ORDER BY, LIMIT, TOP, ROWNUM, FETCH: porcjowanie wyniku

> Bez `ORDER BY` kolejność wierszy jest przypadkowym efektem sposobu przechowywania danych, a każdy z czterech silników tej książki ogranicza liczbę zwracanych wierszy własną, niekompatybilną składnią — ta strona pokazuje obie sprawy razem, bo w praktyce zawsze występują w parze.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
ASCII	American Standard Code for Information Interchange: podstawowy zestaw znaków (litery, cyfry) używany w komputerach
ISO	International Organization for Standardization: organizacja, która wydaje normy techniczne używane na całym świecie
SQL	Structured Query Language: język zapytań do relacyjnych baz danych
T-SQL	Transact-SQL: dialekt SQL używany w Microsoft SQL Server, wywodzący się z języka zapytań oryginalnego silnika Sybase

Fiszka

TYLKO ODCZYT

Sortowanie rosnące i malejące

```
SELECT id, wartosc FROM zamowienia ORDER BY wartosc DESC;
```

`ORDER BY` sortuje wynik według wskazanej kolumny. `DESC` sortuje malejąco, `ASC` — rosnąco i jest wartością domyślną, więc w praktyce pomija się je, jeśli sortowanie ma iść od najmniejszej wartości.

Przykład:

```
 id | wartosc
----+---------
107 | 3200.00
104 | 2890.00
101 | 1250.00
106 |  899.00
102 |  430.50
105 |  150.00
103 |    0.00
108 |    0.00
(8 rows)
```

□□ **ASC** = Automatic Self-Calibration: automatyczna samokalibracja czujnika CO2, która zakłada, że co tydzień czujnik zobaczy czyste powietrze o stężeniu około 400 ppm

□□ **TYLKO ODCZYT**

Sortowanie po wielu kolumnach

```
SELECT id, status, wartosc FROM zamowienia ORDER BY status ASC, wartosc DESC;
```

Kolumny w `ORDER BY` rozstrzygają kolejność po kolei — pierwsza ustala główne grupy, kolejna sortuje wewnątrz wierszy, które w pierwszej kolumnie mają tę samą wartość. Tu zamówienia najpierw grupują się alfabetycznie po statusie, a dopiero w obrębie jednego statusu sortują się malejąco po kwocie.

Przykład:

```
 id |    status     | wartosc
----+---------------+---------
103 | anulowane     |    0.00
108 | anulowane     |    0.00
106 | nowe          |  899.00
105 | nowe          |  150.00
104 | w realizacji  | 2890.00
```

```
107 | zrealizowane  | 3200.00
101 | zrealizowane  | 1250.00
102 | zrealizowane  |  430.50
(8 rows)
```

ASC = Automatic Self-Calibration: automatyczna samokalibracja czujnika CO2, która zakłada, że co tydzień czujnik zobaczy czyste powietrze o stężeniu około 400 ppm

TYLKO ODCZYT

Ograniczenie liczby wierszy z przesunięciem: LIMIT / OFFSET

```
SELECT id, wartosc FROM zamowienia ORDER BY wartosc DESC LIMIT 3 OFFSET 2;
```

Składnia PostgreSQL i MySQL. `LIMIT` ustala, ile wierszy zwrócić, `OFFSET` — ile wierszy pominąć od początku posortowanego wyniku, zanim zacznie się liczenie zwracanych. To druga strona wyniku wartości z poprzedniej Fiszki, pomijająca dwa najwyższe zamówienia.

Przykład:

```
 id | wartosc
----+---------
101 | 1250.00
106 |  899.00
102 |  430.50
(3 rows)
```

TYLKO ODCZYT

Ograniczenie liczby wierszy w SQL Server: TOP

```
SELECT TOP (3) id, wartosc FROM zamowienia ORDER BY wartosc DESC;
```

`TOP` stoi zaraz po `SELECT`, nie na końcu zapytania jak `LIMIT`. Bez `ORDER BY` `TOP` zwraca wiersze w kolejności, w jakiej silnik akurat je odczytał — nieokreślonej z tych samych powodów, co w części „Po co to istnieje” niżej.

