli GROUP BY próba wykonania zapytania zakończyłaby się błędem:

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 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
HAVING AVG(Ocena) >3.00
```

Wynik tego zapytania dla przykładowej bazy danych pokazano na rysunku 44.

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 44.
Wynik zapytania z wykorzystaniem klauzuli HAVING

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

3.4. OPERATORY DZIAŁANIA NA ZBIORACH.

Relacyjny model danych oparty jest na teorii zbiorów i dlatego nie powinna nikogo dziwić obecność w języku SQL operatorów sumy, różnicy i iloczynu zbiorów. Ogólna postać zapytania wykorzystującego te operatory ma postać:

```
Zapytanie_1
  Operator
Zapytanie_2
```

Widać, że operator działa dla dwóch zapytań. Zapytania powinny mieć taką samą strukturę, czyli zawierać taką samą liczbę kolumn tego samego typu.

W języku SQL dostępne są trzy operatory działania na zbiorach:

- UNION lub UNION ALL – suma zbiorów z eliminowaniem powtarzających się wierszy (z opcją ALL – bezwzględna suma zbiorów)
- EXCEPT – różnica zbiorów
- INTERSECT – iloczyn (część wspólna) zbiorów

Działanie na zbiorach zademonstrujemy na przykładzie. Utworzymy dwa zbiory o nazwach Matematyka i Fizyka, które zawierają dane trzech uczniów o najwyższych średnich ocenach odpowiednio z fizyki i matematyki. Zapytanie tworzące zbiór Matematyka ma postać:

```
SELECT TOP 3 Nazwisko,
             Imie,
             Pesel,
             AVG(ocena) as Srednia
FROM Uczniowie JOIN Oceny ON Uczniowie.iducznia=Oceny.iducznia
               JOIN Przedmioty ON Oceny.idprzedmiotu=Przedmioty.idprzedmiotu
WHERE Przedmioty.Nazwa='Matematyka'
GROUP BY Nazwisko,
            Imie,
            Pesel
 ORDER BY Srednia DESC
```

Dla przykładowej bazy danych otrzymujemy wynik, który pokazano na rysunku 45.

Nazwisko	Imie	Pesel	Srednia
Piesek	Jan	92051587746	3.727272
Ryba	Jan	93051587746	3.647058
Kura	Kasia	93022277654	3.277777

Rysunek 45.
Wynik zapytania (najlepsi z matematyki)

Zapytanie tworzące zbiór Fizyka ma postać:

```
SELECT TOP 3 Nazwisko,
             Imie,
             Pesel,
             AVG(ocena) as Srednia
FROM Uczniowie JOIN Oceny ON Uczniowie.iducznia=Oceny.iducznia
               JOIN Przedmioty ON Oceny.idprzedmiotu=Przedmioty.idprzedmiotu
WHERE Przedmioty.Nazwa='Fizyka'
GROUP BY Nazwisko,
            Imie,
            Pesel
 ORDER BY Srednia DESC
```

Dla przykładowej bazy danych otrzymujemy wynik, który pokazano na rysunku 46.

Nazwisko	Imie	Pesel	Srednia
Gazela	Basia	92111177446	3.666666
Łoś	Jola	93060288788	3.642857
Kura	Kasia	93022277654	3.571428

Rysunek 46.
Wynik zapytania (najlepsi z fizyki)

Główne zapytanie zdefiniuje zbiory Fizyka i Matematyka jako wyrażenie CTE, a następnie zdefiniuje odpowiednie działanie na zbiorach.

- Operator sumy zbiorów UNION

Postać zapytania:

```
WITH Fizyka AS
(
        ... Zapytanie tworzące zbiór Fizyka

 ), Matematyka AS
 (
        ... Zapytanie tworzące zbiór Matematyka
 )
SELECT Nazwisko, Imie, Pesel
FROM Fizyka
  UNION
SELECT   Nazwisko, Imie, Pesel
FROM Matematyka
```

Wynik zapytania pokazano na rysunku 47.

Nazwisko	Imie	Pesel
Gazela	Basia	92111177446
Kura	Kasia	93022277654
Łoś	Jola	93060288788
Piesek	Jan	92051587746
Ryba	Jan	93051587746

Rysunek 47.
Wynik zapytania z wykorzystaniem operatora UNION

Uwaga: W wyniku usunięto duplikat – w obu zbiorach występował uczeń o nazwisku Kura Kasia.

- Operator bezwzględnej sumy zbiorów UNION ALL

Postać zapytania:

