2.10. Duomenų grupavimas

Valandų kiekis **konkrečiam projektui**, pvz. Nr. 1:

```
SELECT SUM(Valandos) FROM Vykdymas
WHERE Projektas = 1
```

Kiek **kiekvienam projektui** visi vykdytojai skiria laiko?

SUM stulpelių funkciją reikia taikyti kiekvienai eilučių grupei:

```
SELECT Projektas, SUM(Valandos) AS Valandos
FROM Vykdymas
GROUP BY Projektas
```


Vykdymas

Projektas	*Vykdytojas*	*Statusas*	*Valandos*
1	1	Programuotojas	30
1	2	Dokumentuotojas	100
1	3	Testuotojas	100
1	4	Vadovas	100
2	1	Programuotojas	300
2	2	Analitikas	250
2	4	Vadovas	100
3	1	Programuotojas	250
3	2	Vadovas	400
3	3	Dizaineris	150

Projektas	*Valandos*
1	330
2	650
3	800

100 + 100 + 100 + 30 = 330
300 + 250 + 100 = 650
250 + 400 + 150 = 800

Papildykime rezultatą vykdytojų numeriais:

```
SELECT Projektas, Vykdytojas,
       SUM(Valandos) AS Valandos
FROM Vykdymas GROUP BY Projektas
```

- tai **neteisinga** užklausa.

Vykdymas

Projektas	*Vykdytojas*	*Statusas*	*Valandos*
1	1	Programuotojas	30
1	2	Dokumentuotojas	100
1	3	Testuotojas	100
1	4	Vadovas	100

Grupėje *Projektas* =1 – kelios *Vykdytojas* reikšmės.

Užklausos su grupavimu **SELECT** frazėje, praktiškai, tegali būti:

- stulpelis, paminėtas frazėje **GROUP BY**;
- konstanta;
- reiškinys pagal konstantas ir grupavimo stulpelius;
- jungtinė (stulpelių) funkcija (**SUM**, **MIN**, **MAX** ir kt.);

t.y. tai, kieno rezultatas – 1! reikšmė grupei.

Prieš **GROUP BY** frazę galima naudoti paieškos sąlygą (**WHERE**), kuri yra tikrinama prieš eilučių grupavimą.

```
SELECT Projektas, SUM(Valandos) AS Valandos
FROM Vykdymas
WHERE Valandos > 50
GROUP BY Projektas
```

Projektas	*Valandos*
1	300
2	650
3	800

.... **FROM** *Vykdymas* **WHERE** *Valandos* > 50
GROUP BY *Projektas*

Projektas	*Vykdytojas*	*Statusas*	*Valandos*
1	1	Programuotojas	30
1	2	Dokumentuotojas	100
1	3	Testuotojas	100
1	4	Vadovas	100
2	1	Programuotojas	300
2	2	Analitikas	250
2	4	Vadovas	100
3	1	Programuotojas	250
3	2	Vadovas	400
3	3	Dizaineris	150

.... **FROM** *Vykdymas* **WHERE** *Valandos* > 250
GROUP BY *Projektas*

Projektas	*Vykdytojas*	*Statusas*	*Valandos*
1	1	Programuotojas	30
1	2	Dokumentuotojas	100
1	3	Testuotojas	100
1	4	Vadovas	100
2	1	Programuotojas	300
2	2	Analitikas	250
2	4	Vadovas	100
3	1	Programuotojas	250
3	2	Vadovas	400
3	3	Dizaineris	150

```
SELECT  Projektas,
        SUM(Valandos) AS Valandos
        COUNT(Valandos) AS "Vykdymų skaičius"
FROM Vykdymas
WHERE  Valandos > 250
GROUP BY  Projektas
```

Projektas	Valandos	Vykdymų skaičius
2	300	1
3	400	1

Visų projektų vykdymų (vykdytojų) skaičiai:

```
SELECT Projektas,
       COUNT(*) AS  Vykdytojai
FROM Vykdymas
GROUP BY Projektas
```

Projektas	Vykdytojai
1	4
2	3
3	3

Projektai, kuriuos vykdo daugiau negu 3 vykdytojai:

```
SELECT Projektas, COUNT(*) AS  Vykdytojai
FROM Vykdymas
GROUP BY Projektas
HAVING COUNT(*) > 3
```

Projektas	Vykdytojai
1	4

Sąlyga kiekvienai **grupei**:
GROUP BY su fraze **HAVING**

Galima grupuoti pagal kelis stulpelius.
Tai – grupavimas grupėje.

