Moduł szkoleniowy SQL Structured Query Language

poziom podstawowy 15 godzinny

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Spis treści

1 Wprowadzenie 4
2 Cel 4
3 Opis sposobu realizacji celów 4
4 Treści kształcenia 5
5 Opis założonych osiągnięć ucznia 5
5.1 Korelacja z treściami podstawy programowej 5
6 Sposoby osiągania celów 6
6.1 Przykładowe tematy projektu: 6
6.2 Lista czynności w zależności od roli 7
7 Propozycje kryteriów oceny i metod sprawdzania osiągnięć ucznia 7
7.1 Opis założonych osiągnięć ucznia – przykłady wymagań na poszczególne oceny szkolne 7
7.2 Metody sprawdzania osiągnięć ucznia 8
8 Test końcowy sprawdzający wiedzę 8
9 Lekcje 10
9.1 Lekcja 1 - Bazy danych, podstawowe pojęcia (tabela, dana, wiersz, relacja, klucz podstawowy, klucz obcy, sortowanie, zapytanie, system zarządzania bazą danych), interakcje z bazą danych (Język SQL, aplikacja, formularz raport) 10
9.1.1 Cel lekcji 10
9.1.2 Treść - slajdy z opisem 11
9.1.3 Ćwiczenia 27
9.1.4 Opis założonych osiągnięć ucznia 27
9.2 Lekcja 2 - Języki programowania serwerów SQL, środowisko programowania w języku SQL, wprowadzenie do języka SQL 27
9.2.1 Cel lekcji 27
9.2.2 Treść - slajdy z opisem 28
9.2.3 Ćwiczenia 39
9.2.4 Opis założonych osiągnięć ucznia 39
9.3 Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych 40
9.3.1 Cel lekcji 40
9.3.2 Treść - slajdy z opisem 40
9.3.3 Ćwiczenia 56
9.3.4 Opis założonych osiągnięć ucznia 56
9.4 Lekcja 4 – DML 56
9.4.1 Cel lekcji 56
9.4.2 Treść - slajdy z opisem 57
9.4.3 Ćwiczenia 66

9.4.4 Opis założonych osiągnięć ucznia 66
9.5 Lekcja 5 - DDL 66
9.5.1 Cel lekcji 66
9.5.2 Treść - slajdy z opisem 66
9.5.3 Ćwiczenia 80
9.5.4 Opis założonych osiągnięć ucznia 80
9.6 Lekcja 6 - DCL, uprawnienia, użytkownicy 80
9.6.1 Cel lekcji 80
9.6.2 Treść - slajdy z opisem 80
9.6.3 Ćwiczenia 96
9.6.4 Opis założonych osiągnięć ucznia 96
9.7 Lekcja 7 - Widoki i Funkcje 96
9.7.1 Cel lekcji 96
9.7.2 Treść - slajdy z opisem 96
9.7.3 Ćwiczenia 111
9.7.4 Opis założonych osiągnięć ucznia 111
9.8 Lekcja 8 - Procedury Triggery 111
9.8.1 Cel lekcji 111
9.8.2 Treść - slajdy z opisem 112
9.8.3 Ćwiczenia 125
9.8.4 Opis założonych osiągnięć ucznia 125
9.9 Lekcja 9 - Indeksy, constrainty 125
9.9.1 Cel lekcji 125
9.9.2 Treść - slajdy z opisem 126
9.9.3 Ćwiczenia 143
9.9.4 Opis założonych osiągnięć ucznia 143
9.10 Lekcja 10 - Transakcje, zatwierdzanie zmian, obsługa błędów, wprowadzenie do projektowania bazy danych 143
9.10.1 Cel lekcji 143
9.10.2 Treść - slajdy z opisem 143
9.10.3 Ćwiczenia 156
9.10.4 Opis założonych osiągnięć ucznia 156
9.11 Lekcja 11 – Interdyscyplinarny projekt książki teleadresowej 156
9.11.1 Cel lekcji 156
9.11.2 Treść - slajdy z opisem 157
9.11.3 Opis założonych osiągnięć ucznia 161

1 Wprowadzenie

Język SQL ("Structure Query Language") to strukturalny język zapytań służący do budowania systemów obsługi relacyjnych baz danych (RDBMS) na dowolnej platformie sprzętowej. Obecnie jest on standardowym językiem służącym do obsługi komercyjnych baz danych.
SQL został opracowany w latach 70. w firmie IBM. Szybko stał się standardem w komunikacji z serwerami baz danych. Wiele współczesnych systemów relacyjnych baz danych używa do komunikacji z użytkownikiem SQL, dlatego potocznie mówi się, że korzystanie z relacyjnych baz danych to korzystanie z SQL-a.
Pionierską firmą, która rozpoczęła dystrybucję komercyjnych baz danych, był Oracle. Dalsze wprowadzanie SQL-a, w produktach innych firm, wiązało się nierozłącznie z wprowadzaniem modyfikacji pierwotnego języka.
Pierwotną nazwą języka miał być SEQUEL, jednakże okazało się, że nazwa ta była już zastrzeżona przez brytyjską wytwórnię lotniczą Hawker Siddeley.
Język SQL umożliwia zadawanie pytań do baz danych oraz pozwala na zmienianie struktury baz danych. W opracowaniu tym omawiamy tylko ten fragment języka który służy do zadawania pytań.
Systemy informatyczne od zawsze budowane były z myślą o przetwarzaniu elementarnych informacji, które odpowiednio zamodelowane formułowały odpowiedzi na pytania naukowców, analityków, przedsiębiorców biznesu. Bazy danych są zatem nieodłącznym elementem każdego systemu informatycznego. Żadna, nawet najbardziej użyteczna i wysokowydajna technologia, nie odniesie sukcesu, jeżeli nie będzie dostarczać informacji i wskazówek dla końcowego użytkownika. Dlatego znajomość języka manipulacji na rekordach baz danych – SQL kreuje się na pozycję obowiązkową w umiejętnościach każdej osoby, związanej z branżą IT.
W poszczególnych częściach kursu szczegółowo zostaną opisane podstawowe elementy składni języka zapytań SQL. Zapoznanie się z ich treścią pozwoli na swobodną pracę z tymże językiem w ramach dowolnego systemu zarządzania bazą danych.

2 Cel

Poznanie zagadnień z zakresu relacyjnych baz danych w przykładowym profesjonalnym środowisku relacyjnego systemu zarządzania bazą danych, jakim jest system produkcyjny Microsoft SQL Server 2008. Praktyczne zapoznanie się ze środowiskiem. Przyswojenie dedykowanej terminologii w języku polskim i angielskim. Nabycie umiejętności manipulacji zarówno bazami danych, ich strukturami, jak i przechowywanymi w nich danymi za pomocą języka SQL.

3 Opis sposobu realizacji celów

10 półtoragodzinnych lekcji składających się z przypomnienia wiedzy z poprzedniej lekcji, przedstawienia materiału wraz z przykładami, ćwiczeń praktycznych. Po zakończeniu całego cyklu - przeprowadzenie egzaminu sprawdzającego wiedzę.
Dodatkowo wprowadzono lekcję 11 zawierającą część projektu interdyscyplinarnego łączącego 4 dziedziny wiedzy z zakresu informatyki (SQL, PHP, JavaScript oraz JAVA) w zakresie zaprojektowania bazy danych oraz obiektów służących jej obsłudze.

4 Treści kształcenia

Treść kursu została podzielona na 10 bloków tematycznych – po jednym do każdej lekcji:

1. Bazy danych, podstawowe pojęcia (tabela, dana, wiersz, relacja, klucz podstawowy, klucz obcy, sortowanie, zapytanie, system zarządzania bazą danych), interakcje z bazą danych (Język SQL, aplikacja, formularz raport)
2. Języki programowania serwerów SQL, środowisko programowania w języku SQL, wprowadzenie do języka SQL
3. Złożone zapytania, grupowanie, agregacja danych
4. DDL
5. DML
6. DCL, uprawnienia, użytkownicy
7. Widoki i Funkcje
8. Procedury i wyzwalacze (triggers)
9. Indeksy i ograniczenia (constraints)
10. Transakcje, zatwierdzanie zmian, obsługa błędów, normalizacja bazy danych
11. Projekt interdyscyplinarny – baza teleadresowa kolegów i koleżanek ze szkoły

5 Opis założonych osiągnięć ucznia

Uczeń po odbyciu kursu pozna podstawy teoretyczne związane z bazami danych, pozna techniki projektowania baz danych, nauczy się praktycznego programowania baz danych w środowisku Microsoft SQL Serwer 2008.

5.1 Korelacja z treściami podstawy programowej

Cele kształcenia	Treść kształcenia	Podstawa programowa
Uczeń poznaje podstawowe typy danych	Lekcja 5	Uczeń korzysta z wbudowanych typów danych
Uczeń tworzy procedury, funkcje, triggery oraz constrainty	Lekcje 8 i 9	Uczeń tworzy własne typy danych
Uczeń poznaje transakcje bazodanowe Uczeń potrafi obsługiwać błędy Uczeń prawidłowo wykonuje zapytania SQL Uczeń potrafi modyfikować dane Uczeń potrafi modyfikować struktury danych Uczeń tworzy i modyfikuje obiekty bazy danych	Lekcje: 3, 4, 5, 6, 7, 8, 9	Uczeń przestrzega zasad programowania
Uczeń tworzy widoki, procedury, funkcje, triggery oraz constrainty	Lekcje 8 i 9	Uczeń stosuje instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania
Uczeń tworzy widoki, procedury, funkcje, triggery oraz constrainty	Lekcje 8 i 9	Uczeń tworzy własne funkcje, procedury, obiekty, metody wybranych języków programowania
Uczeń potrafi zainstalować oraz poprawnie	Lekcje: 2	Uczeń wykorzystuje środowisko

Cele kształcenia	Treść kształcenia	Podstawa programowa
skonfigurować środowisko pracy		programistyczne: edytor, kompilator, debuger
Uczeń projektuje bazy danych dla aplikacji (w tym webowych)	Lekcje: 10	Uczeń wykorzystuje języki programowania do tworzenia aplikacji internetowych realizujących zadania po stronie serwera
Uczeń poznaje mechanizmy tworzenia i modyfikacji danych	Wszystkie lekcje modułu	Uczeń pobiera dane aplikacji i przechowuje je w bazie danych
Uczeń potrafi testować zapytania Uczeń poznaje instrukcje modyfikujące obiekty bazodanowe	Lekcje: 2, 3, 4, 5, 6, 7, 8, 9	Uczeń testuje tworzoną aplikację i modyfikuje jej kod źródłowy
Uczeń potrafi komentować kod	Lekcje: 3	Uczeń dokumentuje tworzoną aplikację
Uczeń poznaje podstawy projektowania baz danych Uczeń przenosi bazy danych	Lekcje: 1,2, 10	Uczeń zamieszcza opracowane aplikacje w Internecie
Uczeń poznaje zagadnienia dotyczące bezpieczeństwa, transakcji oraz obsługi błędów	Lekcje: 10	Uczeń zabezpiecza dostęp do tworzonych aplikacji

6 Sposoby osiągania celów

Po odbyciu kursu uczeń powinien wykonać projekt/zadanie, które zmotywuje go do pracy indywidualnej ze środowiskiem programistycznym i bazą danych. Zadanie utrwali zdobytą na kursie wiedzę i zmusi do wykorzystania zdobytej wiedzy teoretycznej w praktyce. Zadanie powinno być sformułowane w sposób otwarty, tak aby każdy z uczniów mógł wybrać coś dla niego interesującego. Powinna też zostać dostarczona lista możliwych tematów projektu do wyboru - dla osób, które nie będą miały pomysłu na projekt.

6.1 Przykładowe tematy projektu:

1. Projekt bazy danych placówki medycznej, z możliwością zapisów na badania (główne tabele: pacjent, lekarz, badanie, wizyta). Opcjonalnie: procedura składowana służąca do zapisów na badanie
2. Projekt bazy danych działu kadr w firmie (główne tabele: pracownik, dział, wypłaty). Opcjonalnie: procedura składowana służąca wyliczenia pensji na podstawie ilości przepracowanych godzin
3. Projekt bazy danych rezerwacji biletów w kinie (główne tabele: sala, film, seans, bilet). Opcjonalnie: procedura składowana służąca rezerwacji biletów

6.2 Lista czynności w zależności od roli

Wykonywane czynności są takie same dla każdej z lekcji, jedynie materiał którego dotyczą jest różny. Może się zdarzyć iż w przypadku barku materiału w danej lekcji dla danej czynności, czynność nie powinna być wykonana.

Czynności nauczyciela	Czynności ucznia
Prezentowanie przygotowanego materiału.	Sporządzanie notatek z prezentowanego materiału
Zadawanie pytań kontrolnych odnośnie już pokazanego materiału	Udzielanie odpowiedzi na pytania nauczyciela
Zadawanie pytań odnośnie materiału, który dopiero będzie pokazany, jeśli możliwe jest uzyskanie odpowiedzi w drodze dedukcji	
Objaśnianie ćwiczeń lekcyjnym, pilnowanie czasu ich wykonania przez uczniów, udzielanie wskazówek	Rozwiązywanie zadań lekcyjnych
Objaśnienie zadań na koniec lekcji, udzielanie wskazówek podczas rozwiązywania, zlecenie dokończenia zadań jako pracę domową	Odrabianie zadań domowych
Sprawdzanie ćwiczeń lekcyjnych i zadań domowych: należy zwrócić uwagę na poprawne formatowanie kodu (wcięcia) poprawne użycie konstrukcji języka dokumentowanie kodu komentarzami otrzymywanie poprawnych rezultatów	
Zapisywanie uwag odnośnie materiałów szkoleniowych i sugestia ewentualnych korekt.	

7 Propozycje kryteriów oceny i metod sprawdzania osiągnięć ucznia

7.1 Opis założonych osiągnięć ucznia – przykłady wymagań na poszczególne oceny szkolne

Ocenę **dopuszczającą (2)** otrzymuje uczeń, który potrafi skonstruować proste, niezłożone zapytania SQL: SELECT, INSERT i UPDATE, zna podstawowe pojęcia związane z bazami danych i silnikami baz danych, potrafi własnymi słowami wyjaśnić podstawowe instrukcje języka.
Ocenę **dostateczną (3)** otrzymuje uczeń, który spełnia wymagania oceny dopuszczającej oraz potrafi stworzyć zapytania SQL zawierające kilka warunków. Zna podstawowe struktury danych w języku SQL. Wykonuje ćwiczenia z niewielką pomocą nauczyciela. Potrafi wskazać podstawowe zastosowania baz danych.
Ocenę **dobrą (4)** otrzymuje uczeń, który spełnia wymagania na ocenę dostateczną oraz potrafi stworzyć skrypt złożony z kilku zapytań. Potrafi łączyć tabele za pomocą instrukcji JOIN. Uczeń powinien być w stanie stworzyć własne obiekty baz danych takie jak widoki i funkcje. Na lekcjach wykonuje samodzielnie zadane ćwiczenia. Umiejętnie korzysta z mechanizmów pozwalających na dbanie o jednorodność i spójność danych (np. klucze i ograniczenia).
Ocenę **bardzo dobrą (5)** otrzymuje uczeń, który spełnia wymagania na ocenę dobrą oraz bardzo dobrze zna teorię omawianą na zajęciach, w trakcie wykonywania ćwiczeń wykazuje inicjatywę, potrafi wskazać kilka rozwiązań zadania, tworzy skrypty języka z wykorzystaniem dobrych praktyk

programistycznych. Uczeń potrafi zaprojektować bazę danych poprawną pod względem normalizacji, relacji i ograniczeń kolumn.
Ocenę **celującą (6)** otrzymuje uczeń, którego wiedza wykracza poza omawiany zakres. Np. potrafi stworzyć skomplikowaną procedurę składowaną zawierającą kursor. Tworzy zaawansowane obiekty baz danych takie jak procedury, triggery, funkcje skalarne i tablicowe.

7.2 Metody sprawdzania osiągnięć ucznia

Ocena zaliczenia przedmiotu powinna być składową dwóch ocen: testu oraz projektu (zaprojektowanej bazy danych).
Kryteria oceny testu:

- 0-7 poprawnych odpowiedzi - 1
- 8-9 poprawnych odpowiedzi - 2
- 10-11 poprawnych odpowiedzi - 3
- 12-13 poprawnych odpowiedzi - 4
- 14-15 poprawnych odpowiedzi - 5

Kryteria oceny projektu:

- Zaprojektowana baza jest logicznie poprawna, występują relacje, typy danych - 3
- W zaprojektowanej bazie widać próbę normalizacji, zaznaczone są klucze podstawowe i obce, występuje logiczna konwencja nazewnictwa kolumn i tabel - 4
- Poza poprawnym projektem bazy danych uczeń napisał poprawną procedurę składowaną - 5

8 Test końcowy sprawdzający wiedzę

15 zadań na 30 minutowy test sprawdzający wiedzę. Pogrubioną czcionką zaznaczono prawidłowe odpowiedzi.

1. Argumentem funkcji MAX() może być kolumna, w której znajdują się dane typu:
   a. Liczby naturalne
   b. Ciągi znaków
   c. Data
   d. Typ money
2. Które z wymienionych funkcji agregujących są zdefiniowane w języku SQL?
   a. AVG()
   b. MODE()
   c. MEDIANA()
   d. SUMA()
3. W bazie danych znajdują się tabele:
   Film (id, tytul, rok, id_rezyser, producent)
   Rezyser (id, imie, nazwisko, rok_urodzenia, narodowosc)

   W wyniku wykonania zapytania:
   SELECT tytul, rok FROM Film WHERE (rok != 1984 OR rok != 2011) AND
   id_rezyser = (SELECT id FROM Rezyser WHERE rok_urodzenia >= 1949 OR rok_urodzenia <= 1951)
   wyświetlone zostaną:
   a. Wszystkie filmy (tytuł i rok produkcji) z tabeli Film.
   b. Wszystkie filmy (tytuł i rok produkcji), które powstały w innych latach niż 1984 i 2011, oraz których reżyserowie urodzili się pomiędzy rokiem 1949 i 1951.

   c. Wszystkie filmy (tytuł i rok produkcji), które powstały w innych latach niż 1984 i 2011, oraz których reżyserowie urodzili się pomiędzy rokiem 1949 i 1951, uszeregowane alfabetycznie.
   d. Żadne filmy nie zostaną wyświetlone.
4. Za pomocą polecenia JOIN możemy łączyć:
   a. Różne tabele ze sobą
   b. Tabelę z widokiem
   c. Tabelę z indeksem
   d. Tabelę z samą sobą
5. Najczęściej używanym zapytaniem w tabeli tbl jest:
   SELECT id, a, b FROM tbl WHERE a = @a AND b = @b;
   Jaki indeks powinien zostać założony na tej tabeli
   a. CREATE INDEX tbl_idx ON a (tbl);
   b. CREATE INDEX tbl_idx ON b (tbl);
   c. CREATE INDEX tbl_idx ON tbl (a, b);
   d. CREATE INDEX tbl_idx ON tbl (a);
6. Którym poleceniem można zamienić "Kowalski" na "Nowak" w kolumnie „Nazwisko” w tabeli Osoby?
   a. MODIFY Osoby SET Nazwisko= 'Nowak' WHERE Nazwisko ='Kowalski'
   b. UPDATE Osoby SET Nazwisko ='Kowalski' INTO Nazwisko ='Nowak'
   c. MODIFY Osoby SET Nazwisko ='Kowalski' INTO Nazwisko ='Nowak'
   d. UPDATE Osoby SET Nazwisko ='Nowak' WHERE Nazwisko ='Kowalski'
7. Za pomocą SQL, dodaj do tabeli Osoby wiersz, gdzie nazwiskiem będzie "Olsen"?
   a. INSERT ('Olsen') INTO Osoby (Nazwisko)
   b. INSERT INTO Osoby (Nazwisko) VALUES ('Olsen')
   c. INSERT 'Olsen' INTO Osoby (Nazwisko)
   d. INSERT INTO Osoby ('Olsen') INTO Nazwisko
8. Jakie zapytanie zwróci wszystkie rekordy z tabeli Osoby, których nazwisko zaczyna się na literę “a”?
   a. SELECT * FROM Osoby WHERE Nazwisko='a'
   b. SELECT * FROM Osoby WHERE Nazwisko ='%a%'
   c. SELECT * FROM Osoby WHERE Nazwisko LIKE 'a%'
   d. SELECT * FROM Osoby WHERE Nazwisko LIKE '%a'
9. Klauzula TOP
   a. Musi być umieszczona na końcu zapytania.
   b. Służy do wyboru rekordów o największej wartości we wskazanej kolumnie.
   c. Jest niedostępna w systemie MSSQL.
   d. Pozwala ograniczyć liczbę wyświetlanych wyników.
10. Złączenia tabel w języku SQL
   a. Są realizowane za pomocą instrukcji MERGE
   b. Powodują trwałe zmiany w strukturze bazy danych
   c. Można wykonywać na dowolnej liczbie tabel (większej niż 2)
   d. Są realizowane za pomocą operatora „ ; ” (średnik)
11. W której postaci normalnej znajduje się poniższa tabela

Imię	Nazwisko	Adres	Pesel
Jan	Kowalski	Ul. Mała 11, 60-003 Poznań	82111112345
Joanna	Kowalska	Ul. Długa 9a, 00-103 Warszawa	78010112345

a. Tabela nie jest znormalizowana
b. W pierwszej postaci normalnej
c. W drugiej postaci normalnej
d. W trzeciej postaci normalnej

12. UPDATE jest częścią języka
    a. DDL
    b. DML
    c. DQL
    d. Żadnego z powyższych
13. Która z poniższych definicji widoku jest prawidłowa?
    a. CREATE VIEW Widok As SELECT imie, nazwisko, pesel FROM pracownicy
    b. CREATE VIEW Widok SELECT * FROM pracownicy
    c. CREATE VIEW Widok FROM SELECT * FROM pracownicy
    d. CREATE VIEW Widok As imie, nazwisko, pesel FROM pracownicy
14. Do czego służy polecenie:
    SELECT imie, nazwisko, pesel INTO pracownicy FROM klienci
    a. Zapytanie wyświetla imię, nazwisko i pesel z tabeli pracownicy
    b. Za pytanie wyświetla imię, nazwisko i pesel z tabeli klienci
    c. Tworzy tabelę klienci
    d. Zapytanie używane przy tworzeniu widoków – na bazie tabeli klienci tworzy widok pracownicy
15. Które rozszerzenie języka SQL wykorzystywane jest w Microsoft SQL Server
    a. PL SQL
    b. PLPsg SQL
    c. T SQL
    d. MS SQL

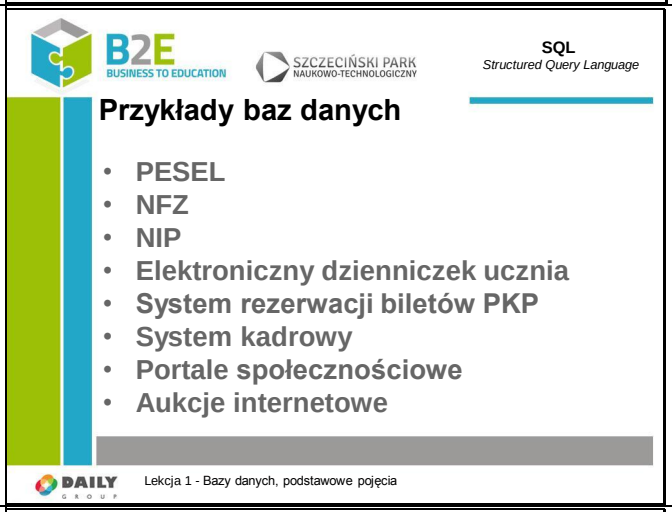
9 Lekcje

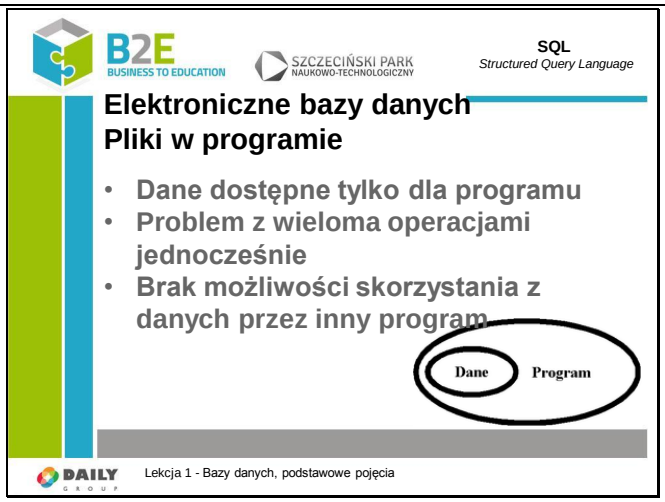
9.1 Lekcja 1 - Bazy danych, podstawowe pojęcia (tabela, dana, wiersz, relacja, klucz podstawowy, klucz obcy, sortowanie, zapytanie, system zarządzania bazą danych), interakcje z bazą danych (Język SQL, aplikacja, formularz raport)

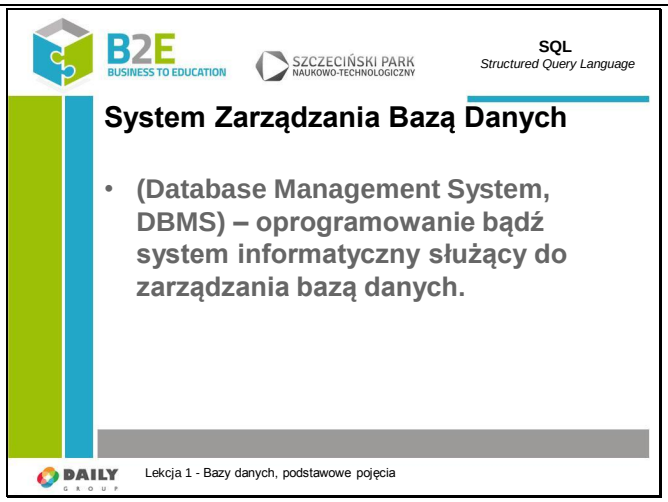
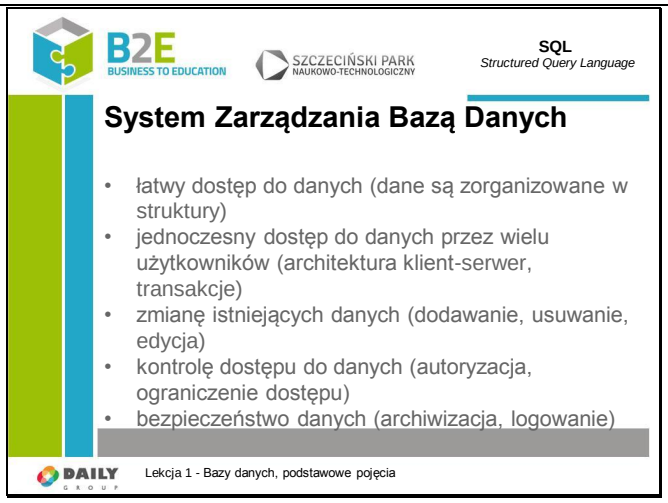
9.1.1 Cel lekcji

Celem lekcji jest wyjaśnienie, czym jest baza danych. Nauczyciel zaprezentuje przykłady bazy danych w życiu codziennym (książka telefoniczna, system rezerwacji biletów lotniczych, system aukcji elektronicznych) – tak, aby uczniowie zdali sobie sprawę, że bazy danych są obecne w ich życiu. Starając się zainteresować tematem nauczyciel zapyta, w jakich bazach danych znajdują się dane uczniów (eWUŚ, PESEL, NFZ, elektroniczny dzienniczek ucznia).
W lekcji zdefiniowane zostaną podstawowe pojęcia dotyczące baz danych: baza danych, system zarządzania bazą danych, tabela, dana, wiersz, relacja, klucz podstawowy, klucz obcy, sortowanie, zapytanie. W ramach przykładów zaprezentowane zostaną proste schematy bazy danych, np. książki telefonicznej. Na ich przykładzie nauczyciel zademonstruje zdefiniowane pojęcia.

9.1.2 Treść - slajdy z opisem

Slajd 1		SQL jest najpopularniejszym językiem programowania baz danych. W tej lekcji wyjaśnimy sobie czym jest baza danych oraz objaśnimy podstawowe pojęcia dotyczące baz danych.
Slajd 2		Pytania do uczniów: jakie znacie przykłady baz danych? W jakich bazach znajdują się Wasze dane?
Slajd 3		W czasie kiedy komputery nie były jeszcze znane tak powszechnie jak dzisiaj, wszystkie informacje były gromadzone na papierze. Pracownicy firmy musieli ręcznie organizować dane. Zapisywać je, wyszukiwać, aktualizować itd. Dzisiaj, oprócz operacji, które muszą wykonać ręcznie, pozostałe wykonuje za pomocą funkcji, które dostarcza relacyjny system bazy danych.

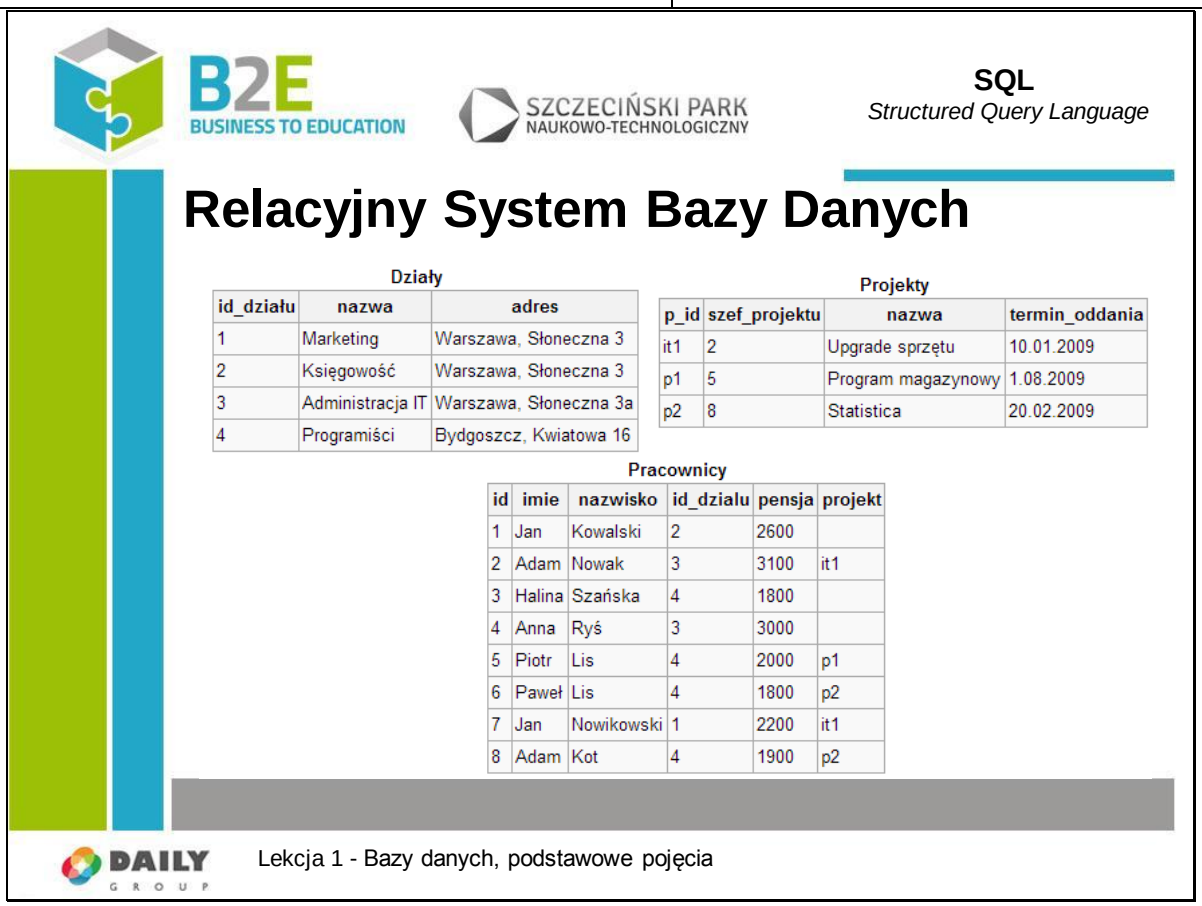
Slajd 4		Najogólniej rzecz biorąc, „baza danych” to zbiór informacji wraz z możliwością łatwego dostępu oraz ich zmiany (tj. modyfikacją, dodawaniem nowych i usuwaniem starych) z poziomu aplikacji z niej korzystającej. Często używamy sformułowania „Używam bazy danych”, co dokładniej oznacza - „korzystam ze zbioru informacji, który łatwo odczytywać i zmieniać”.
Slajd 5		Pytanie do uczniów: czy plik tekstowy z nazwiskami i numerem PESEL jest bazą danych? Plik tekstowy można uznać za bazę danych jeśli odczytanie i modyfikację zapisanych w nim informacji w konkretnym zastosowaniu uznamy za łatwe. Przykładowo niech program wyświetla krótki komunikat, który powinien mieć możliwość dowolnej zmiany. Zastosowanie wtedy zwykłego pliku z tekstem jest jak najbardziej dobrym rozwiązaniem i można go nazwać „bazą danych”. Rysunek na slajdzie przedstawia sytuację, gdzie program korzysta z pliku tekstowego, który jest jego bazą. Wiąże się to jednak z ograniczeniami (wymienione na slajdzie).
Slajd 6		W takim razie czy plik na serwerze (np. FTP), z którego mogą korzystać różne programy jest już „prawdziwą bazą danych”? Mogą z niego korzystać różne programy, lub praktycznie każdy, kto ma dostęp do serwera. W przedstawionej sytuacji nie ma mechanizmu zapewniającego zarządzenia dostępem do danych.

Slajd 7		Brakującym elementem jest System Zarządzania Bazą Danych, który wraz z danymi stanowi BAZĘ DANYCH
Slajd 8		Definicja SZDB
Slajd 9		Cechy, określające system zarządzania bazą danych

Slajd 10

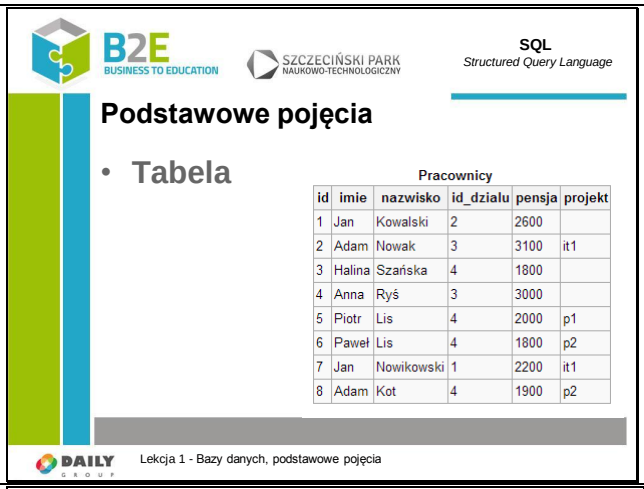
Pytanie do uczniów: czy znacie przykłady SZBD?

Slajd 11

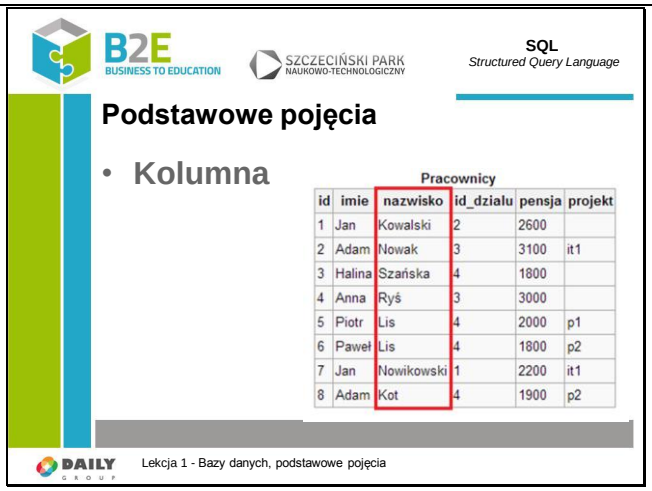


Relacyjny system baz danych przechowuje wszystkie dane w tabelach. Każda tabela zawiera dane na konkretny temat, np. dane o klientach, pracownikach, towarach itp. System bazy danych zarządza tymi informacjami, pozwala m.in. na szybsze ich wyszukiwanie i zorganizowanie.

Slajd 12		W bazach relacyjnych wiele tablic danych pozostaje w relacjach ze sobą (są między sobą powiązane). Bazy relacyjne posiadają wewnętrzne języki programowania, wykorzystujące zwykle SQL do operowania na danych, za pomocą których tworzone są zaawansowane funkcje obsługi danych. Relacyjne bazy danych (jak również przeznaczony dla nich standard SQL) oparte są na kilku prostych zasadach: 1. Wszystkie wartości danych oparte są na typach danych. 2. Wszystkie dane w bazie relacyjnej przedstawiane są w formie dwuwymiarowych tabel (w matematycznym żargonie noszących nazwę „relacji”). Każda tabela zawiera zero lub więcej wierszy (w tymże żargonie – „krotki”) i jedną lub więcej kolumn („atrybutów”). Na każdy wiersz składają się jednakowo ułożone kolumny wypełnione wartościami, które z kolei w każdym wierszu mogą być inne. 3. Po wprowadzeniu danych do bazy, możliwe jest porównywanie wartości z różnych kolumn, zazwyczaj również z różnych tabel, i scalanie wierszy, gdy pochodzące z nich wartości są zgodne. Umożliwia to wiązanie danych i wykonywanie stosunkowo złożonych operacji w granicach całej bazy danych. 4. Wszystkie operacje wykonywane są w oparciu o algebrę relacji, bez względu na położenie wiersza tabeli. Nie można więc zapytać o wiersze, gdzie (x=3) bez wiersza pierwszego, trzeciego i piątego. Wiersze w relacyjnej bazie danych przechowywane są w porządku zupełnie dowolnym – nie musi on odzwierciedlać ani kolejności ich wprowadzania, ani kolejności ich przechowywania. 5. Z braku możliwości identyfikacji wiersza przez jego pozycję pojawia się potrzeba obecności jednej lub więcej kolumn niepowtarzalnych w granicach całej tabeli, pozwalających odnaleźć konkretny wiersz. Kolumny te określa się jako „klucz podstawowy” (ang. primary key) tabeli.

Slajd 13		Tabela składa się z wierszy i kolumn. Wiersze w tabeli są przechowywane w porządku zależnym od implementacji bazy danych. Dla każdego wiersza każda z kolumn posiada jedno pole z wartością. Wszystkie wartości w kolumnie są tego samego typu.
Slajd 14		Dana to pojedynczy wpis w tabeli – jak komórka w MS Excel. Często stosowanym pojęciem jest atrybut.
Slajd 15		Pojedynczy wiersz tabeli nazywany jest rekordem (lub encją) i stanowi najczęściej zbiór danych o pojedynczym obiekcie (ew. grupie obiektów).

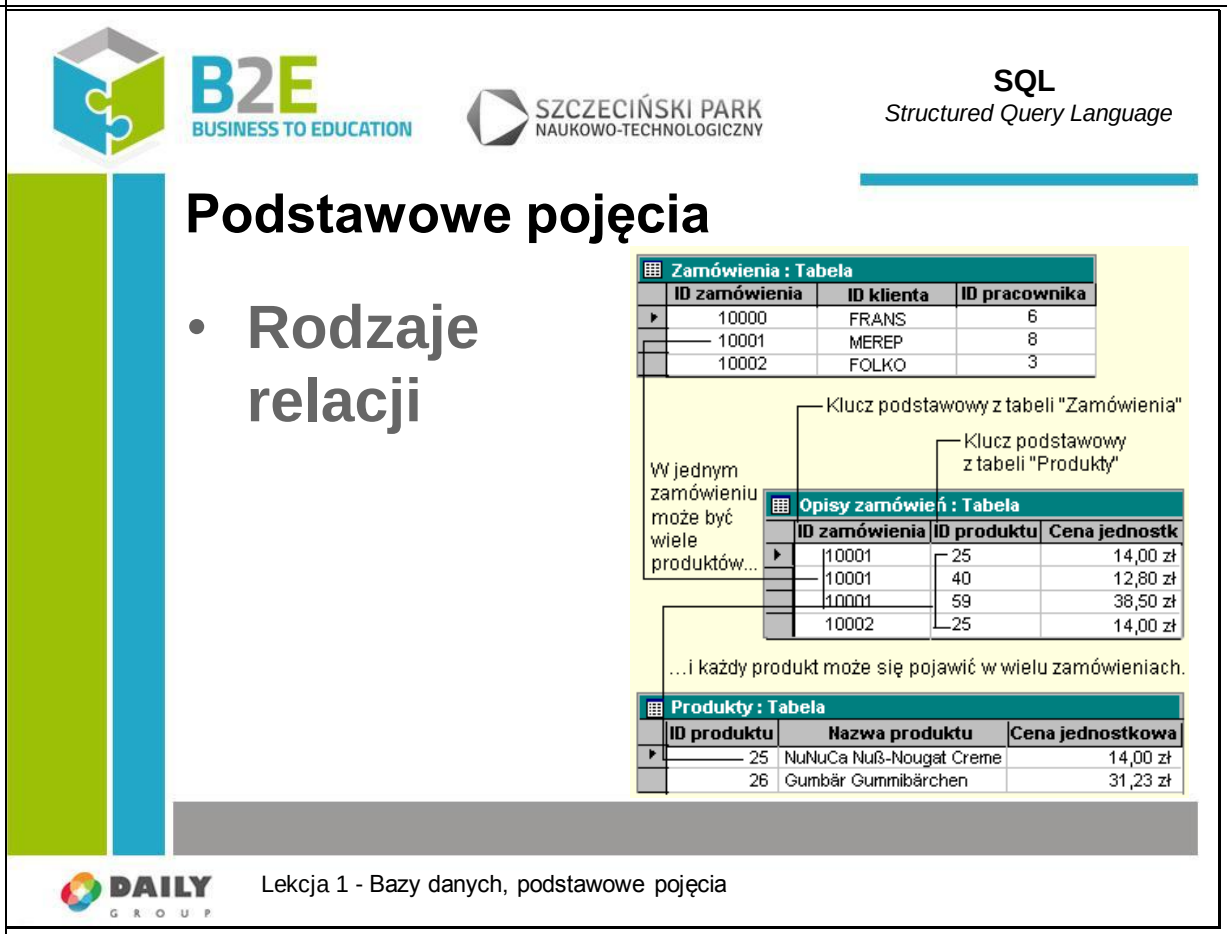
Slajd 16



W relacyjnym modelu baz danych i podobnych, kolumny stanowią zwykle atrybuty jakiegoś obiektu (np. wielkość, grubość, tytuł, nazwisko) i stąd dane zawarte w kolumnach mają najczęściej jeden określony typ. Dodatkowo w bazach obsługiwanych przez język SQL kolumnom nadawane są nazwy, także poza etapem projektowym i nazwy te są unikatowe w obrębie jednej tabeli.

Slajd 17

Każda relacja (prezentowana w postaci np. tabeli) posiada unikatową nazwę, nagłówek i zawartość. Nagłówek relacji to zbiór atrybutów, gdzie atrybut jest parą nazwa_atrybutu: nazwa_typu, zawartość natomiast jest zbiorem krotek (reprezentowanych najczęściej w postaci wiersza w tabeli). W związku z tym, że nagłówek jest zbiorem atrybutów nie jest ważna ich kolejność. Atrybuty zazwyczaj

	utożsamiane są z kolumnami tabeli. Każda krotka (wiersz) wyznacza zależność pomiędzy danymi w poszczególnych komórkach (np. osoba o danym numerze PESEL posiada podane nazwisko i imię oraz adres).
Slajd 18	

1. relacja jeden-do-jednego

W relacji ***jeden-do-jednego*** każdy rekord w tabeli A może mieć tylko jeden dopasowany rekord z tabeli B, i tak samo każdy rekord w tabeli B może mieć tylko jeden dopasowany rekord z tabeli A. Ten typ relacji spotyka się rzadko, ponieważ większość informacji powiązanych w ten sposób byłoby zawartych w jednej tabeli. Relacji ***jeden-do-jednego*** można używać do podziału tabeli z wieloma polami, do odizolowania części tabeli ze względów bezpieczeństwa, albo do przechowania informacji odnoszącej się tylko do podzbioru tabeli głównej.

2. Relacja jeden-do-wielu

Relacja jeden-do-wielu jest najbardziej powszechnym typem relacji.

W relacji ***jeden-do-wielu*** rekord w tabeli A może mieć wiele dopasowanych do niego rekordów z tabeli B, ale rekord w tabeli B ma tylko jeden dopasowany rekord w tabeli A.

3. Relacja wiele-do-wielu

W relacji ***wiele-do-wielu***, rekord w tabeli A może mieć wiele dopasowanych do niego rekordów z tabeli B i tak samo rekord w tabeli B może mieć wiele dopasowanych do niego rekordów z tabeli A. Jest to możliwe tylko przez zdefiniowanie trzeciej tabeli (nazywanej tabelą łącza), której klucz podstawowy składa się z dwóch pól ž kluczy obcych z tabel A i B. Relacja ***wiele-do-wielu*** jest w istocie dwiema relacjami jeden-do-wielu z trzecią tabelą. Na przykład, tabele "Zamówienia" i "Produkty" są powiązane relacją ***wiele-do-wielu*** zdefiniowaną przez utworzenie dwóch relacji jeden-do-wielu z tabelą "Opisy zamówień".

