Baza danych Access

Instrukcja - SELECT

Opracował: Andrzej Nowak

Bibliografia:

Access Baza Danych. Projektowanie i programowanie; S. Roman; (wyd. Helion; 2001r.)

Instrukcja SELECT

składnia:

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Uwaga: Słowo kluczowe `SELECT` jest trochę niefortunne, ponieważ jest ono odpowiedzialne za projekcję (operacje algebry relacyjnej), a nie selekcję.

Selekcją zajmuje się klauzula `WHERE`.

Parametr Predicate

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Parametr Predicate wskazuje, w jaki sposób obsłużyć duplikaty w zwróconych wierszach.

Parametr ten przyjmuje następujące wartości:

```
ALL, TOP, DISTINCTROW, DISTINCT
```

Opcja domyślna ALL zwraca wszystkie wiersze, w tym duplikaty. Jeżeli wśród zwróconych wierszy znajduje się więcej niż jeden wiersz o tych samych wartościach we wszystkich kolumnach, opcja `DISTINCT` zwraca tylko pierwszy.

Opcja TOP numer `[NUMBER]` lub `TOP` procent `[PERCENT]` zwraca określoną liczbę (lub procent) wierszy licząc od góry, przy czym porządek sortowania określa klauzula `ORDER BY`

Opcja DISTINCTROW występuje tylko w programie Microsoft Access. Sprawia, że kwerenda zwraca specyficzne rekordy, a nie specyficzne wartości.

Przykład:

Jeżeli dziesięciu klientów nosi nazwisko Kowalski, kwerenda oparta na instrukcji

`"SELECT DISTINCTROW Nazwisko FROM Klienci"`

zwraca wszystkie 10 rekordów z wartością Kowalski w polu Nazwisko.

Słowo kluczowe `DISTINCTROW` zostało dodane do języka Access SQL po to, aby umożliwić obsługę półsprzężeń, takich jak sprzężenia jeden do wielu, w których wszystkie pola wynikowe pochodzą z tabeli po stronie "jeden".

Opcja `DISTINCTROW` jest opcją domyślną w kwerendach programu Access. W kwerendach, w których nie wywołuje efektu, jest ignorowana.

Słowa kluczowego `DISTINCTROW` nie należy usuwać z okna dialogowego `SQL`.

Opcja `DISTINCTROW` jest stosowana tylko wtedy, kiedy klauzula FROM używa więcej niż jednej tabeli.

Rozważmy następującą instrukcję:

```
SELECT ALL WydNazwa
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydID;
```

Ponieważ istnieje wiele książek wydanych przez to samo wydawnictwo, tabela wynikowa Wszystkie zawiera wiele powtarzających się nazw wydawnictw.

Tabela: `Wszystkie`

```
WydNazwa
-------------------------
Small House
Small House
Small House
Small House
Big House
Big House
Big House
Big House
Big House
Big House
Alpha Press
Alpha Press
Alpha Press
Alpha Press
-------------------------
```

Aby usunąć powtarzające się nazwy wydawnictw, należy dodać słowo kluczowe DISTINCT.

Opcja `DISTINCT` służy do usuwania powtarzających się danych w danej kolumnie kwerendy

Przykład:

```
SELECT DISTINCT WydNazwa
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydID;
```

wynikiem jest Tabela: `Wszystkie`

```
WydNazwa
-------------------------------------
Small House
Big House
Alpha Press
-----------------------------------------
```

Rozważmy teraz co się stanie, jeżeli tabela `Wydawnictwa` zostanie zmodyfikowana poprzez dodanie nowego wydawnictwa o tej samej nazwie co wydawnictwo istniejące, ale o innym identyfikatorze `WydId` i telefonie `WydTelefon`:

Zmodyfikowana tabela `Wydawnictwa`

```
WydId |     WydNazwa    |     WydTelefon
-------------------------------------------------------------------------
1     |     Big House   |     123-456-7890
2     |     Alpha Press      |     999-999-9999
3     |     Small House      |     714-000-0000
4     |     Small House      |     555-123-1111
-------------------------------------------------------------------------
```

Poprzednia instrukcja DISTINCT utworzy tę samą tabelę wynikową co poprzednio, w ten sposób pomijając nowe wydawnictwo.

Potrzeba więc kryterium selekcji, które zwróci obie nazwy wydawnictw, po prostu dlatego, że pochodzą one z różnych wierszy tabeli Wydawnictwa.

To jest właśnie zadanie dla opcji DISTINCTROW.

Zatem instrukcja:

```
SELECT DISTINCTROW WydNazwa
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydID;
```

tworzy tabelę wynikową:

```
WydNazwa
-------------------------------------
Small House
Big House
Alpha Press
Small House
-------------------------------------
```

Zasada działania opcji DISTINCTROW

Rozważmy następujący szablon instrukcji SQL:

```
SELECT DISTINCTROW lista_kolumn

FROM sprzężenie_tabel
```

Parametr lista_kolumn - reprezentuje listę kolumn, o które prosi instrukcja.

