

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

SQL w mgnieniu oka

Autor: Ben Forta

Tłumaczenie: Rafał Jońca, Paulina Sobaś

ISBN: 83-7361-469-9

Tytuł oryginału: [Teach Yourself SQL
in 10 Minutes, 2nd Edition](#)

Format: B5, stron: 248



Na temat języka SQL napisano już wiele książek. Wiele z nich, o niekwestionowanej jakości, obarcza jednak czytelnika mnogością dodatkowych informacji na temat teorii relacyjnych baz danych, ich projektowania i administrowania nimi.

Mimo fundamentalnego znaczenia tych zagadnień użytkownik chciałby jednak skupić się na szczegółach samego języka SQL, poczynając od jego najprostszych elementów, by dopiero później, w miarę doskonalenia swej wiedzy i poznawania coraz bardziej złożonych elementów SQL sięgnąć do tematyki o charakterze bardziej ogólnym. Niniejsza książka jest wolna od opisanego syndromu, a każdy z jej rozdziałów czyta się w ciągu 10 minut. Pasjonująca przygoda z językiem SQL rozpoczyna się już w pierwszym rozdziale; w kolejnych Czytelnik zapoznaje się z coraz bardziej złożonymi zagadnieniami, jak:

- Podstawowe elementy baz danych – tabele, kolumny, wiersze i klucze
- Pobieranie danych z tabeli i ich sortowanie
- Filtrowanie danych za pomocą fraz WHERE i operatorów AND, OR, IN, NOT i LIKE
- Tworzenie unii
- Wstawianie, aktualizacja i usuwanie danych
- Tworzenie i modyfikowanie tabel
- Tworzenie i wykorzystywanie perspektyw
- Wykorzystywanie procedur zapamiętanych
- Zarządzanie transakcjami
- Indeksowanie i powiązania między tabelami za pomocą kluczy
- Zastosowanie języka SQL na gruncie Visual C++ i Visual Basic a oraz popularnych systemów baz danych, jak SQL Server 6x, 7 i 2000, MS Access, MS Query i MS ASP



Spis treści

0 Autorze	8
Wprowadzenie	9
Rozdział 1. Podstawy języka SQL.....	13
Podstawy baz danych	13
Język SQL	18
Ćwicz	19
Podsumowanie	19
Rozdział 2. Pobieranie danych	21
Instrukcja SELECT	21
Pobranie konkretnej kolumny	22
Pobranie wielu kolumn	23
Pobranie wszystkich kolumn	24
Podsumowanie	25
Rozdział 3. Sortowanie otrzymywanych danych	27
Sortowanie danych	27
Sortowanie względem wielu kolumn	29
Sortowanie względem położenia kolumny	30
Określenie kierunku sortowania	31
Podsumowanie	33
Rozdział 4. Filtrowanie danych	35
Stosowanie frazy WHERE	35
Operatory frazy WHERE	36
Podsumowanie	40
Rozdział 5. Zaawansowane filtrowanie danych	41
Łączenie fraz WHERE	41
Operator IN	45
Operator NOT	46
Podsumowanie	48