Vykdytojų skaičiai kiekvienai mokyklai ir kategorijai:

```
SELECT Išsilavinimas, Kategorija,
       COUNT(*) AS Skaičius
FROM  Vykdytojai
WHERE Išsilavinimas IS NOT NULL
GROUP BY Išsilavinimas, Kategorija
ORDER BY Išsilavinimas
```


Vykdytojų skaičiai kiekvienai mokyklai ir kategorijai:

Išsilavinimas	Kategorija	Skaičius
VDU	5	1
VU	2	1
VU	3	2

Vykdytojų valandos, skiriamos visiems projektams:

```
SELECT  Pavardė, SUM(Valandos) AS Valandos
FROM    Vykdytojai, Vykdymas
WHERE  Nr = Vykdytojas
GROUP BY  Pavardė
```

Pateikime ne tik vykdytojo pavardę, bet ir numerį:

```
SELECT  Nr, Pavardė, SUM(Valandos) AS Valandos
FROM Vykdytojai, Vykdymas
WHERE Nr = Vykdytojas
GROUP BY  Nr,  Pavardė
```


Grupavimo rezultatų analizė

Uždavinys: kurie vykdytojai skiria daugiau valandų nei visi vykdytojai vidutiniškai?

Uždavinio sprendimo **žingsniai**:

1. Kiek kiekvienas vykdytojas dirba val.?
2. Kiek vidutiniškai val. dirba vienas vykdytojas?
3. Kurių vykdytojų dirbamos val. viršija vidurkį tarp visų vykdytojų?

Vykdymas

Projektas	Vykdytojas	Statusas	Valandos
1	1	Programuotojas	30
1	2	Dokumentuotojas	100
1	3	Testuotojas	100
1	4	Vadovas	100
2	1	Programuotojas	300
2	2	Analitikas	250
2	4	Vadovas	100
3	1	Programuotojas	250
3	2	Vadovas	400
3	3	Dizaineris	150

Skirtingos spalvos žymi skirtingus vykdytojus – eilučių grupes.

1. Kiek kiekvienas vykdytojas dirba val.?

```
SELECT Vykdytojas, SUM(Valandos) AS Valandos
FROM Vykdymas GROUP BY Vykdytojas)
```

Nr	Valandos
1	580
2	750
3	250
4	200

Šį rezultatą „įsimename“ laikinoje lentelėje:

```
WITH VisųValandos (Nr, Valandos) AS
   (SELECT Vykdytojas, SUM(Valandos) AS Valandos
    FROM Vykdymas GROUP BY Vykdytojas)
```


VisųValandos

Nr	Valandos
1	580
2	750
3	250
4	200

2. Kiek vidutiniškai val. dirba vienas vykdytojas?

```
SELECT AVG(Valandos) FROM VisųValandos
```

?column?
445

3. Nr vykdytojų, dirbančių **daugiau** val. **nei vidutiniškai** vykdytojai dirba valandų:

```
WITH VisųValandos (Nr, Valandos)
     AS (SELECT Vykdytojas, SUM(Valandos)
         FROM Vykdymas GROUP BY Vykdytojas)
SELECT Nr, Valandos
FROM VisųValandos
WHERE Valandos > (SELECT AVG(Valandos)
                  FROM VisųValandos)
```

Nr	Valandos
1	580
2	750

Nr ir **pavardės** vykdytojų dirbančių daugiau val. nei vidutiniškai:

```
WITH VisųValandos (Nr, Valandos) AS
          (SELECT Vykdytojas, SUM(Valandos)
           FROM Vykdymas GROUP BY Vykdytojas)
SELECT A.Nr, Pavardė, Valandos
FROM Vykdytojai AS A, VisųValandos AS B
WHERE A.Nr = B.Nr
   AND Valandos > (SELECT AVG(Valandos)
                   FROM VisųValandos)
```

Nr	Pavardė	Valandos
1	Jonaitis	580
2	Petraitis	750

Tas pat tik rezultatas su vidurkiu:

```
WITH VisųValandos (Nr, Valandos) AS
          (SELECT Vykdytojas, SUM(Valandos)
           FROM Vykdymas GROUP BY Vykdytojas)
SELECT A.Nr, Pavardė, Valandos,
   (SELECT AVG(Valandos) FROM VisųValandos)
FROM Vykdytojai AS A, VisųValandos AS B
WHERE A.Nr = B.Nr
   AND Valandos > (SELECT AVG(Valandos)
                   FROM VisųValandos)
```