Parametr sprzężenie_tabel - reprezentuje sprzężenie tabel. Tabelę reprezentowaną przez sprzężenie_tabel

nazywamy tabelą zwróconą, jeżeli przynajmniej jedna z jej kolumn jest reprezentowana przez parametr lista_kolumn.

Zatem w intrukcji:

```
SELECT DISTINCTROW WydNazwa
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydID;
```

tabela Wydawnictwa jest tabelą zwróconą, tabela Książki nie.

Oto jak działa opcja DISTINCTROW:

1. Utwórz sprzężenia opisane w parametrze sprzężenie_tabel.

2. Wykonaj projekcję tabeli wynikowej na wszystkie kolumny wszystkich tabel zwracanych (a nie tylko na kolumny żądane). Innymi słowy, usuń wszystkie kolumny nie będące częścią tabeli zwracanej.

3. Usuń wszystkie wiersze będące duplikatami. Dwa wiersze są duplikatami, jeżeli składają się z tych samych wierszy każdej tabeli wynikowej. Warto zauważyć, że nie wartości są porównywane, ale wiersze.

   Ten punkt jest potrzebny, ponieważ w programie Access tabela może zawierać dwa wiersze o identycznych wartościach.

Przykład:

Rozważmy następujące tabele:

```
A1    |     A2
---------------------------------
a1    |     x
a2    |     łącze
a3    |     łącze
---------------------------------

B1    |     B2    |     B3
-------------------------------------
b1    |     y     |     z
b2    |     łącze |     łącze2
-------------------------------------

C1    |     C2    |     C3
----------------------------------------
c1    |     t     |     łącze2
c2    |     v     |     łącze2
c3    |     a     |     x
----------------------------------------
```

Instrukcja:

```
SELECT *
FROM
(Temp1 INNER JOIN Temp2 ON Temp1.A2 = Temp2.B2)
INNER JONIN Temp3 ON Temp2.B3 = Temp3.C3;
```

tworzy tabelę wynikową `Wszystkie`:

```
  A1  |  A2    |  B1   |  B2    |  B3   |  C1      |  C2 |  C3
-------------------------------------------------------------------------
  a3  |  łącze|  b2    |  łącze|  łącze2|  c2      |  v  |  łącze2
  a3  |  łącze|  b2    |  łącze|  łącze2|  c1      |  t  |  łącze2
  a2  |  łącze|  b2    |  łącze|  łącze2|  c2      |  v  |  łącze2
  a2  |  łącze|  b2    |  łącze|  łącze2|  c1      |  t  |  łącze2
-------------------------------------------------------------------------
```

Dodajmy teraz słowo kluczowe `DISTINCTROW` i wybierzmy jedną kolumnę:

```
SELECT DIDTINCTROW A1
FROM
(Temp1 INNER JOIN Temp2 ON Temp1.A2 = Temp2.B2)
INNER JONIN Temp3 ON Temp2.B3 = Temp3.C3;
```

Rozważmy teraz projekcję na wiersze jedynej tabeli zwróconej:

```
  A1  |  A2
------------------
  a3  |  łącze
  a3  |  łącze
  a2  |  łącze
  a2  |  łącze
------------------
```

Widać, że pierwsze dwa wiersze tej tabeli są takie same, jak wiersz tabeli Wszystkie, więc w ostatecznej tabeli wynikowej utworzą one tylko jeden wiersz. To samo dotyczy dwóch ostatnich wierszy.

Stąd tabela wynikowa wygląda w następujący sposób:

```
  A1
--------
  a2
  a3
--------
```

Zmieńmy instrukcję SQL w taki sposób, żeby pytała również o kolumnę z tabeli C, w ten sposób czyniąc ją również tabelą zwracaną:

```
SELECT DIDTINCTROW A1, C1
FROM
(Temp1 INNER JOIN Temp2 ON Temp1.A2 = Temp2.B2)
INNER JONIN Temp3 ON Temp2.B3 = Temp3.C3;
```

Projekcja na wiersze tabeli zwracanej wygląda teraz tak:

```
  A1  |  A2    |    C1 |  C2 |  C3
-----------------------------------------
  a3  |  łącze|    c2  |  v  |  łącze2
  a3  |  łącze|    c1  |  t  |  łącze2
  a2  |  łącze|    c2  |  v  |  łącze2
  a2  |  łącze|    c1  |  t  |  łącze2
-----------------------------------------
```

Pary wierszy są specyficzne. Zwróćmy uwagę, że:

wiersz 1 pochodzi z wiersza 1 tabeli A, a wiersz 2 z tabeli C

wiersz 2 pochodzi z wiersza 1 tabeli A, a wiersz 1 z tabeli C

wiersz 3 pochodzi z wiersza 2 tabeli A, a wiersz 2 z tabeli C

wiersz 4 pochodzi z wiersza 2 tabeli A, a wiersz 1 z tabeli C

Zatem tabela zwracana zawiera wszystkie wiersze:

```
  A1  |  C1
------------------
  a2  |  c1
  a2  |  c2
  a3  |  c1
  a3  |  c2
------------------
```

Rozważmy, co się stanie, jeżeli zmienimy trzeci wiersz tabeli A na:

```
  A1  |  A2
------------------
  a1  |  x
  a2  |  łącze
  a2  |  łącze
------------------
```

Po uruchomieniu pierwszej instrukcji DISTINCTROW:

```
SELECT DIDTINCTROW A1
FROM
(Temp1 INNER JOIN Temp2 ON Temp1.A2 = Temp2.B2)
INNER JONIN Temp3 ON Temp2.B3 = Temp3.C3;
```

otrzymamy:

```
  A1
--------
  a2
  a2
--------
```

Porównując ten wynik z poprzednią tabelą wynikową można zauważyć, że klauzula `DISTINCTROW A1` podkreśla fakt, że nawet jeżeli drugi i trzeci wiersz są identyczne co do wartości, to są innymi wierszami, więc każdy z nich ma swój udział w powstaniu ostatecznej tabeli wynikowej.

