

Moduł szkoleniowy SQL Structured Query Language

poziom podstawowy 15 godzinny

Spis treści

1	Metryka dokumentu.....	5
2	Cel.....	6
3	Opis sposobu realizacji celów.....	6
4	Treści kształcenia.....	6
5	Opis założonych osiągnięć ucznia.....	6
6	Sposoby osiągania celów.....	6
6.1	Przykładowe tematy projektu:	7
7	Propozycje kryteriów oceny i metod sprawdzania osiągnięć ucznia	7
8	Test końcowy sprawdzający wiedzę.....	7
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	10
9.1.3	Ćwiczenia.....	18
9.1.4	Opis założonych osiągnięć ucznia.....	18
9.2	Lekcja 2 - Języki programowania serwerów SQL, środowisko programowania w języku SQL, wprowadzenie do języka SQL.....	19
9.2.1	Cel lekcji.....	19
9.2.2	Treść - slajdy z opisem	19
9.2.3	Ćwiczenia.....	26
9.2.4	Opis założonych osiągnięć ucznia.....	26
9.3	Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych.....	26
9.3.1	Cel lekcji.....	26
9.3.2	Treść - slajdy z opisem	26
9.3.3	Ćwiczenia.....	35
9.3.4	Opis założonych osiągnięć ucznia.....	35
9.4	Lekcja 4 – DML.....	35
9.4.1	Cel lekcji.....	35
9.4.2	Treść - slajdy z opisem	36
9.4.3	Ćwiczenia.....	42

9.4.4	Opis założonych osiągnięć ucznia	42
9.5	Lekcja 5 - DDL	42
9.5.1	Cel lekcji.....	42
9.5.2	Treść - slajdy z opisem	42
9.5.3	Ćwiczenia	50
9.5.4	Opis założonych osiągnięć ucznia	50
9.6	Lekcja 6 - DCL, uprawnienia, użytkownicy.....	50
9.6.1	Cel lekcji.....	50
9.6.2	Treść - slajdy z opisem	50
9.6.3	Ćwiczenia	59
9.6.4	Opis założonych osiągnięć ucznia	59
9.7	Lekcja 7 - Widoki i Funkcje	59
9.7.1	Cel lekcji.....	59
9.7.2	Treść - slajdy z opisem	60
9.7.3	Ćwiczenia	67
9.7.4	Opis założonych osiągnięć ucznia	67
9.8	Lekcja 8 - Procedury Triggery	67
9.8.1	Cel lekcji.....	67
9.8.2	Treść - slajdy z opisem	68
9.8.3	Ćwiczenia	75
9.8.4	Opis założonych osiągnięć ucznia	75
9.9	Lekcja 9 - Indeksy, constrainty	75
9.9.1	Cel lekcji.....	75
9.9.2	Treść - slajdy z opisem	75
9.9.3	Ćwiczenia	85
9.9.4	Opis założonych osiągnięć ucznia	85
9.10	Lekcja 10 - Transakcje, zatwierdzanie zmian, obsługa błędów, wprowadzenie do projektowania bazy danych.....	85
9.10.1	Cel lekcji.....	85
9.10.2	Treść - slajdy z opisem	85
9.10.3	Ćwiczenia	92
9.10.4	Opis założonych osiągnięć ucznia	92



1 Metryka dokumentu

Szczeciński Park Naukowo Technologiczny			
Dokument	Moduł Szkoleniowy SQL – Structured Query Language		
Krótki opis dokumentu	Moduł szkoleniowy kursu SQL zawierający opis celów, treści poszczególnych lekcji, ćwiczeń oraz test końcowy sprawdzający wiedzę		
Data druku	12.02.2013	Liczba stron	92
Nazwa pliku	SQL - Moduł szkoleniowy.docx	Status	roboczy

Historia zmian

Nr wersji	Data	Opis	Działanie (*)	Autorzy
0.1	01.02.2013	Utworzenie nowego dokumentu	N	Michał Galas, Daily Group Sp. z o.o.
1.0	12.02.2013	Wersja gotowa do publikacji	Z	Michał Galas, Daily Group Sp. z o.o.

(*) Działanie: N-Nowy, Z-Zmiana, W-Weryfikacja

Lista dystrybucyjna

Imię i nazwisko / Rola	Organizacja

Zgłoszono do odbioru

Imię i nazwisko		Data:		Podpis:	
Imię i nazwisko	Michał Galas	Data:	12.02.2013	Podpis:	

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.

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

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 Server 2008.

6 Sposoby osiągnięcia 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

7 Propozycje kryteriów oceny i metod 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, tytuł, rok, id_reżyser, producent)

Reżyser (id, imię, nazwisko, rok_urodzenia, narodowosc)

W wyniku wykonania zapytania:

```
SELECT tytuł, rok FROM Film WHERE (rok != 1984 OR rok != 2011) AND
```

```
id_reżyser = (SELECT id FROM Reżyser 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)

Moduł szkoleniowy SQL - Structured Query Language str. 8

Człowiek - najlepsza inwestycja



- d. INSERT INTO Osoby ('Olsen') INTO Nazwisko
8. Jakie zapytanie zwróci wszystkie rekordy z tabeli Osoby, których nazwisko zaczyna się na literę "a"?
- SELECT \* FROM Osoby WHERE Nazwisko='a'
  - SELECT \* FROM Osoby WHERE Nazwisko ='%a%'
  - SELECT \* FROM Osoby WHERE Nazwisko LIKE 'a%'**
  - SELECT \* FROM Osoby WHERE Nazwisko LIKE '%a'
9. Klauzula TOP
- Musi być umieszczona na końcu zapytania.
  - Służy do wyboru rekordów o największej wartości we wskazanej kolumnie.
  - Jest niedostępna w systemie MSSQL.
  - Pozwala ograniczyć liczbę wyświetlanych wyników.**
10. Złączenia tabel w języku SQL
- Są realizowane za pomocą instrukcji MERGE
  - Powodują trwałe zmiany w strukturze bazy danych
  - Można wykonywać na dowolnej liczbie tabel (większej niż 2)**
  - 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 |

- Tabela nie jest znormalizowana**
  - W pierwszej postaci normalnej
  - W drugiej postaci normalnej
  - W trzeciej postaci normalnej
12. UPDATE jest częścią języka
- DDL
  - DML**
  - DQL
  - Żadnego z powyższych
13. Która z poniższych definicji widoku jest prawidłowa?
- CREATE VIEW Widok As SELECT imie, nazwisko, pesel FROM pracownicy**
  - CREATE VIEW Widok SELECT \* FROM pracownicy
  - CREATE VIEW Widok FROM SELECT \* FROM pracownicy
  - CREATE VIEW Widok As imie, nazwisko, pesel FROM pracownicy
14. Do czego służy polecenie:
- SELECT imie, nazwisko, pesel INTO pracownicy FROM klienci
- Zapytanie wyświetla imię, nazwisko i pesel z tabeli pracownicy
  - Za pytanie wyświetla imię, nazwisko i pesel z tabeli klienci
  - Tworzy tabelę klienci**
  - 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

**Tradycyjne zbiory danych**

- Kartoteki
- Teczki
- Segregatory



Lekcja 1 - Bazy danych, podstawowe pojęcia

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

**Definicja bazy danych**

- zbiór informacji wraz z możliwością łatwego dostępu oraz ich zmiany

Lekcja 1 - Bazy danych, podstawowe pojęcia

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

**Elektroniczne bazy danych**  
**Pliki w programie**

- Dane dostępne tylko dla programu
- Problem z wieloma operacjami jednocześnie
- Brak możliwości skorzystania z danych przez inny program



Lekcja 1 - Bazy danych, podstawowe pojęcia


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”.

Slajd 6

**Elektroniczne bazy danych**  
**Pliki dostępne dla wielu programów**

- Pliki na serwerze np. FTP
- Wiele aplikacji korzystających z plików
- Zarządzaniem dostępem



Lekcja 1 - Bazy danych, podstawowe pojęcia

Rysunek na slajdzie przedstawia sytuację, gdzie program korzysta z pliku tekstowego, który jest jego bazą. Wiąże się to jednak z ograniczeniami (wymienione na slajdzie). 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 zarządzania dostępem do danych.



Slajd 7

**System Zarządzania Bazą Danych**

- „Coś” pomiędzy danymi a programem

Lekcja 1 - Bazy danych, podstawowe pojęcia

Brakującym elementem jest System Zarządzania Bazą Danych, który wraz z danymi stanowi BAZĘ DANYCH

Slajd 8

**System Zarządzania Bazą Danych**

- (Database Management System, DBMS) – oprogramowanie bądź system informatyczny służący do zarządzania bazą danych.

Lekcja 1 - Bazy danych, podstawowe pojęcia

Definicja SZDB

Slajd 9

**System Zarządzania Bazą Danych**

- łatwy dostęp do danych (dane są zorganizowane w struktury)
- jednoczesny dostęp do danych przez wielu użytkowników (architektura klient-serwer, transakcje)
- zmianę istniejących danych (dodawanie, usuwanie, edycja)
- kontrolę dostępu do danych (autoryzacja, ograniczenie dostępu)
- bezpieczeństwo danych (archiwizacja, logowanie)

Lekcja 1 - Bazy danych, podstawowe pojęcia

Cechy, określające system zarządzania bazą danych

Slajd 10

**System Zarządzania Bazą Danych**

- Przykłady:
  - Oracle
  - IBM DB2
  - Microsoft SQL Server
  - Firebird
  - MySQL
  - PostgreSQL
  - Sybase

Lekcja 1 - Bazy danych, podstawowe pojęcia

Pytanie do uczniów: czy znacie przykłady SZBD?



Slajd 11

**Relacyjny System Bazy Danych**

SQL  
Structured Query Language

| Dział     |                  |                        | Projekt       |                    |               |
|-----------|------------------|------------------------|---------------|--------------------|---------------|
| id_działu | nazwa            | adres                  | p_id_projektu | nazwa              | termin_zdania |
| 1         | Marketing        | Warszawa, Śleszcza 3   | p1            | Uprędkie sprzetu   | 10.01.2009    |
| 2         | Kierownictwo     | Warszawa, Śleszcza 3   | p1            | Program magazynowy | 1.08.2009     |
| 3         | Administracja IT | Warszawa, Śleszcza 3a  | p2            | Statystyka         | 20.02.2009    |
| 4         | Programistka     | Bydgoszcz, Kuwalska 10 |               |                    |               |

| Pracownicy |                 |                               |
|------------|-----------------|-------------------------------|
| id_insa    | nazwisko        | id_działu_pierwszego_projektu |
| 1          | Jan Nowakowski  | 2                             |
| 2          | Adam Nowak      | 3                             |
| 3          | Maria Kozłowska | 4                             |
| 4          | Anna Rys        | 3                             |
| 5          | Paweł Lis       | 4                             |
| 6          | Paweł Lis       | 4                             |
| 7          | Jan Nowakowski  | 1                             |
| 8          | Adam Nowak      | 4                             |

Lekcja 1 - Bazy danych, podstawowe pojęcia

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

**Relacyjny System Bazy Danych**

SQL  
Structured Query Language

- W bazach relacyjnych wiele tablic danych może współpracować ze sobą
- Wszystkie dane w bazie relacyjnej przedstawiane są w formie dwuwymiarowych tabel
- Wszystkie wartości danych oparte są na prostych typach danych
- Możliwe jest porównywanie wartości z różnych kolumn, tabel
- Wszystkie operacje wykonywane są w oparciu o algebrę relacji
- Przez brak możliwości identyfikacji wiersza przez jego pozycję niezbędna jest jedna lub więcej kolumn niepowtarzalnych



Lekcja 1 - Bazy danych, podstawowe pojęcia

W bazach relacyjnych wiele tablic danych może współpracować 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 prostych 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

SQL  
Structured Query Language

---

### Podstawowe pojęcia

- Tabela

| Pracownicy |        |            |           |        |         |
|------------|--------|------------|-----------|--------|---------|
| id         | imie   | nazwisko   | id_dzialu | pensja | projekt |
| 1          | Jan    | Kowalski   | 2         | 2500   |         |
| 2          | Adam   | Nowak      | 3         | 3100   | it1     |
| 3          | Halina | Szańska    | 4         | 1800   |         |
| 4          | Anna   | Ryś        | 3         | 3000   |         |
| 5          | Piotr  | Lis        | 4         | 2000   | p1      |
| 6          | Paweł  | Lis        | 4         | 1800   | p2      |
| 7          | Jan    | Nowikowski | 1         | 2200   | it1     |
| 8          | Adam   | Kot        | 4         | 1900   | p2      |

Lekcja 1 - Bazy danych, podstawowe pojęcia

Tabela składa się z wierszy i kolumn. Wiersze w tabeli są przechowywane w dowolnym porządku. Dla każdego wiersza każda z kolumn posiada jedno pole z wartością. Wszystkie wartości w kolumnie są tego samego typu.

Slajd 14




SQL  
Structured Query Language

---

### Podstawowe pojęcia


- Dana

| Pracownicy |        |            |           |        |         |
|------------|--------|------------|-----------|--------|---------|
| id         | imie   | nazwisko   | id_dzialu | pensja | projekt |
| 1          | Jan    | Kowalski   | 2         | 2500   |         |
| 2          | Adam   | Nowak      | 3         | 3100   | it1     |
| 3          | Halina | Szańska    | 4         | 1800   |         |
| 4          | Anna   | Ryś        | 3         | 3000   |         |
| 5          | Piotr  | Lis        | 4         | 2000   | p1      |
| 6          | Paweł  | Lis        | 4         | 1800   | p2      |
| 7          | Jan    | Nowikowski | 1         | 2200   | it1     |
| 8          | Adam   | Kot        | 4         | 1900   | p2      |

Lekcja 1 - Bazy danych, podstawowe pojęcia

Dana to pojedynczy wpis w tabeli – jak komórka w MS Excel. Często stosowanym pojęciem jest atrybut.

Slajd 15




SQL  
Structured Query Language

---

### Podstawowe pojęcia

- Wiersz

| Pracownicy |        |            |           |        |         |
|------------|--------|------------|-----------|--------|---------|
| id         | imie   | nazwisko   | id_dzialu | pensja | projekt |
| 1          | Jan    | Kowalski   | 2         | 2500   |         |
| 2          | Adam   | Nowak      | 3         | 3100   | it1     |
| 3          | Halina | Szańska    | 4         | 1800   |         |
| 4          | Anna   | Ryś        | 3         | 3000   |         |
| 5          | Piotr  | Lis        | 4         | 2000   | p1      |
| 6          | Paweł  | Lis        | 4         | 1800   | p2      |
| 7          | Jan    | Nowikowski | 1         | 2200   | it1     |
| 8          | Adam   | Kot        | 4         | 1900   | p2      |

Lekcja 1 - Bazy danych, podstawowe pojęcia

Pojedynczy wiersz tabeli nazywany jest rekordem (lub encją) i stanowi najczęściej zbiór danych o pojedynczym obiekcie (ew. grupie obiektów).



Slajd 16

**Podstawowe pojęcia**

- Kolumna

| Pracownicy |        | id_działu  |   | pensja |  | projekt |     |
|------------|--------|------------|---|--------|--|---------|-----|
| id         | imię   | nazwisko   |   |        |  |         |     |
| 1          | Jan    | Kowalski   | 2 | 2600   |  |         |     |
| 2          | Adam   | Nowak      | 3 | 3100   |  |         | it1 |
| 3          | Halina | Szańska    | 4 | 1800   |  |         |     |
| 4          | Anna   | Ryś        | 3 | 3000   |  |         |     |
| 5          | Piotr  | Lis        | 4 | 2000   |  | p1      |     |
| 6          | Paweł  | Lis        | 4 | 1800   |  | p2      |     |
| 7          | Jan    | Nowikowski | 1 | 2200   |  | it1     |     |
| 8          | Adam   | Kot        | 4 | 1900   |  | p2      |     |

Lekcja 1 - Bazy danych, podstawowe pojęcia

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

**Podstawowe pojęcia**

- Relacja

Lekcja 1 - Bazy danych, podstawowe pojęcia

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

**Podstawowe pojęcia**

- Rodzaje relacji

Lekcja 1 - Bazy danych, podstawowe pojęcia

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 łączy), której klucz podstawowy składa się z dwóch pól – 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



**Podstawowe pojęcia**

- Klucz podstawowy (główny) - PK**

Lekcja 1 - Bazy danych, podstawowe pojęcia

Każda relacja (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




**Podstawowe pojęcia**

- Klucz obcy - FK**

Lekcja 1 - Bazy danych, podstawowe pojęcia

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



**Podstawowe pojęcia**

- Sortowanie**

Lekcja 1 - Bazy danych, podstawowe pojęcia

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





Slajd 22

SQL  
Structured Query Language

**Podstawowe pojęcia**

- Zapytanie

Lekcja 1 - Bazy danych, podstawowe pojęcia

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

Slajd 23

SQL  
Structured Query Language

**Interakcje z bazą danych**

- Język SQL
  - Jedyny sposób interakcji z bazą danych
  - Język deklaratywny
  - Ustandaryzowany
    - Różni producenci stosują ten sam standard

