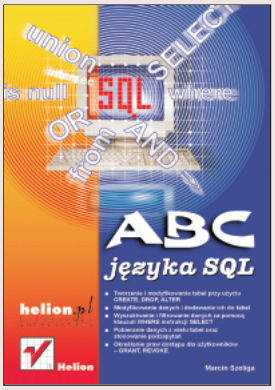
ABC języka SQL



Autor: Marcin Szeliga
ISBN: 83-7197-593-7
Format: B5, stron: 194

Bazy danych stały się głównym składnikiem systemów informatycznych. W zależności od producenta, przeznaczenia i ceny programy zarządzające bazami danych bardzo się od siebie różnią.
Jedynym punktem wspólnym dla wszystkich tych aplikacji jest wbudowana obsługa strukturalnego języka zapytań (ang. Structured Query Language).
SQL służy do tworzenia aplikacji bazodanowych i zarządzania nimi. Używając go, możemy:

- wyszukiwać dane w bazie danych.
- operować danymi – wstawiać je, modyfikować i usuwać.
- definiować dane – dodawać nowe tabele, indeksy i perspektywy.
- sterować danymi – blokować użytkownikom dostęp do poufnych danych.
- modyfikować schemat baz danych bez zmieniania istniejących aplikacji – nowe kolumny i tabele mogą być zawsze dodane bez obawy, że zajdzie konieczność zmiany istniejących programów lub zdefiniowanych z góry ścieżek dostępu.
- formułować zapytania w trybie interakcyjnym lub osadzać je w standardowych językach programowania, takich jak C lub Pascal.

Aby opanować SQL w stopniu wystarczającym do stworzenia i korzystania z bazy danych, wystarczy znajomość dziewięciu poleceń: SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP, GRANT i REVOKE. Książka została przygotowana tak, aby była przydatna zarówno dla uczniów i studentów którzy chcieliby poznać temat projektowania i programowania baz danych (książka powstała na podstawie prowadzonych przez autora wykładów z teorii relacyjnych baz danych i języka SQL) jak i dla administratorów baz danych (książka zawiera kilkadziesiąt przykładów wykorzystania języka SQL do ułatwienia lub zautomatyzowania codziennych prac związanych z zarządzaniem bazami danych).

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 5.
Podzapytania

Podzapytania lub *zapytania zagnieżdżone* to instrukcje `select` umieszczone wewnątrz innych instrukcji `select`. Podzapytania mogą być używane w klauzuli `where` do filtrowania danych lub, co zostanie wyjaśnione w części trzeciej, w instrukcji `insert` do kopiowania danych z jednej tabeli do drugiej. Podzapytań używamy, gdy dane z pewnej tabeli są potrzebne w innym zapytaniu. Podobnie jak możemy zagnieżdżać wywołania funkcji w innych funkcjach, tak samo możemy zagnieżdżać zapytania w innych zapytaniach. Wykorzystując podzapytania można osiągnąć podobny efekt, jaki osiągamy przez wprowadzenie zmiennych w proceduralnych językach programowania. Zamiast wykonania instrukcji i przechowania jej w zmiennej, wykonywane jest drugie wyrażenie wykorzystujące poprzedni wynik, co daje efekt zagnieżdżania pierwszej instrukcji w drugiej.

SZBD wykonuje podzapytania, zaczynając od najbardziej wewnętrznej instrukcji `select`, po to aby wynik tej instrukcji wykorzystać do wykonania zapytań zewnętrznych.

Podzapytania dzielą się na:

- Podzapytania powiązane i nie powiązane. Podzapytanie powiązane wymaga danych z zapytania otaczającego, zanim może być wykonane. Dane zwrócone przez podzapytanie powiązane wprowadzane są z powrotem do zapytania otaczającego. Natomiast podzapytanie nie powiązane wykonuje się przed zapytaniem otaczającym, a jego wyniki są przekazywane do zapytania otaczającego. Podzapytanie nie powiązane jest prostszym przykładem podzapytań, tyle tylko że w praktyce właściwie nie używanym).
- Podzapytania zwracające jeden wiersz i zwracające wiele wierszy. Kryterium podziału stanowi tutaj liczba wierszy zwracanych przez wewnętrzną instrukcję `select`.

