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 Poranna: Bazy danych

Język SQL – podstawy zapytań

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
– podstawy zapytań

Rodzaj zajęć: Wszechnica Poranna
Tytuł: Język SQL – podstawy zapytań
Autor: mgr inż. Andrzej Ptasznik

Redaktor merytoryczny: prof. dr hab. Maciej M Sysło

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 2009

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 – podstawy zapytań

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

Streszczenie

Wykład. Przedstawiona jest krótka historia języka SQL i jego podstawowe cechy. W dalszej części wykładu są omawiane różne aspekty tworzenia zapytań w języku SQL. Podstawowe cechy składni polecenia SELECT są omówione na przykładzie zapytań skierowanych do jednej tabeli. W kolejnych rozdziałach jest pokazane łączenie tabel i zapytania korzystające z danych zapisanych w wielu tabelach, wykorzystanie funkcji agregujących oraz zapytania złożone. W ostatnim rozdziale są zaprezentowane niektóre nowe elementy polecenia SELECT języka SQL. W trakcie wykładu są prezentowane i omawiane różne przykłady zapytań skierowanych do przykładowej bazy danych (ElektronicznyDziennikOcen).

Warsztaty. W ramach warsztatów zostanie przedstawiony sposób instalacji MS SQL Server 2008 Express Edition a następnie uczniowie zostaną zapoznani z SQL Server Menagement Studio, narzędziem klienckim umożliwiającym korzystanie i administrowanie SQL Serwerem. W kolejnych ćwiczeniach będą formułowane różne postaci zapytań, wykorzystujące polecenie SELECT języka SQL. W każdym ćwiczeniu zadaniem ucznia będzie napisanie i wykonanie podanego zapytania a następnie samodzielne napisanie zapytania z niewielką pomocą prowadzącego warsztaty. Kolejne ćwiczenia stopniują trudność wykonywanych zapytań.

Spis treści

Wykład

Warsztaty

WYKŁAD

1 KRÓTKA HISTORIA JĘZYKA SQL

Historia relacyjnego modelu danych rozpoczęła się w roku 1970 wraz z publikacją E.F.Codda pt. ***A Relational Model of Data for Large Shared Data Banks*** (pol. *Relacyjny model danych dla dużych współdzielonych banków danych*). Ten artykuł wzbudził duże zainteresowanie, ponieważ przedstawiał możliwości realizacji i praktyczne zastosowania komercyjnego systemu baz danych. Praca ta stworzyła teoretyczne podstawy budowania baz danych w oparciu o relacyjne podejście do jej modelowania. Opublikowana teoria bardzo szybko zainteresowała potencjalnych twórców i producentów systemów zarządzania bazami danych. Droga od teorii do praktyki bywa często długa i wyboista ale w tym przypadku, ze względu na pilne potrzeby rynku, przebiegała dość szybko i sprawnie. Jednym z podstawowych wyzwań było opracowanie sposobu komunikowania się i korzystania z relacyjnych baz danych, czyli opracowanie specjalnego języka programowania.

W ramach prac nad językiem do obsługi baz danych, prowadzonych w firmie IBM, w roku 1974 powstał język SEQUEL (ang. *Structured English Query Language* – Stukturalny Angielski Język Zapytań), który następnie został rozwinięty i nazwany SQL (ang. *Structured Query Language* – Strukturalny Język Zapytań) – zmiana nazwy wynikała ze sporu prawnego o zastrzeżoną nazwę SEQUEL. Pod koniec lat 70-tych XX wieku, firma ORACLE wypuściła na rynek pierwszy komercyjny system zarządzania bazami danych oparty o SQL. W latach 80-tych i 90–tych, ubiegłego wieku, trwał burzliwy rozwój baz danych opartych na modelu relacyjnym i języku SQL. Ponieważ wielu producentów zaczęło tworzyć rozwiązania baz danych oparte o model relacyjny i język SQL powstawało ryzyko, że u różnych producentów język SQL będzie rozwijał się inaczej. Rozwiązaniem tego problemu było zdefiniowanie standardów języka SQL przez organizację ISO (ang. *International Standards Organization*) i ANSI (ang. *American National Standards Institute*). Definiowanie standardu należy traktować jako wytyczne dla producentów systemów, w jakim kierunku rozwijać kolejne opracowania języka SQL, jakie nowe elementy mogą zostać wprowadzone do języka i jak system baz danych powinien realizować operacje związane z definiowaniem baz danych i ich eksploatacją. Obecnie język SQL jest powszechnie stosowanym językiem komunikacji z bazami danych w systemach opartych o relacyjny model danych,

Krótka historia standardów języka SQL:

- 1986: pierwszy standard SQL (SQL-86),
- 1989: następny standard SQL (SQL-89),
- 1992: kolejna, wzbogacona wersja standardu (SQL-92 lub SQL 2),
- 1999: wdrożenie następnej wersji standardu rozszerzonego o pewne cechy obiektowości (SQL 3),
- 2003: Kolejne rozszerzenie standardu (m.in. włączenie do standardu języka XML) SQL 4.

Język SQL jest nadal rozwijany i trudno przewidzieć, jakie kierunki rozwoju zostaną wybrane, zaś odpowiedzi na to pytanie dostarczać będą kolejne wersje standardu.

2 CECHY JĘZYKA SQL

Język SQL jest zaliczamy do tak zwanych **języków deklaratywnych** (języki czwartej generacji 4GL) zorientowanych na wynik. W językach tego typu użytkownik definiuje, jaki efekt końcowy chce osiągnąć w wyniku działania wybranego polecenia bez określania, w jaki sposób należy to wykonać. O tym „jak" zrealizować polecenie języka SQL decyduje System Zarządzania Bazami Danych, który po otrzymaniu polecenia do wykonania realizuje czynności związane z jego realizacją (analiza składni, optymalizacja, opracowanie planu wykonania polecenia i realizacja przygotowanego planu). Polecenia języka SQL nie zawierają instrukcji sterujących wykonaniem programu, takich jak instrukcje warunkowe czy instrukcje pętli, gdyż użytkownik nie musi określać sposobu wykonania danego polecenia. Logika działania języka SQL w kontekście bazy danych jest oparta na algebrze relacji, czyli działa na zbiorach danych. Cechą charakterystyczną jest wykorzystywanie logiki trójwartościowej, czyli takiej, w której poza wartościami logicznymi **true** (prawda) i **false** (fałsz) występuje także wartość **unknown** (nieznana), reprezentowana przez wartość **null**.

Przetwarzanie poleceń w języku SQL jest realizowane w trybie interpretacji, czyli wysłanie polecenia do Systemu Zarządzania Bazami Danych uruchamia proces jego walidacji, optymalizacji i generowania planu wykonania. Dla poprawy wydajności, serwery baz danych przechowują możliwie długo utworzone plany wykonania, żeby przy kolejnym wywołaniu takiego samego polecenia skorzystać z przygotowanego planu bez konieczności wykonywania czynności przygotowawczych.

Praca z wykorzystaniem SQL może być realizowana na kilka sposobów :

- poprzez interaktywne zadawanie pytań do bazy (monitor),
- budowanie skryptów (zbioru wsadowo wykonywanych zapytań w SQL),
- osadzanie kodu (pojedynczych zapytań i całych procedur) SQL w innych językach programowania (na poziomie aplikacji),
- procedur składowanych (na poziome bazy danych).

Polecenia języka SQL można podzielić na trzy główne grupy:

- Język Definiowania Danych – **DDL** (ang. *Data Definition Language*) – 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).

- Język Manipulacji Danymi – **DML** (ang. *Data Manipulation Language*) – część języka SQL umożliwiająca dokonywanie operacji na danych, takich jak: wstawianie wiersza do tabeli, modyfikowanie istniejących wierszy, usuwanie wierszy oraz pobieranie danych (zapytania). W skład DML wchodzą polecenia: **INSERT** (wstaw wiersze do tabeli), **UPDATE** (zmodyfikuj wiersze w tabeli), **DELETE** (usuń wiersze z tabeli) i **SELECT** (pobierz dane) oraz nowo wprowadzone do standardu polecenie **MERGE** (wykonaj aktualizację zbiorczą).

- Język Kontroli Danych – **DCL** (ang. *Data Control Language*) – część języka SQL umożliwiająca sterowanie prawami dostępu do danych. W skład DCL wchodzą polecenia **GRANT** (przydziel prawo) i **REVOKE** (pozbawienie przydzielonego prawa).

