Podzapytania SQL
Operatory: ANY, ALL, IN, EXISTS

Podzapytania

Przykład:

Mamy następującą tabelę i widok:

```
select * from Klienci;
+-----------+-----------+------------+-------------+
| ID_klient | imie      | nazwisko   | PESEL       |
+-----------+-----------+------------+-------------+
|         1 | Jan       | Kowalski   | 89562314710 |
|         2 | Anna      | Nowakowska | 87985658745 |
|         3 | Katarzyna | Nowicka    | 85214589652 |
|         4 | Andrzej   | Bachleda   | 82145658745 |
+-----------+-----------+------------+-------------+
```

```
select * from vKoszyk;
+-----------+-----------+-------------------+
| ID_koszyk | ID_klient | nazwa_produktu    |
+-----------+-----------+-------------------+
|         1 |         1 | chleb             |
|         2 |         3 | maslo             |
|         3 |         1 | szynka            |
|         4 |         3 | chleb             |
|         5 |         4 | jablko            |
|         6 |         4 | jogurt            |
|         7 |         2 | proszek do prania |
|         8 |         2 | plyn do naczyn    |
+-----------+-----------+-------------------+
```

Podzapytania

Podzapytania – zapytania umiejscowione w innych zapytaniach. Podzapytania można tworzyć w innych podzapytaniach. Stopień zagnieżdżenia zależy od systemu bazodanowego (w tym od ustawień administratora szbd).

Przykłady:

```
select * from Klienci where ID_klient = (select max(ID_klient) from Klienci)
--dane ostatnio utworzonego klienta;

select imie, nazwisko,  PESEL, (select count(*) from vKoszyk where
vKoszyk.ID_klient = Klienci.ID_klient) as liczba_zakupionych_towarow from Klienci;
--przekazanie ID_klient z tabeli Klienci do podzapytania następuje poprzez
odwołanie w postaci: nazwa_tabeli.kolumna, tj. Klienci.ID_klient

select imie, nazwisko, PESEL from (select * from Klienci) as tabelka;
--podzapytanie, które pełni w poleceniu funkcję źródła danych musi zostać w
MySQL/MariaDB nazwane poprzez alias, tj. as tabelka

select P numer_pesel from (select nazwisko, PESEL P from
(select * from Klienci) T) T2;
--zmiana nazwy kolumny w podzapytaniu poprzez alias jest widoczna wyżej, tj. w
zapytaniu, którego źródłem danych jest to podzapytanie
```

Podzapytania

Operator **ANY()**. Warunek w zapytaniu będzie spełniony (true), jeżeli porównywana wartość będzie spełniała warunek z przynajmniej jedną wartością zwróconą w podzapytaniu umieszczonym w operatorze. W podzapytaniu może zostać zwrócona wyłącznie jedna kolumna, ale wiele rekordów (z wyjątkiem przypadku [*]).

Przykłady:

```
select * from Studenci where indeks = ANY (select indeks from Absolwenci);
select * from Klienci where ID_klient = ANY (select ID_klient from vKoszyk);
select PESEL = ANY (select PESEL from Absolwenci), PESEL from Studenci;

select * from Klienci where not ID_klient = ANY (select ID_klient from vKoszyk);
select * from Klienci where ID_klient <> ANY (select ID_klient from vKoszyk);

[*] select * from A where (X,Y) = ANY (select X,Y from B);
```

Podzapytania

Operator **ALL()**. Warunek w zapytaniu będzie spełniony (true), jeżeli porównywana wartość będzie spełniała warunek ze wszystkimi zwróconymi danymi w podzapytaniu umieszczonym w operatorze. W podzapytaniu może zostać zwrócona wyłącznie jedna kolumna, ale wiele rekordów.

Przykłady:

```
select *
from Klienci
where ID_klient <> ALL (
        select ID_klient
        from vKoszyk
    );
```

Podzapytania

Operator/funkcja **EXISTS()**. Jeżeli podzapytanie umieszczone w funkcji zwraca przynajmniej jeden wiersz, to funkcja zwraca wartość *true*, w przeciwnym przypadku *false*. W niektórych systemach bazodanowych (w tym MySQL/MariaDB), EXISTS może występować pomiędzy *select* a *from*.

