Prozorske funkcije u SQL-u

Matić, Josip

VELEUČILIŠTE U VIROVITICI

JOSIP MATIĆ

PROZORSKE FUNKCIJE U SQL-u

ZAVRŠNI RAD

Virovitica, 2022.

VELEUČILIŠTE U VIROVITICI

Preddiplomski stručni studij Elektrotehnike, smjer Telekomunikacije i informatika

Josip Matić

PROZORSKE FUNKCIJE U SQL-u

ZAVRŠNI RAD

radi stjecanja akademskog zvanja
stručnog prvostupnika/stručne prvostupnice elektrotehnike

Virovitica, 2022.

Veleučilište u Virovitici

Preddiplomski stručni studij Elektrotehnike - Smjer Telekomunikacije i informatika

OBRAZAC 1b

ZADATAK ZAVRŠNOG RADA

Student/ica: JOSIP MATIĆ **JMBAG: 1312104277**

Imenovani mentor: mr. sc. Damir Vuk, v. pred.

Imenovani komentor: Enes Ciriković, dipl. ing., v. pred.

Naslov rada:

Prozorske funkcije u SQL-u

Puni tekst zadatka završnog rada:

Student je dužan izraditi rad koristeći pritom relevantne izvore literature.
U radu treba obraditi pojam, sintaksu i upotrebljivost prozorskih (window) funkcija u SQL jeziku. Potrebno je objasniti standardne prozorske funkcije u SQL-u, te napraviti komparaciju njihove sintakse i funkcionalnosti u nekoliko značajnijih DBMS sustava (MySQL, SQL Server,Postgresql, SQLite…). Teorijski dio treba popratiti cjelovitim primjerom SQL upita i rezultata.
Student je dužan kontinuirano tijekom izrade rada (najmanje jednom tjedno), obavješatavati mentora o napretku i uvažavati njegove primjedbe, a tjedan dana prije konačne predaje rada dati ga mentoru na uvid i pregled.

Datum uručenja zadatka studentu/ici: 12.10.2022.
Rok za predaju gotovog rada: 11.11.2022.

Mentor:

mr. sc. Damir Vuk, v. pred.

Komentor:

Enes Ciriković, dipl. ing., v. pred.

Enes Ciriković

VELEUČILIŠTE U VIROVITICI **Završni rad**
Temeljna dokumentacijska kartica

PROZORSKE FUNKCIJE U SQL-u

Sažetak

U radu se obrađuje tema prozorskih funkcija u SQL-u. Rad obuhvaća teorijsku obradu, ali i praktičnu primjenu pomoću više alata za manipulaciju baza podataka. U radu je korištena literatura iz internetskih izvora, knjiga i časopisa, U teorijskom dijelu obrađeni su koncepti, klauzule i prozorske funkcije SQL standarda. U praktičnom dijelu korišteni su i provjeravani jednostavniji primjeri implementirani u četiri sustava za upravljanje bazama podataka: PostgreSQL, SQL server, SQLite i MySQL. Na primjerima su utvrđene sličnosti i razlike u implementaciji u tim sustavima za upravljanje bazama podataka.

(35 stranica, 35 slika, 1 tablica, 10 referenci)

Ključne riječi: OVER, ORDER, PARTITION, Prozorske funkcije, SQL

Mentor: mr. sc. Damir Vuk, v. pred.

Komentor: dipl. ing. Enes Ciriković, v. pred.

VIROVITICA UNIVERSITY OF APPLIED SCIENCES **Bachelor's Thesis Basic documentation card**

SQL WINDOW FUNCTION

Abstract

The paper deals with the topic of window functions in SQL. The work includes theoretical processing, but also practical application using several tools for database manipulation. Literature from internet sources, books and magazines was used in the paper. Concepts, clauses and window functions of the SQL standard were covered in the theoretical part. In the practical part, simpler examples implemented in four database management systems were used and checked: PostgreSQL, SQL server, SQLite and MySQL. Similarities and differences in the implementation of these database management systems have been determined on the examples.

(35 pages, 35 figures, 1 tables, 10 references)

Keywords: OVER, ORDER, PARTITION, SQL, Window function

Supervisor: mr. sc. Damir Vuk, v. pred.

Co-supervisor: dipl. ing. Enes Ciriković, v. pred..