W swojej podstawowej postaci język SQL nie zawiera wielu poleceń ale każde z tych poleceń ma swoją złożoną składnię. Wszystkie działania na relacyjnej bazie danych można wykonać korzystając z poleceń języka SQL. Poniżej przedstawiamy kilka przykładowych poleceń SQL z krótkim objaśnieniem zasad ich działania.

- Definiowanie bazy danych:

CREATE DATABASE NaszaBaza – czyli utwórz nową bazę danych o nazwie NaszaBaza. Zwróćmy uwagę, że w tym poleceniu nie ma mowy o tym, jak tworzy się tę bazę.

- Definiowanie przykładowej tabeli:

```
CREATE TABLE Uczniowie
(
    IdUcznia        int IDENTITY(1,1) NOT NULL,
    Nazwisko        varchar(50) NOT NULL,
    Imie            varchar(50) NOT NULL,
    DataUrodzenia   date NOT NULL,
    CzyChlopak      bit  NOT NULL,
    Pesel           varchar(11) NULL,
    CONSTRAINT PK_uczniowie PRIMARY KEY CLUSTERED
    (IdUcznia ASC)
)
```

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

Zdefiniować nową tabelę o nazwie Uczniowie, która składa się z następujących kolumn:
- *IdUcznia – 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,*
- *CzyChlopak – kolumna typu bit (typ logiczny) z niedopuszczalną wartością null,*
- *Pesel – kolumna typu varchar(11) z dopuszczalną wartością null.*

Dodatkowo definiujemy ograniczenie uznające kolumnę idUcznia 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.

- Wstawianie wierszy do przykładowej tabeli

```
INSERT INTO Uczniowie (Nazwisko, Imie, DataUrodzenia, CzyChlopak, Pesel)
 VALUES('Kot', 'Jan', '1991-07-12','true', '91071276538')

INSERT INTO Uczniowie (Nazwisko, Imie, DataUrodzenia, CzyChlopak, Pesel)
 VALUES('Nowak', 'Janina', '1991-03-22','false', '91032267549')

INSERT INTO Uczniowie (Nazwisko, Imie, DataUrodzenia, CzyChlopak)
 VALUES('Sarnowski', 'Piotr', '1991-12-12','true' )
```

Składnię polecenia **INSERT**, na podstawie podanych przykładów można łatwo zinterpretować następująco: *Wstaw do tabeli o nazwie Uczniowie do kolumn wymienionych po nazwie tabeli wartości zawarte w klauzuli **VALUES***

- Pobieranie części zawartości tabeli Uczniowie :

```
SELECT Nazwisko, Imie, Pesel
FROM Uczniowie
WHERE  CzyChlopak=true
ORDER BY nazwisko
```

Tym razem zaprezentowany został przykład zapytania **SELECT**, które można opisać językiem potocznym: *Pobierz i dostarcz tabelę zawierającą nazwisko, imię i numer Pesel – dane pobrać z tabeli o nazwie Uczniowie (**FROM**). W wyniku zapytania mają znaleźć się tylko chłopcy (**WHERE**). Wynik zapytania uporządkować alfabetycznie według nazwiska (**ORDER BY**)*.

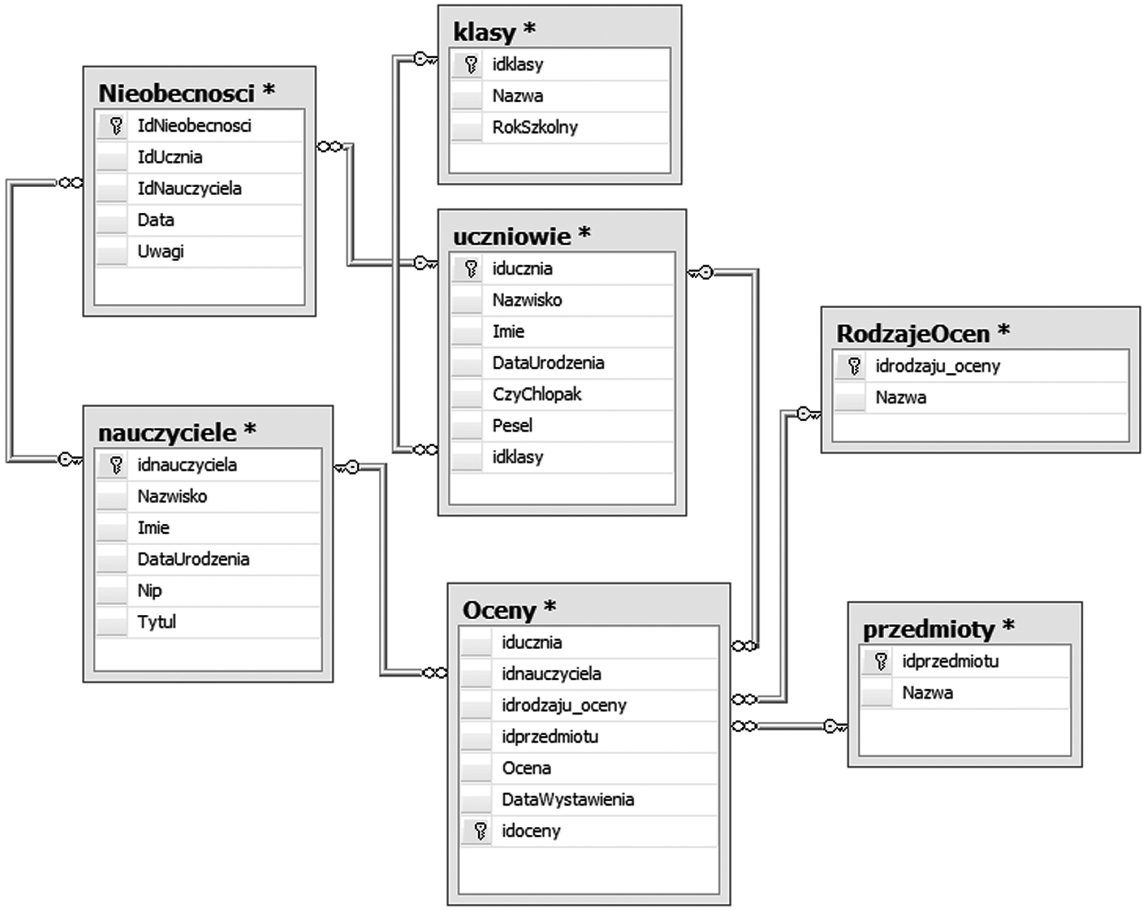
3. PRZYKŁADOWA BAZA DANYCH

Bardzo trudno jest mówić o języku SQL, a szczególnie o poleceniach realizujących zapytania, bez odniesienia do konkretnej bazy danych i dlatego na potrzeby dalszych rozważań omówimy krótko przykładową bazę danych, do której będą odwoływały się wszystkie przykładowe polecenia omawiane w dalszej części wykładu.

Zaprezentowana schematycznie na rys. 1 baza danych udostępnia tabele, dzięki którym można ewidencjonować oceny wystawiane uczniom. Pisanie zapytań w języku SQL wymaga dobrej znajomości bazy danych, których te zapytania dotyczą, dlatego opiszemy krótko poszczególne tabele.

1. Tabela Klasy – tabele tego typu nazywamy **tabelami słownikowymi**. Wiersz zapisany w tej tabeli opisuje jedną klasę. Do opisu klasy są wykorzystywane kolumny: Nazwa i RokSzkolny. Proszę zwrócić uwagę na fakt, że taka sama nazwa klasy może się powtórzyć z inną wartością w kolumnie RokSzkolny.

Elektroniczny dziennik ocen



Rysunek 1.
Przykładowa baza danych

2. Tabela Uczniowie – w tej tabeli zapisujemy dane wszystkich uczniów, których oceny będziemy rejestrować. W tabeli Uczniowie znajduje się klucz obcy idklasy, dzięki któremu możemy w kontekście konkretnego ucznia uzyskać informację, do jakiej klasy uczęszcza. Zwrócić uwagę na kolumnę o nazwie CzyChlopak – typ kolumny został zadeklarowany jako bit, czyli ma dwie wartości logiczne (1 – prawda, 0 – fałsz).
3. Tabela Nauczyciele – w tej tabeli zapisujemy dane wszystkich nauczycieli, którzy wystawiają oceny,
4. Tabele Przedmioty i RodzajeOcen – klasyczne tabele słownikowe, w których są przechowywane nazwy przedmiotów i nazwy rodzajów ocen.
5. Tabela Oceny – to najważniejsza tabela naszej bazy danych, w której są zapisywane wystawiane oceny. W tej tabeli znajdują się cztery klucze obce (iducznia, idnauczyciela, idrodzaju_oceny, idprzedmiotu). Na pierwszy rzut oka dane zapisywane w tabeli Oceny są mało czytelne, ponieważ liczbowe wartości kluczy obcych nic nam nie mówią. Interpretacja tych danych jest możliwa dzięki odpowiednim zapytaniom, w których połączymy te dane z innymi tabelami.
6. Tabela Nieobecnosci – w tej tabeli zapisujemy dane opisujące nieobecności uczniów na zajęciach.