Nr	Pavardė	Valandos	4
1	Jonaitis	580	445
2	Petraitis	750	445

Tas pat tik laikinai „įsimenant“ vidurkį:

```
WITH VisųValandos (Nr, Valandos) AS
          (SELECT Vykdytojas, SUM(Valandos)
           FROM Vykdymas GROUP BY Vykdytojas),
     VisųVidurkis (Vidurkis) AS
          (SELECT AVG(Valandos)
           FROM VisųValandos)
SELECT A.Nr, Pavardė, Valandos, Vidurkis
FROM Vykdytojai AS A, VisųValandos AS B,
     VisųVidurkis
WHERE A.Nr = B.Nr
   AND Valandos > Vidurkis
```


Trumpeninis vidurkio skaičiavimas:

```
WITH VisųValandos (Nr, Valandos) AS
          (SELECT Vykdytojas, SUM(Valandos)
           FROM Vykdymas GROUP BY Vykdytojas),
     VisųVidurkis (Vidurkis) AS
          (SELECT AVG(CAST(Valandos AS FLOAT))
           FROM VisųValandos)
SELECT A.Nr, Pavardė, Valandos,
       CAST(Vidurkis AS DECIMAL(10, 2))
FROM Vykdytojai AS A, VisųValandos AS B,
     VisųVidurkis
WHERE A.Nr = B.Nr AND Valandos > Vidurkis
```

Skaičius su slankiu kableliu

10 skaitmenų, iš kurių 2 po kablelio

Užklausos su dviem laikinomis lentelėmis (*VisųValandos* ir *VisųVidurkis*), vaizduojant vidurkį dviem skaitmenimis po kablelio, rezultatas:

Nr	Pavardė	Valandos	Vidurkis
1	Jonaitis	580	445.00
2	Petraitis	750	445.00

2.11. Aibių operacijos

Užklausos rezultatas - eilučių **aibė**.

Todėl prasminga jų rezultatams taikyti aibių operacijas. SQL aibių operacijos:

- **UNION**
- **UNION ALL**
- **INTERSECT**
- **INTERSECT ALL**
- **EXCEPT**
- **EXCEPT ALL**

- **UNION** ir **UNION ALL** - *R1* ir *R2* eilučių sąjunga. Jei nėra **ALL**, rezultate pašalinamos pasikartojančios eilutės.
- **INTERSECT** ir **INTERSECT ALL** - *R1* ir *R2* eilučių sankirta. Jei nėra **ALL**, rezultate pašalinamos pasikartojančios eilutės.
- **EXCEPT** ir **EXCEPT ALL** - *R1* ir *R2* skirtumas. Jei nėra **ALL**, **prieš** operaciją iš *R1* ir *R2* pašalinamos pasikartojančios eilutės.

Tarkime,

- turime **du** vienodos struktūros **užklausų rezultatus** (lenteles) ***R1*** ir ***R2***
- iš viso yra **penkios skirtingos eilutės**, kurias pažymėkime numeriais nuo 1 iki 5.

R1	*R2*	UNION ALL	UNION	EXCEPT ALL	EXCEPT	INTERSECT ALL	INTERSECT
1	1	1	1	1	2	1	1
1	1	1	2	2	5	1	3
1	3	1	3	2		3	4
2	3	1	4	2		4	
2	3	1	5	4			
2	3	2		5			
3	4	2					
4		2					
4		3					
5		3					
		3					
		3					

		3					
		4					
		4					
		4					
		5					

Vykdytojai, nedalyvaujantys nei viename projekte:

```
SELECT Nr FROM Vykdytojai
EXCEPT
SELECT Vykdytojas FROM Vykdymas
ORDER BY 1
```

Šių vykdytojų Nr ir pavardės:

```
SELECT Nr, Pavardė FROM Vykdytojai
EXCEPT
SELECT Vykdytojas, Pavardė
FROM Vykdymas, Vykdytojai WHERE Vykdytojas= Nr
ORDER BY 1
```


2.12. Lentelių reikšmių konstruktorius

SQL2 leidžia apibrėžti lentelę betarpiškai SQL sakinyje.

Lentelės konstruktoriuje visos eilutės išvardinamos betarpiškai sakinyje.

Kiekvienos eilutės stulpelių reikšmės rašomos tarp ( )

Konstantinės lentelės apibrėžiamos sakiniu **VALUES** – lentelių konstruktoriumi:

VALUES (<eilutė>) {,(<eilutė>)}

Tai - užklausa.

Sistemos data:

VALUES (**CURRENT_DATE**) – užklausa, kurios rezultatas: 1 eilutė ir 1 stulpelis.