Przykład:

```
 id | wartosc
----+---------
107 | 3200.00
104 | 2890.00
101 | 1250.00
(3 rows)
```

SQL = Structured Query Language: język zapytań do relacyjnych baz danych

TYLKO ODCZYT

Przesunięcie i limit w SQL Server: OFFSET ... FETCH NEXT

```
SELECT id, wartosc FROM zamowienia
ORDER BY wartosc DESC
OFFSET 2 ROWS FETCH NEXT 3 ROWS ONLY;
```

Odpowiednik `LIMIT`/`OFFSET` w nowoczesnym T-SQL, dostępny od SQL Server 2012. W odróżnieniu od `TOP`, `OFFSET ... FETCH` wymaga obowiązkowego `ORDER BY` — bez niego zapytanie nie skompiluje się w ogóle, co samo w sobie chroni przed poleganiem na przypadkowej kolejności.

Przykład:

```
 id | wartosc
----+---------
101 | 1250.00
106 |  899.00
102 |  430.50
(3 rows)
```

SQL = Structured Query Language: język zapytań do relacyjnych baz danych · **NEXT** = Near-End Crosstalk: przesłuch zbliżny, czyli ile sygnału z jednej pary przecieka do sąsiedniej pary na tym samym końcu kabla · **T-SQL** = Transact-SQL: dialekt SQL używany w Microsoft SQL Server, wywodzący się z języka zapytań oryginalnego silnika Sybase

TYLKO ODCZYT

Ograniczenie liczby wierszy w starszym Oracle: ROWNUM

```
SELECT id, wartosc FROM (
    SELECT id, wartosc FROM zamowienia ORDER BY wartosc DESC
) WHERE ROWNUM <= 3;
```

> **Pojęcie: ROWNUM.** Pseudokolumna Oracle przypisująca każdemu wierszowi kolejny numer w miarę jego pobierania przez silnik — przed sortowaniem, nie po nim. `ROWNUM` w podzapytaniu wewnętrznym nie istnieje, dopóki podzapytanie się nie wykona, więc numerowanie w zapytaniu zewnętrznym dotyczy już posortowanego wyniku. Zagnieżdżenie w podzapytaniu jest tu obowiązkowe — bez niego `ROWNUM` numerowałby wiersze przed sortowaniem, co pokazuje sekcja „Częste błędy”.

Przykład:

```
 ID | WARTOSC
----+---------
107 |    3200
104 |    2890
101 |    1250
```

□□ TYLKO ODCZYT

Ograniczenie liczby wierszy w Oracle 12c i nowszych: FETCH FIRST

```
SELECT id, wartosc FROM zamowienia ORDER BY wartosc DESC FETCH FIRST 3 ROWS ONLY;
```

Składnia zgodna ze standardem SQL:2008, dostępna w Oracle od wersji 12c. Robi dokładnie to, co konstrukcja z `ROWNUM` w podzapytaniu, bez zagnieżdżania — Oracle rozwiązał we własnym silniku ten sam problem, jaki SQL Server rozwiązał przez `OFFSET ... FETCH`.

Przykład:

```
 ID | WARTOSC
----+---------
107 |    3200
104 |    2890
101 |    1250
```

□□ **SQL** = Structured Query Language: język zapytań do relacyjnych baz danych

◫◫ ZMIENIA SYSTEM

Ograniczenie liczby wierszy w Sybase: SET ROWCOUNT

```
SET ROWCOUNT 3;
SELECT id, wartosc FROM zamowienia ORDER BY wartosc DESC;
SET ROWCOUNT 0;
```

`SET ROWCOUNT` nie jest częścią zapytania, tylko oddzielnym poleceniem sesji — ustawia limit obowiązujący dla wszystkich kolejnych instrukcji, dopóki ktoś go nie zresetuje. Zapomnienie o `SET ROWCOUNT 0` na końcu zostawia limit działający na następne, niepowiązane zapytania w tej samej sesji — częsty błąd w skryptach, które kończą się przedwcześnie po wyjątku, zanim reset zdąży się wykonać.