Szczegóły przykładowej bazy danych zostaną omówione w trakcie warsztatów.

4 PODSTAWY ZAPYTAŃ – OPERACJE NA MODELU RELACYJNYM

Głównym celem wykładu jest bliższe zapoznanie z poleceniem SELECT, przy pomocy którego w języku SQL są realizowane zapytania. Zanim przystąpimy do omawiania polecenia SELECT wyjaśnijmy istotę zapytań do re-

lacyjnej 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)
- Operacja selekcji
- Operacja łączenia

Praktycznie każde zapytanie realizowane za pomocą polecenia języka SQL jest wypadkową tych trzech operacji. Wyjaśnimy teraz na czym polegają te operacje.

1. **Operacja projekcji** – 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, patrz rys. 2.

Tabela wyjściowa

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2
2	Piesek	Jan	1992-05-15	1	92051587746	1
3	Lisek	Kasia	1992-02-22	0	92022277654	2
4	Kurka	Jola	1992-06-02	0	92060288788	2
5	Gąska	Wacek	1991-03-11	1	91031199123	1
6	Krówka	Rysio	1992-05-15	1	92051577646	1

Tabela wynikowa po operacji projekcji

Nazwisko	Imie	Pesel
Kotek	Katarzyna	92031275446
Piesek	Jan	92051587746
Lisek	Kasia	92022277654
Kurka	Jola	92060288788
Gąska	Wacek	91031199123
Krówka	Rysio	92051577646

Rysunek 2.
Operacja projekcji i jej wynik

Operacja projekcji zmniejsza rozmiar tabeli wyjściowej poprzez wyeliminowanie kolumn, które w danym momencie uznajemy za nieistotne.

2. **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, dla których wyrażenie logiczne nie jest spełnione, patrz rys. 3, gdzie za warunek selekcji przyjęto idklasy=2.

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2
2	Piesek	Jan	1992-05-15	1	92051587746	1
3	Lisek	Kasia	1992-02-22	0	92022277654	2
4	Kurka	Jola	1992-06-02	0	92060288788	2
5	Gąska	Wacek	1991-03-11	1	91031199123	1
6	Krówka	Rysio	1992-05-15	1	92051577646	1

← Tabela wyjściowa

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2
3	Lisek	Kasia	1992-02-22	0	92022277654	2
4	Kurka	Jola	1992-06-02	0	92060288788	2

← Tabela wynikowa

Rysunek 3.
Operacja selekcji

3. **Połączenie operacji projekcji i selekcji** – możliwe jest połączenie operacji projekcji i selekcji i dzięki temu można wybrać dowolny podzbiór danych z tabeli wyjściowej, patrz rys. 4.

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2
2	Piesek	Jan	1992-05-15	1	92051587746	1
3	Lisek	Kasia	1992-02-22	0	92022277654	2
4	Kurka	Jola	1992-06-02	0	92060288788	2
5	Gąska	Wacek	1991-03-11	1	91031199123	1
6	Krówka	Rysio	1992-05-15	1	92051577646	1

Tabela wyjściowa

Nazwisko	Imie	idklasy
Kotek	Katarzyna	2
Lisek	Kasia	2
Kurka	Jola	2

Tabela wynikowa – wynik połączenia operacji projekcji i selekcji (Warunek selekcji: idklasy=2

Rysunek 4.
Operacja projekcji i selekcji

4. **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 (na rys. 5 – jest nim idklasy) i klucz podstawowy w tabeli dołączanej (na rys. 5 – również idklasy) do tabeli wyjściowej.

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2
2	Piesek	Jan	1992-05-15	1	92051587746	1
3	Lisek	Kasia	1992-02-22	0	92022277654	2
4	Kurka	Jola	1992-06-02	0	92060288788	2
5	Gąska	Wacek	1991-03-11	1	91031199123	1
6	Krówka	Rysio	1992-05-15	1	92051577646	1

idklasy	Nazwa	RokSzkolny
1	Ia	2008/2009
2	IIa	2008/2009

Rysunek 5.
Łączone tabele

W wyniku operacji łączenia przykładowych tabel z rys. 5, tabelę Uczniowie możemy poszerzyć o kolumny Nazwa i RokSzkolny – warunkiem złączenia jest równość kolumny idklasy w tabeli Uczniowie z kolumną idklasy w tabeli Klasy (patrz rys. 6).

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy	nazwa	RokSzkolny
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2	IIa	2008/2009
2	Piesek	Jan	1992-05-15	1	92051587746	1	Ia	2008/2009
3	Lisek	Kasia	1992-02-22	0	92022277654	2	IIa	2008/2009
4	Kurka	Jola	1992-06-02	0	92060288788	2	IIa	2008/2009
5	Gąska	Wacek	1991-03-11	1	91031199123	1	Ia	2008/2009
6	Krówka	Rysio	1992-05-15	1	92051577646	1	Ia	2008/2009

Rysunek 6.
Wynik operacji łączenia tabel

Operacja łączenia sprowadza dane zapisane w wielu tabelach do postaci jednej tabeli i od tego momentu wszystkie operacje poprawne dla jednej tabeli mogą być wykonywane na tabeli połączonej.

5. **Połączenie operacji projekcji, selekcji i łączenia** – możliwe jest połączenie operacji projekcji, selekcji i łączenia, dzięki czemu możemy wybrać dowolny podzbiór z danych zapisanych w wielu różnych tabelach. Na rys. 7 jest pokazany przykładowy zbiór wynikowy, w którym kolumny Nazwisko i Imie zostały pobrane z tabeli Uczniowie, a kolumny Nazwa i RokSzkolny zostały pobrane z tabeli Klasy (dzięki odpowiedniej

operacji łączenia). W zbiorze wynikowym umieszczone zostały tylko niektóre kolumny z wybranych tabel (dzięki operacji projekcji) oraz tylko te wiersze, dla których jest spełniony warunek Nazwa='IIa' (dzięki operacji selekcji).

Nazwisko	imie	nazwa	RokSzkolny
Kotek	Katarzyna	IIa	2008/2009
Lisek	Kasia	IIa	2008/2009
Kurka	Jola	IIa	2008/2009

Rysunek 7.
Wynik operacji projekcji, selekcji i łączenia

W dalszej części wykładu będziemy poznawać składnię i sposób działania polecenia **SELECT**. Praktycznie wszystkie zapytania są realizowane w oparciu o trzy operacje: projekcję, selekcję i łączenie.

5 POLECENIE SELECT – ZAPYTANIA PROSTE

Należy pamiętać, że wynikiem zapytania jest tabela.

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

W tej części wykładu skoncentrujemy się na formułowaniu zapytań kierowanych do jednej tabeli.

Najprostszą postacią polecenia **SELECT** jest żądanie pobrania wszystkich danych z wybranej tabeli:

SELECT *
FROM Uczniowie

Jako listę kolumn po nazwie poleceniu **SELECT** występuje znak * (gwiazdka), który należy interpretować jako wszystkie dostępne kolumny z tabeli, której nazwa występuje w klauzuli **FROM**. Przykładowy wynik takiego zapytania może mieć postac jak na rys. 8.

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2
2	Piesek	Jan	1992-05-15	1	92051587746	1
3	Lisek	Kasia	1992-02-22	0	92022277654	2
4	Kurka	Jola	1992-06-02	0	92060288788	2
5	Gąska	Wacek	1991-03-11	1	91031199123	1
6	Krówka	Rysio	1992-05-15	1	92051577646	1
7	Zebra	Wojtek	1993-03-13	1	93030399846	1
8	Gazela	Basia	1992-11-11	0	92111177446	2

Rysunek 8.
Wynik zapytania ogólnego

Podstawą zapytań kierowanych do jednej tabeli jest realizacja operacji projekcji i selekcji, dzięki którym możemy wybrać dowolny fragment tabeli wyjściowej.

Operację projekcji w zapytaniach **SELECT** realizujemy poprzez wymienienie listy kolumn, które powinny znaleźć się w tabeli wynikowej.

SELECT Nazwisko, Imie, Pesel, CzyChlopak
FROM Uczniowie

To zapytanie wybiera cztery wymienione kolumny z tabeli Uczniowie (patrz rys. 9).