Užklausos rezultato lentelė, sudaryta iš 1 eilutės ir 3 stulpelių:

```
VALUES (CURRENT_DATE – 1,
        CURRENT_DATE,
        CURRENT_DATE + 1)
```

Tas pat, bet 1-ame stulpelyje, 3-jose eilutėse:

```
VALUES (CURRENT_DATE – 1),
       (CURRENT_DATE),
       (CURRENT_DATE + 1)
```

Sudarytąją lentelę galima rūšiuoti ir grupuoti:

```
VALUES   (CURRENT_DATE – 1),
         (CURRENT_DATE),
         (CURRENT_DATE + 1)
ORDER BY 1 DESC
```

Konstantų lentelę galima apibrėžti **FROM** frazėje:

```
SELECT * FROM
      VALUES (CURRENT_DATE – 1,
              CURRENT_DATE,
              CURRENT_DATE + 1)
```


Sakinyje **VALUES** negalimi stulpelių pavadinimai.

Tai galima padaryti naudojant laikinąją lentelę:

```
SELECT *
FROM  (VALUES (CURRENT_DATE – 1,
               CURRENT_DATE,
               CURRENT_DATE + 1))
      AS Dienos(Vakar, Šiandien, Rytoj)
```

VALUES dažniausiai naudojamas vardinėms konstantoms sužinoti.

Prie datos pridedant (atimant) skaičių, pridedamos (atimamos) dienos:

```
VALUES  (CURRENT_DATE – 1,
         CURRENT_DATE + 1)
```

Bendresnis atvejis – pridedamas ar atimamas laikotarpis (intervalas):

```
VALUES  (CURRENT_DATE – INTERVAL '1 DAY',
         CURRENT_DATE + INTERVAL '1 DAY')
```


Dalis DBVS, ***PostgreSQL*** – tarp jų, leidžia rašyti **SELECT** sakinius ir be frazės **FROM**, frazėje **SELECT** nurodant apskaičiuojamus reiškinius, pvz.,

```
SELECT CURRENT_DATE – 1,
       CURRENT_DATE,
       CURRENT_DATE + 1 ;
```

Šios užklausos rezultatas: 1 eilutė, 3 stulpeliai.

Stulpeliams galima suteikti pavadinimus:

```
SELECT CURRENT_DATE – 1 AS Vakar,
       CURRENT_DATE  AS Šiandien,
       CURRENT_DATE + 1 AS Rytoj ;
```


2.13. Sąlyginiai reiškiniai

Projektų sąrašas su pažymėtu jų ilgaamžiškumu. Tarkime, projektas – trumpalaikis, jei jo trukmė ≤ 6, o jei trukmė > 6, tai jis – ilgalaikis.

```
SELECT Pavadinimas, 'Trumpalaikis'
FROM  Projektai   WHERE  Trukmė <= 6
UNION
SELECT Pavadinimas,  'Ilgalaikis'
FROM  Projektai   WHERE  Trukmė > 6
```


Pavadinimas	*2*	*Trukmė*
Studentų apskaita	Ilgalaikis	12
Buhalterinė apskaita	Ilgalaikis	10
WWW svetainė	Trumpalaikis	2


```
CASE
   WHEN <paieškos sąlyga> THEN NULL | <reiškinys>
  {WHEN <paieškos sąlyga> THEN NULL | <reiškinys>}
 [ ELSE NULL | <reiškinys>]
END
```

```
SELECT Pavadinimas,
     CASE  WHEN Trukmė <= 6
           THEN 'Trumpalaikis'
           ELSE 'Ilgalaikis'  END
FROM Projektai
WHERE  Trukmė IS  NOT  NULL
```


Uždavinys: kiekvienam projektui:

- bendras visų projekto vykdytojų skaičius ir jų skiriamas laikas,
- tie patys duomenys apie informatikų dalyvavimą projekte,
- tie patys duomenys apie statistikų dalyvavimą projekte.

Bendras kiekvieno projekto vykdytojų skaičius ir jų skiriamas laikas:

```
SELECT  Pavadinimas,
   COUNT(DISTINCT Vykdytojas)
          AS "Visi vykdytojai",
   SUM(Valandos) AS "Visų valandos"
FROM Projektai, Vykdymas
WHERE Nr = Projektas
GROUP BY Pavadinimas
```


Bendras kiekvieną projektą vykdančių **informatikų** skaičius ir jų skiriamas laikas:

```
SELECT  Pavadinimas,
    COUNT(DISTINCT Vykdytojas) AS Informatikai,
    SUM(Valandos) AS "Informatikų valandos"
FROM  Projektai, Vykdymas, Vykdytojai
WHERE  Projektai.Nr  = Projektas
    AND Vykdytojai.Nr = Vykdytojas
    AND Kvalifikacija  = 'Informatikas'
GROUP BY Pavadinimas
```