W podzapytaniu nie może wystąpić klauzula `order by`. Klauzula ta może być użyta tylko raz dla całego zapytania.

Podzapytania nie powiązane

Zapytanie wykorzystujące nie powiązane podzapytanie przekazuje sterowanie do wewnętrznej instrukcji `select`, obliczone w nim dane przekazuje do zapytania zewnętrznego i wykonuje je. Ogólny schemat podzapytań nie związanych przedstawiony jest poniżej.

```
SELECT nazwa_kolumny
FROM tabela
WHERE nazwa_kolumny
operator
  (SELECT [DISTINC] nazwa_kolumny
  FROM tabela
  [WHERE warunek]);
```

Na przykład, aby znaleźć nazwę najdroższego towaru, należy wykonać instrukcję:

```
SELECT nazwa
FROM Towar
WHERE cena_sprzedazy =( SELECT max(cena_sprzedazy)
FROM towar);
```

Wyjaśnienie:
SZBD najpierw wykonuje wewnętrzną instrukcję `select`:

```
SELECT max(cena_sprzedazy)
FROM towar);
```

W wyniku przeprowadzenia tej operacji otrzymujemy najwyższą cenę sprzedaży (7 500,00 zł). Następnie wykonywana jest instrukcja

```
SELECT nazwa
FROM Towar
WHERE cena_sprzedazy =7500;
```

po zrealizowaniu której otrzymamy nazwę najdroższego towaru.

Tabela 5.1. *Podzapytanie nie powiązane*

Nazwa
Zegarek z pozytywką

Podzapytania powiązane

Częściej wykorzystywane zapytanie powiązane różni się od zapytania nie powiązanego tym, że pozycje z listy `select` zapytania otaczającego są wykorzystane wewnątrz klauzuli `where` podzapytania. Zapytania powiązane często porównywane są ze złączeniami, ponieważ zawartość tabeli występującej w podzapytaniu będzie porównywana z zawartością tabeli z zapytania otaczającego, podobnie jak w zapytaniu złączającym. Różnica polega na tym, że zamiast warunku złączającego, powiązane podzapytanie odwołuje się do zapytania zewnętrznego przez klauzulę `where` zapytania wewnętrznego.

Aby znaleźć wszystkie towary przypisane do tej samej grupy towarowej, co towar o nazwie *Discipline*, napiszemy:

```
SELECT nazwa, Id_grupy
FROM Towar
WHERE Id_grupy = (SELECT Id_grupy
FROM Towar
WHERE nazwa="Discipline");
```

Tabela 5.2. *Podzapytanie powiązane*

Nazwa	Id_grupy
Vroom	Muzyka
Discipline	Muzyka

Podczas wykonywania podzapytania powiązanego SZBD wykonuje kolejno następujące czynności: w pierwszej kolejności z bazy danych odczytywana jest zawartość wiersza, następnie wykonywane jest podzapytanie, a wartości z aktualnie wybranego wiersza zapytania otaczającego są wykorzystywane w klauzuli `where` podzapytania. Kolejną operacją jest przekazywanie wyników podzapytania do klauzuli `where`. W przypadku, gdy wyrażenie logiczne w warunku klauzuli `where` ma wartość prawda, wiersz jest pobierany do zestawienia wynikowego, w przeciwnym wypadku jest pomijany.