Nazwisko	Imie	Pesel	CzyChlopak
Kotek	Katarzyna	92031275446	0
Piesek	Jan	92051587746	1
Lisek	Kasia	92022277654	0
Kurka	Jola	92060288788	0
Gąska	Wacek	91031199123	1
Krówka	Rysio	92051577646	1

Rysunek 9.
Wynik selekcji wybranych kolumn

Do przedstawionego wyżej zapytania dodamy teraz warunek selekcji:

SELECT Nazwisko, Imie, Pesel, CzyChlopak
FROM Uczniowie
WHERE CzyChlopak=1

Tak sformułowane zapytanie jest połączeniem operacji projekcji i selekcji, jego wynik jest na rys. 10.

Nazwisko	Imie	Pesel	CzyChlopak
Piesek	Jan	92051587746	1
Gąska	Wacek	91031199123	1
Krówka	Rysio	92051577646	1

Rysunek 10.
Wynik projekcji i selekcji

Często chcemy uzyskać wynik zapytania uporządkowany według zadanego kryterium. W poleceniu **SELECT** porządkowanie wyniku zapytania można uzyskać za pomocą dołączonej do zapytania klauzuli **ORDER BY.** Przedstawiona na rys. 11 tabela jest wynikiem następującego zapytania, które poleca uporządkować listę uczniów według klasy, a w obrębie danej klasy – alfabetycznie według nazwiska.

```
SELECT Nazwisko, Imie, Pesel, Idklasy
FROM Uczniowie
WHERE Idklasy=1 OR Idklasy=2
ORDER BY Idklasy ASC, Nazwisko DESC
```

W klauzuli **ORDER BY** wymienione zostały dwie kolumny co należy interpretować następująco: *uporządkuj według Idlasy a w obrębie wierszy o tej samej wartości Idklasy uporządkuj według nazwiska.* Dodatkowo użyto słowa kluczowe **ASC** i **DESC** określające rodzaj uporządkowania:
ASC (***ascending*** – rosnąco)
DESC (***descending*** – malejąco)

W dotychczas przedstawionych przykładach, w kolumnach tabeli wynikowej znajdowały się dane pobrane bezpośrednio z tabeli, czyli w takiej postaci, w jakiej zostały zapisane. W zapytaniach możemy przekształcać pobrane dane do innej postaci w zależności od naszych potrzeb. W kolejnym przykładzie przekształcimy dane zapisane w kolumnie CzyChlopak do postaci bardziej czytelnej:

Nazwisko	Imie	Pesel	Idklasy
Zebra	Wojtek	93030399846	1
Sarenka	Rysio	92121278766	1
Piesek	Jan	92051587746	1
Krówka	Rysio	92051577646	1
Gąska	Wacek	91031199123	1
Stokrotka	Rysio	93121278766	2
Ryba	Jan	93051587746	2
Różyczka	Basia	93111177446	2
Płotka	Wojtek	93030399846	2
Okoń	Rysio	93051577646	2

Rysunek 11.
Wynik zapytania uporządkowany po dwóch kolumnach

```
SELECT Nazwisko, Imie, Pesel,
      CASE CzyChlopak
         WHEN 1 THEN 'Mężczyzna'
         ELSE 'Kobieta'
      END as Płeć
FROM Uczniowie
WHERE Idklasy=2
```

Przykładowy wynik tego zapytania zawiera kolumnę o nazwie Płeć, w której są wyświetlane wartości tekstowe Kobieta lub Mężczyzna, pomimo tego, że takie dane nie są zapisane w tabeli Uczniowie.

Wykorzystane w zapytaniu wyrażenie, zaczynające się od słowa **CASE** należy interpretować następująco; *W zależności od wartości w kolumnie CzyChlopak (**CASE** CzyChlopak); jeżeli wartość kolumny CzyChlopak jest równa 1, to zwróć tekst Mężczyzna (**WHEN 1 THEN** 'Mężczyzna'), a w przeciwnym wypadku zwróć tekst Kobieta(**ELSE** 'Kobieta'); utworzoną kolumnę nazwij Płeć (**AS Płeć**).*

Przykładowy wynik tego zapytania może mieć postać jak na rys. 12.

Nazwisko	Imie	Pesel	Płeć
Kotek	Katarzyna	92031275446	Kobieta
Lisek	Kasia	92022277654	Kobieta
Kurka	Jola	92060288788	Kobieta
Gazela	Basia	92111177446	Kobieta
Konik	Kasia	93031275446	Kobieta
Ryba	Jan	93051587746	Mężczyzna
Kura	Kasia	93022277654	Kobieta
Łoś	Jola	93060288788	Kobieta
Miś	Wacek	93031199123	Mężczyzna

Rysunek 12.
Wynik zapytania z nową kolumną Płeć

6 POLECENIE SELECT – ŁĄCZENIE TABEL

Do tej pory w zapytaniu odwoływaliśmy się do jednej tabeli, a teraz zajmiemy się zapytaniami, których tabele wynikowe będą zawierać dane z wielu tabel. Nie jest to o wiele trudniejsze. Zmieni się jedynie to, że w klauzuli **FROM** należy opisać sposób połączenia tabel, które będą brały udział w zapytaniu. Na przykład:

```
SELECT Uczniowie.* , Klasy.*
FROM Uczniowie JOIN Klasy
ON Uczniowie.Idklasy=Klasy.Idklasy
```

Sens tego zapytania można opisać w następujący sposób: *Wybrać (**SELECT**) wszystkie kolumny z tabeli Uczniowie (Uczniowie.*) oraz wszystkie kolumny z tabeli Klasy (Klasy.*), pobieraj dane z tabeli Uczniowie połączonej z tabelą Klasy (**FROM** Uczniowie **JOIN** Klasy), warunkiem połączenia jest równość wartości Idklasy w obu tabelach, czyli klucz obcy ma być równy kluczowi podstawowemu (**ON** Uczniowie.Idklasy=Klasy.Idklasy).* Przykładowy wynik takiego jest pokazany na rys. 13.

iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy	idklasy	Nazwa	RokSzkolny
2	Piesek	Jan	1992-05-15	1	92051587746	1	1	Ia	2008/2009
5	Gąska	Wacek	1991-03-11	1	91031199123	1	1	Ia	2008/2009
6	Krówka	Rysio	1992-05-15	1	92051577646	1	1	Ia	2008/2009
7	Zebra	Wojtek	1993-03-13	1	93030399846	1	1	Ia	2008/2009
9	Sarenka	Rysio	1992-12-12	0	92121278766	1	1	Ia	2008/2009
1	Kotek	Katarzyna	1992-03-12	0	92031275446	2	2	IIa	2008/2009
3	Lisek	Kasia	1992-02-22	0	92022277654	2	2	IIa	2008/2009
4	Kurka	Jola	1992-06-02	0	92060288788	2	2	IIa	2008/2009

Rysunek 13.
Wynik połączenia dwóch tabel

Przykładowe zapytanie z użyciem omówionych do tej pory operacji;

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie,
            CASE CzyChlopak
              WHEN 1 THEN 'Mężczyzna'
             ELSE 'Kobieta'
           END as Płeć,
           Klasy.Nazwa, Klasy.RokSzkolny
 FROM Uczniowie JOIN Klasy ON Uczniowie.Idklasy=Klasy.Idklasy
WHERE  YEAR(Uczniowie.DataUrodzenia)=1992
ORDER BY Płeć, Nazwisko DESC
```

Przykładowy wynik takiego zapytania jest pokazany na rys. 14.

Nazwisko	Imie	Płeć	Nazwa	RokSzkolny
Sarenka	Rysio	Kobieta	Ia	2008/2009
Orka	Kasia	Kobieta	IIc	2008/2009
Lisek	Kasia	Kobieta	IIa	2008/2009
Kurka	Jola	Kobieta	IIa	2008/2009
Kotek	Katarzyna	Kobieta	IIa	2008/2009
Gazela	Basia	Kobieta	IIa	2008/2009
Foka	Jola	Kobieta	IIc	2008/2009
Antylopa	Kasia	Kobieta	IIc	2008/2009
Rekin	Jan	Mężczyzna	IIc	2008/2009
Piesek	Jan	Mężczyzna	Ia	2008/2009
Krówka	Rysio	Mężczyzna	Ia	2008/2009

Rysunek 14.
Wynik bardziej złożonego zapytania

Kolejny przykład. Chcemy napisać zapytanie, które przygotuje wykaz uczniów (nazwisko i imię) oraz dane nauczyciela (nazwisko i imię oraz stopień zawodowy), który wystawił ocenę i datę wystawienia oceny tym uczniom, którzy w roku 2009 otrzymali z fizyki ocenę 5, wynik uporządkować malejąco według daty wystawienia oceny.