Jeżeli zastąpimy słowo kluczowe `DISTINCTROW` słowem `DISTINCT`, tabela wynikowa będzie zawierać tylko jeden wiersz, ponieważ podstawą do porównań są wartości każdego wiersza.

Oczywiście sprawa wyglądałaby inaczej, gdyby wszystkie tabela miały klucz, ponieważ wtedy wartość wiersza określałby ten klucz.

Teraz widać wyraźnie, dlaczego nie należy tworzyć dwóch różnych wierszy o tych samych wartościach kolumn, mimo że Access pozwala na to (nie jest to jednak zgodne z wymogami prawdziwych relacyjnych baz danych).

Zwróćmy uwagę co się dzieje, jeżeli wszystkie tabele wskazane w parametrze `sprzężenie_tabel` są tabelami zwracanymi.

Ma to miejsce na przykład wtedy, kiedy parametr `sprzężenie_tabel` wskazuje tylko jedną tabelę. W takim przypadku projekcja nie daje żadnego efektu, a ponieważ każdy wiersz tabeli wynikowej `sprzężenie_tabel` musi pochodzić z niepowtarzalnej kombinacji wierszy tabel wynikowych, wnioskujemy, że opcja `DISTINCTROW` ma dokładnie taki sam efekt co opcja `ALL` - innymi słowy, opcja `DISTINCTROW` jest ignorowana.

Porównując opcję `DISTINCTROW` z opcją `DISTINCT` możemy stwierdzić, że różnią się one tylko tym, że instrukcja `DISTINCT` zwraca różne wartości, a nie wartości z różnych wierszy.

Jeżeli jednak żądane kolumny każdej zwracanej tabeli jednoznacznie identyfikują swoje wiersze, będą to te same wartości.

Zilustrujmy to zagadnienie na przykładzie tabeli `Wydawnictwa`

Załóżmy, że zwracamy klucz `WydId` dla tabeli `Wydawnictwa`:

```
SELECT DISTINCTROW WydId, WydNazwa
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId;
```

Tabela wynikowa zwróci wszystkie wiersze tabeli Wydawnictwa posiadające co najmniej jedną książkę w tabeli `Książki`

```
WydID |  WydNazwa
-----------------------------------
3     |  Small House
1     |  Big House
2     |  Alpha Press
4     |  Small House
---------------------------------
```

Jest to tak naprawdę półsprzężenie:

```
Wydawnictwa półsprzężenie (Wydawnictwa.WydId = Książki.WydId) Książki
```

Półsprzężenie jest projekcją sprzężenia na jedną z tabel (w tym przypadku na tabelę `Wydawnictwa`).

Zatem, jak przyznaje sam Microsoft, zadanie opcji `DISTINCTROW` polega na zwracaniu półsprzężeń, które można aktualizować.

Oczywiście ta sama instrukcja z opcją `DISTINCT` zamiast opcji `DISTINCTROW` zwróci tę samą tabelę wynikową.

Jest tu jednak duża różnica. Ponieważ instrukcja `DISTINCT` może kompletnie ukryć pochodzenie wartości zwracanych, Access nie może pozwolić, aby taką tabelę wynikową można było aktualizować.

Rozważmy na przykład tabelę `DISTINCT` omawianą wcześniej

```
WydNazwa
-------------------------------------
Small House
Big House
Alpha Press
-----------------------------------------
```

Modyfikacja nazwy `Small House` w tabeli wynikowej byłaby katastrofą, ponieważ nie byłoby wiadomo, które wydawnictwo `Small House` zostało zmodyfikowane.

Z drugiej strony tabela wynikowa `DISTINCTROW` ma "przedstawiciela" każdego wiersza tabeli Wydawnictwa

```
WydNazwa
-------------------------------------
Small House
Big House
Alpha Press
Small House
-----------------------------------------
```

Chociaż modyfikacja tej konkretnej tabeli to może nienajlepszy pomysł, ponieważ nie można powiedzieć, który `Small House` jest który, można na przykład zmienić obie nazwy.

Access nie dopuszcza aktualizacji tabeli wynikowej w przypadku instrukcji `DISTINCT`, ale dopuszcza aktualizację tabeli wynikowej w przypadku instrukcji `DISTINCTROW`.

Słowo kluczowe `DISTINCTROW` jest używane przez Access domyślnie, kiedy kwerendę tworzymy za pomocą okna projektowego kwerend.