```
WITH Fizyka AS
(
        ... Zapytanie tworzące zbiór Fizyka

 ), Matematyka AS
 (
        ... Zapytanie tworzące zbiór Matematyka
 )
SELECT Nazwisko, Imie, Pesel
FROM Fizyka
  UNION ALL
SELECT   Nazwisko, Imie, Pesel
FROM Matematyka
```

Wynik pokazano na rysunku 48.

Nazwisko	Imie	Pesel
Gazela	Basia	92111177446
Łoś	Jola	93060288788
Kura	Kasia	93022277654
Piesek	Jan	92051587746
Ryba	Jan	93051587746
Kura	Kasia	93022277654

Rysunek 48.
Wynik zapytania z wykorzystaniem operatora UNION ALL

Uwaga: W wyniku dwukrotnie występuje uczeń o nazwisku Kura Kasia.

- Operator różnicy zbiorów EXCEPT

Postać zapytania:

```
WITH Fizyka AS
(
        ... Zapytanie tworzące zbiór Fizyka

 ), Matematyka AS
 (
        ... Zapytanie tworzące zbiór Matematyka
 )
SELECT Nazwisko, Imie, Pesel
FROM Fizyka
  EXCEPT
SELECT   Nazwisko, Imie, Pesel
FROM Matematyka
```

Wynik zapytania pokazano na rysunku 49.

Nazwisko	Imie	Pesel
Gazela	Basia	92111177446
Łoś	Jola	93060288788

Rysunek 49.
Wynik zapytania z wykorzystaniem operatora EXCEPT

Uwaga: W wyniku nie występuje uczeń o nazwisku Kura Kasia, ponieważ był obecny w obu zbiorach.

- Operator iloczynu zbiorów INTERSECT

Postać zapytania:

```
WITH Fizyka AS
(
        ... Zapytanie tworzące zbiór Fizyka

 ), Matematyka AS
 (
        ... Zapytanie tworzące zbiór Matematyka
```

```
 )
SELECT Nazwisko, Imie, Pesel
FROM Fizyka
  INTERSECT
SELECT   Nazwisko, Imie, Pesel
FROM Matematyka
```

Wynik zapytania pokazano na rysunku 51.

Nazwisko	Imie	Pesel
Kura	Kasia	93022277654

Rysunek 51.
Wynik zapytania z wykorzystaniem operatora INTERSECT

Uwaga: W wyniku występuje uczeń o nazwisku Kura Kasia, ponieważ był obecny w obu zbiorach.

3.5. 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. Istnieje 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