```
SELECT Uczniowie.Nazwisko+' '+Uczniowie.Imie AS Uczen,
       Nauczyciele.Nazwisko+' ' Nauczyciele.Imie AS Nauczyciel,
       Oceny.DataWystawienia, Oceny.Ocena
FROM Uczniowie JOIN Oceny ON Uczniowie.Iducznia=Oceny.IdUcznia
            JOIN Nauczyciele ON Nauczyciele.IdNauczyciela=Oceny.IdNauczyciela
            JOIN Przedmioty ON Oceny.Idprzedmiotu=Przedmioty.Idprzedmiotu
WHERE  YEAR(DataWystawienia) =2009 AND Ocena=5 AND Przedmioty.Nazwa='Fizyka'
```

W tym zapytaniu bez większego problemu zostały połączone cztery tabele. Przykładowy wynik tego zapytania jest przedstawiony na rys. 15.

Uczen	Nauczyciel	DataWystawienia	Ocena
Piesek Jan	Lew Wojciech	2009-01-05	5.00
Piesek Jan	Lew Wojciech	2009-02-09	5.00
Kotek Katarzyna	Lew Wojciech	2009-02-07	5.00
Konik Kasia	Pantera Saba	2009-01-27	5.00
Łoś Jola	Gepard Hala	2009-01-30	5.00
Konik Kasia	Gepard Hala	2009-01-27	5.00
Sarenka Rysio	Gepard Hala	2009-01-24	5.00
Okoń Rysio	Lew Wojciech	2009-01-08	5.00

Rysunek 15.
Wynik złożonego zapytania, w którym zostały połączone cztery tabele

Omawiając przykłady łączenia tabel koncentrowaliśmy się na podstawowej operacji, opartej na **złączeniu wewnętrznym** (ang. *inner join*), które powoduje że tylko wiersze, które spełniają warunek łączenia, znajdą się w tabeli wynikowej. W poprzednim przykładzie, ci uczniowie, którzy w roku 2009 nie otrzymali oceny 5 z fizyki, nie pojawią się w wyniku zapytania. W przypadku stosowania tak zwanego **połączenia zewnętrznego** (ang. *outer join*) będziemy mogli zapewnić występowanie w tabeli wynikowej nawet tych wierszy z wybranej tabeli, dla których nie jest spełniony warunek połączenia. Zademonstrujemy to na następującym przykładzie:

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie, Oceny.DataWystawienia, Ocena
FROM Uczniowie LEFT OUTER JOIN Oceny
```

ON Uczniowie.iducznia=Oceny.Iducznia **AND** Oceny.Ocena=2
AND YEAR(DataWystawienia)=2009 **AND** MONTH(DataWystawienia)=2

Nazwisko	Imie	DataWystawienia	Ocena
Kotek	Katarzyna	2009-02-05	2.00
Kotek	Katarzyna	2009-02-05	2.00
Kotek	Katarzyna	2009-02-12	2.00
Kotek	Katarzyna	2009-02-12	2.00
Piesek	Jan	NULL	NULL
Lisek	Kasia	NULL	NULL
Kurka	Jola	2009-02-07	2.00
Kurka	Jola	2009-02-07	2.00

Rysunek 16.
Wynik zapytania z użyciem połączenia zewnętrznego

Przykładowy wynik tego zapytania jest pokazany na rys. 16. W porównaniu z przykładem wcześniejszym, istotne są trzy różnice:

- W operacji łączenia wykorzystano opcję **LEFT OUTER JOIN** (lewostronne łączenie zewnętrzne) – która zapewnia, że do wyniku zapytania, oprócz wierszy spełniających warunek łączenia, zostaną dodane wiersze z tabeli po lewej stronie operatora **JOIN** (w naszym przypadku tabela Uczniowie), dla których warunek łączenia jest niespełniony.
- Warunki selekcji **AND** Oceny.Ocena=2 **AND** YEAR(DataWystawienia)=2009 **AND** MONTH(DataWystawienia)=2 zostały umieszczone w klauzuli **ON** a nie w klauzuli **WHERE**.
- W wyniku zapytania, dla tych wierszy, które nie spełniają warunku łączenia, w kolumnach DataWystawienia i Ocena, występuje warto NULL

Istnieją jeszcze inne operatory połączeń, które można wykorzystywać zamiast operatora **JOIN** (np. **APPLY** lub **PIVOT**), ale ich omówienie wykracza poza ramy tego wykładu.

7 POLECENIE SELECT – WYKORZYSTANIE FUNKCJI AGREGUJĄCYCH

Zapytania SQL mogą być także wykorzystane do wykonywania obliczeń na podstawie danych zawartych w tabelach. Do tego celu służą **funkcje agregujące**. Język SQL udostępnia pięć podstawowych funkcji agregujących:

- **COUNT** – wyznacza liczbę wierszy otrzymanych w wyniku zapytania,
- **SUM** – sumuje zawartość kolumny (lub wyrażenia obliczonego na podstawie danych) dla wszystkich wierszy w wyniku zapytania,
- **AVG** – oblicza średnią arytmetyczną zawartości kolumny (lub wyrażenia obliczonego na podstawie danych) dla wszystkich wierszy w wyniku zapytania,
- **MIN** – określa wartość minimalną dla kolumny w wyniku zapytania,
- **MAX** – określa wartość maksymalną dla kolumny w wyniku zapytania.

W różnych Systemach Zarządzania Bazami Danych mogą być dostępne jeszcze inne funkcje agregujące (głównie realizujące obliczanie wartości statystycznych), zaprezentowany zbiór pięciu funkcji jest powszechnie obowiązującym standardem.

Zapytania, które wykorzystują funkcje agregujące zwracają jeden wiersz zawierający wynik obliczeń dla danej funkcji. W pierwszym przykładzie napiszemy zapytanie, w których chcemy policzyć, ilu uczniów jest zapisanych w tabeli Uczniowie, rys. 17.

Zapytania używajace funkcji agregujących mogą wykorzystywać łączenie tabel (klauzula **FROM**) oraz warunki selekcji (klauzula **WHERE**). W kolejnym przykładzie obliczamy, ilu uczniów jest w klasie IIa, rys. 18.

```
SELECT COUNT(*) AS IluUczniow
FROM Uczniowie
```

IluUczniow
24

Rysunek 17.
Przykład użycia funkcji agregującej **COUNT** i jej wynik

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

IluUczniow
13

Rysunke 18.
Wyznaczanie liczby uczniów w klasie IIa

Jeśli chcemy w jednym zapytaniu wyznaczyć liczbę uczniów w poszczególnych klasach i wyniki umieścić w tabeli, to musimy zastosować dodatkową klauzulę **GROUP BY** (grupuj według), jak w przykładzie na rys. 19.

```
SELECT Klasy.Nazwa,
       COUNT(*) AS IluUczniow
FROM Uczniowie JOIN Klasy ON Uczniowie.idklasy=Klasy.idklasy
GROUP BY Klasy.Nazwa
```

Nazwa	IluUczniow
Ia	5
Ib	1
IIa	13
IIc	5

Rysunek 19.
Wynik zapytania z użyciem funkcji agregującej i grupowaniem wyników

Działanie klauzuli **GROUP BY** polega na zastosowaniu funkcji agregującej do każdej grupy wierszy w wyniku zapytania, które mają tę sama wartość w kolumnie (lub w kolumnach, bo można podać w tej klauzuli listę kolumn) podanej jako parametr grupowania. W naszym przykładzie kolumną, według której są grupowane dane, jest Nazwa z tabeli Klasy, czyli funkcja **COUNT** zlicza ilość wierszy dla każdej klasy oddzielnie.