Rozdział 6. Filtrowanie za pomocą znaków wieloznacznych.....	49
Korzystanie z operatora LIKE	49
Wskazówki dotyczące używania znaków wieloznacznych	54
Podsumowanie	54
Rozdział 7. Tworzenie pól obliczanych.....	55
Pojęcie pól obliczanych	55
Konkatenacja pól.....	56
Przeprowadzanie obliczeń matematycznych	62
Podsumowanie	63
Rozdział 8. Modyfikacja danych za pomocą funkcji	65
Czym są funkcje.....	65
Stosowanie funkcji.....	67
Podsumowanie	73
Rozdział 9. Funkcje agregujące	75
Funkcje agregujące	75
Agregacja tylko różnorodnych wartości	81
Łączenie funkcji agregujących.....	83
Podsumowanie	83
Rozdział 10. Grupowanie danych	85
Omówienie grupowania danych.....	85
Tworzenie grup	86
Filtrowanie grup	88
Grupowanie i sortowanie	90
Kolejność fraz instrukcji SELECT	92
Podsumowanie	92
Rozdział 11. Zapytania zagnieżdżone	93
Zagnieżdżanie zapytań	93
Filtrowanie na podstawie zapytań zagnieżdżonych	94
Zapytania zagnieżdżone jako pola obliczane.....	97
Podsumowanie	99
Rozdział 12. Łączenie tabel	101
Czym są złączenia?	101
Tworzenie złączeń.....	104
Podsumowanie	110
Rozdział 13. Tworzenie rozbudowanych złączeń.....	111
Stosowanie aliasów tabel	111
Typy złączeń	113
Złączenia i funkcje agregujące.....	119
Złączenia i ich warunki	120
Podsumowanie	120
Rozdział 14. Łączenie zapytań	121
Łączenie zapytań.....	121
Tworzenie unii	122
Podsumowanie	126

Rozdział 15. Wstawianie danych	127
Wstawianie danych	127
Kopiowanie z jednej tabeli do innej.....	133
Podsumowanie	134
Rozdział 16. Aktualizacja i usuwanie danych	135
Aktualizacja danych	135
Usuwanie danych	137
Wskazówki związane z aktualizacją lub usuwaniem danych	138
Podsumowanie	139
Rozdział 17. Tworzenie i modyfikacja tabel	141
Tworzenie tabel.....	141
Aktualizacja tabel.....	147
Usuwanie tabel.....	148
Zmiana nazwy tablicy	149
Podsumowanie	149
Rozdział 18. Stosowanie perspektyw.....	151
Perspektywy	151
Tworzenie widoków.....	154
Podsumowanie	160
Rozdział 19. Korzystanie z zapamiętanych procedur	161
Zapamiętane procedury	161
Dlaczego warto używać zapamiętanych procedur	162
Wykonywanie zapamiętanych procedur	164
Tworzenie zapamiętanych procedur	165
Podsumowanie	168
Rozdział 20. Zarządzanie transakcjami	169
Działanie transakcji.....	169
Sterowanie transakcjami	171
Podsumowanie	175
Rozdział 21. Kursory	177
Działanie kursorów	177
Praca z kursorami	178
Podsumowanie	182
Rozdział 22. Zaawansowane funkcje języka SQL.....	183
Ograniczenia	183
Omówienie indeksów.....	189
Wyzwalacze	191
Bezpieczeństwo baz danych.....	193
Podsumowanie	194
Dodatek A Skrypty przykładowych tabel.....	195
Omówienie przykładowych tabel.....	195
Tworzenie przykładowych tabel	199
Wypełnienie przykładowych tabel.....	202

Dodatek B	Praca z popularnymi aplikacjami	213
	Konfiguracja źródeł danych ODBC.....	213
	Allaire ColdFusion.....	215
	Allaire JRun 3.x	215
	DB2	216
	Informix Dynamic Server 7.x	216
	Microsoft Access.....	217
	Microsoft ASP.....	218
	Microsoft Query.....	219
	Microsoft SQL Server 6.x.....	220
	Microsoft SQL Server 7.....	220
	Microsoft SQL Server 2000.....	221
	Microsoft Visual Basic.....	221
	Microsoft Visual C++	222
	Oracle 8.....	223
	Query Tool.....	223
	Sybase	224
Dodatek C	Składnia instrukcji SQL.....	225
	ALTER TABLE.....	225
	COMMIT	226
	CREATE INDEX.....	226
	CREATE PROCEDURE	226
	CREATE TABLE	227
	CREATE VIEW.....	227
	DELETE.....	227
	DROP	228
	INSERT.....	228
	INSERT SELECT	228
	ROLLBACK	229
	SELECT	229
	UPDATE	229
Dodatek D	Typy danych języka SQL.....	231
	Tekstowe typy danych	232
	Numeryczne typy danych.....	233
	Typy danych daty i czasu.....	234
	Binarne typy danych	235
Dodatek E	Słowa kluczowe języka SQL	237
	Skorowidz	241

Rozdział 10.