Parametr ReturnColumnDescription

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Parametr `ReturnColumnDescription` opisuje kolumny lub kombinację kolumn, które mają zostać zwrócone. Parametr ten przyjmuje następujące wartości:

- * (znak wskazuje wszystkie kolumny)
- nazwa kolumny
- wyrażenie złożone z nazw kolumn umieszczonych w nawiasach kwadratowych wraz z łańcuchami znaków i operatorami łańcuchów znaków, na przykład `[WydId] & " - " & [Tytuł]`

Zwróć uwagę, że zgodnie ze składnią instrukcji SELECT, parametr `ReturnColumnDescription` można powtórzyć dowolną liczbę razy.

Jeżeli dwie zwrócone kolumny (różnych tabel) mają tę samą nazwę, nazwy kolumn należy kwalifikować za pomocą nazw tabel.

Na przykład aby kwalifikować nazwę kolumny `WydId`, piszemy `Książki.WydId` oraz `Wydawnictwa.WydId`. Aby wskazać wszystkie kolumny tabel Książki piszemy `Książki.*`.

Wreszcie każdy parametr `ReturnColumnDescription` może kończyć się klauzulą:

```
[AS AliasName]
```

dzięki czemu zwracana kolumna otrzymuje nową nazwę.

Na przykład poniższa instrukcja:

```
SELECT DISTINCTROW
[ISBN] & "z wydawnictwa " & [WydNazwa] AS [ISBN oraz WydNazwa]
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId;
```

zwraca jednokolumnową tabelę wynikową `ISBN-Wyd`

```
ISBN oraz WydNazwa
-------------------------------------
0-12-345678-9 z wydawnictwa Small House
0-11-345678-9 z wydawnictwa Small House
0-321-32132-1 z wydawnictwa Small House
0-55-123456-9 z wydawnictwa Small House
0-12-333433-3 z wydawnictwa Big House
0-103-45678-9 z wydawnictwa Big House
0-91-335678-7 z wydawnictwa Big House
0-99-999999-9 z wydawnictwa Big House
1-22-233700-0 z wydawnictwa Big House
1-1111-1111-1 z wydawnictwa Big House
0-91-045678-5 z wydawnictwa Alpha Press
0-55-555555-9 z wydawnictwa Alpha Press
0-99-777777-7 z wydawnictwa Alpha Press
0-123-45678-0 z wydawnictwa Alpha Press
-------------------------------------
```

Dzięki opcji `AS AliasName` możemy nie tylko nadać nazwę „złożonej kolumnie", ale zmienić nazwy kolumn o identycznych nazwach bez konieczności kwalifikacji nazw kolumn.

Klauzula

FROM TableExpression

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Klauzula FROM określa tabele (lub kwerendy), z których instrukcja `SELECT` ma pobrać wiersze.

Wyrażenie TableExpression może być nazwą tabeli, nazwami tabel oddzielonymi przecinkami lub sprzężeniem. Wyrażenie to może też zawierać składnie `AS AliasName`, dzięki której można utworzyć alias nazwy tabeli.

Jeżeli nawy tabel są oddzielone przecinkami w klauzuli `FROM`, instrukcja tworzy iloczyn kartezjański .

Na przykład instrukcja:

```
SELECT *
FROM Autorzy, Wydawnictwa;
```

tworzy iloczyn kartezjański dwóch tabel.

Klauzula

WHERE RowCondition

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Wyrażenie RowCondition określa, jakie wiersze mają się znaleźć w tabeli wynikowej.

Wyrażenie to może zawierać:

- nazwy kolumn,
- stałe,
- operatory arytmetyczne (=, <, >, <=, >=, <>, BETWEEN),
- operatory logiczne (AND, OR, XOR, NOT, IMP),
- funkcje.

Oto kilka przykładów:

- `WHERE Tytuł LIKE 'F*'`
- `WHERE Len(Trim(Tytuł)) > 10`
- `WHERE InStr(Tytuł, "Wiatr") > 0 AND Len(Trim(Tytuł)) > 10`
- `WHERE DataSprzedaży = #25/01/01#`

Uwaga: daty umieszczamy między znakami funta (#) , natomiast łańcuchy znaków ujmujemy w cudzysłów (" ").

Klauzula

GROUP BY GroupByCriteria

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Klauzula GROUP BY pozwala na tworzenie grup rekordów w celu obliczenia wartości

`funkcji agregującej(Avg, Count, Min, Max, First, Last, StDev, StDevP, Var` oraz `VarP).`

Spełnia podobne zadanie, co kwerenda sumująca.

Wyrażenie `GroupByCriteria` może zawierać do dziesięciu nazw kolumn. Porządek nazw kolumn określa poziomy grup od najwyższego do najniższego.

Na przykład poniższa instrukcja wyświetla listę nazw wydawnictw wraz z najniższą ceną książki (z tabeli Książki) każdego z nich:

```
SELECT Wydawnictwa.WydNazwa, MIN(Cena) AS [Najniższa cena]
FROM Wydawnictwa INNET JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
GROUP BY Wydawnictwa.WydNazwa;
```

Tabela wynikowa:

```
WydNazwa         |  Najniższa cena
-----------------------------------
Alpha Press      |  12,00 zł
Big House        |  15,00 zł
Small House      |  22,95 zł
-----------------------------------------------------------------
```

Klauzula

HAVING GroupCriteria

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Klauzuli HAVING używamy wraz z opcją **GROUP BY**, aby określić kryterium (w sensie funkcji agregujących) służące do wskazania wierszy, które chcemy wyświetlić.