```
select nazwisko, etat, placa
from pracownicy
where idzesp=30
and etat='kierownik'
```

Lekcja 1 - Bazy danych, podstawowe pojęcia

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 standaryzacją zajmuje się specjalny międzynarodowy komitet, w skład którego wchodzi 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.

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.

Próba zaprojektowania bazy danych wypożyczalni filmów razem z uczniami, użycie pojęć zdefiniowanych w pierwszej części lekcji

### Przykład rozwiązania ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 25: Próba wspólnego zaprojektowania bazy danych wypożyczalni filmów DVD, wykorzystanie pojęć przedstawionych w czasie lekcji.

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

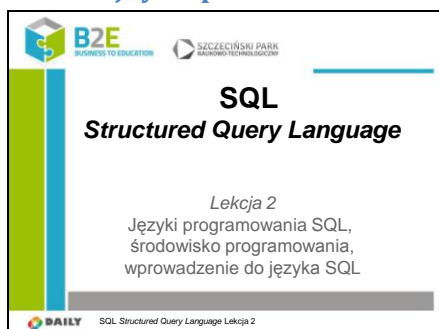
### 9.2.1 Cel lekcji

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 wyjaśnimy sobie czym jest baza danych oraz objaśnimy podstawowe pojęcia dotyczące baz danych.

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

**Środowisko programowania**

- Microsoft SQL Server Management Studio
- Wersja bezpłatna – Express
- Toad for SQL Server Freeware

SQL Structured Query Language

SQL Server Management Studio Express

Toad World

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

Środowiska programowania TSQL. Microsoft oferuje wraz z bazą danych SQL Server 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

**MS SQL Management Studio Express**

- Do pobrania ze strony Microsoft
- <http://www.microsoft.com/en-us/download/details.aspx?id=29062>

Microsoft SQL Server 2012 Express

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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\SQLEXPRTW\_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

SQL Structured Query Language

SQL Server 2008 R2

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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

**MS SQL Management Studio Express**

SQL Structured Query Language

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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



Slajd 8

**Wprowadzenie do języka SQL**

- SQL to język dostępu do bazy danych
- Grupy poleceń języka:
  - DDL - język definiowania danych (Data Definition Language)
  - DML - Język manipulowania danych (Data Manipulation Language)
  - DCL - Język sterowania danych (Data Control Language)
  - Język zapytań (Query Language)

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

- 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

**Wprowadzenie do języka SQL**

- Polecenie SQL może być zapisane:
  - w jednym bądź wielu wierszach
  - dużymi lub małymi literami
- Polecenia powinny być zakończone średnikiem

```
SELECT * FROM pracownicy;
```

```
select * from PRACOWNICY;
```

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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

**Wyświetlanie elementów (projekcja)**

- Wybór wartości z tabeli, np.

```
SELECT imie, nazwisko, pesel FROM pracownicy;
```

- Operatory arytmetyczne, np. +, -, \*, /

```
SELECT nazwisko, placa_zasadnicza*3, premia + 300 FROM pracownicy;
```

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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 dodawanie, odejmowanie, mnożenie i dzielenie.

Slajd 11

**Aliasy**

- Alternatywna nazwa atrybutu
- Można stosować słowo AS

```
SELECT nazwisko, placa_zasadnicza*1.2 AS nowa_placa, premia + 300 AS premia_swiateczna FROM pracownicy;
```

|   | Result   | Messages   |                   |
|---|----------|------------|-------------------|
|   | nazwisko | nowa_placa | premia_swiateczna |
| 1 | Kowalski | 2760.000   | 550.00            |
| 2 | Nowak    | 3240.000   | 400.00            |

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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

**NULL**

- Wartość pusta w tabeli
- Wynikiem jakiegokolwiek operacji z NULL jest NULL

```
SELECT * FROM pracownicy;
```

| stanowisko  | imie       | nazwisko | pesel            | stanowisko  |
|-------------|------------|----------|------------------|-------------|
| Manager     | Jan        | Kowalski | 220200 320310027 | Manager     |
| Specjalista | Kamil      | Nowak    | 150200 880102007 | Specjalista |
| Specjalista | Przemysław | Nowak    | 270200 890202008 | Specjalista |

Lekcja 2 - Język programowania SQL, środowisko, wprowadzenie do języka SQL

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.

Slajd 13

**ISNULL**

- ISNULL(kolumna, wartość\_w\_przypadku\_NULL)
- Możliwość zmiany wartości pustej na inną, np.

```
SELECT nazwisko,
 plan_w_zamieszkiwaniu*1.2 AS nowa_placa,
 ISNULL(premia,0) + 300 premia_wzrostnosc
FROM pracownicy;
```

| stanowisko  | imie       | nazwisko | pesel            | stanowisko  |
|-------------|------------|----------|------------------|-------------|
| Manager     | Jan        | Kowalski | 220200 320310027 | Manager     |
| Specjalista | Kamil      | Nowak    | 150200 880102007 | Specjalista |
| Specjalista | Przemysław | Nowak    | 270200 890202008 | Specjalista |

Lekcja 2 - Język programowania SQL, środowisko, wprowadzenie do języka SQL

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

Slajd 14

**Porządkowanie elementów**

- Klauzula ORDER BY
- Słowa kluczowe DESC i ASC
- ORDER BY występuje zawsze na końcu zapytania

```
SELECT imie, nazwisko, pesel FROM pracownicy
ORDER BY nazwisko DESC, imie ASC;
```

Lekcja 2 - Język programowania SQL, środowisko, wprowadzenie do języka SQL

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

**Eliminowanie duplikatów**

- Polecenie DISTINCT

```
SELECT DISTINCT stanowisko FROM pracownicy;
```

| stanowisko  |
|-------------|
| Manager     |
| Specjalista |

Lekcja 2 - Język programowania SQL, środowisko, wprowadzenie do języka SQL

Słowo kluczowe DISTINCT musi występować zaraz po słowie kluczowym SELECT. *SELECT stanowisko FROM pracownicy* Takie zapytanie wyświetli wszystkie stanowiska obejmowane wśród pracowników. Jeżeli pojawiają 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

**Selekcja wybranych danych**

- Klauzula WHERE
- Ogólna składnia:  
SELECT kolumna1, kolumna2, kolumna3  
FROM tabela  
WHERE kolumna **operator** wartość
- Operatory: =, !=, >, >=, <, <=

```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE placza_zasadnicza > 2000
```

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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

**Selekcja wybranych danych**

- Operatory BETWEEN ... AND ...

```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE placza_zasadnicza BETWEEN 2000 AND 3000
```

- IN

```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE stanowisko IN ('Specjalista', 'Manager')
```

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

Operator BETWEEN AND umożliwia wybór wartości z danego zakresu, operator IN wybór wartości zawierających się w zdefiniowanym zbiorze

## Slajd 18

**Selekcja wybranych danych**

- LIKE

```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE nazwisko LIKE 'M%';
```

```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE nazwisko LIKE '%ski';
```

- IS NULL

```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE premia IS NULL;
```

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

Operator LIKE umożliwia wybór wartości które zawierają pewien ciąg znaków, np. zaczynają się od litery „M”, lub kończą na „ski”.

## Slajd 19

**Selekcja wybranych danych**

- Negacja operatorów SQL
- NOT BETWEEN ... AND ...
- NOT IN
- NOT LIKE
- IS NOT NULL




```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE stanowisko NOT IN ('Specjalista', 'Manager');
```

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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

  
Structured Query Language

### Selekcja wybranych danych

- Łączenie warunków
  - AND
  - OR

```
SELECT imie, nazwisko, pesel FROM pracownicy
WHERE stanowisko NOT IN ('Specjalista', 'Manager')
AND placazasadnicza > 2000;
```

 Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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

**B2E**  
BUSINESS TO EDUCATION

SZCZECIŃSKI PARK  
NAUKOWO-TECHNOLOGICZNY

**SQL**

Structured Query Language

## Ćwiczenia

|   | nazwisko   | imie      | placa_zasadnicza | premia | pesel       | stanowisko  |
|---|------------|-----------|------------------|--------|-------------|-------------|
| 1 | Kowalski   | Jen       | 2300.00          | 250.00 | 82091104357 | Manager     |
| 2 | Nowak      | Karol     | 2700.00          | 100.00 | 89010123897 | Specjalista |
| 3 | Proszka    | Marczena  | 2700.00          | NULL   | 89121203456 | Specjalista |
| 4 | Burzych    | Pawel     | 1900.00          | 500.00 | 78032309123 | Specjalista |
| 5 | Makłowicz  | Marek     | 2000.00          | NULL   | 54013112345 | Specjalista |
| 6 | Naramowicz | Magdalena | 2100.00          | 200.00 | 77121312098 | Specjalista |
| 7 | Witos      | Jacek     | 3000.00          | 500.00 | 69100967234 | Manager     |

**DAILY**  
PROGRAMING

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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

**SQL**  
Structured Query Language

### Podsumowanie

```

SELECT [DISTINCT] (* , kolumna [AS alias], ...)
FROM tabela WHERE warunek [AND | OR warunek ...]
ORDER BY (kolumna, wyrażenie) [ASC | DESC];

```

|          |                                                         |
|----------|---------------------------------------------------------|
| SELECT   | Wybiera dane z tabeli                                   |
| alias    | Można stosować tylko do kolumn i wyrażen (nie do *)     |
| *        | Oznacza wszystkie kolumny                               |
| DISTINCT | Eliminuje duplikaty ze zbioru wyników                   |
| FROM     | 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 określania kryterium sortowania                |
| ASC      | Rosnący porządek sortowania                             |
| DESC     | Malejący porządek sortowania                            |

Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

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.

## 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

**SQL**  
*Structured Query Language*

*Lekcja 3*  
Złożone zapytania,  
grupowanie, agregacja danych

SQL Structured Query Language Lekcja 3

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

**Przypomnienie**

- Polecenie SELECT
- WHERE
- ORDER BY
- DISTINCT
- Alias

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Powtórzenie materiału z poprzedniej lekcji.

Slajd 3

**Ograniczanie wyników**

```
SELECT TOP 3 imie, nazwisko, pesel FROM pracownicy
```

- Polecenie TOP:
  - Używana zaraz po SELECT
  - Wartość liczbowa, np. TOP 3
  - Wartość procentowa, np. TOP 10 PERCENT

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Ograniczanie wyników w SQL realizowane jest słowem TOP. Możliwe jest ograniczenie ilościowe i procentowe.

Slajd 4

**Komentarze w SQL**

- "--" – pojedynczy wiersz
- "/\*" i "\*/" – cała sekcja

```
SELECT TOP 3
 imie,
 --nazwisko,
 pesel
FROM pracownicy

SELECT TOP 3
 /*imie,
 nazwisko, */
 pesel
FROM pracownicy
```

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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**

| id_pracownika | imie      | nazwisko   | pesel      | id_miejsca | miasto    |
|---------------|-----------|------------|------------|------------|-----------|
| 1             | Jan       | Nowak      | 8001010101 | 1          | Warszawa  |
| 2             | Anna      | Kowalska   | 8502020202 | 2          | Kraków    |
| 3             | Michał    | Wiśniewski | 8803030303 | 3          | Łódź      |
| 4             | Magdalena | Świątek    | 9004040404 | 4          | Wrocław   |
| 5             | Adam      | Łach       | 9205050505 | 5          | Gdańsk    |
| 6             | Julia     | Wojcik     | 9506060606 | 6          | Bydgoszcz |
| 7             | Patryk    | Michalski  | 9807070707 | 7          | Katowice  |
| 8             | Wiktoria  | Nowak      | 0108080808 | 8          | Warszawa  |
| 9             | Maciej    | Adamczyk   | 0409090909 | 9          | Warszawa  |
| 10            | Olivia    | Nowak      | 0710101010 | 10         | Warszawa  |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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

**SQL**  
Structured Query Language

### Łączenie tabel

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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

Slajd 7

**SQL**  
Structured Query Language

### Złączenie naturalne

```
SELECT imie, nazwisko, pesel, ulica, numer, miasto
FROM pracownicy, miejsca
```

- Zbiór wszystkich możliwych kombinacji rekordów z obu tabel

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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

**SQL**  
Structured Query Language

### Złączenie naturalne

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

- INNER JOIN ... ON ...

Klauzule ON i USING są różnymi sposobami na podanie warunku złączenia, czyli wskazania wspólnych kolumn łączonych tabel.

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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

Moduł szkoleniowy SQL - Structured Query Language str. 28

Człowiek - najlepsza inwestycja





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 JOIN miejsca
ON pracownicy.nr_miejsca = miejsca.nr_miejsca
```

| nr | imie      | nazwisko   | pesel      | ulica | numer | miasto |
|----|-----------|------------|------------|-------|-------|--------|
| 1  | Jan       | Kowalski   | 8201154527 | Włosa | 13    | Poznań |
| 2  | Karl      | Rosak      | 801012387  | Włosa | 13    | Poznań |
| 3  | Paul      | Burach     | 7632239123 | Włosa | 4/12A | Kraków |
| 4  | Mark      | Makowicz   | 5401112345 | Włosa | 4/12A | Kraków |
| 5  | Magdalena | Nowoswidka | 771212388  | Włosa | 13    | Poznań |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Złączenie naturalne eliminuje z wyniku niepasujące (niepełniające 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

**Złączenia zewnętrzne**

- **OUTER JOIN** nie pomija wyników

**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

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

| nr | imie      | nazwisko   | pesel      | ulica     | numer | miasto   |
|----|-----------|------------|------------|-----------|-------|----------|
| 1  | Jan       | Kowalski   | 8201154527 | Włosa     | 13    | Poznań   |
| 2  | Karl      | Rosak      | 801012387  | Włosa     | 13    | Poznań   |
| 3  | Marina    | Przedkita  | 8912123456 | WŁAŁ      | WŁAŁ  | WŁAŁ     |
| 4  | Paul      | Burach     | 7632239123 | Włosa     | 4/12A | Kraków   |
| 5  | Mark      | Makowicz   | 5401112345 | Włosa     | 4/12A | Kraków   |
| 6  | Magdalena | Nowoswidka | 771212388  | Włosa     | 13    | Poznań   |
| 7  | Janek     | Włosa      | 6910867234 | Gumowidka | 34    | Warszawa |
| 8  | Marcjan   | Makowicz   | 6712123878 | WŁAŁ      | WŁAŁ  | WŁAŁ     |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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.

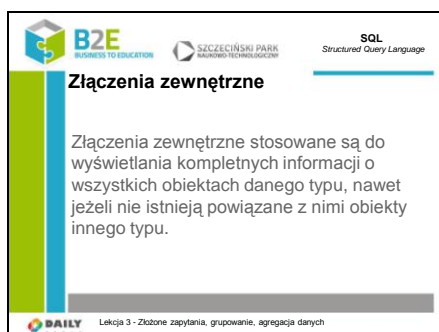
OUTER JOIN:

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



**Złączenia zewnętrzne**

Złączenia zewnętrzne stosowane są do wyświetlania kompletnej informacji o wszystkich obiektach danego typu, nawet jeżeli nie istnieją powiązane z nimi obiekty innego typu.

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Podsumowanie OUTER JOIN

Slajd 12



**Złączenie krzyżowe**

```
SELECT * FROM pracownicy CROSS JOIN miejsca
```

| id_pracownika | id_miejsca | imie | nazwisko | data_urodzenia | stanowisko | id_miejsca | id_pracownika | imie | nazwisko | data_urodzenia | stanowisko |
|---------------|------------|------|----------|----------------|------------|------------|---------------|------|----------|----------------|------------|
| 1             | 1          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 1          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 2          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 2          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 3          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 3          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 4          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 4          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 5          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 5          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 6          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 6          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 7          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 7          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 8          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 8          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 9          | Jan  | Kowalski | 1980-01-01     | Pracownik  | 9          | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |
| 1             | 10         | Jan  | Kowalski | 1980-01-01     | Pracownik  | 10         | 1             | Jan  | Kowalski | 1980-01-01     | Pracownik  |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Wynikiem złączenia krzyżowego jest iloczyn kartezjański łączonych obiektów. (Iloczynem kartezjańskim dwóch zbiorów jest zbiór zawierający wszystkie kombinacje ich elementów. Ponieważ tabele reprezentują zbiory, iloczynem kartezjańskim dwóch tabel jest tabela zawierająca wszystkie możliwe kombinacje wierszy łączonych tabel). W przeciwieństwie do innych typów złączeń w tym wypadku łączone tabele nie muszą mieć wspólnych kolumn. Złączenia tego typu są rzadko stosowane w znormalizowanych bazach danych i służą raczej do generowania danych testowych niż do wybierania danych.