Bendras kiekvieną projektą vykdančių **statistikų** skaičius ir jų skiriamas laikas:

```
SELECT  Pavadinimas,
    COUNT(DISTINCT Vykdytojas) AS Statistikai,
    SUM(Valandos) AS "Statistikų valandos"
FROM  Projektai, Vykdymas, Vykdytojai
WHERE  Projektai.Nr  = Projektas
    AND Vykdytojai.Nr = Vykdytojas
    AND Kvalifikacija  = 'Statistikas'
GROUP BY Pavadinimas
```


Pavadinimas	*Visi vykdytojai*	*Visų valandos*
Studentų apskaita	4	330
Buhalterinė apskaita	3	650
WWW svetainė	3	800

Pavadinimas	*Informatikai*	*Informatikų valandos*
Studentų apskaita	1	30
Buhalterinė apskaita	1	300
WWW svetainė	1	250

Pavadinimas	*Statistikai*	*Statistikų valandos*
Studentų apskaita	1	100
Buhalterinė apskaita	1	250
WWW svetainė	1	400

Rezultatas, kai visi duomenys apie projektą vienoje eilutėje:

Pavadinimas	***Visi vykdytojai***	***Visų valandos***	***...***	***Statistikai***	***Statistikų valandos***
Studentų apskaita	4	330	...	1	30
Buhalterinė apskaita	3	650	...	1	250
WWW svetainė	3	800	...	1	400


```
SELECT  Pavadinimas,
   COUNT(DISTINCT Vykdytojas) AS "Visi vykdytojai",
   SUM(Valandos) AS "Visų valandos",
   COUNT(DISTINCT CASE
         WHEN Kvalifikacija ='Informatikas'
         THEN Vykdytojas END) AS "Visi informatikai",
   SUM( CASE WHEN Kvalifikacija = 'Informatikas'
         THEN Valandos  END)
         AS "Informatikų valandos",
```



```
   COUNT(DISTINCT CASE
         WHEN Kvalifikacija = 'Statistikas'
         THEN Vykdytojas END) AS "Visi statistikai",
   SUM(CASE WHEN Kvalifikacija = 'Statistikas'
         THEN Valandos END) AS "Statistikų valandos"
FROM  Projektai, Vykdymas, Vykdytojai
WHERE  Projektai.Nr  = Projektas AND
        Vykdytojai.Nr = Vykdytojas
GROUP BY Pavadinimas
```


Skaliarinės funkcijos: **NULLIF** ir **COALESCE**:

Sąlyginis reiškinys	Ekvivalenti funkcija
CASE WHEN e1 = e2 **THEN NULL ELSE** e1 **END**	**NULLIF**(e1,e2)
CASE **WHEN** e1 **IS NOT NULL** **THEN** e1 **ELSE** e2 **END**	**COALESCE**(e1,e2)
CASE WHEN e1 **IS NOT NULL** **THEN** e1 **ELSE COALESCE**(e2,...,eN) **END**	**COALESCE**(e1,e2,...,eN)

Vykdytojų kategorijos, jei jas padidintume vienetu, o jei kategorija nebuvo suteikta, priskirtume pirmą kategoriją:

```
SELECT  Pavardė,
        Kategorija AS "Esama kategorija",
        COALESCE(Kategorija, 0) + 1
                AS "Naujoji kategorija"
FROM Vykdytojai
```


2.14. Rekursyviosios užklausos

- **WITH** užklausos gali tapti rekursyviomis greta parašius raktinį žodį **RECURSIVE**.
- Tuomet tarpinį rezultatą apibrėžiančioje užklausoje galima rekursyviai kreiptis į apibrėžiamą laikinąją lentelę.
- Šios užklausos standartizuotos SQL3.
- Rekursijai išreikšti pasitelkiamos aibių operacijos su užklausų rezultatais, dažniausiai, jų sąjunga.
- Primoji sąjungos užklausa nusako pirmąjį, o kita – bendrąjį rekursijos atvejį.

Įprastas rekursyvios užklausos pavyzdys – sveikų skaičių nuo 1 iki 100 sumą:

```
WITH RECURSIVE Skaičiai(N) AS (
  VALUES (1)
    UNION ALL
  SELECT N + 1 FROM Skaičiai WHERE N < 100
)
SELECT SUM(N) FROM Skaičiai
```

- **UNION ALL** pakeitus į **UNION**, rezultatas nekinta;
- Sąlyga **WHERE** yra svarbi, nes be jos, rekursija, o tuo pačiu ir visa užklausa nesibaigtų sėkmingai.

- Rekursyvios užklausos naudingos apdorojant hierarchinius ar grafo struktūros duomenis.
- Apibrėžkime naują lentelę *Kvalifikacijos*, kuri apibrėžtų darbuotojų kvalifikacijų hierarchija.
- Tuomet prasminga ieškoti kvalifikacijų, kurios **tiesiogiai ar netiesiogiai** yra susietos su konkrečia kvalifikacija, pvz. ieškoti specifinių informatikai kvalifikacijų – informatiko specializacijų.