Przykłady:

```
select * from Klienci where EXISTS (select * from vKoszyk);

select EXISTS(SELECT * from vKoszyk where vKoszyk.ID_klient =
Klienci.ID_klient) as zakup, imie, nazwisko, PESEL from Klienci;
```

Podzapytania

Operator **IN()**. Umożliwia sprawdzenie zawierania się wartości w wartościach wierszy danej kolumny, zwróconych przez podzapytanie. W podzapytaniu może zostać zwrócona wyłącznie jedna kolumna, ale wiele rekordów.

Przykłady:

```
select * from Klienci where ID_klient IN (select ID_klient from
vKoszyk where year(data_zakupu) = '2016');

select imie, nazwisko, PESEL from Studenci where not PESEL IN
(select PESEL from Absolwenci);
```

Przykładowe składnie przedstawionych operatorów
Operator: ANY

Wybierz produkty, których cena jest większa niż cena jakiegokolwiek produktu z kategorii 'Elektronika'.

```
SELECT *
FROM produkty
WHERE cena > ANY (
        SELECT cena
        FROM produkty
        WHERE kategoria = 'Elektronika'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ANY

Wybierz klientów, którzy dokonali więcej zakupów niż jakikolwiek klient z Warszawy.

```
SELECT *
FROM klienci
WHERE ilosc_zakupow > ANY (
        SELECT ilosc_zakupow
        FROM klienci
        WHERE miasto = 'Warszawa'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ANY

Znajdź dostawców, którzy dostarczają więcej produktów niż jakikolwiek dostawca z Krakowa.

```
SELECT *
FROM dostawcy
WHERE ilosc_produktow > ANY (
        SELECT ilosc_produktow
        FROM dostawcy
        WHERE miejscowosc = 'Krakow'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ANY

Wybierz faktury, których kwota jest mniejsza niż jakakolwiek faktura klienta o ID 10.

```
SELECT *
FROM faktury
WHERE kwota < ANY (
        SELECT kwota
        FROM faktury
        WHERE klient_id = 10
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ANY

Wybierz kategorie produktów, które mają więcej produktów niż jakakolwiek kategoria z magazynu w Poznaniu.

```
SELECT *
FROM kategorie
WHERE ilosc_produktow > ANY (
        SELECT ilosc_produktow
        FROM kategorie
        WHERE magazyn = 'Poznań'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ALL

Wybierz produkty, których cena jest większa niż cena wszystkich produktów z kategorii 'Elektronika'.

```
SELECT *
FROM produkty
WHERE cena > ALL (
        SELECT cena
        FROM produkty
        WHERE kategoria = 'Elektronika'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ALL

Wybierz klientów, którzy dokonali więcej zakupów niż wszyscy klienci z Warszawy.

```
SELECT *
FROM klienci
WHERE ilosc_zakupow > ALL(
        SELECT ilosc_zakupow
        FROM klienci
        WHERE miejscowosc = 'Warszawa'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ALL

Znajdź dostawców, którzy dostarczają więcej produktów niż wszyscy dostawcy z Krakowa.

```
SELECT *
FROM dostawcy
WHERE ilosc_produktow > ALL (
        SELECT ilosc_produktow
        FROM dostawcy
        WHERE miejscowosc = 'Krakow'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ALL

Wybierz faktury, których kwota jest mniejsza niż wszystkie faktury klienta o ID 10.

```
SELECT *
FROM faktury
WHERE kwota < ALL (
        SELECT kwota
        FROM faktury
        WHERE klient_id = 10
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: ALL

Wybierz kategorie produktów, które mają więcej produktów niż wszystkie kategorie z magazynu w Poznaniu.

```
SELECT *
FROM kategorie
WHERE ilosc_produktow > ALL (
        SELECT ilosc_produktow
        FROM kategorie
        WHERE magazyn = 'Poznań'
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: EXISTS

Wybierz dostawców, którzy dostarczają jakiekolwiek produkty z kategorii 'Elektronika'.

```
SELECT *
FROM dostawcy
WHERE EXISTS (
        SELECT *
        FROM produkty
        WHERE kategoria = 'Elektronika'
            AND dostawca_id = dostawcy.id
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: EXISTS

Wybierz klientów, którzy dokonali jakiekolwiek zakupy w magazynie w Warszawie.