Na przykład poniższa instrukcja jest rozwinięciem instrukcji poprzedniej – dodaliśmy opcję `HAVING` ograniczającą tabelę zwracaną do wydawnictw, których najniższa cena książki wynosi mniej niż 20,00 zł:

```
SELECT Wydawnictwa.WydNazwa, MIN(Cena) AS [Najniższa cena]
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
GROUP BY Wydawnictwa.WydNazwa
HAVING MIN(Cena) < 20,00;
```

Tabela wynikowa:

```
WydNazwa          |  Najniższa cena
-----------------------------------
Alpha Press       |  12,00 zł
Big House         |  15,00 zł
-----------------------------------------------------------------
```

Zwróć uwagę, że klauzula `WHERE` określa wiersze należące do grupy i stąd ma wpływ na wartość funkcji agregujących, natomiast klauzula `HAVING` ma jedynie wpływ na to, które wartości zostaną wyświetlone.

Klauzula

ORDER BY OrderByCriteria

```
SELECT [Predicate] ReturnColumnDescription
FROM TableExpression
[WHERE RowCondition]
[GROUP BY GroupByCriteria]
[HAVING GroupCriteria]
[ORDER BY OrderByCriteria]
```

Klauzula `ORDER BY` określa porządek wierszy w tabeli wynikowej.

Parametr `OrderByCriteria` przyjmuje postać:

```
OrderByCriteria ::= {ColumnName [ASC | DESC]},  …
```

Innymi słowy, jest to lista kolumn określających porządek sortowania. Najpierw wiersze są sortowane według pierwszej wskazanej kolumny, a następnie wiersze o identycznych wartościach w pierwszej kolumnie są sortowane według wartości drugiej wskazanej kolumny, itd.

Instrukcja UNION

Instrukcja UNION tworzy sumę dwóch lub więcej tabel. Oto jej składnia:

```
[TABLE] Query
{UNION [ALL] [TABLE] Query}, …
```

Parametr Query reprezentuje instrukcję SELECT, nazwę procedury składniowej lub nazwę tabeli składowanej poprzedzonej słowem kluczowym TABLE.

Opcja ALL sprawia, że Access dodaje wszystkie rekordy. Bez tej opcji Access nie dodaje powtarzających się wierszy. Opcja ALL poprawia działanie instrukcji, jest więc zalecana nawet w przypadku braku wystąpienia powtarzających się wierszy.

Przykład:

Poniższa instrukcja tworzy sumę wszystkich wierszy tabeli Książki oraz wierszy tabeli Nowe_Książki, zawierających cenę większą od 25,00 zł (Cena > 25,00), stosując tabelę wynikową według kolumny Tytuł.

```
TABLE Książki
UNION ALL
SELECT * FROM Nowe_Książki WHERE Cena > 25,00
ORDER BY Tytuł;
```

UWAGA:

- Wszystkie kwerendy w operacji UNION muszą zwracać tę samą liczbę pól. Jednak pola te nie muszą mieć tego samego rozmiaru, ani typu danych.
- Składanie kolumn odbywa się według ich porządku w klauzulach kwerend, nie według ich nazw.
- W pierwszej instrukcji SELECT (jeżeli taka istnieje) można używać aliasów, aby zmienić nazwy zwracanych kolumn.
- Klauzulę ORDER BY można zastosować na końcu ostatniej kwerendy, aby nadać określony porządek zwracanym danym. Należy używać nazw kolumn z pierwszej kwerendy.
- Klauzule GROUP BY i HAVING można użyć w każdym argumencie kwerendy w celu grupowania zwracanych danych.
- Tablicy wynikowej instrukcji UNION nie można aktualizować.
- Instrukcja UNION nie jest częścią standardu SQL-92.

Instrukcja UPDATE

Instrukcja UPDATE – równoważna z kwerendą Update – aktualizuje dane tabeli lub tabel.

Oto jej składnia:

```
UPDATE TableName  |  QueryName
SET NewValueExpression,  …..
WHERE Criteria;
```

Klauzula WHERE ogranicza aktualizację do wybranych wierszy.

Przykład:

Poniższa instrukcja aktualizuje kolumnę Cena tabeli Książki wstawiając nowe ceny z tabeli Nowe_Ceny posiadającej kolumnę ISBN i kolumnę Cena:

```
UPDATE
Książki INNER JOIN Nowe_Ceny ON Książki.ISBN = Nowe_Ceny.ISBN
SET Książki.Cena = Nowe_Ceny.Cena
WHERE Książki.Cena <> Nowe_Ceny.Cena;
```

Zwróćmy uwagę, że instrukcja UPDATE nie tworzy tabeli wynikowej. Aby określić, które wiersze zostaną zaktualizowane, należy uruchomić odpowiednią kwerendę SELECT:

```
SELECT * FROM
Książki INNET JOIN Nowe_Ceny ON Książki.ISBN = Nowe_Ceny.ISBN
WHERE Książki.Cena <> Nowe_Ceny.Cena;
```

Instrukcja DELETE

Instrukcja DELETE – równoznaczna z kwerendą Delete – usuwa wiersze z tabeli.