Kvalifikacijos

Pokvalifikacis	*Kvalifikacija*
Medicinos statistikas	Statistikas
DB specialistas	Informatikas
DB projektuotojas	DB specialistas
DB administratorius	DB specialistas
PostgreSQL administratorius	DB administratorius
MySQL administratorius	DB administratorius
Programuotojas	Informatikas


```
WITH RECURSIVE Lygmenys
     (Pavadinimas, Grupė, Lygmuo) AS (
  SELECT Pokvalifikacis, Kvalifikacija, 1
  FROM  Kvalifikacijos
  WHERE  Kvalifikacija = 'Informatikas'
     UNION ALL
  SELECT  A.Pokvalifikacis, A.Kvalifikacija,
          B.Lygmuo + 1
  FROM  Kvalifikacijos A, Lygmenys B
  WHERE  A. Kvalifikacija = B.Poklasis )
SELECT  Pavadinimas, Lygmuo
FROM Lygmenys  ORDER BY Lygmuo, Pavadinimas
```


Užklausos rezultatas – kvalifikacijų, kurios laikomos informatikos kvalifikacijos specializacijomis, pavadinimai ir lygmuo, nusakantis specializacijos gylį informatikos atžvilgiu:

Pavadinimas	*Lygmuo*
DB specialistas	1
Programuotojas	1
DB administratorius	2
DB projektuotojas	2
MySQL administratorius	3
PostgreSQL administratorius	3

2.15. Papildomos paieškos galimybės

Duomenų sumavimas grupuojant

- Taikant **grupavimą**, kiekvienam projektui skaičiavome bendrą visų vykdytojų skiriamą laiką:

```
SELECT Projektas, SUM(Valandos) AS Valandos
FROM Vykdymas
GROUP BY Projektas
```

Projektas	*Valandos*
1	330
2	650
3	800

- Rezultatą, kuriame yra tiek eilučių, kiek yra vykdomų projektų, galima papildyti sumine eilute.
- Tam naudojamas **duomenų sumavimas grupuojant**:

```
SELECT Projektas, SUM(Valandos) AS Valandos
FROM Vykdymas
GROUP BY CUBE (Projektas)
```

Projektas	*Valandos*
1	330
2	650
3	800
NULL	1780

Paskutinėje eilutėje NULL atitinka „Viso:“

- Grupuojant **pagal kelis stulpelius**, CUBE grupavimo rezultatą papildo suminėmis eilutėmis pagal visas grupavimo stulpelių reikšmių kombinacijas.
- Apskaičiuokime vykdytojus kiekvienam išsilavinimui ir kategorijai:

```
SELECT Išsilavinimas, Kategorija,
       COUNT(*) AS "Vykdytojų skaičius"
FROM Vykdytojai
WHERE Išsilavinimas IS NOT NULL
GROUP BY CUBE(Išsilavinimas, Kategorija)
```


Išsilavinimas	*Kategorija*	*Vykdytojų skaičius*
VDU	5	1
VDU	NULL	1
VU	2	1
VU	3	2
VU	NULL	3
NULL	2	1
NULL	3	2
NULL	5	1
NULL	NULL	4

2-oje eilutėje – visų baigusiųjų VDU skaičius, 5-oje – VU, 6-oje – 2-ros kategorijos vykdytojai ir pan.

- Neretai pakanka hierarchinio sumavimo, sumas kaupiant tik kairiesiems grupavimo stulpeliams.
- ROLLUP žymi **hierarchinį sumavimą**.
- Pateiktoje užklausoje CUBE pakeitus į ROLLUP, nelieka sumavimo vien tik kategorijoms (6-8 eil.),

```
SELECT Išsilavinimas, Kategorija,
       COUNT(*) AS "Vykdytojų skaičius"
FROM Vykdytojai
WHERE Išsilavinimas IS NOT NULL
GROUP BY ROLLUP(Išsilavinimas, Kategorija)
ORDER BY Išsilavinimas, Kategorija
```


Šios užklausos rezultatas:

Išsilavinimas	*Kategorija*	*Vykdytojų skaičius*
VDU	5	1
VDU	NULL	1
VU	2	1
VU	3	2
VU	NULL	3
NULL	NULL	4

Sumavimas grupuojant standartizuotas SQL3.
PostgreSQL atsirado ver. 9.5.
SQL3 yra ir daugiau sumavimo grupuojant galimybių.