W kolejnym zapytaniu chcemy otrzymać listę uczniów z klasy IIa oraz ich średnią ocen otrzymanych w roku 2009, rys. 20:

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie, AVG(Oceny.Ocen) as Średnia
FROM Uczniowie JOIN Oceny ON Uczniowie.Iducznia=Oceny.IdUcznia
                JOIN Uczniowie.Idklasy=Klasy.Idklasy
WHERE YEAR(Oceny.DataWystawienia)=2009 AND Klasy.Nazwa='IIa'
GROUP BY Uczniowie.Nazwisko, Uczniowie.Imie
ORDER BY Średnia
```

Nazwisko	Imie	Średnia
Łoś	Jola	4.714285
Konik	Kasia	4.000000
Kura	Kasia	3.111111
Różyczka	Basia	3.000000
Kurka	Jola	3.000000
Kotek	Katarzyna	2.958333
Lisek	Kasia	2.866666
Okoń	Rysio	2.857142
Gazela	Basia	2.800000
Miś	Wacek	2.777777
Płotka	Wojtek	2.777777
Ryba	Jan	2.500000

Rysunek 20.
Wynik zapytania ze średnimi ocenami uczniów

Język SQL udostępnia jeszcze jedną klauzulę wykorzystywaną przy grupowaniu z zastosowaniem funkcji agregujących. Przypuśćmy, że w poprzednim zapytaniu chcielibyśmy otrzymać listę tylko tych uczniów, dla których obliczona średnia ocena jest większa od 3.00. Takiego warunku selekcji nie możemy jednak zapisać w klauzuli **WHERE**, ponieważ jest ona wykonywana w momencie, gdy nie jest znany jeszcze wynik obliczeń funkcji agregującej. Potrzebujemy więc sprawdzenia warunku Średnia › 3.00 dopiero wtedy, gdy znane będą wartości średnie. Klauzula **HAVING**, dzięki której rozwiążemy ten problem, nazywana jest **opóźnionym warunkiem selekcji** i jest wykorzystywana do selekcji według wartości obliczonych przez funkcje agregujące. Zmodyfikujemy nasz przykład i zastosujemy klauzulę **HAVING**:

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie, AVG(Oceny.Ocen) as Średnia
FROM Uczniowie JOIN Oceny ON Uczniowie.Iducznia=Oceny.IdUcznia
                    JOIN Uczniowie.Idklasy=Klasy.Idklasy
WHERE  YEAR(Oceny.DataWystawienia)=2009 AND Klasy.Nazwa='IIa'
GROUP BY Uczniowie.Nazwisko, Uczniowie.Imie
HAVING AVG(Oceny.Ocena) › 3.00
ORDER BY Średnia
```

Nazwisko	Imie	Średnia
Łoś	Jola	4.714285
Konik	Kasia	4.000000
Kura	Kasia	3.111111

Rysunek 21.
Wynik zapytania z selekcją niektórych uczniów

8 POLECENIE SELECT – ZAPYTANIA ZŁOŻONE

Polecenie **SELECT** języka SQL umożliwia **zagnieżdżanie zapytań**, czyli wykorzystanie zapytania w wewnątrz innego zapytania. Dzięki tej właściwości można za pomocą jednego polecenia wykonywać bardzo złożone operacje na danych. Omówimy to chcąc przygotować listę uczniów, zawierającą nazwisko i imię ucznia oraz nazwę klasy, którzy w roku 2009 nie otrzymali oceny niedostatecznej z fizyki. Należy zwrócić uwagę, że chcemy pobrać z bazy dane, które nie są bezpośrednio w niej zapisane, bo jeżeli uczeń nie otrzymał oceny to w bazie danych nie ma żadnego zapisu tego faktu. Rozwiązując ten problem korzystamy z pewnych zależności logicznych. Pomyślmy o tym problemie jako o działaniu na następujących zbiorach:
A – zbiór wszystkich uczniów,
B – zbiór uczniów, którzy otrzymali w roku 2009 ocenę niedostateczną z fizyki,
C – poszukiwany zbiór uczniów, którzy w roku 2009 nie otrzymali oceny niedostatecznej z fizyki.

Wyrażenie: **C = A – B** opisuje rozwiązanie naszego problemu, czyli poszukiwany zbiór możemy otrzymać jako różnicę dwóch innych zbiorów.

Zbiór A, czyli zbiór wszystkich uczniów, możemy otrzymać za pomocą prostego zapytania, które przygotuje zbiór uczniów zawierający nazwisko i imię ucznia oraz nazwę klasy:

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie, Klasy.Nazwa,
FROM Uczniowie JOIN Klasy ON Uczniowie.idklasy=Klasy.idklasy
```

Zbiór B, czyli zbiór uczniów, którzy otrzymali ocenę niedostateczną z fizyki otrzymamy za pomocą innego zapytania:

```
SELECT DISTINCT Iducznia
FROM Oceny JOIN Przedmioty ON Oceny.Idprzedmiotu=Przedmioty.Idprzedmiotu
WHERE Przedmioty.Nazwa='Fizyka' AND YEAR(Oceny.DataWystawienia)=2009
AND Oceny.Ocena=2
```

Działanie nowej opcji **DISTINCT** polega na eliminowaniu w wyniku zapytania powtarzających się wierszy (jeżeli uczeń otrzymał kilka ocen niedostatecznych z fizyki w roku 2009, to w zbiorze wynikowym jego identyfikator (Iducznia) pojawiłby się wiele razy).

Pozostaje nam teraz zrealizować operację różnicy zbiorów;

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie, Klasy.Nazwa,
FROM Uczniowie JOIN Klasy
    ON Uczniowie.idklasy=Klasy.idklasy                    <-- Zbór A
WHERE Iducznia NOT IN                                     <-- Operator różnicy zbiorów
        (SELECT DISTINCT Iducznia
          FROM Oceny JOIN Przedmioty
            ON  Oceny.Idprzedmiotu=Przedmioty.Idprzedmiotu    <-- Zbór B
             WHERE Przedmioty.Nazwa='Fizyka' AND
               YEAR(Oceny.DataWystawienia)=2009 AND Oceny.Ocena=2)
```

Pokazaliśmy jeden przykład zapytania złożonego, ukazujący dodatkowe mozliwości, jakimi dysponujemy przy pisaniu zapytań do baz danych z wykorzystaniem języka SQL. Trudno wymienić wszystkie sytuacje, w których można wykorzystywać podzapytania, ale jest jedna zasada ogólna:

Podzapytanie może być wykorzystane wszędzie tam, gdzie ma sens wynik tego podzapytania

Ponieważ podstawową formą wyników zapytań jest tabela, to możemy podzapytania zwracajace tablę wykorzystywać zamiast fizycznych tabel będących częcią bazy danych, na przykład w klauzuli **FROM**.

Zapytania mogą zwracać także jedną wartość, gdy zapytanie zwraca jedną kolumnę i w wyniku powstaje jeden wiersz, to w takim przypadku możemy umieścic podzapytanie wszędzie tam, gdzie ma sens taka wartość.

```
SELECT AVG(YEAR(GETDATE()) – YEAR(Uczniowie.DataUrodzenia)) as ŚredniWiek
FROM Uczniowie
```

Takie zapytanie powinno zwrócić średni wiek uczniów zapisanych w tabeli Uczniowie (funkcja **AVG** oblicza średnią arytmetyczna wyrażenia, w którym od roku pobranego z daty systemowej (**GETDATE**()) odejmujemy rok pobrany z daty urodzenia), czyli zapytanie zwróci jedną wartość i moglibyśmy taką postać zapytania wykorzystać w innym, kótre ma zwrócić dane uczniów, których wiek jest poniżej średniego wieku wszystkich uczniów.

Nie byłoby problemu gdybyśmy chcieli wybrać uczniów, których wiek jest mniejszy od pewnej danej wartości np. 17.5, wtedy zapytanie miałoby postać:

```
SELECT Nazwisko,Imie,Pesel
FROM Uczniowie
WHERE (YEAR(GETDATE()) –  YEAR(Uczniowie.DataUrodzenia) ‹ 17.5
```

Ponieważ chcemy, żeby wiek ucznia był mniejszy od średniego wieku wszystkich uczniów, to musimy konkretną wartość (17.5) zastąpić zapytaniem obliczającym tę średnią, czyli tak jak poniżej:

```
SELECT Nazwisko,Imie,Pesel
FROM Uczniowie
WHERE (YEAR(GETDATE()) – YEAR(Uczniowie.DataUrodzenia)
      ‹ (SELECT AVG(YEAR(GETDATE()) – YEAR(Uczniowie.DataUrodzenia)) as ŚredniWiek
        FROM  Uczniowie)
```

Przytoczyliśmy kilka przykładów zapytań złożonych, żeby pokazać dodatkowe możliwości języka SQL budowania takich zapytań.

9 POLECENIE SELECT – CO JESZCZE POTRAFIĘ

Omówiliśmy jedynie elementarne podstawy zapytań realizowanych w języku SQL, które mogą stanowić dobrą podstawę do rozpoczęcia nauki pisania zapytań w tym języku. Wraz z kolejnymi wersjami standardu języka SQL pojawiają się nowe możliwość. W tej części wykładu omówimy niektóre nowe możliwości polecenia **SELECT** języka SQL.