Slajd 13

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### Złączenie tabeli z nią samą

- Złączenie tabeli z nią samą jest jedną z technik języka SQL, odpowiadającą użyciu zmiennych w proceduralnych językach programowania.

**Ważne zasady:**

- Trzeba utworzyć różne aliasy dla łączonej tabeli i w ramach zapytania konsekwentnie odwoływać się do aliasów, a nie do nazwy tabeli.
- Każdy rekord, w którym wartości atrybutu złączenia będą sobie równe, zostanie dodany do wyniku złączenia, co spowoduje powstanie duplikatów rekordów.

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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 14

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### Złączenie tabeli z nią samą

```
SELECT p.imie, p.nazwisko, p.pesel, m.imie, m.nazwisko
FROM pracownicy p INNER JOIN pracownicy m
ON p.przełożony = m.sz_pracownika
```

| imie        | nazwisko     | pesel       | imie  | nazwisko |
|-------------|--------------|-------------|-------|----------|
| Jan         | Kowalski     | 82091104357 | Jacek | Witos    |
| Karol       | Nowak        | 80010123987 | Jan   | Kowalski |
| Marcina     | Przepiórka   | 89121203456 | Jan   | Kowalski |
| Paweł       | Burzyński    | 7803209123  | Jacek | Witos    |
| Marek       | Makłowicz    | 54013112345 | Jacek | Witos    |
| Magdalena   | Naranowiczka | 77121312098 | Jan   | Kowalski |
| Jacek       | Witos        | 69100967234 | Jacek | Witos    |
| Maksymilian | Makłowicz    | 67121209876 | Jacek | Witos    |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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 15

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### Unie

- UNION
  - Umożliwia połączenie dwóch tabel
  - Nazwy kolumn muszą być takie same (można użyć alias)

| imie    | nazwisko  | imie        | pesel |
|---------|-----------|-------------|-------|
| Młociąg | Witkowski | 82041112389 |       |
| Marcin  | Kowalski  | 77122986123 |       |

| sz_pracownika | nazwisko     | imie        | wilaz_pracownika | imie   | pesel       |
|---------------|--------------|-------------|------------------|--------|-------------|
| 1             | Kowalski     | Jan         | 230101           | 230101 | 82091104357 |
| 2             | Nowak        | Karol       | 270101           | 130101 | 80010123987 |
| 3             | Przepiórka   | Marcina     | 270101           | N/A    | 89121203456 |
| 4             | Burzyński    | Paweł       | 180101           | 500101 | 7803209123  |
| 5             | Makłowicz    | Marek       | 200101           | N/A    | 54013112345 |
| 6             | Naranowiczka | Magdalena   | 270101           | 200101 | 77121312098 |
| 7             | Witos        | Jacek       | 200101           | 500101 | 69100967234 |
| 8             | Makłowicz    | Maksymilian | 200101           | N/A    | 67121209876 |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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).

Slajd 16

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### Unie

```
SELECT nazwisko, imie, pesel FROM pracownicy
UNION
SELECT nazwisko, imie, pesel FROM klienci;
```

| nazwisko     | imie        | pesel       |
|--------------|-------------|-------------|
| Burzyński    | Paweł       | 7803209123  |
| Kowalski     | Jan         | 82091104357 |
| Makłowicz    | Marek       | 54013112345 |
| Marcin       | Karłowicz   | 77122986123 |
| Makłowicz    | Maksymilian | 67121209876 |
| Młociąg      | Witkowski   | 82041112389 |
| Naranowiczka | Magdalena   | 77121312098 |
| Nowak        | Karol       | 80010123987 |
| Przepiórka   | Marcina     | 89121203456 |
| Witos        | Jacek       | 69100967234 |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Powyższy przykład pokazuje jak w SQL połączyć dwie tabele razem wynikiem takiego zapytania będzie najpierw cała tabela pracownicy, a poniżej cała tabela klienci



Slajd 17

**Wylączenie**

- EXCEPT – wyświetlenie tylko kolumn z jednej tabeli, które nie znajdują się w drugiej
- Przykład, pokaż wszystkich pracowników, którzy nie są klientami:

```
SELECT nazwisko, imie, pesel FROM pracownicy
EXCEPT
SELECT nazwisko, imie, pesel FROM klienci;
```

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Jeśli chcemy wyszukać wartości tabeli A, które nie znajdują się w tabeli B – korzystamy z polecenia EXCEPT. Konstrukcja jest identyczna jak przy UNION.

Slajd 18

**Funkcje agregujące**

- COUNT - czyli zliczanie ilości zwróconych wierszy z zapytania,
- SUM - wynik z dodawania wartości, ze wszystkich elementów występujących w zapytaniu (każda kolumna osobno),
- AVG - wartość średnia (arytmetyczna), ze wszystkich wartości występujących w zapytaniu (również każda kolumna osobno),
- MIN - wartość najmniejsza ze wszystkich występujących w kolumnie,
- MAX - wartość największa ze wszystkich dostępnych w kolumnie.

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Wprowadzenie do funkcji agregujących. Każda z nich pochodzi z j. angielskiego – w ten sposób można łatwo zapamiętać.

Slajd 19

**Grupowanie GROUP BY**

- GROUP BY:
  - Umożliwia grupowanie wyników względem warunku, np.
  - Pokaż ilość pracowników w podziale na miejsca pracy

```
SELECT COUNT(*) nr_miejsc FROM pracownicy GROUP BY nr_miejsc;
```

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

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 20

**COUNT**

- Zliczanie liczby rekordów
- Przykład liczba pracowników w poszczególnych miastach

```
SELECT miasto, COUNT(imie)
FROM pracownicy JOIN JOIN miast
ON pracownicy.nr_miejsc = miast.nr_miejsc GROUP BY miasto;
```

| miasto | (No column name) |
|--------|------------------|
| 1      | Kraków 2         |
| 2      | Poznań 3         |
| 3      | Warszawa 1       |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Funkcja COUNT liczy wystąpienia bazy danych w tym przypadku dla stycznia koszty wystąpiły dwa razy dla pozostałych miesięcy tylko raz, w klauzuli group by wskazujemy poczym mają być grupowane dane. Jeśli użyli byśmy w powyższym zapytaniu zaraz za SELECT DISTINCT wynik byłby dla stycznia 1 ponieważ DISTINCT nakazuje eliminację identycznych rekordów.





Slajd 21

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### SUM

- SUM – sumuje wartości liczbowe, np. suma pensji w poszczególnych miastach

```
SELECT miasto, SUM(placa_zasadnicza)
FROM pracownicy INNER JOIN miasta
ON pracownicy.nr_miejsc = miasta.nr_miejsc GROUP BY miasto;
```

|   | miasto   | (No column name) |
|---|----------|------------------|
| 1 | Kraków   | 3500.00          |
| 2 | Poznań   | 7100.00          |
| 3 | Warszawa | 3500.00          |

**DAILY** Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Przykład wykorzystania funkcji SUM

Slajd 22

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### AVG

- AVG – wylicza średnią
- Średnia łączna płaca w różnych miastach:

```
SELECT miasto, AVG(placa_zasadnicza + ISNULL(premia, 0))
FROM pracownicy INNER JOIN miasta
ON pracownicy.nr_miejsc = miasta.nr_miejsc GROUP BY miasto;
```

|   | miasto   | (No column name) |
|---|----------|------------------|
| 1 | Kraków   | 2200.000000      |
| 2 | Poznań   | 2550.000000      |
| 3 | Warszawa | 3500.000000      |

**DAILY** Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Przykład wykorzystania funkcji AVG

Slajd 23

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### MIN i MAX

- MIN – to wartość minimalna, a MAX wartość maksymalna

```
SELECT miasto, MIN(placa_zasadnicza + ISNULL(premia, 0)) as Minimum,
MAX(placa_zasadnicza + ISNULL(premia, 0)) as Maksimum
FROM pracownicy INNER JOIN miasta
ON pracownicy.nr_miejsc = miasta.nr_miejsc GROUP BY miasto;
```

|   | miasto   | Minimum | Maksimum |
|---|----------|---------|----------|
| 1 | Kraków   | 2000.00 | 2400.00  |
| 2 | Poznań   | 2300.00 | 2800.00  |
| 3 | Warszawa | 3500.00 | 3500.00  |

**DAILY** Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Przykład wykorzystania funkcji MIN i MAX

Slajd 24

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### Grupowanie

- HAVING – umożliwia stosowanie warunku na podstawie funkcji agregujących
- Przykład – wyświetli miasta z pensją średnią większą niż 2500

```
SELECT miasto, AVG(placa_zasadnicza + ISNULL(premia, 0)) as Średnia
FROM pracownicy INNER JOIN miasta
ON pracownicy.nr_miejsc = miasta.nr_miejsc GROUP BY miasto
having AVG(placa_zasadnicza + ISNULL(premia, 0)) > 2500;
```


|   | miasto   | Średnia     |
|---|----------|-------------|
| 1 | Poznań   | 2550.000000 |
| 2 | Warszawa | 3500.000000 |

**DAILY** Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych


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

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.



**B2E**  
BUSINESS TO EDUCATION



**SZCZECIŃSKI CENTRUM PARK**  
ROZWIĄZANIA TECHNOLOGICZNE

**SQL**  
Structured Query Language


## Ćwiczenia

### Pracownicy

| id | imie       | nazwisko   | zaw     | zakra_poznaczenie | praca      | penal      | zaw_nazwa   | zaw_czas | praca_nazwa |
|----|------------|------------|---------|-------------------|------------|------------|-------------|----------|-------------|
| 1  | Kowalski   | Jan        | 2300.00 | 230.00            | 0000-00-00 | 0000-00-00 | Pracownik   | 1        | 2           |
| 2  | Nowak      | Janet      | 2500.00 | 250.00            | 0000-00-00 | 0000-00-00 | Pracownik   | 1        | 1           |
| 3  | Prokopenko | Przemysław | 2700.00 | 270.00            | 0000-00-00 | 0000-00-00 | Specjalista | 0002     | 1           |
| 4  | Boruch     | Jan        | 2800.00 | 280.00            | 0000-00-00 | 0000-00-00 | Specjalista | 0003     | 1           |
| 5  | Malikowski | Jan        | 2900.00 | 290.00            | 0000-00-00 | 0000-00-00 | Specjalista | 2        | 2           |
| 6  | Nowakowski | Jan        | 3000.00 | 300.00            | 0000-00-00 | 0000-00-00 | Specjalista | 2        | 2           |
| 7  | Witko      | Jan        | 3000.00 | 300.00            | 0000-00-00 | 0000-00-00 | Pracownik   | 2        | 2           |
| 8  | Wojcik     | Michał     | 3000.00 | 300.00            | 0000-00-00 | 0000-00-00 | Specjalista | 0002     | 2           |

### Miejsca

| id | miasto   | zaw        | praca      |
|----|----------|------------|------------|
| 1  | Warszawa | 0000-00-00 | 0000-00-00 |
| 2  | Warszawa | 0000-00-00 | 0000-00-00 |



Lekcja 2 - Języki programowania SQL, środowisko, wprowadzenie do języka SQL

## 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 magagerów

Ćwiczenie 10:

Pokaż najwyższą pensję średnią premie w podziale na stanowiska

Slajd 27



| Podsumowanie              |                                                                  |
|---------------------------|------------------------------------------------------------------|
| JOIN                      | Łączy dane z tabeli                                              |
| INNER JOIN ... ON ...     | Pokazuje tylko wyniki, które są łączą wspólna wartość            |
| (LEFT) (RIGHT) OUTER JOIN | Nie wyklucza wartości                                            |
| GROUPING                  | Grupuje wyniki po zadanym warunku                                |
| HAVING                    | Pozwala na użycie funkcji agregujących w warunku WHERE           |
| UNION                     | Łączy tabele                                                     |
| EXCEPT                    | Pokazuje wartości z jednej tabeli, które nie występują w drugiej |
| MIN / MAX                 | Pozwala wyliczyć wartość maksymalną i minimalną                  |
| COUNT                     | Zlicza wiersze                                                   |
| SUM                       | Sumuje wartości                                                  |
| AVG                       | Liczy wartość średnią                                            |

Lekcja 3 - Złożone zapytania, grupowanie, agregacja danych

Tabela podsumowująca poznane wyrażenia

### 9.3.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 26. Uczniowie w ramach ćwiczenia będą musieli napisać zapytania SQL na bazie tabel przygotowanych przez szkoleniowca.

### 9.3.4 Opis złożonych osiągnięć ucznia

Po tej lekcji uczniowie będą potrafili 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

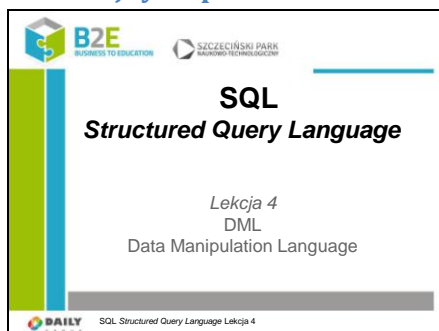
### 9.4.1 Cel lekcji

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

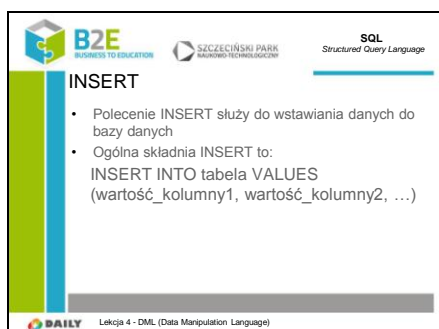


### Slajd 3



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

SQL  
Structured Query Language

### INSERT - przykład

```
INSERT INTO klienci VALUES (1, 'Wiśniewski', 'Mikołaj', '82040112389');
INSERT INTO klienci VALUES (2, 'Karłowicz', 'Marcin', '73122990123');
```

```
SELECT * FROM klienci
```

| nr_klienta | nazwisko   | imie    | pesel       |
|------------|------------|---------|-------------|
| 1          | Wiśniewski | Mikołaj | 82040112389 |
| 2          | Karłowicz  | Marcin  | 73122990123 |

Lekcja 4 - DML (Data Manipulation Language)

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




SQL  
Structured Query Language

### INSERT – wartości puste

- INSERT umożliwia wstawienie wiersza bez uzupełniania wszystkich wartości
- Sposób 1:

```
INSERT INTO klienci VALUES (3, 'Galeas', 'Jan', NULL);
```

- Sposób 2:

```
INSERT INTO klienci(nr_klienta, nazwisko, imie) VALUES (4, 'Zimna', 'Marcin');
```

Lekcja 4 - DML (Data Manipulation Language)

Umieszczanie pustych wartości w wierszach można zrealizować w dwojaki sposób: albo używamy 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




SQL  
Structured Query Language

### INSERT – wartość domyślna

- Do wstawienia wartości domyślnej służy DEFAULT
- Przykład:

```
INSERT INTO klienci VALUES (5, 'Wiśniewska', 'Joanna', DEFAULT);
```

| nr_klienta | nazwisko   | imie    | pesel       |
|------------|------------|---------|-------------|
| 1          | Wiśniewski | Mikołaj | 82040112389 |
| 2          | Karłowicz  | Marcin  | 73122990123 |
| 3          | Galeas     | Jan     | NULL        |
| 4          | Zimna      | Marcin  | NULL        |
| 5          | Wiśniewska | Joanna  | NULL        |

Lekcja 4 - DML (Data Manipulation Language)

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




SQL  
Structured Query Language

### INSERT z SELECT



- Wstawianie wielu wierszy z innej tabeli
- INSERT INTO tabela SELECT ... FROM tabela 2

```
INSERT INTO klienci(nazwisko, imie, pesel)
SELECT imie, nazwisko, pesel FROM pracownicy;
```

Lekcja 4 - DML (Data Manipulation Language)

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



**SQL**  
 Structured Query Language

### UPDATE

- UPDATE służy do aktualizacji istniejących w tabeli wierszy
- Składnia polecenia:
- UPDATE nazwa\_tabeli SET kolumna = nowa\_wartość WHERE kolumna2=wartość


 Lekcja 4 - DML (Data Manipulation Language)