Eilučių skaičiaus rezultate ribojimas

- Praktikoje, neretai tenka vykdyti užklausas kai lentelėse yra labai daug eilučių.
- Testuojant užklausas, patogu pasinaudoti **rezultato eilučių ribojimo** galimybe.
- SELECT gale fraze

  FETCH FIRST <eilučių skaičius> ROWS ONLY

  apibrėžiama eilučių skaičiaus rezultate riba.
- Fraze OFFSET <eilučių skaičius> nurodoma, kiek pirmųjų eilučių praleisti.


```
SELECT Nr, Pavardė
FROM Vykdytojai
ORDER BY Nr DESC
FETCH FIRST 2 ROWS ONLY OFFSET 1
```

Vykdytojai eilutės perrenkamos *Nr* mažėjimo tvarka, praleidžiam pirmoji didžiausiu *Nr* eilutė ir pateikiamos dvi kitos eilutės:

Nr	*Pavardė*
4	Onaitytė
3	Gražulytė

Rezultato eilučių numeravimas

SELECT frazėje, panaudojus funkciją

ROW_NUMBER() OVER (<rikiavimo tvarka>)

užklausos **rezultato eilutės sunumeruojamos** – apibrėžiamas stulpelis su užklausos rezultato eilutės numeriais.

Eilutės numeruojamos OVER nurodyto stulpelio reikšmių didėjimo ar mažėjimo tvarka, neatsižvelgiant į užklausos rezultato rikiavimo tvarką.

```
SELECT ROW_NUMBER() OVER
        (ORDER BY Pavadinimas ASC) AS Eilė,
        Nr, Pavadinimas
FROM Projektai
ORDER BY Pavadinimas ASC
```

Eilė	*Nr*	*Pavadinimas*
1	2	Buhalterinė apskaita
2	1	Studentų apskaita
3	3	WWW svetainė

Stulpelyje *Eilė* yra eilutės stulpelyje *Pavadinimas* esančios reikšmės numeris tarp visų *Pavadinimas* reikšmių, skaičiuojant reikšmes alfabeto tvarka.

ROW_NUMBER() reikšmė nepriklauso nuo užklausos rezultato rikiavimo:

```
SELECT ROW_NUMBER() OVER
        (ORDER BY Pavadinimas ASC) AS Eilė,
        Nr, Pavadinimas
FROM Projektai
ORDER BY Pavadinimas DESC
```

Eilė	*Nr*	*Pavadinimas*
3	3	WWW svetainė
2	1	Studentų apskaita
1	2	Buhalterinė apskaita

2.16. Sisteminis katalogas

- **Sisteminiame kataloge** (sisteminėse lentelėse) saugoma DB struktūros (lentelių, stulpelių,..) aprašas.
- RDB saugo pakankamai detalų savo pačios aprašą.
- Sisteminės lentelės sukuriamos automatiškai, DB sukūrimo metu.
- Sisteminio katalogo keitimas - išimtinė DBVS teisė.
- Peržiūrėti gali praktiškai visi vartotojai.
- Aprašas pateikiamas konkrečios DBVS dokumentacijoje.

Vykdydama SQL sakinius, DBVS pastoviai kreipiasi į sisteminį katalogą. Pvz., kad įvykdyti užklausą dviems lentelėms, DBVS turi:

- patikrinti, ar egzistuoja tos dvi lentelės;
- patikrinti, ar dabartinis vartotojas, turi teisę kreiptis į jas;
- patikrinti, ar lentelėse yra stulpeliai, nurodyti užklausoje;
- nustatyti, kuriai lentelei priklauso stulpeliai;
- kiekvienam stulpeliui nustatyti duomenų tipą.

- Visa reikiama informacija yra iš anksto apibrėžtose **lentelėse.** Todėl DBVS paieškai gali naudoti ypač efektyvius metodus ir algoritmus.
- Sisteminių lentelių vardai skirtingose DBVS skiriasi.
- **PostgreSQL** sisteminio katalogo lentelės yra **schemoje** ***pg_catalog***.
- **PostgreSQL** sisteminio katalogo informacija standartizuotai pateikta (virtualiose) lentelėse, esančiose **schemoje** ***information_schema***.

Lentelių schemos

- DB-je visos lentelės yra logiškai padalintos į **schemas**.
- **Lentelės vardas** turi būti **unikalus schemoje**.
- DB-je gali būti kelios lentelės tuo pačiu vardu, bet skirtingose schemose.
- Visas (pilnas) lentelės vardas – <schema>.<lentelės vardas>

Pvz. *Stud.Projektai*, *Temp.Projektai*, *Aaaa.Projektai* - tai **skirtingos** lentelės.