```
Select Nazwisko,
       Imie,
       Pesel
FROM Uczniowie
WHERE idklasy NOT IN  (
                    SELECT Idklasy
                        FROM Klasy
                    WHERE Nazwa LIKE ,%a%'
                        )
```

Podzapytanie może być wykonane niezależnie od zapytania nadrzędnego.

Dla przykładowej bazy danych otrzymujemy wynik pokazany na rysunku 52.

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 52.
Wynik zapytania z wykorzystaniem podzapytania nieskorelowanego

Zapytanie złożone skorelowane

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

Podzapytanie nie może być wykonane samodzielnie, ponieważ odwołuje się do tabeli Uczniowie.

Dla przykładowej bazy danych otrzymujemy wynik pokazany na rysunku 53.

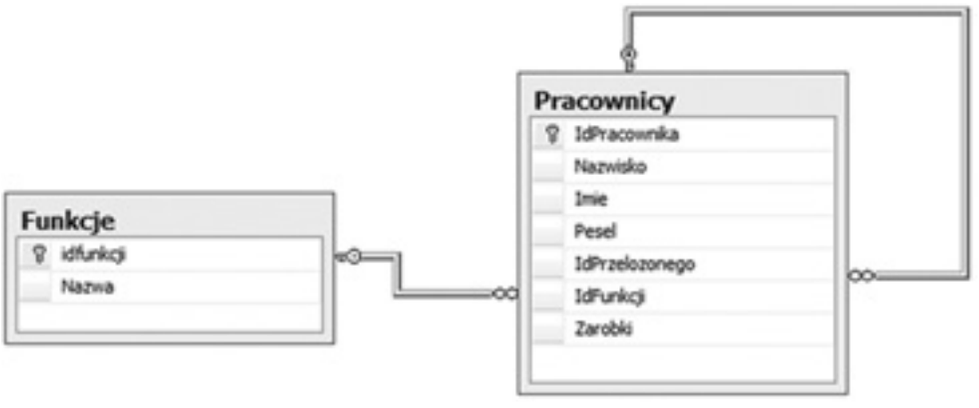
Nazwisko	Imie	Pesel	Srednia
Piesek	Jan	92051587746	2.980000
Gąska	Wacek	91031199123	2.857142
Krówka	Rysio	92051577646	2.781818
Zebra	Wojtek	93030399846	3.075757
Sarenka	Rysio	92121278766	2.563636

Rysunek 53.
Wynik zapytania z wykorzystaniem podzapytania skorelowanego

Zapytania złożone pozwalają w jednym poleceniu wykonać bardzo złożone operacje. Wprowadzone do standardu języka SQL wyrażenia CTE (opisane w rozdziale 2.3) rozszerzają możliwości poleceń SQL, umożliwiając definiowanie zbiorów nazwanych, które mogą być wykorzystywane do realizacji danego polecenia SQL.

3.6. FUNKCJE SZEREGUJĄCE.

Kolejność wierszy w modelu relacyjnym jest z założenia niezdefiniowana. Trudno w klasycznym ujęciu języka SQL wybrać wiersze, jeżeli w kryterium wyboru chcielibyśmy odwołać się do pozycji tych wierszy w zbiorze wynikowym. Przykładowo, gdybyśmy chcieli otrzymać dane faktur w wybranym miesiącu, których wartość jest mniejsza od pięciu faktur o wartości największej. Krótka analiza tego problemu wskazuje, że najprostszą drogą jego rozwiązania byłoby uporządkowanie faktur wystawionych w danym miesiącu według ich wartości, a następnie do zbioru wynikowego przekazać tylko te, które są na pozycji większej niż pięć. Problem pozwalają rozwiązać wprowadzone w SQL Server 2005 funkcje szeregujące. Działanie funkcji szeregujących omówimy na bazie kilku przykładów. Przykłady zapytań będą odnosiły się do fragmentu bazy danych pokazanego na rysunku 54.



Rysunek 54.
Przykładowy fragment bazy danych

Ogólna składnia funkcji szeregujących jest następująca:

FunkcjaSzeregujaca OVER ([PARTITION BY Wyrażenie] ORDER BY Wyrażenie)

Dostępne są cztery funkcje szeregujące:

- ROW_NUMBER() – zwraca kolejny numer wiersza według określonego porządku
- RANK() – zwraca pozycję danego wiersza w rankingu według określonego porządku (ranking nieciągły – jeżeli dwa wiersze będą na pierwszej pozycji to kolejne miejsce w rankingu będzie 3)
- DENSE_RANK() – zwraca pozycję danego wiersza w rankingu według określonego porządku (ranking ciągły – jeżeli dwa wiersze będą na pierwszej pozycji to kolejne miejsce w rankingu będzie 2)
- NTILE(n) – realizuje podział wyniku zapytania na n części – zwraca numer części, do której został przydzielony dany wiersz według określonego porządku

Powyższy opis funkcji szeregujących pokazuje konieczność określenia porządku związanego z funkcją szeregującą (klauzula OVER +ORDER BY). Dodatkowo można wykorzystać klauzulę PARTITION BY – która definiuje „okno", w którym działa funkcja szeregująca.

Działanie omawianych funkcji pokażemy na kilku przykładach.

Zapytanie, które przygotuje listę pracowników z kolumną określającą numer wiersza uporządkowanego według zarobków pracownika (odnosi się do fragmentu bazy danych pokazanego na rysunku 54).