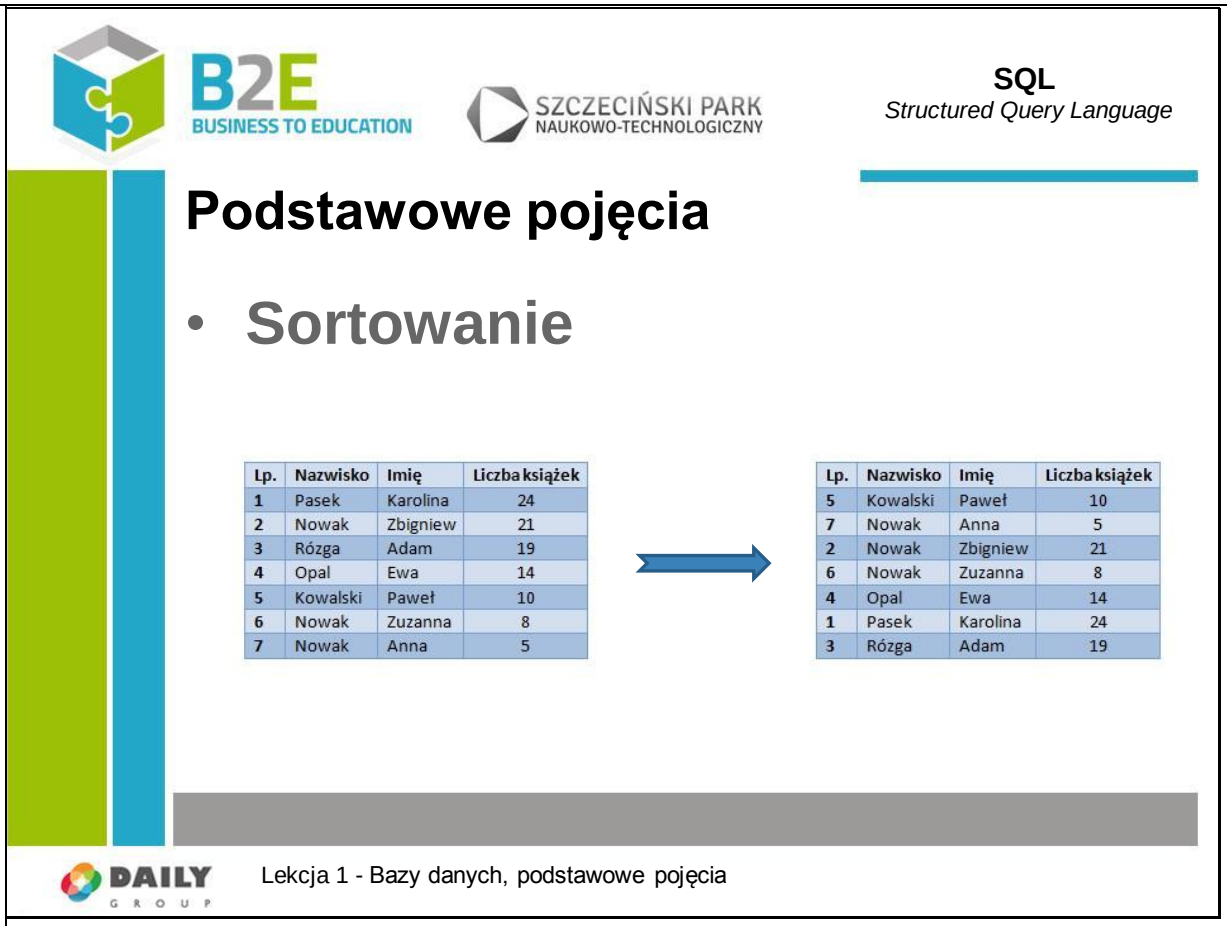
Slajd 19

Każda tabela posiada tzw. klucz główny (primary key). Klucz ten jest unikatowym identyfikatorem w relacji i może być kombinacją kilku kolumn, często jednak obejmuje jedną kolumnę (jeden atrybut). Klucz ma za zadanie jednoznacznie identyfikować każdą krotkę (wiersz) – wartości w wyznaczonych kolumnach są jako zestaw niepowtarzalne w danej tabeli.

Slajd 20

Innym rodzajem klucza jest tzw. klucz obcy (foreign key). Jest to zbiór atrybutów jednej tabeli (relacji) wskazujący wartości klucza kandydującego innej tabeli. Służy do wskazywania zależności pomiędzy danymi składowanymi w różnych tabelach. Klucze w modelu relacyjnym służą m.in. do sprawdzania spójności danych w bazie. Głównie dotyczy to kluczy obcych, na które nałożony jest wymóg, że w tabeli wskazywanej musi istnieć wartość klucza wskazującego.

Slajd 21



Dodatkowym elementem modelu relacyjnego jest zbiór operacji służących do przeszukiwania i manipulacji danymi. Przykładem operacji jest sortowanie danych.

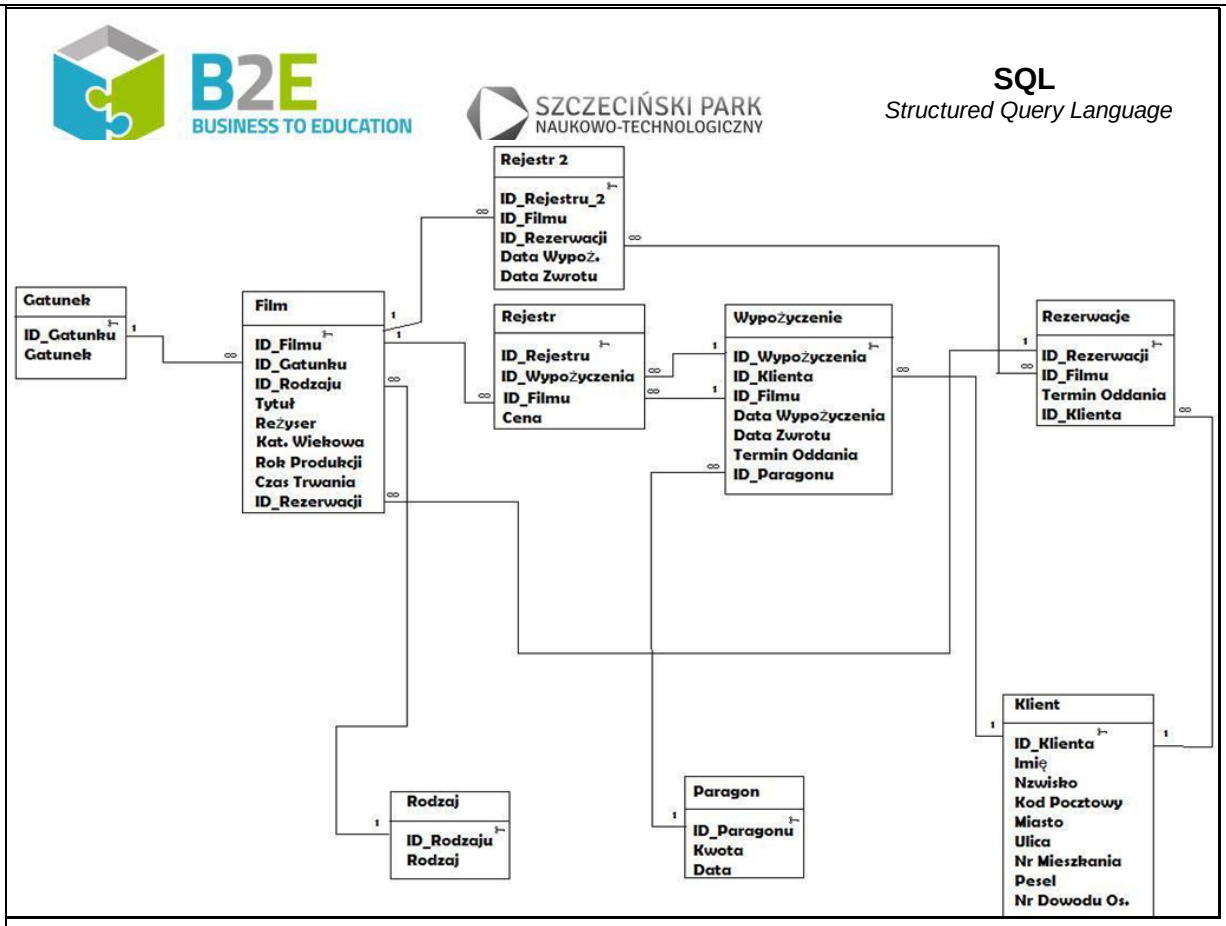
Slajd 22

Najbardziej typowym zastosowaniem bazy danych jest zapytanie o wiersze spełniające konkretne kryteria, np. lista kluczowych klientów z tabeli klienci.

Slajd 23		Jakakolwiek interakcja aplikacji z bazą danych odbywa się za pomocą języka SQL. Jest to jedyny sposób komunikowania się aplikacji z bazą danych. SQL jest językiem deklaratywnym (posługując się nim specyfikujemy tylko co chcemy otrzymać). Nie specyfikujemy sposobu (algorytmu) w jaki ma być zrealizowane zadanie. Przykładem polecenia SQL może być zapytanie do bazy danych poszukujące informacje o klientach banku ze Szczecina, którzy w ciągu ostatniego miesiąca wypłacili z bankomatu łącznie powyżej 8000 PLN. W tym zapytaniu specyfikujemy tylko jakie dane nas interesują. Sposób ich wyszukania jest automatycznie dobierany przez SZBD. SQL jest językiem ustandaryzowanym. Jego standardyzacją zajmuje się specjalny międzynarodowy komitet, w skład którego wchodzą przedstawiciele największych producentów SZBD (IBM, Microsoft, Oracle). Dotychczas opracowano trzy standardy języka SQL, kolejno rozszerzające jego funkcjonalność. Standardy te to: SQL-92, SQL-99, SQL-2003. Przykład prostego polecenia SQL będącego zapytaniem do bazy danych przedstawiono na slajdzie. Zapytanie to wyszukuje pracowników (nazwisko, etat, płaca) zatrudnionych w zespole o numerze 30 na etacie kierownika.

Slajd 24		Język SQL jest narzędziem dostępu do bazy danych stosowanym głównie przez projektantów aplikacji, projektantów baz danych i administratorów baz danych. Standardowym sposobem korzystania z bazy danych przez użytkowników końcowych są aplikacje. Należy jednak pamiętać, że na poziomie programistycznym aplikacje również komunikują się z bazą danych za pomocą poleceń SQL. Ze względu na funkcjonalność, wyróżnia się dwa rodzaje aplikacji, tj. formularze i raporty. Aplikację pierwszego rodzaju należy postrzegać jako elektroniczny formularz (z polami, listami, elementami wyboru) wypełniany przez użytkownika. Formularze umożliwiają pełną obsługę danych, tj. wstawianie, modyfikowanie, usuwanie i wyszukiwanie. Raporty umożliwiają wyłącznie odczytywanie danych z bazy i prezentowanie ich w różnej postaci, głównie tekstu lub wykresu.
Slajd 25		Próba zaprojektowania bazy danych wypożyczalni filmów razem z uczniami, użycie pojęć zdefiniowanych w pierwszej części lekcji.

Slajd 26



Przykład rozwiązania ćwiczenia

Slajd 27

Pytanie do uczniów o podanie najważniejszych ich zdaniem baz danych w dzisiejszej rzeczywistości. Przykłady odpowiedzi:

- Rachunki bankowe
- Mapy – nawigacje samochodowe
- Bazy urzędu skarbowego
- Bazy NFZ – np. eWUŚ
- Baza stron WWW – np. Google

Slajd 28		Pytanie do uczniów o podanie baz danych, zawierających ich dane. Przykłady odpowiedzi: • Rachunki bankowe • Mapy – nawigacje samochodowe • Baza uczniów szkołu • NFZ • Baza Urzędu Miasta
Slajd 29		Odpowiedź prawidłowa: Dana

Slajd 30

Biblioteka Kongresu

To chyba akurat nikogo nie dziwi. Biblioteka Kongresu przechowuje 130 milionów dokumentów, wśród których można znaleźć autentycznie wszystko – od archiwalnych książek kucharskich aż po egzemplarze gazet z czasów kolonialnych. Gdyby wycisnąć z nich wszystkie literki, same słowa zajęłyby 20 terabajtów. BC rozrasta się o 10 000 egzemplarzy dziennie, a do przechowywania tego wszystkiego potrzeba półek o łącznej długości aż 530 mil. Pięć milionów dokumentów jest już dostępnych w wersji cyfrowej.

World Data Centre for Climate

Którym zarządzają Meteorologiczny Instytut Maxa Plancka oraz Niemieckie Centrum Komputerowej Analizy Klimatu. WDCC chwali się tym, że posiada bazę danych o objętości ponad 220 terabajtów, do których można dostać się przez internet oraz dodatkowo 110 terabajtów symulacji klimatycznych, a także sześć petabajtów przechowywanych na taśmach magnetycznych.

YouTube

Przy powstawaniu serwisu, przewidywano że nigdy nie będzie on wymagał więcej niż 45 terabajtów. Ze względu na to, że filmiki wrzucane przez użytkowników różnią się od siebie długością i rozmiarem, ciężko oszacować rozmiar bazy danych YT dzisiaj, ale ”magiczne” 45 terabajtów zostało z pewnością już dawno przekroczone. Co godzinę na YouTubie pojawia się 60 godzin wideo, oglądane są na nim cztery miliardy filmów dziennie, co daje łącznie ponad trzy miliardy godzin oglądania na miesiąc. Robi wrażenie.

ChoicePoint

Jedna z dziwniejszych firm, jakie kiedykolwiek istniały. ChoicePoint, zlokalizowana niedaleko Atlanty, zajmowała się zbieraniem danych osobowych, które potem sprzedawała tym, którzy oferowali najwyższą cenę. Udało jej się wejść w posiadanie 17 miliardów dokumentów na temat 250 milionów

	osób – gdyby je wydrukować sięgnęłyby z Ziemi do Księżyca i z powrotem 77 razy. Dane zbierane przez ChoicePoint zawierały wszystko na temat Amerykanów – rejestry karanych, adresy, numery telefonów, wydruki z kart kredytowych i tak dalej, i tak dalej. Niestety okazało się przy okazji, że wyciekały z firmy, a do tego wielokrotnie zdarzało się, że podawane przez CP informacje okazywały się nie do końca poprawne, co potrafiło doprowadzić do tego, że niektóre osoby traciły pracy, kiedy okazywało się na przykład, że wbrew temu, co mówiły, były jednak karane. **Google** To też chyba nikogo tutaj nie dziwi. Jednak dane o dokładnych rozmiarach baz danych Google`a nie są znane, bo firma ich nie ujawnia. To co jednak wiadomo, to to, że centra danych korporacji z Mountain View pożarły w 2010 roku kilowatogodzin czyli połowę tego, co produkuje w ciągu roku słynna Zapora Hoovera – tama wysoka na ponad 220 metrów i długa na blisko 400. I że każdego dnia Google analizuje 24 petabajty danych. Żeby je wydrukować na papierze, trzeba by ściąć ponad 1,2 miliona drzew. **CIA** Tutaj małe zaskoczenie, aczkolwiek w sumie to żadne zaskoczenie – bazy danych CIA są uznawane za jedne z największych na świecie, choć nikt nie ma zielonego pojęcia, ile informacji się w nich znajduje. A wszystko to dlatego, że większość z nich jest utajnionych. China Mobile Tutaj też dokładnie nie wiadomo, niełatwo wyciąga się takie informacje od Chińczyków. Ale swego czasu jedną z największych baz danych miał znany operator telefonii komórkowej Sprint, który z 53 milionami użytkowników posiadał bazę danych zawierającą blisko trzy miliardy rekordów. A China Mobile oferuje swoje usługi 70% Chińczyków, mając tym samym 655 milionów klientów. Poczujcie cenną różnicę.

9.1.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 25: Próba wspólnego zaprojektowania bazy danych wypożyczalni filmów DVD, wykorzystanie pojęć przedstawionych w czasie lekcji oraz dodatkowo na slajdach 26-30.

9.1.4 Opis założonych osiągnięć ucznia

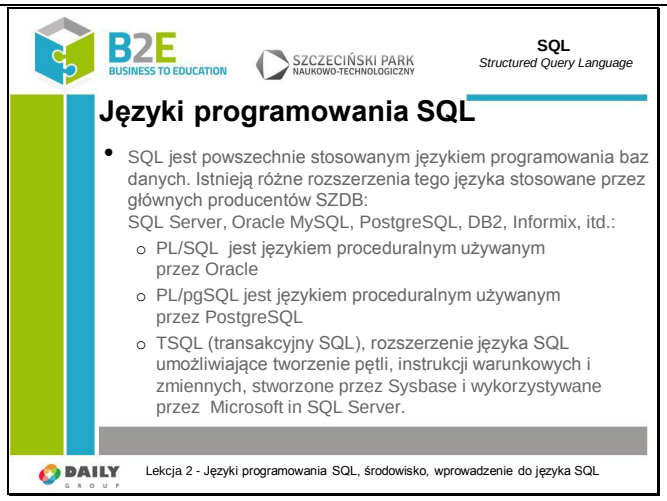
Po tej lekcji uczniowie poznają definicję bazy danych, podstawowe pojęcia z nią związane. Będą potrafili wskazać przykłady baz danych.

9.2 Lekcja 2 - Języki programowania serwerów SQL, środowisko programowania w języku SQL, wprowadzenie do języka SQL

9.2.1 Cel lekcji

Celem lekcji jest poznanie różnych rozszerzeń języka SQL, stosowanych przez producentów systemów zarządzania bazą danych. Uczniowie zostaną się ze środowiskiem Microsoft SQL Management Studio Express. Zostaną poinstruowani w jaki sposób można pobrać i zainstalować środowisko.
W drugiej części lekcji uczniowie poznają podstawowe kwerendy (zapytania) SQL, a w ramach ćwiczeń – spróbują praktycznie wykorzystać wiedzę.

9.2.2 Treść - slajdy z opisem

Slajd 1		SQL jest najpopularniejszym językiem programowania baz danych. W tej lekcji poznamy różne rodzaje języka SQL, oraz poznamy podstawy tego języka.
Slajd 2		
Slajd 3		Wyjaśnienie, że pomimo wspólnego standardu istnieje wiele różnych rozszerzeń. Podstawowa składnia wszystkich rozszerzeń języka jest taka sama, jednak istnieje wiele różnic. W tym kursie zajmiemy się językiem SQL w ujęciu rozszerzenia TSQL, promowanym przez Microsoft.

Slajd 4

Środowiska programowania TSQL. Microsoft oferuje wraz z bazą danych SQL Serwer narzędzie Microsoft SQL Server Management Studio (często używa się skrótu MSSMS). Możliwe jest pobranie wersji bezpłatnej oznaczonej przez producenta hasłem „Express". Poza oficjalnym narzędziem dostępne są również narzędzia innych firm, np. Toad (również darmowe).

Slajd 5

Management Studio wraz z bazą danych można pobrać ze strony Microsoftu. Adres podany jest na slajdzie. Warto zwrócić uwagę aby pobrać wersję SZDB wraz z MS SQL SMS: ENU\x64\SQLEXPRWT_x64_ENU.exe:

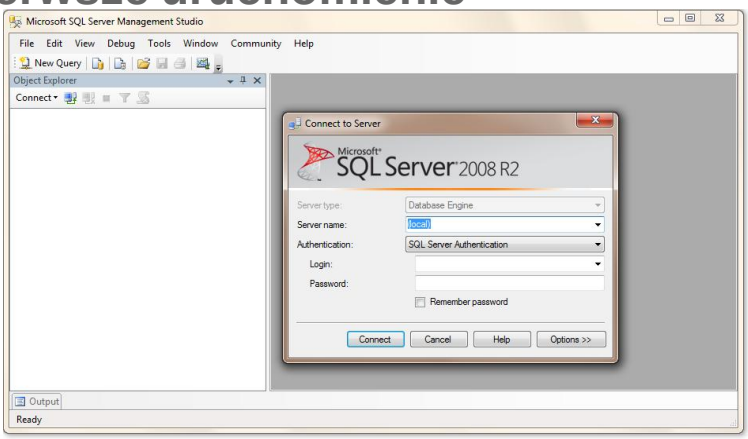
„Express with Tools (with LocalDB) Includes the database engine and SQL Server Management Studio Express)

This package contains everything needed to install and configure SQL Server as a database server. Choose either LocalDB or Express depending on your needs above."

Slajd 6

MS SQL Management Studio Express

- **Pierwsze uruchomienie**

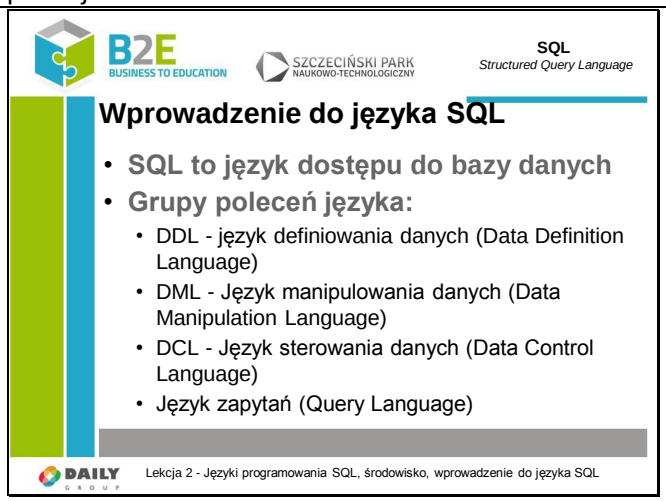


Po instalacji oprogramowania należy się zalogować do bazy danych: wybieramy serwer lokalny, podajemy użytkownika i hasło, które zostało zdefiniowane w procesie instalacji. Domyślnym użytkownikiem administracyjnym jest sa (super administrator).

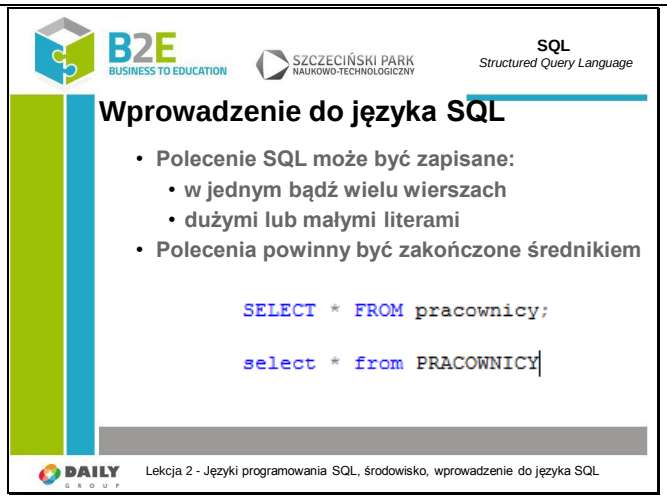
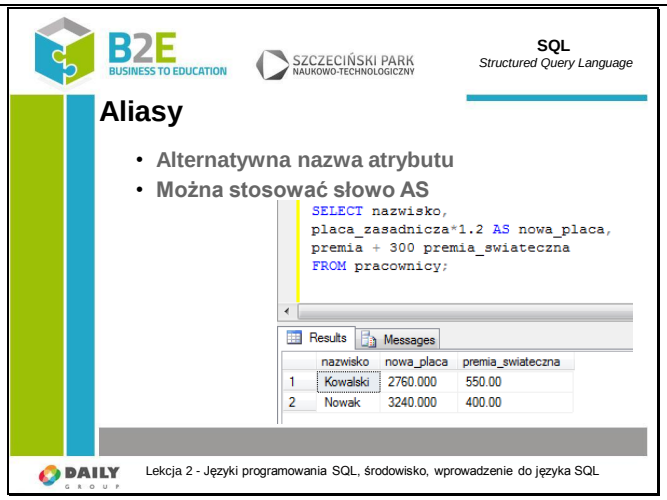
Slajd 7

Opis elementów programu MS SQL SMS. Okno programu składa się z sekcji obiektów bazy danych po lewej stronie, rekcji roboczej (wykorzystywanej na przykład do wpisywania zapytań TSQL) na środku, poniżej – okno rezultatów zapytań TSQL oraz okna właściwości po prawej stronie.

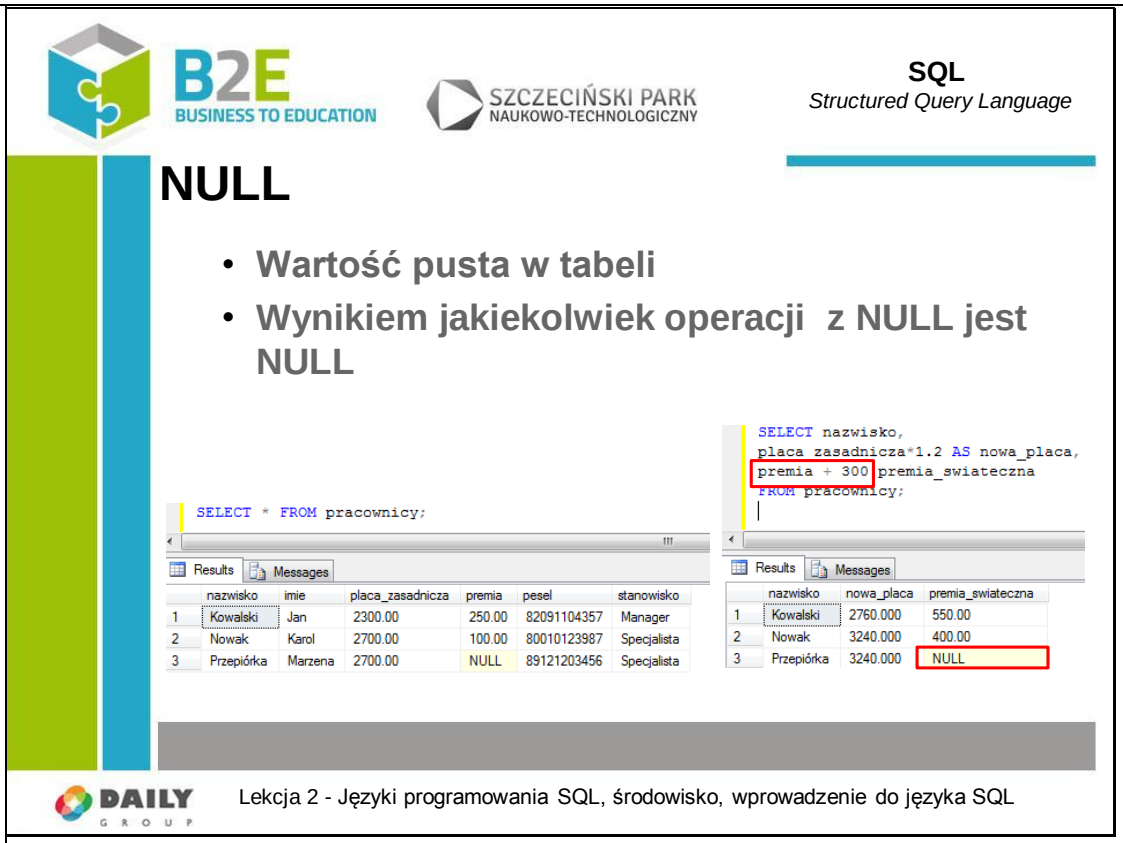
Slajd 8



- Język definiowania danych (Data Definition Language - DDL), który umożliwia definiowanie struktury danych zawartych w bazie.
- Język manipulowania danych (Data Manipulation Language - DML),który umożliwia wypełnienie, modyfikowanie i usuwanie danych z bazy.
- Język sterowania danych (Data Control Language), który umożliwia sterowanie transakcjami tj. akceptacja lub wycofanie.
- Język zapytań (Query Language), który umożliwia pobieranie informacji z bazy za pośrednictwem określonych zapytań, warunków.

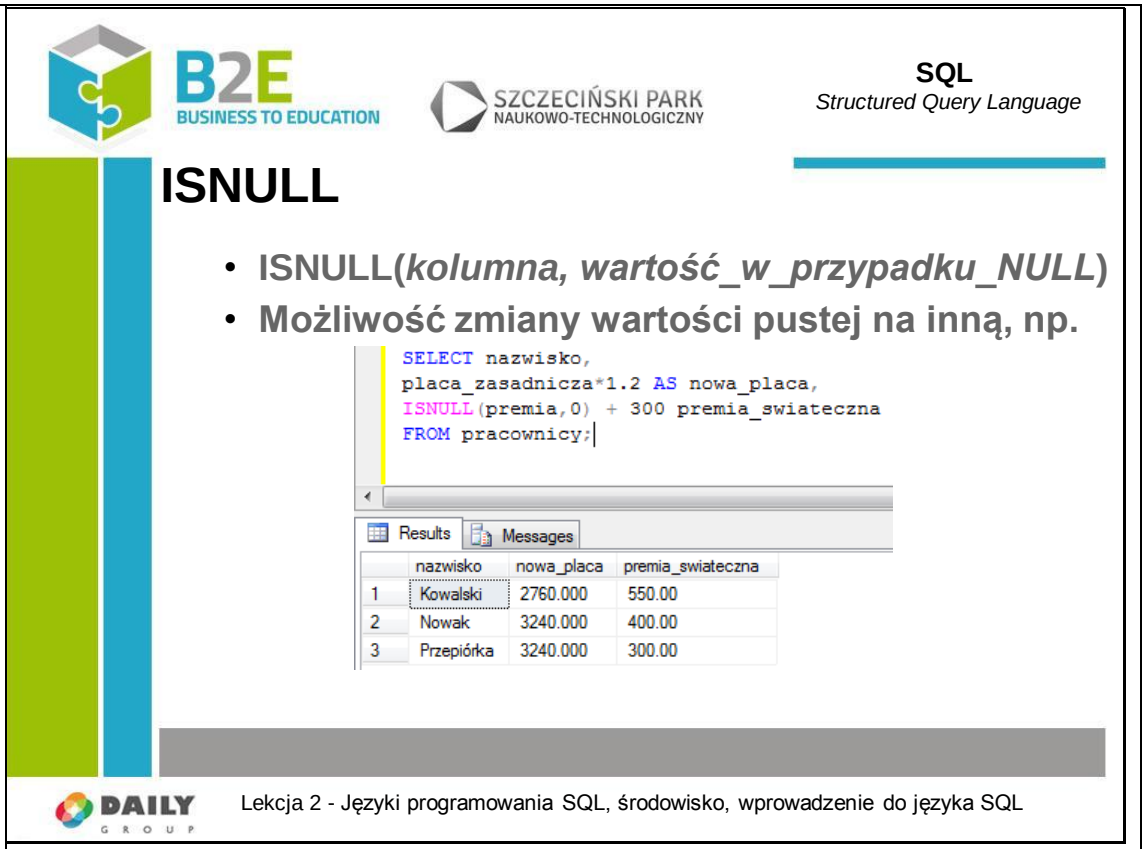
Slajd 9		W SQL występuje dowolność wielkich i małych liter. W ramach ujednolicania rozszerzeń języków SQL, Microsoft dopuszcza stosowanie średnika na końcu polecenia (w PL/SQL jest wymagany).
Slajd 10		Przykład zapytania SELECT: pobranie imion i nazwisk wszystkich pracowników. W poleceniu tłumaczonym z j. angielskiego Wybierz imię, nazwisko, pesel z tabeli pracownicy, występują nazwy kolumn (atrybuty) oraz nazwa tabeli (pracownicy). W wyrażeniach SELECT można korzystać z operacji arytmetycznych takich jak dodawania, odejmowanie, mnożenie i dzielenie.
Slajd 11		W celu wyświetlenia kolumn o nazwach innych niż kolumny tabeli można stosować aliasy. Alias wpisujemy po nazwie kolumny lub po słowie kluczowym AS.

Slajd 12



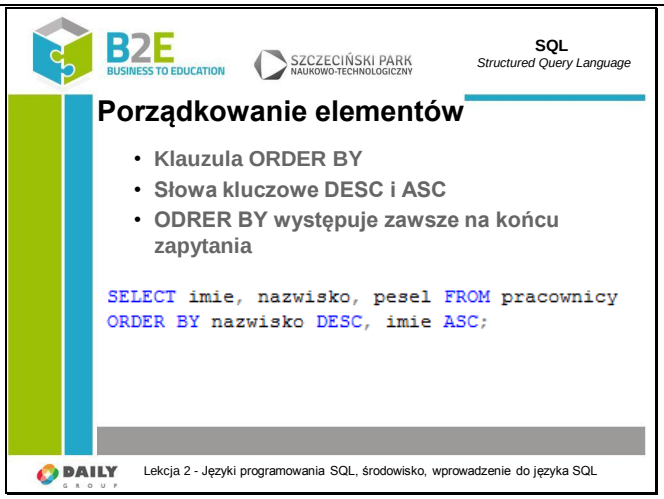
W bazach danych występuje wartość pusta NULL. Nie jest to 0, czy pusty ciąg znaków – tylko wartość nieokreślona. Wynikiem dodawania, odejmowania, mnożenia czy dzielenia wartości NULL jest również wartość NULL, co zaprezentowano na przykładzie.
Ciekawostka: w Oracle w przeciwieństwie do MSSQL NULL traktowany jest jak pusty string.

Slajd 13



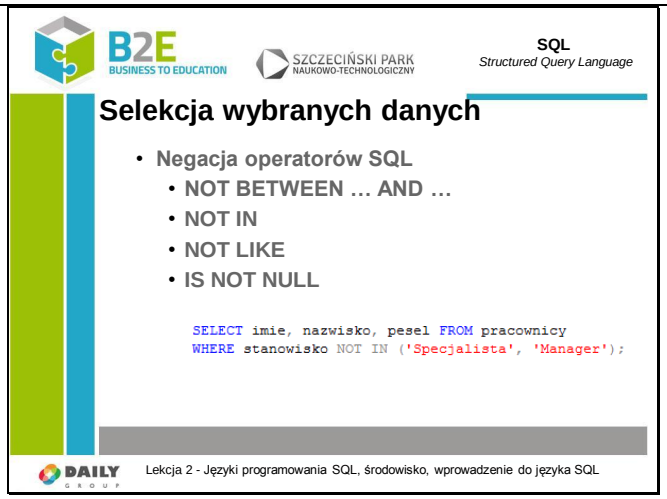
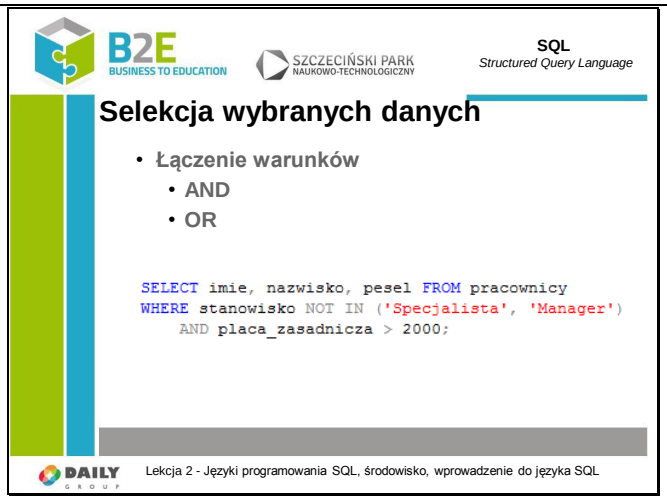
Możliwością uniknięcia problemów z kolumnami zawierającymi wartość NULL jest polecenie ISNULL, które ma dwa paramerty. Pierwszym jest nazwa kolumny, drugim wartość jaka ma być podstawiona w przypadku gdy kolumna zawiera wartość NULL.

Slajd 14

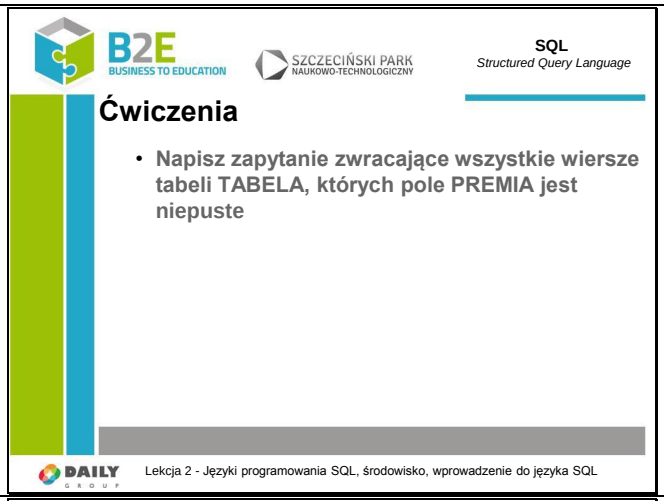
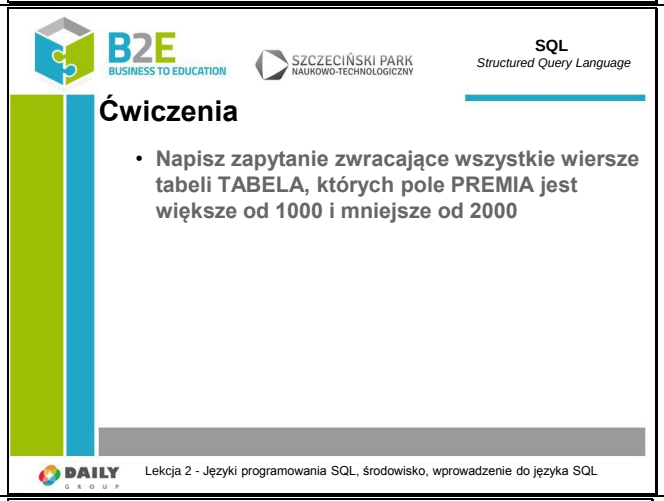
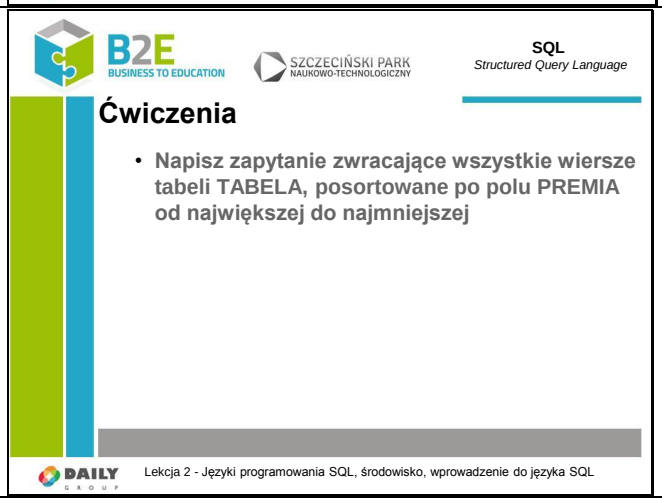


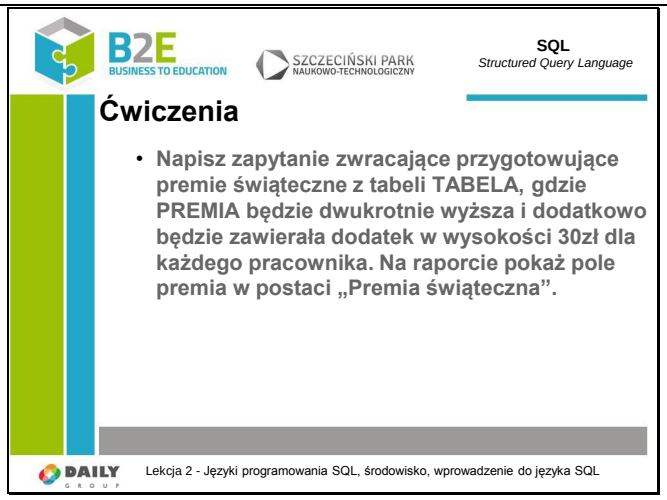
Sortowanie wyników uzyskuje się przy wykorzystaniu klauzuli ORDER BY. Sortowanie rosnące (ascending) jest domyślne. Po nazwie kolumny można użyć słów kluczowych ASC i DESC w celu uzyskania oczekiwanej kolejności.

Slajd 15		Słowo kluczowe DISTINCT musi występować zaraz po słowie kluczowym SELECT. SELECT stanowisko FROM pracownicy Takie zapytanie wyświetli wszystkie stanowiska obejmowane wśród pracowników. Jeżeli pojawią się dwa takie same stanowiska, tylko jedno zostanie wyświetlone. Słowo DISTINCT eliminuje wiersze, które posiadają duplikaty we wszystkich kolumnach wyspecyfikowanych w wyrażeniu SELECT. Tylko jedno słowo DISTINCT może zostać użyte w całym zapytaniu SELECT.
Slajd 16		Używając klauzuli WHERE możemy zawęzić zwracane wartości do tylko tych które spełniają wybrane kryteria. Np. kiedy chcemy znaleźć pracowników zarabiających więcej niż 2000zł.
Slajd 17		Operator BETWEEN AND umożliwia wybór wartości z danego zakresu, operator IN wybór wartości zawierających się w zdefiniowanym zbiorze

Slajd 18		Operator LIKE umożliwia wybór wartości które zawierają pewien ciąg znaków, np. zaczynają się od litery „M”, lub kończą na „ski”.
Slajd 19		Negowanie warunków jest możliwe za pomocą słowa kluczowego NOT. NOT występuje przed operatorem. Wyjątkiem jest operator IS NULL, gdzie negator występuje w środku: IS NOT NULL.
Slajd 20		Operatory logiczne mogą być stosowane jednocześnie w tej samej klauzuli WHERE. Służą do tego operatory logiczne AND i OR. AND posiada wyższy priorytet niż OR, a zmiana priorytetu jest możliwa za pomocą nawiasów.

Slajd 21	 nazwisko imie placa_zasadnicza premia pesel stanowisko 1 Kowalski Jan 2300.00 250.00 82091104357 Manager 2 Nowak Karol 2700.00 100.00 80010123987 Specjalista 3 Przepiórka Marzena 2700.00 NULL 89121203456 Specjalista 4 Burzych Paweł 1900.00 500.00 78032309123 Specjalista 5 Makłowicz Marek 2000.00 NULL 54013112345 Specjalista 6 Naramowicka Magdalena 2100.00 200.00 77121312098 Specjalista 7 Witos Jacek 3000.00 500.00 69100967234 Manager	Ćwiczenie 1: Napisz zapytanie zwracające wszystkich Managerów Ćwiczenie 2: Napisz zapytanie zwracające wszystkich specjalistów, których łączna płaca jest wyższa niż 2300zł Ćwiczenie 3: Napisz zapytanie wyświetlające wszystkie kobiety wśród pracowników (końcówka imienia „a”) Ćwiczenie 4: Napisz zapytanie wyświetlające pracowników, których roczna płaca zasadnicza jest większa niż 25.000zł Ćwiczenie 5: Napisz zapytanie wyświetlające pracowników, którzy nie otrzymują premii Ćwiczenie 6: Napisz zapytanie wyświetlające pracowników, których premia jest pomiędzy 200 a 300zł Ćwiczenie 7: Napisz zapytanie wyświetlające łączny roczny dochód pracowników Ćwiczenie 8: Napisz zapytanie wyświetlające pracowników urodzonych w latach 80-tych Ćwiczenie 9: Napisz zapytanie wyświetlające specjalistów urodzonych w latach 80-tych, zarabiających więcej niż 2.700zł Ćwiczenie 10: Napisz zapytanie wyświetlające pracowników, których imię i nazwisko zaczyna się na literę M

Slajd 22		SELECT * FROM TABELA WHERE PREMIA IS NOT NULL
Slajd 23		SELECT * FROM TABELA WHERE PREMIA IS BETWEEN 1000 AND 2000
Slajd 24		SELECT * FROM TABELA ORDER BY PREMIA DESC

Slajd 25		SELECT (PREMIA*2)+30 AS [Premia świąteczna] FROM TABELA
Slajd 26		

SELECT	Wybiera listę kolumn
alias	Można stosować tylko do kolumn i wyrażeń (nie do *)
*	Oznacza wszystkie kolumny
DISTINCT	Eliminuje duplikaty ze zbioru wynikowego
FROM t	Określa relację, z której odczytujemy dane
WHERE	Określa warunki wyboru wierszy, zawiera wartości kolumn
AND/OR	Łączy warunki w klauzuli WHERE
()	Pozwala na zmianę priorytetu operatorów
ORDER BY	Służy do określenia kryterium sortowania
ASC	Rosnący porządek sortowania
DESC	Malejący porządek sortowania

Tabela podsumowująca poznane wyrażenia

9.2.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 21. Uczniowie w ramach ćwiczenia będą musieli napisać zapytania SQL z tabel przygotowanych przez szkoleniowca. Dodatkowe ćwiczenia przedstawiono na slajdach 22-25.

9.2.4 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą wiedzieli jak pobrać i zainstalować środowisko MS SMS. Dowiedzą się jak konstruować proste zapytania SQL, polegające na wyświetlaniu informacji z bazy, sortowaniu, definiowaniu podstawowych warunków zapytania.