Oto jej składnia:

```
DELETE
FROM TabelaName
WHERE Criteria
```

Parametr Criteria określa, które wiersze mają zostać usunięte.

Instrukcja DELETE może usunąć wszystkie dane z tabeli, ale nie strukturę tabeli. Aby usunąć strukturę tabeli, należy użyć instrukcji DROP.

Instrukcji DELETE usuwa rekordy tabeli mające relację jeden-do-wielu.

Jeżeli jest włączone usuwanie kaskadowe, kiedy usuniemy wiersz znajdujący się po stronie „jeden” relacji, wszystkie związane wiersze znajdujące się po stronie „wiele” również zostaną usunięte.

Wynik działania instrukcji DELETE jest nieodwracalny.

Pamiętajmy o wykonaniu kopii zapasowych tabel zanim zaczniemy usuwać z nich dane. Aby określić, które wiersze zostaną usunięte, można uruchomić odpowiednią kwerendę SELECT.

Instrukcja INSERT INTO

Instrukcja INSERT INTO wstawia nowe wiersze do tabeli. Można to zrobić podając wartości nowego wiersza za pomocą następującej składni:

```
INSERT INTO Target [(FieldName,     ……    )]
VALUE (Value1,   ….   )
```

Jeżeli nie podamy nazwy pola (lub nazw pól), należy podać wartości dla każdego pola tabeli.

Oto kilka przykładów użycia instrukcji `INSERT INTO`:

Poniższa instrukcja wstawia nowy wiersz do tabeli `Książki`:

```
INSERT INTO Książki
VALUES  (”1-000-00000-0”,  ”SQL to przyjemność” ,   1,   25 );
```

Poniższa instrukcja wstawia nowy wiersz do tabeli `Książki`. Kolumny `Cena` i `WydId` otrzymują wartość `NULL`

```
INSERT INTO Książki (ISBN, Tytuł)
VALUES (”1-1111-1111-1” , ”Na rybach”);
```

Aby wstawić wiele wiersz, należy użyć następującej składni:

```
INSERT INTO Target [(FieldName,   ….. )]
SELECT FieldName,  …..
FROM TableExpression
```

W obu przypadkach parametr `Target` reprezentuje nazwę tabeli lub kwerendy, do której chcemy wstawić wiersze. Można stosować tylko takie kwerendy, które można aktualizować. Wszystkie aktualizacje będą odzwierciedlane w tabelach bazowych.

Wyrażenie `TableExpression` reprezentuje nazwę tabeli, z której są pobierane rekordy, nazwę zapisanej kwerendy lub instrukcję `SELECT`.

Załóżmy, że tabela `Nowe_Książki` posiada trzy pola: `ISBN`, `WydId` i `Cena`.

Poniższa instrukcja wstawia wiersze z tabeli `Książki` do tabeli `Nowe_Książki`.

Instrukcja ta wstawia tylko te książki, które kosztują ponad 20,00 zł `(Cena > 20 )`.

```
INSERT INTO Nowe_Książki
SELECT ISBN, WydId, Cena
FROM Książki
WHERE Cena > 20;
```

Instrukcja SELECT INTO

Instrukcja SELECT ... INTO – równoznaczna w kwerendą `MakeTable` – tworzy nową tabelę i wstawia do niej dane z innej tabeli. Oto jej składnia:

```
SELECT FieldName,  ….
INTO NewTableName
FROM Source
WHERE RowCondition
ORDER BY OrderCondition
```

Parametr `FieldName` reprezentuje nazwę pola, które ma zostać skopiowane do nowej tabeli.

Parametr Source reprezentuje nazwę tabeli, z której pobieramy dane. Może to być również nazwa kwerendy lub instrukcja wykonująca sprzężenie.

Na przykład poniższa instrukcja tworzy nową tabelę `Drogie_Książki` i wstawia do niej książki z tabeli Książki, które kosztują ponad `45,00 zł`.

```
SELECT Tytuł, ISBN
INTO Drogie_Książki
FROM Książki
WHERE Cena > 45
ORDER BY Tytuł;
```

UWAGA:

- Instrukcja `SELECT ….. INTO` występuje tylko w języku Access SQL.
- Instrukcja `SELECT ..... INTO` nie tworzy indeksów w nowej tabeli.

Instrukcja TRANSFORM

Instrukcja TRANSFORM (nie jest częścią standardu SQL-92) tworzy kwerendy krzyżowe. Oto jej podstawowa składnia:

```
TRANSFORM AggregateFunction
SelectStatement
PIVOT ColumnHeadingsColumn [IN (Value,   ….)]
```

Parametr AggregateFunction reprezentuje jedną z funkcji agregujących programu Access:

(Avg, Count, Min, Max, Sum, First, Last, StDev, StDevP, Var i VarP).