```
SELECT ROW_NUMBER() OVER (ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
```

Przykładowy wynik zapytania pokazano na rysunku 55.

Nr	Nazwisko	Imie	Zarobki
1	Kot	Jan	7000,00
2	Pies	Janina	6100,00
3	Okoń	Stanisław	6100,00
4	Leszcz	Piotr	5200,00
5	Szczupak	Maria	5200,00
6	Lin	Karol	4950,00
7	Myszka	Szara	4950,00
8	Szczurek	Paweł	4950,00
9	Królik	Violetta	4500,00
10	Płotka	Zofia	2700,00

Rysunek 55.
Przykładowy wynik zapytania(z funkcją ROW_NUMBER())

Należy zwrócić uwagę na fakt, że wiersze o tej samej wartości kryterium porządkowania umieszczone zostaną w wyniku w kolejności nieokreślonej. Na przykład wiersze o nr 6-8.

Kolejne zapytanie pokazuje działanie funkcji RANK().

```
SELECT RANK() OVER (ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
```

Przykładowy wynik zapytania pokazano na rysunku 56. Podobnie jak w przypadku funkcji ROW_NUMBER, wiersze o tej samej pozycji rankingu w wyniku zapytania umieszczone są w kolejności nieokreślonej.

Nr	Nazwisko	Imie	Zarobki
1	Kot	Jan	7000,00
2	Pies	Janina	6100,00
2	Okoń	Stanisław	6100,00
4	Leszcz	Piotr	5200,00
4	Szczupak	Maria	5200,00
6	Lin	Karol	4950,00
6	Myszka	Szara	4950,00
6	Szczurek	Paweł	4950,00
9	Królik	Violetta	4500,00
10	Płotka	Zofia	2700,00

Rysunek 56.
Przykładowy wynik zapytania(z funkcją RANK())

Kolejne zapytanie pokazuje działanie funkcji DENSE_RANK() (ranking ciągły). Funkcja ta określa, które miejsce w rankingu według określonego porządku zajmuje dany wiersz. Podobnie jak w powyższych przykładach, wiersze na tej samej pozycji rankingu są udostępnione w kolejności nieokreślonej.

Przykładowy wynik zapytania pokazano na rysunku 57.

```
SELECT DENSE_RANK() OVER (ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
```

Nr	Nazwisko	Imie	Zarobki
1	Kot	Jan	7000,00
2	Pies	Janina	6100,00
2	Okoń	Stanisław	6100,00
3	Leszcz	Piotr	5200,00
3	Szczupak	Maria	5200,00
4	Lin	Karol	4950,00
4	Myszka	Szara	4950,00
4	Szczurek	Paweł	4950,00
5	Królik	Violetta	4500,00
6	Płotka	Zofia	2700,00

Rysunek 57.
Przykładowy wynik zapytania(z funkcją DENSE_RANK())

Funkcja NTILE, której przykład wykorzystania pokazany jest w kolejnym zapytaniu dokonuje podziału wyniku zapytania na n części. Funkcja taka może być wykorzystywana w zadaniach analitycznych polegajacych na segmentacji danych. Podział realizowany jest proporcjonalnie według określonego kryterium.

```
SELECT NTILE() OVER (ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
```

Przykładowy wynik zapytania pokazano na rysunku 58.

Popatrzmy teraz na różnice. ROW_NUMBER zwraca numer rekordu, więc zawsze będą to kolejne liczny naturalne. RANK zwraca pozycję w rankingu. Jeżeli dwa rekordy są identyczne, wtedy pozycja będzie ta sama. Zauważmy, że żaden rekord nie ma numeru 3. To dlatego, że w tym przypadku pozycja trzecia jest na równi z pozycją drugą (dwa srebrne medale, brązowego nie przyznano). DANSE_RANK działa podobnie do RANK, z tym, że nie pomija żadnej liczby naturalnej. Mamy zatem pierwsze miejsce, dwa drugie miejsca i jedno trzecie.

Nr	Nazwisko	Imie	Zarobki
1	Kot	Jan	7000,00
1	Pies	Janina	6100,00
1	Okoń	Stanisław	6100,00
2	Leszcz	Piotr	5200,00
2	Szczupak	Maria	5200,00
2	Lin	Karol	4950,00
3	Myszka	Szara	4950,00
3	Szczurek	Paweł	4950,00
4	Królik	Violetta	4500,00
4	Płotka	Zofia	2700,00

Rysunek 58.
Przykładowy wynik zapytania(z funkcją NTILE())

Te cztery różne funkcje wystarczają do zrealizowania większości zapytań numerujących dane. To, którą użyjemy, będzie zależało od konkretnego przypadku i konkretnych wymagań. Reasumując warto zaznaczyć, że zastosowanie tych funkcji zdecydowanie wykracza poza przedstawione tutaj przykłady. ROW_NUMBER jest na przykład powszechnie stosowane do stronicowania (paginacji) danych w aplikacjach internetowych.

Kolejny przykład demonstruje wykorzystanie klauzuli PARTITION BY. Wykorzystana w zapytaniu klauzula **PARTITION BY** IDFunkcji powoduje, że funkcja szeregująca generuje swoje wartości oddzielnie dla każdej grupy wierszy o tej samej wartości w kolumnie IdFunkcji.