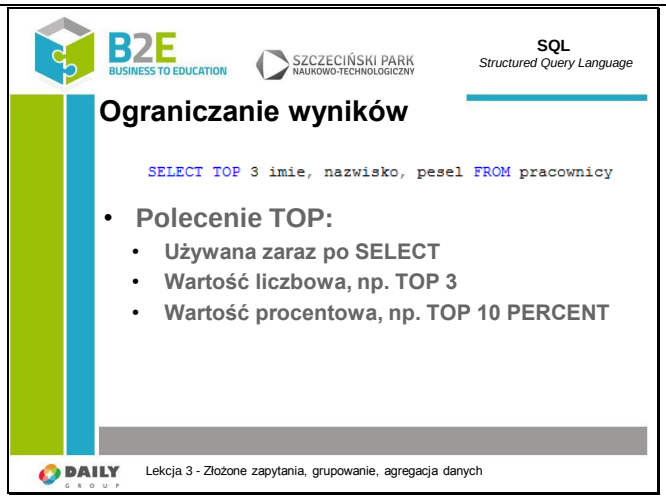
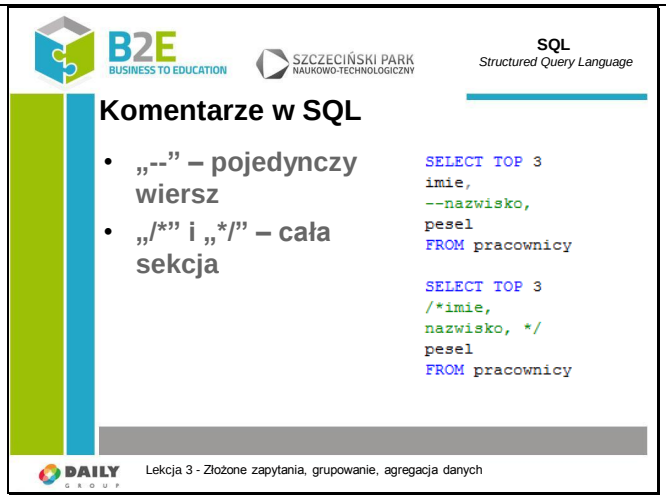
9.3 Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

9.3.1 Cel lekcji

Celem lekcji jest zaznajomienie uczniów z bardziej skomplikowanymi zapytaniami SQL niż w poprzedniej lekcji. Uczniowie dowiedzą się, jakie są sposoby łączenia danych z wielu tabel. W drugiej części lekcji przedstawione zostaną zagadnienia związane z funkcjami agregującymi oraz grupowaniem danych. Dzięki ćwiczeniom praktycznym, uczniowie praktycznie wykorzystają zdobytą w czasie lekcji wiedzę.

9.3.2 Treść - slajdy z opisem

Slajd 1		Na poprzedniej lekcji zajmowaliśmy się prostymi zapytaniami do bazy danych, teraz przejdziemy do bardziej złożonych zapytań, nauczymy się łączyć tabele, grupować dane. Dowiemy się jak wybierać wartości minimalne i maksymalne, nauczymy się sumować.
Slajd 2		Powtórzenie materiału z poprzedniej lekcji.

Slajd 3		Ograniczanie wyników w SQL realizowane jest słowem TOP. Możliwe jest ograniczenie ilościowe i procentowe. Używanie TOP bez klauzuli ORDER BY jest mocno niezalecane z uwagi na możliwą przypadkowość wyniku! Ciekawostka: Od MSSQL 2012 są też bardziej praktyczna metody tj. OFFSET (ilość rekordów do pominięcia) i FETCH ilość rekordów do pobrania.
Slajd 4		Język SQL umożliwia stosowanie komentarzy. W celu komentowania jednego wiersza stosuje się dwa minusy. Można również zakomentować całą sekcję – zaczynamy komentarz od ukośnika i gwiazdki, kończymy komentarz gwiazdką i ukośnikiem.

Slajd 5

Łączenie tabel

dbo.pracownicy

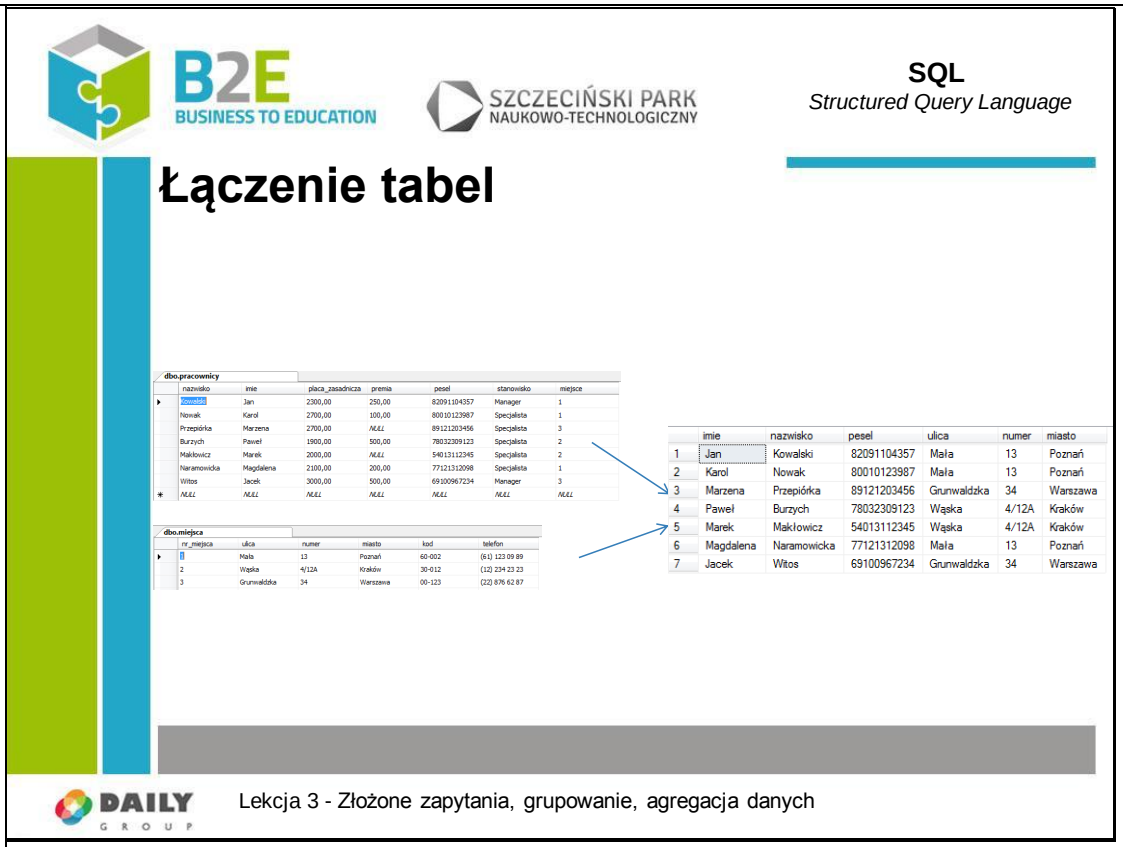
nazwisko	imie	placa_zasadnicza	premia	pesel	stanowisko	miejsce
Kowalski	Jan	2300,00	250,00	82091104357	Manager	1
Nowak	Karol	2700,00	100,00	80010123987	Specjalista	1
Przepiórka	Marzena	2700,00	NULL	89121203456	Specjalista	3
Burzych	Paweł	1900,00	500,00	78032309123	Specjalista	2
Makłowicz	Marek	2000,00	NULL	54013112345	Specjalista	2
Naramowicka	Magdalena	2100,00	200,00	77121312098	Specjalista	1
Witos	Jacek	3000,00	500,00	69100967234	Manager	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

dbo.miejsca

nr_miejsca	ulica	numer	miasto	kod	telefon
1	Mała	13	Poznań	60-002	(61) 123 09 89
2	Wąska	4/12A	Kraków	30-012	(12) 234 23 23
3	Grunwaldzka	34	Warszawa	00-123	(22) 876 62 87

Na slajdzie widzimy dwie tabele – znaną z poprzednich lekcji „pracownicy” oraz „miejsca” możemy sobie wyobrazić potrzebę połączenia tych tabel, np. w celu przygotowania zestawienia – pracownicy wraz z miejscem zamieszkania.

Slajd 6



	imie	nazwisko	pesel	ulica	numer	miasto
1	Jan	Kowalski	82091104357	Mała	13	Poznań
2	Karol	Nowak	80010123987	Mała	13	Poznań
3	Marzena	Przepiórka	89121203456	Grunwaldzka	34	Warszawa
4	Paweł	Burzych	78032309123	Wąska	4/12A	Kraków
5	Marek	Makłowicz	54013112345	Wąska	4/12A	Kraków
6	Magdalena	Naramowicka	77121312098	Mała	13	Poznań
7	Jacek	Witos	69100967234	Grunwaldzka	34	Warszawa

Często w praktyce musimy połączyć wyniki z kilku tabel. Wyobraźmy sobie zadanie – przygotować raport pracowników wraz z ich miejscem pracy.

Slajd 7

	nazwisko	pesel	ulica	numer	miasto
1	Jan	Kowalski	82091104357	Mała	13 Poznań
2	Karol	Nowak	80010123987	Mała	13 Poznań
3	Marzena	Przepi...	89121203456	Mała	13 Poznań
4	Paweł	Burzych	78032309123	Mała	13 Poznań
5	Marek	Makło...	54013112345	Mała	13 Poznań
6	Magdalena	Naram...	77121312098	Mała	13 Poznań
7	Jacek	Witos	69100967234	Mała	13 Poznań
8	Jan	Kowalski	82091104357	Wąska	4/1... Kraków
9	Karol	Nowak	80010123987	Wąska	4/1... Kraków
10	Marzena	Przepi...	89121203456	Wąska	4/1... Kraków
11	Paweł	Burzych	78032309123	Wąska	4/1... Kraków
12	Marek	Makło...	54013112345	Wąska	4/1... Kraków
13	Magdalena	Naram...	77121312098	Wąska	4/1... Kraków
14	Jacek	Witos	69100967234	Wąska	4/1... Kraków
15	Jan	Kowalski	82091104357	Grunwaldzka	34 Warszawa
16	Karol	Nowak	80010123987	Grunwaldzka	34 Warszawa
17	Marzena	Przepi...	89121203456	Grunwaldzka	34 Warszawa
18	Paweł	Burzych	78032309123	Grunwaldzka	34 Warszawa
19	Marek	Makło...	54013112345	Grunwaldzka	34 Warszawa

JOIN (lub CROSS JOIN) może być stosowany w różnych wersjach

Wynikiem złączenia naturalnego jest zbiór wierszy łączonych tabel; dla tych wierszy wartości kolumn określonych jako warunek złączenia są takie same. Ponieważ w relacyjnych bazach danych informacje są podzielone pomiędzy tabele zawierające dane o obiektach jednego typu, złączenie naturalne jest najczęściej wykorzystywanym (i domyślnym) złączeniem obiektów.

W przykładowej bazie danych informacje o pracownikach i ich miejscach pracy przechowywane są w powiązanych ze sobą tabelach. Dlatego aby wyświetlić pracowników i ich miejsca pracy musimy skorzystać z obu tabel. Przy czym z reguły nie chodzi nam o uzyskanie poniższego wyniku - który wyświetla wszystkie możliwe kombinacje danych.

Slajd 8

Poprawnie napisana instrukcja powinna zwrócić nam tylko miejsca do których przypisany jest konkretny pracownik. Żeby to osiągnąć, musimy określić warunek złączenia, czyli poinformować serwer baz danych, co łączy zapisane w obu tabelach dane. W tym przypadku jest to identyfikator miejsca pracy — zwróć uwagę, że kolumna *miejsce* występuje w obu tabelach, czyli na podstawie tego identyfikatora jesteśmy w stanie sensownie połączyć informacje o pracownikach i miejscach.

INNER JOIN … ON .. Łączy tabele zgodnie ze zdefiniowanym warunkiem.

JOIN (z ang. łączyć) – występuje zawsze w części po FROM w zapytaniu SQL. Tłumacząc powyższą kwerendę (zapytanie):

Wyświetl imię, nazwisko, pesel, ulica, numer, miasto z tabeli pracownicy i połącz z tabelą miejsca, gdzie miejsce w tabeli pracownicy jest takie samo jak nr_miejsca w tabeli miejsca.

Slajd 9

Złączenia zewnętrzne

- **INNER JOIN pomija wyniki, które nie mają odpowiedników obu tabelach**

```
SELECT imie, nazwisko, pesel, ulica, numer, miasto
FROM pracownicy INNER JOIN miejsca
ON pracownicy.nr_miejsca = miejsca.nr_miejsca
```

Results | Messages

	imie	nazwisko	pesel	ulica	numer	miasto
1	Jan	Kowalski	82091104357	Mała	13	Poznań
2	Karol	Nowak	80010123987	Mała	13	Poznań
3	Paweł	Burzych	78032309123	Wąska	4/12A	Kraków
4	Marek	Makłowicz	54013112345	Wąska	4/12A	Kraków
5	Magdalena	Naramowicka	77121312098	Mała	13	Poznań
6	Jacek	Witos	69100967234	Grunwaldzka	34	Warszawa

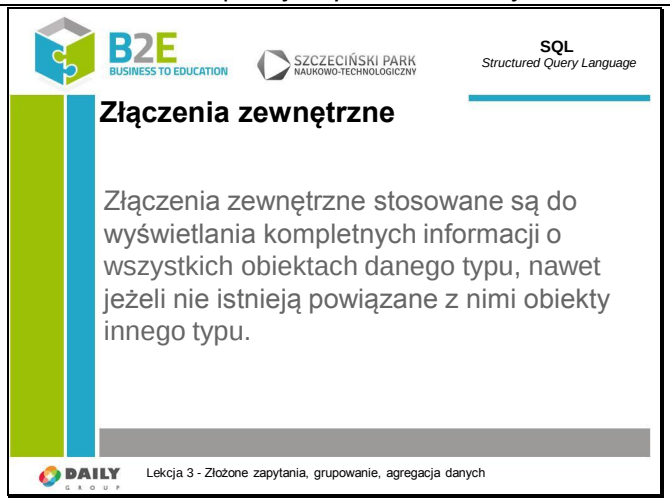
Złączenie naturalne eliminuje z wyniku niepasujące (niespełniające warunku złączenia) wiersze. To dobrze, bo w innym przypadku otrzymalibyśmy zawierający mnóstwo powtórzeń i niepotrzebnych danych iloczyn kartezjański. Ale z drugiej strony ten sam warunek złączenia usunął z wyniku rekordy niemające odpowiedników w łączonej tabeli. Czyli wynik instrukcji wcale nie musi zawierać danych wszystkich naszych pracowników.

Slajd 10

Czasami chcielibyśmy uzyskać komplet danych z jednej tabeli, nawet jeżeli nie są one powiązane z danymi w innych tabelach. Umożliwia nam to złączenie zewnętrzne. Wynikiem lewo- lub prawostronnego złączenia zewnętrznego jest zbiór wierszy łączonych tabel, dla których wartości kolumn określonych jako warunek złączenia są takie same; zbiór ten uzupełniony jest pozostałymi wierszami z lewej lub prawej łączonej tabeli. Nieistniejące wartości reprezentowane są w wyniku złączenia przez wartość NULL.

OUTER JOIN:
LEFT OUTER JOIN – nie pomija wyników z lewej tabeli
RIGHT OUTER JOIN – nie pomija wyników z prawej tabeli
OUTER JOIN – nie pomija wyników z żadnej tabeli

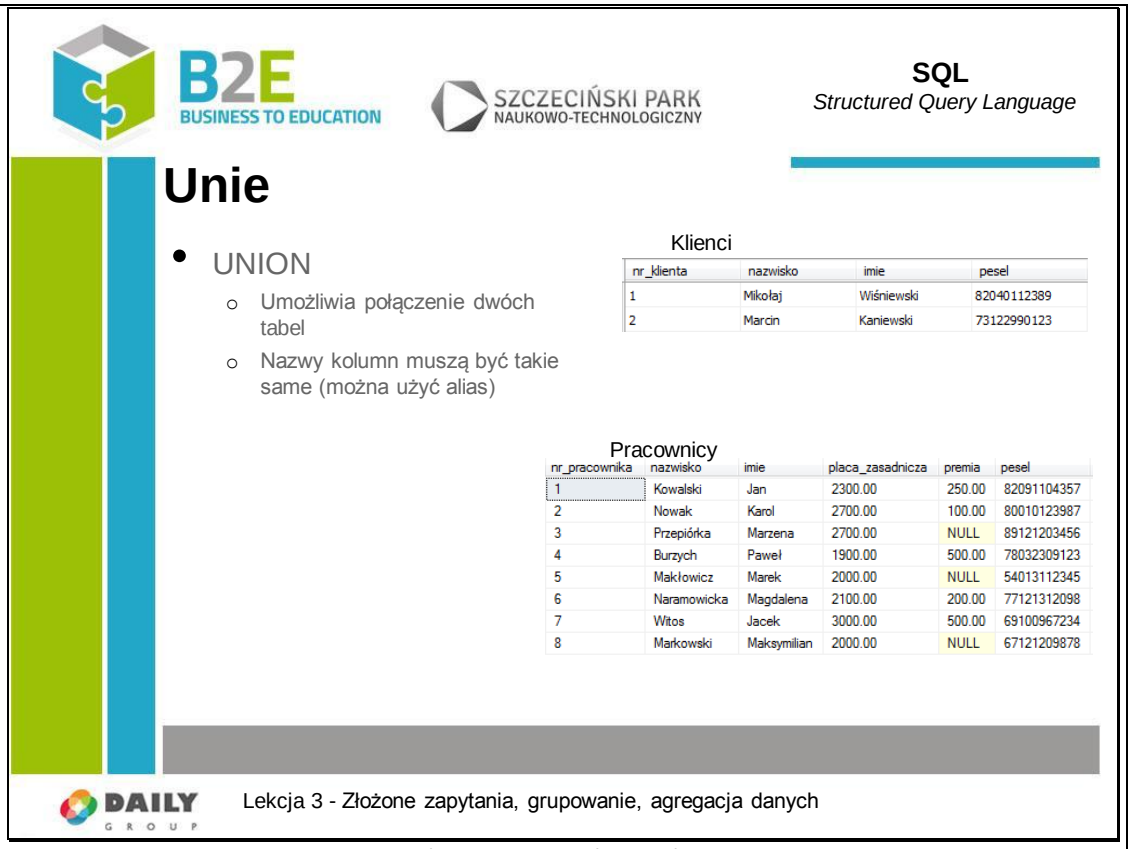
Slajd 11



Podsumowanie OUTER JOIN

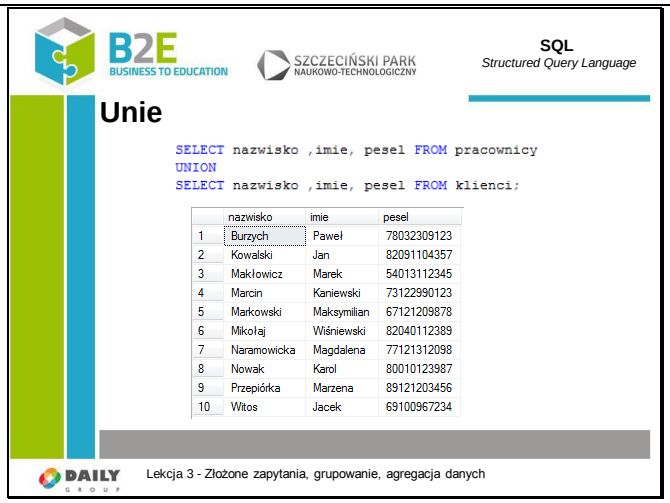
Slajd 12		Złączenie tabeli z nią samą wykonywane jest w taki sam sposób, jak omawiane do tej pory złączenia różnych tabel. Chociaż serwery bazodanowe nie tworzą kopii złączonej tabeli, to wszystkie operacje przeprowadzane są tak, jakby dotyczyły dwóch identycznych tabel. Złączenie tabeli z nią samą stosujemy, kiedy chcemy wybrać rekordy z tabeli na podstawie wspólnych wartości atrybutów rekordów tej samej tabeli.
Slajd 13		Złączenia tabeli z samą sobą są często wykorzystywane do rekurencyjnego odczytania danych, na przykład informacji o podwładnych (podwładny osoby *X* może być przełożonym osoby *Y*, która z kolei może być przełożonym osoby *Z*). W testowej bazie danych nie ma zapisanych takich zależności. Przykład jest czysto szkoleniowy.

Slajd 14

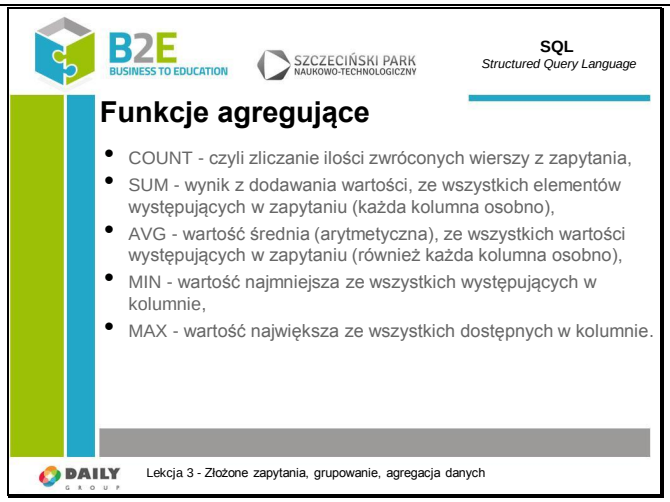


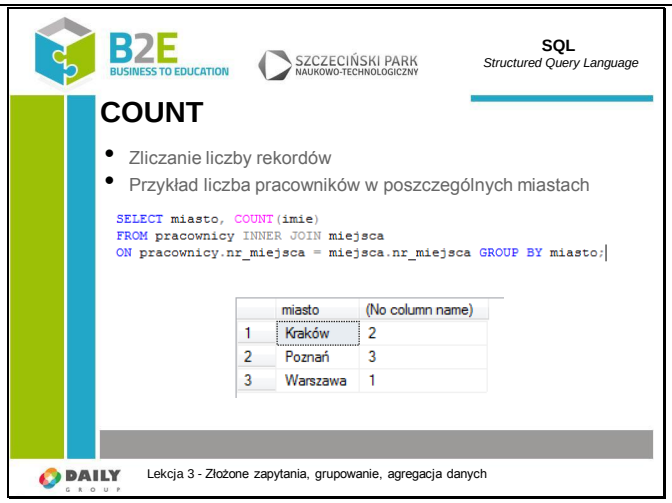
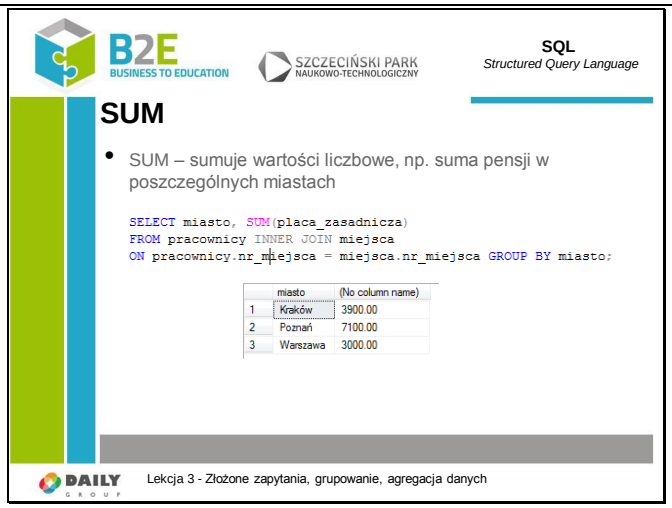
Unie umożliwiają łączenie tabel. Jeśli chcemy wyświetlić wszystkie osoby bez względu czy są to klienci czy pracownicy – możemy użyć właśnie polecenia UNION (warunkiem są takie same nazwy kolumn).
Ciekawostka: działania na zbiorach sortują wyniki i usuwają duplikaty. Wyjątkiem jest UNION ALL

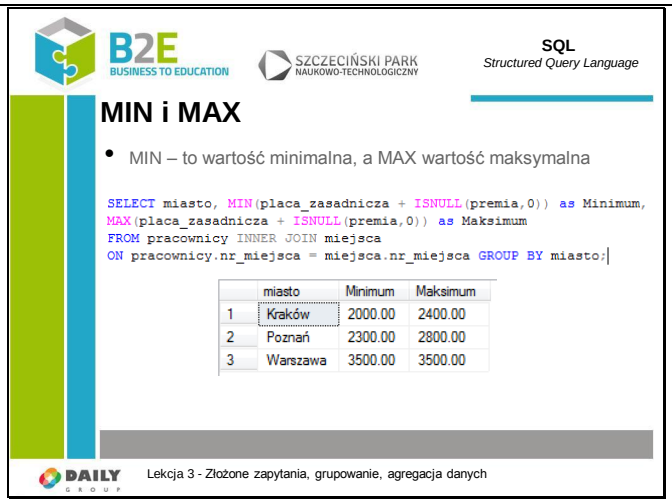
Slajd 15



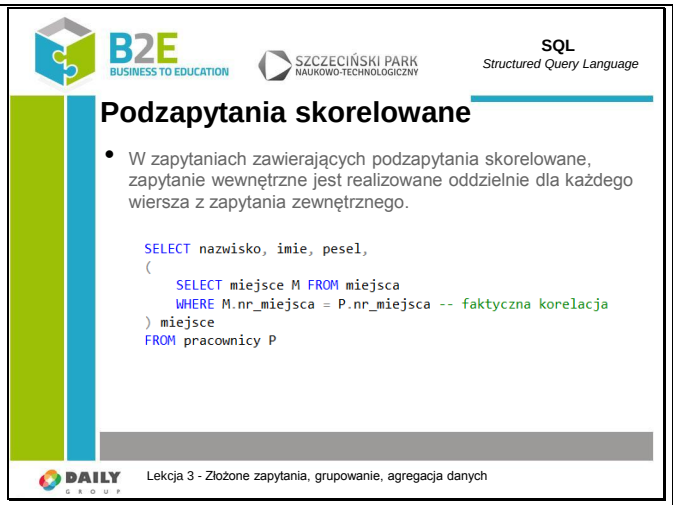
Powyższy przykład pokazuje jak w SQL połączyć dwie tabele razem. Polecenie UNION eliminuje duplikaty, a co za tym idzie sortuje wynik połączenia obu tabel. Rezultat będzie posortowany po nazwisku, a w dalszej kolejności po imieniu i numerze pesel

Slajd 16		Jeśli chcemy wyszukać wartości tabeli A, które nie znajdują się w tabeli B – korzystamy z polecenia EXCEPT. Konstrukcja jest identyczna jak przy UNION.
Slajd 17		Wprowadzenie do funkcji agregujących. Każda z nich pochodzi z j. angielskiego – w ten sposób można łatwo zapamiętać. Z wyjątkiem funkcji COUNT(*), wszystkie funkcje agregujące wykonują krok eliminacji NULL tak, że wartości NULL nie są uwzględniane w ostatecznym wyniku obliczeń.
Slajd 18		SQL jest językiem zapytań zorientowanym na przetwarzanie zbiorów, a nie pojedynczych rekordów, dlatego jego integralnym składnikiem są polecenia grupujące dane. Właśnie te polecenia, wraz z funkcjami sumującymi, stosowane są w praktyce do uzyskiwania odpowiedzi na złożone zapytania, dotyczące zawartości bazy danych. Przykład ilustruje użycie klauzuli GROUP BY.

Slajd 19		Funkcja COUNT liczy wystąpienia bazie danych w tym przypadku zliczamy dla każdego z miast – ilu pracowników jest zatrudnionych w konkretnej lokalizacji. W klauzuli *group by* wskazujemy po czym mają być grupowane dane.
Slajd 20		Przykład wykorzystania funkcji SUM
Slajd 21		

Slajd 22		
Slajd 23		Klauzula HAVING służy do ograniczania zbioru wierszy, zwracanych w wyniku wykonania klauzuli GROUP BY. Między HAVING a GROUP BY występuje zależność podobna do tej, która wiąże klauzulę WHERE z poleceniem SELECT. W przeciwieństwie do klauzuli WHERE, która operuje na wierszach tabel podawanych w zapytaniu, HAVING działa na zbiorze wierszy wynikowych
Slajd 24		Podzapytanie jest zapytaniem umieszczonym wewnątrz innego zapytania, tzw. zapytania zewnętrznego. Podzapytania najczęściej umieszcza się w warunkach w klauzulach WHERE i HAVING zapytania zewnętrznego, niektóre SZBD dopuszczają również stosowanie podzapytań w klauzulach SELECT i FROM. Podzapytania stosuje się w ostateczności, gdy nie można zastosować połączenia typu UNION lub JOIN.

Slajd 25



W zapytaniach zawierających podzapytania skorelowane, zapytanie wewnętrzne jest realizowane oddzielnie dla każdego wiersza z zapytania zewnętrznego.

Slajd 26

Ćwiczenie 1:
Pokaż w jakich miejscowościach pracują pracownicy
Ćwiczenie 2:
Pokaż pracowników z miejscowości Warszawa
Ćwiczenie 3:
Pokaż wszystkie osoby (klienci i pracownicy)
Ćwiczenie 4:
Pokaż pracowników i ich przełożonych
Ćwiczenie 5:
Pokaż najniższą pensję całej firmie
Ćwiczenie 6:

	Pokaż średnią pensję w podziale na miejscowości Ćwiczenie 7: Pokaż miejscowość, gdzie pracuje najwięcej pracowników Ćwiczenie 8: Pokaż ilość pracowników w każdej miejscowości Ćwiczenie 9: Pokaż średnią pensję specjalistów i managerów Ćwiczenie 10: Pokaż najwyższą pensję średnią premie w podziale na stanowiska	
Slajd 27		SELECT TOP 3 * FROM TBL_UCZEN WHERE IMIE=„Anna”
Slajd 28		SELECT AVG(OCENA) FROM TBL_UCZEN WHERE KLASA=„3A”

Slajd 29		SELECT COUNT(*) FROM TBL_UCZEN WHERE KLASA=„3A”
Slajd 30		SELECT AVG(OCENA) FROM TBL_UCZEN GROUP BY KLASA=„3A”

Slajd 31

Tabela podsumowująca poznane wyrażenia

9.3.3 Ćwiczenia

Ćwiczenia zostały przedstawione na slajdach 26-30. Uczniowie w ramach ćwiczenia będą musieli napisać zapytania SQL na bazie tabel przygotowanych przez szkoleniowca.

9.3.4 Opis założonych osiągnięć ucznia

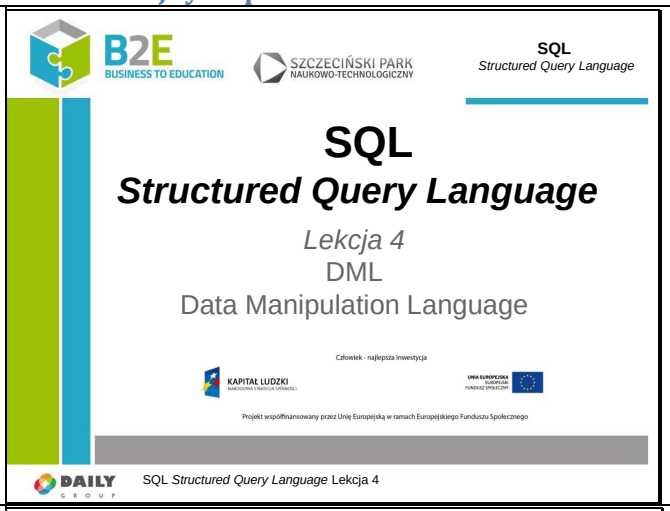
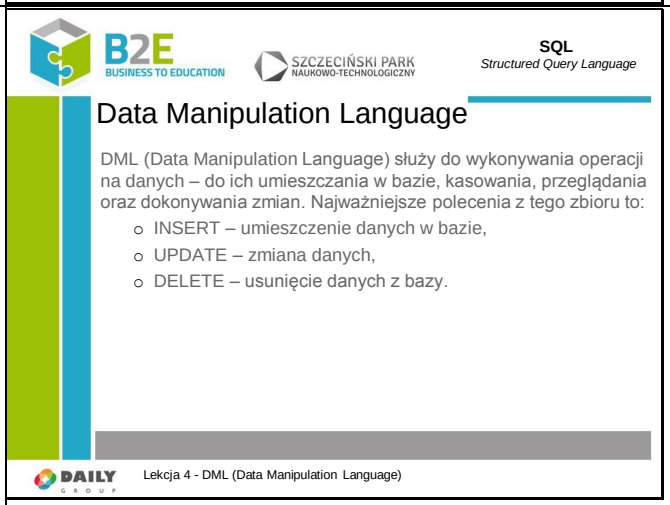
Po tej lekcji uczniowie będą potrafić budować złożone zapytania SQL, łączyć wiele tabel ze sobą na różne sposoby, grupować dane w tabelach, wyliczać średnie, wartości minimalne i maksymalne, liczyć wiersze w tabeli. Uczniowie powinni być w stanie budować zapytania złożone z podzapytań.

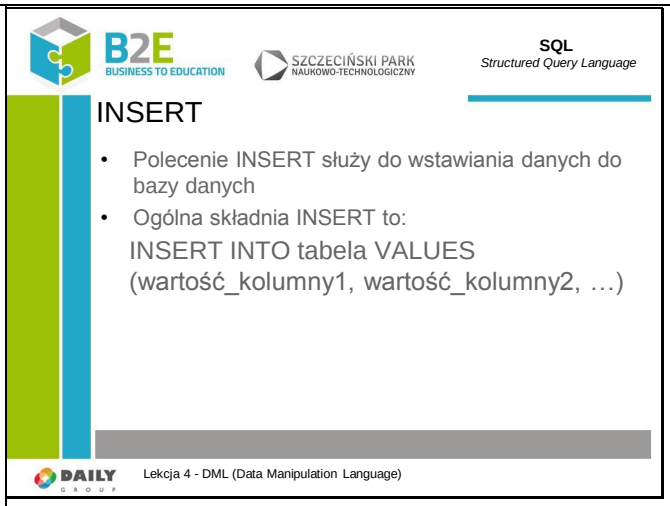
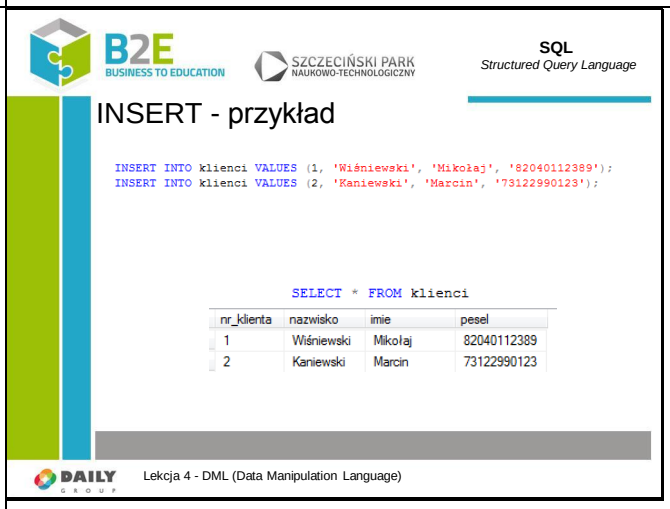
9.4 Lekcja 4 – DML

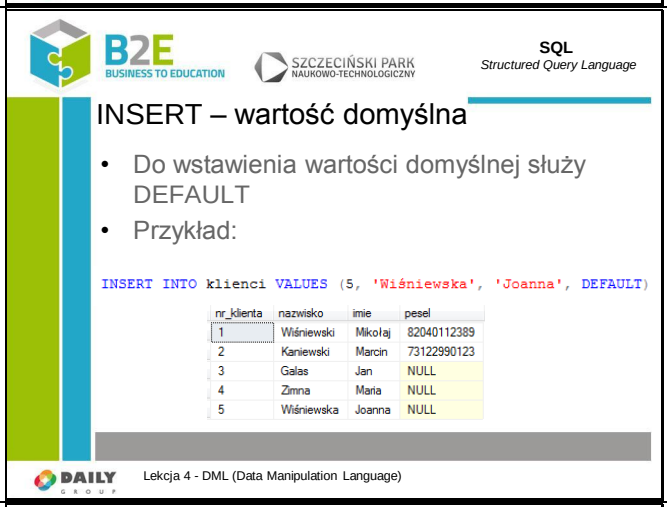
9.4.1 Cel lekcji

Celem lekcji jest wyjaśnienie języka DML, szczegółowe zapoznanie z podstawowymi poleceniami tego języka: INSERT, UPDATE oraz DELETE.

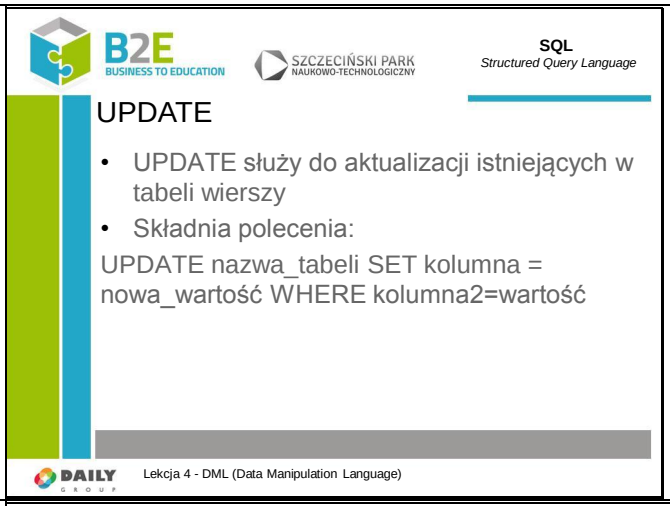
9.4.2 Treść - slajdy z opisem

Slajd 1		DML (Data Manipulation Language) służy do wykonywania operacji na danych – do ich umieszczania w bazie, kasowania, przeglądania oraz dokonywania zmian. Najważniejsze polecenia z tego zbioru to: INSERT, UPDATE oraz DELETE
Slajd 2		
Slajd 3		Część osób do DML zalicza także zapytania typu SELECT, jednak przeglądanie danych nie do końca można zaliczyć jako manipulacja danymi – w tej kwestii informatycy nie są zgodni.

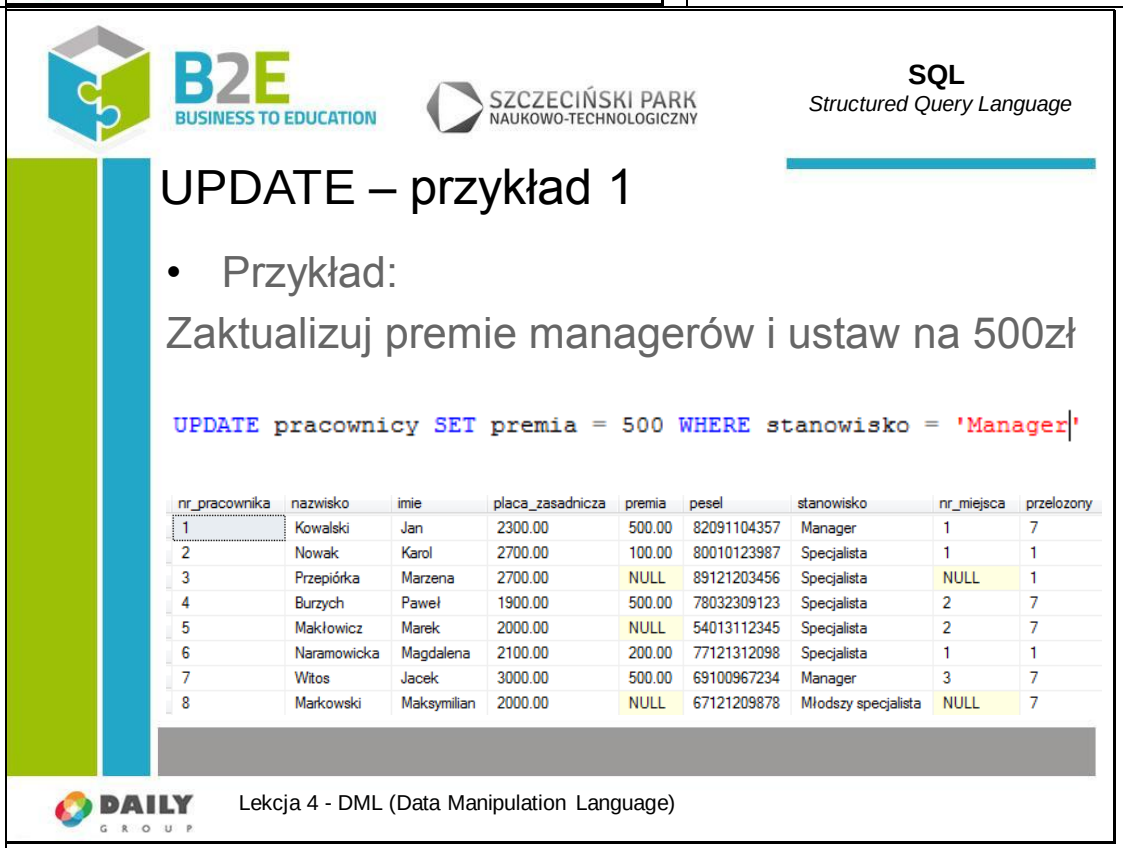
Slajd 4		Do wstawiania wierszy do tabeli służy polecenie INSERT. Najprostszą wersję tego polecenia przedstawiono na przykładzie. Polecenie zaczyna się od słów kluczowych INSERT INTO po którym podaje się nazwę tabeli, słowo kluczowe VALUES i w nawiasie listę wartości oddzielonych przecinkami. Wartości na tej liście odpowiadają kolejnym atrybutom relacji (w kolejności w jakiej atrybuty te zostały zdefiniowane). Wpisami na liście wartości mogą być: słowa kluczowe NULL i DEFAULT oraz konkretne wartości atrybutów i ewentualnie podzapytania zwracające jedną wartość. Jeżeli zamiast konkretnej wartości poda się NULL, wówczas nowy wiersz będzie miał pustą wartość atrybutu odpowiadającego pozycji na której wpisano NULL. Podanie DEFAULT spowoduje, że w odpowiedniej kolumnie zostanie zapisana jego wartość domyślna.
Slajd 5		Przykład umieszczania danych w bazie. Za pomocą dwóch wierszy polecenia INSERT wstawiamy do bazy danych numer klienta, nazwisko, imię oraz PESEL klienta. Poniżej sprawdzamy zawartość bazy – upewniamy się że wartości zostały odpowiednio umieszczone w bazie danych.

Slajd 6		Umieszczanie pustych wartości w wierszach można zrealizować w dwojaki sposób: albo używany wartości pustej NULL w miejscach gdzie nie znamy wartości, albo wymieniamy po nazwie tabeli wartości które zamierzamy uzupełnić. Oba sposoby dają identyczny efekt
Slajd 7		Kolumny tabeli posiadają często wartość domyślną. Użycie polecenia DEFAULT w poleceniu SQL umożliwia wstawienie wartości domyślnej. W przykładzie użyto DEFAULT dla kolumny pesel, której wartością domyślną jest wartość pusta - NULL.
Slajd 8		Wstawianie wielu wierszy pojedynczo jest bardzo czasochłonne. W języku SQL możemy wykorzystać zawartość innych tabel do uzupełnienia tabeli. W przykładzie użyto danych z tabeli pracownicy w celu wstawienia danych do tabeli klienci.

Slajd 9



Slajd 10



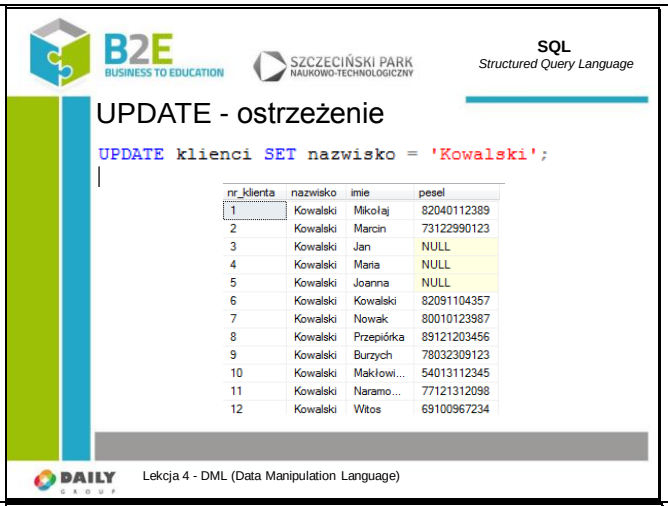
nr_pracownika	nazwisko	imie	placa_zasadnicza	premia	pesel	stanowisko	nr_miejsca	przelozony
1	Kowalski	Jan	2300.00	500.00	82091104357	Manager	1	7
2	Nowak	Karol	2700.00	100.00	80010123987	Specjalista	1	1
3	Przepiórka	Marzena	2700.00	NULL	89121203456	Specjalista	NULL	1
4	Burzych	Paweł	1900.00	500.00	78032309123	Specjalista	2	7
5	Makłowicz	Marek	2000.00	NULL	54013112345	Specjalista	2	7
6	Naramowicka	Magdalena	2100.00	200.00	77121312098	Specjalista	1	1
7	Witos	Jacek	3000.00	500.00	69100967234	Manager	3	7
8	Markowski	Maksymilian	2000.00	NULL	67121209878	Młodszy specjalista	NULL	7

Polecenie zmienia premię na 500 dla wszystkich rekordów, gdzie stanowisko to Manager

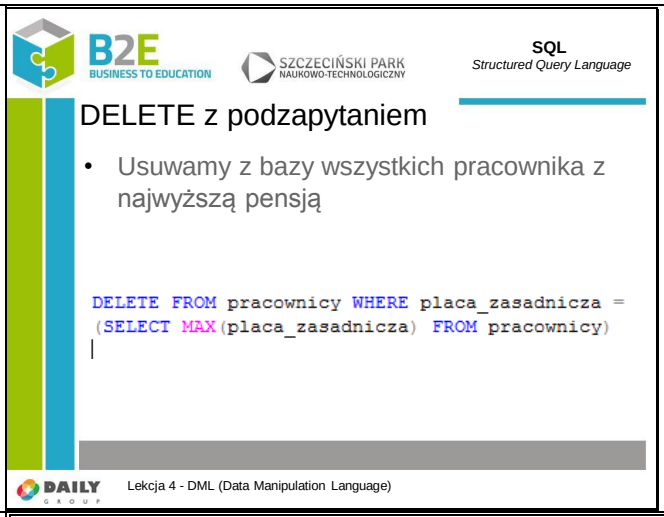
Slajd 11	B2E BUSINESS TO EDUCATION · SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY
	SQL *Structured Query Language*
	UPDATE – przykład 2
	• Przykład:
	Powiększ wynagrodzenie Kowalskiego i Nowaka o 20%
	`UPDATE pracownicy SET placa_zasadnicza = placa_zasadnicza*1.2`
	`WHERE nazwisko = 'Kowalski' OR nazwisko = 'Nowak'`
	`UPDATE pracownicy SET placa_zasadnicza = placa_zasadnicza*1.2`
	`WHERE nazwisko IN ('Kowalski', 'Nowak')`
	DAILY GROUP Lekcja 4 - DML (Data Manipulation Language)
		Oba polecenia wykonują taką samą operację, tj. powiększają kwotę płacy zasadniczej o 20% (mnożą płacę przez 1,2) w rekordach gdzie nazwisko to Nowak lub Kowalsk

Slajd 12	B2E BUSINESS TO EDUCATION · SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY
	SQL *Structured Query Language*
	UPDATE – przykład 3
	• Przykład:
	Zmień nazwisko pracownika z Naramowicka na Naramowicka-Przybylska
	`UPDATE pracownicy SET nazwisko = 'Naramowicka-Przybylska'`
	`WHERE nazwisko = 'Naramowicka'`
	DAILY GROUP Lekcja 4 - DML (Data Manipulation Language)
		Przykład aktualizuje pole nazwisko gdzie nazwisko to Naramowicka

Slajd 13	B2E BUSINESS TO EDUCATION · SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY SQL *Structured Query Language* **UPDATE - ostrzeżenie** **UWAGA!!!!** • Proszę pamiętać, aby w zapytaniu UPDATE definiować odpowiednie wartości dla WHERE. • W przypadku braku zdefiniowanego warunku WHERE w tabeli zostaną zmienione WSZYSTKIE wiersze! DAILY GROUP Lekcja 4 - DML (Data Manipulation Language)	Bardzo ważna uwaga: wiele początkujących osób zapomina dodać klauzulę WHERE do polecenia UPDATE, skutkuje to zastąpieniem wartości we wszystkich wierszach

Slajd 14		Bardzo ważna uwaga: wiele początkujących osób zapomina dodać klauzulę WHERE do polecenia UPDATE, skutkuje to zastąpieniem wartości we wszystkich wierszach
Slajd 15		Poznaliśmy sposób umieszczania danych w tabeli, ich modyfikacji – teraz czas na usuwania danych. DELETE to ostatnie z wyrażeń DML. Podobie jak w przypadku UPDATE, należy zwrócić szczególną uwagę na warunek WHERE, tak aby nie skasować wszystkich danych z tabeli.
Slajd 16		Przykład usuwania rekordu – poniżej widzimy odpowiedź systemu. Widzimy, że został usunięty jeden wiersz.

