Bazy danych

Wykład 5_3

Temat: Wyszukiwanie danych w języku T-SQL

Sławomir Świętoniowski

Plan wykładu

1. **Polecenie SELECT.**
2. Złączenia tabel.
3. Podzapytania.

Instrukcja SELECT: wyszukiwanie danych

Instrukcja SELECT zwraca (domyślnie) wszystkie wiersze spełniające warunek wyszukiwania. Składnia – ważniejsze elementy:

```
SELECT [ ALL | DISTINCT ] [ TOP [ PERCENT ] [ WITH TIES ] columns
    [ AS column_heading ]
    [ INTO new_table_name ]
    FROM table_source  [ AS table_alias ]
    [ INNER | {{ LEFT | RIGHT | FULL } [ OUTER ] }  | [ CROSS ] ] | JOIN
    table2
    [ [ AS ] table_alias ]
    [ WHERE search_condition ]
    [ GROUP BY [ ALL group_by_expression ] ]
    [ HAVING search_condition ]
    [ ORDER BY order_expression [ ASC | DESC ] ]
```

Instrukcja SELECT Parametry (1)

- **ALL** – zwrócone mogą być wiersze z powtórzeniami wartości (ustawienie domyślne);
- **DISTINCT** – zwracane są wyłącznie różne wiersze;
- **TOP n [PERCENT] –** zwracane jest *n* pierwszych wierszy (pierwsze *n* procent wierszy);
- **WITH TIES** – zwrócone może być więcej, niż *n* pierwszych wierszy, jeśli mają one wartość sortowanego pola taką samą, jak ostatni, *n-ty* wiersz; wymagana jest klauzula ORDER BY;
- **columns** – kolumny, które mają być zwrócone w wyniku; symbol “*” zwraca wszystkie kolumny;
- **AS column_heading** – przy wyświetlaniu wyników nazwa kolumny jest zmieniona na „column heading”;
- **INTO** – tworzona jest nowa tabela, do której kopiowane są wyniki zapytania;
- **FROM** – określna tabele (jedną lub więcej), z których mają być zwrócone wiersze;

Instrukcja SELECT Parametry (2)

- **INNER | {LEFT, RIGHT, FULL} OUTER | CROSS JOIN** – złączenie z inną tabelą – odpowienio: wewnętrzne, zewnętrzne (lewo- i prawostronne, pełne), krzyżowe;
- **WHERE** – określa wyrażenie ograniczające zbiór zwracanych wierszy;
- **search_condition** – wyrażenie, które musi być spełnione, aby dany wiersz był zwrócony.
- **GROUP BY** – określa kolumny (lub wyrażenia nieagregujące), względem których mają być grupowane wiersze w wyniku;
- **HAVING** – klauzula definiująca agregację lub warunek zapytania przy grupowaniu;
- **ORDER BY –** definiuje kolumny, według których ma być sortowany wynikowy zbiór wierszy;
- **ASC | DESC** – porządek sortowania – odpowiednio: rosnący (domyślnie) albo malejący.

SELECT: klauzula WHERE

Klauzula WHERE pozwala na zdefiniowanie warunku, który ogranicza wynikowy zbiór wierszy. Dopuszczalne są następujące operatory:

- **=, < > (różny), <, >, >=, <=** – zwyczajne operatory porównania;
- **BETWEEN wyrażenie1 AND wyrażenie2** – operator “pomiędzy” (łącznie z końcami przedziału);
- **IN | NOT IN (element1, element2, ..., elementN)** – wartość kolumny musi (albo nie może) należeć do podanego zbioru;
- **LIKE wyrażenie** – pozwala na porównania przybliżone; można je definiować za pomocą znaków wieloznacznych (ang. *wildcards*):
  - “%”d – dowolny znak lub cyfra;
  - “-”d – dowolny pojedynczy znak;
  - “[ ]” - dowolny znak umieszczony w nawiasach.