SADRŽAJ

I. UVOD

1.1. Predmet i cilj rada

U ovom radu analizira se pojam prozorske funkcije u SQL-u. Iako nije često zastupljena tema, obuhvaća znakovite prednosti. U radu će se analizirati problematika vezana uz ovu temu, a sastoji se od teorijskog i praktičnog dijela.

Svrha ovoga rada analiza je potencijala prozorske funkcije u SQL-u koje predstavljaju novitet u analitičkom prikazu podataka. Cilj rada je prikazati utjecaj alata za manipulaciju baza podataka na prozorskim funkcijama u SQL-u. Početna pretpostavka je da prozorske funkcije imaju znakovit potencijal prikaza podataka koji se temelji na bogatom jeziku SQL.

1.2. Izvori podataka i metode prikupljanja

Za pisanje teorijskog dijela rada koristila se domaća i strana literatura. S obzirom na činjenicu da se radi o relativno novoj temi, većina literature obuhvaća stranu literaturu u vidu znanstvenih i stručnih članaka, dostupnih u online bazama podataka. Za izradu praktičnog dijela korištena je baza podataka koju sam prethodno sam izradio, a izgrađena je u četiri SUBP-a.

1.3. Sadržaj i struktura rada

Rad je organiziran u šest cjelina. Prve četiri cjeline odnosit će se na teorijski dio rada, dok glavnina rada, tj. praktični dio rada slijedi u petoj i šestoj cjelini. Na početku rada je uvod u teoriju SQL-a i prozorskih funkcija. Praktični dio rada odnosi se na sličnosti i razlike prozorskih funkcija kroz implementaciju u četiri SUBP-a.

II. POJAM SQL-a

2.1. Uvod u SQL

SQL (eng. Structured Query Language) je standardizirani-strukturirani jezik za manipulaciju podataka u bazi podataka. Preteča SQL jezika datira početkom 1970-tih godina, nazivao se SEQEL, a nastao je u kompaniji IBM.

Primarna ideja SQL-a je standardiziranje jezika koji će raditi u većini različitih programa za manipulaciju baza. SQL jezik standardiziran je 1986. godine ANSI standardom i 1987. godine ISO standardom. Tijekom godina nastanka mijenjaju se i dodaju nove funkcionalnosti samoga jezika, ali se i jezik prilagođava radnom SUBP-u (eng. DataBase Management System, hrv. Sustav za upravljanje bazom podataka). Svaki SUBP ima sličnu, ali ne i istu sintaksu SQL jezika. (Kramberge et. al, 2018: 47-50)

Većina modernih aplikacija ima potrebu za bazom podataka, za manipulacijom podataka. SQL pomaže programerima u upisu, izmjeni, brisanju i pregled podataka, bez izvornog ulaska na SUBP, ukoliko je uspostavljena komunikacija između SUBP-a i aplikacije.

Razvijanje modela koji će omogućiti brži i ekonomičniji pristup bazama podataka razvija znanstvenik Edgar Frank Codd (1923.-2003.). Koji je dao ideje za razvijanje relacijske baze podataka, Coddove ideje provelo je u praksi nekoliko tvrtki, među kojima je i Oracle. Kasnije relacijski model dobiva niz nadogradnji koje ga čine danas kakvim jest.[1]

SQL je relacijski jezik koji omogućuje: DDL(eng. Data Definition Language) i DML(eng. Data Manipulation Language). Unutar skupine DDL naredbi nalaze se DCL(eng. Data Control Language) naredbe.

SQL relacijski jezik omogućuje :

- Definiciju sheme baze podataka (DDL)
- Definiciju Manipulacije podacima (DML)
- Jezik za kontrolu/upravljanje BP (DCL)

[1] Više podataka o Codd-u: https://www.britannica.com/biography/Edgar-Frank-Codd

2.2. Definicijska shema baze podataka (DDL)

DDL su funkcije koje se odnose na osnovne operacije nad bazom.

Prikaz nekih DDL naredbi:

- CREATE (stvaranje nove tablice)
- ALTER (izmjena sadržaja u tablici)
- DROP (brisanje tablice)

2.3. Definicija manipulacije podacima (DML)

Pomoću DDL-a strukturiramo tablicu, nakon čega trebamo popuniti bazu i tu dolazi DML do značaja. Pomoću DML funkcija možemo umetnuti, izmijeniti, obrisati i dohvatiti informacije iz tablice.