Slajd 17



W warunku WHERE w poleceniu DELETE możemy zagnieżdżać podzapytania – podobnie jak w SELECT.

Slajd 18

Ćwiczenie 1:

Wstaw dwie miejscowości do tabeli Miejsca

Ćwiczenie 2:

Dodaj po dwóch pracowników do każdej z wstawionych w ćw. 1 lokalizacji

Ćwiczenie 3:

Podnieś wszystkim premie o 30zł, pamiętaj o osobach, które dotychczas nie miały premii

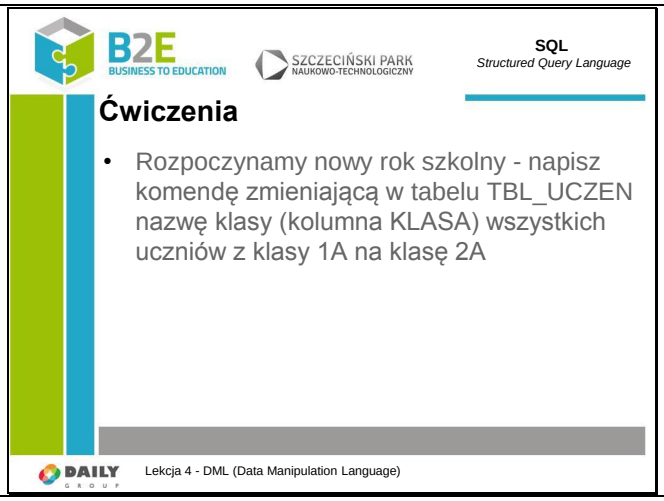
Ćwiczenie 4:

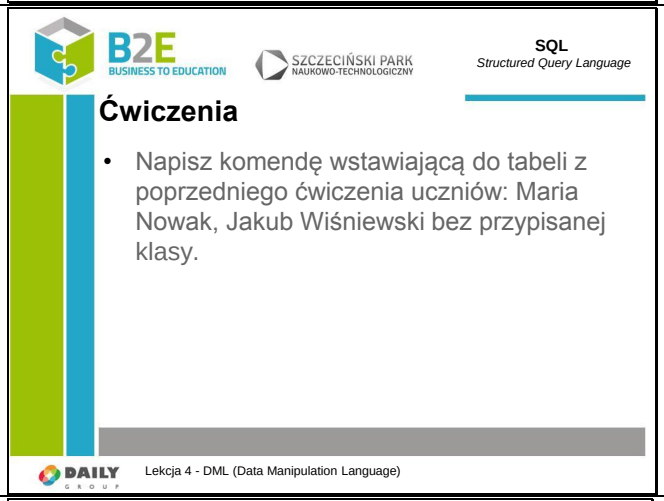
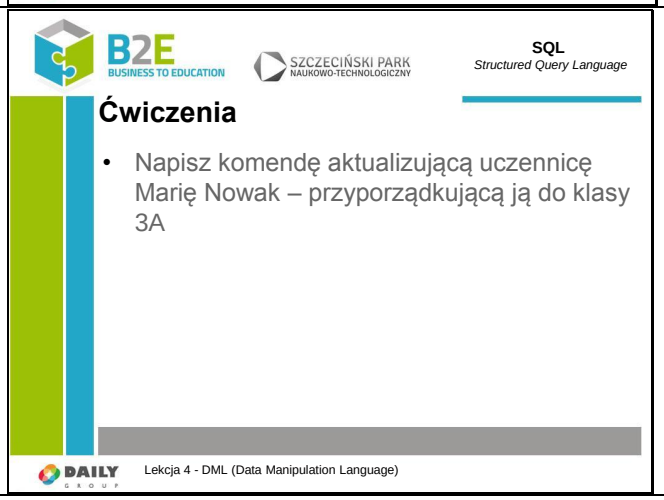
Zatrudniamy naszych klientów, wstaw do tabeli pracownicy osoby z tabeli klienci

Ćwiczenie 5:

Zmień miejsce zatrudnienia osób pracujących w Warszawie – przenosimy biuro do Szczecina

Ćwiczenie 6:

	Rezygnujemy z biura w Warszawie – usuń lokalizację z tabeli miejsca Ćwiczenie 7: Zwiększ pensje zasadnicze pracowników ze Szczecina o 30% Ćwiczenie 8: Osoby, które dotychczas nie miały przypisanej lokalizacji zostają przypisane do Szczecina – uaktualnij bazę danych Ćwiczenie 9: Znaleźliśmy nowych klientów – wprowadź kilka dodatkowych osób Ćwiczenie 10: Zaktualizuj nazwisko pani Marzeny – po ślubie zmieniła nazwisko na Nowak	
Slajd 19		DELETE FROM TBL_UCZEN WHERE KLASA=„4A”
Slajd 20		UPDATE TBL_UCZEN SET KLASA = „2A” WHERE KLASA = „1A”

Slajd 21		INSERT INTO TBL_UCZEN(IMIE, NAZWISKO, KLASA) VALUES („Jan”, „Kowalski”, „1A”)
Slajd 22		INSERT INTO TBL_UCZEN(IMIE, NAZWISKO, KLASA) VALUES („Maria”, „Nowak”, NULL) INSERT INTO TBL_UCZEN(IMIE, NAZWISKO, KLASA) VALUES („Jakub”, „Wiśniewski”, NULL)
Slajd 23		UPDATE TBL_UCZEN SET KLASA = „1A” WHERE IMIE = „Maria” AND NAZWISKO=„Nowak”

Slajd 24		Tabela podsumowująca poznane wyrażenia

9.4.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 19-23. Uczniowie napiszą instrukcje modyfikujące dane w bazie danych.

9.4.4 Opis założonych osiągnięć ucznia

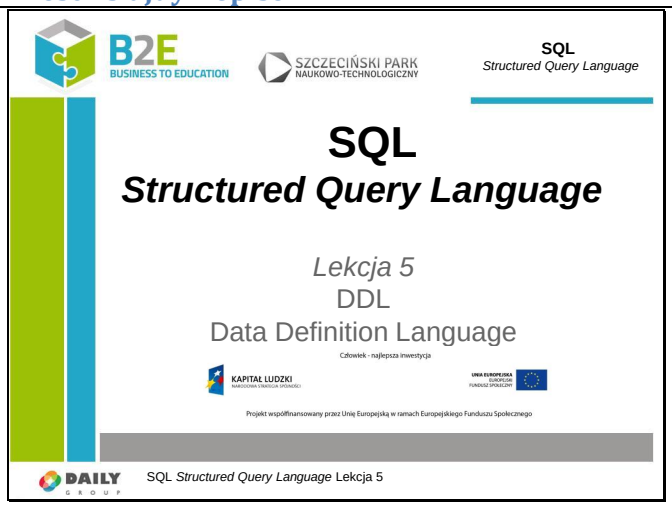
We wcześniejszych lekcjach uczniowie zapoznali się z poleceniem SELECT, potrafią już odczytywać dane z bazy. W tej lekcji nauczyli się modyfikować dane – wstawiać nowe rekordy, aktualizować i modyfikować istniejące.

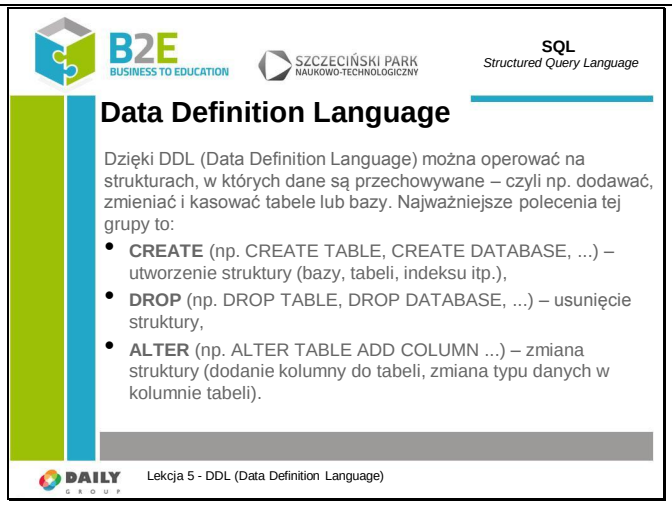
9.5 Lekcja 5 - DDL

9.5.1 Cel lekcji

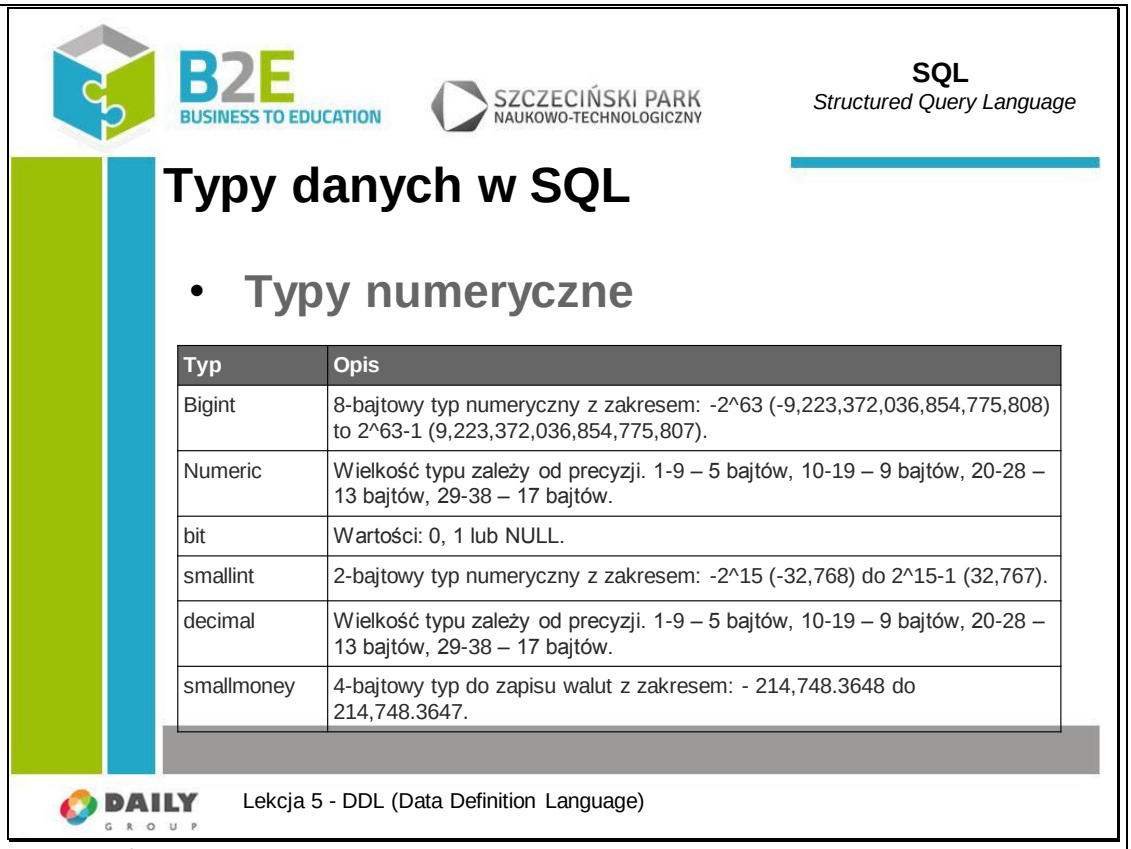
Celem lekcji jest zapoznanie się z językiem DDL. W tej części kursu uczniowie zapoznają się z mechanizmami tworzącymi tabele w bazie danych. Przed przystąpieniem do definiowania tabel, uczniowie zostaną zapoznani z typami danych w języku SQL.

9.5.2 Treść - slajdy z opisem

Slajd 1		Kolejna dziedzina języka SQL – DDL – Data Definition Language, służy do definiowania obiektów bazy danych. Nauczyliśmy się wyświetlać dane z bazy (SELECT) oraz je modyfikować (DML: INSERT, UPDATE, DELETE). Przyszedł czas na tworzenie, modyfikację i usuwanie baz danych, tabel i innych obiektów.

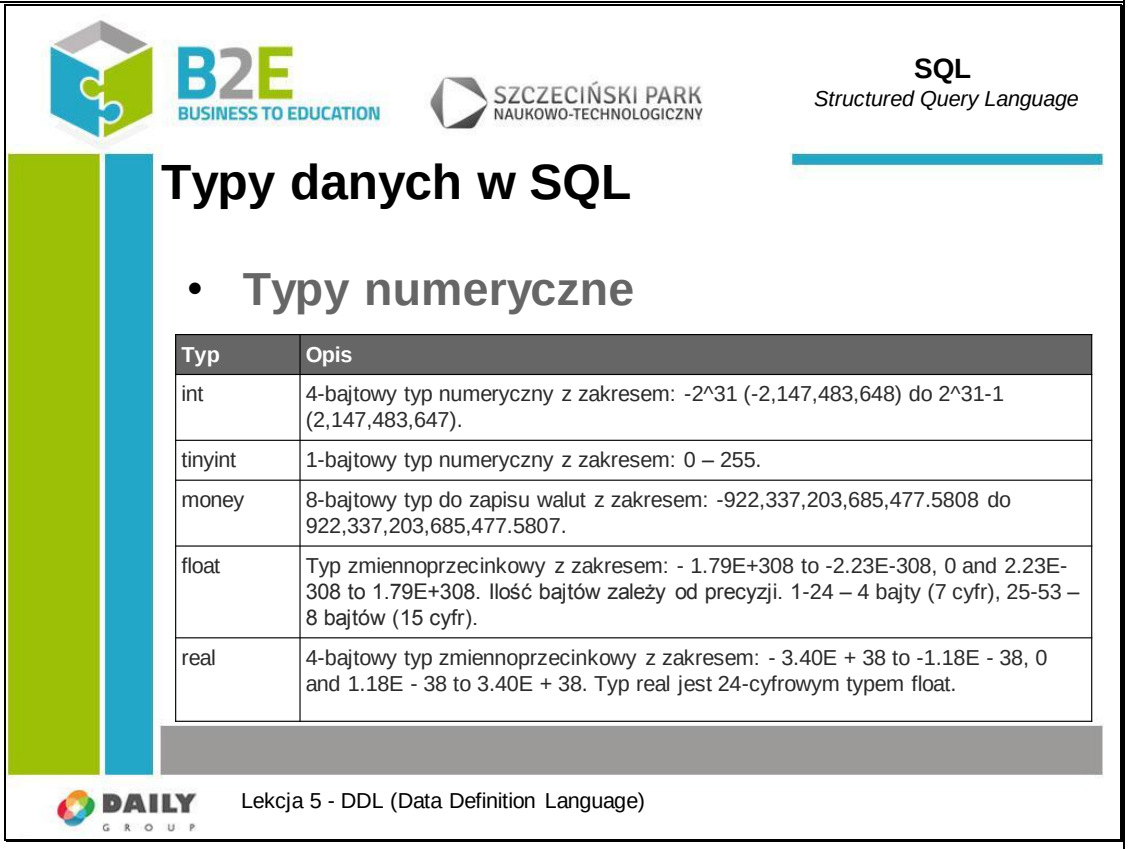
Slajd 2		UPDATE służy do aktualizacji istniejących danych w tabeli. INSERT używane jest do dodawania nowych wierszy do tabeli. INSERT pozwala na dodawanie nowych wierszy na dwa sposoby. W jednym z nich definiujemy nazwy kolumn, zaś w drugim pomijamy nazwy kolumn wprowadzając same wartości. W przypadku drugiego sposobu, musimy wziąć pod uwagę kolejność kolumn w tabeli. DELETE używane jest do usuwania istniejących wierszy z tabeli.
Slajd 3		Wyjaśnienie, że pomimo wspólnego standardu istnieje wiele różnych rozszerzeń. Podstawowa składnia wszystkich rozszerzeń języka jest taka sama, jednak istnieje wiele różnic. W tym kursie zajmiemy się językiem SQL w ujęciu rozszerzenia TSQL, promowanym przez Microsoft.
Slajd 4		Zanim będziemy mogli stworzyć tabelę musimy poznać typy danych (w tym kolumn), które możemy użyć w języku SQL. Większość z przedstawionych typów dotyczy wszystkich rozszerzeń języka SQL. Jednak warto pamiętać, że w tym kursie skupiamy się na języku TSQL

Slajd 5

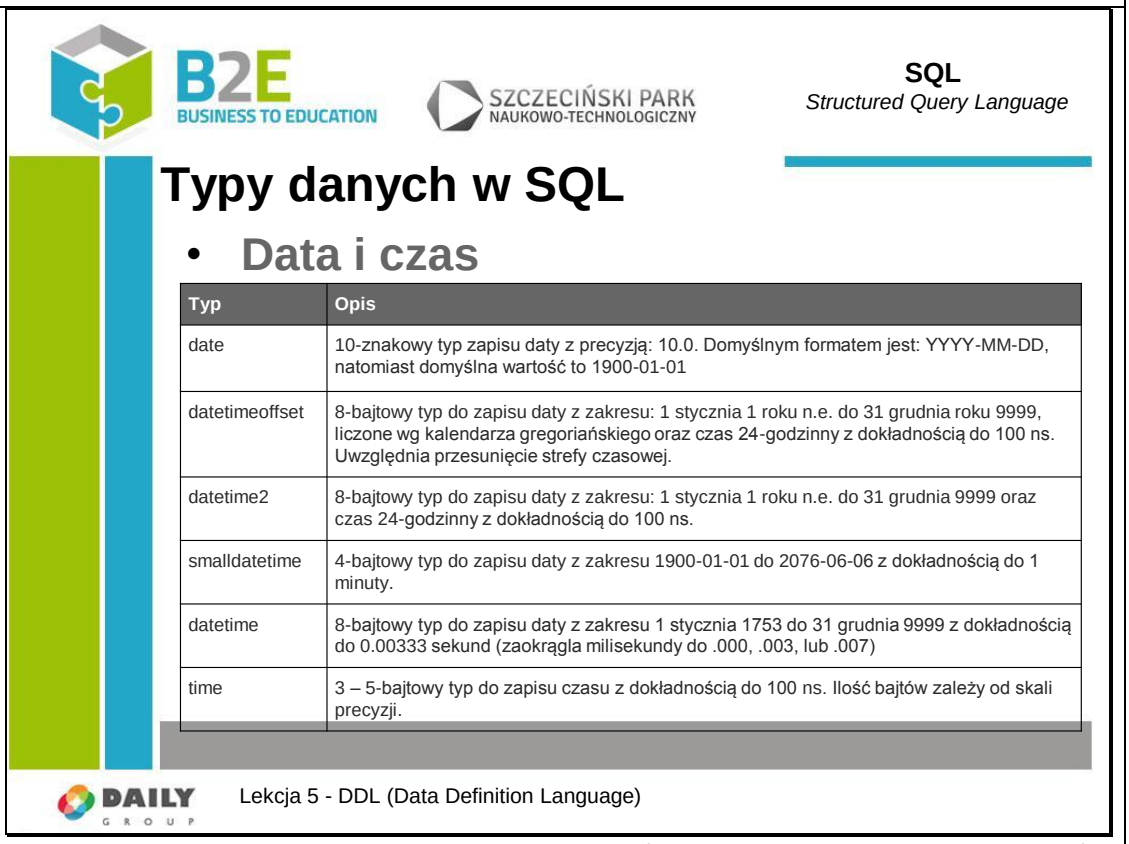
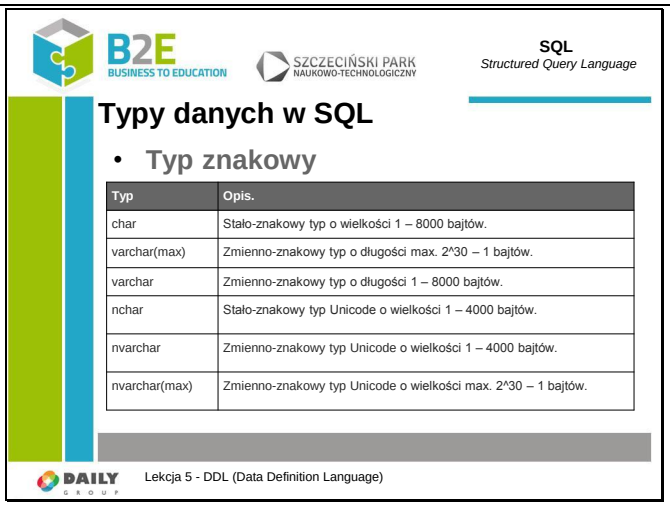


Do wartości numerycznych zalicza się typy przedstawione w tabeli.

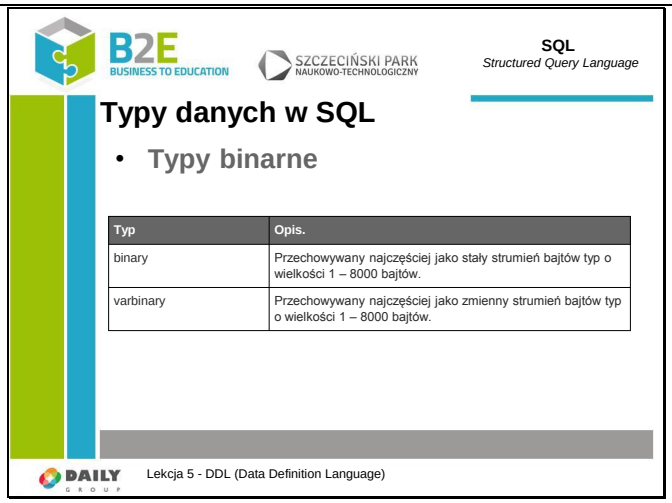
Slajd 6



Zanim będziemy mogli stworzyć tabelę musimy poznać typy danych (w tym kolumn), które

	możemy użyć w języku SQL. Większość z przedstawionych typów dotyczy wszystkich rozszerzeń języka SQL. Jednak warto pamiętać, że w tym kursie skupiamy się na języku TSQL
Slajd 7	 Dane zapisywane jako data i czas powinny zostać zaprezentowane za pomocą typów przedstawionych w tabeli.
Slajd 8		Do wartości znakowych zalicza się typy przedstawione w tabeli

Slajd 9



Do wartości binarnych zalicza typy przedstawione w tabeli

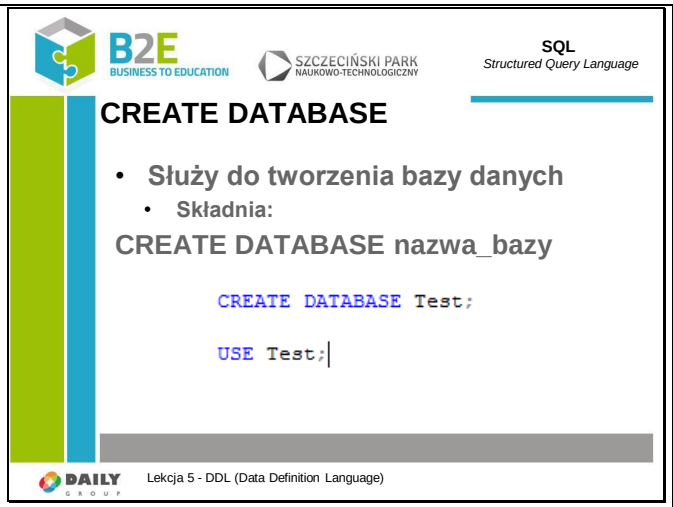
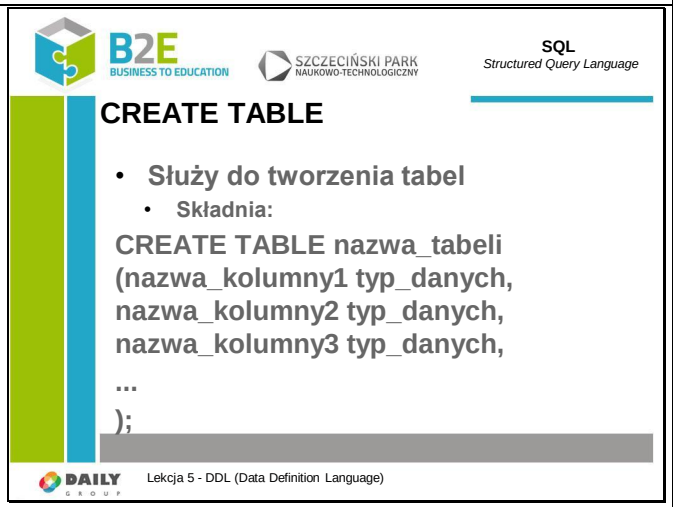
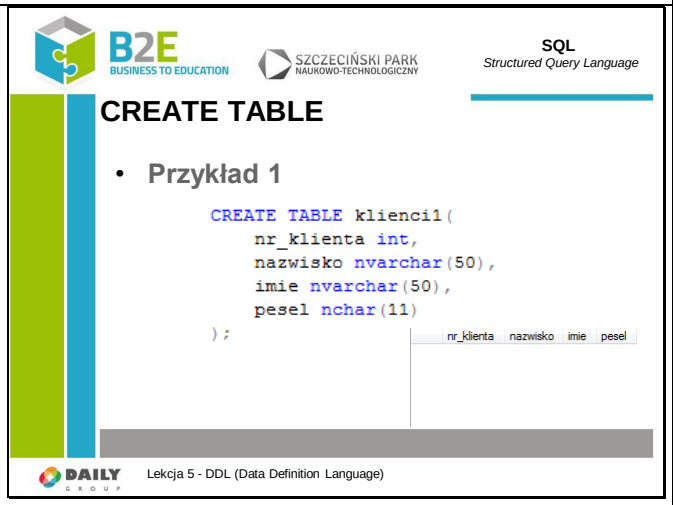
Slajd 10

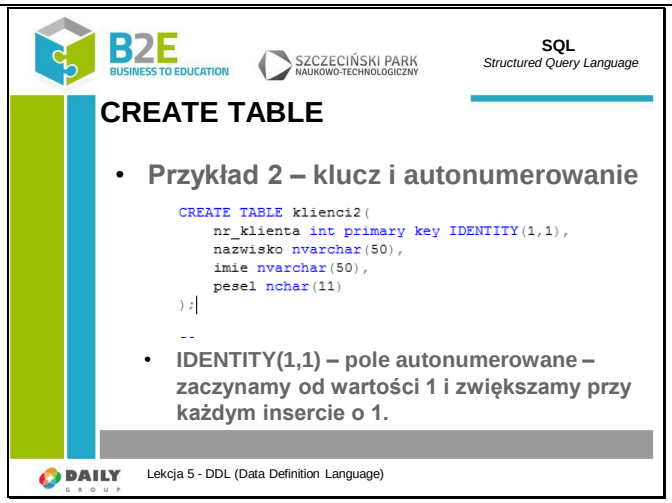
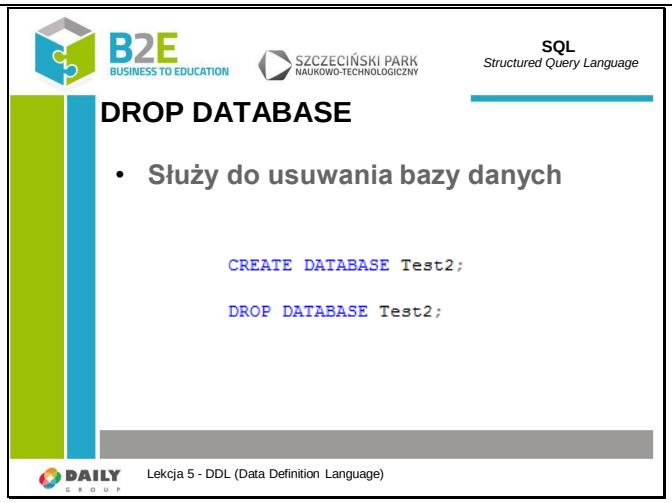
SQL Server oferuje typy do reprezentowania obiektów geometrycznych i geograficznych

Slajd 11

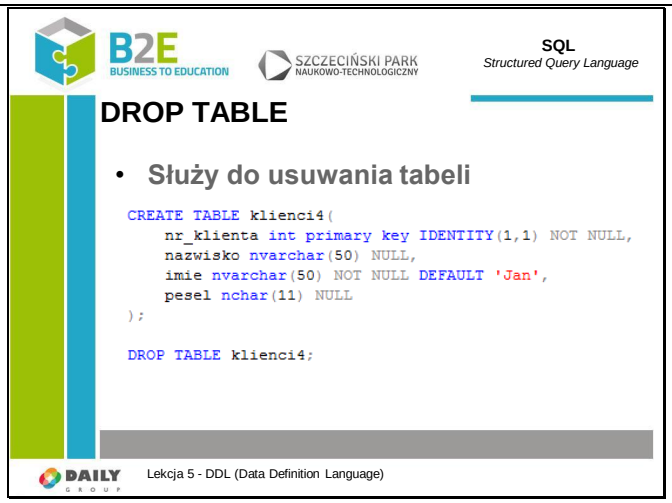
Pozostałe typy zostały przedstawione w tabeli

Slajd 12

Slajd 13		Pamiętajmy, że SZBD może zawierać wiele baz. Często używamy skrótu myślowego i nazywamy bazą danych SZBD, faktycznie system zarządzania bazą danych zawiera wiele baz danych. USE wykorzystywany jest do wyboru bazy danych, w kontekście której zamierzamy pracować.
Slajd 14		CREATE służy do tworzenia nowych tabel, w których później możemy przechowywać różnego rodzaju dane. Tworząc nową tabelę musimy podać jej nazwę, nazwę kolumny/kolumn oraz typ danych danej kolumny. Można też podawać dodatkowe informacje, podczas tworzenia tabeli takie jak np. kodowanie znaków itd.
Slajd 15		Przykład CREATE tworzący tabelę klienci. Tworząc tabele trzeba zwrócić uwagę na to, jaki typ danych będziemy przechowywać, aby był on najlepiej dopasowany. Jeżeli przechowujemy wyłącznie liczby, to nie ma potrzeby tworzenia kolumny typu tekstowego itd.

Slajd 16		Najczęściej chcemy posiadać kolumnę w tabeli, która będzie przechowywała unikalne wartości automatycznie numerowane. Zastosowanie tego jest bardzo proste. Dzięki takiemu rozwiązaniu każdy wiersz w tabeli posiada unikalny "identyfikator". W tym celu możemy wykorzystać **IDENTITY** oraz definicję klucza głównego. Ciekawostka: Z punktu widzenia programisty, IDENTITY jako sposób na klucz główny (PK), jest straszną bolączką ORMów (mapowań relacyjno obiektowych).
Slajd 17		Tworząc tabelę mamy wpływ na to czy dana kolumna będzie dopuszczała wartości puste, bądź nie. Służy do tego wyrażenie NULL lub NOT NULL po typie zmiennej. Dodatkowo możemy kolumnie nadać wartość domyślną – polecenie DEFAULT.
Slajd 18		DROP DATABASE usuwa całą bazę danych. Warto zwrócić uwagę, że operacja jest nieodwracalna i należy z niej korzystać ze szczególną ostrożnością.

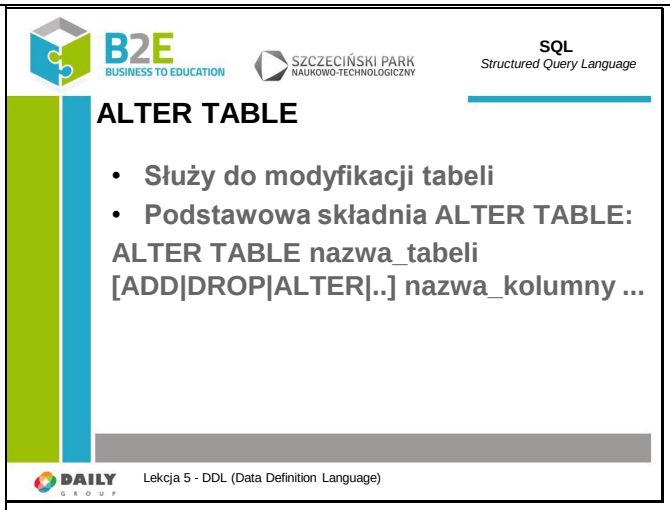
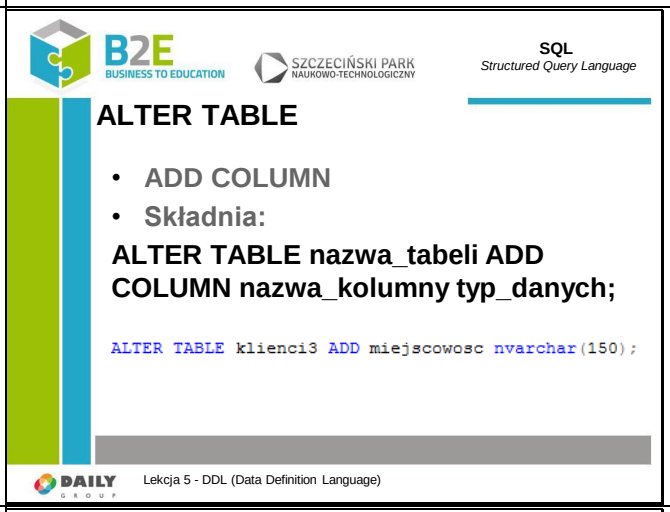
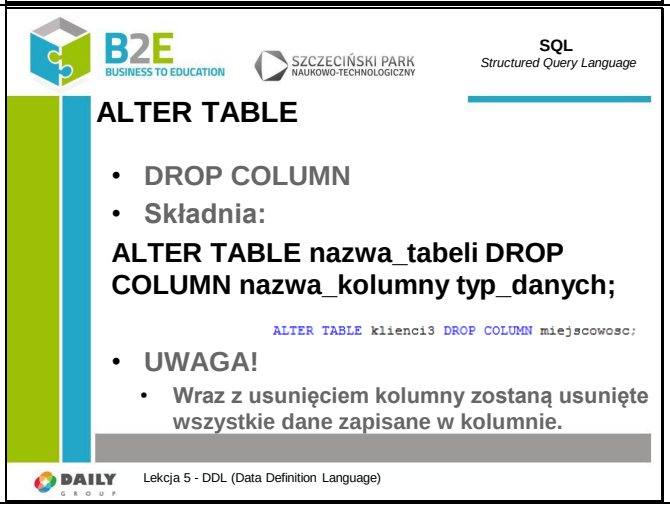
Slajd 19

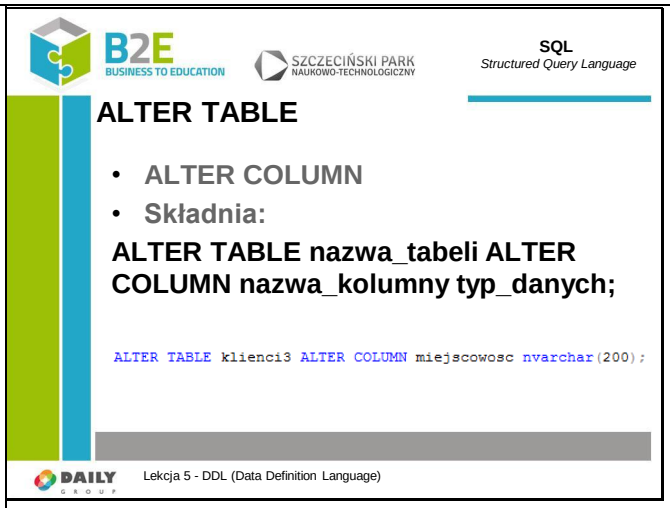
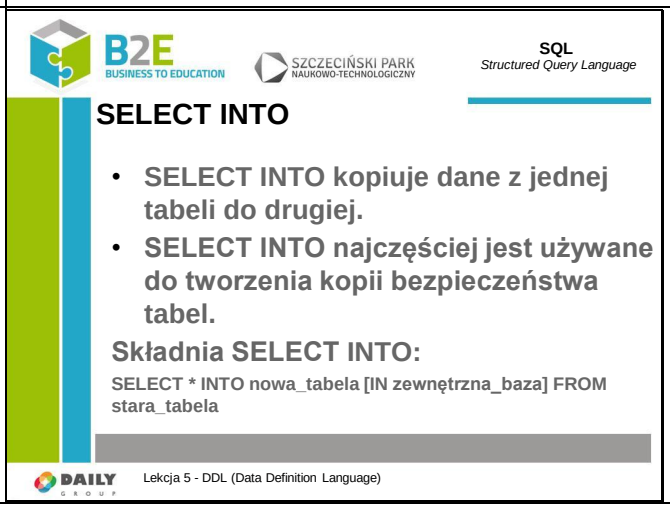


DROP TABLE usuwa tabelę. Warto zwrócić uwagę, że operacja jest nieodwracalna i należy z niej korzystać ze szczególną ostrożnością.

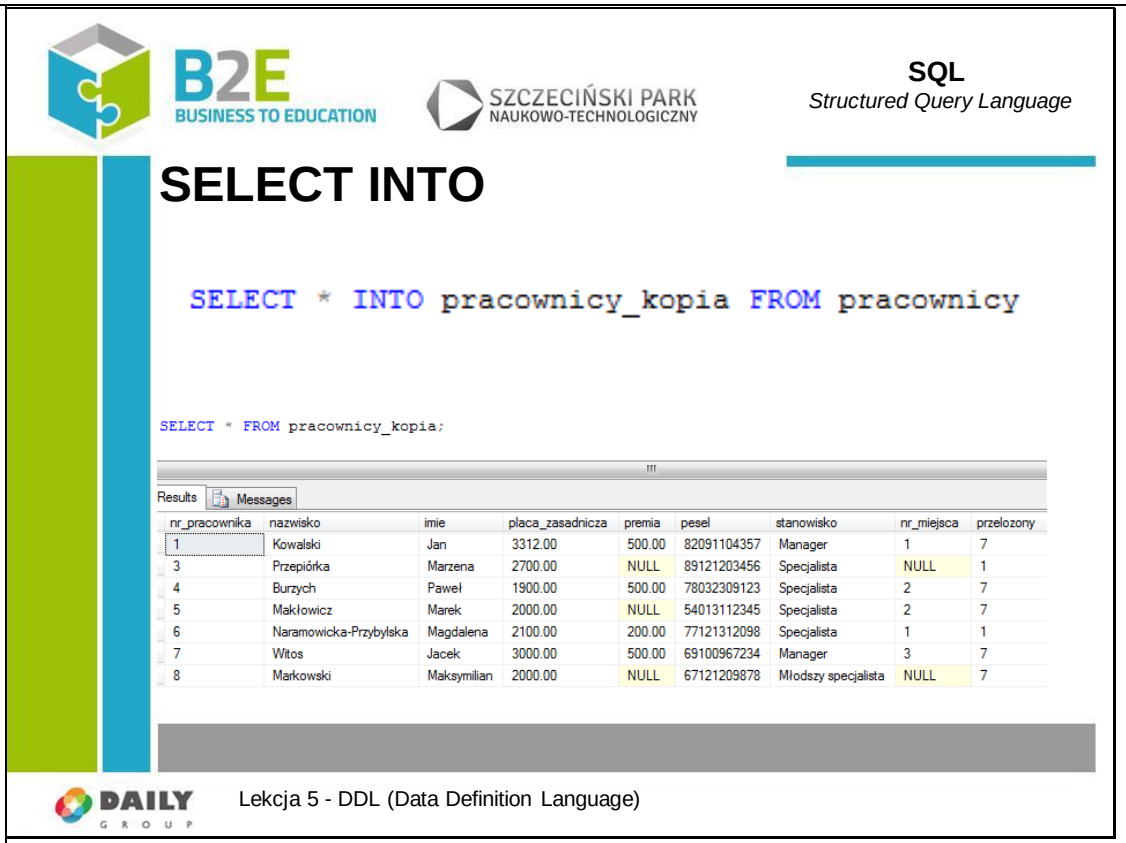
Slajd 20

ALTER służy głównie służy do modyfikacji istniejących elementów w bazie. Początkujący użytkownicy będą głównie korzystali z polecania **ALTER TABLE**, które będzie odnosiło się do tabel. Poleceniem tym możemy modyfikować tabele między innymi dodając i usuwając kolumny, zmieniając kolumny oraz typy kolumn.
W TSQL posiadamy inne typy zapytań **ALTER**, np. ALTER DATABASE (odnosi się do bazy danych), ALTER FUNCTION (odnosi się do funkcji), ALTER USER (odnosi się do użytkowników), ALTER VIEW (odnosi się do widoków), ALTER PROCEDURE (odnosi się do procedur).
Na slajdzie przedstawiono możliwe opcje ALTER DATABASE. Najważniejsze z nich to zmiana kodowania, szyfrowania, opcji kursorów.

Slajd 21		
Slajd 22		Przykład ALTER TABLE dodający kolumnę do istniejącej tabeli.
Slajd 23		Przykład ALTER TABLE usuwający kolumnę z istniejącej tabeli.

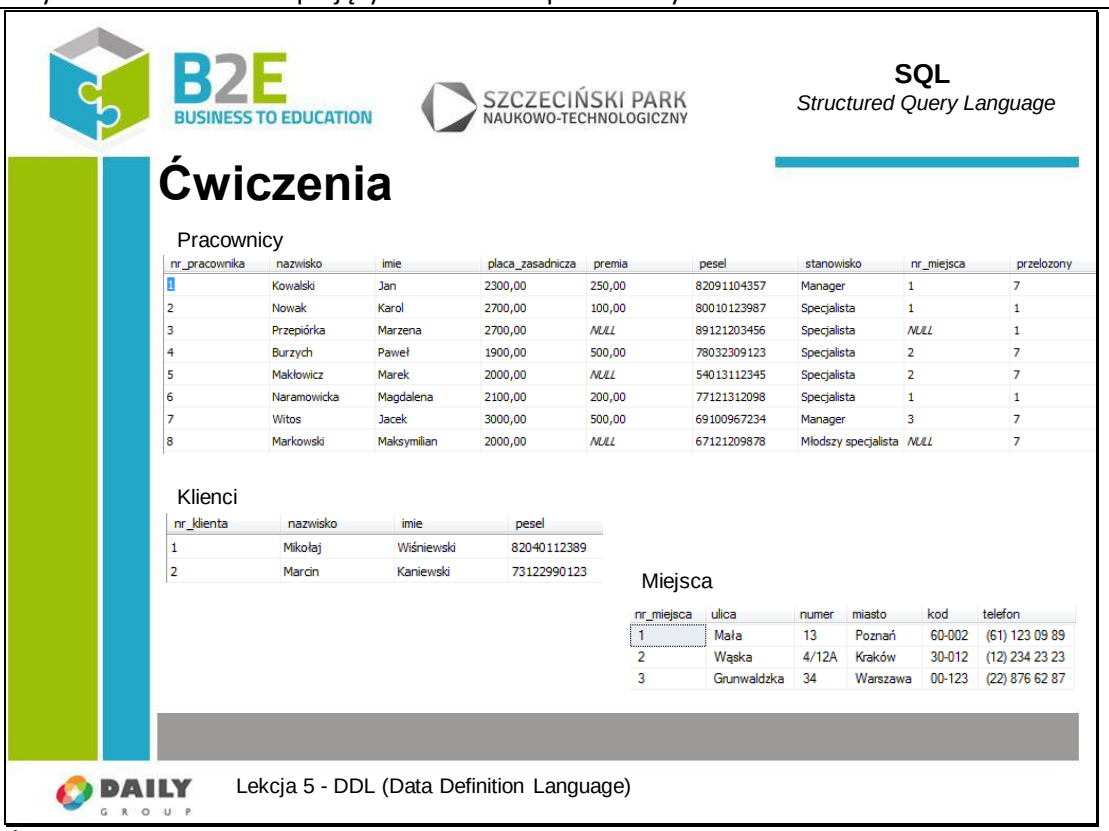
Slajd 24		Przykład ALTER TABLE modyfikujący kolumnę w istniejącej tabeli.
Slajd 25		SELECT INTO – szybka metoda kopiowania tabel.