Przykład:

```
 id | wartosc
----+---------
107 | 3200.00
104 | 2890.00
101 | 1250.00
(3 rows affected)
```

◫◫ **SET** = Switch Embedded Teaming: zespół kart sieciowych zbudowany w przełączniku wirtualnym Hyper-V; łączy dwie karty w jedno łącze bez ustawień po stronie przełącznika fizycznego, więc bez LACP

◫◫ TYLKO ODCZYT

Wszystkie warianty ograniczenia wyniku

Silnik	Ograniczenie liczby wierszy	Przesunięcie
PostgreSQL	`LIMIT n`	`OFFSET m`
MySQL / MariaDB	`LIMIT n` (albo `LIMIT m, n`)	`OFFSET m`
SQL Server	`TOP (n)` lub `OFFSET ... FETCH NEXT n ROWS ONLY`	`OFFSET m ROWS` (tylko z FETCH)
Oracle 12c+	`FETCH FIRST n ROWS ONLY`	`OFFSET m ROWS`
Oracle starsze	`ROWNUM <= n` w podzapytaniu	dwa zagnieżdżone `ROWNUM`
Sybase ASE	`SET ROWCOUNT n` przed zapytaniem	brak wbudowanego odpowiednika

Uważaj

- `TOP` bez `ORDER BY` i `LIMIT` bez `ORDER BY` kompilują się bez ostrzeżenia, ale zwracają wiersze w kolejności, której nikt nie gwarantuje.
- `ROWNUM <= n` zastosowany przed `ORDER BY`, zamiast w podzapytaniu, filtruje pierwsze `n` wierszy w kolejności odczytu, a dopiero potem je sortuje.

NEXT = Near-End Crosstalk: przesłuch zbliżny, czyli ile sygnału z jednej pary przecieka do sąsiedniej pary na tym samym końcu kabla

Po co to istnieje

Pojęcie: nieokreślona kolejność. Tabela relacyjna to zbiór wierszy, nie lista — a zbiór z definicji nie ma kolejności. Silnik przechowuje wiersze fizycznie w jakiejś kolejności (zwykle w kolejności wstawienia, na stronach danych na dysku), ale ta fizyczna kolejność nie jest częścią żadnej umowy z zapytaniem. `VACUUM` w PostgreSQL, przebudowa indeksu, `UPDATE` przenoszący wiersz w nowe miejsce po przekroczeniu rozmiaru strony, równoległe wykonanie zapytania na kilku wątkach naraz — każde z tych zdarzeń może zmienić kolejność, w jakiej wiersze trafiają do wyniku zapytania bez `ORDER BY`. Zapytanie, które przez pół roku zwracało dane „w kolejności wstawienia", może zacząć zwracać je inaczej z dnia na dzień bez żadnej zmiany w samym zapytaniu — bo to nigdy nie była gwarancja, tylko przypadkowy efekt uboczny.

Jak to działa

Rozwiń: jak to działa pod spodem

Sortowanie po numerze kolumny. Zamiast nazwy kolumny `ORDER BY` przyjmuje też jej numer pozycyjny w liście `SELECT`.

```
SELECT nazwa, miasto FROM klienci ORDER BY 2;
```

Przykład:

```
           nazwa               |  miasto
-------------------------------+-----------
 Podlaski Handel               | Białystok
 Baltic Trade Sp. z o.o.       | Gdańsk
 Kowalski Handel               | Kraków
 Nowak i Wspólnicy Sp. z o.o.  | Warszawa
 Zielona Dolina S.A.           | Wrocław
(5 rows)
```

Działa identycznie jak `ORDER BY miasto`, ale nie mówi tego wprost. Nie sięgamy po tę składnię w kodzie, który ma przetrwać dłużej niż jedno zapytanie napisane naprędce w konsoli — jeśli ktoś później zmieni kolejność kolumn w `SELECT`, na przykład doda kolumnę na początek listy, `ORDER BY 2` zacznie sortować po zupełnie innej kolumnie, bez żadnego błędu i bez żadnego ostrzeżenia od silnika.