SELECT...FROM...WHERE Przykład

```
-- Proste wyszukiwanie przedmiotów, które zawierają w nazwie podany wzorzec.
CREATE PROC Przedmiot_szukanie
@Par_Nazwa varchar(128) = ' ' -- Wzorzec, który musi wystąpić w nazwie przedmiotu.
AS
BEGIN  -- Kolumny [KodPrzedmiotu] i [Forma] mają zmienione nagłówki przez AS.
SELECT KodPrzedmiotu AS [Kod], Nazwa, Semestr, ECTS,
            FormaZajec AS [Forma]
FROM Przedmiot
WHERE Nazwa LIKE '%' + @Par_Nazwa + '%'
ORDER BY KodPrzedmiotu ASC -- Sortowanie rosnące według pola [KodPrzedmiotu].
END
GO

-- Przykładowe wywołanie.
EXEC Przedmiot_szukanie 'sieci'
```

SELECT: kolumny wyliczeniowe

W instrukcji SELECT można podawać, oprócz nazw pól, także wyrażenia obliczane na podstawie pól i funkcji.

```
-- Zapytanie: "zwróć tytuł, imię i nazwisko pracowników – jako pojedyncze pole,
-- posortowane po nazwisku".
SELECT Tytul + Imie + Nazwisko AS [Pracownik]
FROM Pracownik
ORDER BY Nazwisko ASC
GO

-- Zapytanie: "zwróć numer indeksu, imię, nazwisko studenta i liczbę miesięcy,
-- które upłynęły od daty jego zapisania (staż)".
-- Staż jest obliczany na podstawie pola [DataZapisania] i bieżącej daty systemowej.
SELECT NrIndeksu, Imie, Nazwisko,
DATEDIFF(Month, DataZapisania, GETDATE( )) AS [Staz na uczelni]
FROM Student
ORDER BY [Staz na uczelni] DESC, Nazwisko ASC
GO
```

Klauzule GROUP BY i HAVING

- Klauzula GROUP BY umożliwia utworzenie podsumowań dla wybranych grup tabeli.
- Jeżeli wraz z GROUP BY ma być użyta klauzula WHERE, to musi być ona umieszczona przed grupowaniem. Klauzula WHERE nie może zawierać funkcji agregujących.
- Każde pole wymienione na liście SELECT, nie będące wyrażeniem agregującym, musi być umieszczone także w klauzuli GROUP BY, albo listy pól po SELECT i GROUP BY muszą być identyczne.
- Zalecane jest używanie wraz z GROUP BY klauzuli ORDER BY, aby grupowane wiersze zostały zwrócone w podanej kolejności. W przeciwnym razie kolejność ta nie jest określona.
- Klauzula HAVING pozwala na ograniczenie liczby grupowanych wierszy na podstawie podanego warunku, który może zawierać funkcje agregujące.

GROUP BY i HAVING Przykład

```
-- Procedura zliczająca studentów, mieszkających w poszczególnych miejscowościach.
-- Pod uwagę brane są tylko te miejscowości, których nazwa zawiera podany wzorzec
-- i w których mieszka co najmniej zadana liczba studentów.
CREATE PROCEDURE Student_zlicz_miejsc
@Par_Miejscowosc varchar(40) = ' ', -- Wzorzec nazwy miejscowości.
@Par_MinStud int = 0 -- Minimalna liczba studentów.
AS
BEGIN
SELECT Miejscowosc, COUNT(*) AS [Liczba studentow]
FROM Student
WHERE Miejscowosc LIKE '%' + @Par_Miejscowosc + '%'
GROUP BY Miejscowosc
HAVING COUNT(*) >= @Par_MinStud  -- Warunek minimalnej liczby studentów.
ORDER BY [Liczba studentow] DESC -- Sortowanie malejące po liczbie studentów.
END
GO
```

Plan wykładu

1. Polecenie SELECT.
2. **Złączenia tabel.**
3. Podzapytania.

Złączenia tabel

Złączenia służą do przedstawiania połączonych danych z więcej, niż jednej tabeli**.**

- Połączenie opiera się na porównaniu wartości kolumn w obu tabelach, które często opowiadają relacji kluczy głównych i obcych.
- Wyróżniane są następujące rodzaje złączeń:
  - wewnętrzne (INNER JOIN);
  - własne (SELF-JOIN) – odmiana złączenia wewnętrznego;
  - zewnętrzne (LEFT | RIGHT | FULL OUTER JOIN);
  - krzyżowe (CROSS JOIN).
- Są dwa rodzaje składni określającej złączenie:
  - warunek złączenia w klauzuli WHERE (składnia stara, niezalecana);
  - warunek złączenia w klauzuli FROM (nowa, zalecana składnia, zgodna ze standardem ANSI-92).