Slajd 26

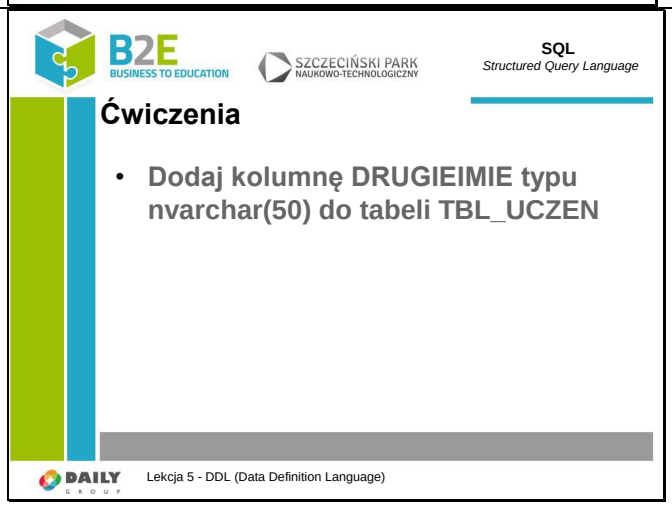


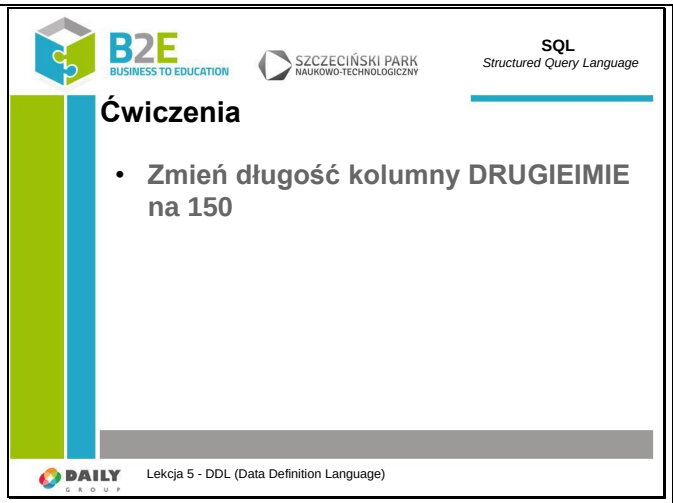
Przykład SELECT INTO kopiujący dane z tabeli pracownicy.

Slajd 27



Ćwiczenie 1:
Utwórz nową bazę danych o nazwie Testowa_Inicjały

	Ćwiczenie 2: Utwórz nowe tabele zgodnie ze slajdem Ćwiczenie 3: Usuń tabelę miejsca Ćwiczenie 4: Zmodyfikuj tabelę klienci – dodaj kolumnę wartość zamówienia (typ decimal z dwoma miejscami po przecinku) Ćwiczenie 5: Zmodyfikuj kolumnę wartość zamówienia – zmień typ na money z domyślną wartością 0 Ćwiczenie 6: Skopiuj tabelę klienci do tabeli kontrahenci Ćwiczenie 7: Usuń kolumnę wartość zamówienia w tabeli kontrahenci Ćwiczenie 8: Usuń bazę testowa	
Slajd 28		SELECT * INTO TBL_UCZEN_BCKP FROM TBL_UCZEN
Slajd 29		ALTER TABLE TBL_UCZEN ADD COLUMN DRUGIEIMIE nvarchar(50);

Slajd 30		ALTER TABLE TBL_UCZEN ALTER COLUMN DRUGIEIMIE nvarchar(150);
Slajd 31		CREATE DATABASE SZKOLA
Slajd 32		DROP DATABASE SZKOLA

Slajd 33		Tabela podsumowująca poznane wyrażenia

9.5.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 27-32. Uczniowie przyswoją wiedzę dotyczącą tworzenia i modyfikacji tabel w bazie danych.

9.5.4 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą wiedzieć jak tworzy się i modyfikuje tabele w bazie danych.

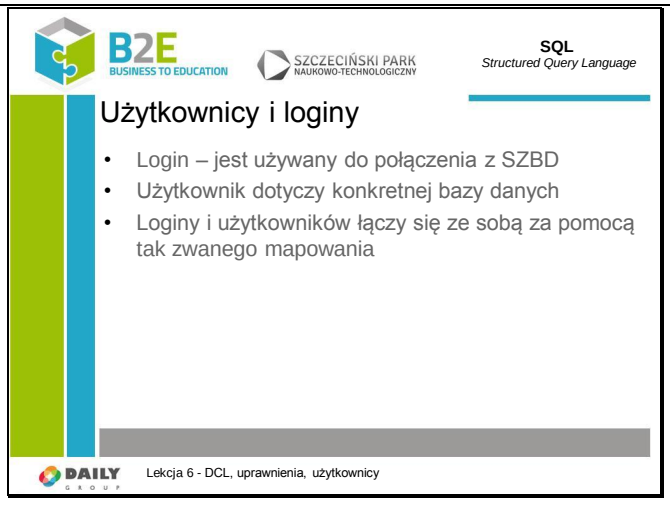
9.6 Lekcja 6 - DCL, uprawnienia, użytkownicy

9.6.1 Cel lekcji

Celem lekcji jest zapoznanie uczniów z pojęciem użytkownika i loginu w bazie danych. Uczniowie dowiedzą się jak przyznawać i odbierać uprawnienia do obiektów bazodanowych użytkownikom i grupom użytkowników.

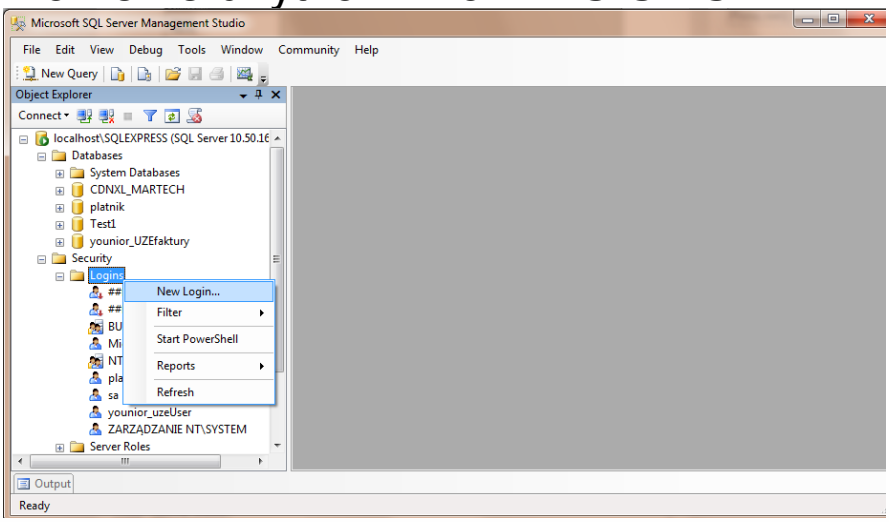
9.6.2 Treść - slajdy z opisem

Slajd 1		DCL (Data Control Language) – to język wykorzystywany do nadawania uprawnień użytkownikom i grupom do obiektów baz danych. W tej lekcji przyjrzymy się uprawnieniom, użytkownikom i nauczymy się tworzyć polecenia SQL do sterowania użytkownikami i uprawnieniami.

Slajd 2		Przypomnienie materiału z poprzedniej lekcji
Slajd 3		Definicja DCL
Slajd 4		Definicja loginu i użytkownika. Login to obiekt funkcjonujący w kontekście SZDB, a użytkownik w kontekście bazy danych. Na kolejnych slajdach przedstawiono tworzenie loginu i użytkownika do niego za pomocą MS SQL Management Studio oraz za pomocą TSQL

Slajd 5

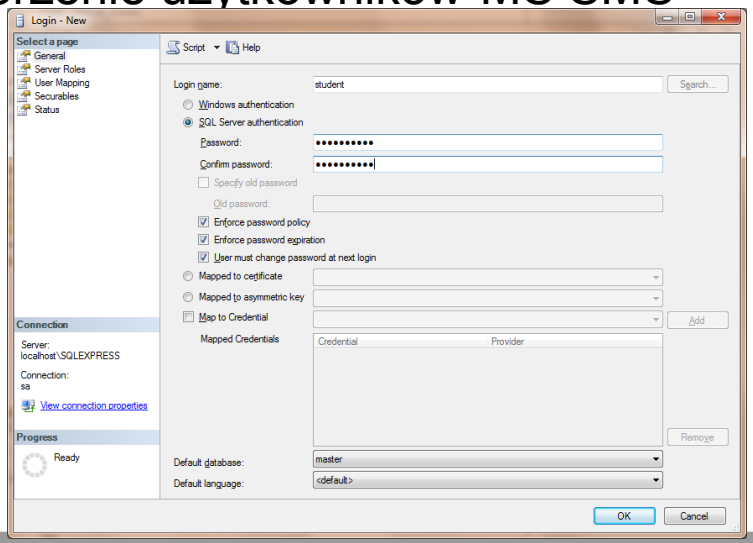
Tworzenie użytkowników MS SMS



Sekcja z loginami znajduje się w drzewie SZBD w sekcji Security -> Logins

Slajd 6

Tworzenie użytkowników MS SMS



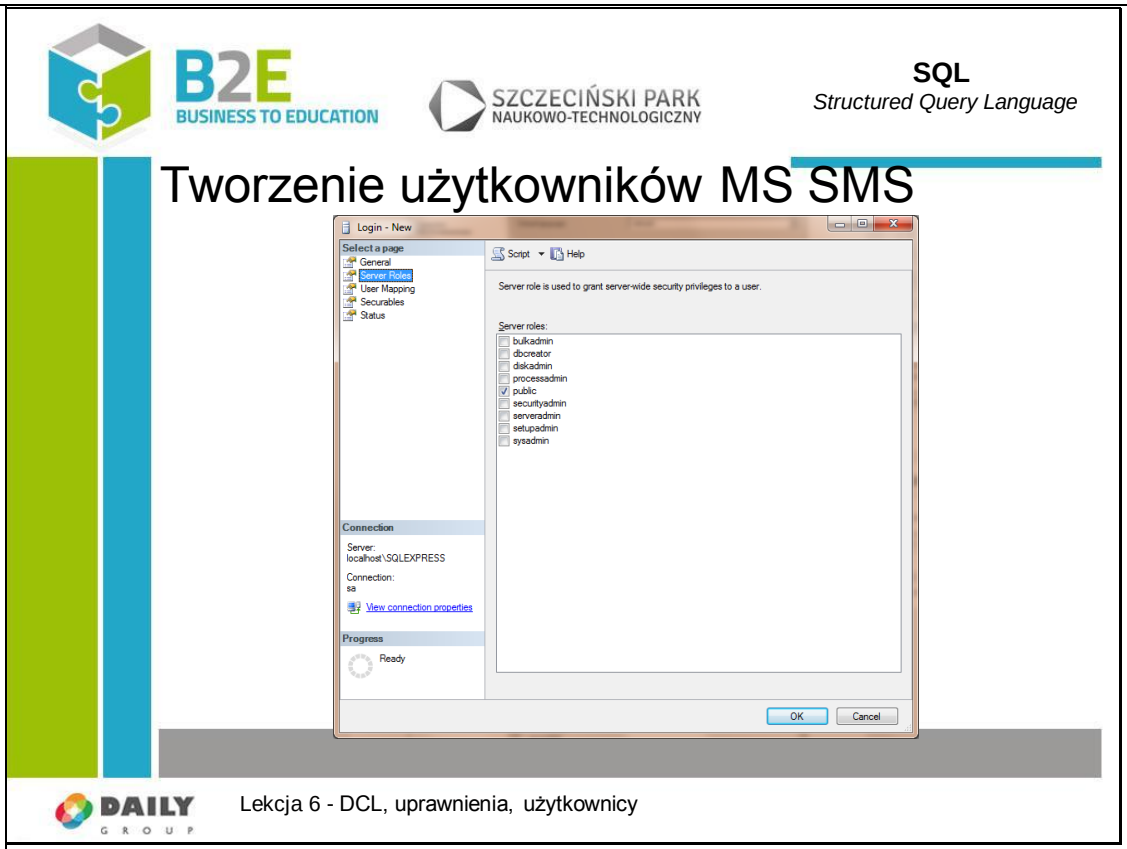
Po wybraniu New login pojawia nam się okno. W zakładce General wprowadzamy nazwę,

	wybieramy tryb logowania. Możliwa jest autentykacja Windows lub SQL. W przypadku Windows logować się z systemu windows, w którym użytkownik jest zalogowany. Ta opcja kierowana jest głównie do użytkowników domenowych. Jeśli chcemy umożliwić logowanie się użytkownikom z różnych systemów operacyjnych wybieramy opcję SQL Server authentication. Sekcje Default Database oraz Default language są widoczne zarówno dla loginu z autoryzacją SQL jak i Windows. W Default Database określamy domyślną bazę danych dla loginu.
Slajd 7	 W przypadku wybrania autoryzacji SQL, należy wprowadzić hasło dla loginu oraz poniżej wybrać jedną lub kilka z trzech opcji: - Enforce password Policy – hasło nie może zawierać w sobie części nazwy loginu i nie może być krótsze niż 7 znaków oraz powinno zawierać cyfry, duże i małe litery oraz znaki nie alfanumeryczne. Informacje pobierane są z polityki grupowej. - Enforce password expiration – hasło wygasa po przekroczeniu wartości określonej w polityce grupowej. - User must change password at next logon – wymaga zmiany hasła użytkownika przy kolejnym logowaniu.

Slajd 8

Na stronie **Server Roles** wybieramy role serwerowe, przypisane dla tego loginu:
bulkadmin – zezwala na operację masowego wstawiania danych (BULK INSERT),
dbcreator – zezwala na tworzenie, usuwanie, modyfikację bazy danych oraz dodawanie do niej nowych członków (CREATE DATABASE),
diskadmin – zezwala na zarządzanie plikami .mdf i .ldf (ALTER),
processadmin – zezwala na kontrolę procesów (ALTER ANY CONNECTION oraz ALTER SERVER STATE),
securityadmin – zezwala na zarządzanie loginami i uprawnieniami (ALTER ANY LOGIN),
serveradmin – zezwala na konfigurację całego serwera (ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN),
setupadmin –zezwala na zarządzanie serwerami połączonymi (ALTER ANY LINKED SERVER),
sysadmin – zezwala na pełną kontrolę nad bazami danych (CONTROL SERVER with GRANT).

Slajd 9

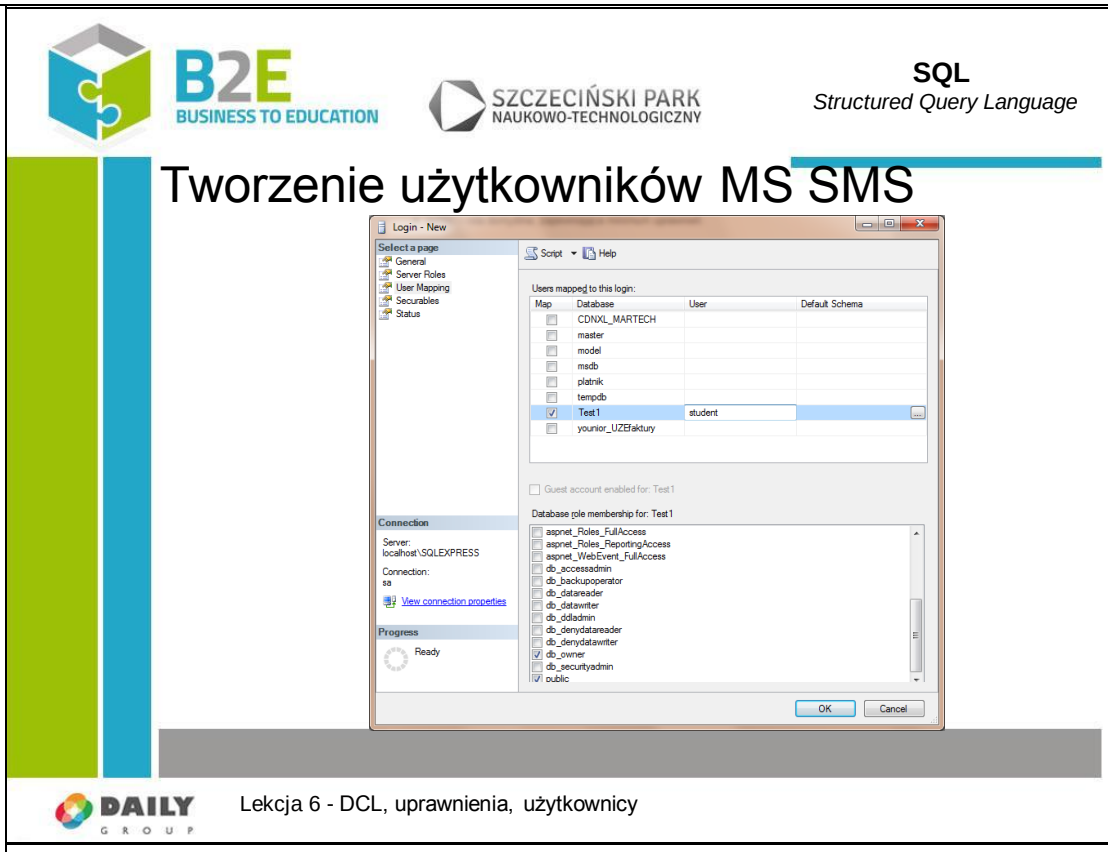


Na stronie **Server Roles** wybieramy role serwerowe, przypisane dla tego loginu:
bulkadmin – zezwala na operację masowego wstawiania danych (BULK INSERT),
dbcreator – zezwala na tworzenie, usuwanie, modyfikację bazy danych oraz dodawanie do niej nowych członków (CREATE DATABASE),
diskadmin – zezwala na zarządzanie plikami .mdf i .ldf (ALTER),
processadmin – zezwala na kontrolę procesów (ALTER ANY CONNECTION oraz ALTER SERVER STATE),
securityadmin – zezwala na zarządzanie loginami i uprawnieniami (ALTER ANY LOGIN),
serveradmin – zezwala na konfigurację całego serwera (ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN),
setupadmin –zezwala na zarządzanie serwerami połączonymi (ALTER ANY LINKED SERVER),
sysadmin – zezwala na pełną kontrolę nad bazami danych (CONTROL SERVER with GRANT).

Slajd 10

Na stronie **User Mapping** zaznaczamy, do której bazy użytkownik będzie posiadał uprawnienia (**Users mapped to this login**), a następnie w sekcji **Database role membership for** : nazwa_bazy zaznaczamy role, które chcemy nadać:
db_accessadmin – zezwala na dodawanie i usuwanie kont,
db_backupoperator – zezwala na wykonywanie kopii zapasowych,
db_datareader – zezwala na odczyt baz danych,
db_datawriter – zezwala na zapisywanie i modyfikację baz danych,
db_ddladmin – zezwala na modyfikację i usuwanie obiektów baz danych,
db_denydatareader – nie zezwala na odczyt baz danych,
db_dentydatawriter – nie zezwala na zapisywanie i modyfikację baz danych,
db_owner – zezwala na pełną kontrolę nad bazą danych,
db_securityadmin – zezwala na zarządzanie uprawnieniami oraz rolami baz danych,
public – rola domyślna, zapewniająca minimum uprawnień.

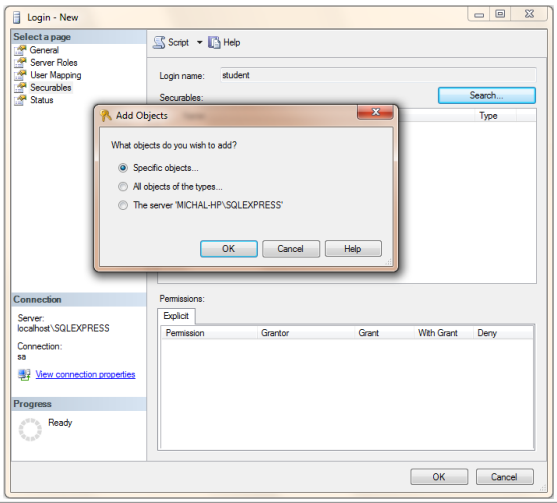
Slajd 11



Na stronie **User Mapping** zaznaczamy, do której bazy użytkownik będzie posiadał uprawnienia (**Users mapped to this login**), a następnie w sekcji **Database role membership for** : nazwa_bazy zaznaczamy role, które chcemy nadać:
db_accessadmin – zezwala na dodawanie i usuwanie kont,
db_backupoperator – zezwala na wykonywanie kopii zapasowych,
db_datareader – zezwala na odczyt baz danych,
db_datawriter – zezwala na zapisywanie i modyfikację baz danych,
db_ddladmin – zezwala na modyfikację i usuwanie obiektów baz danych,
db_denydatareader – nie zezwala na odczyt baz danych,
db_dentydatawriter – nie zezwala na zapisywanie i modyfikację baz danych,
db_owner – zezwala na pełną kontrolę nad bazą danych,
db_securityadmin – zezwala na zarządzanie uprawnieniami oraz rolami baz danych,
public – rola domyślna, zapewniająca minimum uprawnień.

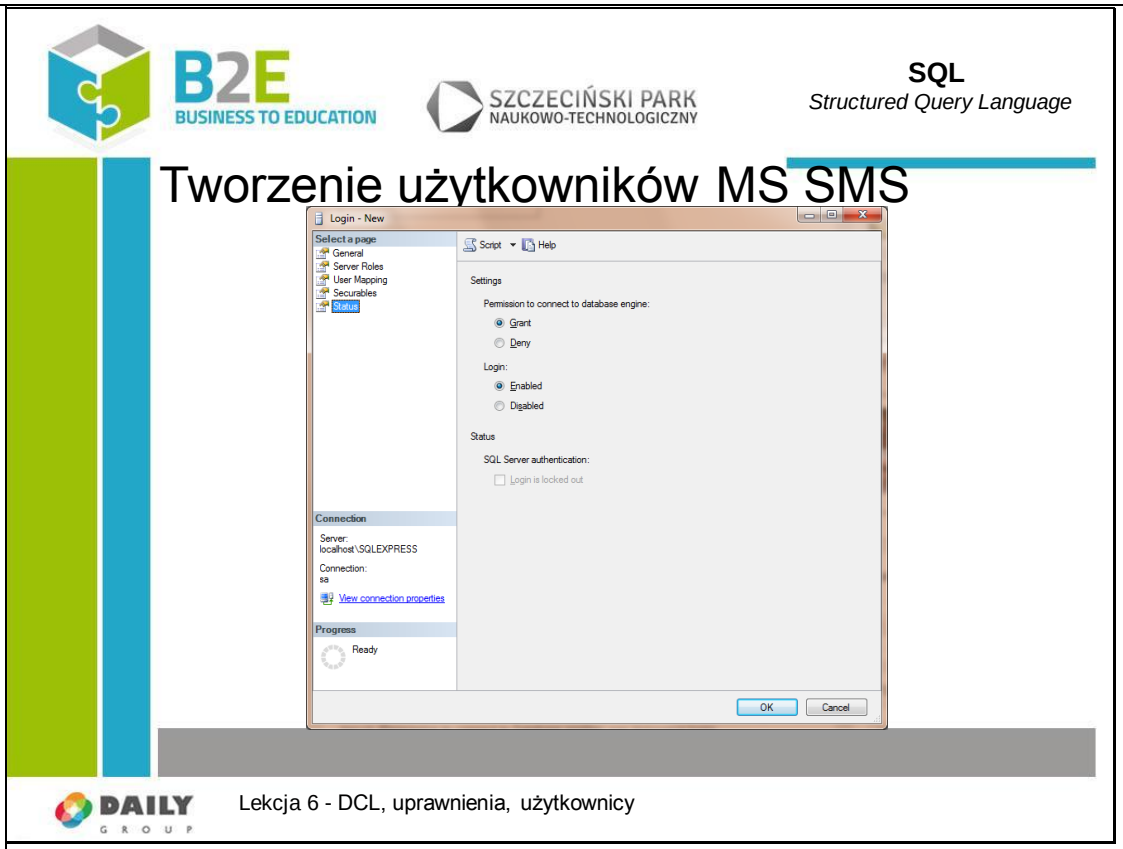
Slajd 12

Tworzenie użytkowników MS SMS



Strona **Securables** służy do przypisywania uprawnień do obiektów zabezpieczanych dla tego loginu.

Slajd 13

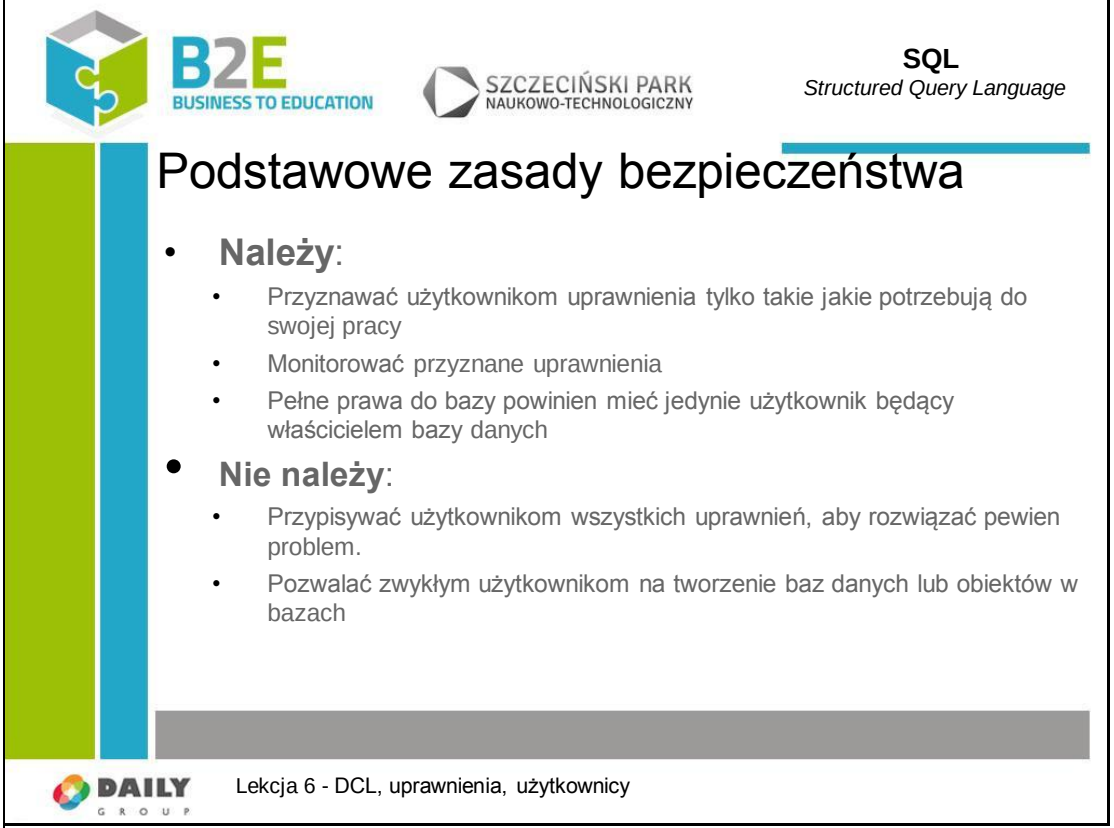


Strona **Status** służy do nadawania lub odejmowania uprawnień dla loginu do łączenia się z bazą danych (**Permission to connect to Database engine**) oraz blokowania konta.

Slajd 14

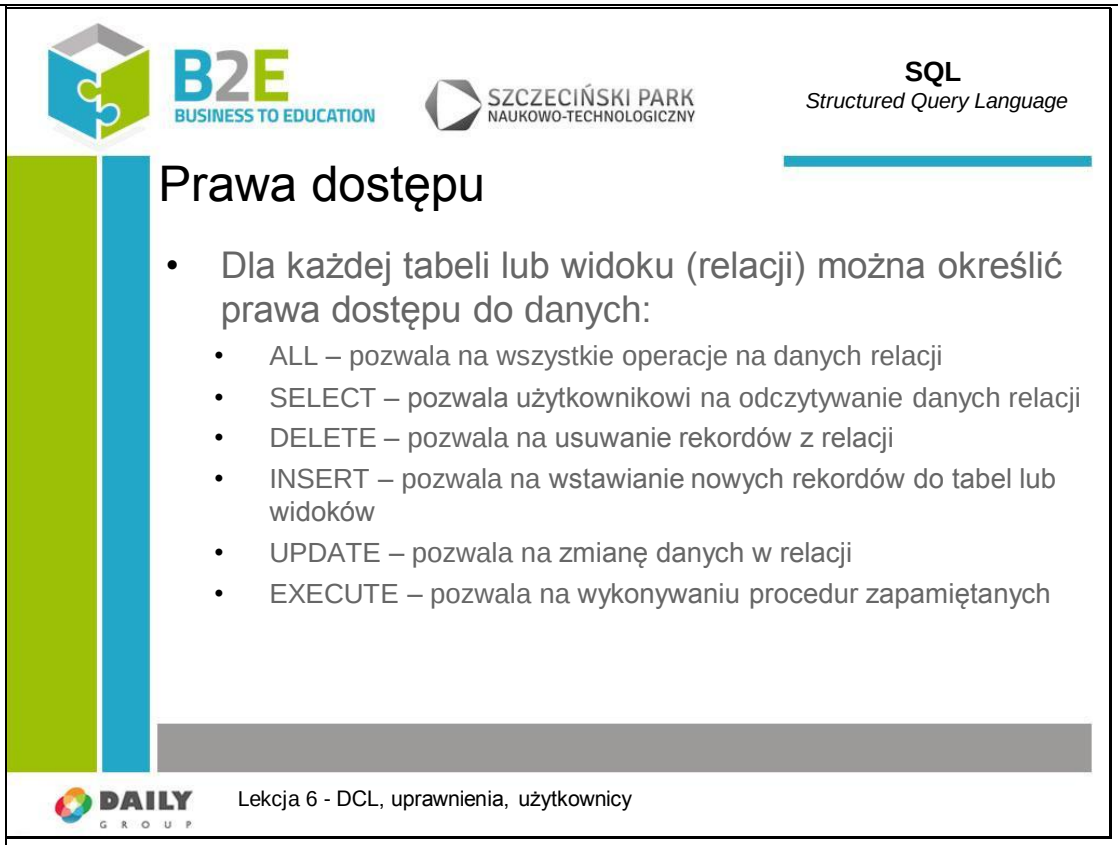
Tworzenie nowego loginu za pomocą TSQL i nadawanie odpowiednich uprawnień.

Slajd 15



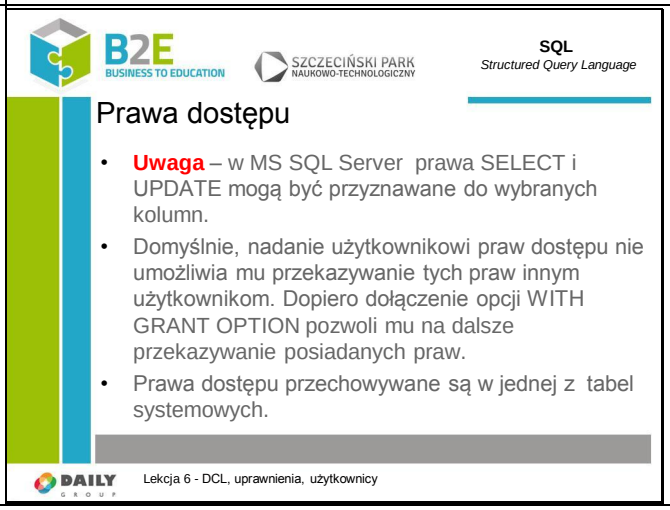
Najważniejsze zasady, o których należy pamiętać podczas przydzielania uprawnień

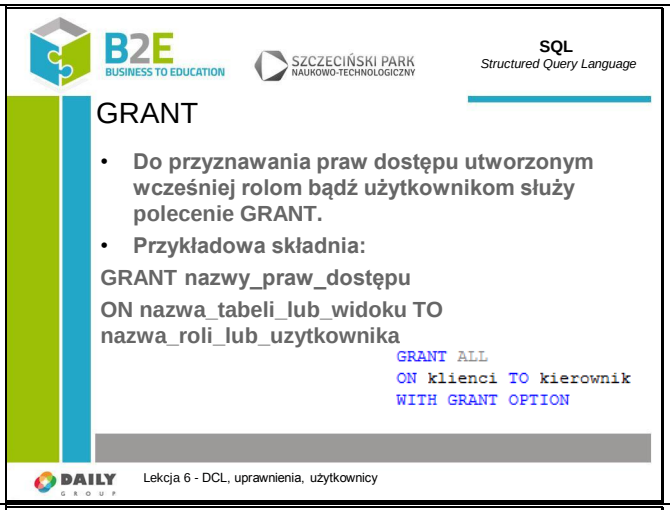
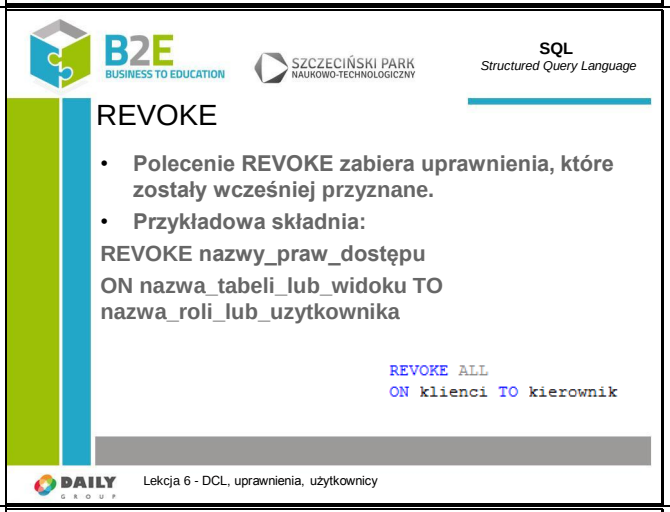
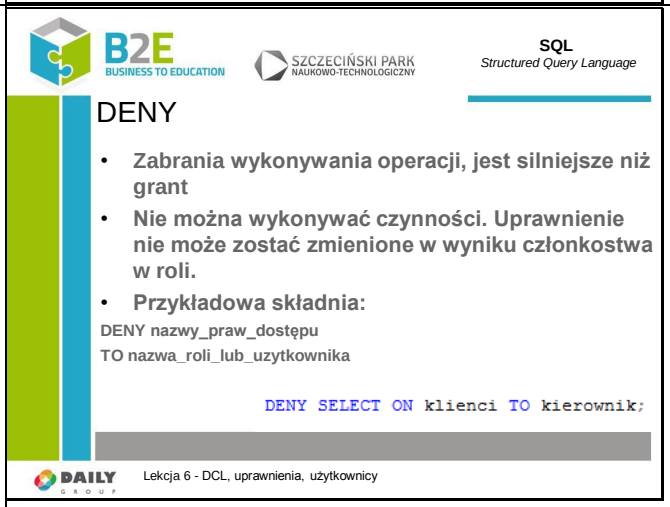
Slajd 16

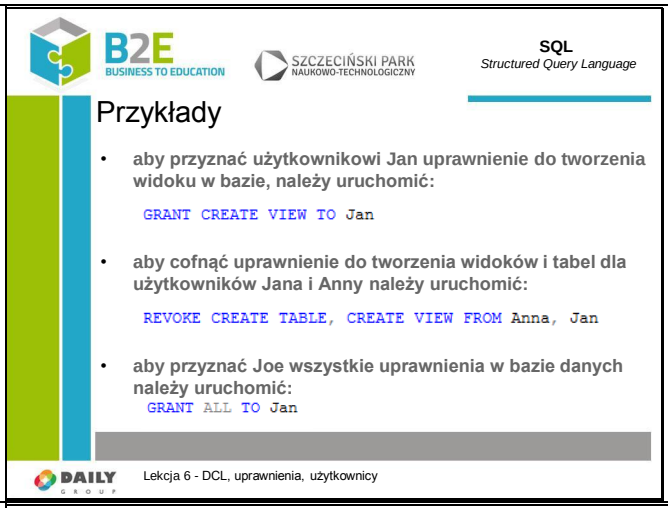


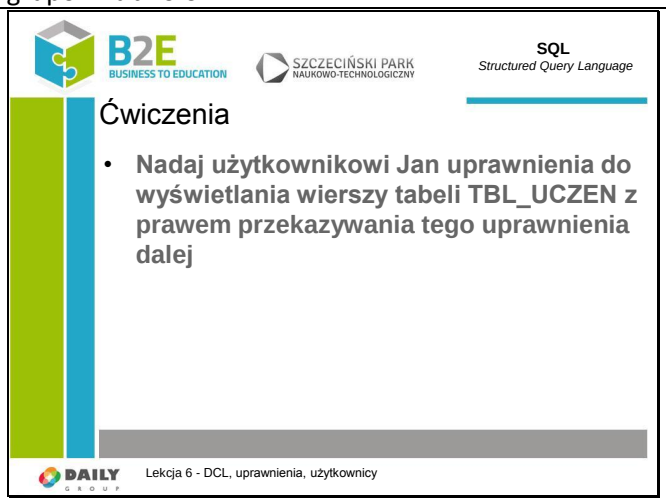
Na slajdzie znajduje się zestawienie wszystkich praw dostępu, jakie można zastosować dla tabel i widoków.

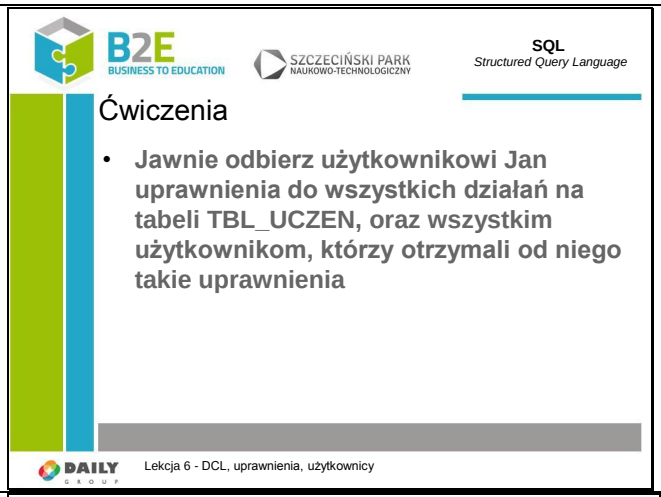
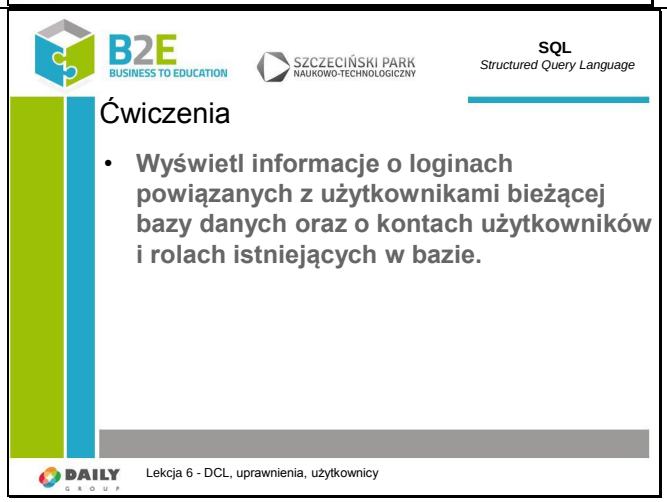
Slajd 17



Slajd 18		Definicja i przykład użycia polecenia GRANT
Slajd 19		Polecenie REVOKE – służące od odbierania praw do obiektów
Slajd 20		Inaczej niż w przypadku polecenia REVOKE, polecenie DENY bezpośrednio zabiera uprawnienie polecenia. Przykładowo, jeżeli Jan jest członkiem roli bazy danych, a rola ta ma uprawnienie CREATE TABLE, Jan może również tworzyć tabele. Jednak, jeżeli Jan powinien mieć zabronione tworzenie tabel, mimo tego, że jako członek roli posiada to uprawnienie, można zabronić Janowi wykonywania tego polecenia. Dlatego, Jan nie będzie mógł uruchomić wyrażenia CREATE TABLE, pomimo tego, że normalnie rola przydzieliła mu to prawo.

Slajd 21		Przegląd kilku przykładów jest najlepszym sposobem na zrozumienie działania omówionych poleceń.
Slajd 22	 Ćwiczenie 1: Utwórz nową bazę danych o nazwie Testowa_Inicjały Ćwiczenie 2: Utwórz użytkownika Nazwisko_studenta1 z poziomu MS SMS z domyślną bazą Testowa_Inicjały Ćwiczenie 3: Utwórz użytkownika Nazwisko_studenta2 z poziomu TSQL z domyślną bazą Testowa_Inicjały Ćwiczenie 4: Wykorzystaj procedury: sp_password sp_defaultdb sp_defaultlanguage	

	sp_helplogins sp_droplogin dla użytkownika Nazwisko_studenta2 Ćwiczenie 5: Wykorzystując procedurę *sp_changedbowner* zmień właściciela bazy danych Testowa_Inicjały na Nazwisko_studenta1 Ćwiczenie 6: wykorzystując polecenia GRANT,REVOKE i DENY ustaw wybrane 3 uprawnienia (CREATE DATABASE, CREATE TABLE, CREATE PROCEDURE, CREATE DEFAULT, CREATE RULE, CREATE VIEW, CREATE FUNCTION, BACKUP DATABASE I BACKUP LOG) wybranym użytkownikom lub rolom (w razie potrzeby utworzyć dodatkowych użytkowników lub role) Ćwiczenie 7: wykorzystując polecenia GRANT,REVOKE i DENY ustaw wybrane uprawnienia (SELECT, INSERT, UPDATE, DELETE, EXECUTE, REFERENCES) dla wybranych obiektów bazy *Test* poszczególnym użytkownikom, grupom lub rolom Ćwiczenie 8: Zobrazuj na przykładach skutki przyznania lub zabronienia poszczególnym użytkownikom, grupom lub rolom	
Slajd 23		GRANT SELECT ON TBL_UCZEN TO Jan WITH GRANT OPTION;
Slajd 24		GRANT ALL ON TBL_UCZEN TO Jan;

Slajd 25		REVOKE ALL ON TBL_UCZEN TO Jan;
Slajd 26		DENY ALL ON TBL_UCZEN TO Jan CASCADE;
Slajd 27		EXEC sp_helpuser

Slajd 28		Tabela podsumowująca poznane wyrażenia

9.6.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 22-27. Uczniowie utworzą nową bazę danych, użytkowników oraz tabela, a następnie dokonają szeregu modyfikacji uprawnień tych obiektów.

9.6.4 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie nauczą się świadomie przydzielać prawa do obiektów bazodanowych, poznają najlepsze praktyki, a następnie w czasie ćwiczeń praktycznie wykorzystają zdobytą wiedzę.

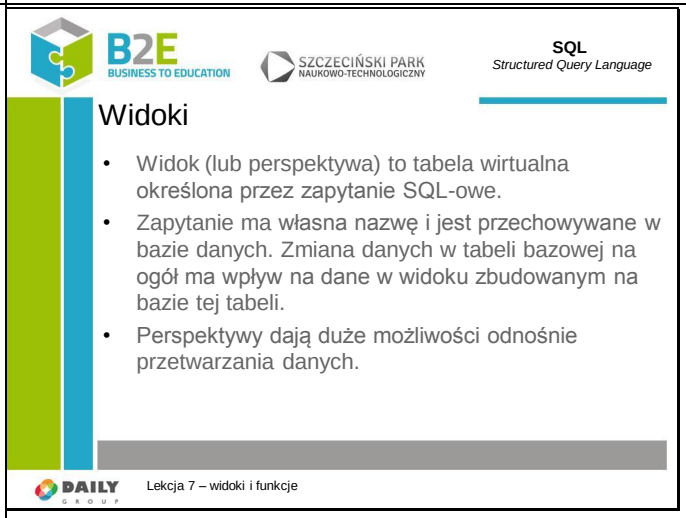
9.7 Lekcja 7 - Widoki i Funkcje

9.7.1 Cel lekcji

Celem lekcji jest przedstawienie dwóch typów obiektów w bazie danych: widoków oraz funkcji. Uczniowie po odbyciu lekcji powinni być w stanie samodzielnie zaprogramować proste widoki i funkcje użytkownika.