## Slajd 10


**SQL**  
 Structured Query Language

### UPDATE – przykład 1

- Przykład:  
Zaktualizuj premie managerów i ustaw na 500zł



```
UPDATE pracownicy SET premia = 500 WHERE stanowisko = 'Manager';
```

| nr_pracownika | nazwisko   | imie      | placa_zasadnicza | premia | stanowisko  | nr_miejsca | premiowy |
|---------------|------------|-----------|------------------|--------|-------------|------------|----------|
| 1             | Kowalski   | Jan       | 2300.00          | 500.00 | Manager     | 1          | 7        |
| 2             | Nowak      | Krzysztof | 2700.00          | 500.00 | Specjalista | 1          | 1        |
| 3             | Proszynski | Wojciech  | 2700.00          | NULL   | Specjalista | NULL       | 1        |
| 4             | Burzyński  | Tomasz    | 1900.00          | 500.00 | Specjalista | 2          | 7        |
| 5             | Makowski   | Wojciech  | 2000.00          | NULL   | Specjalista | 2          | 7        |
| 6             | Kawalec    | Magdalena | 2100.00          | 200.00 | Specjalista | 1          | 1        |
| 7             | Wit        | Jan       | 3000.00          | 500.00 | Manager     | 3          | 7        |
| 8             | Makowski   | Michał    | 2000.00          | NULL   | Specjalista | NULL       | 7        |


 Lekcja 4 - DML (Data Manipulation Language)

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

## Slajd 11


**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');
```


 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 Kowalski

Slajd 12

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY 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** Lekcja 4 - DML (Data Manipulation Language)

Przykład aktualizuje pole nazwisko gdzie nazwisko to Naramowicka

Slajd 13

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY 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** 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

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**UPDATE - ostrzeżenie**

```
UPDATE klienci SET nazwisko = 'Kowalski';
```

| nr_klienta | nazwisko | imie        | pesel       |
|------------|----------|-------------|-------------|
| 1          | Kowalski | Marek       | 82040112389 |
| 2          | Kowalski | Marcin      | 73122990123 |
| 3          | Kowalski | Jan         | NULL        |
| 4          | Kowalski | Marta       | NULL        |
| 5          | Kowalski | Joanna      | NULL        |
| 6          | Kowalski | Krzysztof   | 82091104357 |
| 7          | Kowalski | Nowak       | 80010123987 |
| 8          | Kowalski | Przepiórka  | 89121203456 |
| 9          | Kowalski | Burzyński   | 79022990123 |
| 10         | Kowalski | Makowski    | 54013112345 |
| 11         | Kowalski | Naramowicka | 77121312098 |
| 12         | Kowalski | Witos       | 69100967234 |

**DAILY** 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 15

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**DELETE**

- DELETE używane jest do usuwania istniejących wierszy z tabeli.
- Składnia:  
DELETE FROM nazwa\_tabeli WHERE nazwa\_kolumny=wartosc;

**DAILY** Lekcja 4 - DML (Data Manipulation Language)

Poznaliśmy sposób umieszczania danych w tabeli, ich modyfikacji – teraz czas na usuwania danych. DELETE to ostatnie z wyrażeń DDL. Podobnie jak w przypadku UPDATE, należy zwrócić szczególną uwagę na warunek WHERE, tak aby nie skasować wszystkich danych z tabeli.





Slajd 16

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**DELETE - przykład**

- Usuwanie z bazy pracownika o nazwisku Nowak

```
DELETE FROM pracownicy WHERE nazwisko = 'Nowak';
```

(1 row(s) affected)

**DAILY** Lekcja 4 - DML (Data Manipulation Language)

Przykład usuwania rekordu – poniżej widzimy odpowiedź systemu. Widzimy, że został usunięty jeden wiersz.

Slajd 17

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**DELETE z podzapytaniem**

- Usuwanie z bazy wszystkich pracowników z najwyższą pensją

```
DELETE FROM pracownicy WHERE placa_zasadnicza =
(SELECT MAX(placa_zasadnicza) FROM pracownicy)
```

**DAILY** Lekcja 4 - DML (Data Manipulation Language)

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

Slajd 18

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**DELETE**

- DELETE używane jest do usuwania istniejących wierszy z tabeli.
- Składnia:  
DELETE FROM nazwa\_tabeli WHERE  
nazwa\_kolumny=wartosc;

**DAILY** Lekcja 4 - DML (Data Manipulation Language)

Poznaliśmy sposób umieszczania danych w tabeli, ich modyfikacji – teraz czas na usuwania danych. DELETE to ostatnie z wyrażeń DDL. Podobnie jak w przypadku UPDATE, należy zwrócić szczególną uwagę na warunek WHERE, tak aby nie skasować wszystkich danych z tabeli.

Slajd 19

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Ćwiczenia**

**Pracownicy**

| id_pracownika | imie       | nazwisko | placa   | placa_zasadnicza | premia | premi | stanowisko | nr_miejsca | profesjonalnosc |
|---------------|------------|----------|---------|------------------|--------|-------|------------|------------|-----------------|
| 1             | Nowak      | Jan      | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 1          | 1               |
| 2             | Nowak      | Jan      | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 1          | 1               |
| 3             | Przybylski | Henryk   | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 2          | 2               |
| 4             | Burzyński  | Henryk   | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 2          | 2               |
| 5             | Malinowski | Henryk   | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 2          | 2               |
| 6             | Nowakowski | Henryk   | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 3          | 3               |
| 7             | Włosa      | Jan      | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 3          | 3               |
| 8             | Przybylski | Henryk   | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 3          | 3               |

**Klenci**

| id_klienta | imie  | nazwisko | placa   | placa_zasadnicza | premia | premi | stanowisko | nr_miejsca | profesjonalnosc |
|------------|-------|----------|---------|------------------|--------|-------|------------|------------|-----------------|
| 1          | Nowak | Jan      | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 1          | 1               |
| 2          | Nowak | Jan      | 2500,00 | 2500,00          | 0,00   | 0,00  | Pracownik  | 1          | 1               |

**Miejsca**

| nr_miejsca | id_miejsca | id_klienta | id_pracownika | id_miejsca | id_klienta | id_pracownika |
|------------|------------|------------|---------------|------------|------------|---------------|
| 1          | 1          | 1          | 1             | 1          | 1          | 1             |
| 2          | 2          | 2          | 2             | 2          | 2          | 2             |
| 3          | 3          | 3          | 3             | 3          | 3          | 3             |

**DAILY** Lekcja 4 - DML (Data Manipulation Language)

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 20

**B2E**  
BUSINESS TO EDUCATION

**SZCZECIŃSKI PARK**  
NAUKOWO-TECHNOLOGICZNY

**SQL**

Structured Query Language

## Podsumowanie

|        |                             |
|--------|-----------------------------|
| INSERT | Wstawia wiersze do tabeli   |
| UPDATE | Aktualizuje wartość wierszy |
| DELETE | Usuwa wiersze z tabeli      |

 **DAILY**

Lekcja 4 - DML (Data Manipulation Language)

Tabela podsumowująca poznane wyrażenia

### 9.4.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 19. Uczniowie napiszą instrukcje modyfikujące dane w bazie danych.

### 9.4.4 Opis założonych osiągnięć ucznia

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

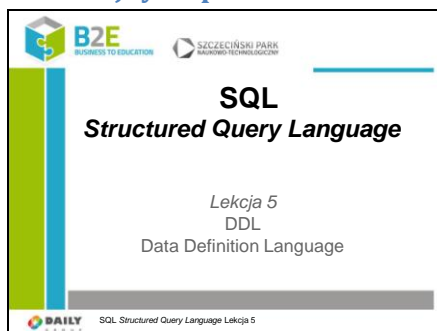
## 9.5 Lekcja 5 - DDL

### 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

**Data Definition Language**

Dzięki DDL (Data Definition Language) można operować na strukturach, w których dane są przechowywane – czyli np. dodawać, zmieniać i kasować tabelę lub bazy. Najważniejsze polecenia tej grupy to:

- **CREATE** (np. CREATE TABLE, CREATE DATABASE, ...) – utworzenie struktury (bazy, tabeli, indeksu itp.),
- **DROP** (np. DROP TABLE, DROP DATABASE, ...) – usunięcie struktury,
- **ALTER** (np. ALTER TABLE ADD COLUMN ...) – zmiana struktury (dodanie kolumny do tabeli, zmiana typu danych w kolumnie tabeli).

Lekcja 5 - DDL (Data Definition Language)

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

**Typy danych w SQL**

- Typy numeryczne
- Data i czas
- Typ znakowy
- Typy binarne
- Typy przestrzenne

Lekcja 5 - DDL (Data Definition Language)

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

**Typy danych w SQL**

- Typy numeryczne

| Typ        | Opis                                                                                                                                  |
|------------|---------------------------------------------------------------------------------------------------------------------------------------|
| bigint     | 8-bajtowy typ numeryczny z zakresem: -2 <sup>63</sup> (-9,223,372,036,854,775,808) do 2 <sup>63</sup> -1 (9,223,372,036,854,775,807). |
| Numeric    | Wielkość typu zależy od precyzji: 1-9 – 5 bajtów, 10-19 – 9 bajtów, 20-28 – 13 bajtów, 29-38 – 17 bajtów.                             |
| bit        | Wartości: 0, 1 lub NULL.                                                                                                              |
| smallint   | 2-bajtowy typ numeryczny z zakresem: -2 <sup>15</sup> (-32,768) do 2 <sup>15</sup> -1 (32,767).                                       |
| decimal    | Wielkość typu zależy od precyzji: 1-9 – 5 bajtów, 10-19 – 9 bajtów, 20-28 – 13 bajtów, 29-38 – 17 bajtów.                             |
| smallmoney | 4-bajtowy typ do zapisu walut z zakresem: -214,748,3648 do 214,748,3647.                                                              |

Lekcja 5 - DDL (Data Definition Language)

Do wartości numerycznych zalicza się typy przedstawione w tabeli.

Slajd 6

**Typy danych w SQL**

- Typy numeryczne

| Typ     | Opis                                                                                                                                                                             |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| int     | 4-bajtowy typ numeryczny z zakresem: -2 <sup>31</sup> (-2,147,483,648) do 2 <sup>31</sup> -1 (2,147,483,647).                                                                    |
| tinyint | 1-bajtowy typ numeryczny z zakresem: 0 – 255.                                                                                                                                    |
| money   | 8-bajtowy typ do zapisu walut z zakresem: -922,337,203,685,477,5808 do 922,337,203,685,477,5807.                                                                                 |
| float   | Typ zmiennoprzecinkowy z zakresem: -1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308. Ilość bajtów zależy od precyzji: 1-24 – 4 bajty (7 cyfr), 25-53 – 8 bajtów (15 cyfr). |
| real    | 4-bajtowy typ zmiennoprzecinkowy z zakresem: -3.40E+38 to -1.18E-38, 0 and 1.18E-38 to 3.40E+38. Typ real jest 24-cyfrowym typem float.                                          |

Lekcja 5 - DDL (Data Definition Language)

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

**Typy danych w SQL**

- Data i czas

| Typ            | Opis                                                                                                                                                                                                                      |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data           | 10-znakowy typ zapisu daty z precyzją 10.0. Domyślnym formatem jest: YYYY-MM-DD, natomiast dopuszczalne wartości to 1900-01-01.                                                                                           |
| datetimeoffset | 8-bajtowy typ do zapisu daty z zakresu: 1 stycznia 1 roku n.e. do 31 grudnia roku 9999, liczone wg kalendarza gregoriańskiego oraz czas 24-godzinny z dokładnością do 100 ns. Uwzględnienie przesunięcia strefy czasowej. |
| datetime2      | 8-bajtowy typ do zapisu daty z zakresu: 1 stycznia 1 roku n.e. do 31 grudnia 9999 oraz czas 24-godzinny z dokładnością do 100 ns.                                                                                         |
| smalldatetime  | 4-bajtowy typ do zapisu daty z zakresu 1900-01-01 do 2079-06-06 z dokładnością do 1 minuty.                                                                                                                               |
| datetime       | 8-bajtowy typ do zapisu daty z zakresu 1 stycznia 1753 do 31 grudnia 9999 z dokładnością do 0.0000001 sekund.                                                                                                             |
| time           | 3 – 5 bajtowy typ do zapisu czasu z dokładnością do 100 ns. Liczba bajtów zależy od skali precyzji.                                                                                                                       |

Lekcja 5 - DDL (Data Definition Language)

Dane zapisywane jako data i czas powinny zostać zaprezentowane za pomocą typów przedstawionych w tabeli.

Slajd 8

**Typy danych w SQL**

- Typ znakowy

| Typ      | Opis                                                                      |
|----------|---------------------------------------------------------------------------|
| char     | Stało-znakowy typ o wielkości 1 – 8000 bajtów.                            |
| text     | Zmiennie-znakowy typ o długości max. 2 <sup>30</sup> – 1 bajtów.          |
| varchar  | Zmiennie-znakowy typ o długości 1 – 8000 bajtów.                          |
| nchar    | Stało-znakowy typ Unicode o wielkości 1 – 4000 bajtów.                    |
| nvarchar | Zmiennie-znakowy typ Unicode o wielkości 1 – 4000 bajtów.                 |
| ntext    | Zmiennie-znakowy typ Unicode o wielkości max. 2 <sup>30</sup> – 1 bajtów. |

Lekcja 5 - DDL (Data Definition Language)

Do wartości znakowych zalicza się typy przedstawione w tabeli

Slajd 9

**Typy danych w SQL**

- Typy binarne

| Typ       | Opis                                                                                                         |
|-----------|--------------------------------------------------------------------------------------------------------------|
| binary    | Przechowywany najczęściej jako stały strumień bajtów typ o wielkości 1 – 8000 bajtów.                        |
| varbinary | Przechowywany najczęściej jako zmienny strumień bajtów typ o wielkości 1 – 8000 bajtów.                      |
| image     | Zmiennie-bajtowy typ o długości 0 – 2 <sup>31</sup> – 1, przeznaczony do składowania obrazów w bazie danych. |

Lekcja 5 - DDL (Data Definition Language)

Do wartości binarnych zalicza typy przedstawione w tabeli

Slajd 10

**Typy danych w SQL**

- Typy przestrzenne

| Typ       | Opis                                                                                                                                                     |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| geography | Typ do przechowywania danych geograficznych, zaimplementowany w .NET CLR. Wykorzystywany jest głównie do zapisu pozycji GPS. Uwzględnia krzywiznę Ziemi. |
| geometry  | Typ do przechowywania typów geometrycznych (w szczególności figur), zaimplementowany w .NET CLR.                                                         |

Lekcja 5 - DDL (Data Definition Language)

SQL Server oferuje typy do reprezentowania obiektów geometrycznych i geograficznych



Slajd 11

**Typy danych w SQL**

- Pozostałe typy

| Typ              | Opis                                                                                                                     |
|------------------|--------------------------------------------------------------------------------------------------------------------------|
| cursor           | Typ danych dla zmiennych parametrów wyjściowych procedury, które zawierają doniesienie do kursora.                       |
| hierarchid       | Typ o zmiennej długości danych, służący do reprezentowania pozycji danej informacji w hierarchii.                        |
| sql_variant      | Typ danych, przechowujący różne wartości typów obsługiwanych przez SQL Server. Jest odpowiednikiem typu var w języku C#. |
| table            | Typ tabelaryczny.                                                                                                        |
| uniqueidentifier | Typ umożliwiający automatyczne generowanie unikalnych liczb binarnych w bazie danych.                                    |
| xml              | Typ XML-owy.                                                                                                             |

Lekcja 5 - DDL (Data Definition Language)

Pozostałe typy zostały przedstawione w tabeli

Slajd 12

**Typy danych w SQL**

- Zapamiętaj, że:
  - SQL Server 2008+ oferuje, oprócz typów standardowych, typy do zapisu danych geograficznych oraz figur geometrycznych, często wykorzystywanych w systemach GIS.
  - Typ real to tak naprawdę float(24).
  - Do zapisu daty i czasu powinno używać się typów, jakie oferuje SQL Server 2008 R2, czyli: time, datetime2, datetimeoffset.

Lekcja 5 - DDL (Data Definition Language)

Slajd 13

**CREATE DATABASE**

- Służy do tworzenia bazy danych
  - Składnia:

```
CREATE DATABASE nazwa_bazy
```

```
CREATE DATABASE Test;
```

```
USE Test;|
```

Lekcja 5 - DDL (Data Definition Language)

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 TABLE**

- Służy do tworzenia tabel
  - Składnia:

```
CREATE TABLE nazwa_tabeli
```

```
(nazwa_kolumny1 typ_danych,
```

```
 nazwa_kolumny2 typ_danych,
```

```
 nazwa_kolumny3 typ_danych,
```

```
 ...
```

```
);
```

Lekcja 5 - DDL (Data Definition Language)

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

  
**CREATE TABLE**  

- Przykład 1