Grupowanie danych

W tym rozdziale opisana jest funkcja grupowania danych, która umożliwia podsumowywanie podzbiorów tabeli. Wprowadzone są też dwie nowe frazy instrukcji SELECT: GROUP BY i HAVING.

Omówienie grupowania danych

W poprzednim rozdziale opisywałem funkcje agregujące języka SQL używane do tworzenia podsumowań danych. Umożliwiały one zliczanie wierszy, obliczanie sumy i średniej, a także znajdowanie wartości największej i najmniejszej. Wszystko to odbywało się bez potrzeby pobierania wszystkich danych.

Do tej pory obliczenia przeprowadzane były na wszystkich danych w tabeli, lub na danych spełniających warunek określony we frazie WHERE. Oto krótkie przypomnienie — poniższy przykład zwraca wszystkie produkty oferowane przez dostawcę DLL01:

wejście

```
SELECT COUNT(*) AS liczba_prod
FROM Produkty
WHERE dost_id = 'DLL01';
```

wyjście

```
liczba_prod
-----
4
```

W jaki sposób pobrać liczbę produktów oferowanych przez poszczególnych producentów? Jak uzyskać dane dotyczące tylko tych producentów, którzy oferują jeden produkt lub oferują powyżej 10 produktów?

W takiej sytuacji niezastąpione jest grupowanie. Umożliwia ono podzielenie danych na logiczne zbiory i przeprowadzenie funkcji agregujących na każdej z grup osobno.

Tworzenie grup

Grupy tworzy się za pomocą frazy `GROUP BY` w instrukcji `SELECT`. Najłatwiej zrozumieć to na przykładzie:

wejście

```
SELECT dost_id, COUNT(*) AS liczba_prod
FROM Produkty
GROUP BY dost_id;
```

wyjście

dost_id	liczba_prod
BRS01	3
DLL01	4
FNG01	2

analiza

Powyższa instrukcja `SELECT` określa dwie kolumny, `dost_id` (która zawiera identyfikator dostawcy) i `liczba_prod` (która jest polem obliczanym za pomocą funkcji agregującej `COUNT(*)`). Fraza `GROUP BY` sprawia, że SZBD sortuje wyniki i grupuje je względem `dost_id`. Powoduje to obliczanie `liczba_prod` dla każdego unikalnego `dost_id`, zamiast zbiorowo dla całej tabeli. W ten sposób można się dowiedzieć, iż dostawca BRS01 oferuje 3 produkty, dostawca DLL01 4 produkty, a dostawca FNG01 2 produkty.

Ponieważ pojawiła się fraza `GROUP BY`, nie trzeba było określać każdej grupy, by poznać jej wartość. Wszystko zostało wykonane automatycznie. Fraza `GROUP BY` powoduje, że system zarządzania najpierw grupuje dane, a następnie przeprowadza funkcję agregującą osobno dla każdej grupy, zamiast dla wszystkich wyników.

Zanim jednak rozpocznie się stosowanie frazy `GROUP BY`, warto dokładniej poznać jej działanie:

- ♦ Fraza `GROUP BY` może zawierać dowolną liczbę kolumn. Umożliwia to tworzenie zagnieżdżonych grup, a tym samym bardziej szczegółowe opracowywanie danych.
- ♦ W przypadku wprowadzenia kilku nazw kolumn we frazie, dane podsumowywane są dla ostatniej określonej kolumny (dla jej grup). Oznacza to, że nie jest możliwe uzyskanie danych dla wszystkich wymienionych kolumn.
- ♦ Wszystkie kolumny wymienione we frazie `GROUP BY` muszą być kolumnami pobieranymi z bazy lub pełnymi wyrażeniami (ale nie funkcjami agregującymi). Jeśli wyrażenie występuje po `SELECT`, w takiej samej postaci musi się znaleźć we frazie `GROUP BY`. Nie można w tym wypadku stosować aliasów.
- ♦ Większość implementacji SQL nie dopuszcza, aby we frazie `GROUP BY` znalazły się kolumny typów danych o zmiennej długości (na przykład pola tekstowe lub memo).
- ♦ Wszystkie kolumny występujące w instrukcji `SELECT` muszą się także znaleźć we frazie `GROUP BY` (nie dotyczy to funkcji agregujących).
- ♦ Jeśli grupowana kolumna zawiera wiersz z wartością `NULL`, powstanie osobna grupa o nazwie `NULL`. Jeśli istnieje kilka wierszy o wartości `NULL`, zostaną one scalone w jedną grupę.
- ♦ W przypadku występowania dodatkowych fraz, funkcja `GROUP BY` musi się pojawić po frazie `WHERE`, ale przed `ORDER BY`.



Fraza `ALL`. Pewne implementacje SQL (na przykład Microsoft SQL Server) obsługują opcjonalną frazę `ALL` dla `GROUP BY`. Fraza ta może posłużyć do zwrócenia wszystkich grup, nawet tych, dla których agregacja spowodowałaby zwrócenie wartości `NULL`. Szczegółów należy szukać w dokumentacji SZBD.



Określanie kolumn za pomocą położeń względnych. Niektóre implementacje SQL umożliwiają podanie we frazie `GROUP BY` położeń kolumn z listy `SELECT`. Można wtedy na przykład napisać `GROUP BY 2, 1`, aby grupowanie najpierw odbyło się względem drugiej kolumny, a następnie pierwszej. Choć ten skrótowy zapis jest bardzo kuszący, nie obsługują go wszystkie implementacje, a dodatkowo niesie ze sobą ryzyko pojawienia się błędów po zmianie kolejności kolumn.

Filtrowanie grup

Poza samą możliwością grupowania danych, język SQL oferuje także filtrowanie na podstawie danych zebranych dla poszczególnych grup. Na przykład można wyświetlić wszystkich klientów, którzy dokonali przynajmniej dwóch zamówień. Takie filtrowanie musi się odbywać wobec pełnych grup, a nie poszczególnych wierszy.

Nie jest możliwe posłużenie się tutaj frazą `WHERE` opisaną w rozdziale 4., gdyż powoduje ona filtrowanie wierszy i to jeszcze przed rozpoczęciem grupowania. Inaczej mówiąc, fraza `WHERE` nie wie, czym jest grupowanie.

Jaka jest więc alternatywa dla `WHERE`? Język SQL wprowadza dodatkową frazę `HAVING`. Jest ona bardzo podobna do `WHERE`. W zasadzie wszystkie opisane do tej pory techniki filtrowania związane z `WHERE` mogą zostać także użyte we frazie `HAVING`. Jedyna różnica polega na tym, iż `WHERE` filtryje wiersze, a `HAVING` grupy.



Fraza `HAVING` obsługuje wszystkie operatory frazy `WHERE`. Rozdziały 4. i 5. opisywały proste i zaawansowane filtrowanie danych. Wszystkie opisane tam operatory można z powodzeniem stosować we frazie `HAVING`. Składnia jest identyczna, zmienia się tylko słowo kluczowe.

Oto przykład filtrowania grup:

wejście

```
SELECT k1_id, COUNT(*) AS zamowienia
FROM Zamowienia
GROUP BY k1_id
HAVING COUNT(*) >= 2;
```

wyjście

k1_id	zamowienia
1000000001	2

analiza

Pierwsze trzy wiersze są bardzo podobne do wcześniejszego przykładu z tego rozdziału. Ostatni wiersz wprowadza frazę `HAVING`, która przepuszcza tylko te grupy, które posiadają minimum dwa zamówienia — `COUNT(*) >= 2`.