9.7.2 Treść - slajdy z opisem

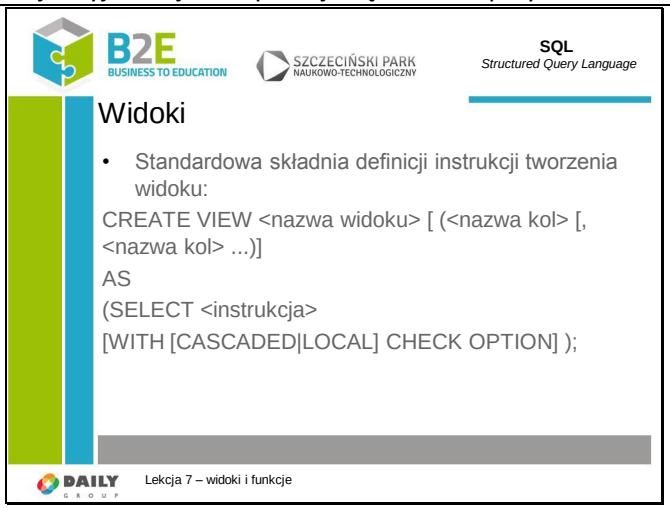
Slajd 1		W tej lekcji poznamy inne niż tabele obiekty bazy danych: widoki oraz funkcje. Widoki umożliwiają „zapisanie” skomplikowanych zapytań do późniejszego – szybkiego wykorzystania. Funkcje to obiekty, które już poznaliśmy. Na poprzednich lekcjach używaliśmy funkcji systemowych MIN, MAX, SUM, AVG. SQL Serwer umożliwia definicję własnych funkcji użytkownika – w tej lekcji dowiemy się w jaki sposób.

Slajd 2		Przypomnienie materiału z poprzedniej lekcji.
Slajd 3		Widoki to tabele wirtualne. Wykorzystywane są w sytuacji, gdy np. mamy tabelę posiadającą 10 kolumn, a na potrzebę aplikacji chcemy prezentować tylko część z nich.

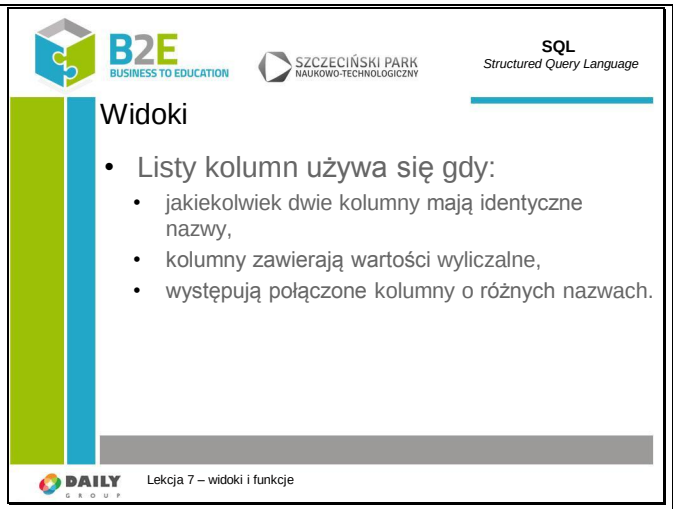
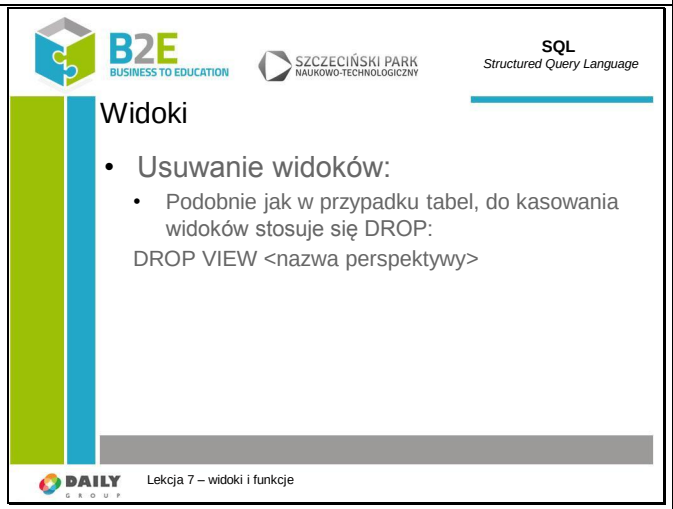
Slajd 4

Slajd wyjaśnia jakie operacje są możliwe przy tworzeniu widoków.

Slajd 5



Definicja widoku jest podobna do wszystkich definicji obiektów w bazie danych. Create, rodzaj obiektu, nazwy kolumn, słówko AS (jako) i SELECT, który chcemy zapisać w widoku.

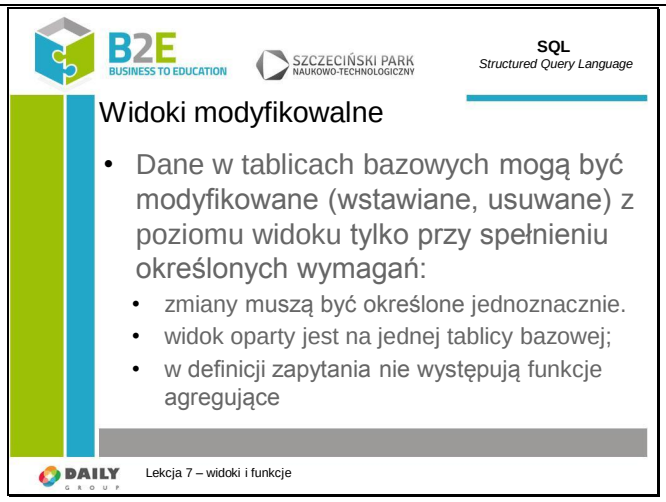
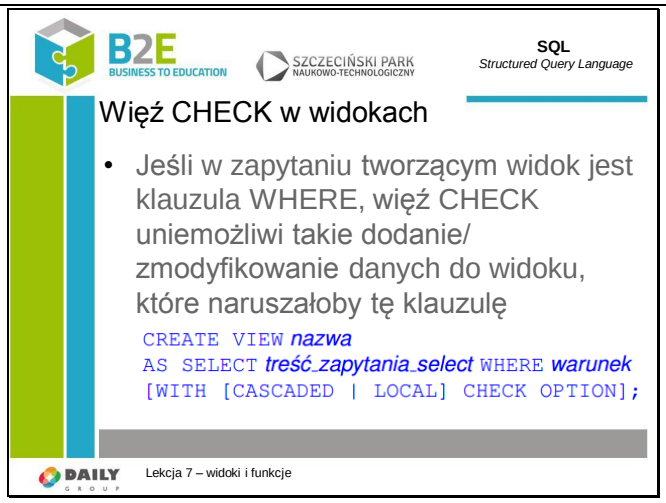
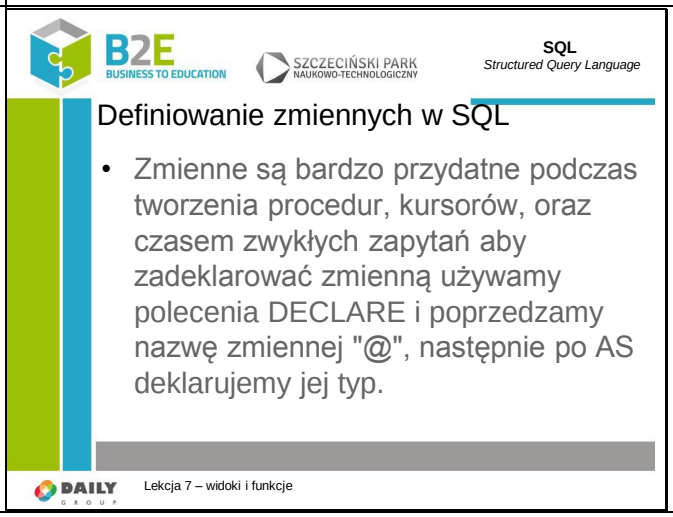
Slajd 6		W widokach możemy zapisać SELECT * FROM table lub SELECT col1, col2, … FROM table. Slajd wyjaśnia kiedy unikać gwiazdki.
Slajd 7		Usuwanie widoków – tak jak wszystkich obiektów w bazie danych – dokonywane jest instrukcją DROP
Slajd 8		Przykład prezentuje widok, który łączy dwie tabele i zawęża zwracane wartości tylko do autorów pochodzących z miasta Oakland.

Slajd 9

Przykład prezentuje widok, który dokonuje obliczeń na kolumnach tabeli titles.

Slajd 10

Przykład prezentuje widok, który łączy dwie tabele i zawęża zwracane wartości tylko do autorów pochodzących z miasta Oakland.

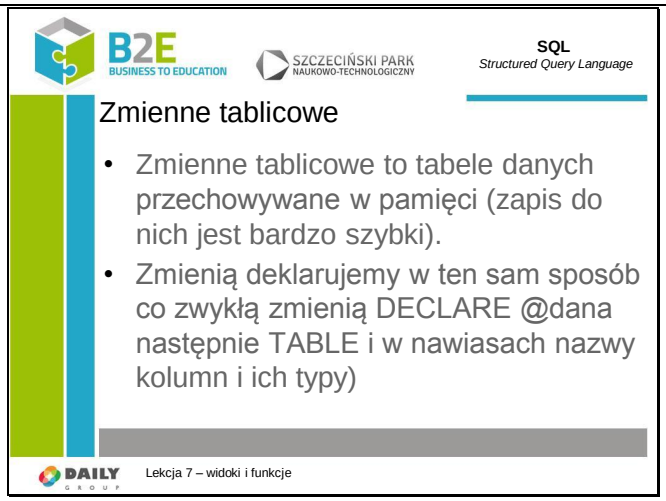
Slajd 11		Dane w tablicach bazowych mogą być modyfikowane (wstawiane, usuwane) z poziomu widoku tylko przy spełnieniu określonych wymagań: ● zmiany muszą być określone jednoznacznie. ● perspektywa oparta jest na jednej tablicy bazowej; odniesienia są tylko do kolumn tej tablicy zawiera tylko jedno zapytanie (bez UNION, EXCEPT, INTERSECT), ● w definicji zapytania nie występują funkcje agregujące (nie ma GROUP BY, HAVING, DISTINCT).
Slajd 12		CASCADED i LOCAL mają znaczenie jeśli widok tworzony jest na podstawie innych widoków. Jeżeli wybierzemy CASCADED, klauzule WHERE “podwidoków” też są sprawdzane, nawet jeśli nie nałożono na nie jawnie więzu (CONSTRAINT – wyjaśniony w kolejnych lekcjach) CHECK.
Slajd 13		Typy zmiennych mogą być dokładnie takie same jak typy tabel, które omawialiśmy na poprzednich lekcjach

Slajd 14

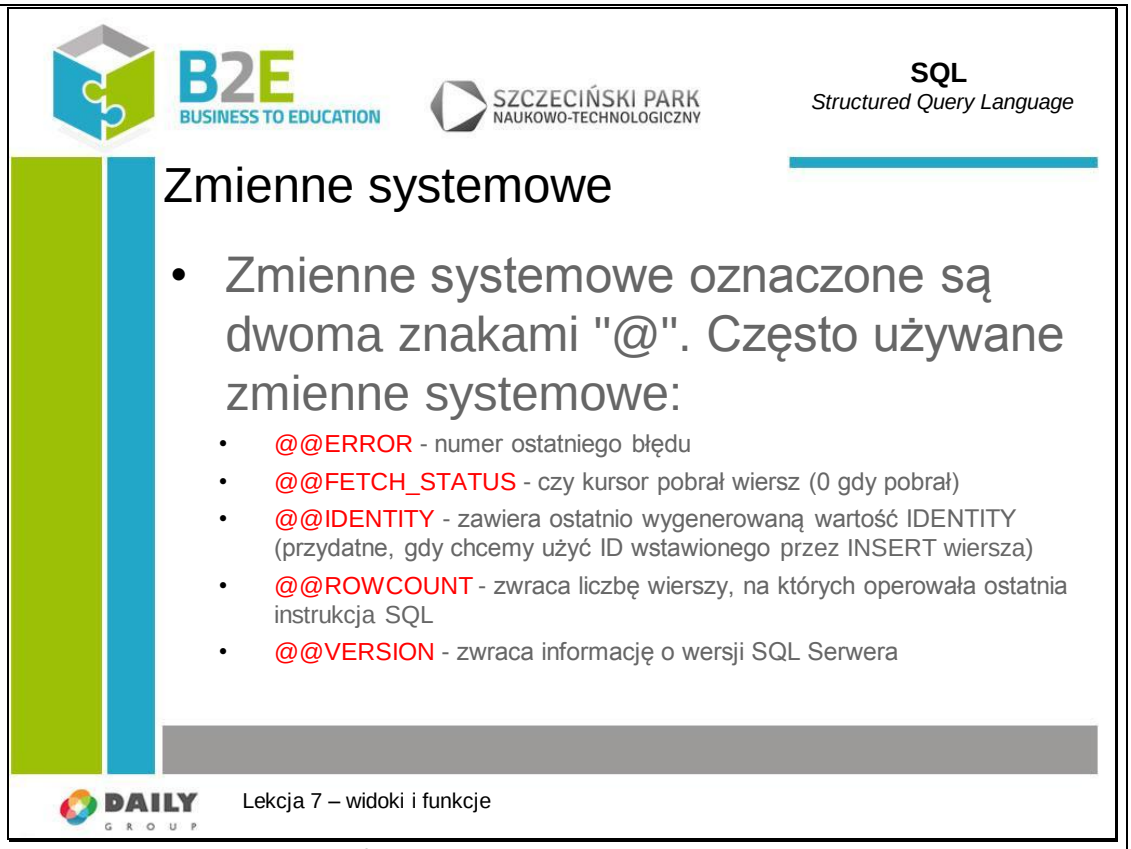
Definicja zmiennej @dana, przypisanie do niej wartości 1, a następnie wyświetlenie wartości za pomocą znanego nam polecenia SELECT. Zmiennej możemy użyć nie tylko w selekcie ale również w warunkach.

Slajd 15

Pamiętacie naszą tabelę pracownicy? Zdefiniowanie warunku za pomocą zmiennej @dana

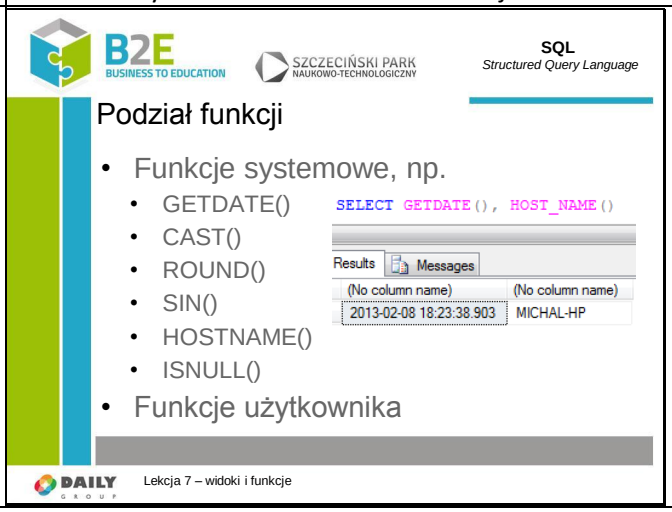
Slajd 16		Definicja zmiennych tablicowych
Slajd 17		Na początku deklarujemy zmienną tablicową @dana, jej kolumny i ich typy, przypadku zmiennej skalarnej nie możemy użyć SET ale możemy umieścić dane w tabeli przy użyciu INSERT INTO wyniku z poniższego zapytania. Zwróćcie szczególną uwagę na zapis ---kolejny selekt--- jest to działanie celowe, aby zademonstrować jak wprowadza się komentarze w t-sql "--" oznacza komentarz lub /* wykomentowany tekst */. Ostatni SELECT powoduje wyświetlenie zawartości tabeli @dane. Uwaga po zakończeniu ostatniego select wartość zmiennej jest wymazywana.

Slajd 18

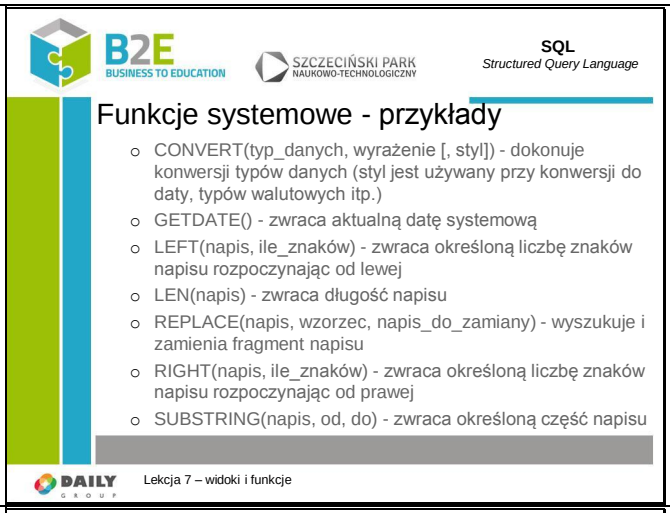
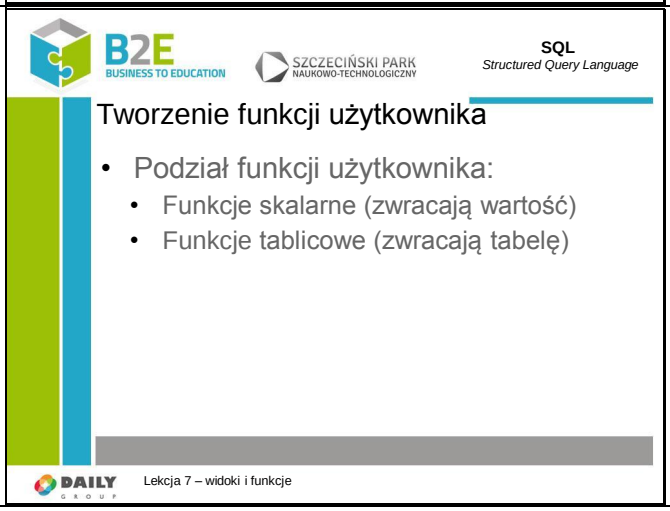


Zmienne systemowe – to zdefiniowane już w SQL serwer dostępne w każdej bazie danych.

Slajd 19



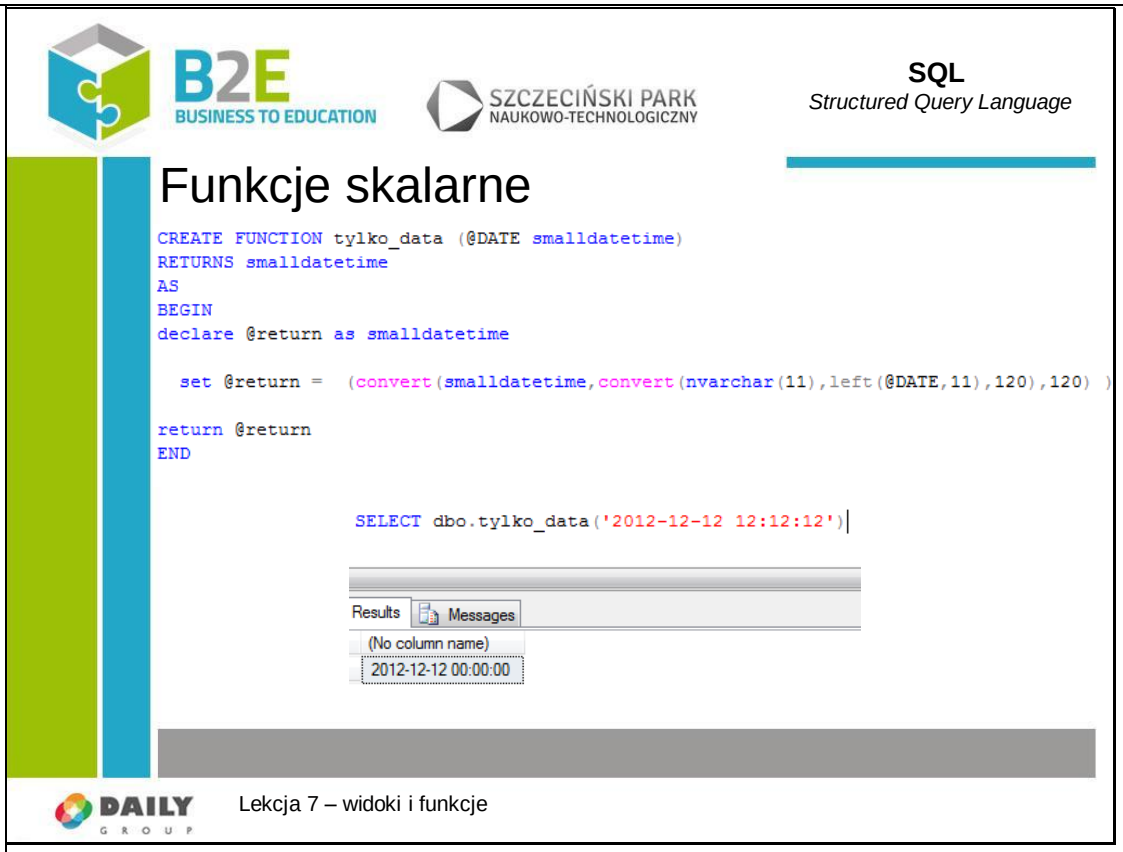
Funkcje dzielimy na funkcje systemowe – dostępne w każdej instalacji MSSQL oraz funkcje użytkownika, które zostały napisane i zachowane w konkretnej bazie danych.

Slajd 20		Przykłady najczęściej wykorzystywanych funkcji systemowych.
Slajd 21		Zajmijmy się teraz tworzeniem funkcji w t-sql. Możemy tworzyć funkcje użytkownika które zwracają skalarne lub tablicowe wartości. Funkcji możemy używać w widokach, w innych funkcja, w aplikacjach oraz procedurach.

Slajd 22

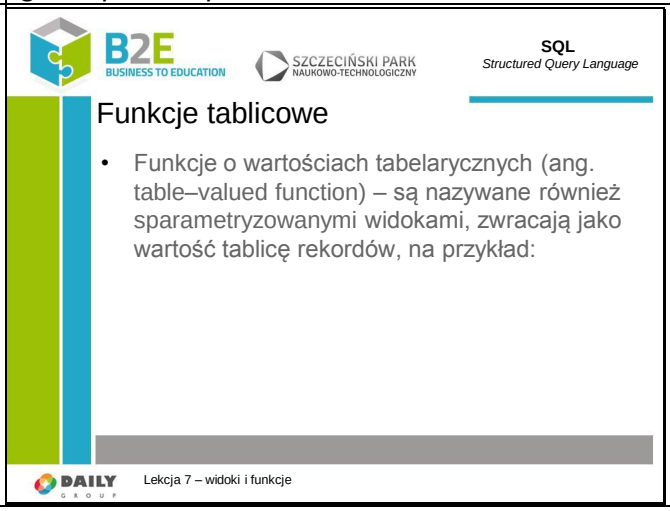
Ogólna składnia tworzenia funkcji wygląda następująco. Funkcja przyjmuje parametry (jeden lub wiele) oraz zwraca wartość (funkcja skalarna zwraca jedną wartość). Podczas definicji funkcji musimy określić ilość, typ i nazwy parametrów oraz typ zwracanej wartości.
Zamiast function body – wprowadza się treść funkcji.

Slajd 23



Powyższy przykład demonstruje w jaki sposób stworzyć funkcje tylko data, czyli datę bez godziny i minuty.

Slajd 24

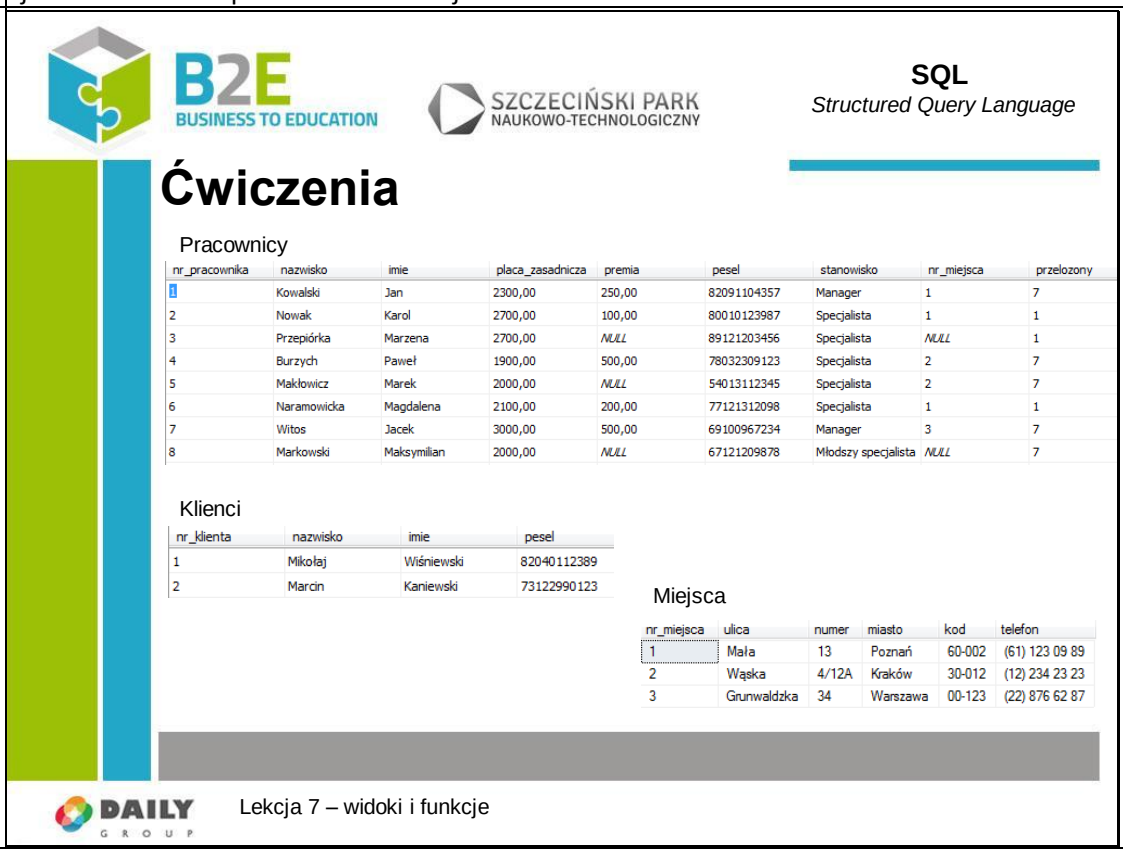


Definicja funkcji tablicowych, funkcje te często są nazywane widokami parametryzowalnymi.

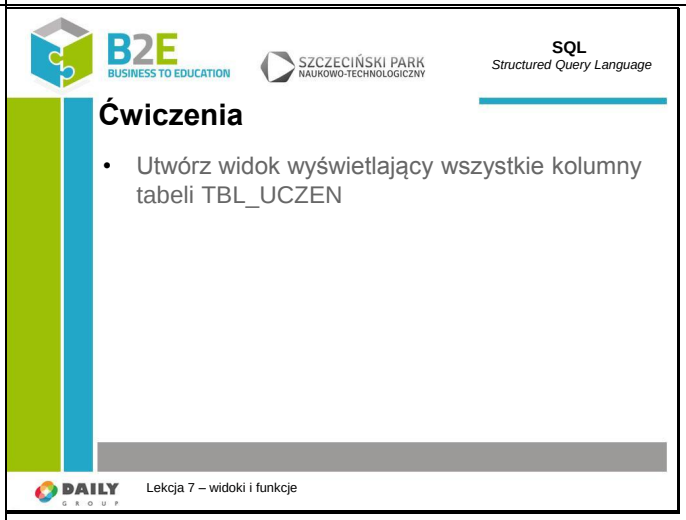
Slajd 25

Przykład demonstruje użycie funkcji w celu otrzymania zapytania z tabeli dane z wartością jako warunkiem i parametrem funkcji.

Slajd 26



	Ćwiczenie 1: Utwórz widok zwracający pracowników i miejsca pracy Ćwiczenie 2: Utwórz widok zwracający zatrudnione kobiety. Zaktualizuj pensje zasadniczą o 20% poprzez wykorzystanie UPDATE na widoku. Sprawdź czy dane zmieniły się w tabeli. Ćwiczenie 3: Utwórz zmienną tabelaryczną z takimi samymi kolumnami jak klienci. Wpisz kilka przykładowych wartości do zmiennej. Dodaj wartości ze zmiennej do tabeli klienci Ćwiczenie 4: Utwórz funkcję zwracającą średnią płacę, na podstawie lokalizacji (miejsca pracy) Ćwiczenie 5: Utwórz funkcję zwracającą premię na podstawie numeru pracownika Ćwiczenie 6: Utwórz funkcję tabelaryczną zwracającą imię, nazwisko i pesel pracownika na podstawie miejsca zatrudnienia Ćwiczenie 7: Utwórz funkcję zwracającą adres (w formie tabeli) na podstawie numeru pracownika Ćwiczenie 8: Zmień funkcję z ćwiczenia 7, tak aby zwracała wartość skalarną Ćwiczenie 9: Usuń utworzone widoki i funkcje	
Slajd 27		SELECT GETDATE()

Slajd 28		SELECT @@ERROR
Slajd 29		DECLARE @ILE int SET @ILE=5
Slajd 30		CREATE VIEW mojwidok AS SELECT * FROM TBL_UCZEN

Slajd 31		CREATE FUNCTION dbo.liczbaUczniow (@klasa nvarchar(50)) RETURNS int AS BEGIN DECLARE @ile int SELECT @ile = COUNT(*) FROM TBL_UCZEN WHERE KLASA=@klasa RETURN @ile END
Slajd 32		Tabela podsumowująca poznane wyrażenia

9.7.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 26-31, zadaniem uczniów jest utworzenie widoków i funkcji na podstawie zdobytej wiedzy.

9.7.4 Opis założonych osiągnięć ucznia

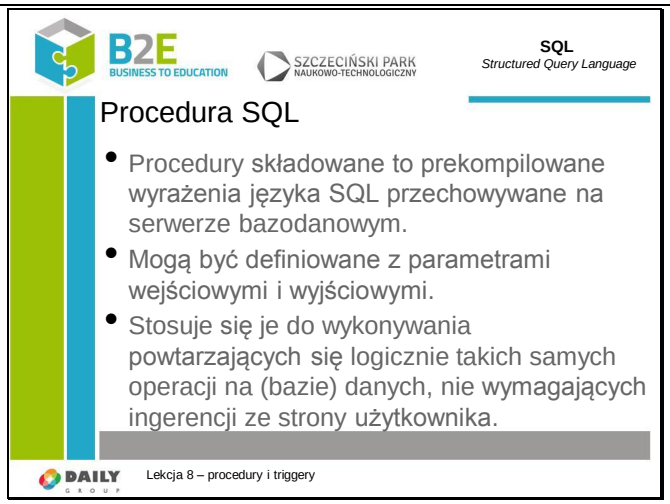
Po tej lekcji uczniowie będą potrafić samodzielnie zaprogramować widoki i funkcje w bazie danych.

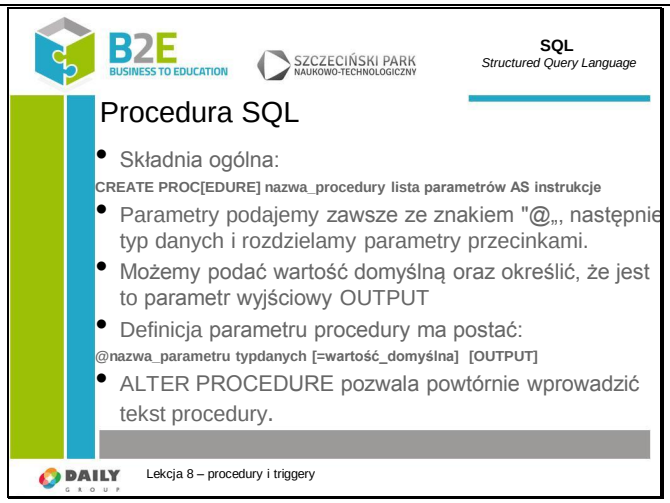
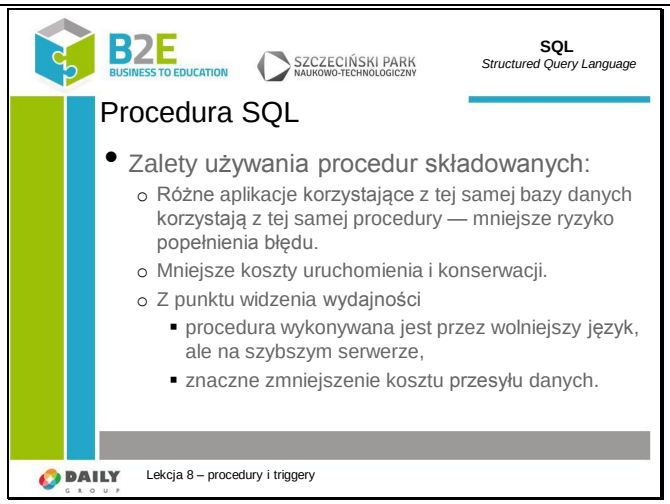
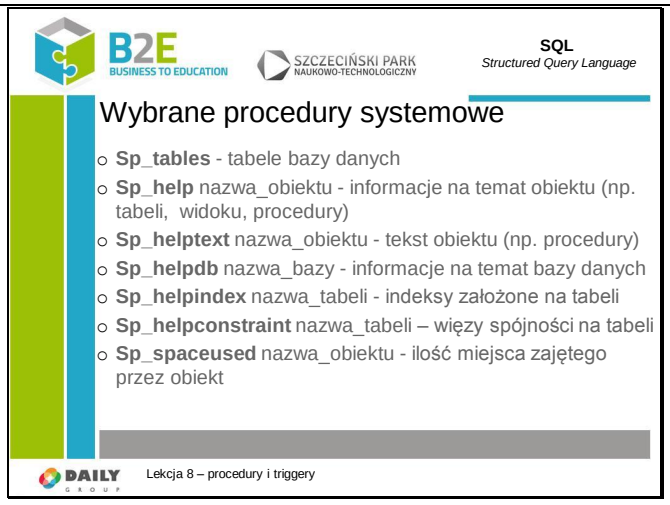
9.8 Lekcja 8 - Procedury Triggery

9.8.1 Cel lekcji

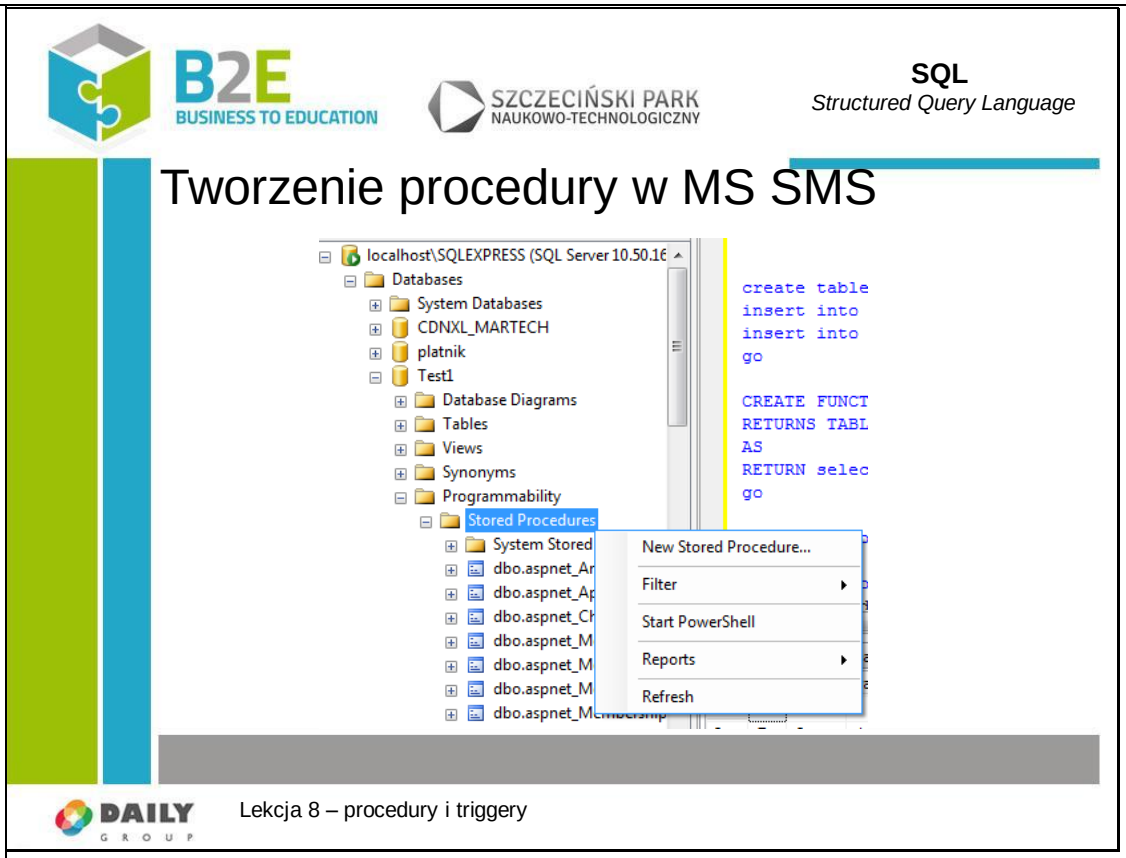
Celem lekcji prezentacja dwóch kolejnych typów obiektów baz danych: procedur oraz wyzwalaczy (triggerów). Szkoleniowiec zademonstruje kilka przykładów wykorzystania procedur i triggerów.

9.8.2 Treść - slajdy z opisem

Slajd 1		W tej lekcji poznamy kolejne dwa typy obiektów baz danych: procedury składowane i wyzwalacze(triggery).
Slajd 2		Podsumowanie poprzedniej lekcji.
Slajd 3		Procedura składowana, to nic innego jak specjalna struktura w bazie danych, która przypomina klasyczne programowanie. Każda z tworzonych przez nas procedur, może zawierać parametry i zwracać określone wartości, może również wykonywać pewne operacje.

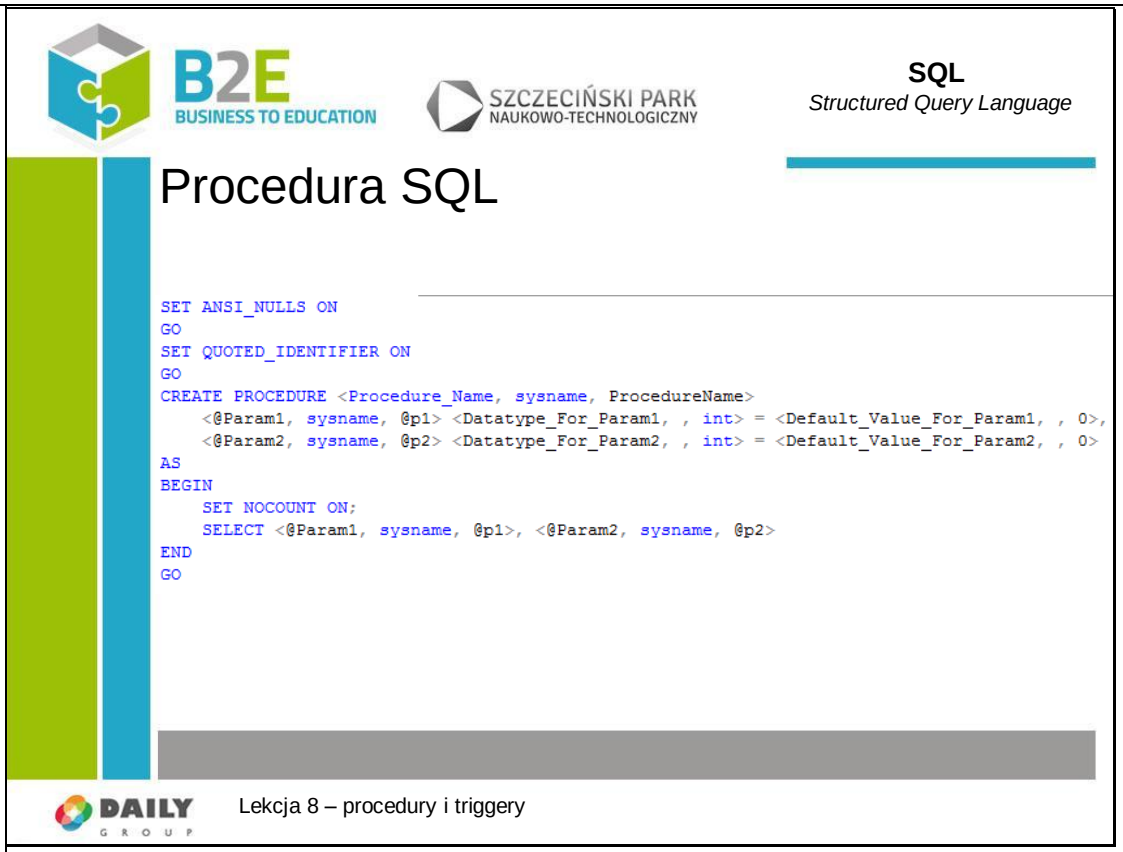
Slajd 4		Procedura składowana, to nic innego jak specjalna struktura w bazie danych, która przypomina klasyczne programowanie. Każda z tworzonych przez nas procedur, może zawierać parametry i zwracać określone wartości, może również wykonywać pewne operacje.
Slajd 5		Zalety stosowania procedur składowanych przedstawiono na slajdzie.
Slajd 6		Zalety stosowania procedur składowanych przedstawiono na slajdzie.

Slajd 7



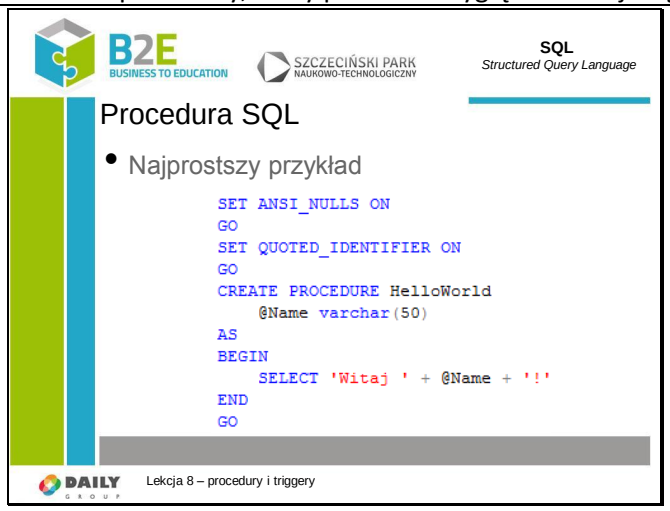
Zarządzanie procedurami składowanymi jest bardzo proste, bo możemy to robić przy pomocy Microsoft SQL Server Management Studio. Tam w Object Explorerze znajdują się wszystkie bazy danych, a jeśli rozwiniemy odpowiednio drzewo, to znajdziemy również gałąź o nazwie Programmability, a w niej Stored Procedures. W tym właśnie miejscu, trzymane są wszystkie procedury przeznaczone dla konkretnej bazy danych. Idąc głębiej, możemy uzyskać dostęp do sporej kolekcji procedur systemowych. Aby utworzyć nową procedurę, wystarczy kliknąć prawym przyciskiem myszy na katalogu Stored procedures i z menu kontekstowego wybrać opcję New stored procedure.