Schemos kuriamos SQL sakiniu

```
CREATE SCHEMA <schemos vardas>
```

Pvz.,

```
CREATE SCHEMA Stud
```

Tuščią schemą (nėra jokių lentelių ir kt. objektų) galima sunaikinti:

```
DROP SCHEMA <schemos vardas>
```

Pvz.,

```
DROP SCHEMA Stud
```


Numatytoji schema: vartojo prisijungimo vardas.

Jei prisijungta

```
psql -U stud Darbai
```

tai kreipinys į lentelę *Projektai* yra tapatus *Stud.Projektai,* t.y.

```
SELECT * FROM Stud.Projektai
SELECT * FROM Projektai
```

yra tapačios užklausos.

Užklausa kitai lentelei:

```
SELECT * FROM Mag.Projektai
```

Sisteminių lentelių **pavyzdžiai**:

*information_schema.**tables*** – visos DB lentelės

*information_schema.**columns*** – visų lentelių visi stulpeliai

*information_schema.**triggers*** – trigeriai

*information_schema.**views*** – virtualiosios lentelės

Visas aprašas: ***PostgreSQL*** dokumentacijoje.

Kiekvienas DB objektas (lentelė, view ir kt.) nusakomas pilnu vardu:

<schema (angl. *schema*)>.<vardas (angl. *name*)>

Sisteminių lentelių raktai:

Information_Schema.Tables:
Table_Schema, Table_Name

Information_Schema.Views:
View_Schema, View_Name

Information_Schema.Columns:
Table_Schema, Table_Name, Column_Name

Duomenys apie visus visų lentelių stulpelius:

```
SELECT * FROM information_schema.columns
SELECT * FROM INFORMATION_SCHEMA.COLUMNS
```

Lentelės *Information_Schema.Tables* stulpelių vardai ir jų tipai:

```
SELECT  Column_Name, Data_Type
FROM    Information_Schema.Columns
WHERE   Table_Schema = 'information_schema' AND
        Table_Name = 'tables'
```


Visų lentelių visų stulpelių vardai ir jų tipai:

```
SELECT  Table_Schema, Table_Name,
        Column_Name, Data_Type
FROM    Information_Schema.Columns

SELECT  TABLE_SCHEMA, TABLE_NAME,
        COLUMN_NAME, DATA_TYPE
FROM    INFORMATION_SCHEMA.COLUMNS
```

šios užklausos – tapačios.

Lentelės *Stud.Projektai* visų stulpelių vardai ir tipai:

```
SELECT  Column_Name, Data_Type
FROM  Information_Schema.Columns
WHERE Table_Schema = 'stud'  AND
        Table_Name = 'projektai'

SELECT  Column_Name, Data_Type
FROM  Information_Schema.Columns
WHERE Table_Schema = 'stud'  AND
        Table_Name = 'Projektai'
```

- kita lentelė!

Visų lentelių stulpelių skaičiai:

```
SELECT  Table_Schema, Table_Name, COUNT(*)
FROM    Information_Schema.Columns
GROUP BY  Table_Schema, Table_Name

SELECT  Table_Name, COUNT(*)
FROM    Information_Schema.Columns
GROUP BY  Table_Name
```

– logiškai **neteisinga**!

Lentelių, esančių sisteminio katalogo schemoje *Information_Schema*, vardai:

```
SELECT  Table_Name
FROM    Information_Schema.Tables
WHERE   Table_Schema = 'information_schema'
ORDER BY  1

SELECT DISTINCT  Table_Name
FROM    Information_Schema.Columns
WHERE   Table_Schema = 'information_schema'
ORDER BY  1
```

- neefektyvi!

Visų pastoviųjų (ne laikinųjų) realiųjų (ne virtualiųjų) lentelių stulpelių skaičiai:

```
SELECT  A.Table_Schema, A.Table_Name, COUNT(*)
FROM    Information_Schema.Columns  A,
        Information_Schema.Tables   B
WHERE   A.Table_Schema = B.Table_Schema  AND
        A.Table_Name   = B.Table_Name    AND
        B.Table_Type = 'BASE TABLE'
GROUP BY  A.Table_Schema, A.Table_Name
```

Lentelės, esančios schemoje *stud*:

```
SELECT  Table_Name
FROM    Information_Schema.Tables
WHERE   Table_Schema =  'stud'
ORDER BY  1
```

Visos lentelės:

```
SELECT  Table_Schema, Table_Name
FROM    Information_Schema.Tables
ORDER BY  1, 2
```


Visos lentelės, neturinčios trigerių:

```
    SELECT  Table_Schema, Table_Name
    FROM    Information_Schema.Tables
EXCEPT
    SELECT  Table_Schema, Table_Name
    FROM    Information_Schema.Triggers
ORDER BY  1, 2
```