Złączenie tabel jako operacja algebry relacyjnej

Złączeniem (wewnętrznym) tabel *R* i *S* jest tabela *T*, w której wiersze mają połączone kolumny z tabel *R* i *S*. W tabeli *T* są tylko te wiersze, w których nastąpiło dopasowanie wartości określonych kolumn z obu tabel.

Tabela *R*

Imie	Nazwisko	KodPrzedmiotu
Jan	Kowalski	INF507
Andrzej	Jabłoński	INF517
Krzysztof	Nowak	INF517
Leszek	Morawski	INF517

N — 1

Tabela *S*

KodPrzedmiotu	Nazwa	ECTS
INF407	Bazy danych	4
INF507	Sieciowe bazy danych	4
INF517	Grafika komputerowa	5

Złączone tabele *R* i *S*

Imie	Nazwisko	KodPrzedmiotu	Nazwa	ECTS
Jan	Kowalski	INF507	Sieciowe bazy danych	4
Andrzej	Jabłoński	INF517	Grafika komputerowa	5
Krzysztof	Nowak	INF517	Grafika komputerowa	5
Leszek	Morawski	INF517	Grafika komputerowa	5

```
-- Odpowiednik w języku SQL (nowa składnia zgodna z ANSI-92):
SELECT Imie, Nazwisko, KodPrzedmiotu, Nazwa, ECTS
FROM R INNER JOIN S ON R.KodPrzedmiotu = S.KodPrzedmiotu
```

Złączenia tabel
Składnia nowa i stara

-- Oba poniższe zapytania zwracają identyczny wynik:
-- „zwróć dane przedmiotów i nazwiska ich opiekunów merytorycznych”.
-- Nowa składnia, zgodna z ANSI-92 (zalecana).
-- Warunek złączenia jest podany w klazuli FROM.

```
SELECT KodPrzedmiotu AS [Kod], Nazwa,  Nazwisko AS [Opiekun merytoryczny]
FROM Przedmiot INNER JOIN Pracownik
    ON Przedmiot.Opiekun = Pracownik.IdPracownika
```

-- Składnia stara (niezalecana).
-- Warunek złączenia jest podany w klazuli WHERE.

```
SELECT KodPrzedmiotu AS [Kod], Nazwa, Nazwisko AS [Opiekun merytoryczny]
FROM Przedmiot, Pracownik
WHERE Przedmiot.Opiekun = Pracownik.IdPracownika
```

Złączenia tabel
Zalety nowej składni (ANSI)

- Przejrzystość i zrozumiałość – zwłaszcza przy złączeniach wielu tabel jednocześnie.
- Zapewnienie podwójnych złączeń zewnętrznych, których nie obsługuje stara składnia.
- Możliwość użycia poleceń dla optymalizatora zapytań, określających sposób złączenia tabel. Opcja ta nie jest dostępna w starej składni.
- W kolejnych wersjach MS SQL Server stara składnia może nie być obsługiwana.

Złączenie wewnętrzne: INNER JOIN

Jest to najczęściej stosowane złączenie. Wiersze zwracane w wyniku złączenia wewnętrznego mają taką samą wartość określonej kolumny w obu łączonych tabelach.

```
-- Zapytanie: "zwróć studentów z podanej grupy".
CREATE PROCEDURE Student_Grupa_szukanie
@Par_NrGrupy varchar(7) = ' '
AS
BEGIN
SELECT NrIndeksu AS [Numer indeksu], Imie, Nazwisko, NrGrupy AS [Grupa]
FROM Student INNER JOIN Grupa -- Nazwy tabel i rodzaj ich złączenia.
   ON Student.IdGrupy = Grupa.IdGrupy -- Opis kolumn łączących.
-- Nie wszystkie kolumny łączące nie muszą występować na liście SELECT.
WHERE NrGrupy LIKE '%' + @Par_NrGrupy + '%'
ORDER BY NrGrupy ASC, Nazwisko ASC
END
```