Parametr ColumnHeadingsColumn reprezentuje kolumnę, która zostanie przestawiona, aby jej wartości stały się nagłówkami kolumn tabeli wynikowej kwerendy krzyżowej.

Parametr Value w klauzuli IN określa nagłówki stałych kolumn.

Parametr SelectStatement reprezentuje zmodyfikowaną instrukcję SELECT, która używa klauzuli GROUP BY. W szczególności instrukcja SELECT musi zawierać co najmniej dwie kolumny w klauzuli GROUP BY i nie może zawierać klauzuli HAVING.

Załóżmy na przykład, że chcemy wyświetlić ogólną liczbę książek każdego wydawnictwa według ceny. Instrukcja SELECT:

```
SELECT  WydNazwa, Cena,  COUNT(Tytuł)  AS  Suma
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
GROUP BY WydNazwa,  Cena;
```

W efekcie uzyskamy tabelę:

WydNazwa	Cena	Suma
Big House	15,00 zł	1
Big House	20,00 zł	1
Big House	25,00 zł	2
Big House	49,00 zł	1
Medium House	12,00 zł	2
Medium House	20,00 zł	1
Medium House	34,00 zł	1
Medium House	49,00 zł	1
Small House	49,00 zł	1

Tabela wynikowa – tak naprawdę nie wyświetla informacji w żądanej formie. Trudno na przykład powiedzieć, ile książek kosztuje po 20,00 zł. Pamiętajmy, że ta mała tabela wynikowa jest tylko ilustracją.

Aby z powyższej instrukcji utworzyć kwerendę krzyżową należy:

1. Nad instrukcją `SELECT` wpisać klauzulę `TRANSFORM` i umieścić w niej funkcję agregującą, której wartość chcemy obliczyć.
2. Pod instrukcją `SELECT` wpisać klauzulę `PIVOT` i umieścić w niej kolumnę, której wartości mają się stać nagłówkami kolumn. Ponadto z instrukcji `SELECT` usuwamy referencje do tej kolumny.

W ten sposób otrzymujemy instrukcje:

```
TRANSFORM COUNT (Tytuł)
SELECT  Cena
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
GROUP BY  Cena
PIVOT WydNazwa;
```

tworzącą tabelę wynikową:

Cena	Big House	Medium House	Small House
12,00 zł		2	
15,00 zł	1		
20,00 zł	1	1	
25,00 zł	2		
34,00 zł		1	
49,00 zł	1	1	1

Możemy tworzyć grupy wierszy w oparciu o wartości więcej niż jednej kolumny.

Załóżmy na przykład, że tabela `Książki` posiada również kolumnę `Rabat`, która informuje o rabacie na książki (jako procent ceny).

Jeżeli kolumnę `Rabat` umieścimy w klauzuli `SELECT` i `GROUP BY`, otrzymamy instrukcję:

```
TRANSFORM COUNT (Tytuł)
SELECT  Cena,  Rabat
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
GROUP BY  Cena, Rabat
PIVOT WydNazwa;
```

A po jej wykonaniu otrzymamy tabelę wynikową:

Cena	Rabat	Big House	Medium House	Small House
12,00 zł	30%		2	
15,00 zł	20%	1		
20,00 zł	20%		1	
20,00 zł	30%	1		
25,00 zł	10%	1		
25,00 zł	20%	1		
34,00 zł	10%		1	
49,00 zł	10%	1		
49,00 zł	30%		1	1

W tym przypadku każdy wiersz reprezentuje specyficzną parę cena-rabat.

Kwerenda krzyżowa może również zawierać dodatkowe agregacje wierszy – wystarczy dodać następne funkcje agregujące do klauzuli SELECT:

```
TRANSFORM COUNT (Tytuł)
SELECT  Cena, Count(Cena) AS Liczba, SUM(Cena) AS Suma
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
GROUP BY  Cena
PIVOT WydNazwa;
```

W efekcie czego otrzymujemy tabelę wynikową:

Cena	Liczba	Suma	Big House	Medium House	Small House
12,00 zł	2	24,00 zł		2	
15,00 zł	1	15,00 zł	1		
20,00 zł	2	40,00 zł	1	1	
25,00 zł	2	50,00 zł	2		
34,00 zł	1	34,00 zł		1	
49,00 zł	3	147,00 zł	1	1	1

Dodając nazwy kolumn stałych, możemy zmienić porządek kolumn lub pominąć kolumny w tabeli wynikowej kwerendy krzyżowej.

Na przykład poniższa instrukcja wygląda jak poprzednia w wyjątkiem klauzuli **PIVOT**:

```
TRANSFORM COUNT (Tytuł)
SELECT  Cena, Count(Cena) AS Liczba, SUM(Cena) AS Suma
FROM Wydawnictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
GROUP BY  Cena
PIVOT WydNazwa IN ”Small House”, ”Medium House”;
```

Tabela wynikowa

Cena	Liczba	Suma	Small House	Medium House
12,00 zł	2	24,00 zł		2
15,00 zł	1	15,00 zł		
20,00 zł	2	40,00 zł		1
25,00 zł	2	50,00 zł		
34,00 zł	1	34,00 zł		1
49,00 zł	3	147,00 zł	1	1

Zwróć uwagę, że zmienił się porządek kolumn i że kolumna Big House nie jest wyświetlana.