Prikaz nekih DML naredbi:

- SELECT (dohvaća podatke iz tablice)
- INSERT (ubacuje n-torku)
- UPDATE (izmjena n-torke)
- DELETE (brisanje n-torke)

2.4. Jezik za kontrolu/upravljanje BP (DCL)

DCL je jezik kontrole podataka, omogućuje konfiguraciju sigurnosti baza što čini administrator.

Najjednostavniji SQL podskup, jer se sastoji od samo tri naredbe:

- GRANT (davanje dopuštenja)
- REVOKE (uzimanje dopuštenja)
- DENY (sprječavanje korisnika da primi dopuštenje)

III. POJAM I VAŽNOST PROZORSKIH FUNKCIJA

3.1. Nastanak prozorske funkcije

Prozorske funkcije predstavljaju novije proširenje SELECT komande u SQL-u. Korištenje baza podataka u poslovanju eksponencijalno raste, ali i potražnja prema analitičkim prikazima podataka. Uvođenjem prozorskih funkcija (eng. window function), nudi se mogućnost računanja agregatnih vrijednosti na razini svakog retka. To je značajno unaprjeđenje u odnosu na dosadašnje agregatne funkcije koje koriste GROUP by klauzulu.

Prozorske funkcije definirane su ISO SQL:2003 standardom, a naknadnim izmjenama detaljnije su razrađene, od kojih se najviše ističe standard ISO SQL:2008. Nisu podržane od strane mnogih SUBP-a, kao što je primjer s podrškom SQL jezika.

3.2. Pojam i važnost

Razvojni programeri su ih izbjegavali u svom radu, jer su nove, te su kratko zastupljene u korištenju. Donose značajnija odstupanja u sintaksi od alternativnih pristupa. Ponekad su nazivane analitičkim funkcijama ili analitičkim okvirima.

Prozorske funkcije najjednostavnije se definiraju kao funkcije izračuna, koje vrše izračun preko skupa redaka, a rezultat vraćaju za svaki samostalni redak. Usporedive su s vrstom izračuna koje se može izvesti agregatnom funkcijom, ali nisu iste. U osnovi, izvode neke izračune koji su slični agregatnoj funkciji, ali s dodatnom prednošću, jer omogućavaju pristup svakom individualnom retku.

Članak na web stranici *sqlite.org* navodi: „Prozorska funkcija se razlikuje od ostalih SQL funkcija, po svojoj jedinstvenoj karakteristici u sadržavanju klauzule OVER. Ukoliko funkcija sadrži klauzulu OVER, onda je to sigurno prozorska funkcija.“

Ujedno imaju pojednostavljeni i skraćeni kod, koji je čitljiviji i učinkovitiji od alternativnih pristupa. Prozorske funkcije nude mogućnost grupiranja redova podataka na specifične načine. Pogodne su kod analitičkog i statističkog prikazivanja podataka, jer lagano ističu odstupanja individualnog retka od vrijednosti funkcije.

Važnost funkcija prozora ogleda se u jednostavnosti prikazivanja odstupanja vrijednosti n-torke od skupne vrijednosti, koja može biti:

- podijeljena na particije,
- suma nekih vrijednosti,
- redoslijedna vrijednost i
- kumulativna vrijednost

Teorija je popraćena primjerima putem baze koja je kreirana isključivo za pisanje završnog rada. Baza je samostalno kreirana, te svako podudaranje sa stvarnim osobama je slučajno. Prozorske funkcije nisu podržane od strane mnogih SUBP-a, zato je baza primijenjena u slijedećim alatima: SQL server, MySQL, PosqreSQL, SQLite. Na slici 1. prikazan je DDL kod kreirane baze 'osoblje', a na slici 2. prikazan je izgled relacije 'osoblje'.