```
CREATE TABLE klienci1(
 nr_klienta int,
 nazwisko nvarchar(50),
 imie nvarchar(50),
 pesel nchar(11)
);
```

| nr_klienta | nazwisko | imie | pesel |
|------------|----------|------|-------|
|------------|----------|------|-------|

Lekcja 5 - DDL (Data Definition Language)

Przykład CREATE tworzący tabelę klienci. Tworząc tabelę 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

  
**CREATE TABLE**  

- Przykład 2 – klucz i autonumerowanie

```
CREATE TABLE klienci2(
 nr_klienta int primary key IDENTITY(1,1),
 nazwisko nvarchar(50),
 imie nvarchar(50),
 pesel nchar(11)
);
```

- IDENTITY(1,1) – pole autonumerowane – zaczynamy od wartości 1 i zwiększamy przy każdym insercie o 1.

Lekcja 5 - DDL (Data Definition Language)

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.

## Slajd 17

  
**CREATE TABLE**  

- Przykład 3 – wartości domyślnie, możliwość wartości pustej

```
CREATE TABLE klienci3(
 nr_klienta int primary key IDENTITY(1,1) NOT NULL,
 nazwisko nvarchar(50) NULL,
 imie nvarchar(50) NOT NULL DEFAULT 'Jan',
 pesel nchar(11) NULL
);
```

- DEFAULT – wartość domyślna
- NULL/NOT NULL – czy możliwa jest wartość pusta

Lekcja 5 - DDL (Data Definition Language)

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**  

- Służy do usuwania bazy danych

```
CREATE DATABASE Test2;

DROP DATABASE Test2;
```

Lekcja 5 - DDL (Data Definition Language)

DROP DATABASE usuwa całą bazę danych. Warto zwrócić uwagę, że operacja jest nieodwracalna i należy z niej korzystać ze szczególną ostrożnością.





Slajd 19

SQL  
Structured Query Language

### DROP TABLE

- Służy do usuwania tabeli

```
CREATE TABLE klienci4(
 nr_klienta int primary key IDENTITY(1,1) NOT NULL,
 nazwisko nvarchar(50) NULL,
 imie nvarchar(50) NOT NULL DEFAULT 'Jan',
 pesel nchar(11) NULL
);

DROP TABLE klienci4;
```

Lekcja 5 - DDL (Data Definition Language)

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

Slajd 20

SQL  
Structured Query Language

### ALTER DATABASE

- Służy do dokonywania zmian bazy danych

```
ALTER DATABASE database_name
SET
(
 { <optionspec> [,...<N>] [WITH <termination>] }
)

<optionspec> ::=
(
 <collation_option>
 | <change_tracking_option>
 | <cursor_option>
 | <database_snapshot_option>
 | <data_correlation_optimization_option>
 | <db_name_option>
 | <db_state_option>
 | <db_user_access_option>
 | <external_access_option>
 | <parameterization_option>
 | <recovery_option>
 | <service_broker_option>
 | <snapshot_option>
 | <sql_option>
)

<collation_option> ::=
```

Lekcja 5 - DDL (Data Definition Language)

ALTER służy głównie służy do modyfikacji istniejących elementów w bazie. Początkujący użytkownicy będą głównie korzystali z polecenia **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 cursorów.

Slajd 21

SQL  
Structured Query Language


### ALTER TABLE

- Służy do modyfikacji tabeli
- Podstawowa składnia ALTER TABLE:  
ALTER TABLE nazwa\_tabeli  
[ADD|DROP|ALTER|..] nazwa\_kolumny ...

Lekcja 5 - DDL (Data Definition Language)



Slajd 22



SQL  
Structured Query Language

### ALTER TABLE

- ADD COLUMN
- Składnia:

```
ALTER TABLE nazwa_tabeli ADD
COLUMN nazwa_kolumny typ_danych;
```

```
ALTER TABLE klienci3 ADD miejscowosc nvarchar(150);
```

 Lekcja 5 - DDL (Data Definition Language)

Przykład ALTER TABLE dodający kolumnę do istniejącej tabeli.

Slajd 23



SQL  
Structured Query Language

### ALTER TABLE

- DROP COLUMN
- Składnia:

```
ALTER TABLE nazwa_tabeli DROP
COLUMN nazwa_kolumny typ_danych;
```

```
ALTER TABLE klienci3 DROP COLUMN miejscowosc;
```

- UWAGA!
- Wraz z usunięciem kolumny zostaną usunięte wszystkie dane zapisane w kolumnie.

 Lekcja 5 - DDL (Data Definition Language)

Przykład ALTER TABLE usuwający kolumnę z istniejącej tabeli.

Slajd 24




SQL  
Structured Query Language

### ALTER TABLE

- ALTER COLUMN
- Składnia:



```
ALTER TABLE nazwa_tabeli ALTER
COLUMN nazwa_kolumny typ_danych;
```

```
ALTER TABLE klienci3 ALTER COLUMN miejscowosc nvarchar(200);
```

 Lekcja 5 - DDL (Data Definition Language)

Przykład ALTER TABLE modyfikujący kolumnę w istniejącej tabeli.

Slajd 25




SQL  
Structured Query Language

### SELECT INTO

- SELECT INTO kopiuje dane z jednej tabeli do drugiej.
- SELECT INTO najczęściej jest używane do tworzenia kopii bezpieczeństwa tabel.

Składnia SELECT INTO:

```
SELECT * INTO nowa_tabela [IN zewnetrzna_baza] FROM
stara_tabela
```

 Lekcja 5 - DDL (Data Definition Language)

SELECT INTO – szybka metoda kopiowania tabel.



Slajd 26

**SQL**  
Structured Query Language

### SELECT INTO

```
SELECT * INTO pracownicy_kopia FROM pracownicy
```

```
SELECT * FROM pracownicy_kopia;
```

| id_pracownika | nazwisko             | imie    | data_urodzenia | penja  | penal      | stanowisko         | tytulo | przynajmniej |
|---------------|----------------------|---------|----------------|--------|------------|--------------------|--------|--------------|
| 1             | Kowalski             | Jan     | 2012-01-01     | 500.00 | 6200100007 | Manager            | 1      | 7            |
| 2             | Przybylska           | Marysia | 2010-01-01     | NA.L   | 6101000008 | Specjalista        | NA.L   | 1            |
| 3             | Butycki              | Paula   | 2005-01-01     | 500.00 | 7800000010 | Specjalista        | 2      | 7            |
| 4             | Michalski            | Robert  | 2002-01-01     | NA.L   | 7400000011 | Specjalista        | 2      | 7            |
| 5             | Napieralska-Podolska | Regina  | 2005-01-01     | 200.00 | 7700000012 | Specjalista        | 1      | 1            |
| 6             | Witek                | Julia   | 2005-01-01     | 600.00 | 6900000013 | Manager            | 3      | 7            |
| 7             | Michalski            | Matylda | 2005-01-01     | NA.L   | 6700000014 | Włoszy specjalista | NA.L   | 7            |

Lekcja 5 - DDL (Data Definition Language)

Przykład SELECT INTO kopiujący dane z tabeli pracownicy.

Slajd 27

**SQL**  
Structured Query Language

### Ćwiczenia

**Pracownicy**

| id_pracownika | nazwisko             | imie    | data_urodzenia | penja  | penal      | stanowisko         | tytulo | przynajmniej |
|---------------|----------------------|---------|----------------|--------|------------|--------------------|--------|--------------|
| 1             | Kowalski             | Jan     | 2012-01-01     | 500.00 | 6200100007 | Manager            | 1      | 7            |
| 2             | Przybylska           | Marysia | 2010-01-01     | NA.L   | 6101000008 | Specjalista        | NA.L   | 1            |
| 3             | Butycki              | Paula   | 2005-01-01     | 500.00 | 7800000010 | Specjalista        | 2      | 7            |
| 4             | Michalski            | Robert  | 2002-01-01     | NA.L   | 7400000011 | Specjalista        | 2      | 7            |
| 5             | Napieralska-Podolska | Regina  | 2005-01-01     | 200.00 | 7700000012 | Specjalista        | 1      | 1            |
| 6             | Witek                | Julia   | 2005-01-01     | 600.00 | 6900000013 | Manager            | 3      | 7            |
| 7             | Michalski            | Matylda | 2005-01-01     | NA.L   | 6700000014 | Włoszy specjalista | NA.L   | 7            |

**Klienci**

| id_klienta | nazwisko  | imie       | penal      |
|------------|-----------|------------|------------|
| 1          | Michalski | Włodzisław | 6200000001 |
| 2          | Michalski | Krzysztof  | 7200000002 |

**Miejsca**

| id_miejsca | id_klienta | numer | stanowisko | tytulo | przynajmniej |
|------------|------------|-------|------------|--------|--------------|
| 1          | Michalski  | 13    | Penja      | 620000 | 65 120 00 00 |
| 2          | Michalski  | 4 120 | Michalski  | 620000 | 62 000 00 00 |
| 3          | Michalski  | 30    | Michalski  | 620000 | 62 000 00 00 |

Lekcja 5 - DDL (Data Definition Language)

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

| Podsumowanie                |                                    |
|-----------------------------|------------------------------------|
| Najważniejsze polecenia DDL |                                    |
| CREATE                      | Tworzy nowe obiekty w bazie danych |
| DROP                        | Usuwa obiekty                      |
| ALTER                       | Modyfikuje obiekty                 |
| SELECT INTO                 | Tworzy kopię tabeli i kopiuje dane |

SQL  
Structured Query Language

Lekcja 5 - DDL (Data Definition Language)

Tabela podsumowująca poznane wyrażenia

### 9.5.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 27. 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

**SQL**  
**Structured Query Language**

Lekcja 6  
DCL, uprawnienia, użytkownicy

SQL Structured Query Language Lekcja 5

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**

- CREATE
- DROP
- ALTER
- SELECT INTO

Lekcja 6 - DCL, uprawnienia, użytkownicy

Przypomnienie materiału z poprzedniej lekcji



Slajd 3

**Data Control Language**

DCL (Data Control Language) ma zastosowanie do nadawania uprawnień do obiektów baz danych. Najważniejsze polecenia w tej grupie to:

- GRANT – służące do nadawania uprawnień do pojedynczych obiektów lub globalnie konkretnemu użytkownikowi
- REVOKE – służące do odbierania wskazanych uprawnień konkretnemu użytkownikowi (np. REVOKE ALL PRIVILEGES ON EMPLOYEE FROM PIOTR - odebranie użytkownikowi wszystkich praw do tabeli EMPLOYEE).
- DENY.

Lekcja 6 - DCL, uprawnienia, użytkownicy

Definicja DCL

Slajd 4

**Użytkownicy i loginy**

- Login – jest używany do połączenia z SZDB
- Użytkownik dotyczy konkretnej bazy danych
- Loginy i użytkowników łączy się ze sobą za pomocą tak zwanego mapowania

Lekcja 6 - DCL, uprawnienia, użytkownicy

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**

Lekcja 6 - DCL, uprawnienia, użytkownicy

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

Slajd 6

**Tworzenie użytkowników MS SMS**

Lekcja 6 - DCL, uprawnienia, użytkownicy

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.

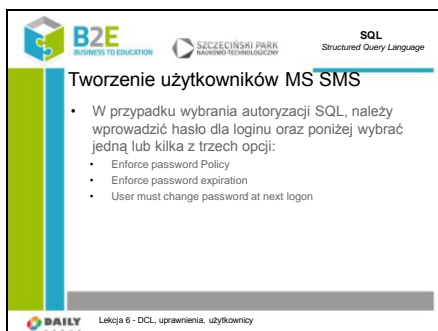
Sekcje Default Database oraz Default language są widoczne zarówno dla loginu z autoryzacją SQL jak

Moduł szkoleniowy SQL - Structured Query Language str. 51

Człowiek - najlepsza inwestycja

i Windows. W Default Database określamy domyślną bazę danych dla loginu.

## Slajd 7



**SQL**  
Structured Query Language

### Tworzenie użytkowników MS SMS

- W przypadku wybrania autoryzacji SQL, należy wprowadzić hasło dla loginu oraz poniżej wybrać jedną lub kilka z trzech opcji:
  - Enforce password Policy
  - Enforce password expiration
  - User must change password at next logon

**DAILY** Lekcja 6 - DCL, uprawnienia, użytkownicy


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



**SQL**  
Structured Query Language

### Tworzenie użytkowników MS SMS

- Na stronie Server Roles wybieramy role serwerowe, przypisane dla tego loginu:
  - bulkadmin
  - dbcreator
  - diskadmin
  - processadmin
  - securityadmin
  - serveradmin
  - setupadmin
  - sysadmin

**DAILY** Lekcja 6 - DCL, uprawnienia, użytkownicy

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),

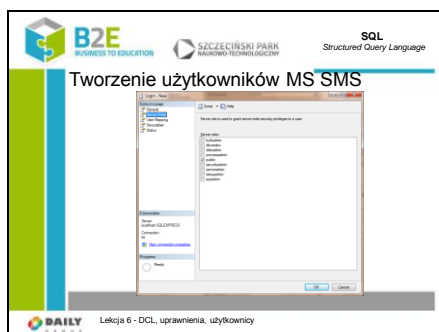
securityadmin – zezwala na zarządzanie loginami i uprawnieniami (ALTER ANY LOGIN),

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 rolę 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\_denydatawriter – 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,

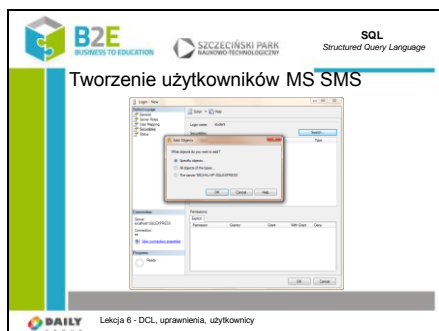
db\_denydatawriter – nie zezwala na zapisywanie i modyfikację baz danych,

db\_owner – zezwala na pełną kontrolę nad bazą danych,

db\_securityadmin – zezwala na zarządzanie uprawnieniami oraz rolami baz danych,

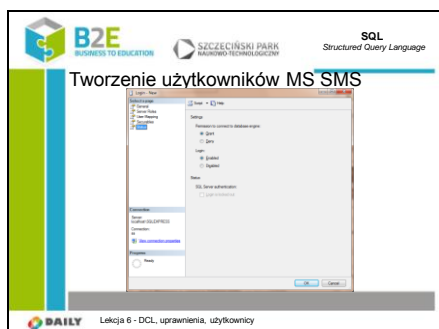
public – rola domyślna, zapewniająca minimum uprawnień.

Slajd 12



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

**SQL**  
Structured Query Language

### Tworzenie użytkowników TSQL

- Taki sam efekt uzyskamy za pomocą polecenia:

```
USE [master]
GO
CREATE LOGIN [user] WITH PASSWORD = 'P@ssw0rd!' NOT ENCRYPTED, DEFAULT_DATABASE=[master], DEFAULT_SCHEMA=[dbo], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
CREATE USER [user] FOR LOGIN [user]
GO
EXEC sp_addaccesskey @'db_name', @'user_name', @'password'
GO
EXEC sp_addaccesskey @'db_name', @'user_name', @'password'
GO
EXEC sp_addaccesskey @'db_name', @'user_name', @'password'
GO
EXEC sp_addaccesskey @'db_name', @'user_name', @'password'
GO
```

Lekcja 6 - DCL, uprawnienia, użytkownicy

Tworzenie nowego loginu za pomocą TSQL i nadawanie odpowiednich uprawnień

Slajd 15

**SQL**  
Structured Query Language

### Podstawowe zasady bezpieczeństwa

- Należy:**
  - Przyznawać użytkownikom uprawnienia tylko takie jakie potrzebują do swojej pracy
  - Monitorować przyznane uprawnienia
  - Pełne prawa do bazy powinien mieć jedynie użytkownik będący właścicielem bazy danych
- Nie należy:**
  - Przypisywać użytkownikom wszystkich uprawnień, aby rozwiązać pewien problem.
  - Pozwalać zwykłemu użytkownikowi na tworzenie baz danych lub obiektów w bazach

Lekcja 6 - DCL, uprawnienia, użytkownicy

Najważniejsze zasady, o których należy pamiętać podczas przydzielania uprawnień

Slajd 16

**SQL**  
Structured Query Language

### Prawa dostępu

- Dla każdej tabeli lub widoku (relacji) można określić prawa dostępu do danych:
  - ALL – pozwala na wszystkie operacje na danych relacji
  - SELECT – pozwala użytkownikowi na odczytywanie danych relacji
  - DELETE – pozwala na usuwanie rekordów z relacji
  - INSERT – pozwala na wstawianie nowych rekordów do tabel lub widoków
  - UPDATE – pozwala na zmianę danych w relacji
  - EXECUTE – pozwala na wykonywaniu procedur zapamiętanych

Lekcja 6 - DCL, uprawnienia, użytkownicy

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

Slajd 17

**SQL**  
Structured Query Language



### Prawa dostępu

- Uwaga** – w MS SQL Server prawa SELECT i UPDATE mogą być przyznawane do wybranych kolumn.
- Domyślnie, nadanie użytkownikowi praw dostępu nie umożliwia mu przekazywanie tych praw innym użytkownikom. Dopiero dołączenie opcji WITH GRANT OPTION pozwoli mu na dalsze przekazywanie posiadanych praw.
- Prawa dostępu przechowywane są w jednej z tabel systemowych.

Lekcja 6 - DCL, uprawnienia, użytkownicy




Slajd 18

  **SQL**  
Structured Query Language

### GRANT


- Do przyznawania praw dostępu utworzonym wcześniej rołom bądź użytkownikom służy polecenie GRANT.
- Przykładowa składnia:  
GRANT nazwy\_praw\_dostępu  
ON nazwa\_tabeli\_lub\_widoku TO  
nazwa\_rol\_i\_lub\_uzytkownika