```
SELECT ROW_NUMBER() OVER (PARTITION BY IDFunkcji
                                ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
```

Przykładowy wynik zapytania pokazano na rysunku 59.

W wyniku zapytania należy zwrócić uwagę na wartości zwracane w kolumnie Nr. Numeracja realizowana jest dla grup wierszy o tej samej wartości kolumny podanej jako wyrażenie w klauzuli PARTITON BY.

Nr	Nazwisko	Imie	Zarobki
1	Kot	Jan	7000,00
1	Pies	Janina	6100,00
2	Okoń	Stanisław	6100,00
3	Leszcz	Piotr	5200,00
4	Szczupak	Maria	5200,00
5	Płotka	Zofia	2700,00
1	Lin	Karol	4950,00
1	Myszka	Szara	4950,00
2	Szczurek	Paweł	4950,00
3	Królik	Violetta	4500,00

Rysunek 59.
Przykładowy wynik zapytania (wykorzstanie klauzuli PARTITION BY)

Zastosowanie funkcji szeregujących do rozwiazania konkretnych problemów pokazane zostanie w kolejnych przykładach, w których wykorzystujemy także wyrażenia CTE.

Kolejny przykład pokazuje zapytanie, które przygotowuje listę pracowników na pozycjach od 4 do 8 uporządkowanych według wartości zarobków.

```
WITH Lista AS
```

```
(
SELECT ROW_NUMBER() OVER (ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
)
SELECT *
FROM Lista
WHERE  Nr BETWEEN 4 AND 8
```

Przykładowy wynik zapytania pokazano na rysunku 60.

Nr	Nazwisko	Imie	Zarobki
4	Leszcz	Piotr	5200,00
5	Szczupak	Maria	5200,00
6	Lin	Karol	4950,00
7	Myszka	Szara	4950,00
8	Szczurek	Pawel	4950,00

Rysunek 60.
Przykładowy wynik zapytania (wybieranie danych z określonego zakresu)

Kolejny przykład zwraca pracowników, których zarobki są na drugim miejscu względem ich wartości.

```
WITH Lista AS
(
SELECT RANK() OVER (ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
)
SELECT *
FROM Lista
WHERE  Nr =2
```

Przykładowy wynik zapytania pokazano na rysunku 61.

Nr	Nazwisko	Imie	Zarobki
2	Pies	Janina	6100,00
2	Okoń	Stanisław	6100,00

Rysunek 61.
Przykładowy wynik zapytania (osoby na określonym miejscu rankingu)

Poniższe zapytanie zwraca wiersze przydzielone do drugiej grupy według zarobków przy podziale na cztery segmenty.

```
WITH Lista AS
(
SELECT NTILE(4) OVER (ORDER BY Zarobki DESC) as Nr,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
)
SELECT *
FROM Lista
WHERE  Nr =2
```

Przykładowy wynik zapytania pokazano na rysunku 62.

Nr	Nazwisko	Imie	Zarobki
2	Leszcz	Piotr	5200,00
2	Szczupak	Maria	5200,00
2	Lin	Karol	4950,00

Rysunek 62.
Przykładowy wynik zapytania (osoby w okreslonym segmencie)

Na zakończenie, w kolejnym przykładzie, pokazano wykorzystanie klauzuli OVER PARTITION BY z funkcją agregujacą, a nie szeregującą.

Zapytania zwraca dane tych pracowników, których zarobki są większe niż średnia zarobków pracowników piastujących tę samą funkcję.