Slika 1. Prikaz DDL koda baze 'osoblje'

```
1  CREATE TABLE `osoblje` (
2    `ID_zaposlenika` int(11) NOT NULL AUTO_INCREMENT,
3    `Ime_zaposlenika` varchar(45) NOT NULL,
4    `Prezime_zaposlenika` varchar(45) NOT NULL,
5    `Placa` int(11) DEFAULT NULL,
6    `Radni_staž_u_godinama` int(11) DEFAULT NULL,
7    `Struka` varchar(45) DEFAULT NULL,
8    `datum_rodenja` date DEFAULT NULL,
9    PRIMARY KEY (`ID_zaposlenika`)
10 ) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8mb4
```

Izvor: autor

Slika 2. Relacija 'osoblje'

ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	Radni_staž_u_godinama	Struka	datum_rodenja
1	Josip	Matić	6800	0	Programer	1997-05-05
2	Nikica	Polić	10000	10	Ekonomist	1987-12-03
3	Lorena	Varga	9000	5	Programer	1992-12-30
4	Carlos	Petrović	7800	1	Programer	1999-01-04
5	Monika	Veselovac	7500	2	Ekonomist	1997-04-08
6	Alen	Nikić	8200	3	Elektroinžinjer	1989-12-24
7	Ivana	Horvat	5500	2	Ekonomist	1997-04-08
8	Dolores	Perić	10200	10	Programer	1989-09-20
9	Hrvoje	Nikolić	12850	7	Programer	1991-03-02
10	Ana	Dokić	4500	0	Trgovac	1999-12-11
11	Hrvoje	Nad	6500	1	Ekonomist	1999-08-25
12	Ivan	Soldo	14000	11	Elektroinžinjer	1990-05-12
13	Tomislav	Agatić	6700	1	Ekonomist	1999-08-08

Izvor: autor

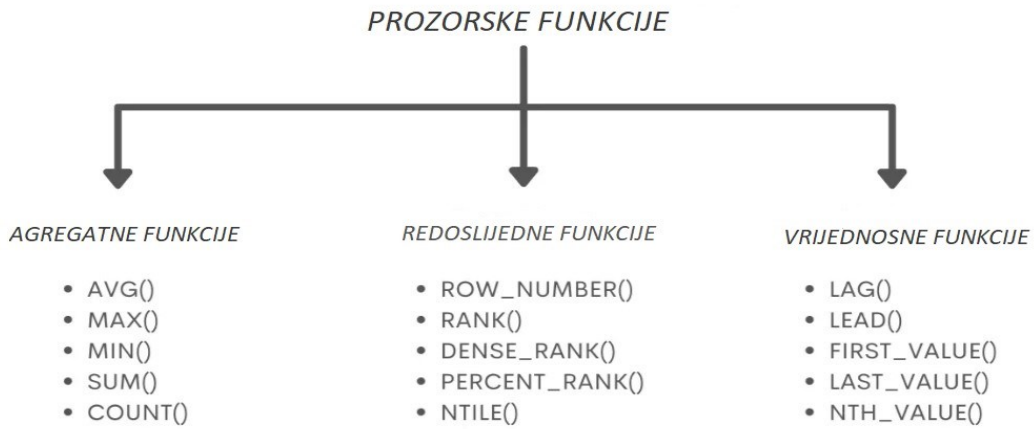
3.3. Kategorizacija prozorskih funkcija

Funkcije prozora mogu biti primijenjene na slijedećim funkcijama:

- Agregatne funkcije
- Redoslijedne funkcije
- Vrijednosne funkcije

Detaljna podjela prozorskih funkcija prikazana je na slici 3.

Slika 3. Podjela prozorskih funkcija



Izvor : prijevod autor, izvorna slika na eng. jeziku, dostupna na: https://towardsdatascience.com/a-guide-to-advanced-sql-window-functions-f63f2642cbf9

3.3.1. Agregatne funkcije

Agregatna funkcija (eng. aggregate function) izvodi matematičke operacije koje vrše: operaciju izračuna, operaciju brojanja ili određuje pozicijsku vrijednost. Primjeri nekih agregatnih funkcija:

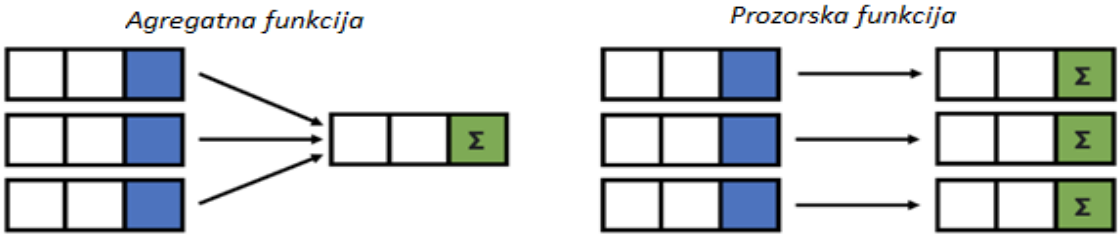
- AVG() – vraća srednju vrijednost, moguće ju je koristiti samo ako je varijabla brojčanog tipa.
- SUM() – vraća sumiranu vrijednost određenih redaka, moguće ju je koristiti samo ako je varijabla brojčanog tipa.
- MAX() i MIN() – vraćaju najveću, odnosno najmanju vrijednost, moguće ju je koristiti samo ako je varijabla brojčanog tipa.