Złączenie własne: SELF JOIN

Złączenie własne jest odmianą złączenia wewnętrznego, w której dana tabela jest łączona z samą sobą.

```
-- Zapytanie: "zwróć studentów, którzy mają takie samo nazwisko".
CREATE PROCEDURE Student_Nazwisko
@Par_Nazwisko varchar(7) = ' '
AS
BEGIN
SELECT stud1.NrIndeksu, stud1.Imie, stud1.Nazwisko,
stud2.NrIndeksu, stud2.Imie, stud2.Nazwisko, stud1.Miejscowosc
FROM Student stud1 INNER JOIN Student stud2 -- Wykorzystywane są aliasy.
    ON stud1.Nazwisko = stud2.Nazwisko -- Dopasowywane kolumny.
WHERE (stud1.Nazwisko LIKE '%' + @Par_Nazwisko + '%')
AND   (stud1.NrIndeksu < stud2.NrIndeksu) -- Warunek zapobiegający powtórzeniom.
ORDER BY stud1.Nazwisko ASC
END
```

Złączenie zewnętrzne: OUTER JOIN

Złączenie zewnętrzne zwraca wszystkie wiersze z tabeli, która została określona jako zewnętrzna (LEFT OUTER, RIGHT OUTER, FULL OUTER), nawet jeśli wartości łączonych kolumn nie są równe.

- Wiersze tabeli zewnętrznej, dla których nie istnieje dopasowanej wartości z tabeli wewnętrznej, zawierają w kolumnie łączonej wartość NULL.
- Złączenie lewostronne (LEFT OUTER JOIN) i prawostronne (RIGHT OUTER JOIN) nie różnią się między sobą poza kolejnością specyfikacji tabeli zewnętrznej i wewnętrznej.
- Złączenie zewnętrzne pełne (FULL OUTER JOIN) zwraca wszystkie odpowiadające sobie wiersze z łączonych tabel, a następnie wiersze, dla których dopasowanie nie zachodzi.

Złączenie zewnętrzne: OUTER JOIN – przykład

```
-- Zwracanie grup i ich starostów (jeśli są ustanowieni).
CREATE PROC Grupa_szukanie
@Par_RokRozpoczecia smallint = 1990 -- Rok rozpoczęcia studiów przez grupę.
AS
BEGIN
-- Imię i nazwisko starosty grupy są sklejane w pojedynczy ciąg znaków
-- po prawostronnym obcięciu spacji przez funkcję RTRIM( ).
SELECT NrGrupy AS [Nazwa grupy], RokRozpoczecia AS [Rok rozpoczecia],
RTRIM (Imie) + ' ' + RTRIM (Nazwisko) AS [Starosta], NrIndeksu AS [Nr indeksu]
-- Lewostronne złączenie zewnętrzne zwraca wszystkie grupy spełniające warunek.
-- WHERE, nawet jeśli nie mają one starostów (wówczas są to pola NULL).
FROM Grupa LEFT OUTER JOIN Student
      ON Grupa.Starosta = Student.NrIndeksu
WHERE RokRozpoczecia >= @Par_RokRozpoczecia
END
```

Złączenie krzyżowe: CROSS JOIN

Wynikiem złączenia krzyżowego są wiersze, zawierające wszystkie możliwe kombinacje wartości określonych kolumn (iloczyn kartezjański tabel).
Liczba zwracanych wierszy może być bardzo duża.

```
-- Złączenie krzyżowe.
-- Zapytanie: "wypisz wszystkie kombinacje imion i nazwisk studentów".
CREATE PROCEDURE Student_Imie_Nazwisko
AS
BEGIN
SELECT DISTINCT stud1.Imie, stud2.Nazwisko
FROM Student stud1 CROSS JOIN Student stud2
ORDER BY stud2.Nazwisko, stud1.Imie
END
GO
```

Plan wykładu

1. Polecenie SELECT.
2. Złączenia tabel.
3. **Podzapytania.**

Podzapytania

Podzapytanie jest zapytaniem SQL, które występuje wewnątrz innego zapytania i może być użyte w miejscu wyrażenia.

- Wynikiem podzapytania może być pojedynczy wiersz lub kolumna tabeli.
- Wyniki podzapytania i złączenia są często takie same. Złączenie jest jednak bardziej wydajne niż podzapytanie. Wyjątkiem jest sytuacja, w której mają być usunięte powtarzające się wiersze (DISTINCT). Wówczas podzapytanie użyte z NOT EXISTS działa szybciej.
- Podzapytania są konieczne w przypadku zapytań, które:
  - zawierają warunki nieistnienia lub niewchodzenia w skład (NOT EXISTS, NOT IN);
  - wymagają użycia funkcji agregującej w klauzuli WHERE.