```
SELECT *
FROM klienci
WHERE EXISTS (
        SELECT *
        FROM zakupy
        WHERE magazyn = 'Warszawa'
            AND klient_id = klienci.id
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: EXISTS

Znajdź kategorie, które mają jakiekolwiek produkty dostępne w magazynie w Krakowie.

```
SELECT *
FROM kategorie
WHERE EXISTS (
        SELECT *
        FROM produkty
        WHERE magazyn = 'Kraków'
            AND kategoria_id = kategorie.id
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: EXISTS

Wybierz faktury, które dotyczą klientów z Krakowa.

```
SELECT *
FROM faktury
WHERE EXISTS (
        SELECT *
        FROM klienci
        WHERE miasto = 'Krakow'
            AND klient_id = klienci.id
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: EXISTS

Wybierz dostawców, którzy dostarczają produkty z najniższą ceną dostępną w magazynie.

```
SELECT *
FROM dostawcy
WHERE EXISTS (
        SELECT *
        FROM produkty
        WHERE cena = (
                SELECT MIN(cena)
                FROM produkty
            )
            AND dostawca_id = dostawcy.id
    );
```

Przykładowe składnie przedstawionych operatorów
Operator: IN – przykłady na podstawie listy wartości

Wybierz wszystkie produkty, które są w kategoriach 'Elektronika', 'Książki' lub 'Ubrania'.

```
SELECT *
FROM produkty
WHERE kategoria IN ('Elektronika', 'Książki', 'Ubrania');
```

Przykładowe składnie przedstawionych operatorów
Operator: IN – przykłady na podstawie listy wartości

Wybierz klientów, którzy mieszkają w Warszawie, Krakowie lub Wrocławiu.

```
SELECT *
FROM klienci
WHERE miasto IN ('Warszawa', 'Krakow', 'Wroclaw');
```

Podzapytania
- wiele poziomów zagnieżdżeń podzapytań

Wybierz dostawców, którzy dostarczają produkt o najwyższej cenie wśród produktów, które są dostępne w magazynie w Warszawie.

```
SELECT dostawca
FROM dostawcy
WHERE produkt_id IN (
        SELECT id
        FROM produkty
        WHERE cena = (
                SELECT MAX(cena)
                FROM produkty
                WHERE magazyn = 'Warszawa'
            )
    );
```

Podzapytania
- wiele poziomów zagnieżdżeń podzapytań

Znajdź kategorie produktów, które mają największą liczbę produktów w magazynie w Krakowie, spośród kategorii, które mają więcej niż 10 produktów.

```
SELECT kategoria
FROM kategorie
WHERE id IN (
        SELECT kategoria_id
        FROM produkty
        WHERE magazyn = 'Krakow'
        GROUP BY kategoria_id
        HAVING COUNT(id) > 10
            AND COUNT(id) = (
                SELECT MAX(prod_count)
                FROM (
                        SELECT kategoria_id,
                            COUNT(id) AS prod_count
                        FROM produkty
                        WHERE magazyn = 'Krakow'
                        GROUP BY kategoria_id
                    ) AS temp_table
            )
    );
```

Podzapytania
- wiele poziomów zagnieżdżeń podzapytań

Znajdź klientów, którzy dokonali zakupu produktu z najwyższą ceną wśród produktów kupionych w magazynie w Poznaniu.

```
SELECT klient
FROM klienci
WHERE id IN (
        SELECT klient_id
        FROM zakupy
        WHERE produkt_id IN (
                SELECT id
                FROM produkty
                WHERE cena = (
                        SELECT MAX(cena)
                        FROM produkty
                        WHERE magazyn = 'Poznań'
                    )
            )
    );
```

Operacje na zbiorach (pionowe)

Operatory:

- union
- except (wprowadzone w MariaDB od wersji 10.3, operator minus - w Oracle)
- intersect (wprowadzone w MariaDB od wersji 10.3)

UNION

Union

Unia – służy do łączenia wyników wielu zapytań w jeden wynik. Podczas łączenia zapytań należy zachować jednakową liczbę kolumn w poszczególnych wynikach.
Domyślnie wynik zapytania nie uwzględnia zdublowanych rekordów (opcja DISTINCT). Użycie opcji ALL wyświetla również powtórzone rekordy.