- COUNT() – vraća koliko redaka postoji, moguće ju je koristiti čak i ako varijabla nije brojčanog tipa.
- FIRST()- vraća prvu vrijednost, moguće ju je koristiti čak i ako varijabla nije brojčanog tipa.
- LAST-() vraća posljednju vrijednost, moguće ju je koristiti čak i ako varijabla nije brojčanog tipa.

Agregatna funkcija dopušta primjenu funkcija koje mogu djelovati samostalno ili u sklopu funkcije prozora. Glavna karakteristika koja ih razlikuje je u pristupu prikazivanja podataka.

Samostalno djelovanje agregatne funkcije, uvijek u izlaznoj relaciji vraća skalar[2] (eng. Scalar), vrši operacije na skupu redaka, a rezultat operacije je vrijednost svedena na jedan redak. Korištenje klauzule GROUP by veže se uz samostalno djelovanje agregatne funkcije ukoliko se žele grupirati redci, prema nekoj grupi ili više grupa. Ukoliko agregatna funkcija djeluje sa mogućnosti prozora, onda izlazna relacija vraća rezultat za svaki individualni redak. Glavna razlika prikazana je na slici 4.

Slika 4: Glavna razlika između prozorske i agregatne funkcije



Izvor : prijevod autor, izvorna slika na eng. jeziku, preuzeta sa: towards dana science, https://towardsdatascience.com/a-guide-to-advanced-sql-window-functions-f63f2642cbf9

Najznačajnija sličnost između djelovanja agregatne funkcije i funkcije prozora je u korištenju sličnih funkcija za izvođenje operacija npr.: AVG(), SUM(), COUNT(), MIN() i MAX(). Obje funkcije vrše operacije na više redova tj. vrijednosti n-torke. Grupiranje podataka proizvoljno se određuje u jednom redu ili više njih.

Razlika između funkcija u korištenju je klauzule OVER(), te izlaznom rezultatu. Izlazni rezultat korištenja agregatnih funkcija je skalar, odnosno više skalara, ako se vrši

[2] Skalar- veličine koje možemo izraziti samo jednim podatkom, najčešće brojčanom vrijednošću.

grupiranje po određenim atributima, dok prozorska funkcija vraća rezultat za svaki individualni redak. (Koidan K, 2020.)

Na slici 5. i 6. prikazana je razlika korištenja agregatne i prozorske funkcije. Korištena relacija u primjeru je 'osoblje'. Relacija klauzule GROUP BY prikazuje prve predstavnike grupa, te sumiranu plaću po grupi, a grupa je razdvojena po struci. Relacija prozorske funkcije prikazuje sve zaposlenike, te sumiranu plaću za sve zaposlenike iste particije.

Slika 5. Agregatna funkcija

```
1   SELECT ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika, Placa,
2       SUM(Placa)
3   FROM osoblje
4   GROUP by Struka
```

	ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	SUM(Placa)
1	2	Nikica	Polić	10000	36200
2	6	Alen	Nikić	8200	22200
3	1	Josip	Matić	6800	46650
4	10	Ana	Dokić	4500	4500

Izvor: autor

Slika 6. Prozorska agregatna funkcija

```
1   SELECT ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika, Placa,
2       SUM(Placa) OVER (PARTITION by Struka) as suma_place_po_struci
3   FROM osoblje
```

	ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	suma_place_po_struci
1	2	Nikica	Polić	10000	36200
2	5	Monika	Veselovac	7500	36200
3	7	Ivana	Horvat	5500	36200
4	11	Hrvoje	Nad	6500	36200
5	13	Tomislav	Agatić	6700	36200
6	6	Alen	Nikić	8200	22200
7	12	Ivan	Soldo	14000	22200
8	1	Josip	Matić	6800	46650
9	3	Lorena	Varga	9000	46650
10	4	Carlos	Petrović	7800	46650
11	8	Dolores	Perić	10200	46650
12	9	Hrvoje	Nikolić	12850	46650
13	10	Ana	Dokić	4500	4500