```
GRANT ALL
ON klienci TO kierownik
WITH GRANT OPTION
```

 Lekcja 6 - DCL, uprawnienia, użytkownicy

Definicja i przykład użycia polecenia GRANT

Slajd 19

  **SQL**  
Structured Query Language

### REVOKE


- Polecenie REVOKE zabiera uprawnienia, które zostały wcześniej przyznane.
- Przykładowa składnia:  
REVOKE nazwy\_praw\_dostępu  
ON nazwa\_tabeli\_lub\_widoku TO  
nazwa\_rol\_i\_lub\_uzytkownika

```
REVOKE ALL
ON klienci TO kierownik
```

 Lekcja 6 - DCL, uprawnienia, użytkownicy

Polecenie REVOKE – służące od odbierania praw do obiektów


Slajd 20

  **SQL**  
Structured Query Language

### DENY



- Zabrania wykonywania operacji, jest silniejsze niż grant
- Nie można wykonywać czynności. Uprawnienie nie może zostać zmienione w wyniku członkostwa w roli.
- Przykładowa składnia:  
DENY nazwy\_praw\_dostępu  
TO nazwa\_rol\_i\_lub\_uzytkownika

```
DENY SELECT ON klienci TO kierownik;
```

 Lekcja 6 - DCL, uprawnienia, użytkownicy

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

  **SQL**  
Structured Query Language


### Przykłady

- aby przyznać użytkownikowi Jan uprawnienie do tworzenia widoku w bazie, należy uruchomić:  

```
GRANT CREATE VIEW TO Jan
```
- aby cofnąć uprawnienie do tworzenia widoków i tabel dla użytkowników Jana i Anny należy uruchomić:  

```
REVOKE CREATE TABLE, CREATE VIEW FROM Anna, Jan
```
- aby przyznać Joe wszystkie uprawnienia w bazie danych należy uruchomić:  

```
GRANT ALL TO Jan
```

 Lekcja 6 - DCL, uprawnienia, użytkownicy

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



SQL  
Structured Query Language

**Podsumowanie**

|        |                       |
|--------|-----------------------|
| GRANT  | nadawanie uprawnień   |
| DENY   | odmawianie uprawnień  |
| REVOKE | odбиieranie uprawnień |

 Lekcja 6 - DCL, uprawnienia, użytkownicy

Tabela podsumowująca poznane wyrażenia

### 9.6.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 22. Uczniowie utworzą nową bazę danych, użytkowników oraz tabelę, 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

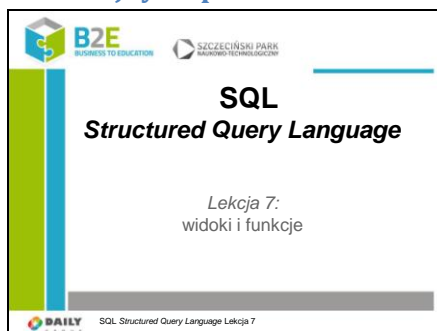
### 9.7.1 Cel lekcji

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.

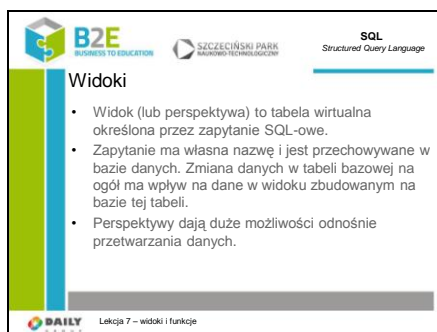
Widoki umożliwiają „zapisanie” skomplikowanych zapytań do późniejszego – szybkiego wykorzystania. Funkcje to obiekty, które już poznaliśmy. Na poprzednich lekcjach używaliśmy funkcji systemowych MIN, MAX, SUM, AVG. SQL Serwer umożliwia definicję własnych funkcji użytkownika – w tej lekcji dowiemy się 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

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Widoki**

- Standardowa składnia definicji instrukcji tworzenia widoku:

```
CREATE VIEW <nazwa widoku> [(<nazwa kol> [, <nazwa kol> ...])]
AS
(SELECT <instrukcja>
[WITH [CASCADED|LOCAL] CHECK OPTION]);
```

**DAILY** Lekcja 7 – widoki i funkcje

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

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Widoki**

- Listy kolumn używa się gdy:

- jakiegokolwiek dwie kolumny mają identyczne nazwy,
- kolumny zawierają wartości wyliczalne,
- występują połączone kolumny o różnych nazwach.

**DAILY** Lekcja 7 – widoki i funkcje

W widokach możemy zapisać SELECT \* FROM table lub SELECT col1, col2, ... FROM table. Slajd wyjaśnia kiedy unikać gwiazdki.

Slajd 7

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Widoki**

- Usuwanie widoków:

- Podobnie jak w przypadku tabel, do kasowania widoków stosuje się DROP:

```
DROP VIEW <nazwa perspektywy>
```

**DAILY** Lekcja 7 – widoki i funkcje

Usuwanie widoków – tak jak wszystkich obiektów w bazie danych – dokonywane jest instrukcją DROP

Slajd 8

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Widoki - przykład**

```
CREATE VIEW oaklanders
AS
SELECT au_fname, au_lname, title
FROM authors, titles, titleauthors
WHERE authors.au_id = titleauthors.au_id
AND titles.title_id = titleauthors.au_id
AND city = 'Oakland';
```

**DAILY** Lekcja 7 – widoki i funkcje

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



Slajd 9

**Widoki - przykład**

```
CREATE VIEW currentinfo (PUB, TYPE, INCOME, AVO_PRICE, AVO_SALES)
AS
SELECT pub_id, type, sum(price*ytd_sales), avg(price), avg(ytd_sales)
FROM titles
GROUP BY pub_id, type;
```

SQL  
Structured Query Language

Lekcja 7 – widoki i funkcje

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

Slajd 10

**Widoki - przykład**

```
CREATE VIEW cities (Author, Authorcity, Pub, Pubcity)
AS
SELECT au_lname, authors.city, pub_name, publishers.city
FROM authors, publishers
WHERE authors.city = publishers.city;
```

SQL  
Structured Query Language

Lekcja 7 – widoki i funkcje

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

Slajd 11

**Widoki modyfikowalne**

- Dane w tablicach bazowych mogą być modyfikowane (wstawiane, usuwane) z poziomu widoku tylko przy spełnieniu określonych wymagań:
  - zmiany muszą być określone jednoznacznie.
  - widok oparty jest na jednej tablicy bazowej;
  - w definicji zapytania nie występują funkcje agregujące

SQL  
Structured Query Language

Lekcja 7 – widoki i funkcje

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

**Więź CHECK w widokach**

- Jeśli w zapytaniu tworzącym widok jest klauzula WHERE, więź CHECK uniemożliwi takie dodanie/ zmodyfikowanie danych do widoku, które naruszałoby tę klauzulę

```
CREATE VIEW nazwa
AS SELECT treść.zapytania.select WHERE warunek
[WITH [CASCADED | LOCAL] CHECK OPTION];
```

SQL  
Structured Query Language

Lekcja 7 – widoki i funkcje

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

**Definiowanie zmiennych w SQL**

- Zmienne są bardzo przydatne podczas tworzenia procedur, kursorów, oraz czasem zwykłych zapytań aby zadeklarować zmienną używamy polecenia DECLARE i poprzedzamy nazwę zmiennej "@", następnie po AS deklarujemy jej typ.

Lekcja 7 – widoki i funkcje

Typy zmiennych mogą być dokładnie takie same jak typy tabel, które omawialiśmy na poprzednich lekcjach

Slajd 14

**Definiowanie zmiennych w SQL**

- Przykład

```
DECLARE @dana AS INT
SET @dana = 1
SELECT @dana
```

Lekcja 7 – widoki i funkcje

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

Slajd 15

**Definiowanie zmiennych w SQL**

- Przykład

```
DECLARE @dana AS INT
SET @dana = 1000
SELECT imie, nazwisko FROM pracownicy WHERE placa_srednia <= @dana
```

Lekcja 7 – widoki i funkcje

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

Slajd 16

**Zmienne tablicowe**

- Zmienne tablicowe to tabele danych przechowywane w pamięci (zapis do nich jest bardzo szybki).
- Zmienia deklarujemy w ten sam sposób co zwykłą zmienną DECLARE @dana następnie TABLE i w nawiasach nazwy kolumn i ich typy)

Lekcja 7 – widoki i funkcje

Definicja zmiennych tablicowych

Slajd 17

**Zmienne tablicowe**

- Przykład

```
DECLARE @dana TABLE (koszty int, miesiac char(10))
INSERT INTO @dana (koszty, miesiac)
SELECT Koszty, miesiac FROM @dana
---kolejny selekt---
SELECT * FROM @dana
```

Lekcja 7 – widoki i funkcje

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 @dana. Uwaga po zakończeniu ostatniego select wartość zmiennej jest wymazywana.

Slajd 18

**Zmienne systemowe**

- Zmienne systemowe oznaczone są dwoma znakami "@". Często używane zmienne systemowe:

- @@ERROR - numer ostatniego błędu
- @@FETCH\_STATUS - czy kursor pobral wiersz (0 gdy pobral)
- @@IDENTITY - zawiera ostatnio wygenerowaną wartość IDENTITY (przydatne, gdy chcemy użyć ID wstawionego przez INSERT wiersza)
- @@ROWCOUNT - zwraca liczbę wierszy, na których operowała ostatnia instrukcja SQL
- @@VERSION - zwraca informację o wersji SQL Serwera

Lekcja 7 – widoki i funkcje

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

Slajd 19

**Podział funkcji**

- Funkcje systemowe, np.

- GETDATE()
- CAST()
- ROUND()
- SIN()
- HOSTNAME()
- ISNULL()

- Funkcje użytkownika

Lekcja 7 – widoki i funkcje

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

Slajd 20

**Funkcje systemowe - przykłady**

- o CONVERT(typ\_danych, wyrażenie [, styl]) - dokonuje konwersji typów danych (styl jest używany przy konwersji do daty, typów walutowych itp.)
- o GETDATE() - zwraca aktualną datę systemową
- o LEFT(napis, ile\_znaków) - zwraca określoną liczbę znaków napisu rozpoczynając od lewej
- o LEN(napis) - zwraca długość napisu
- o REPLACE(napis, wzorec, napis\_do\_zamiany) - wyszukuje i zamienia fragment napisu
- o RIGHT(napis, ile\_znaków) - zwraca określoną liczbę znaków napisu rozpoczynając od prawej
- o SUBSTRING(napis, od, do) - zwraca określoną część napisu

Lekcja 7 – widoki i funkcje

Przykłady najczęściej wykorzystywanych funkcji systemowych





Slajd 21



SQL  
Structured Query Language

### Tworzenie funkcji użytkownika

- Podział funkcji użytkownika:
  - Funkcje skalarne (zwracają wartość)
  - Funkcje tablicowe (zwracają tabelę)

DAILY GROUP Lekcja 7 – widoki i funkcje

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 funkcjach, w aplikacjach oraz procedurach.

Slajd 22



SQL  
Structured Query Language

### Funkcje skalarne

- Składnia:

```
CREATE FUNCTION function_name
(
 @parameter_name parameter_data_type
 [= default] [READONLY]
 [...n]
)
RETURNS return_data_type
[AS]
BEGIN
 function_body
 RETURN scalar_expression
END
```

DAILY GROUP Lekcja 7 – widoki i funkcje

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

Slajd 23



SQL  
Structured Query Language

### Funkcje skalarne

```
CREATE FUNCTION tylko_data (@date smalldatetime)
RETURNS smalldatetime
AS
BEGIN
 declare @return as smalldatetime
 set @return = (convert(smalldatetime,convert(varchar(11),left(@date,11),120),120))
 return @return
END
```

```
SELECT dbo.tylko_data('2012-12-12 12:12:12')
```

| Results             |
|---------------------|
| 2012-12-12 00:00:00 |

DAILY GROUP Lekcja 7 – widoki i funkcje

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

Slajd 24



SQL  
Structured Query Language

### Funkcje tablicowe

- Funkcje o wartościach tabelarycznych (ang. table-valued function) – są nazywane również sparametryzowanymi widokami, zwracają jako wartość tablicę rekordów, na przykład:

DAILY GROUP Lekcja 7 – widoki i funkcje

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



Slajd 25

**SQL**  
Structured Query Language

### Funkcje tablicowe

```

create table dane (ip int identity(1,1), nazwa varchar(30), wartosc int default 0)
insert into dane (nazwa, wartosc) values ('a'),('b'),('c'),('d'),('e')
insert into dane (nazwa, wartosc) values ('f'),('g'),('h'),('i'),('j'),('k'),('l'),('m'),('n')
go

CREATE FUNCTION dbo.FunkcjaTablicowa (@wartosc int)
RETURNS TABLE
AS
RETURN select * from (select ip, nazwa, wartosc from dane where wartosc=@wartosc) as sub
go