Można się przekonać, iż fraza `WHERE` nie przeprowadziłaby poprawnego filtrowania, ponieważ musi ono bazować na wartości z agregacji grupy, a nie wartości znajdujących się w poszczególnych wierszach.



Różnica między `HAVING` i `WHERE`. Można na to zagadnienie spojrzeć inaczej. Fraza `WHERE` filtruje dane przed grupowaniem, natomiast `HAVING` po. Jest to ważna różnica — wiersze wyeliminowane przez frazę `WHERE` w ogóle nie zostaną wzięte pod uwagę przy tworzeniu grup. Powoduje to zmianę wartości pól obliczanych, a tym samym zmianę wyświetlanych grup, gdy przeprowadzane jest filtrowanie.

Czy zachodzi potrzeba jednoczesnego stosowania fraz `WHERE` i `HAVING` w jednym zapytaniu? W pewnych sytuacjach jest ona nawet konieczna. Przypuśćmy na przykład, iż poprzednie zapytanie powinno zwrócić dowolnych klientów z więcej niż jednym zamówieniem, ale pod uwagę należy brać tylko ostatni rok. W takiej sytuacji fraza `WHERE` spowoduje przeanalizowanie zamówień tylko z ostatnich 12 miesięcy, a fraza `HAVING` wskaże tylko klientów z co najmniej dwoma zamówieniami.

Oto inny przykład. Zapytanie powoduje wyświetlenie listy dostawców z więcej niż jednym produktem o cenie powyżej 10 złotych.

wejscie

```
SELECT dost_id, COUNT(*) AS liczba_prod
FROM Produkty
WHERE prod_cena >= 10
GROUP BY dost_id
HAVING COUNT(*) >= 2;
```

wyjście

dost_id	liczba_prod
-----	-----
BRS01	3
FNG01	2

analiza

Zapytanie to wymaga krótkiego wyjaśnienia. Pierwszy wiersz to prosta instrukcja `SELECT` z funkcją agregującą — podobnie jak w poprzednich przykładach. Fraza `WHERE` filtruje wszystkie wiersze, których `prod_cena` jest mniejsza od 10 złotych. Następnie dane są grupowane pod kątem `dost_id`, a fraza `HAVING` przepuszcza tylko grupy zawierające więcej niż jedno zamówienie. Bez frazy `WHERE` zostałyby pobrany dodatkowy wiersz, ponieważ dostawca `DLL01` sprzedaje trzy produkty po cenie poniżej 10 złotych.

wejście

```
SELECT dost_id, COUNT(*) AS liczba_prod
FROM Produkty
GROUP BY dost_id
HAVING COUNT(*) >= 2;
```

wyjście

```
dost_id  liczba_prod
-----  -
BRS01    3
DLL01    4
FNG01    2
```



Stosowanie HAVING i WHERE. Fraza HAVING jest tak podobna do frazy WHERE, że większość SZBD traktuje je identycznie, jeśli nie określono frazy GROUP BY. Mimo to warto samemu jasno rozdzielać obie frazy, czyli HAVING stosować tylko z GROUP BY, a WHERE używać do filtrowania na niższym poziomie.

Grupowanie i sortowanie

Trzeba zdać sobie sprawę z tego, iż frazy GROUP BY i ORDER BY są bardzo różne, choć wykonują podobne zadanie. Tabela 10.1 wymienia różnice występujące między obiema frazami.

Tabela 10.1. Frazy ORDER BY i GROUP BY

ORDER BY	GROUP BY
Sortuje wygenerowane wyjście.	Grupuje wiersze. Wyjście nie musi być jednak posortowane względem grup.
Można stosować dla dowolnych kolumn (także tych, które nie są zwracane).	Można zastosować tylko wobec zwracanych kolumn lub wyrażeń. Wystąpić muszą wszystkie zwracane kolumny lub wyrażenia.
Stosowanie frazy nie jest zawsze wymagane.	Stosowanie frazy jest konieczne, jeśli używa się kolumn (lub wyrażeń) z funkcjami agregującymi.