1. Tworzenie wyniku zapytania w języku XML

   Poniższe zapytanie przygotowuje listę klas pobierając dane z tabeli Klasy. Dodanie do tego zapytania klauzuli **FOR XML** (**FOR XML AUTO,ROOT**('ListaKlas'),**ELEMENTS**) powoduje, że wynik zapytania zamiast w postaci tabeli jest zwracany jako dokument XML (XML – jest obecnie bardzo rozpowszechnionym standardem wymiany danych pomiędzy różnymi systemami).

```
SELECT Klasy.Nazwa, Klasy.RokSzkolny
FROM Klasy
FOR XML AUTO,ROOT('ListaKlas'),ELEMENTS
```

Wynik takiego zapytania ma następującą postać – z pewnoscią nie jest to tabela.

```
‹ListaKlas›
 ‹Klasy›
  ‹Nazwa›Ia‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasy›
 ‹Klasy›
  ‹Nazwa›IIa‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasy›
 ‹Klasy›
  ‹Nazwa›Ib‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasy›
 ‹Klasy›
  ‹Nazwa›IIb‹/Nazwa›
  ‹RokSzkolny›2008/2009‹/RokSzkolny›
 ‹/Klasy›
‹/ListaKlas›
```

2. Operacje na zbiorach danych z wykorzystaniem operatorów **UNION**, **EXCEPT** i **INTERSECT**

Tabela relacyjna jest zbiorem danym, możemy więc wykonywać na tych zbiorach różne operacje, w tym te najprostsze, czyli branie sumy, różnicy i części wspólnej zbiorów. A zatem na wynikach dwóch zapytań (są to zbiory) możemy wykonywać wymienione operacje. Język SQL udostępnia trzy operatory do działania na zbiorach:

- **UNION** – sumuje dwa zbiory

  Zapytanie, które przygotuje listę uczniów z klasy o wartości idklasy=1 oraz z klasy o idklasy=2 może mieć postać:

```
SELECT Nazwisko, Imie, Pesel FROM Uczniowie
WHERE idklasy=1
                UNION
                  SELECT Nazwisko, Imie, Pesel FROM Uczniowie
WHERE idklasy=2
```

- **EXCEPT** – tworzy różnicę dwóch zbiorów

  Zapytanie, które przygotuje listę uczniów z klasy o idklasy=1 za wyjątkiem tych, którzy urodzili się w marcu, ma postać:

```
SELECT Nazwisko, Imie, Pesel FROM Uczniowie
WHERE idklasy=1
EXCEPT
SELECT Nazwisko, Imie, Pesel FROM Uczniowie
WHERE MONTH(DataUrodzenia)=3
```

- **INTERSECT** – tworzy część wspólna zbiorów

  Zapytanie, które przygotuje listę uczniów urodzonych w marcu, których nazwisko zaczyna się na literę K ma postać:

```
SELECT Nazwisko, Imie, Pesel FROM Uczniowie
WHERE MONTH(DataUrodzenia)=3
INTERSECT
SELECT Nazwisko, Imie, Pesel FROM Uczniowie
WHERE nazwisko LIKE 'K%'
```

W zapytaniach wykonujących operacje na zbiorach dla wyników dwóch innych zapytań musi być spełniony warunek: oba zapytania musza zwracać tabele z taką samą liczbę kolumn a w tych kolumnach muszą być wartości tego samego typu.

3. Tworzenie tabel przestawnych

   Tabele przestawne są bardzo przydatne przy tworzeniu różnych zestawień, gdy pewne wartości znajdują się na przecięciu wiersza i kolumny. W poniższym przykładzie (patrz również wynik na rys. 22) jest tworzona tabela przestawna, która w pierwszej kolumnie wyświetla listę przedmiotów a kolejne kolumny reprezentują różne klasy – obliczane wartości dla klas są średnią ocen z danego przedmiotu dla wszystkich uczniów danej klasy. Wartości NULL występują wtedy, gdy w danej klasie nikt nie otrzymał oceny z danego przedmiotu.

```
SELECT *
 FROM
(
SELECT Przedmioty.Nazwa as Przedmiot,
    Klasy.Nazwa as Klasa,
    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
```

```
) as A
PIVOT
    (AVG(Ocena) FOR Klasa in ([Ia],[IIa],[IIc])) as B
```

Przedmiot	Ia	IIa	IIc
Fizyka	2.746987	2.960227	NULL
Geografia	2.918032	3.021978	4.333333
Informatyka	2.881578	2.848314	NULL
Literatura	NULL	3.125000	NULL
Matematyka	2.919354	3.060439	NULL

Rysunek 22.
Tabela przestawna

Prezentowana postać zapytania wykonującego tabelę przestawną z pewnością nie jest prosta i w pełni zrozumiała. Na zakończenie wykładu chcieliśmy pokazać coś, co będzie wymagało od uczestnika pewnego wysiłku, a w literaturze bądź w Internecie można znaleźć pełniejsze wyjaśnienie, jakie jest działanie powyższego zapytania.

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

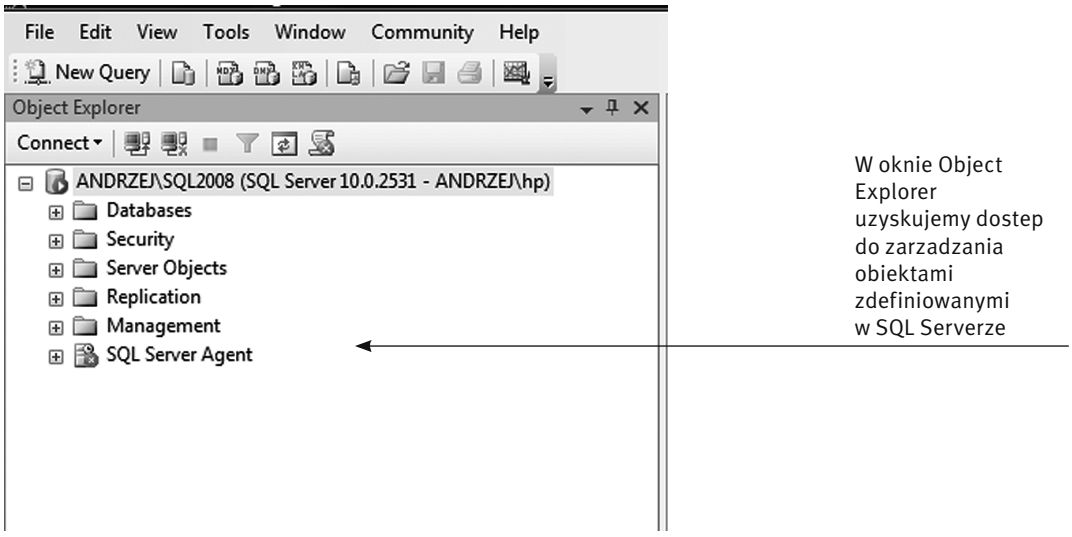
WARSZTATY

Ćwiczenie 1. Zapoznanie się ze środowiskiem MS SQL Server 2008 i bazą danych ElektronicznyDziennikOcen
Wspólnie z prowadzącym poznajemy środowisko Systemu Zarządzania Bazami Danych MS SQL Server 2008. Korzystanie z MS SQL Server 2008 umożliwia specjalne oprogramowanie SQL Server Menagement Studio.

1. Uruchamiamy SQL Server Menagement Studio (lokalizacje programu poda prowadzący warsztaty.
2. Po uruchomieniu programu, logujemy się do SQL Servera – w okienku do logowania wpisujemy w pola wartości takie, jak podano na rys. 23.