Podzapytania zwracające jeden wiersz

Podzapytania tego typu można traktować jak zwykłe wyrażenia. W szczególności podzapytania tego typu mogą być wykorzystywane ze standardowymi operatorami porównań, np. z klauzulą `beetwen`. Dlatego jeśli chcemy znaleźć, powiedzmy, te towary, które zostały kupione za cenę niższą od średniej ceny zakupu towarów, należy wykonać polecenie:

```
SELECT nazwa, cena_zakupu
FROM Towar
WHERE cena_zakupu< (SELECT avg(cena_zakupu)
FROM Towar);
```

Wyjaśnienie:
SZBD najpierw wykonuje wewnętrzną instrukcję `select`:

```
SELECT avg(cena_zakupu)
FROM Towar;
```

W jej wyniku otrzymujemy jedną wartość będącą średnią ceną zakupu towarów (5 073,6429 zł). Następnie SZBD sprawdza ceny zakupów wszystkich towarów z tabeli *Towar* i sprawdza, czy któraś z nich nie jest mniejsza od 5 073,64. Jeżeli tak, nazwa towaru zostaje dodana do tabeli wynikowej.

Tabela 5.3. *Podzapytanie zwracające jeden wiersz*

Nazwa	Cena_zakupu
Tajemnicze zioła na kaszel	1 200,00 zł
Magiczny napój na ból głowy	2 000,00 zł
Vroom	150,00 zł
Discipline	590,00 zł
Black & White	99,00 zł
Palmy (dzikie)	222,00 zł
Pomarańcza (mechaniczna)	450,00 zł
Sofa błękitna	500,00 zł
Zegarek z pozytywką	3 000,00 zł
Wypłata dla Bjorga	3 000,00 zł
Wypłata dla Galadrieli	3 500,00 zł
Wypłata dla Gandalfa	4 000,00 zł
Mieszanka ziół wschodu	2 320,00 zł

Podzapytania zwracające wiele wierszy

Przypomnijmy sobie jedną z niedawno opisywanych instrukcji:

```
SELECT nazwa, Id_grupy
FROM Towar
WHERE Id_grupy = (SELECT Id_grupy
FROM Towar
WHERE nazwa="Discipline");
```

Proszę zauważyć, że jeśli w tabeli *Towar* znajdowałoby się więcej towarów o nazwie *Discipline*, to powyższe zapytanie nie miałoby sensu. Ponieważ wewnętrzne zapytanie zwróciłoby listę wartości, a nie pojedynczy wiersz, SZBD nie mógłby poprawnie obliczyć wyniku zapytania zewnętrznego i zamiast listy towarów z podanej grupy otrzymalibyśmy komunikat błędu. Przekształceniem tej instrukcji tak, aby podzapytanie mogło zwrócić dowolną liczbę wartości (włączając w to 0), a przy tym działało zgodnie z naszymi oczekiwaniami, zajmiemy się w bieżącym punkcie.

Podzapytania zwracające listę wartości mogą być wykorzystane w wyrażeniach typu `in`, `not in` oraz łącznie ze słowami kluczowymi `exists`, `any` lub `all`.

Podzapytania z wyrażeniem IN

Wyrażenie `in` jest wykorzystywane do sprawdzenia, czy wartość należy do pewnego zbioru. Podzapytanie może być wykorzystane do wybrania tego zbioru wartości. Przekształćmy naszą instrukcję tak, aby wykorzystać w podzapytaniu wyrażenie `in`:

```
SELECT nazwa, Id_grupy
FROM Towar
WHERE Id_grupy IN (SELECT Id_grupy
FROM Towar
WHERE nazwa="Discipline");
```

Tabela 5.4. *Podzapytanie wykorzystujące wyrażenie in*

Nazwa	Id_grupy
Vroom	Muzyka
Discipline	Muzyka

Wynik działania obu instrukcji w tym przypadku jest identyczny, ale gdybyśmy chcieli znaleźć wszystkie towary sprzedawane na sztuki, to poprawna instrukcja musi wyglądać, tak jak zostało to pokazane poniżej (ponieważ w tabeli *Towar* znajduje się kilka towarów sprzedawanych na sztuki).

```
SELECT nazwa, Id_grupy, jednostka
FROM Towar
WHERE Id_grupy IN (SELECT Id_grupy
FROM Towar
WHERE jednostka="szt");
```