select * from dbo.FunkcjaTablicowa(1)
drop function dbo.FunkcjaTablicowa
drop table dane

```

Lekcja 7 – widoki i funkcje

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

Slajd 26

**SQL**  
Structured Query Language

### Ćwiczenia

**Pracownicy**

| id_pracownika | imie_nazwisko | zawod    | zobacz_miejsca_pracy | zobacz_pensje | zobacz_miejsca_pracy | zobacz_pensje |
|---------------|---------------|----------|----------------------|---------------|----------------------|---------------|
| 1             | Nowak         | Jan      | 2000,00              | 200,00        | Warszawa             | 1             |
| 2             | Nowak         | Jan      | 2000,00              | 200,00        | Warszawa             | 1             |
| 3             | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 4             | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 5             | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 6             | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 7             | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 8             | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 9             | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 10            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 11            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 12            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 13            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 14            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 15            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 16            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 17            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 18            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 19            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |
| 20            | Proszynski    | Wojciech | 2000,00              | 200,00        | Warszawa             | 1             |

**Klienci**

| id_klienta | imie_nazwisko | zawod | zobacz_miejsca_pracy | zobacz_pensje | zobacz_miejsca_pracy | zobacz_pensje |
|------------|---------------|-------|----------------------|---------------|----------------------|---------------|
| 1          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 2          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 3          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 4          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 5          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 6          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 7          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 8          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 9          | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 10         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 11         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 12         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 13         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 14         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 15         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 16         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 17         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 18         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 19         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |
| 20         | Nowak         | Jan   | 2000,00              | 200,00        | Warszawa             | 1             |

**Miejsca**

| id_miejsca | id_klienta | id_pracownika | zobacz_miejsca_pracy | zobacz_pensje | zobacz_miejsca_pracy | zobacz_pensje |
|------------|------------|---------------|----------------------|---------------|----------------------|---------------|
| 1          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 2          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 3          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 4          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 5          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 6          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 7          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 8          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 9          | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 10         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 11         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 12         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 13         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 14         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 15         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 16         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 17         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 18         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 19         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |
| 20         | Nowak      | Jan           | 2000,00              | 200,00        | Warszawa             | 1             |

Lekcja 7 – widoki i funkcje

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



| Podsumowanie                         |                                                                                         |
|--------------------------------------|-----------------------------------------------------------------------------------------|
| Widok                                | tabela wirtualna określona przez zapytanie SQL-owe                                      |
| Funkcje o wartościach skalarnych     | zwracają jako wynik pojedynczą wartość                                                  |
| Funkcje o wartościach tabelarycznych | są nazywane również sparametryzowanymi widokami, zwracają jako wartość tablicę rekordów |

Lekcja 7 – widoki i funkcje

Tabela podsumowująca poznane wyrażenia

### 9.7.3 Ćwiczenia

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

### 9.7.4 Opis założonych osiągnięć ucznia

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

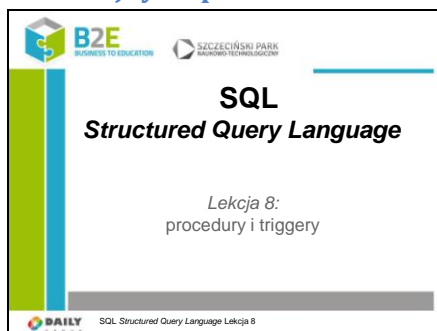
## 9.8 Lekcja 8 - Procedury Triggery

### 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.

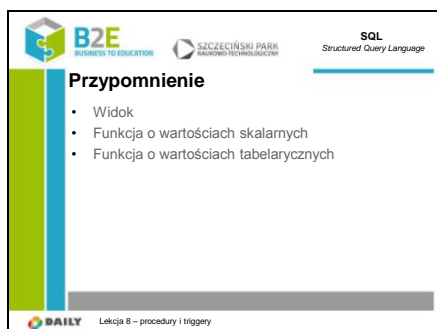
## 9.8.2 Treść - slajdy z opisem

Slajd 1



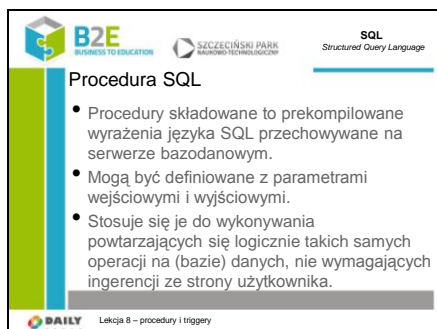
W tej lekcji poznamy kolejne dwa typy obiektów baz danych: procedury składowane i wyzwalacze(triggerzy).

Slajd 2



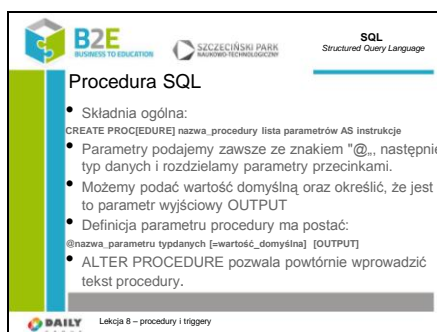
Podsumowanie poprzedniej lekcji.

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





SQL  
Structured Query Language

---



### Procedura SQL

- Zalety używania procedur składowanych:
  - o Różne aplikacje korzystające z tej samej bazy danych korzystają z tej samej procedury — mniejsze ryzyko popełnienia błędu.
  - o Mniejsze koszty uruchomienia i konserwacji.
  - o Z punktu widzenia wydajności
    - procedura wykonywana jest przez wolniejszy język, ale na szybszym serwerze,
    - znaczne zmniejszenie kosztu przesyłu danych.


Lekcja 8 – procedury i triggerzy

Zalety stosowania procedur składowanych przedstawiono na slajdzie.

Slajd 6





SQL  
Structured Query Language

---



### Wybrane procedury systemowe

- o Sp\_tables - tabele bazy danych
- o Sp\_help nazwa\_obiektu - informacje na temat obiektu (np. tabeli, widoku, procedury)
- o Sp\_helptext nazwa\_obiektu - tekst obiektu (np. procedury)
- o Sp\_helpdb nazwa\_bazy - informacje na temat bazy danych
- o Sp\_helpindex nazwa\_tabeli - indeksy założone na tabeli
- o Sp\_helpconstraint nazwa\_tabeli - więzy spójności na tabeli
- o Sp\_spaceused nazwa\_obiektu - ilość miejsca zajętego przez obiekt


Lekcja 8 – procedury i triggerzy

Zalety stosowania procedur składowanych przedstawiono na slajdzie.

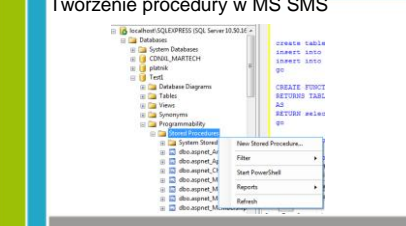
Slajd 7





SQL  
Structured Query Language

---

### Tworzenie procedury w MS SMS




Lekcja 8 – procedury i triggerzy

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.

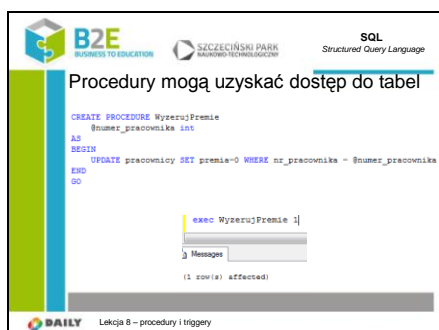


bardzo przydatne. Np. w tym przypadku określamy imię użytkownika. Każdy parametr musi zaczynać się od znaku @. Musimy podać także jego typ oraz ewentualnie długość. Parametry (jeśli jest ich więcej niż jeden), rozdzielamy przecinkiem.

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



**SQL**  
Structured Query Language

Procedury mogą uzyskać dostęp do tabel

```
CREATE PROCEDURE WyszeruPremie
 @numer_pracownika int
AS
BEGIN
 UPDATE pracownicy SET premia=0 WHERE nr_pracownika = @numer_pracownika
END
GO
```

exec WyszeruPremie 1

Messages

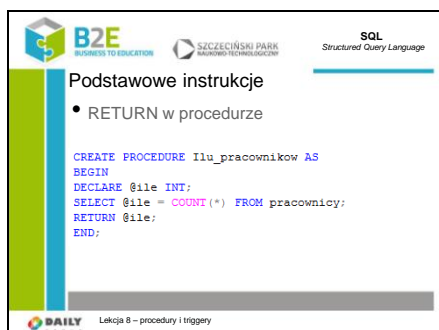
(1 row(s) affected)

Lekcja 8 – procedury i triggerzy

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 procedurę, 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



**SQL**  
Structured Query Language

Podstawowe instrukcje

- RETURN w procedurze

```
CREATE PROCEDURE Ilu_pracownikow AS
BEGIN
 DECLARE @ile INT;
 SELECT @ile = COUNT(*) FROM pracownicy;
 RETURN @ile;
END;
```

Lekcja 8 – procedury i triggerzy

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

**Podstawowe instrukcje**

- PRINT

```
DECLARE @Imie VARCHAR(9), @Nazwisko VARCHAR(20)
SELECT @nazwisko = Nazwisko, @imie=imie FROM pracownicy
WHERE nr_pracownika=1

PRINT 'Imię=' + @imie + ', Nazwisko=' + @Nazwisko
```

Messages  
Imię=Jan, Nazwisko=Kowalski

Lekcja 8 – procedury i triggerzy

Do debuggowania (śledzenia poszczególnych kroków fragmentu kodu), najczęściej używa się instrukcji PRINT, która w konsoli umożliwia wyświetlenie (wydruk) zmiennej lub ciągu znaków.

Slajd 15

**Podstawowe instrukcje**

- Instrukcja warunkowa

IF warunek  
instrukcja lub blok  
[ELSE  
instrukcja lub blok]

```
IF $pensja > 0
BEGIN
 INSERT INTO pracownicy (nazwisko, placa_zasadnicza, nr_miejsc)
 VALUES (@nazwisko, $pensja, @miejsc);
 PRINT 'Wstawiono pracownika';
END
ELSE
 PRINT 'Niepoprawna pensja';
```

Lekcja 8 – procedury i triggerzy

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

**Podstawowe instrukcje**

- Instrukcja iteracji

WHILE warunek  
instrukcja lub blok

```
WHILE (SELECT AVG(placa_zasadnicza) FROM pracownicy) < 2000
BEGIN
 UPDATE pracownicy SET placa_zasadnicza = placa_zasadnicza * 1.2;
 IF (SELECT MAX(placa_zasadnicza) FROM pracownicy) > 4000 BREAK;
 ELSE CONTINUE;
END;

SELECT * FROM pracownicy;
```

Lekcja 8 – procedury i triggerzy

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

**SELECT w procedurze**

- Procedura oprócz parametrów wejściowych i wyjściowych może wykonywać jedno lub kilka zapytań SELECT

```
CREATE PROC PokazTabele AS
BEGIN
 SELECT * FROM pracownicy;
 SELECT * FROM klienci;
END;
GO
EXEC PokazTabele;
```

Lekcja 8 – procedury i triggerzy

Przykład procedury wywołującej dwa zapytania typu SELECT. Rezultatem działania tej procedury będzie wyświetlenie zawartości obu tabel: klienci i pracownicy.



Slajd 18

**Triggery (wyzwalacze)**

- Predefiniowane operacje na danych wyzwalane zdarzeniami.
- Definiowane przez użytkownika i uruchamiane automatycznie.
- Inne nazwy: wyzwalacze, procedury wyzwalane.

Lekcja 8 – procedury i triggery

Definicja triggerów

Slajd 19

**Triggery (wyzwalacze)**

- Triggery uruchamiane są zdarzeniami:
  - INSERT
  - UPDATE
  - DELETE
- Triggery mogą być uruchamiane:
  - BEFORE
  - AFTER
- Triggery mogą być uruchamiane:
  - FOR EACH ROW
  - FOR EACH STATEMENT

Lekcja 8 – procedury i triggery

Wyzwalacze uruchamiane są zdarzeniami: INSERT, UPDATE, DELETE.

Wyzwalacze mogą być uruchamiane: BEFORE, AFTER, czyli odpowiednio przed i 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

**Triggery (wyzwalacze)**

- Składnia:  
CREATE TRIGGER nazwa\_wyzwalacza  
ON tabela  
FOR [INSERT|UPDATE|DELETE]  
AS instrukcje
- Po słowie FOR (równoważnie AFTER) możemy napisać jedną, dwie lub wszystkie trzy nazwy (oddzielone przecinkami).
- ALTER TRIGGER pozwala powtórnie wprowadzić zmodyfikowany tekst wyzwalacza.

Lekcja 8 – procedury i triggery

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

**Triggery – przykład**

- Uniemożliwienie usuwania danych  
CREATE TRIGGER wywali ON pracownicy  
FOR DELETE AS  
BEGIN PRINT 'Nie możesz usunąć danych z tej tabeli!';  
SELECT \* FROM pracownicy;  
ROLLBACK;  
END;
- Próba usunięcia  
DELETE FROM pracownicy;

Lekcja 8 – procedury i triggery

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



## Slajd 22

**SQL**  
Structured Query Language

### Usuwanie triggerów i procedur

- Usuwanie triggerów

```
DROP TRIGGER nazwa_triggera
```

- Usuwanie procedur składowanych

```
DROP PROCEDURE nazwa_procedury
DROP PROC nazwa_procedury
```

Lekcja 8 – procedury i trigger

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

## Slajd 23

**SQL**  
Structured Query Language

### Ćwiczenia

**Pracownicy**

| id_pracownika | imie_nazwisko | zaw | zaw_nazwa  | praca      | praca_nazwa | id_miejsca | id_pracownika | praca_nazwa |
|---------------|---------------|-----|------------|------------|-------------|------------|---------------|-------------|
| 1             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 1          | 1             | Pracownik   |
| 2             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 2          | 2             | Pracownik   |
| 3             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 3          | 3             | Pracownik   |
| 4             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 4          | 4             | Pracownik   |
| 5             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 5          | 5             | Pracownik   |
| 6             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 6          | 6             | Pracownik   |
| 7             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 7          | 7             | Pracownik   |
| 8             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 8          | 8             | Pracownik   |
| 9             | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 9          | 9             | Pracownik   |
| 10            | Nowak         | Jan | 2000-01-01 | 2000-01-01 | Pracownik   | 10         | 10            | Pracownik   |

**Klienci**

| id_klienta | imie_nazwisko | zaw | zaw_nazwa  |
|------------|---------------|-----|------------|
| 1          | Nowak         | Jan | 2000-01-01 |
| 2          | Nowak         | Jan | 2000-01-01 |
| 3          | Nowak         | Jan | 2000-01-01 |

**Miejsca**

| id_miejsca | zaw   | zaw_nazwa | zaw_nazwa  |
|------------|-------|-----------|------------|
| 1          | Nowak | Jan       | 2000-01-01 |
| 2          | Nowak | Jan       | 2000-01-01 |
| 3          | Nowak | Jan       | 2000-01-01 |

Lekcja 8 – procedury i trigger

## 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




**SQL**  
 Structured Query Language

### Podsumowanie

|                      |                                                                           |
|----------------------|---------------------------------------------------------------------------|
| Procedura składowana | Pełnospisowe wyrażenie języka SQL, przechowywane na serwerze baz danych   |
| 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  |

**DAILY** SQL Structured Query Language Lekcja 8 – procedury i trigger

Tabela podsumowująca poznane wyrażenia

### 9.8.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 23. 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 trigger 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 – constraint, służące zachowaniu integralności danych.

### 9.9.2 Treść - slajdy z opisem

Slajd 1

**SQL**  
 Structured Query Language

### Lekcja 9: indeksy, constrainty

**DAILY** SQL Structured Query Language Lekcja 9

W trakcie tej lekcji poznamy strukturę optymalizującą bazę danych jaką jest indeks. W drugiej części lekcji poznamy ograniczenie bazy – constraint, służące zachowaniu integralności danych.

Slajd 2




**SQL**  
 Structured Query Language

### Przypomnienie

- Procedura składowana
- Wyzwalacz
- Warunek IF
- Pętla WHILE

**DAILY** SQL Structured Query Language Lekcja 9 – indeksy, constrainty

Przypomnienie pojęć z poprzednich zajęć

## Slajd 3

  **SQL**  
Structured Query Language

## Indeksy

Indeks jest specjalną strukturą danych wprowadzoną w celu zwiększenia prędkości wykonywania operacji na tabeli. Indeks w bazie danych jest odpowiednikiem spisu treści w książce.



Lekcja 9 – indeksy, constrainty

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 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

  **SQL**  
Structured Query Language

## Indeksy

- Tworzenie indeksów  
`CREATE INDEX nazwa_indeksu ON tabela (nazwa_indeksu);`
- lub  
`ALTER TABLE nazwa_tabeli ADD INDEX (nazwa_indeksu);`



Lekcja 9 – indeksy, constrainty



Slajd 5

Indeksy – do czego służą?

- Zwiększają wydajność
- Zapisują gdzie znajduje się dana wartość
- Umożliwiają wyszukiwanie w tabelach bez potrzeby sprawdzania zawartości od początku do końca

SQL  
Structured Query Language

Lekcja 9 – indeksy, constrainty

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.

Slajd 6

Indeksy – przykład użycia

Wyszukiwanie binarne (Szukamy 13)

1. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

2. 9 10 11 12 13 14 15

3. 13 14 15

4. 13

SQL  
Structured Query Language

Lekcja 9 – indeksy, constrainty

Źródło: microsoft.com

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 połówek 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

Moduł szkoleniowy SQL - Structured Query Language str. 77

Człowiek - najlepsza inwestycja

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 jakiegokolwiek funkcji sortującej.

Źródło: microsoft.com

Slajd 7



**Indeksy – kiedy stosować**

- Indeks optymalizuje zapytania = stosować wszędzie?
- Nie:
  - Indeks zajmuje miejsce w bazie danych
  - Indeks optymalizuje odczyt, pogarsza czas zapisu
- Stosujemy indeksy na polach, które najczęściej wykorzystujemy w warunku WHERE
- EXPLAIN – ułatwia podejmowanie decyzji gdzie stosować indeks

SQL Structured Query Language

DAILY Lekcja 9 – indeksy, constrainty

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. 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.



Slajd 8

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Indeksy – podział**

- Ze wzgl. na liczbę wskazań indeksu
  - Indeks gęsty
  - Indeks rzadki, przykład:

Lekcja 9 – indeksy, constrainty

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

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Indeksy – podział**

- Ze względu na liczbę poziomów
  - Indeksy jednopoziomowe
  - Indeksy wielopoziomowe

Lekcja 9 – indeksy, constrainty

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

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Indeks główny**

- Primary key (PK)

Lekcja 9 – indeksy, constrainty

**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.

Slajd 11

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Indeks zgrupowany**

- Clustered index

Lekcja 9 – indeksy, constrainty

**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

**SQL**  
Structured Query Language

**Plik danych z polem indeksowania**

|    |    |
|----|----|
| 7  | 25 |
| 10 | 15 |
| 3  | 6  |
| 20 |    |

**Plik indeksu**

| Wartość pola indeksu | Wskaźnik |
|----------------------|----------|
| 1                    | 0        |
| 2                    | 0        |
| 3                    | 0        |
| 4                    | 0        |
| 5                    | 0        |
| 6                    | 0        |

Lekcja 9 – Indeksy, constrainty

**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

**Indeksy – przykłady**

- Tworzenie indeksów przy tworzeniu tabeli:

```
CREATE TABLE nazwa_tabeli (
 kolumna1 INT,
 kolumna2 INT,
 UNIQUE indeks_unikalny (kolumna1),
 INDEX indeks_zwykly (kolumna2)
)
```

**SQL**  
Structured Query Language

Lekcja 9 – Indeksy, constrainty

Slajd 14

**Indeksy – przykłady**

- Tworzenie indeksów w istniejącej tabeli:

```
ALTER TABLE nazwa_tabeli
ADD UNIQUE indeks_unikalny (kolumna1),
ADD INDEX indeks_zwykly (kolumna2)
```

**SQL**  
Structured Query Language

Lekcja 9 – Indeksy, constrainty



Slajd 15

SQL  
Structured Query Language

Indeksy – przykłady

- Możemy też utworzyć sam indeks za pomocą CREATE INDEX

```
CREATE UNIQUE INDEX indeks_unikalny ON nazwa_tabeli (kolumna1)
CREATE INDEX indeks_zwykly ON nazwa_tabeli (kolumna2)
```

Lekcja 9 – Indeksy, constrainty

Slajd 16

SQL  
Structured Query Language

Indeksy – przykłady

- Indeks na kilku polach

```
CREATE INDEX indeks_imie ON osoby (imie)
CREATE INDEX indeks_nazw ON osoby (nazwisko)
CREATE INDEX indeks_imie_nazw ON osoby (imie, nazwisko)
```

Lekcja 9 – Indeksy, constrainty

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

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### Klauzula CONSTRAINT

- o CONSTRAINT = więź, ograniczenie
- o Ograniczenie podobne do indeksu, chociaż może również służyć do ustanowienia relacji z inną tabelą.
- o Za pomocą klauzuli CONSTRAINT w instrukcjach ALTER TABLE i CREATE TABLE można tworzyć i usuwać ograniczenia.
- o Istnieją dwa typy klauzul CONSTRAINT:
  - ograniczenie dla jednego pola,
  - ograniczenie dla większej liczby pól.

**DAILY** Lekcja 9 – indeksy, constraint

Slajd 18

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

### Podstawowe CONSTRAINT

- NOT NULL,
- CHECK,
- UNIQUE,
- PRIMARY KEY,
- FOREIGN KEY

**DAILY** Lekcja 9 – indeksy, constraint

W trakcie wcześniejszych lekcji omówiliśmy już NULL/NOT NULL oraz PRIMARY KEY, na kolejnych slajdach przyjrzymy się pozostałym.

Slajd 19

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY 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_płaca CHECK ((płaca_zasadnicza > 0));
```

**DAILY** Lekcja 9 – indeksy, constraint

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** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY 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** Lekcja 9 – indeksy, constraint

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

**FOREIGN KEY**

- Klucz obcy - dodatkowa kolumna lub zbiór kolumn w danej tabeli z wartościami, stanowiącymi klucz główny w innej

Lekcja 9 – indeksy, constrainty

Przykład tabeli pracownicy, gdzie nr\_miejsca jest kluczem obcym do kolumny nr\_miejsca w tabeli miejsca

Slajd 22

**FOREIGN KEY - tworzenie**

```
CREATE TABLE Nazwa_tabeli
(
 kolumna1 TYP,
 kolumnaN TYP,
 CONSTRAINT nazwa_klucza_obcego
 FOREIGN KEY (nazwa_kolumny_z_tworzonej_tabeli)
 REFERENCES tabela_referencyjna (kolumna_tabeli_referencyjna)
)
```

Lekcja 9 – indeksy, constrainty

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

**FOREIGN KEY**

- Najważniejsze informacje
  - o Klucz obcy stanowi kolumnę, bądź zbiór kolumn, będących kluczem głównym w innej tabeli.
  - o Łączenie tabel może odbywać się poprzez zapytanie z wykorzystaniem INNER/OUTER JOIN.
  - o Klucze obce tworzy się w T-SQL, dodając odpowiednie wartości w klauzuli CONSTRAINT FOREIGN KEY.

Lekcja 9 – indeksy, constrainty

Podsumowanie informacji o kluczach obcych



Slajd 24

| B2E                   |            | SZCZECIŃSKI PARK       |                | SQL                       |            |
|-----------------------|------------|------------------------|----------------|---------------------------|------------|
| BUSINESS TO EDUCATION |            | NAUKOWO-TECHNOLOGICZNY |                | Structured Query Language |            |
| <b>Ćwiczenia</b>      |            |                        |                |                           |            |
| <b>Pracownicy</b>     |            |                        |                |                           |            |
| nr_pracownika         | imie       | nazwisko               | pesel          | stanowisko                | nr_miejsca |
| 1                     | Kowalski   | Jan                    | 12345678901234 | Pracownik                 | 1          |
| 2                     | Nowak      | Karl                   | 12345678901234 | Specjalista               | 1          |
| 3                     | Prochaska  | Natasha                | 12345678901234 | Specjalista               | 1          |
| 4                     | Burczyk    | Paula                  | 12345678901234 | Specjalista               | 2          |
| 5                     | Malinowski | Paula                  | 12345678901234 | Specjalista               | 3          |
| 6                     | Malinowski | Paula                  | 12345678901234 | Specjalista               | 1          |
| 7                     | Witek      | Jan                    | 12345678901234 | Pracownik                 | 2          |
| 8                     | Malinowski | Paula                  | 12345678901234 | Specjalista               | 1          |

| Klienci    |            |
|------------|------------|
| nr_klienta | imie       |
| 1          | Nowak      |
| 2          | Malinowski |

| Miejsca    |            |
|------------|------------|
| nr_miejsca | imie       |
| 1          | Nowak      |
| 2          | Malinowski |

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

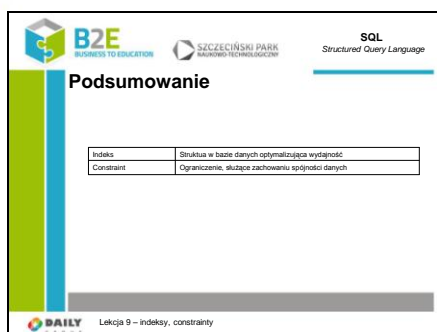
Ćwiczenie 7:

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 CONSTRAINT do kolumny płaca zasadnicza w tabeli pracownicy

Slajd 25



**Podsumowanie**

|            |                                                   |
|------------|---------------------------------------------------|
| Indeks     | Sposób w bazie danych optymalizująca wydajność    |
| Constraint | Ograniczenie, służące zachowaniu spójności danych |

Lekcja 9 – indeksy, constrainty

Tabela podsumowująca poznane wyrażenia

### 9.9.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdzie 24. 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.

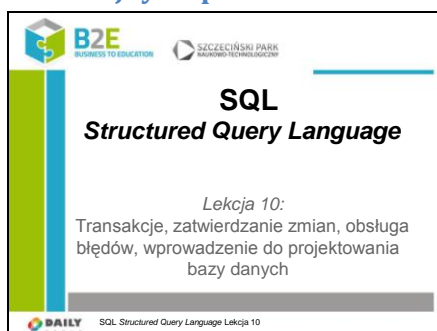
## 9.10 Lekcja 10 - Transakcje, zatwierdzanie zmian, obsługa błędów, wprowadzenie do projektowania bazy danych

### 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



**SQL**  
**Structured Query Language**

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

SQL Structured Query Language Lekcja 10

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

**Przypomnienie**

- INDEKS
- CONSTRAINT

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

Podsumowanie zagadnień z poprzedniej lekcji

Slajd 3

**Transakcje**

- BEGIN TRAN - rozpoczęcie transakcji
- COMMIT TRAN - zatwierdzenie transakcji
- ROLLBACK TRAN - wycofanie transakcji

Domyślnie ustawiona jest opcja IMPLICIT\_TRANSACTIONS na OFF. Przy takim ustawieniu, jeśli nie zastosujemy BEGIN TRAN, system traktuje każdą instrukcję DML jako osobną transakcję i zatwierdza ją. Aby to wyłączyć, należy użyć (wtedy BEGIN TRAN przestaje być konieczne):  
**SET IMPLICIT\_TRANSACTIONS ON**

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

Podstawowe komendy związane z transakcjami.

Slajd 4

**Transakcje**

- Przykład

```
BEGIN TRANSACTION UsuwaniePracownika
WITH MARK N'Usuwanie pracownika';
GO
USE Test1;
GO
DELETE FROM pracownicy WHERE nr_pracownika = 13;
GO
COMMIT TRANSACTION UsuwaniePracownika;
GO
```

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

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

Slajd 5

**Obsługa błędów**

```
create table dane (id int, produkt varchar(30));
BEGIN TRANSACTION; -- początek transakcji
BEGIN TRY
 INSERT INTO dane (id, produkt) VALUES (1, 'maki');
 SELECT 2/0;
END TRY
BEGIN CATCH
 SELECT
 ERROR_NUMBER() AS ErrorNumber,
 ERROR_MESSAGE() AS ErrorMessage;
 IF @@ERROR > 0
 ROLLBACK TRANSACTION; -- wycofanie zmian
END CATCH;
IF @@ERROR > 0
 COMMIT TRANSACTION; -- zakończenie zmian
SELECT * FROM dane;
```

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

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

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Obsługa błędów**

- Kiedy stosować?
  - W transakcjach - po wykryciu błędu wycofujemy transakcję,
  - W procedurach składowanych - wykrywamy błędy powstałe głównie w wyniku niepoprawnych wartości parametrów przez użytkownika,
  - W triggerach - podobnie jak w transakcjach, po napotkaniu błędu wycofywana jest transakcja wywołująca trigger,
  - W blokach kodu SQL - wszelkie rozbudowane bloki kodu wymagają wykrywania błędów

**DAILY** Lekcja 8 – procedury i triggerzy

Kiedy obsługa błędów jest niezbędna.

Slajd 7

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**Obsługa błędów**

- Jak wykrywać błędy?
  - IF...ELSE
  - @@ERROR
  - TRY...CATCH

**DAILY** Lekcja 8 – procedury i triggerzy

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 - przechwyc i obsłuż)

Slajd 8

**B2E** **SZCZECIŃSKI PARK** **SQL**  
BUSINESS TO EDUCATION NAUKOWO-TECHNOLOGICZNY Structured Query Language

**TRY CATCH**

- Składnia:

```
BEGIN TRY
 { sql_statement | statement_block }
END TRY
BEGIN CATCH
 [{ sql_statement | statement_block }]
END CATCH [;]
```
- W bloku try – kod stanowiący potencjalne ryzyko
- W bloku catch – obsługa ryzyka

**DAILY** Lekcja 8 – procedury i triggerzy

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

**Normalizacja**

Normalizacja bazy danych jest to proces mający na celu eliminację powtarzających się danych w relacyjnej bazie danych.

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

**1NF - Pierwsza postać normalna**

Jej jedynym warunkiem jest aby każda składowa w każdej krotce była elementarna (nie dawała podzielić się na mniejsze wartości). Ważną cechą relacji utworzonych zgodnie z modelem relacyjnym jest to, że zawsze są znormalizowane - spełniają 1NF.

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

**2NF - Druga postać normalna**

- Utwórz oddzielne tabele dla zestawów wartości, odnoszących się do wielu rekordów.
- Ustal powiązania tabel za pomocą klucza obcego.

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

B2E BUSINESS TO EDUCATION SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY SQL Structured Query Language

**3NF - Trzecia postać normalna**

- Wypełnij pola, które nie zależą od klucza.

DAILY Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

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

B2E BUSINESS TO EDUCATION SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY SQL Structured Query Language

**Ćwiczenie normalizacja**

- Wypełnij pola, które nie zależą od klucza. (uzyskaj 1NF)

| NrStudenta | Imię     | Pok-Dor | Klasa1 | Klasa2 | Klasa3 |
|------------|----------|---------|--------|--------|--------|
| 1022       | Nowak    | 412     | 101-07 | 143-01 | 159-02 |
| 4123       | Kowalski | 216     | 201-01 | 211-02 | 214-01 |

DAILY Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

Spróbujmy na innej tabeli: jak powinna wyglądać ta tabela w 1NF?



Slajd 14

**Ćwiczenie normalizacja**

- Pierwsza postać normalna: brak powtarzających się grup

| NrStudenta | Doradca  | Pok-Dor | NrKlasy |
|------------|----------|---------|---------|
| 1022       | Nowak    | 412     | 101-07  |
| 1022       | Nowak    | 412     | 143-01  |
| 1022       | Nowak    | 412     | 159-02  |
| 4123       | Kowalski | 216     | 201-01  |
| 4123       | Kowalski | 216     | 211-02  |
| 4123       | Kowalski | 216     | 214-01  |

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

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

**Ćwiczenie normalizacja**

- Druaga postać normalna: eliminowanie powtarzających się danych

| NrStudenta | Doradca  | Pok-Dor |
|------------|----------|---------|
| 1022       | Nowak    | 412     |
| 4123       | Kowalski | 216     |

| NrStudenta | NrKlasy |
|------------|---------|
| 1022       | 101-07  |
| 1022       | 143-01  |
| 1022       | 159-02  |
| 4123       | 201-01  |
| 4123       | 211-02  |
| 4123       | 214-01  |

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

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

**Ćwiczenie normalizacja**

- Trzecia postać normalna: eliminowanie danych, które nie zależą od klucza

| Student:   |          |     | Wydział: |       |         |
|------------|----------|-----|----------|-------|---------|
| NrStudenta | Doradca  |     | Nazwa    | Pokój | Wydział |
| 1022       | Nowak    | 412 | Nowak    | 412   | 42      |
| 4123       | Kowalski |     | Kowalski | 216   | 42      |

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

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 plusy i minusy**

Cel normalizacji:

- uniknięcie redundancji (tj. powtarzania się pól z identycznymi wartościami w różnych tabelach);
- wyeliminowanie niewygodnych relacji wieloznacznych;
- uniknięcie anomalii przy aktualizacji: modyfikacji, wstawianiu i usuwaniu;
- uniknięcie niespójności.

Koszt:

- Mnożenie liczby tabel – wydłużenie czasu dostępu do danych.

Poszukiwanie kompromisu (dwa współzawodniczące cele):

- Zapobieganie anomalom oraz zapewnienie rozsądnego czasu dostępu do danych.

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

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



Slajd 18

**SQL**  
Structured Query Language

**Ćwiczenia**

Wprowadź obsługę błędów w procedurze (TRY-CATCH i RAISERROR):

```
CREATE PROCEDURE podzieli1
(
 @dzielnia float,
 @dzielnik float
)
AS
BEGIN
 SELECT @dzielnia / @dzielnik
END
```

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

Np.

CREATE PROCEDURE podzieli1

(

@dzielnia float,

@dzielnik float

)

AS

IF @dzielnik = 0

RAISERROR('Dzielenie przez  
ZERO!',14,1)

ELSE

SELECT @dzielnia / @dzielnik

END

Slajd 19

**SQL**  
Structured Query Language

**Ćwiczenia**

Przeprowadź normalizację tabeli:

| miasto                  | miasto_prawidłowo | miasto_prawidłowo | miasto_prawidłowo         |
|-------------------------|-------------------|-------------------|---------------------------|
| Łódź, miasto dla dzieci | Katowice          | Nowak             | ul. Życioka 10 m. 6.1.001 |
| Łódź                    | Warszawa          | Wojcik            | ul. Moniuszki 1.1.001     |
| Łódź                    | Łódź              | Łódź              | ul. Życioka 10 m. 6.1.001 |
| Łódź                    | Katowice          | Nowak             | ul. Życioka 10 m. 6.1.001 |
| Łódź                    | Warszawa          | Kat               | Świętokrzyska 8           |
| Łódź                    | Warszawa          | Wojcik            | ul. Moniuszki 1.1.001     |

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

Slajd 20

**SQL**  
Structured Query Language

**Podsumowanie**

|               |                                                                  |
|---------------|------------------------------------------------------------------|
| TRANSACTION   | Zestaw operacji w SQL, zaimplementowany fizycznie zabezpieczenie |
| COMMIT        | Służy do zatwierdzenia transakcji                                |
| ROLLBACK      | Służy do odroczenia transakcji                                   |
| TRY ... CATCH | Zabezpieczenie kodu                                              |
| Normalizacja  | Eliminacja powtarzających się danych w relacyjnej bazie danych   |
| 1NF           | Pierwsza postać normalna                                         |
| 2NF           | Druga postać normalna                                            |
| 3NF           | Tercja postać normalna                                           |

Lekcja 10 – Transakcje, obsługa błędów, wprowadzenie do projektowania bazy danych

Tabela podsumowująca poznane wyrażenia

### 9.10.3 Ćwiczenia

Ćwiczenie zostało przedstawione na slajdach 18 i 19, 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.