Izvor: autor

3.3.2. Redoslijedne funkcije

Sam naziv redoslijednih funkcija govori da se radi o nekom poretku, koji je vraćen za svaki individualni redak. U primjeni najčešće se rezultati odvajaju po određenim particijama. U redoslijedne funkcije spadaju:

- ROW_NUMBER()- vraća broj retka unutar particije, sve članove gledamo kao jednu particiju ukoliko nije korišteno grupiranje particija. Numeriranje kreće od broja 1.
- RANK()- numerira retke, istu vrijednost imaju članovi iste grupe. Kada funkcija pronađe dvije vrijednosti koje su identične unutar iste particije, dodijelit će im isti broj ranga. Slijedeći broj u poretku biti će uvećan za jedan od prethodnog ranga + članova grupe. Numeriranje kreće od broja 1. Obavezna je upotreba ORDER BY klauzule kojom definiramo članove neke grupe.
- DENSE_RANK()- Slično numeriranju RANK() funkcijom, osim što broj članova neke grupe ne utječe na numeriranje idućeg člana. Članovi iste grupe imaju istu vrijednost, dok idući član ima vrijednost uvećanu za 1. Obavezna upotreba ORDER BY klauzule.
- PERCENT_RANK()- Vrijednost varira od 0-1. Vrijednost 0 poprima prvi redak, dok vrijednost 1 poprima zadnji redak. Vrijednosti se iskazuju kao kumulativne sume podijeljene sa ukupnom sumom.
- NTILE(N)- Argumentom N[3] dijelimo retke u grupe. Tako da se ukupan broj redaka podijeli s argumentom N. Ukoliko dobijemo decimalni ostatak, prve grupe poprimaju više članova, kako bi ostale grupe ostale stabilne.

Na slici 7. prikazana je relacija redoslijedne funkcije ROW_NUMBER(). Korištena relacija u primjeru je 'osoblje'. U relaciji je prikazano sadržavanje istih imena. Atribut *Brojac_istih_imena* izveden je funkcijom ROW_NUMBER(), a on prikazuje broj istih imena prikazanih u relaciji. Rezultat relacije sa slike 7 moguće je prikazati i funkcijom RANK().

[3] Argument se nalazi unutar zagrada funkcije.

Slika 7. Redoslijedna funkcija ROW_NUMBER()

	ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	Broj_sličnih_imena
1	6	Alen	Nikić	8200	1
2	10	Ana	Dokić	4500	1
3	4	Carlos	Petrović	7800	1
4	8	Dolores	Perić	10200	1
5	9	Hrvoje	Nikolić	12850	1
6	11	Hrvoje	Nad	6500	2
7	12	Ivan	Soldo	14000	1
8	7	Ivana	Horvat	5500	1
9	1	Josip	Matić	6800	1
10	3	Lorena	Varga	9000	1
11	5	Monika	Veselovac	7500	1
12	2	Nikica	Polić	10000	1
13	13	Tomislav	Agatić	6700	1

Izvor: autor

3.3.3. Vrijednosne funkcije

Rezultat prikaza funkcije vrijednosti je skalar, vraćen je za svaku n-torku, te se mijenja izborom funkcije. U vrijednosne funkcije spadaju:

- LAG()- Vraća vrijednost prethodnog retka u trenutni red iste particije. Svaki prvi redak poprima vrijednost null.
- LEAD()- Vraća vrijednost sljedećeg retka u trenutni red iste particije. Svaki zadnji redak poprima vrijednost null.
- FIRST_VALUE()- vraća prvu navedenu vrijednost. Ukoliko ima podjela na particije, vraća prvu vrijednost svake particije zasebno.
- LAST_VALUE()- Slično kao FIRST_VALUE(), samo vraća zadnju vrijednost.
- Nth_VALUE()- Slično kao prošle dvije funkcije, samo je vraćanje N-te vrijednosti.