Slajd 8

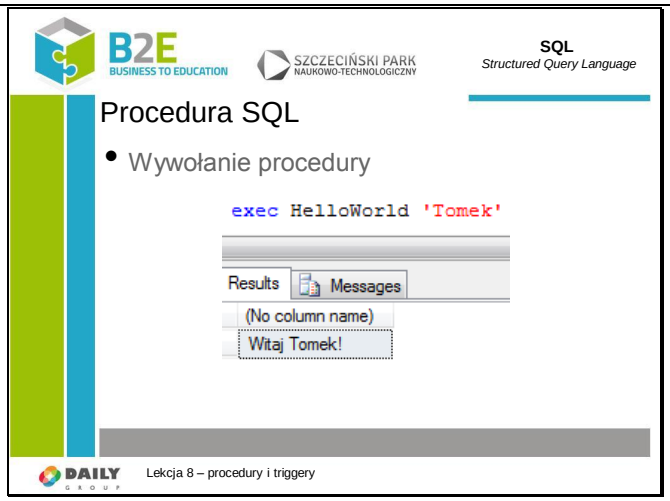
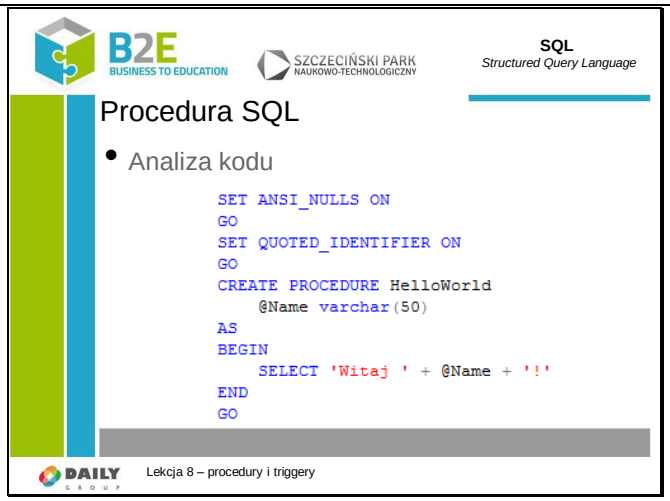


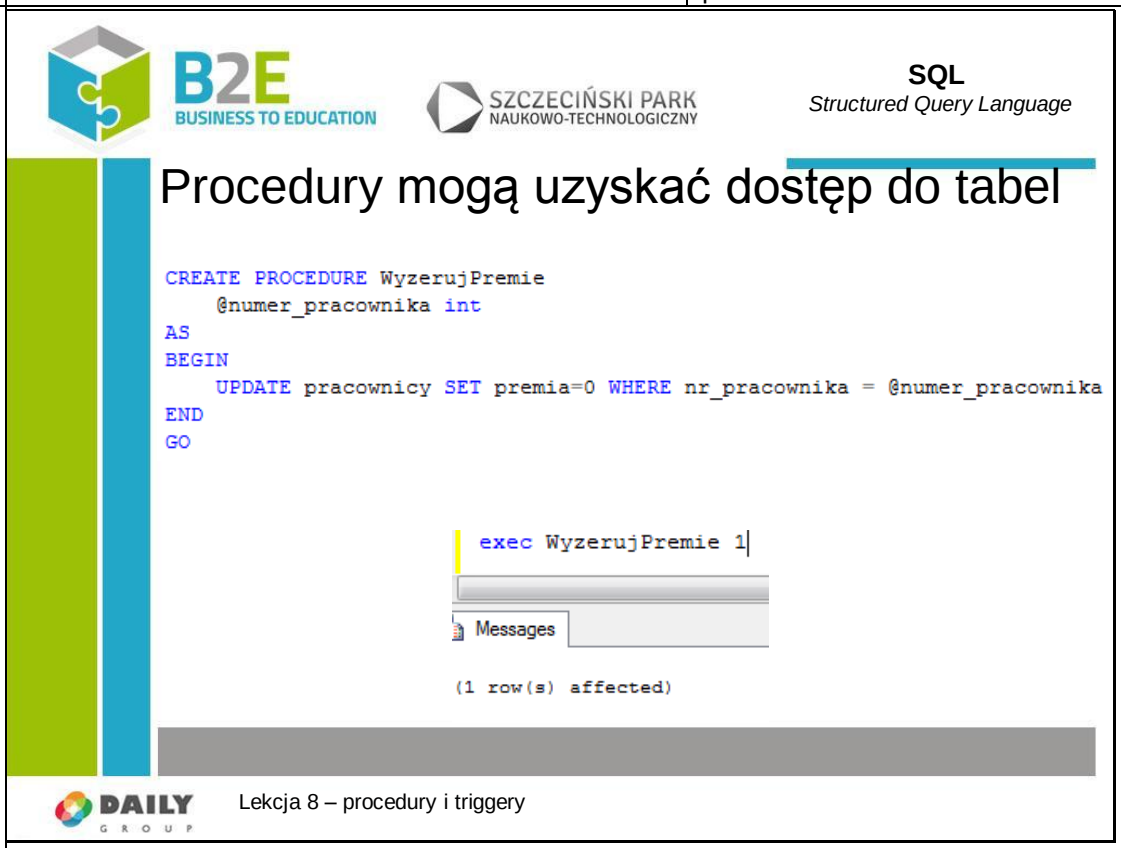
Wybranie opcji „New stored procedure” w MS SMS, spowoduje utworzenie pustego szablonu procedury, który powinien wyglądać mniej więcej tak.

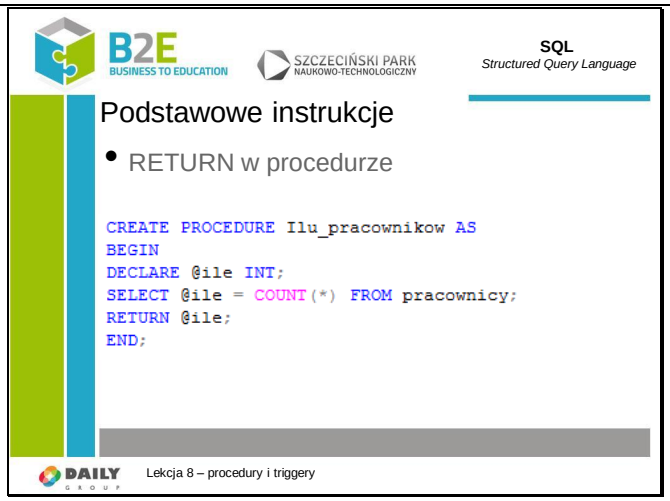
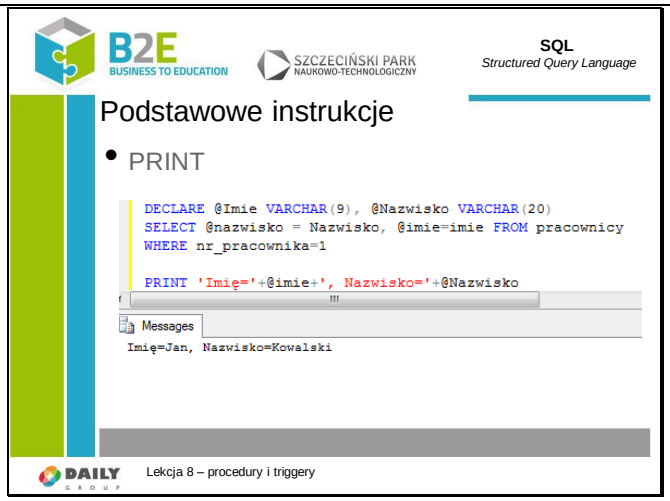
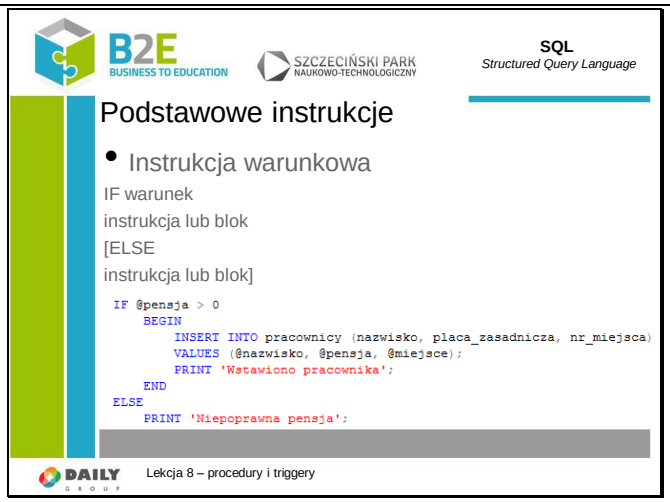
Slajd 9

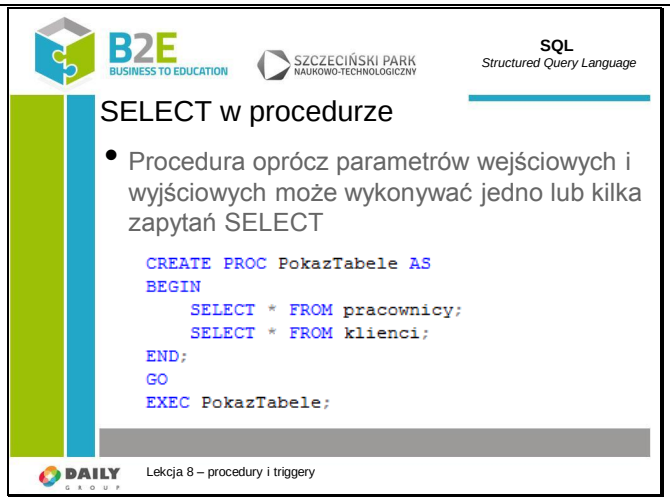


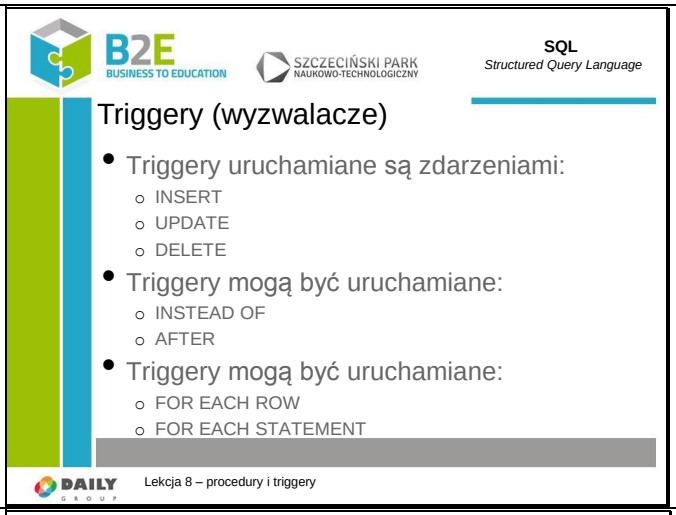
Bardzo prosty przykład procedury składowanej. W szkielecie z poprzedniego sklajdu wprowadziliśmy parametr wejściowy @Name typu varchar(50), procedura wykonuje polecenie SELECT i wyświetla Witaj + parametr + wykrzyknik.

Slajd 10		Wywołanie procedury z poprzedniego slajdu. Procedurę wywołujemy zawsze za pomocą polecenia EXEC (z ang. execute).
Slajd 11		Przeanalizujmy teraz poszczególne polecenia w naszej procedurze: W linii 1 oraz 3, określamy ustawienia Servera SQL. Pierwszy zapis mówi o zasadach porównywania wartości *NULL*, natomiast drugi zapis oznacza sposób używania cudzysłowów. Dopiero w linii 5, zaczynam właściwy kod procedury. Aby stworzyć procedurę, wykorzystujemy standardowe polecenie *CREATE*. Następnie, mówimy jaki obiekt chcemy utworzyć i jaką nazwę ma przyjąć. Kolejnym krokiem, będzie określenie parametrów. Choć są one opcjonalne, jednak mogą okazać się bardzo przydatne. Np. w tym przypadku określamy imię użytkownika. Każdy parametr musi zaczynać się od znaku @. Musimy podać także jego typ oraz ewentualnie długość. Parametry (jeśli jest ich więcej niż jeden), rozdzielamy przecinkiem. Po utworzeniu nagłówka procedury, w linii 7, stosujemy słowo kluczowe *AS*, które pozwala nam na wprowadzenie właściwego ciała procedury. Treść procedury, umieszczamy pomiędzy kolejnymi słowami kluczowymi - *BEGIN* oraz *END*.

		Kluczowym poleceniem naszej procedury, jest zawarte w linii 10 „ SELECT 'Witaj ' + @Name + '!'”, która wykonuje *SELECT* z naszym imieniem. Procedura ta nie odwołuje się do żadnej tabeli z bazy danych, jedynie wyświetla powitanie na podstawie podanego parametru.
Slajd 12		Procedury składowane w SQL mają dostęp do tabel w bazie danych. Na powyższym przykładzie zademonstrowano procedurę składowaną ustawiającą premię pracownika na 0. Jeśli operację zerowania premii przeprowadzamy często – warto stworzyć procedrę, gdzie podając numer pracownika kasujemy premię. Dzięki temu nie musimy każdorazowo używać odpowiedniego polecenia UPDATE. Poniżej przykład wywołania procedury wraz z odpowiedzią serwera – jeden wiersz został zaktualizowany.

Slajd 13		Procedura zliczy wszystkie wiersze z tabeli pracownicy i zwróci wartość poprzez RETURN. Stosuje się ją zwykle do przekazania stanu obliczeń wywołania procedury, np. czy i jaki wystąpił błąd.
Slajd 14		Do debuggowania (śledzenia poszczególnych kroków fragmentu kodu), najczęściej używa się instrukcji PRINT, która w konsoli umożliwia wyświetlenie (wydruk) zmiennej lub ciągu znaków.
Slajd 15		Przykład warunku IF-ELSE. Jeśli pensja jest większa niż zero – wykonywany jest blok instrukcji zawarty pomiędzy BEGIN oraz END, w przeciwnym razie (ELSE) wykonywany jest PRINT.

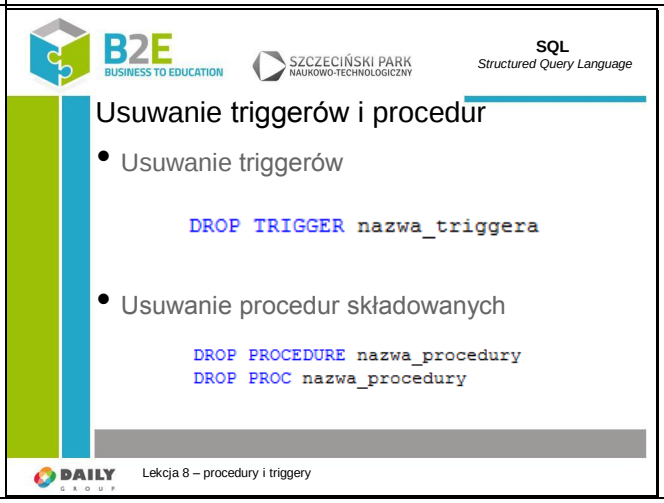
Slajd 16		Przykład warunku iteracji - WHILE. Istnieje możliwość wyjścia z pętli używając instrukcji BREAK. Instrukcja CONTINUE powoduje, że reszta instrukcji w pętli jest ignorowana i następuje wykonanie kolejnej iteracji.
Slajd 17		Przykład procedury wywołującej dwa zapytania typu SELECT. Rezultatem działania tej procedury będzie wyświetlenie zawartości obu tabel: klienci i pracownicy.
Slajd 18		Definicja triggerów

Slajd 19		Wyzwalacze uruchamiane są zdarzeniami: INSERT, UPDATE, DELETE. Wyzwalacze mogą być uruchamiane: INSTEAD OF, AFTER, czyli odpowiednio zamiast lub po modyfikacji danych. Wyzwalacze mogą być uruchamiane: FOR EACH ROW, FOR EACH STATEMENT, czyli dla każdego wiersza, którego dotyczy działanie lub raz dla całej instrukcji.
Slajd 20		Składnia wyzwalacza zawiera nazwę triggera, tabelę, na której trigger będzie działał oraz warunek wystąpienia: INSERT – trigger zostanie uruchomiony w czasie wstawiania danych do tabeli UPDATE – trigger zostanie uruchomiony w czasie aktualizacji danych w tabeli DELETE – trigger zostanie uruchomiony w czasie usuwania danych z tabeli

Slajd 21

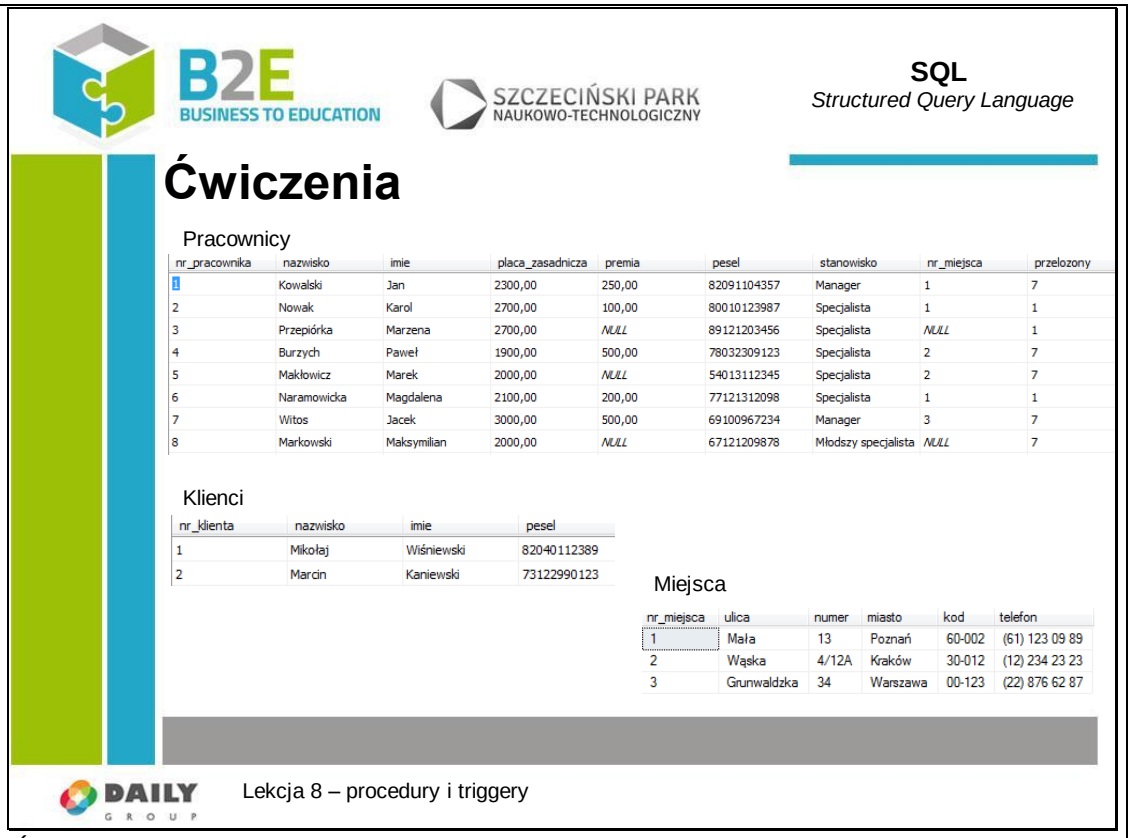
Przykład demonstrujący utworzenie triggera, który zabezpiecza nas przed omyłkowym usunięciem danych z tabeli pracownicy. Po zdefiniowaniu triggera, próbujemy usunąć dane.

Slajd 22

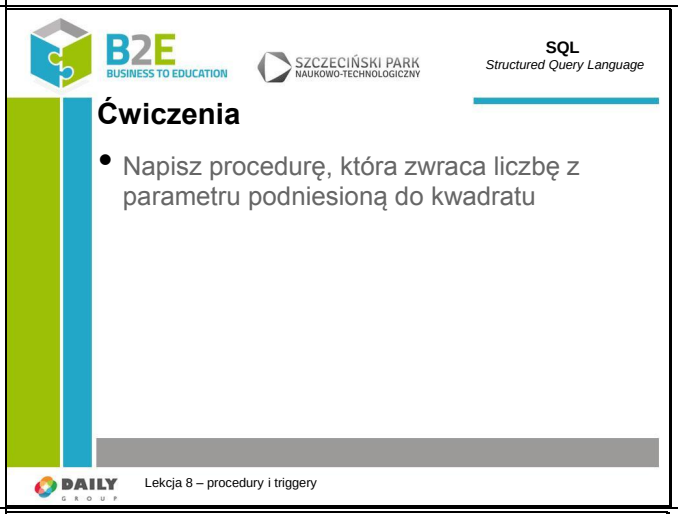


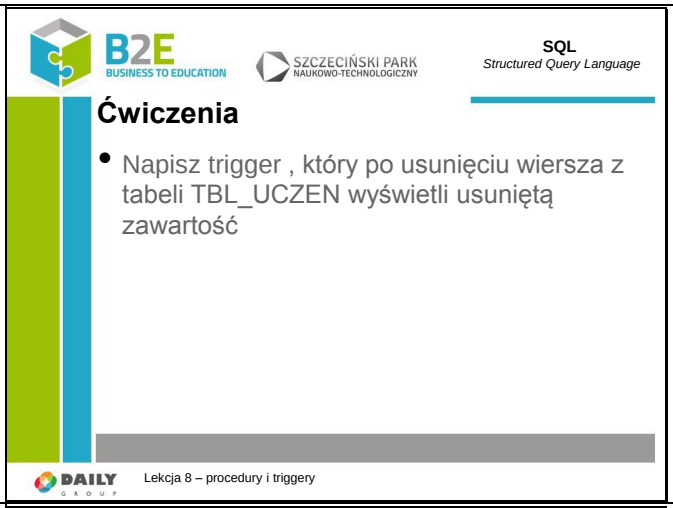
Usuwanie obiektów typu triggei i procedura składowana. Proszę zwrócić uwagę, że SQL (T-SQL) wyrażenia PROC i PROCEDURE są równoważne.

Slajd 23



Ćwiczenie 1:
Trigger uniemożliwiający usuwanie danych z tabeli miejsca.
Ćwiczenie 2:
Utwórz trigger, który w utworzy kopię danych kasowanych z tabeli klienci w tabeli klienci_kopia z identyczną strukturą jak klienci i dodatkową kolumną data usunięcia.
Ćwiczenie 3:
Utwórz trigger, który w czasie wstawiania nowych osób do tabeli pracownicy, automatycznie doda ich do tabeli klienci.
Ćwiczenie 4:
Napisz procedurę składowaną, która wylicza planowany budżet na pensje i premie na najbliższe 12 miesięcy.
Ćwiczenie 5:
Napisz procedurę składowaną, do wykorzystania przez aplikację drukującą koperty, która zwraca w tabeli imię, nazwisko i adres zatrudnienia pracownika w jednej kolumnie.

Slajd 24		EXEC OBLICZ
Slajd 25		CREATE PROCEDURE potega @liczba int AS BEGIN SELECT @liczba * @liczba END
Slajd 26		DROP PROCEDURE potega

Slajd 27		CREATE TRIGGER ttbl_uczen ON tbl_uczen AFTER DELETE AS BEGIN SELECT * FROM DELETED END
Slajd 28		CREATE TRIGGER ttbl_uczen1 ON tbl_uczen AFTER INSERT AS BEGIN PRINT „Gratulacje dodałeś wiersz” END

Slajd 29

Procedura składowana	Prekompilowane wyrażenia języka SQL przechowywane na serwerze bazodanowym
Wyzwalacz	Predefiniowane operacje na danych wyzwalane zdarzeniami
Warunek IF	Instrukcja warunkowa wykorzystywana w funkcjach, procedurach i triggerach
Pętla WHILE	Instrukcja iteracji wykorzystywana w funkcjach, procedurach i triggerach

Tabela podsumowująca poznane wyrażenia

9.8.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 23-28. Uczniowie wykorzystają zdobytą wiedzę przy tworzeniu procedur i triggerów.

9.8.4 Opis założonych osiągnięć ucznia

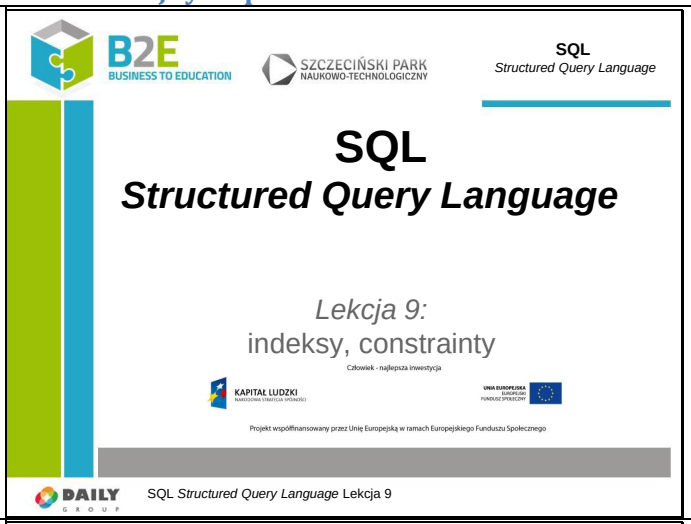
Po tej lekcji uczniowie będą potrafili samodzielnie implementować procedury i triggery w języku TSQL.

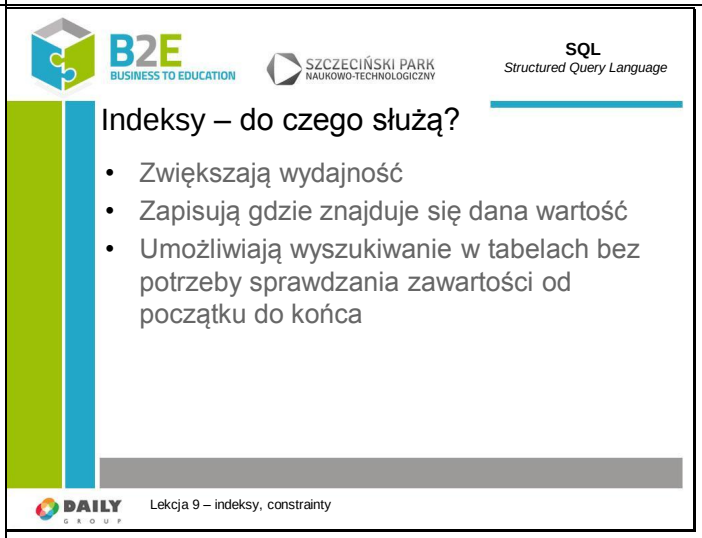
9.9 Lekcja 9 - Indeksy, constrainty

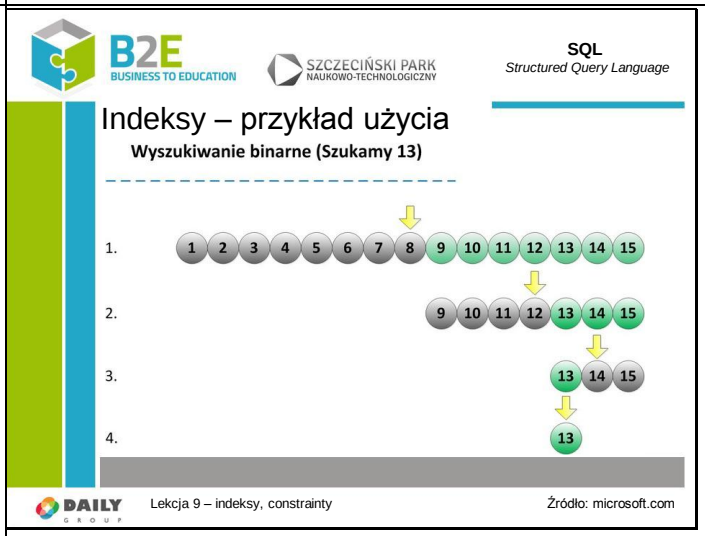
9.9.1 Cel lekcji

Celem lekcji jest poznanie struktury optymalizującej bazę danych jaką jest indeks. W drugiej części lekcji uczniowie poznają ograniczenia bazy – contraint, służące zachowaniu integralności danych.

9.9.2 Treść - slajdy z opisem

<table>
<tr>
<td>Slajd 1</td>
<td> 
</td>
<td>W trakcie tej lekcji poznamy strukturę optymalizującą bazę danych jaką jest indeks. W drugiej części lekcji poznamy ograniczenie bazy – contraint, służące zachowaniu integralności danych.</td>
</tr>
<tr>
<td>Slajd 2</td>
<td> 
</td>
<td>Przypomnienie pojęć z poprzednich zajęć</td>
</tr>
<tr>
<td>Slajd 3</td>
<td> 
</td>
<td>Po co kartkować całą książkę dla znalezienia jednej interesującej nas informacji, jeśli możemy zajrzeć do spisu treści i na jego podstawie odnaleźć stronę, na której znajduje się to, czego szukamy. Zaoszczędzimy w ten sposób cenny czas choćby dlatego, że spis treści jest zwykle zorganizowany w sposób alfabetyczny, co znacznie upraszcza wyszukanie frazy, która nas interesuje. Indeks w bazie danych wykorzystuje się przy zapytaniach typu DQL (SELECT), które mają na celu wyszukiwanie odpowiednich wartości w bazie danych. Podczas realizowania zapytania optymalizator (Serwer</td>
</tr>
</table>

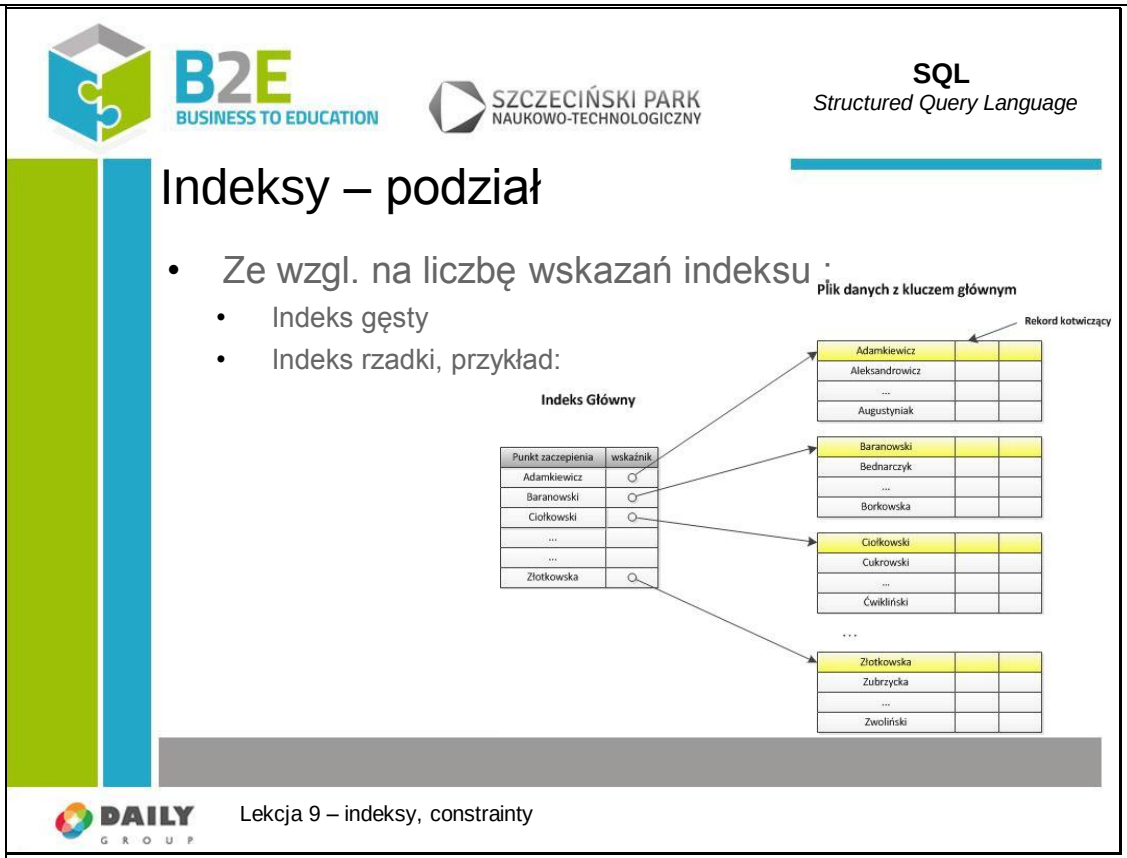
		SQL) najpierw przeszukuje indeks, który jest uporządkowany, a następnie na podstawie indeksu odczytuje odpowiednie rekordy. Indeks posiada strukturę logiczną i fizyczną niezależną od tabeli, do jakiej się odwołuje. Posiada również własną przestrzeń dyskową oraz jest automatycznie utrzymywany przez system zarządzania bazą danych.
Slajd 4		
Slajd 5		Bez indeksów SQL serwer oferuje nam dostęp do danych – są one dostępne. Jednak przy większych zbiorach (niż nasze tabele klienci, miejsca, pracownicy), musimy zwrócić uwagę na wydajność bazy danych. W tym momencie powinniśmy zastanowić się nad indeksami, gdyż dzięki nim szybciej odnajdujemy dane w tabeli. Podczas wykonywania polecenia typu DQL (SELECT*)* serwer bazy danych musi wykonać bardzo dużo operacji – wybieranie danych, sortowanie wyników itd. SQL Server może pracować z bardzo dużą ilością rekordów, a często z dziesiątkami milionów rekordów w tabeli, dlatego niezwykle ważną rolę odgrywa w nim optymalizacja wszelkiego rodzaju

		wyszukiwania oraz sortowania. Rekordy ułożone są w tabeli w takiej kolejności, w jakiej zostały dodane*, co oznacza, że jeśli spróbujemy wyszukać w bazie danych np. sklepu ceny danego produktu, to SQL Server musi za każdym razem na nowo przetrząsać od początku do końca wszystko, co tam się znajduje. Kiedy produktów będzie parę tysięcy, może to potrwać dosyć długo. W takiej sytuacji pomocą służą nam indeksy. Indeks nałożony na pole jest niczym innym jak kopią jego zawartości, tyle że posortowaną i odpowiednio ułożoną. *Taka sytuacja występuje domyślnie, w przypadku gdy PK jest indeksem klastrowanym. Można to zmienić wskazując inny indeks klastrowany niż PK.
Slajd 6		Mamy tysiąc wartości posortowanych rosnąco, a nas interesują rekordy z zakresu od 600 do 700. Jeśli rekordy są posortowane rosnąco, SQL Server może zacząć od środkowego rekordu tego zbioru i sprawdzić, czy znaleziona wartość jest mniejsza, czy większa od podanego zakresu. Naturalnie, jeśli wartość środkowego elementu jest mniejsza od podanego zakresu, przeszukuje tylko późniejsze rekordy (również przez wyszukiwanie połówkowe w nowym zbiorze, stanowiącym połowę zbioru poprzedniego), jeśli wartość tego elementu jest większa od podanego zakresu, SQL Server szuka jedynie w rekordach wcześniejszych. Jeśli z kolei poszukiwana wartość mieści się w podanym zakresie, porusza się w obu kierunkach, dopóki nie wypadnie poza założony zakres. W sytuacji, gdy nie byłoby

		posortowanej tablicy, musiałby najpierw albo ją posortować, albo przejrzeć wszystkie wartości w celu wyodrębnienia tych, które go interesują. Slajd przedstawia przykład binarnego wyszukiwania elementu numer 13 w zbiorze uporządkowanym 15-elementowym. Dzięki posortowaniu rekordów SQL Server na wstępie odrzucił całe mnóstwo rekordów, które nie pasują do naszego zapytania. Dodatkowo mamy zrealizowaną automatycznie funkcję sortowania, bez uruchamiania jakiejkolwiek funkcji sortującej. Źródło: microsoft.com
Slajd 7	 Niestety, nie możemy stosować indeksów wszędzie gdzie tylko możliwe. To dlatego, że ceną za zwiększenia wydajności, jakie oferuje nam indeks, jest zwiększenie rozmiarów bazy, gdyż indeks również potrzebuje trochę miejsca. Przestrzeń dyskowa jest z kolei kluczowym aspektem podczas projektowania bazy danych. A zatem już na początku została obalona postawiona teza o popieraniu rozrzutności podczas indeksowania danych. Indeksy zazwyczaj

	zajmują tyle samo miejsca co dane i jest to sytuacja zupełnie naturalna. Łatwiej jest zaakceptować wydatki na przestrzeń dyskową niż akceptować czas oczekiwania na zapytanie zwiększony z sekund do kilku godzin. Należy się zastanowić, na których polach indeks będzie nam potrzebny, a na których jest zwyczajnie zbędny. W tym momencie pomocne będzie środowisko SQL Server, które udzieli nam diagnostycznej informacji o wykonywaniu zapytania, na podstawie której przekonamy się, czy wykorzystał on nasze indeksy. Wystarczy zapytanie SELECT poprzedzić słowem EXPLAIN, a system nam podpowie. O wyborze indeksu decyduje optymalizator kwerend określający, które (ewentualnie) indeksy będą najbardziej użyteczne. Warto również w tym punkcie określić, czym jest selektywność indeksu. Jest to nic innego, jak parametr określający, czy indeks na określonych kolumnach może być przydatny. Wartość tego parametru wyliczamy ze wzoru: S = U/W gdzie: • S – selektywność; • U – liczba unikalnych wartości dla kolumny; W – liczba wszystkich wierszy w kolumnie. Jeśli wartość selektywności indeksu wynosi mniej niż 85%, SQL Server raczej nie będzie z niego korzystał. Indeksy, z wyjątkiem grupujących, w pewnych sytuacjach mogą wydłużać czas operacji wstawiania i modyfikowania danych. Indeks grupujący powinien być tworzony dla kolumny, według której użytkownicy często sortują dane odczytywane z tabeli lub dla kolumn przechowujących wartości, na podstawie których zwracane są zbiory danych.

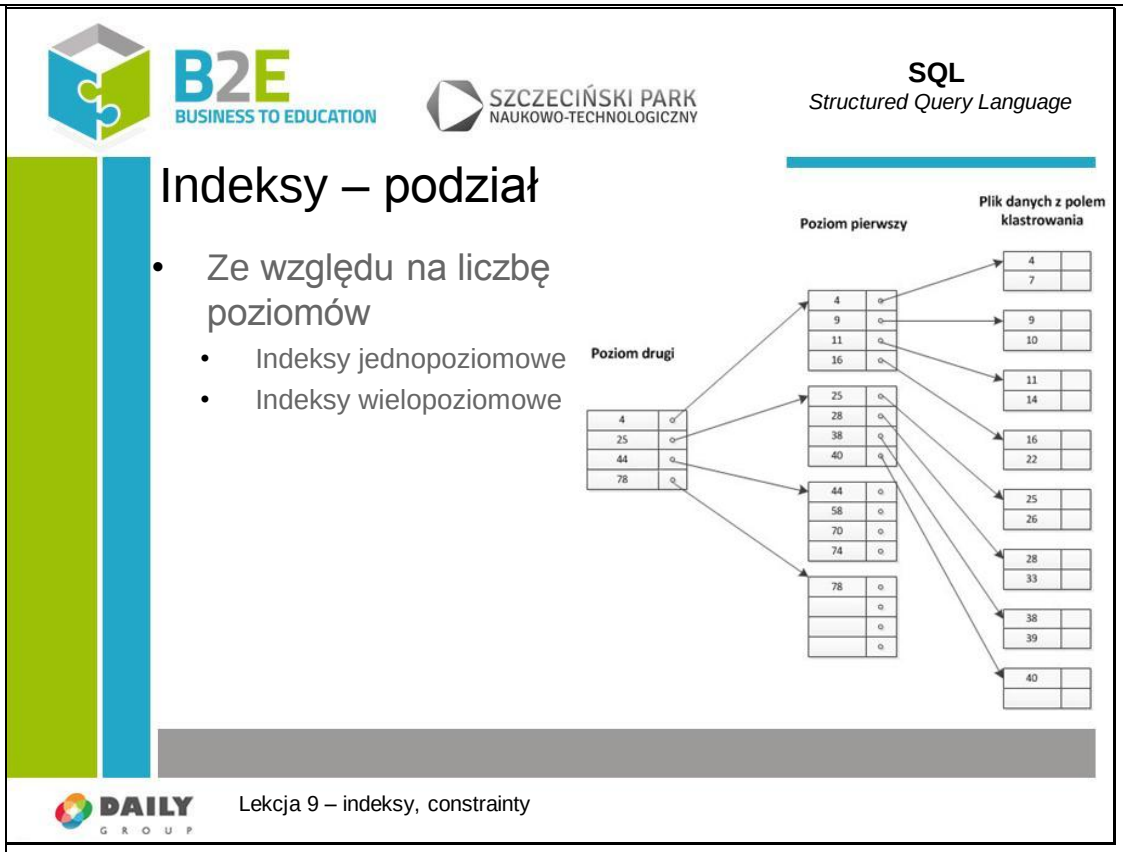
Slajd 8



Z punktu widzenia liczby wskazań indeksu do pliku danych rozróżnia się:

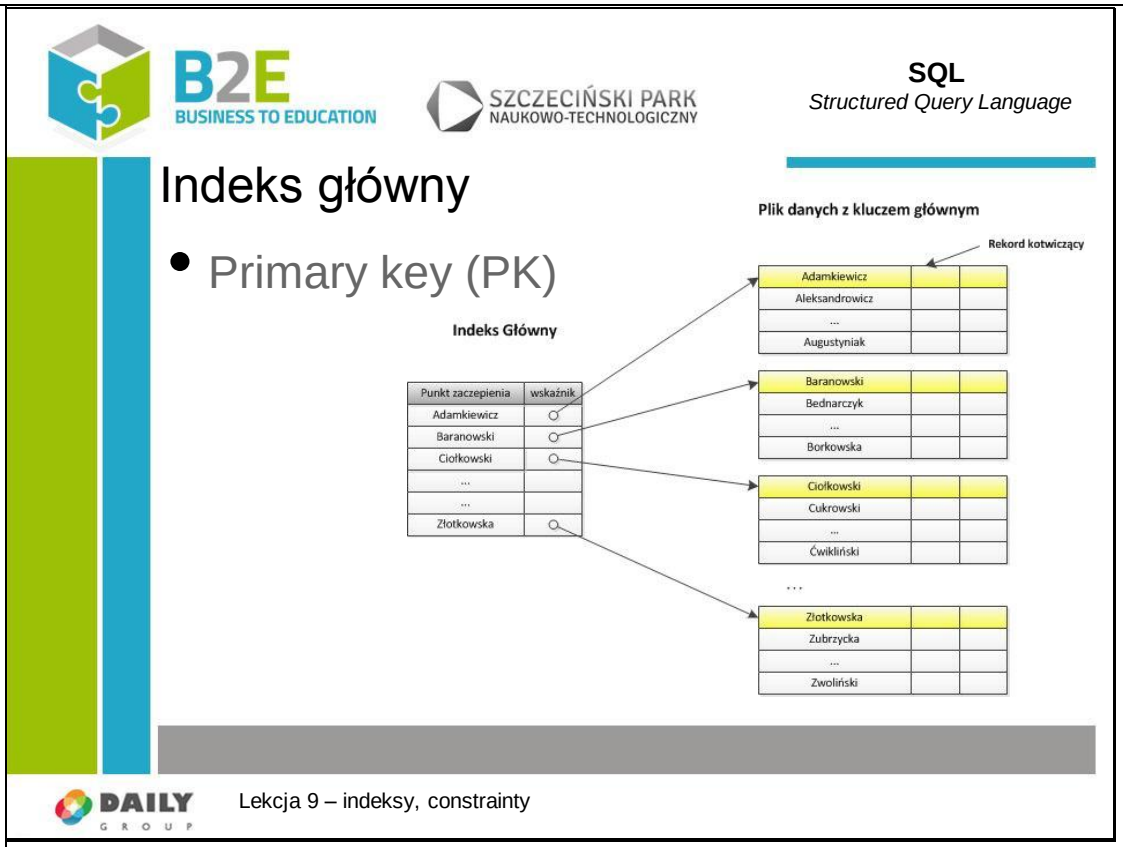
- Indeks gęsty (dense) – zawiera wpis dla każdej wartości klucza wyszukiwania, czyli dla każdego rekordu.
- Indeks rzadki (sparse) – posiada wpis jedynie dla niektórych wartości wyszukiwania (np. bloków).

Slajd 9



Na slajdzie przykład indeksu wielopoziomowego.
Indeksy wielopoziomowe – dla pierwszego poziomu tworzymy indeks podstawowy i nazywamy go indeksem drugiego poziomu. Analogicznie dla poziomu drugiego, gdzie tworzy się indeks poziomu trzeciego.

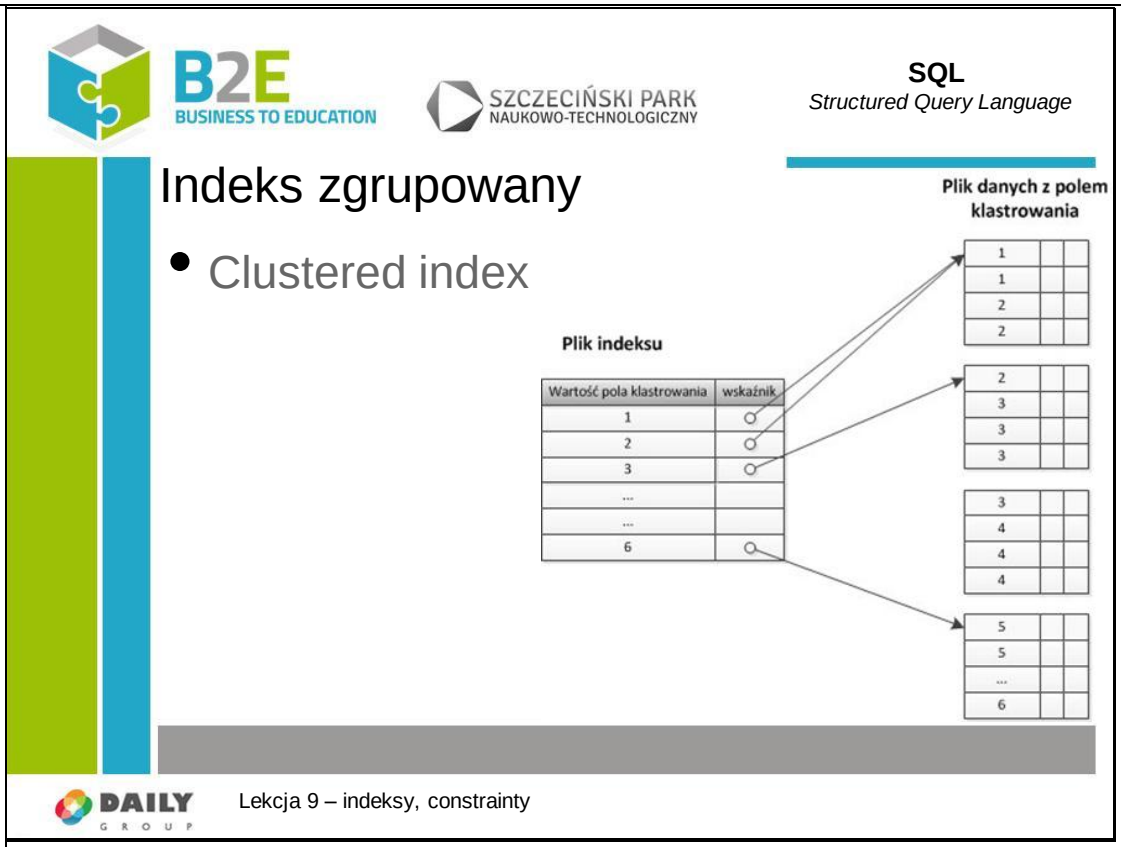
Slajd 10



Indeks główny (Primary index) – zwany także podstawowym, jest założony na kluczu podstawowym pliku uporządkowanego i zawiera jeden klucz dla każdego bloku dyskowego. Pierwszy z rekordów danego bloku nazywamy rekordem zaczepienia lub rekordem kotwiczącym. Należy on do grupy indeksów rzadkich.