```
WITH Lista AS
(
SELECT AVG(Zarobki) OVER (PARTITION BY IdFunkcji ) as SredniaGrupy,
        Nazwisko,Imie,Zarobki
FROM Pracownicy
)
SELECT *
FROM Lista
WHERE  Zarobki›=SredniaGrupy
```

Przykładowy wynik zapytania pokazano na rysunku 63.

SredniaGrupy	Nazwisko	Imie	Zarobki
7000,00	Kot	Jan	7000,00
5060,00	Pies	Janina	6100,00
5060,00	Okoń	Stanislaw	6100,00
5060,00	Leszcz	Piotr	5200,00
5060,00	Szczupak	Maria	5200,00
4950,00	Lin	Karol	4950,00
4800,00	Myszka	Szara	4950,00
4800,00	Szczurek	Pawel	4950,00

Rysunek 63.
Przykładowy wynik zapytania (agregacja w oknie)

Podsumowując, funkcje szeregujące oraz klauzula OVER są dodatkowym potencjałem umożliwiającym tworzenie zapytań realizujących dodatkowe zadania w zależności od potrzeb.

Uwaga: Omówione przykładowe zapytania zostaną wykonane i omówione przez prowadzącego kurs.

3.7. ĆWICZENIA

3.7.1. Ćwiczenie 4 – Wykorzystanie operatora złączenia zewnętrznego.

W ramach ćwiczenia należy napisać zapytanie skierowane do bazy danych ElektronicznyDziennikOcen, które przygotuje zestawienie zawierające następujące dane:

- nazwisko ucznia,
- imię ucznia,

- numer Pesel,
- nazwę przedmiotu,
- średnią ocenę danego ucznia z danego przedmiotu.

W zestawieniu powinny być dane wszystkich uczniów w połączeniu ze wszystkimi przedmiotami zapisanymi w tabeli Przedmioty.

Zapytanie wykonać z wykorzystaniem operatora złączeń OUTER JOIN.

Dodatkowo wykonać zapytanie zwracające te same dane w formie tabeli przestawnej.

Uwaga: W bazie danych mogą być zapisane dane uczniów, którzy nie mają wystawionej jeszcze żadnej oceny oraz nie każdy uczeń ma wystawione oceny z wszystkich dostępnych przedmiotów.

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

3.7.2. Ćwiczenie 5 – Zapytanie z wykorzystaniem operatorów działania na zbiorach.

W ramach ćwiczenia należy napisać zapytanie zwracające dane uczniów:

- nazwisko ucznia,
- imię ucznia,
- numer Pesel,
- nazwę klasy,

dla tych uczniów, którzy w swojej klasie znajdują się wśród 3 najlepszych, według średniej ocen, z fizyki, a nie znajdują się w drugiej połowie zestawienia według średniej ocen z geografii.

Zapytanie wykonać z wykorzystaniem operatorów działania na zbiorach.

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

3.7.3. Ćwiczenie 6 – Zapytania rekurencyjne.

W ramach ćwiczenia należy napisać zapytanie, które dla tabel Pracownicy i Funkcje, pokazanych na rysunku 64, zwróci wynik pokazany na rysunku 65. Podstawowym problem jest utworzenie zawartości kolumny ścieżka, która opisuje pełną ścieżkę podległości dla danego pracownika.

Tabela Pracownicy zawiera klucz obcy IdPrzelozonego, który odwołuje się do innego wiersza tej tabeli, dzięki czemu możemy zapisać hierarchię pracowników.

W zapytaniu należy wykorzystać rekurencyjne możliwości wyrażeń CTE.

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