Collation. Pojęcie: collation. Zestaw reguł określających, jak silnik porównuje i sortuje wartości tekstowe — które znaki uznaje za „mniejsze”, jak traktuje wielkość liter i znaki diakrytyczne. Collation `C` (albo `POSIX`) sortuje bajtowo, według wartości liczbowej kodu znaku, więc wielkie litery trafiają przed małymi, a `Ż` jako znak spoza podstawowego zakresu ASCII może wylądować za literą `z`. Collation `pl_PL` sortuje lingwistycznie — `Żabka` znajdzie się w sąsiedztwie innych słów na `Z`, tak jak w polskim słowniku. Ten sam `ORDER BY nazwa` na tej samej tabeli może dać dwa różne porządki w zależności wyłącznie od collation ustawionego dla bazy albo kolumny — temat wart świadomej decyzji przy zakładaniu bazy, nie domyślnego ustawienia zostawionego przypadkowi.

Paginacja przez OFFSET i dlaczego zwalnia. Pojęcie: paginacja. Podział dużego wyniku na kolejne, mniejsze porcje — strony — pobierane osobnymi zapytaniami, zamiast jednym zapytaniem zwracającym wszystko naraz. Najprostszy sposób jej realizacji to rosnący `OFFSET`: strona pierwsza to `LIMIT 20 OFFSET 0`, druga to `LIMIT 20 OFFSET 20`, setna to `LIMIT 20 OFFSET 1980`. Problem w tym, że `OFFSET` nie mówi silnikowi „przeskocz od razu do wiersza numer 1980” — silnik i tak musi posortować cały kwalifikujący się zbiór i policzyć kolejno pierwsze 1980 wierszy, żeby wiedzieć, które odrzucić, zanim zwróci kolejne 20. Im dalsza strona, tym więcej pracy wykonanej wyłącznie po to, żeby zostać wyrzuconą.

> **Pojęcie: paginacja kluczowa.** Alternatywa dla `OFFSET`, w której kolejna strona nie liczy się od początku wyniku, tylko od ostatniego wiersza poprzedniej strony — stąd angielska nazwa keyset pagination, po polsku czasem „paginacja przez klucz” albo „metoda seek”.

```
SELECT id, wartosc FROM zamowienia
WHERE wartosc < 1250.00
```

```
ORDER BY wartosc DESC
LIMIT 3;
```

Przykład:

```
 id | wartosc
----+---------
106 |  899.00
102 |  430.50
105 |  150.00
(3 rows)
```

Zamiast pomijać wiersze przez `OFFSET`, warunek `WHERE wartosc < 1250.00` odwołuje się do wartości z ostatniego wiersza poprzedniej strony (`101`, `1250.00`) i od razu każe silnikowi szukać dalej w indeksie od tego miejsca — bez liczenia i odrzucania wszystkiego, co było wcześniej. Koszt takiego zapytania nie rośnie wraz z numerem strony, tak jak rośnie przy `OFFSET`. Cena tej metody to brak swobodnego skoku na dowolny numer strony — działa dobrze dla „następna strona”, gorzej dla „przejdź od razu na stronę 47”.

