Operatory wyrażeń (logika i formatowanie wewnątrz etapów)

Wprowadzenie:

> Operatory wyrażeń to „funkcje" używane **wewnątrz etapów potoku** (najczęściej `$project`, `$addFields` i `$group`) do obliczania wartości — warunków, działań arytmetycznych, formatowania tekstu i dat. Różnią się od operatorów zapytań (tych z `find()` i `$match`), które jedynie filtrują dokumenty. Operatory wyrażeń *tworzą* nowe wartości.

Wszystkie poniższe komendy wykonuje się w powłoce mongosh, w ramach potoku agregacji (zob. strona „Aggregation pipelines").

> **Kluczowa różnica składni:** operatory porównań w wyrażeniach przyjmują argumenty w **tablicy**, np. `{ $gt: ["$kwota", 100] }`. To nie to samo co forma zapytania `{ kwota: { $gt: 100 } }` używana w `$match` / `find()`. Mylenie tych dwóch form to najczęstszy błąd początkujących.

1. Operator `$cond` – warunek if/then/else

```
db.zamowienia.aggregate([
  { $addFields: {
      kategoriaKwoty: {
        $cond: { if: { $gte: ["$kwota", 500] }, then: "duze", else: "male" }
      }
  } }
])
```

Zwraca jedną z dwóch wartości w zależności od warunku — odpowiednik `if/else`. Powyżej każdemu zamówieniu dopisywane jest pole `kategoriaKwoty` równe `"duze"`, gdy kwota wynosi co najmniej 500, w przeciwnym razie `"male"`.

Skrócona składnia tablicowa (kolejność: warunek, then, else):

```
{ $cond: [ { $gte: ["$kwota", 500] }, "duze", "male" ] }
```

> **Screen:** wynik z dopisanym polem `kategoriaKwoty`.

2. Operator `$switch` – wielokrotny wybór

```
db.zamowienia.aggregate([
  { $addFields: {
      prog: {
        $switch: {
          branches: [
            { case: { $gte: ["$kwota", 1000] }, then: "premium" },
            { case: { $gte: ["$kwota", 500] },  then: "standard" }
          ],
          default: "podstawowy"
        }
      }
  } }
])
```

Sprawdza kolejno warunki z listy `branches` i zwraca wynik pierwszego pasującego (`then`). Jeśli żaden nie pasuje, zwraca `default`. To czytelniejsza alternatywa dla wielu zagnieżdżonych `$cond`. Warunki są sprawdzane od góry, więc kolejność ma znaczenie.

3. Operator `$ifNull` – wartość domyślna dla braków

```
db.zamowienia.aggregate([
  { $addFields: { rabat: { $ifNull: ["$rabat", 0] } } }
])
```

Zwraca pierwszą wartość, jeśli nie jest ona `null` ani nie brakuje jej w dokumencie; w przeciwnym razie zwraca wartość zastępczą. Tutaj brakujące pole `rabat` zostaje uzupełnione zerem, co zabezpiecza dalsze obliczenia przed błędami na wartościach `null`.

> **Screen:** dokument z uzupełnionym polem `rabat`.

4. Operatory porównań w wyrażeniach

```
{ $eq:  ["$status", "oplacone"] }   // równe
{ $ne:  ["$status", "anulowane"] }  // różne
{ $gt:  ["$kwota", 100] }           // wiékse niż (>)
{ $gte: ["$kwota", 100] }           // wiékse lub równe
{ $lt:  ["$kwota", 100] }           // mniejsze niż
{ $cmp: ["$a", "$b"] }              // -1, 0 lub 1
```

Zwracają wartość logiczną (`true` / `false`), więc używa się ich zwykle jako warunku w `$cond` lub `$switch`. Pamiętaj o składni tablicowej — dwa porównywane wyrażenia podaje się jako elementy tablicy.

5. Operatory logiczne – `$and`, `$or`, `$not`

```
{ $and: [ { $gte: ["$kwota", 100] }, { $eq: ["$status", "oplacone"] } ] }
{ $or:  [ { $eq: ["$vip", true] }, { $gt: ["$kwota", 1000] } ] }
{ $not: [ { $eq: ["$status", "anulowane"] } ] }
```

Łączą kilka warunków logicznych. Przyjmują tablicę wyrażeń i zwracają wartość logiczną — przydatne do budowania złożonych warunków w `$cond`.

6. Operatory arytmetyczne

```
{ $add:      ["$kwota", "$rabat"] }   // dodawanie
{ $subtract: ["$kwota", "$rabat"] }   // odejmowanie
{ $multiply: ["$kwota", 1.23] }       // mnożenie (np. brutto)
{ $divide:   ["$suma", "$liczba"] }   // dzielenie
{ $mod:      ["$liczba", 2] }         // reszta z dzielenia
{ $round:    ["$kwota", 2] }          // zaokrąglenie do 2 miejsc
```

Wykonują działania na liczbach. Argumentami mogą być zarówno pola dokumentu (z prefiksem `$`), jak i wartości stałe. Dostępne są też `$abs`, `$ceil` i `$floor`.