Tabela 5.5. *Wyrażenie in zwracające listę wierszy*

Nazwa	Id_grupy	Jednostka
Vroom	Muzyka	szt
Discipline	Muzyka	szt
Black & White	IT	szt
Palmy (dzikie)	Film	szt
Pomarańcza (mechaniczna)	Film	szt
Sofa błękitna	Sztuka	szt
Zegarek z pozytywką	Sztuka	szt

Większość SZBD wymaga, aby kolumny występujące na liście wyboru wewnętrznego (kolumny występujące w wewnętrznych klauzulach `where` lub `having`) występowały w tej samej kolejności w klauzuli `select` zewnętrznego zapytania.

Podzapytania z wyrażeniem NOT IN

Podzapytania mogą być również wykorzystane do tworzenia list w klauzuli `not in`. Wartość logiczna takiego wyrażenia jest prawdziwa, gdy wartość testowana nie należy do listy wartości.

Nie zaleca się korzystania z wyrażeń `NOT IN` w podzapytaniach z powodu ich niskiej wydajności. W większości wypadków można zamiast tego skorzystać z zapytania wykorzystującego złączenie zewnętrzne.

Przypuśćmy, że chcemy utworzyć listę grup towarowych, w których nie ma ani jednego towaru. W tym celu należy wyszukać nazwy tych grup towarowych, dla których w tabeli *Towar* nie ma ani jednego wpisu.

```
SELECT nazwa
FROM Grupa
WHERE Id_Grupy NOT  IN (SELECT Id_Grupy
FROM Towar);
```

Tabela 5.6. *Podzapytanie wykorzystujące wyrażenie not in*

Nazwa
Pusta
Foto

Alternatywną metodę uzyskania tego wyniku przedstawię w następnym punkcie.

Podzapytania z wyrażeniem EXIST

W przypadku podzapytań czasami chcemy jedynie sprawdzić, czy wiersz spełniający podane warunki istnieje w bazie danych. Najprostszą metodą sprawdzenia, czy dany wiersz występuje w podanej tabeli jest użycie wyrażenia `exist`.

Jeżeli podzapytanie zwraca dowolną wartość, to klauzula `exist` zwraca wartość logiczną `true` (prawda).

Aby za pomocą operatora `exist` znaleźć te towary, które są przypisane do jakichkolwiek grup towarowych, napiszemy:

```
SELECT Nazwa
FROM Towar
WHERE EXIST (SELECT Id_Grupy
FROM Grupa
WHERE Grupa.Id_grupy=Towar.Id_Grupy);
```

Tabela 5.7. *Podzapytanie wykorzystujące wyrażenie exist*

Nazwa
Tajemnicze zioła na kaszel
Magiczny napój na ból głowy
Vroom
Discipline
Black & White
Palmy (dzikie)
Pomarańcza (mechaniczna)
Sofa błękitna
Zegarek z pozytywką
Wypłata dla Bjorga
Wypłata dla Galadrieli
Wypłata dla Gandalfa
Mieszanka ziół wschodu

Częstym problemem administratora bazy danych jest znalezienie duplikatów rekordów (wielu wierszy danej tabeli zawierających dane o tym samym obiekcie). Przypuśćmy, że przez pomyłkę dodaliśmy do tabeli *Pracownik* dodatkowy wiersz zawierający powtórzone dane o Galadrieli. Jedynie dane przechowywane w kolumnie Id_Pracownika (ponieważ wartości tej kolumny uzupełniane są przez SZBD automatycznie) są unikalne. Jednym z rozwiązań tego problemu może być poniższe podzapytanie zawierające operator `exist`.

```
SELECT *
FROM Pracownik
WHERE EXISTS
   (SELECT Imię
               FROM Pracownik Pracownik_sub
               WHERE  Pracownik_sub.Imię = Pracownik.Imię
               AND  Pracownik_sub.Nazwisko = Pracownik.Nazwisko
               AND  Pracownik_sub.Id_Pracownika <> Pracownik..Id_Pracownika);
```

Wyjaśnienie:
SZBD odczytuje po kolei wszystkie wiersze w tabeli *Pracownik*, sprawdzając, dla których wartości atrybutów *Imię* i *Nazwisko* są identyczne. Następnie sprawdza, dla których z wyselekcjonowanych wierszy wartości atrybutu *Id_Pracownika* są różne. Jeżeli istnieje wiersza spełniający te kryteria, zostaje on wybrany przez zewnętrzną instrukcję `SELECT`.