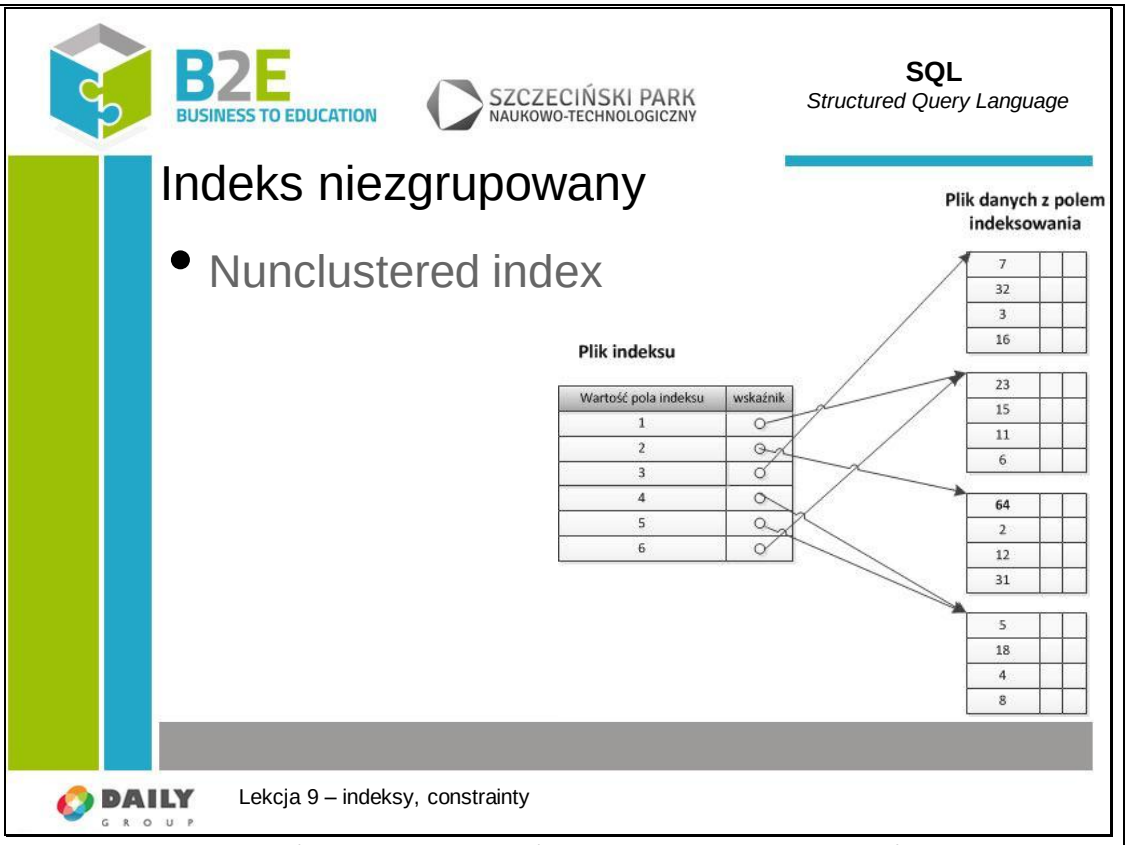
PK jest indeksem klastrowanym (domyślnie). Wskazanie innego indeksu klastrowanego (można jeden) powoduje zmianę uporządkowania struktury pliku zgodnie ze wskazanym indeksem klastrowanym.

Slajd 11



Indeks zgrupowany (Clustered index) – jest założony na atrybucie niebędącym kluczem podstawowym pliku uporządkowanego (nieunikatowym) porządkującym pliku uporządkowanego. Indeks zawiera jeden klucz dla każdej wartości atrybutu. Posiada dwa pola – pierwsze ma ten sam typ co pole klastrowania, drugie – wskaźnik. Indeks zawiera wpis do każdej odrębnej wartości klastrowania oraz wskaźnik na pierwszy blok, do którego ona należy. Należy do grupy indeksów rzadkich.

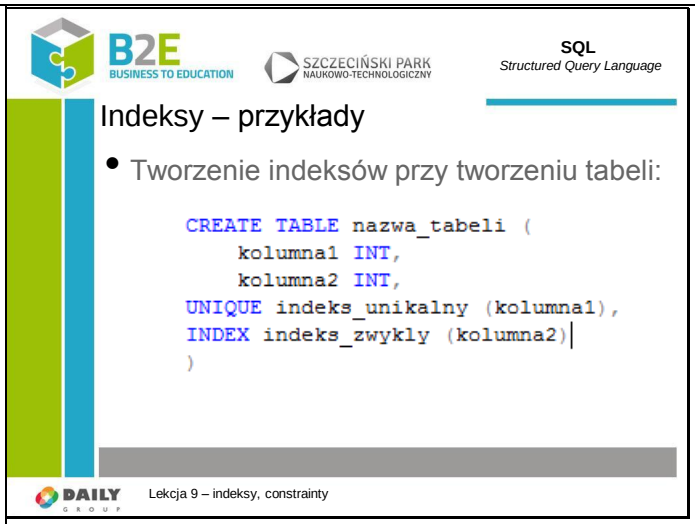
Slajd 12



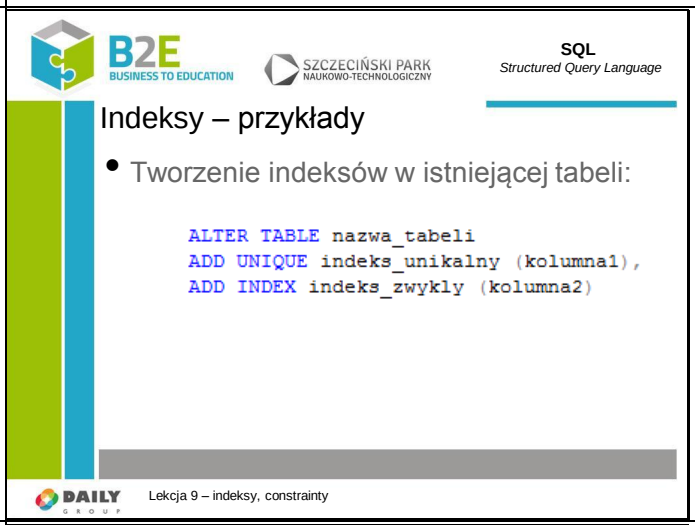
Indeks niezgrupowany (Nunclustered index) – jest zakładany na pole, które ma unikatowe wartości w każdym rekordzie lub które nie jest polem klucza i posiada powtarzające się wartości. Plik indeksu niezgrupowanego jest uporządkowany i posiada dwa pola – jedno jest tego typu co wybrane pole niebędące polem uporządkowania pliku (pole indeksujące), drugie –wskaźnikiem na blok lub rekord. Dla jednego pliku może być wiele indeksów drugorzędnych. Należy do grupy indeksów zagęszczonych. Wskaźniki we wpisach indeksu są wskaźnikami na bloki.

Jest on zakładany na atrybucie indeksowym pliku danych, który nie jest atrybutem porządkującym tego pliku. Każdy rekord pliku danych posiada swój odpowiednik w rekordzie indeksu. Stąd indeks wtórny jest indeksem gęstym. Rekord indeksu wtórnego składa się z dwóch pól – wartości pola indeksowego i wskaźnika albo do rekordu albo do bloku danych zawierającego ten rekord.

Slajd 13



Slajd 14



Slajd 15

Slajd 16		Dwa powyższe przykłady mogą wydawać się podobne, jednak występuje pomiędzy nimi zasadnicza różnica. Rozważmy, jak zostanie zinterpretowany przez system SQL Server poniższy przykład: SELECT * FROM osoby WHERE imie='Jan' AND nazwisko='Kowalski' Jeśli tabela posiada dwa różne indeksy, każdy na pojedynczej kolumnie, baza danych wykona to zapytanie w następujących krokach: wyszuka wszystkie rekordy, gdzie występuje imie = Jan; wyszuka wszystkie rekordy, gdzie występuje nazwisko = Kowalski; obliczy część wspólną zbiorów rekordów z pierwszego i drugiego kroku, i zwróci ją jako wynik zapytania. W drugim przypadku, tzn. kiedy indeks jest nałożony na kolumnach (imię, nazwisko), baza danych może wyszukać potrzebne dane w jednym kroku. Sprawdzi równocześnie wartości w polach „imię” i „nazwisko”.
Slajd 17		

Slajd 18	B2E BUSINESS TO EDUCATION · SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY SQL *Structured Query Language* **Podstawowe CONSTRAINT** • NOT NULL, • CHECK, • UNIQUE, • PRIMARY KEY, • FOREIGN KEY DAILY GROUP Lekcja 9 – indeksy, constrainty	W trakcie wcześniejszych lekcji omówiliśmy już NULL/NOT NULL oraz PRIMARY KEY, na kolejnych slajdach przyjrzymy się pozostałym.

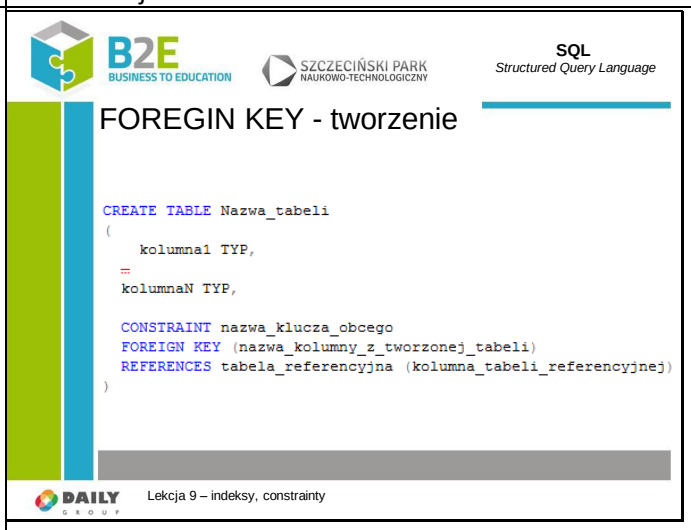
Slajd 19	B2E BUSINESS TO EDUCATION · SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY
	SQL *Structured Query Language*
	CHECK
	• Umożliwia sprawdzenie (walidację) wartości przed zapisem do tabeli w bazie danych, np.
	`ALTER TABLE dbo.pracownicy WITH CHECK`
	`ADD CONSTRAINT CK_placa CHECK ((placa_zasadnicza>(0)));`
	DAILY GROUP Lekcja 9 – indeksy, constrainty
		Wykonanie operacji z przykładu uniemożliwi wstawianie nowych wierszy do tabeli pracownicy, lub modyfikację istniejących zmieniających płacę zasadniczą na wartość mniejszą lub równą 0. Uwaga – w warunku można użyć także funkcję użytkownika lub funkcję systemową!

Slajd 20	B2E BUSINESS TO EDUCATION · SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY
	SQL *Structured Query Language*
	UNIQUE
	• Umożliwia sprawdzenie (walidację) wartości przed zapisem do tabeli w bazie danych pod kątem niepowtarzalności
	`ALTER TABLE pracownicy`
	`ADD UNIQUE (pesel)`
	DAILY GROUP Lekcja 9 – indeksy, constrainty
		Wykonanie operacji z przykładu uniemożliwi wstawianie nowych wierszy do tabeli pracownicy, lub modyfikację istniejących zmieniających płacę zasadniczą na wartość mniejszą lub równą 0.

Slajd 21

Przykład tabeli pracownicy, gdzie nr_miejsca jest kluczem obcym do kolumny nr_miejsca w tabeli miejsca.

Slajd 22

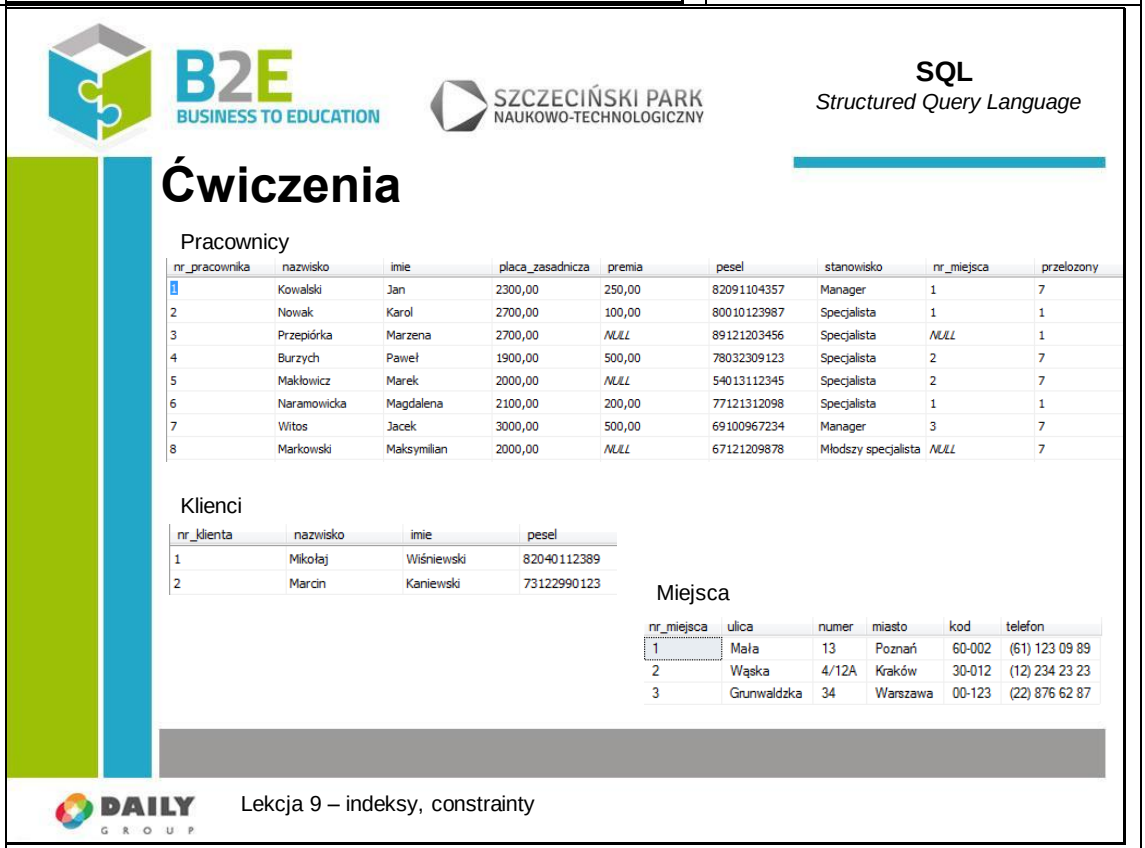


T-SQL pozwala na oznaczenie danej kolumny, bądź zbioru kolumn, jako klucza obcego. Jedną z metod jest zdefiniowanie tzw. funkcji CONSTRAINT podczas tworzenia tabeli. Składnia przykładowego zapytania została przedstawiona na slajdzie. Jeżeli zapytanie zostanie wykonane bezbłędnie, w SQL Server Management Studio, w lokalizacji Object Explorer->Baza Danych->Nazwa Tabeli->Keys, utworzony klucz obcy powinien funkcjonować włącznie z podaną nazwą.

Slajd 23

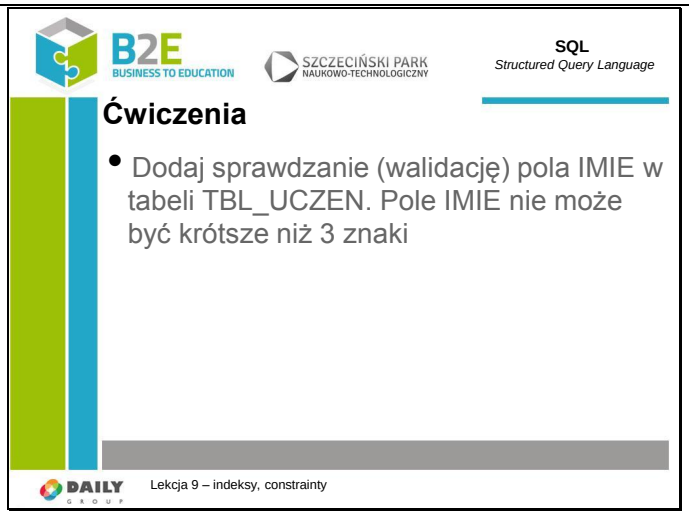
Podsumowanie informacji o kluczach obcych.

Slajd 24



Ćwiczenie 1:
Utwórz klucze główne w trzech tabelach
Ćwiczenie 2:
Utwórz klucz obcy w tabeli pracownicy – kolumna nr_miejsca
Ćwiczenie 3:
Dodaj unikalność pola pesel
Ćwiczenie 4:
Wprowadź mechanizm zabezpieczający w tabeli pracownicy przed dodaniem wiersza z premią mniejszą niż 0
Ćwiczenie 5:
Jak zoptymalizować zapytania do tabeli klientów zakładając, że zazwyczaj wyszukiwani są po

	nazwisku? Zaproponuj skrypt wprowadzający zmianę. Ćwiczenie 6: Jakie indeksy należy założyć na tabeli pracownicy, najczęstsze zapytania to wyszukiwaniu po parze kolumn imię i nazwisko oraz po numerze pesel Ćwiczenie 7: Napisz skrypt definiujący tabelę pracownicy, gdzie pola imię, nazwisko, pesel i stanowisko nie mogą być puste, nr_pracownika jest PK, nr_miejsca FK, pesel musi być unikalny, a płaca większa niż 1600zł Ćwiczenie 8: Napisz funkcję sprawdzającą czy wartość nie jest większa od płacy minimalnej (1600zł) podłącz funkcję jako CHECK COINSTRAINT do kolumny płaca zasadnicza w tabeli pracownicy	
Slajd 25		ALTER TABLE TBL_UCZEN ADD PRIMARY KEY (id_ucznia)
Slajd 26		CREATE INDEX i_tbl_uczen ON TBL_UCZEN

Slajd 27		ALTER TABLE TBL_UCZEN ADD UNIQIE (PESEL)
Slajd 28		ALTER TABLE TBL_UCZEN WITH CHECK ADD CONSTRAINT CK_IMIE CHECK(LEN(IMIE)>=3)
Slajd 29		CREATE UNIQUE INDEX indeks_unikalny ON TBL_UCZEN (ID_UCZNIA)

Slajd 30		Tabela podsumowująca poznane wyrażenia

9.9.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 24-29. Zadaniem uczniów będzie praktyczna implementacja indeksów oraz constraintów.

9.9.4 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą mogli samodzielnie implementować tabele, z odpowiednimi ograniczeniami służącymi zachowaniu integralności bazy danych. Dodatkowo po zapoznaniu się z mechanizmem indeksów będą mogli podjąć próbę optymalizacji baz danych.

9.10 Lekcja 10 - Transakcje, zatwierdzanie zmian, obsługa błędów, wprowadzenie do projektowania bazy danych

9.10.1 Cel lekcji

Celem lekcji jest wyjaśnienie transakcyjności bazy danych, wyjaśnienie istoty obsługi błędów oraz wprowadzenie do projektowania baz danych.

9.10.2 Treść - slajdy z opisem

Slajd 1		Ostatnia lekcja w kursie SQL dotyczyć będzie transakcyjności baz danych. Omówimy istotę obsługi błędów. Oraz zajmiemy się zagadnieniami związanymi z normalizacją baz danych – jako podstawy do projektowania baz.

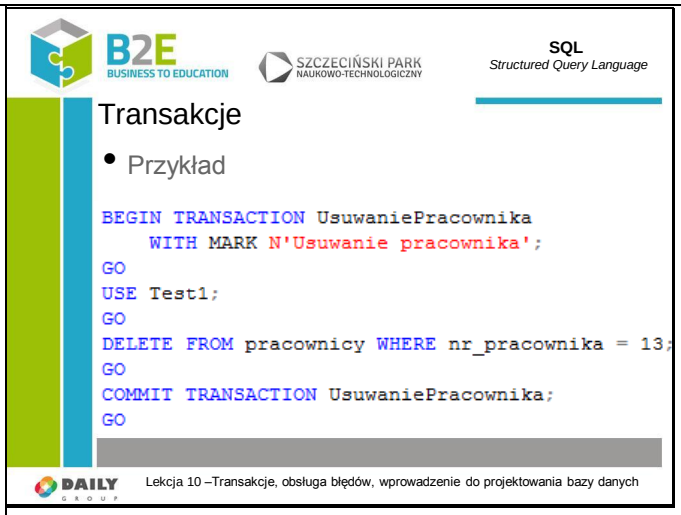
Slajd 2

Podsumowanie zagadnień z poprzedniej lekcji

Slajd 3

Podstawowe komendy związane z transakcjami.

Slajd 4

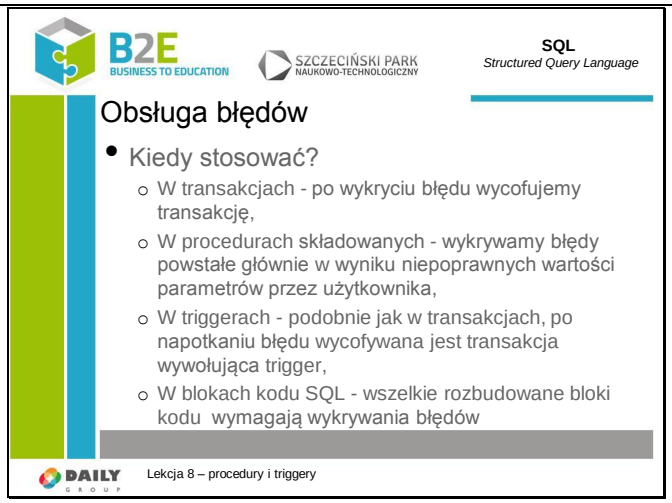


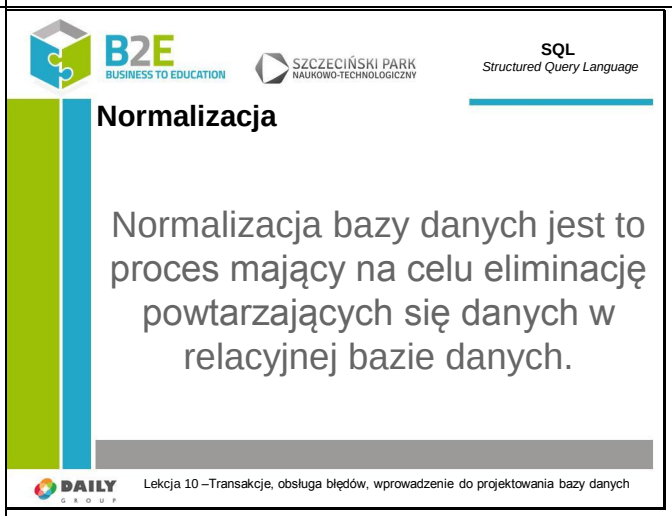
WITH MARK – oznacza transakcje odpowiednim opisem (można go później odnaleźć w logu transakcji).

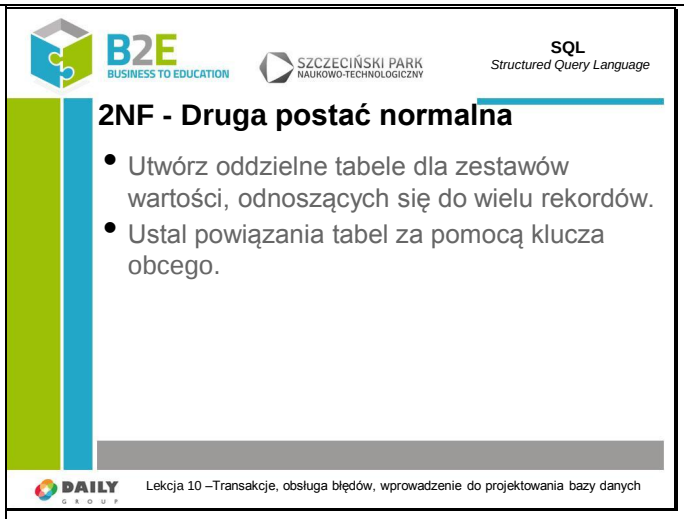
Slajd 5

Tak jak w innych językach programowania, tak w SQLu powinniśmy zadbać o obsługę błędów – to znaczy jeśli coś pójdzie nie tak w trakcie wykonywania kodu, powinniśmy „obsłużyć" problem i odpowiednio zareagować. Na slajdzie przedstawiono przykład prawidłowej obsługi błędów w SQL.

Przykład przedstawia celowe spowodowanie błędu: dzielenie przez 0, jego obsługa polega na odczytaniu z systemu wszystkich możliwych danych i zaprezentowanie użytkownikowi.

Slajd 6		Kiedy obsługa błędów jest niezbędna.
Slajd 7		Jak można wykrywać błędy? Metod na to jest wiele. Po pierwsze można zastosować instrukcje warunkowe, np. IF...ELSE, do sprawdzania wartości zmiennych jeszcze przed wystąpieniem błędu. Druga metoda to wykorzystanie istniejących w SZBD funkcji wykrywających błędy. W systemie Microsoft SQL Server taką funkcją jest @@ERROR, która zwraca numer błędu napotkanego w ostatnio napotkanego błędu w bieżącej sesji. Aktualnie istnieją także inne - lepsze metody wykrywania i obsługi błędów. Chodzi o strukturalną obsługę wyjątków. Jako wyjątek rozumiemy błąd, który wymaga obsługi, jedną z takich metod jest TRY...CATCH (TRY - próbuj, CATCH - przechwyć i obsłuż)

Slajd 8		Wyjaśnienie try – catch. Konstrukcja stosowana jest w większości obecnych języków programowania i jest jednym ze sposobów obsługi wyjątków. Jeśli w bloku TRY nastąpi błąd – nastąpi przejście do bloku CATCH. Jeśli nie będzie błędu – blok CATCH nigdy nie zostanie wywołany.
Slajd 9		Główna idea polega na trzymaniu danych w jednym miejscu, a w razie potrzeby linkowania do danych. Taki sposób tworzenia bazy danych zwiększa bezpieczeństwo danych i zmniejsza ryzyko powstania niespójności (w szczególności problemów anomalii). Istnieją sposoby ustalenia czy dany schemat bazy danych jest "znormalizowany", a jeżeli jest to jak bardzo. Jednym ze sposobów jest przyrównanie danej bazy do schematów zwanych postaciami normalnymi (ang. normal forms lub NF).
Slajd 10		Przykład: Czy pole adres jest polem elementarnym czy nie? Jeśli wiemy, że w czasie operowania na bazie zawsze będziemy potrzebowali całego adresu, to pole adres możemy uznać za elementarne. Jeśli jednak dopuszczamy możliwość, że będziemy potrzebowali tylko samej miejscowości, to wtedy pole adres nie jest już elementarne i nie spełnia 1NF. Należy więc rozbić pole adres, np. na pola: ulica, miejscowość, kod pocztowy (czyli na pola elementarne).

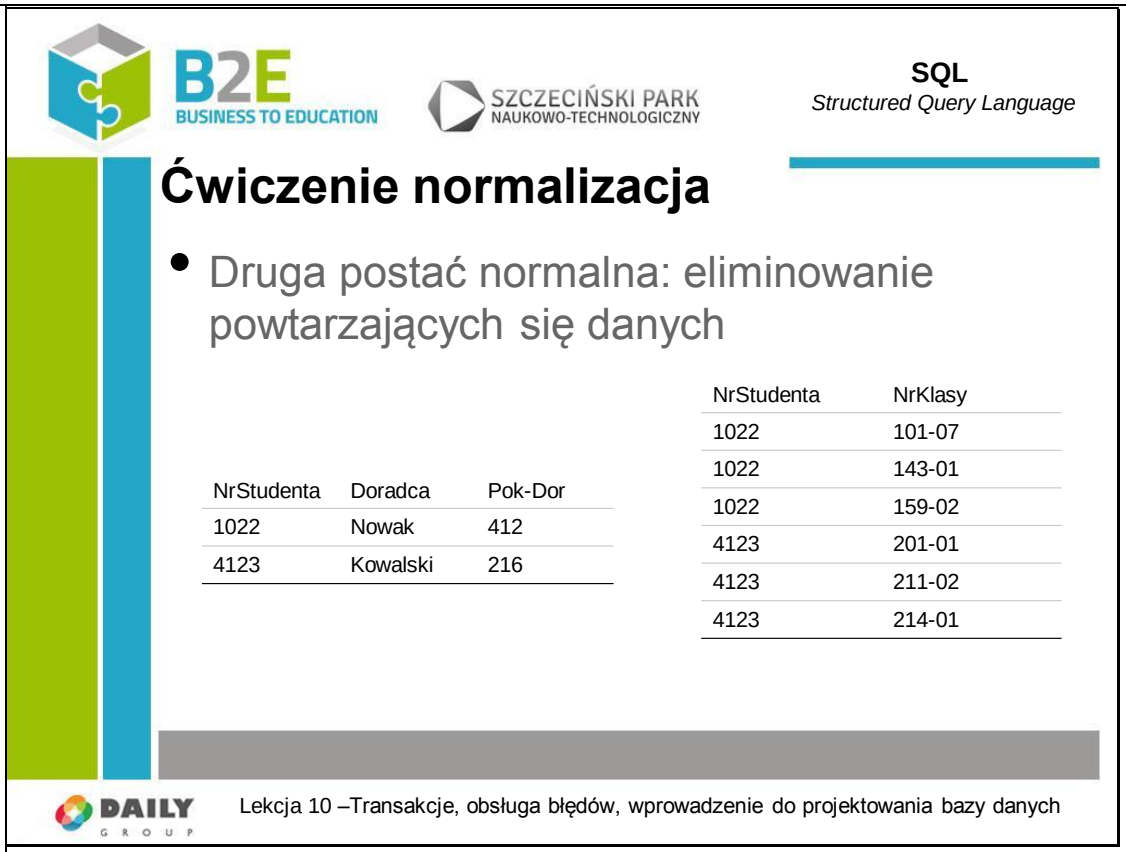
Slajd 11		Rekordy nie powinny zależeć od niczego innego tylko od klucza podstawowego tabeli (w razie potrzeby może to być klucz złożony). Rozważmy na przykład adres klienta w systemie księgowym. Obecność adresu konieczna jest w tabeli Klienci, ale również w tabelach Zamówienia, Wysyłka, Faktury, Należności i Inkaso. Zamiast przechowywać adres w postaci wpisu w każdej tabeli, przechowuje się go w jednym miejscu: albo w tabeli Klienci, albo w oddzielnej tabeli Adresy.
Slajd 12		Wartości rekordu, które nie są częścią jego klucza, nie należą do tabeli. Zazwyczaj, jeśli zawartość grupy pól odnosi się do więcej niż jednego rekordu tabeli, należy rozważyć umieszczenie tych pól w oddzielnej tabeli. Na przykład w tabeli Rekrutacja pracowników może znajdować się nazwa i adres uczelni, którą ukończył kandydat. Do korespondencji seryjnej potrzebna jest jednak kompletna lista uczelni. Jeśli informacje o uczelniach przechowywane są w tabeli Kandydaci, nie ma możliwości wyświetlenia listy uczelni bez aktualnych kandydatów. Utwórz oddzielną tabelę Uczelnie i połącz ją z tabelą Kandydaci za pomocą klucza z kodem uczelni. WYJĄTEK: Stosowanie reguł trzeciej postaci normalnej, chociaż teoretycznie wskazane, nie zawsze jest praktyczne. Chcąc wyeliminować wszystkie możliwe wewnętrzne zależności pomiędzy polami tabeli Klienci, należałoby utworzyć oddzielne tabele dla miast, kodów pocztowych, przedstawicieli handlowych, klas

		klienta i innych czynników, które mogą być zduplikowane w wielu rekordach. Normalizacja oznacza teoretycznie poprawę wydajności. Jednak wiele mniejszych tabel może spowodować spadek wydajności lub brak możliwości otwarcia pliku i przekroczenie pojemności pamięci.
Slajd 13	 Spróbujmy na innej tabeli: jak powinna wyglądać ta tabela w 1NF?	

Slajd 14

Rozwiązaniem problemu jest relacja jeden-do-wielu, w której nie należy strony jeden i strony wielu umieszczać w tej samej tabeli. Zamiast tego, należy utworzyć inną tabelę w pierwszej postaci normalnej, eliminując powtarzające się grupy (NrKlasy), tak jak to przedstawiono na slajdzie.

Slajd 15

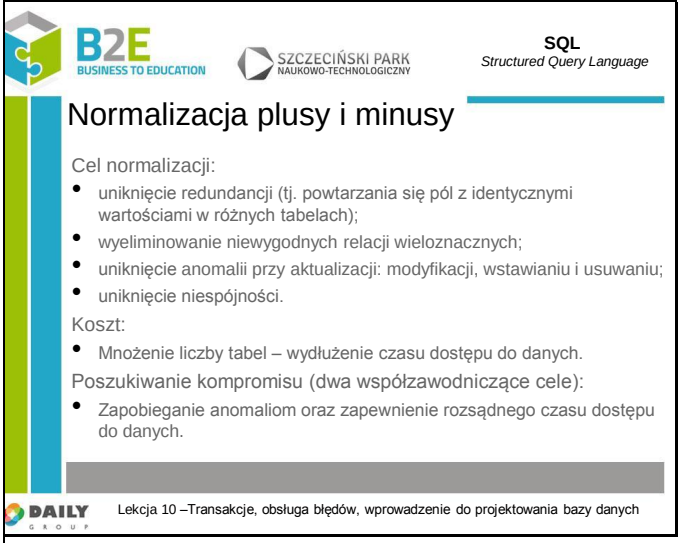


W poprzedniej tabeli dla każdego pola NrStudenta istnieje wiele wartości w polach NrKlasy. Pole NrKlasy nie jest czynnościowo zależne od pola NrStudenta (klucz podstawowy), dlatego ta relacja nie znajduje się w drugiej postaci normalnej.
Drugą postać normalną przedstawiono na podstawie następujących dwóch tabel: Pierwsza postać normalna: studenci i rejestracja.

Slajd 16

W ostatnim przykładzie pole Pok-Dor (numer pokoju doradcy) jest czynnościowo zależne od atrybutu Doradca. Rozwiązaniem jest przeniesienie tego atrybutu z tabeli Studenci do tabeli Wydział, tak jak to przedstawiono na slajdzie.

Slajd 17

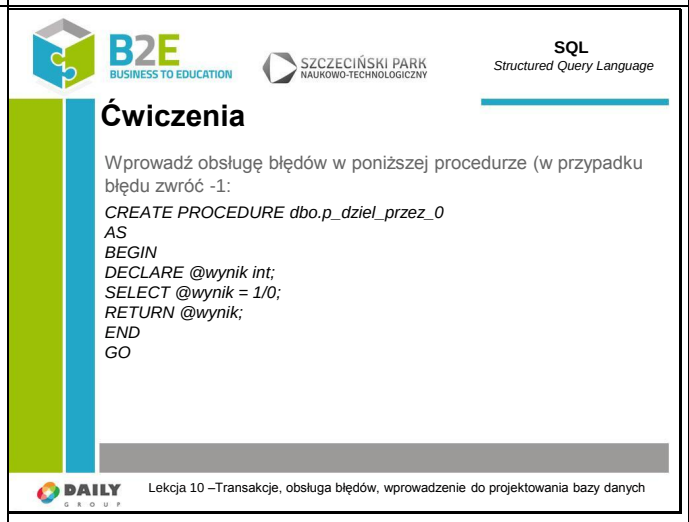


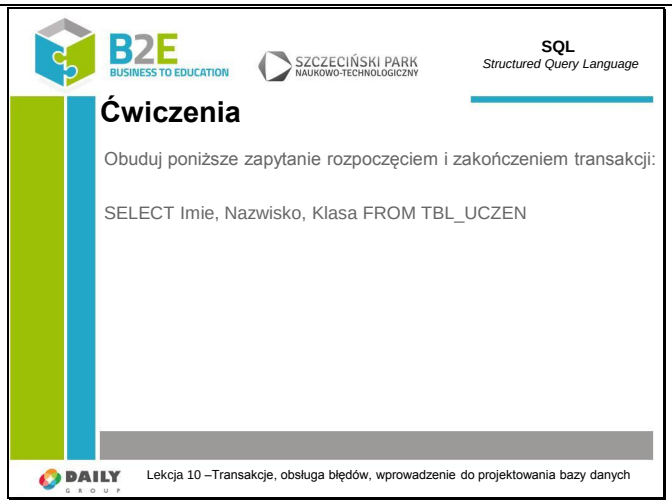
Normalizacja ma swoje wady i zalety. W praktyce podczas projektowania baz danych najczęściej szukamy kompromisu.

Slajd 18

Np.

```
CREATE PROCEDURE podziel1
(
@dzielna float,
@dzielnik float
)
AS
IF @dzielnik = 0
RAISERROR('Dzielenie przez ZERO!',14,1)
ELSE
SELECT @dzielna / @dzielnik
END
```

Slajd 19		
Slajd 20		CREATE PROCEDURE dbo.p_dziel_przez_0 AS BEGIN BEGIN TRY DECLARE @wynik int; SELECT @wynik = 1/0; RETURN @wynik; END TRY BEGIN CATCH RETURN -1; END CATCH END GO
Slajd 21		CREATE PROCEDURE dbo.p_dziel_przez_0 AS BEGIN BEGIN TRY DECLARE @wynik int; SELECT @wynik = 1/0; RETURN @wynik; END TRY BEGIN CATCH SELECT -1, ERROR_NUMBER() AS ErrorNumber RETURN -1; END CATCH END GO

Slajd 22		SELECT @@ERROR
Slajd 23		BEGIN TRANSACTION MojaTransakcja GO SELECT Imie, Nazwisko, Klasa FROM TBL_UCZEN GO COMMIT TRANSACTION MojaTransakcja GO
Slajd 24		

Slajd 25

Tabela podsumowująca poznane wyrażenia

9.10.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 18 - 24, będzie polegało na wprowadzeniu obsługi błędów w procedurze składowanej oraz na normalizacji przykładowej tabeli.

9.10.4 Opis założonych osiągnięć ucznia

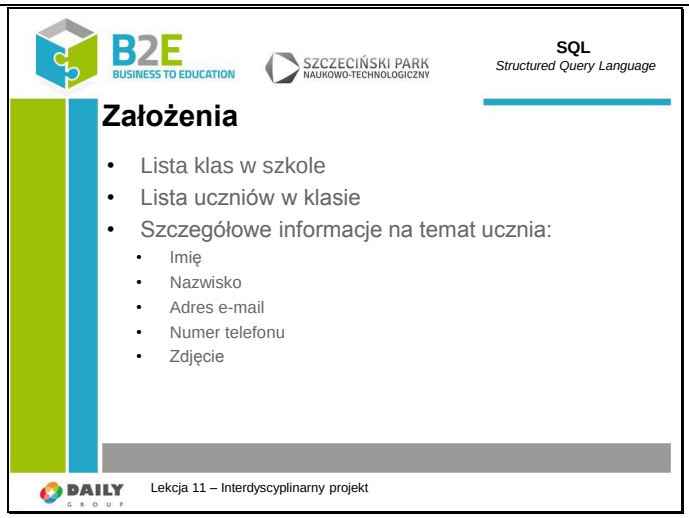
Po tej lekcji uczniowie będą mogli podjąć próbę zaprojektowania bazy danych. Bloki kodu przez nich tworzone będą zawierały obsługę błędów, będą stosować transakcyjne podejście w programowaniu baz danych.

9.11 Lekcja 11 – Interdyscyplinarny projekt książki teleadresowej

9.11.1 Cel lekcji

Celem lekcji jest zaprojektowanie bazy danych do użycia w na stronie WWW oraz w aplikacji mobilnej z funkcjonalnością książki teleadresowej kolegów i koleżanek ze szkoły lub klasy. W tej lekcji uczniowie dowiedzą się jak praktycznie wykorzystać wiedzę zdobytą w tym module na potrzeby pierwszego projektu informatycznego składającego się z elementów wykorzystujących cztery języki programowania: SQL, PHP, JavaScript i JAVA.

9.11.2 Treść - slajdy z opisem

Slajd 1		Proces powstawania książki teleadresowej kolegów/koleżanek ze szkoły. W module SQL pokażemy jak zaprojektować bazę danych, na PHP i JavaScript, jak stworzyć stronę WWW do dodawania kolegów i koleżanek z klasy do takiej książki, a w module JAVA nauczymy się, jak zaprogramować prostą aplikację na telefony z systemem operacyjnym Android do przeszukiwania danych o uczniach z różnych klas.
Slajd 2		Założenia naszej aplikacji – chcemy aby aplikacja umożliwiała prezentację listy klas w naszej szkole. Tak abyśmy mogli swobodnie przeglądać uczniów w każdej z klas. Po wybraniu szukanego ucznia chcielibyśmy zaprezentować użytkownikom naszej aplikacji (mobilnej lub webowej) podstawowe informacje dotyczące ucznia: imię, nazwisko, adres e-mail, numer telefonu oraz zdjęcie.
Slajd 3		W tym celu potrzebne nam będą dwie tabele, które zgodnie z najlepszymi praktykami oznaczymy prefixem „tbl_”: tbl_uczen oraz tbl_klasa.

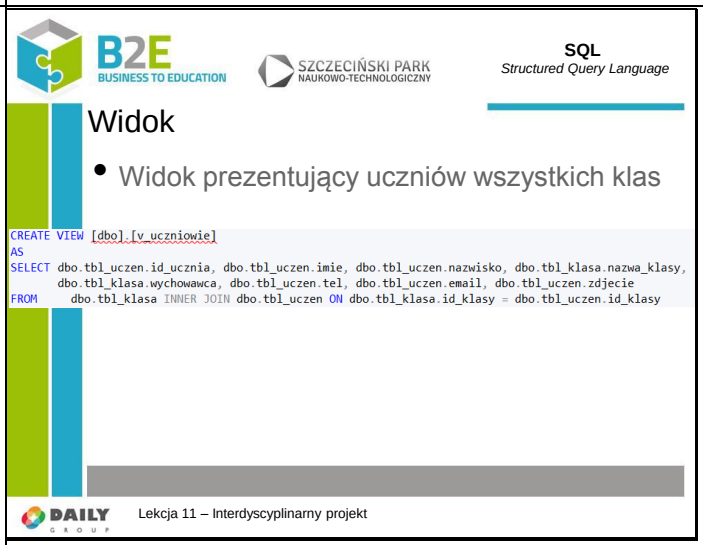
Slajd 4

Tworzymy relacyjną bazę danych. Nawet przy dwóch tabelach taką relację należy utworzyć. Kluczem głównym tabeli tbl_uczen będzie id_ucznia, kluczem głównym tabeli tbl_klasa – id_klasy. Kluczem obcym tabeli tbl_uczen będzie id_klasy.

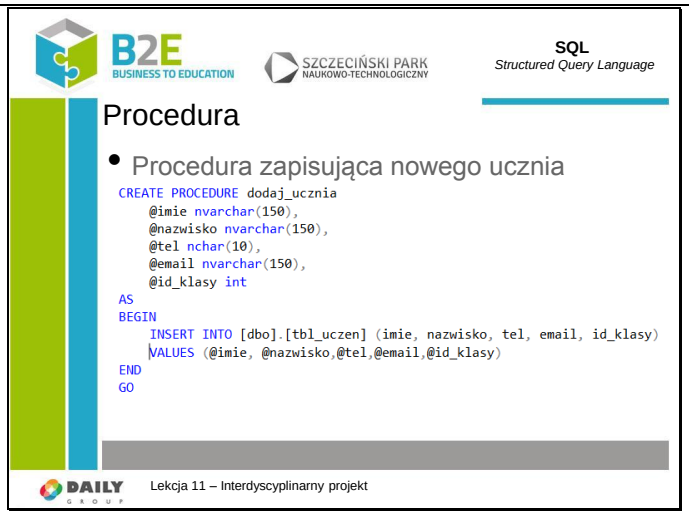
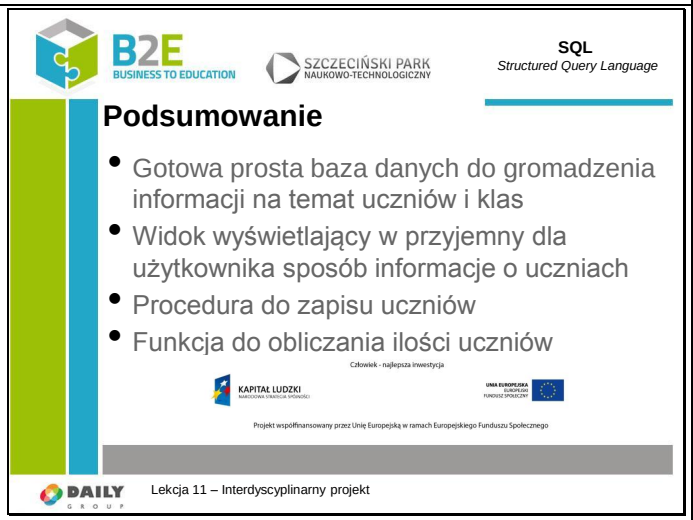
Slajd 5

Przygotujmy skrypt tworzący bazę danych oraz obie tabele, wraz z kluczami i relacją.

Slajd 6



Na potrzeby czytelnej dla użytkownika prezentacji listy uczniów z przyporządkowaniem do klasy – utwórzmy właściwy widok.

Slajd 7		Dobrze jest nie wykonywać operacji zapisu z kodu PHP przez bezpośrednie napisanie kwerendy INSERT – utwórzmy więc procedurę do zapisu nowego ucznia.
Slajd 8		Być może jako programistom przyda nam się funkcja zwracająca ilość uczniów w klasie. Utwórzmy taką funkcję.
Slajd 9		Podsumowanie utworzonych obiektów bazodanowych.

9.11.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie przygotują dla siebie projekt bazy danych do wykorzystania w ich pierwszym projekcie informatycznym – książki teleadresowej kolegów i koleżanek z klasy i ze szkoły dostępnej poprzez WWW oraz aplikację mobilną.