Pierwsza różnica wymieniona w tabeli 10.1 jest niezmiernie ważna. Choć nie jest to wymagane, to najczęściej grupy będą wyświetlane w sposób posortowany. Co więcej, choć dany system może zawsze sortować grupy po zastosowaniu frazy GROUP BY, konieczny może okazać się inny sposób sortowania. Choć grupowanie odbywa się w taki, a nie inny sposób, nie oznacza to jednocześnie,

iż sortowanie także musi odbywać się w ten sam sposób. Zawsze można zastosować opcjonalną frazę `ORDER BY`, aby wymusić odpowiednie sortowanie pogrupowanych danych.



Nie zapomnij o frazie `ORDER BY`. Ogólnie rzecz ujmując, za każdym razem, gdy stosuje się frazę `GROUP BY`, powinno się także stosować frazę `ORDER BY`, gdyż jest to jedyny sposób zapewnienia odpowiedniego sortowania danych. Nigdy nie należy polegać na sortowaniu za pomocą `GROUP BY`.

Oto krótki przykład, który zademonstruje użycie fraz `GROUP BY` i `ORDER BY`. Przedstawione zapytanie `SELECT` jest bardzo podobne do poprzednich. Pobiera numery zamówień i liczbę zamawianych elementów dla wszystkich zamówień zawierających minimum trzy elementy.

wejście

```
SELECT zam_numer, COUNT(*) AS elementy
FROM ElementyZamowienia
GROUP BY zam_numer
HAVING COUNT(*) >=3;
```

wyjście

zam_numer	elementy
20006	3
20007	5
20008	5
20009	3

Aby posortować wyjście na podstawie liczby zamówionych elementów, wystarczy tylko dodać odpowiednią frazę `ORDER BY`.

wejście

```
SELECT zam_numer, COUNT(*) AS elementy
FROM ElementyZamowienia
GROUP BY zam_numer
HAVING COUNT(*) >=3
ORDER BY elementy, zam_numer;
```

wyjście

zam_numer	elementy
20006	3

20009	3
20007	5
20008	5

analiza

W tym przykładzie fraza `GROUP BY` służy do grupowania danych na podstawie numeru zamówienia (kolumna `zam_numer`), więc funkcja `COUNT(*)` zwraca liczbę elementów w poszczególnych zamówieniach. Fraza `HAVING` filtruje dane, więc zwracane są tylko zamówienia z więcej niż dwoma elementami. Na końcu wyniki są sortowane za pomocą frazy `ORDER BY`.

Kolejność fraz instrukcji **SELECT**

Nadszedł chyba najlepszy czas na omówienie kolejności występowania fraz w instrukcji `SELECT`. Tabela 10.2 zawiera poprawną kolejność wszystkich omówionych do tej pory fraz.

Tabela 10.2. Frazy instrukcji `SELECT` i ich kolejność

Fraza	Opis	Wymagane
<code>SELECT</code>	zwracanie kolumn lub wyrażeń	tak
<code>FROM</code>	pobranie danych zawartych w tabelach	tylko, gdy wymagane są dane z tabel w tabelach
<code>WHERE</code>	filtrowanie wierszy	nie
<code>GROUP BY</code>	tworzenie grup	tylko do obliczania funkcji agregujących na grupach
<code>HAVING</code>	filtrowanie grup	nie
<code>ORDER BY</code>	sortowanie wyjścia	nie

Podsumowanie

W rozdziale 9. czytelnik zapoznał się z zastosowaniem funkcji agregujących. W powyższym rozdziale wykorzystał frazę `GROUP BY` do przeprowadzania funkcji agregujących względem określonych grup elementów. Fraza `HAVING` służy do filtrowania grup. Dodatkowo w rozdziale pojawiło się dokładne wyjaśnienie różnic między `GROUP BY` i `ORDER BY` oraz między `WHERE` i `HAVING`.