Tabela 5.8. *Wykorzystanie operatora exist do wyszukiwania duplikatów danych*

ID_Pracownika	Imię	Nazwisko	Ulica	Miasto	Stanowisko	Data_zatrudnienia
4	Galadriela	Yavanna	Szybka 2/5	Śródziemie	Sprzedawca	1999-08-04
5	Galadriela	Yavanna	Szybka 2/5	Śródziemie	Sprzedawca	1999-08-04

Podzapytania z wyrażeniem NOT EXIST

Jeżeli interesuje nas brak pewnego wiersza w bazie danych, a nie jego występowanie, możemy sprawdzić to za pomocą wyrażenia `not exist`. Aby za pomocą operatora `not exist` znaleźć te towary, które nie są przypisane do jakichkolwiek grup towarowych, napiszemy:

```
SELECT Nazwa
FROM Towar
WHERE   NOT EXIST   (SELECT Id_Grupy
FROM Grupa
WHERE Grupa.Id_grupy=Towar.Id_Grupy);
```

Tabela 5.9. *Podzapytanie wykorzystujące wyrażenie not exist*

Nazwa
Golgothian

Podzapytania z operatorami ALL i ANY

Operatory `all` i `any` mogą być stosowane w podzapytaniach zwracających wiele wierszy. Podaje się je w klauzulach `where` i `having` razem z operatorem porównania.

Operator ANY

W przypadku użycia operatora `any` wartość z bieżącego wiersza jest porównywana z każdą wartością podzapytania w oparciu o operator porównania. Jeśli jedno z porównań zwraca wartość *prawda*, to całe porównanie jest traktowane jako prawda. Innymi słowy, wiersz zostanie wybrany, jeżeli wyrażenie jest zgodne z co najmniej jedną wartością wybraną w podzapytaniu.

Aby wybrać towary, których cena zakupu jest większa niż najniższa cena zakupu dowolnego towaru, należy wykonać instrukcję:

```
SELECT Nazwa, Cena_zakupu
FROM Towar
WHERE cena_zakupu > ANY (SELECT DISTINCT cena_zakupu
from Towar);
```

Wyjaśnienie:
Minimalna cena zakupu wynosi 99 zł. SZBD sprawdza ceny zakupu wszystkich towarów i, jeżeli jest ona większa niż 99 złotych, zwraca dany wiersz w wyniku zapytania. Słowo kluczowe `DISTINCT` zapobiega wybieraniu przez niektóre SZBD wielokrotnie tych samych wierszy (cena zakupu *Sofy* jest większa od ceny zakupu kilku innych towarów).

Poniższa instrukcja wykorzystuje operator `any` do wybrania nazw tylko tych towarów, których jednorazowo sprzedano więcej niż 20 jednostek.

```
SELECT Id_Towaru
FROM Operacja
WHERE [ilość jednostek] > ANY (SELECT DISTINCT [ilość jednostek]
FROM Operacja
WHERE [ilość jednostek] >20) AND Id_Typu=3;
```

Wyjaśnienie:
SZBD sprawdza, czy ilość jednostek dla każdej przeprowadzonej operacji jest większa niż 20, następnie sprawdza, czy operacja była operacją sprzedaży i, jeżeli oba warunki są spełnione (operator `and`), zwraca w wyniku nazwę towaru.