Podkwerendy

SQL pozwala na używanie instrukcji SELECT wewnątrz następujących instrukcji:

- SELECT
- SELECT ..... INTO
- INSERT ..... INTO
- DELETE
- UPDATE

Wewnętrzna instrukcja SELECT jest nazywana **podkwerendą** i jest głównie stosowana w klauzuli WHERE głównej kwerendy.

Składnia podkwerend przyjmuje trzy możliwe fromy:

Forma 1:

```
Comparison [ ANY | SOME | ALL ] (SQLStatement)
```

Parametr Comparison reprezentuje wyrażenie, po którym następuje słowo kluczowe porównujące to wyrażenie z wartością zwracaną przez podkwerendę. Składnia porównuje daną wartość z wartościami otrzymanymi z innej kwerendy.

Na przykład poniższa instrukcja zwraca wszystkie tytuły i ceny książek tabeli Książki, których ceny są wyższe niż maksymalna cena wszystkich książek tabeli Książki2:

```
SELECT Tytuł, Cena
FROM Książki
WHERE Cena > (SELECT Max(Cena) FROM Książki3);
```

Ponieważ podkwerenda zwraca tylko jedną wartość, nie musimy używać słów kluczowych ANY, SOME lub ALL.

Poniższa instrukcja wybiera z tabeli Książki wszystkie tytuły i ceny książek droższych niż wszystkie (ALL) książki wydane przez wydawnictwo Big House:

```
SELECT Tytuł, Cena
FROM Książki
WHERE Cena > ALL
(SELECT Cena
FROM Wydwanictwa INNER JOIN Książki
ON Wydawnictwa.WydId = Książki.WydId
WHERE WydNazwa = ”Big House”);
```

Zwróćmy uwagę, że słowa kluczowe ANY i SOME mają to samo znaczenie i zwracają wszystkie wybrane wiersze, które czynią porównanie prawdziwym dla co najmniej jednej wartości zwróconej przez podkwerendę. Na przykład jeżeli w poprzednim przykładzie zastąpilibyśmy słowo ALL słowem SOME, tabela zwracana składałaby się z wszystkich tytułów i cen książek droższych niż najtańsza książka wydana przez wydawnictwo Big House.

Forma 2

```
Expression   [ NOT ]   IN   (SQLStatement)
```

Składania ta wyszukuje wartości kolumny w tabeli wynikowej innej kwerendy.

Na przykład poniższa instrukcja zwraca wszystkie tytuły książek przechowywane w tabeli Książki, które nie pojawiają się w tabeli `Książki2`.

```
SELECT Tytuł
FROM Książki
WHERE Tytuł   NOT   IN    (SELECT  Tytuł    FROM Książki2);
```

Forma 3

```
[  NOT   ]    EXISTS     (SQLSTatement)
```

Składnia ta sprawdza, czy element istnieje (został zwrócony) w podkwerendzie.

Na przykład poniższa instrukcja wybiera wszystkie wydawnictwa, które nie posiadają książek w tabeli Książki:

```
SELECT WydNazwa
FROM Wydwanictwa
WHERE NOT EXISTS
(SELECT * FROM  Książki  WHERE Książki.WydId = Wydawnictwa.WydId);
```

Zwróć uwagę, że tabela Wydawnictwa jest wywoływana w tej podkwerendzie. To sprawia, że Access ewaluuje podkwerendę raz dla każdej wartości `Wydwanictwa.WydId` w tabeli `Wydwanictwa`.

UWAGA:

- Jeżeli stosujemy formę 1 lub 2, podkwerenda musi zwrócić jedną kolumnę. W innym przypadku wystąpi błąd.
- W instrukcji `SELECT` tworzącej podkwerendę stosujemy te same reguły, co w każdej innej instrukcji `SELECT`. Musimy ją jednak umieścić w nawiasach.

Parametry

W języku Access SQL można używać parametrów w celu uzyskania informacji od użytkownika w czasie działania kwerendy. Wiersz PARAMETERS musi być pierwszym wierszem instrukcji i ma następującą składnię:

```
PARAMETERS    Name     DataType, …..
```

Technikę tę zilustrujemy przykładem.

Poniższa instrukcja poprosi użytkownika o podanie łańcucha znaków będącego częścią tytułu książki i zwróci wszystkie książki tabeli Książki o tytułach zawierających podany łańcuch znaków. Zwróć uwagę na średnik kończący wiersz PARAMETERS.

```
PARAMETERS [Wpisz część tytułu książki ]    TEXT;
SELECT *
FROM Książki
WHERE InStr(Tytuł,  [Wpisz część tytułu książki ] )    >  0;
```

Funkcja **InStr**(Tekst1, Tekst2) zwraca pierwszą lokalizację łańcucha znaków Tekst2 w łańcuchu znaków Tekst1. Zwróćmy uwagę, że parametr Name jest powtórzony w klauzuli WHERE i zostanie wypełniony wartością wprowadzoną przez użytkownika, dzięki temu, że parametr Name występuje w klauzuli PARAMETERS.