Schemat zapytania:

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT]
```

Więcej informacji: https://mariadb.com/kb/en/union/

Union - zasada działania

```
mysql> select * from Q1;
+---+
| K |
+---+
| A |
| B |
| C |
+---+
```

```
mysql> select * from Q2;
+---+
| K |
+---+
| C |
| D |
| E |
+---+
```

```
mysql> select * from Q1 union
select * from Q2;
+---+
| K |
+---+
| A |
| B |
| C |
| D |
| E |
+---+
```

```
mysql> select * from Q1
union all select * from Q2;
+---+
| K |
+---+
| A |
| B |
| C |
| C |
| D |
| E |
+---+
```

Union

Przykład składni:

```
select imie from student union select
nazwisko from student order by imie;
```

```
+------------+
| imie       |
+------------+
| Buła       |
| Jan        |
| Janina     |
| Kowalski   |
| Muszka     |
| Nowak      |
| Nowakowska |
| Stefania   |
+------------+
8 rows in set
```

Uwagi:
-***order by*** i ***limit*** umieszczamy na końcu całości zapytania
- ***where*** umieszczamy indywidualnie dla każdego selecta wchodzącego w skład całości
- liczba kolumn poszczególnych selectów musi być jednakowa
- nazwy kolumn wyniku zapytania są takie jak pierwszego selecta - można to zmienić poprzez zastosowanie aliasu (as)

Union

Przykład składni:

```
select imie as X, nazwisko as Y
from student
where nazwisko<>'Michalik'
group by imie, nazwisko having
nazwisko<>'Malczak'
union all
select ID_produkt, nazwa_produktu
from produkt
order by X;
```

```
+----------+-------------+
| X        | Y           |
+----------+-------------+
| 1        | Monitor     |
| 2        | Chleb       |
| 3        | Komputer    |
| 4        | Myszka      |
| 5        | Książka     |
| 6        | Stół        |
| Jan      | Muszka      |
| Jan      | Kowalski    |
| Janina   | Nowakowska  |
| Janina   | Buła        |
| Stefania | Nowak       |
+----------+-------------+
11 rows in set
```

Uwagi:
-***gorup by*** i ***having*** umieszczamy indywidualnie dla każdego selecta wchodzącego w skład całości
- typy zwracanych danych w danej kolumnie mogą być mieszane

Union

Przykład – dane z różnych baz i serwerów (silnik FEDERATED):

```
CREATE TABLE `uzytkownicy` (
`ID_uzytkownicy` int(11) NOT NULL AUTO_INCREMENT,
`login` varchar(60) NOT NULL,
PRIMARY KEY (`ID_uzytkownicy`)
) ENGINE=FEDERATED DEFAULT CHARSET=utf8
CONNECTION='mysql://login:haslo@149.156.136.154/BAZA/tabl_uzytkownicy'
```

```
CREATE TABLE `users` (
`ID_users` int(11) NOT NULL AUTO_INCREMENT,
`login` varchar(60) NOT NULL,
PRIMARY KEY (`ID_users`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
select login from uzytkownicy union all select login from users;
```

Zadanie wykorzystujące operator pionowy UNION

Mamy tabele: pracownicy, kontrahenci.
Utwórz jedną listę zawierającą nazwiska pracowników i kontrahentów.

```
SELECT nazwisko FROM pracownicy
UNION
SELECT nazwisko FROM kontrahenci;
```

Zadanie wykorzystujące operator pionowy UNION

Mamy tabele: zamowienia_wewnetrzne, zamowienia_zewnetrzne.
Znajdź unikalne daty zamówień z tabel zamowienia_wewnetrzne i zamowienia_zewnetrzne.

```
SELECT data_zamowienia FROM zamowienia_wewnetrzne
UNION
SELECT data_zamowienia FROM zamowienia_zewnetrzne;
```

Zadanie wykorzystujące operator pionowy UNION

Mamy tabele: produkty, uslugi.
Znajdź wszystkie unikalne nazwy produktów i usług oferowanych przez firmę.

```
SELECT nazwa FROM produkty
UNION
SELECT nazwa FROM uslugi;
```

Zadanie wykorzystujące operator pionowy UNION

Mamy tabele: sprzedaz, zwroty.
Znajdź wszystkie unikalne ID produktów, które zostały sprzedane lub zwrócone.