Podzapytania – przykłady (1)

```
-- Poniższe zapytanie można realizować alternatywnie za pomocą złączeń
-- (jest ono tak zaimplementowane w procedurze Student_Grupa_szukanie).
-- Zapytanie: "zwróć studentów z podanej grupy".

CREATE PROCEDURE Student_Grupa_podzapytanie
@Par_NrGrupy varchar(7) = ''
AS
BEGIN
SELECT NrIndeksu AS [Numer indeksu], Imie, Nazwisko
FROM Student
WHERE IdGrupy IN (SELECT IdGrupy
                  FROM Grupa
                  WHERE NrGrupy LIKE '%' + @Par_NrGrupy + '%')
ORDER BY Nazwisko ASC, Imie ASC
END
GO
```

Podzapytania – przykłady (2)

```
-- Poniższego zapytania NIE MOŻNA realizować alternatywnie
-- za pomocą złączeń, ponieważ odwołuje się do warunków nieistnienia.
-- Zapytanie: "zwróć studentów, którzy nie wnieśli żadnej opłaty od podanej daty".

CREATE PROCEDURE Student_Oplata_podzapytanie
@Par_DataWplaty datetime
AS
BEGIN
SELECT DISTINCT NrIndeksu AS [Numer indeksu], Imie, Nazwisko
FROM Student
WHERE NrIndeksu NOT IN    (SELECT NrIndeksu
                           FROM Oplata
                           WHERE DataWplaty >= @Par_DataWplaty)
ORDER BY Nazwisko ASC, Imie ASC
END
GO
```

Podzapytania – przykłady (3)

```
-- Poniższego zapytania NIE MOŻNA realizować alternatywnie
-- za pomocą złączeń, ponieważ wykorzystuje ono funkcję agregującą
-- w klauzuli WHERE.
-- Zapytanie: "zwróć pracowników, których wynagrodzenie jest niższe
--                  od średniej płacy wszystkich pracowników".

CREATE PROCEDURE Pracownik_Placa_podzapytanie
AS
BEGIN
SELECT Tytul, Imie, Nazwisko, Pensja
FROM Pracownik
WHERE Pensja <    (SELECT AVG(Pensja)
                   FROM Pracownik)
ORDER BY Pensja ASC, Nazwisko ASC, Imie ASC
END
GO
```

Podzapytania jako tabele pochodne

W zapytaniu można zastosować tabelę pochodną, która jest zapytaniem zagnieżdżonym, zawartym w klauzuli FROM.

- Do tabeli pochodnej można odwoływać się przez alias i można traktować ją jak normalną tabelę lub jak dynamiczny widok.
- Podzapytanie może zawierać agegacje (np. COUNT), grupowania (GROUP BY) i sumy wierszy (UNION).

Podzapytania jako tabele pochodne – przykład

```
-- Zapytanie:
-- "wypisz wszystkie kombinacje imion i nazwisk studentów i pracowników".

CREATE PROCEDURE Student_Prac_podzapytanie
AS
BEGIN
SELECT DISTINCT stud1.Imie, stud2.Nazwisko
FROM    (SELECT Imie FROM Student
          UNION
         SELECT Imie FROM Pracownik) stud1
CROSS JOIN
          (SELECT Nazwisko FROM Student
           UNION
           SELECT Nazwisko FROM Pracownik) stud2
ORDER BY stud2.Nazwisko, stud1.Imie
END
GO
```

Literatura

1. BEYNON-DAVIES P., *Systemy baz danych – nowe wydanie*, WNT, Warszawa 2003.
2. DATE C. J., DARWEN H., *SQL. Omówienie standardu języka*, WNT, Warszawa 2000 (książka dostępna w bibliotece WSIZ "Copernicus").
3. MICROSOFT, *Books On-Line* – dokumentacja systemu *MS SQL Server*, Microsoft Corp. 1988 – 2000.
4. RANKINS R., JENSEN P., BERTUCCI P., *Microsoft SQL Server 2000. Księga eksperta.*, HELION, Gliwice 2003 (książka dostępna w bibliotece WSIZ „Copernicus").
5. Strona MSDN: http://msdn.microsoft.com.
6. WAYMIRE R., SAWTELL R., *MS SQL Server 2000 dla każdego*, HELION, Gliwice 2002 (książka dostępna w bibliotece WSIZ "Copernicus").

Bazy danych

Wykład 5_3

Dziękuję za uwagę !