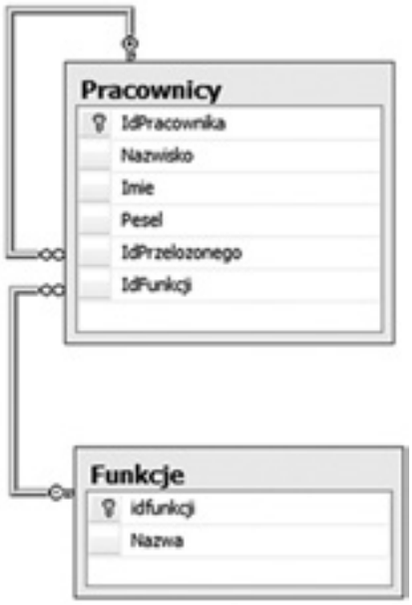
3.7.4. Ćwiczenie 7 – Zapytania z wykorzystaniem funkcji szeregujących.

W ramach ćwiczenia należy napisać i wykonać zapytanie, które zwróci dane uczniów zawierające nazwisko, imię i numer Pesel, dla tych uczniów, którzy w swoich klasach są wśród 8 najlepszych, według średniej oceny, z fizyki i jednocześnie ich średnia ocen z wszystkich przedmiotów jest wyższa od średniej ocen całej klasy.

Zapytanie należy zrealizować z wykorzystaniem funkcji szeregujących.

Uwaga: Zapytanie może być zrealizowane na wiele sposobów – po wykonaniu porównać sposób wykonania różnych uczestników kursu.

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



Rysunek 64.
Schemat tabel Pracownicy i Funkcje

Pracownik	Funkcja	Poziom	idpracownika	sciezka
Jan Kot	Prezes	0	1	
Janina Pies	Dyrektor Departamentu	1	2	Prezes-Jan Kot/
Stanisław Okoń	Dyrektor Departamentu	1	3	Prezes-Jan Kot/
Zofia Plotka	Dyrektor Działu	2	4	Prezes-Jan Kot/Dyrektor Departamentu-Janina Pies/
Piotr Leszcz	Dyrektor Działu	2	5	Prezes-Jan Kot/Dyrektor Departamentu-Janina Pies/
Maria Szczupak	Kierownik Zespołu	3	6	Prezes-Jan Kot/Dyrektor Departamentu-Janina Pies/Dyrektor Działu-Zofia Plotka/
Karol Lin	Kierownik Zespołu	3	7	Prezes-Jan Kot/Dyrektor Departamentu-Janina Pies/Dyrektor Działu-Zofia Plotka/
Szara Myszka	Referent	4	8	Prezes-Jan Kot/Dyrektor Departamentu-Janina Pies/Dyrektor Działu-Zofia Plotka/Kierownik Zespołu-Maria Szczupak/
Paweł Szczurek	Referent	4	9	Prezes-Jan Kot/Dyrektor Departamentu-Janina Pies/Dyrektor Działu-Zofia Plotka/Kierownik Zespołu-Maria Szczupak/
Violetta Królik	Asystent	5	10	Prezes-Jan Kot/Dyrektor Departamentu-Janina Pies/Dyrektor Działu-Zofia Plotka/Kierownik Zespołu-Maria Szczupak/Referent-Szara Myszka/

Rysunek 65.
Oczekiwany wynik zapytania

4 PODSUMOWANIE.

Zagadnienia związane z wykorzystaniem zaawansowanych możliwości języka SQL są złożone, ale ich poznanie i zrozumienie umożliwia realizowanie bardzo złożonych zadań w ramach jednego polecenia SQL. W trakcie naszego kursu pokazane zostały przykłady wykorzystania wyrażeń CTE do realizacji zapytań rekurencyjnych, funkcji szeregujących oraz nowych operatorów złączeń tabel (PIVOT i APPLY). Wielu aspektów wykorzystania zaawansowanych technik zapytań, ze względu na ograniczony czas kursu, nie omawialiśmy. Widza zdobyta w trakcie ćwiczeń powinna umożliwić dalsze jej pogłębianie w ramach samodzielnych zadań.

5 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., Skinner 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
6. Castro E., *Po prostu XML*, Helion, Gliwice 2001
7. Walmsley P., *Wszystko o XML Schema*, Wydawnictwo Naukowo-Techniczne, Warszawa 2007
8. Kwiatkowski H., *Praca magisterska*, WWSI, Warszawa 2011

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