```
SELECT produkt_id FROM sprzedaz
UNION
SELECT produkt_id FROM zwroty;
```

INTERSECT

Intersect

Intersect – wyświetla wspólne rekordy pochodzące z poszczególnych wyników zapytań składowych. Podczas łączenia zapytań należy zachować jednakową liczbę kolumn w poszczególnych wynikach.
Operator Intersect nie jest oprogramowany w wielu systemach bazodanowych.

Schemat zapytania:

```
SELECT ...
INTERSECT SELECT ...
[INTERSECT SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT]
```

Więcej informacji: https://mariadb.com/kb/en/intersect/

Intersect - zasada działania

```
mysql> select * from Q1;
+---+
| K |
+---+
| A |
| B |
| C |
+---+
```

```
mysql> select * from Q2;
+---+
| K |
+---+
| C |
| D |
| E |
+---+
```

```
mysql> select * from Q1 intersect select * from Q2;
+---+
| K |
+---+
| C |
+---+
```

Uwagi:
Zasady umieszczania pozostałych operatorów w ramach całości zapytania (tj. group by, having, where, itp.) są analogicznie jak dla UNION

Zadanie wykorzystujące operator pionowy INTERSECT

Mamy tabele: pracownicy, kontrahenci.
Znajdź nazwiska, które występują zarówno w tabeli pracownicy, jak i kontrahenci.

```
SELECT nazwisko FROM pracownicy
INTERSECT
SELECT nazwisko FROM kontrahenci;
```

Zadanie wykorzystujące operator pionowy INTERSECT

Mamy tabele: zamowienia_wewnetrzne, zamowienia_zewnetrzne.
Znajdź wspólne daty zamówień z tabel zamowienia_wewnetrzne i zamowienia_zewnetrzne.

```
SELECT data_zamowienia FROM zamowienia_wewnetrzne
INTERSECT
SELECT data_zamowienia FROM zamowienia_zewnetrzne;
```

Zadanie wykorzystujące operator pionowy INTERSECT

Mamy tabele: produkty, uslugi.
Znajdź wszystkie nazwy produktów i usług, które są takie same.

```
SELECT nazwa FROM produkty
INTERSECT
SELECT nazwa FROM uslugi;
```

Zadanie wykorzystujące operator pionowy INTERSECT

Mamy tabele: sprzedaz, zwroty.
Znajdź ID produktów, które zostały sprzedane i później zwrócone.

```
SELECT produkt_id FROM sprzedaz
INTERSECT
SELECT produkt_id FROM zwroty;
```

Zadanie wykorzystujące operator pionowy INTERSECT

Mamy tabele: ksiazki, wypozyczenia, zamowienia.
Chcemy znaleźć tytuły książek, które zostały zarówno wypożyczone, jak i zamówione, ale tylko te, które mają tytuł zaczynający się na literę "A".

```
SELECT tytul FROM ksiazki
WHERE id IN (
    SELECT id_ksiazki FROM wypozyczenia
    INTERSECT
    SELECT id_ksiazki FROM zamowienia
)
AND tytul LIKE 'A%';
```

EXCEPT

EXCEPT

Except – wyświetla wszystkie rekordy z wyjątkiem tych z zapytania umieszczonym po operatorze except. Podczas łączenia zapytań należy zachować jednakową liczbę kolumn w poszczególnych wynikach.
Operator except/minus nie jest oprogramowany w wielu systemach bazodanowych.

Schemat zapytania:

```
SELECT ...
EXCEPT SELECT ...
[EXCEPT SELECT ...]
[ORDER BY [column [, column ...]]]
[LIMIT]
```

Więcej informacji: https://mariadb.com/kb/en/except/

Except - zasada działania

```
mysql> select * from Q1;
+---+
| K |
+---+
| A |
| B |
| C |
+---+
```

```
mysql> select * from Q2;
+---+
| K |
+---+
| C |
| D |
| E |
+---+
```

```
mysql> select * from Q1
except select * from Q2;
+---+
| K |
+---+
| A |
| B |
+---+
```

```
mysql> select * from Q2
except select * from Q1;
+---+
| K |
+---+
| D |
| E |
+---+
```

Uwagi:
Zasady umieszczania pozostałych operatorów w ramach całości zapytania (tj. group by, having, where, itp.) są analogicznie jak dla UNION

Zadanie wykorzystujące operator pionowy EXCEPT

Mamy tabele: produkty, sprzedane_produkty.
Znajdź nazwy produktów, które nie zostały jeszcze sprzedane.

```
SELECT nazwa_produktu
FROM produkty
EXCEPT
SELECT p.nazwa_produktu
FROM produkty p
    JOIN sprzedane_produkty sp ON p.id_produktu =
sp.id_produktu;
```