Częste błędy

ROWNUM przed ORDER BY zamiast po nim. `SELECT id, wartosc FROM zamowienia WHERE ROWNUM <= 3 ORDER BY wartosc DESC;` wygląda logicznie, ale `ROWNUM` przypisuje numery w kolejności fizycznego odczytu, przed sortowaniem — więc filtr wybiera trzy dowolne wiersze, a `ORDER BY` tylko je sortuje, nie zmieniając, które trzy zostały wybrane. Poprawka to zagnieżdżenie zapytania z `ORDER BY` w podzapytaniu i filtrowanie `ROWNUM` dopiero na zewnątrz, tak jak w Fiszce tej strony.

Rosnący OFFSET w paginacji rankingu z częstymi zmianami danych. Jeśli między pobraniem strony pierwszej a drugiej ktoś wstawi nowy wiersz, który trafia przed granicą sortowania, `OFFSET` przesuwa się o jeden wiersz i ten sam rekord może pojawić się na obu stronach albo zniknąć z obu. Paginacja kluczowa nie ma tego problemu, bo nie liczy pozycji, tylko porównuje wartość klucza.

TOP bez PERCENT i bez ORDER BY w SQL Server. `SELECT TOP (10) * FROM zamowienia;` bez `ORDER BY` zwróci dziesięć wierszy w kolejności, jaką akurat wybrał optymalizator — często zgodną z kolejnością klucza głównego, ale to przypadek, nie gwarancja, i pierwsza zmiana planu wykonania może ją zmienić.

□□ Ćwiczenia

> Zadania sprawdzające.

Ćwiczenie 1 — druga strona wyniku

Zamówienia z Fiszki tej strony sortujemy malejąco po `wartosc`. Piszemy zapytanie w składni PostgreSQL zwracające trzecią stronę wyniku, przy założeniu, że jedna strona ma dwa wiersze.

Odpowiedź

```
SELECT id, wartosc FROM zamowienia ORDER BY wartosc DESC LIMIT 2 OFFSET 4;
```

Strona pierwsza to wiersze 1–2 (`OFFSET 0`), strona druga to wiersze 3–4 (`OFFSET 2`), strona trzecia to wiersze 5–6, czyli `OFFSET 4`:

```
 id | wartosc
----+---------
102 |  430.50
105 |  150.00
(2 rows)
```

Pytania kontrolne

1. Dlaczego zapytanie bez ORDER BY, które przez rok zwracało wiersze w kolejności wstawienia, może pewnego dnia zacząć zwracać je w innej kolejności bez żadnej zmiany w samym zapytaniu?

Odpowiedź

Kolejność wierszy bez `ORDER BY` nigdy nie była gwarantowana — była przypadkowym efektem fizycznego sposobu przechowywania danych. Operacje takie jak `VACUUM`, przebudowa indeksu czy zmiana planu wykonania na równoległy mogą zmienić tę fizyczną kolejność, nie naruszając przy tym żadnej reguły SQL.

2. Dlaczego paginacja przez rosnący OFFSET zwalnia na dalszych stronach, a paginacja kluczowa nie?

Odpowiedź

`OFFSET` każe silnikowi policzyć i odrzucić wszystkie wiersze przed żądaną stroną, więc koszt rośnie liniowo z numerem strony. Paginacja kluczowa filtruje `WHERE` względem wartości ostatniego wiersza poprzedniej strony, więc silnik może od razu wejść w odpowiednie miejsce indeksu, niezależnie od tego, która to strona z kolei.

Powiązane: SELECT: kolejność zapisu a kolejność wykonania · Czym jest indeks: i dlaczego to nie jest „przyspieszacz tabeli” · Skąd się wziął SQL i dlaczego każdy silnik mówi nim inaczej

Źródła: dokumentacja PostgreSQL, rozdział „LIMIT and OFFSET” · dokumentacja Microsoft, „ORDER BY Clause (Transact-SQL)” · Oracle Database SQL Language Reference, „Row Limiting Clause” · ISO/IEC 9075-2:2016 § 7.13

Revision #47
Created 2026-09-24 22:41:49 UTC by Tomasz Konieczny
Updated 2026-10-26 14:10:51 UTC by Tomasz Konieczny