Tabela 5.10. *Podzapytanie wykorzystujące operator any*

Nazwa	Cena_zakupu
Tajemnicze zioła na kaszel	1 200,00 zł
Magiczny napój na ból głowy	2 000,00 zł
Vroom	150,00 zł
Discipline	590,00 zł
Palmy (dzikie)	222,00 zł
Pomarańcza (mechaniczna)	450,00 zł
Sofa błękitna	500,00 zł
Zegarek z pozytywką	3 000,00 zł
Wypłata dla Bjorga	3 000,00 zł
Wypłata dla Galadrieli	3 500,00 zł
Wypłata dla Gandalfa	4 000,00 zł
Golgothian	50 000,00 zł
Mieszanka ziół wschodu	2 320,00 zł

Tabela 5.11. *Złożone podzapytanie wykorzystujące operator any*

Id_Towaru
Tajemnicze zioła na kaszel
Magiczny napój na ból głowy
Mieszanka ziół wschodu

Operator ALL

W przypadku użycia operatora `all` warunek musi być spełniony przez wszystkie wartości wybrane w podzapytaniu. Innymi słowy, jeśli wykonujemy porównanie oparte na równości, to wartość z lewej strony równania musi być równa każdej wartości wyniku podzapytania, żeby wynik całości też był prawdziwy. Aby wybrać nazwy wszystkich towarów, których cena sprzedaży przewyższyła cenę sprzedaży któregokolwiek z towarów zakwalifikowanych do grupy *Muzyka*, należy wykonać instrukcje:

```
SELECT Nazwa, Cena_sprzedazy
FROM Towar
WHERE Cena_sprzedazy > ALL (SELECT DISTINCT Cena_sprzedazy
FROM Towar
WHERE Id_Grupy=1);
```

Wyjaśnienie:
Najwyższa cena sprzedaży towarów z grupy *Muzyka* wynosi 900 zł. Zapytanie zwróci listę tych towarów, których cena sprzedaży jest wyższa niż 900 zł.

Tabela 5.12. *Podzapytanie wykorzystujące operator all*

Nazwa	Cena_sprzedazy
Tajemnicze zioła na kaszel	2 000,00 zł
Magiczny napój na ból głowy	3 320,00 zł
Sofa błękitna	1 400,00 zł
Zegarek z pozytywką	7 500,00 zł
Mieszanka ziół wschodu	3 050,00 zł

Zagnieżdżanie podzapytań

Możliwe jest wielokrotne zagnieżdżanie podzapytań. Podzapytanie może być zagnieżdżone zarówno w standardowym zapytaniu, jak i w innym podzapytaniu. Jedyne ograniczenie ilości poziomów zagnieżdżeń wynika z wydajności zapytania. W praktyce ograniczeniem przy zagnieżdżaniu zapytań jest wyłącznie czas ich obliczania przez SZBD.

Podzapytania w klauzuli HAVING

Język SQL umożliwia stosowanie podzapytań zarówno w klauzuli `where`, jak i w klauzuli `having`.

Aby wybrać grupę towarową, w której towary mają najniższą średnią cenę sprzedaży, napiszemy:

```
SELECT Id_Grupy, Avg (cena_sprzedazy)
FROM Towar
GROUP BY Id_Grupy
HAVING Avg  (cena_sprzedazy) = (SELECT Min(AVG (cena_sprzedazy))
FROM Towar);
```
[1]

[1] Niektóre SZBD nie umożliwiają zagnieżdżania funkcji agregujących. W takim wypadku próba wykonania instrukcji

```
SELECT Min(AVG (cena_sprzedazy))
FROM Towar;
```

zakończy się błędem.

Wyjaśnienie:
SZBD oblicza średnią cenę sprzedaży towarów dla poszczególnych grup, a następnie wybiera tę grupę, dla której obliczona wartość jest równa najniższej z obliczonych średnich cen sprzedaży.

Tabela 5.12. *Podzapytanie w klauzuli having*

Id_Grupy	Expr1001
Wewnętrzne	0,00 zł