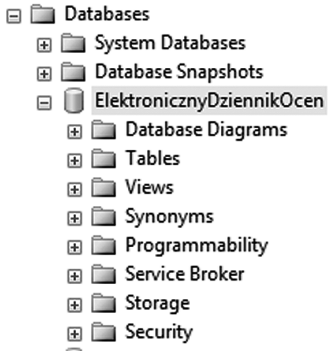
Rysunek 23.
Logowanie do SQL Servera 2008

3. Po poprawnym zalogowaniu się, pojawia się okna SQL Server Menagement Studio, rys. 24.



Rysunek 24.
Okno Object Explorer

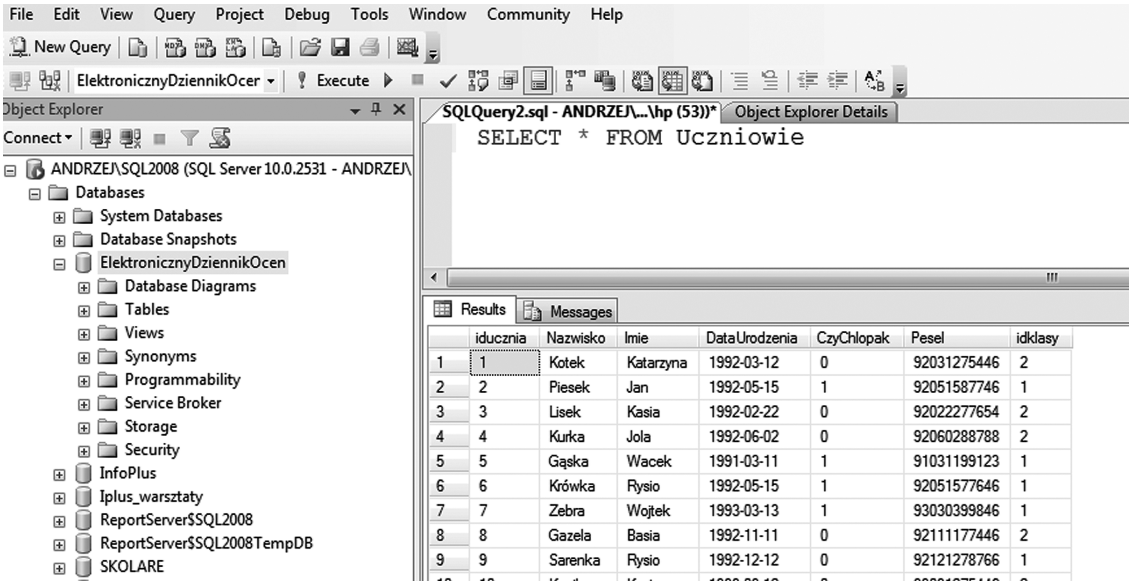
4. Wspólnie z prowadzącym poznajemy wybrane elementy środowiska SQL Servera.
5. Rozwijamy folder Databases i wybieramy bazę danych ElektronicznyDziennikOcen – otrzymamy widok folderów związanych z wybraną bazą danych.



Rysunek 25.
Foldery bazy danych ElektronicznyDziennikOcen

6. Wspólnie z prowadzącym zapoznajemy się z elementami bazy danych.
7. Aby wykonać zapytanie do bazy klikamy na przycisku New Query (nowe zapytanie) – uruchamia się okienko edycyjne, w którym będziemy wpisywać zapytania, rys. 26.

Wpisujemy zapytanie; **SELECT * FROM** Uczniowie, wybierające wszystkie wiersze z tabeli Uczniowie, i klikamy na przycisku (poleceniu) Execute lub naciskamy klawisz F5, patrz rys. 26. W wyniku otrzymujemy tabelę z zapisanymi danymi uczniów.



	iducznia	Nazwisko	Imie	DataUrodzenia	CzyChlopak	Pesel	idklasy
1	1	Kotek	Katarzyna	1992-03-12	0	92031275446	2
2	2	Piesek	Jan	1992-05-15	1	92051587746	1
3	3	Lisek	Kasia	1992-02-22	0	92022277654	2
4	4	Kurka	Jola	1992-06-02	0	92060288788	2
5	5	Gąska	Wacek	1991-03-11	1	91031199123	1
6	6	Krówka	Rysio	1992-05-15	1	92051577646	1
7	7	Zebra	Wojtek	1993-03-13	1	93030399846	1
8	8	Gazela	Basia	1992-11-11	0	92111177446	2
9	9	Sarenka	Rysio	1992-12-12	0	92121278766	1

Rysunek 26.
Proste zapytanie do bazy danych ElektronicznyDzienikOcen i jego wynik

Ćwiczenie 2. Proste zapytania skierowane do jednej tabel

1. Rozwijamy folder Databases i wybieramy bazę danych o nazwie ElektronicznyDziennikOcen.
2. Przechodzimy do edytora zapytań naciskając przycisk New Query.
3. W oknie edytora wpisujemy treść zapytania i naciskamy klawisz F5.
4. Napisać i wykonać następujące zapytanie:

SELECT Nazwisko, Imie, DataUrodzenia
FROM Uczniowie
WHERE YEAR(DataUrodzenia)=1992
Funkcja YEAR z wartości daty wybiera rok.

5. Zadanie do samodzielnego wykonania (z pomocą prowadzącego):
   Napisać zapytanie, które przygotuje tabelę, zawierającą następujące dane: nazwisko i imię ucznia, jego datę urodzenia i numer Pesel dla tych uczniów, którzy urodzili się później niż 31-12-1992. Wynik zapytania uporządkować malejąco według daty urodzenia.

Ćwiczenie 3. Zapytania wykorzystujące łączenie tabel

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. Napisać w oknie edycyjnym i wykonać następujące zapytanie (naciskając klawisz F5):

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie,
       Przedmioty.Nazwa,
       Oceny.DataWystawienia,
      Oceny.Ocena
FROM Uczniowie JOIN Oceny ON Uczniowie.Iducznia=Oceny.IdUcznia
               JOIN Przedmioty ON Przedmioty.Idprzedmiotu=Oceny.Idprzedmiotu
WHERE YEAR(Oceny.DataWystawienia)=2009 AND Oceny.Ocena<3
        AND Przedmioty.Nazwa='Matematyka'
ORDER BY Oceny.DataWystawienia
```

Jakie dane zawiera tabela będąca wynikiem tego zapytania?

4. Zadanie do samodzielnego wykonania (z pomocą prowadzącego):
   Napisać zapytanie, które przygotuje tabelę zawierającą następujące dane; nazwisko i imię ucznia, nazwę przedmiotu, datę wystawienia oceny oraz wartość tej oceny dla ocen z fizyki wystawionych uczniom klasy IIa w roku 2009. Wynik zapytania uporządkować malejąco według daty wystawienia oceny.

Ćwiczenie 4. Zapytania wykorzystujące funkcje agregujące

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. Napisać w oknie edycyjnym i wykonać następujące zapytanie (naciskając klawisz F5):

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie,
        COUNT(*) as IleNiedostatecznych
FROM Uczniowie join Oceny
   ON Uczniowie.iducznia=Oceny.iducznia
WHERE YEAR(Oceny.DataWystawienia)=2009
GROUP BY Uczniowie.Nazwisko, Uczniowie.Imie
ORDER BY IleNiedostatecznych DESC
```

Jakie dane zawiera tabela będąca wynikiem tego zapytania?

4. Zadanie do samodzielnego wykonania
   Napisać zapytanie, które przygotuje tabelę zawierającą następujące dane; nazwę przedmiotu oraz średnią ocen wystawionych uczniom klasy IIa w roku 2009 z poszczególnych przedmiotów. W wyniku zapytania powinny znaleźć się tylko te przedmioty, dla których wartość średniej jest niższa od 3.5. Wynik zapytania uporządkować malejąco względem wartości średniej oceny.

Ćwiczenie 5. Zapytania złożone

1. W folderze Databases wybrać bazę danych ElektronicznyDziennikOcen.
2. Przejść do edytora zapytań naciskając przycisk New Query.
3. Napisać w oknie edycyjnym i wykonać następujące zapytanie (wciskając klawisz F5):

```
SELECT Uczniowie.Nazwisko, Uczniowie.Imie, Klasy.Nazwa,
FROM Uczniowie JOIN Klasy
     ON Uczniowie.idklasy=Klasy.idklasy
```

```
WHERE Iducznia NOT IN
                (SELECT DISTINCT Iducznia
                FROM Oceny JOIN Przedmioty
                        ON  Oceny.Idprzedmiotu=Przedmioty.Idprzedmiotu
                    WHERE Przedmioty.Nazwa='Fizyka' AND
                        YEAR(Oceny.DataWystawienia)=2009 AND Oceny.Ocena=2)
```

Jakie dane zawiera tabela będąca wynikiem tego zapytania?

4. Zadanie do samodzielnego wykonania.
   Napisać zapytanie, które przygotuje tabelę zawierającą następujące dane: nazwisko i imię nauczyciela dla tych nauczycieli, którzy w marcu 2009 nie wystawili oceny niedostatecznej uczniom klasy IIa. . Wynik zapytania uporządkować rosnąco według nazwiska nauczyciela.

Ćwiczenie 6. Co jeszcze potrafię

Zadanie do samodzielnego wykonania. Napisać zapytanie, które przygotuje listę uczniów którzy otrzymali ocenę 5 z fizyki w marcu 2009, z wyjątkiem tych uczniów, którzy otrzymali w tym samym miesiącu ocenę 2 z matematyki.

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