7. Operatory łańcuchowe (tekstowe)

```
{ $concat:  ["$imie", " ", "$nazwisko"] }  // sklejanie tekstu
{ $toUpper: "$status" }                    // wielkie litery
{ $toLower: "$email" }                     // małe litery
{ $substr:  ["$kod", 0, 3] }               // wycinek (od, długość)
{ $split:   ["$email", "@"] }              // podział na tablicé
{ $trim:    { input: "$nazwa" } }          // usuniécie spacji z brzegów
```

Służą do manipulacji tekstem. `$concat` łączy fragmenty (uwaga: jeśli któryś jest `null`, wynik też będzie `null` — warto opakować w `$ifNull`). `$split` zwraca tablicę, więc często łączy się go z operatorami tablicowymi.

8. Operator `$dateToString` – formatowanie dat

```
db.zamowienia.aggregate([
  { $addFields: {
      dataTekst: {
        $dateToString: { format: "%Y-%m-%d", date: "$data", timezone: "Europe/Warsaw" }
```

```
    }
  } }
])
```

Zamienia pole typu data na sformatowany tekst według wzorca `format`. Opcjonalny `timezone` przelicza datę na wskazaną strefę przed sformatowaniem (bez niego daty są w UTC).

Najczęstsze symbole formatu:

```
%Y  rok (4 cyfry)      %H  godzina (00-23)
%m  miesiąc (01-12)    %M  minuty (00-59)
%d  dzień (01-31)      %S  sekundy (00-59)
```

> **Screen:** dokument z polem `dataTekst` w formacie RRRR-MM-DD.

9. Operatory na datach – składowe i obliczenia

```
{ $year:      "$data" }   // sam rok
{ $month:     "$data" }   // sam miesiąc
{ $dayOfMonth: "$data" }   // sam dzień
```

Wyciągają pojedynczy składnik daty jako liczbę. Często używane w `$group` do grupowania np. po roku lub miesiącu.

Różnica i przesunięcie dat:

```
{ $dateDiff: { startDate: "$data", endDate: "$$NOW", unit: "day" } }
{ $dateAdd:  { startDate: "$data", unit: "day", amount: 30 } }
```

`$dateDiff` liczy odstęp między dwiema datami w zadanej jednostce (zmienna `$$NOW` to bieżący czas serwera), a `$dateAdd` dodaje określony okres do daty.

10. Operatory tablicowe

```
{ $size:       "$produkty" }              // liczba elementów
{ $arrayElemAt: ["$produkty", 0] }        // element o indeksie
{ $first:      "$produkty" }              // pierwszy element
{ $last:       "$produkty" }              // ostatni element
```

Operują na polach będących tablicami. `$size` bywa przydatne np. po `$lookup`, by policzyć liczbę dopasowań.

Przekształcanie i filtrowanie tablic:

```
// zostaw tylko produkty droższe niż 100
{ $filter: { input: "$produkty", as: "p", cond: { $gt: ["$$p.cena", 100] } } }


// wyciągnij samą nazwé z każdego produktu
{ $map: { input: "$produkty", as: "p", in: "$$p.nazwa" } }
```

`$filter` zwraca podzbiór tablicy spełniający warunek, a `$map` przekształca każdy element. W obu zmienna pętli (tu `$$p`) ma podwójny prefiks `$$`, bo odnosi się do zmiennej lokalnej, a nie do pola dokumentu.

11. Konwersje typów

```
{ $toInt:    "$kwotaTekst" }   // tekst -> liczba całkowita
{ $toDouble: "$kwotaTekst" }   // tekst -> liczba zmiennoprzecinkowa
{ $toString: "$kwota" }        // liczba -> tekst
{ $toDate:   "$dataTekst" }    // tekst -> data
{ $type:     "$kwota" }        // zwraca nazwé typu pola
```

Zamieniają wartości między typami — niezbędne przy porządkowaniu danych zaimportowanych jako tekst. Bardziej elastyczny jest `$convert`, który pozwala wskazać typ docelowy i wartość zastępczą na wypadek błędu konwersji:

```
{ $convert: { input: "$kwotaTekst", to: "double", onError: 0, onNull: 0 } }
```

12. Przykład złożony – operatory w akcji

```
db.zamowienia.aggregate([
  { $addFields: {
      brutto: { $round: [ { $multiply: ["$kwota", 1.23] }, 2 ] },
      miesiac: { $dateToString: { format: "%Y-%m", date: "$data" } },
      etykieta: {
        $cond: [ { $gte: ["$kwota", 500] }, "VIP", "zwykly" ]
      },
      rabat: { $ifNull: ["$rabat", 0] }
  } }
])
```

Jeden etap `$addFields` liczy kwotę brutto (zaokrągloną), wyciąga miesiąc zamówienia jako tekst, nadaje etykietę na podstawie progu kwoty i uzupełnia brakujący rabat zerem. Pokazuje, jak operatory wyrażeń składa się ze sobą, by w jednym przebiegu przygotować dane do raportu.

> **Screen:** wynik z czterema wyliczonymi polami.

Podsumowanie praktyczne

```
// Najczęściej używane operatory wyrażeń
{ $cond:   [ { $gte: ["$kwota", 500] }, "duze", "male" ] }   // warunek
{ $ifNull: ["$rabat", 0] }                                   // wartość domyślna
{ $dateToString: { format: "%Y-%m-%d", date: "$data" } }     // formatowanie daty
{ $concat: ["$imie", " ", "$nazwisko"] }                     // sklejanie tekstu
{ $round:  [ { $multiply: ["$kwota", 1.23] }, 2 ] }          // arytmetyka
```

Revision #1
Created 9 June 2026 09:41:32 by Tomasz Konieczny
Updated 9 June 2026 09:42:25 by Tomasz Konieczny