Na slici 8. prikazan je rezultat upita funkcije LAG(). Korištena relacija u primjeru je 'osoblje' s atributima: *ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika* i *Placa*. Izvedeni atribut: *Oscilacija_između_plaća* dobiven je funkcijom LAG(). Funkcija LAG() omogućuje pristup vrijednosti koja je pohranjena u retku iznad trenutnog retka. *Oscilacija_između_plaća* dobiva se ako se od trenutnog oduzima vrijednost prethodnog retka.

Slika 8. Vrijednosna funkcija LAG()

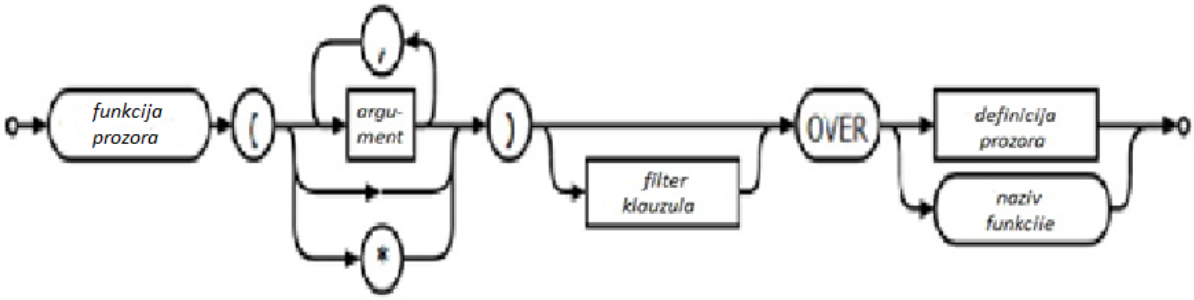
	ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	Oscilacija_između_plaća
1	1	Josip	Matić	6800	*NULL*
2	2	Nikica	Polić	10000	3200
3	3	Lorena	Varga	9000	-1000
4	4	Carlos	Petrović	7800	-1200
5	5	Monika	Veselovac	7500	-300
6	6	Alen	Nikić	8200	700
7	7	Ivana	Horvat	5500	-2700
8	8	Dolores	Perić	10200	4700
9	9	Hrvoje	Nikolić	12850	2650
10	10	Ana	Dokić	4500	-8350
11	11	Hrvoje	Nad	6500	2000
12	12	Ivan	Soldo	14000	7500
13	13	Tomislav	Agatić	6700	-7300

Izvor: autor

IV. SQL STRUKTURA I KLAUZULE PROZORSKE FUNKCIJA

Prozorska funkcija uvodi novitet u analitičkom prikazu podataka, kod kojeg je sintaksa čitljivija i jednostavnija od alternativnih pristupa. Sintaksa prozorske funkcije relativno se razlikuje od prethodnog SQL iskustva. Na slici 9. prikazana je struktura prozorske funkcije.

Slika 9. Struktura prozorske funkcije



izvor: autor, izvorna slika na eng. jeziku, preuzeta sa: SQLite, https://www.sqlite.org/windowfunctions.html

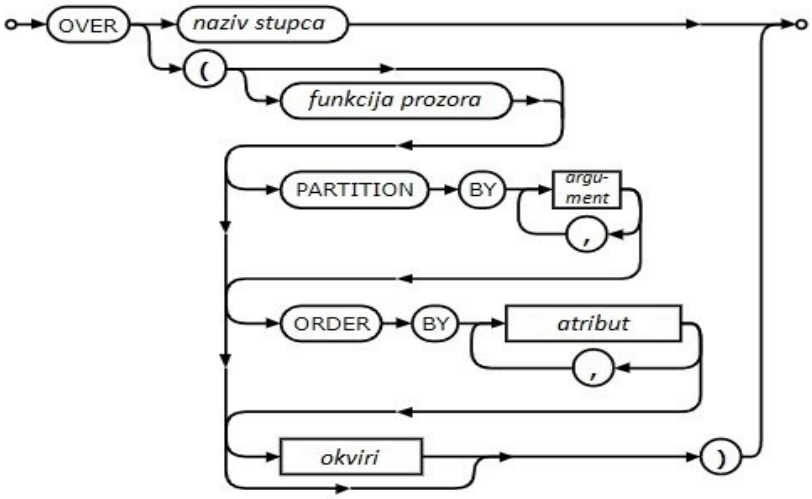
Prozorska funkcija započinje upit funkcijom, koja može biti jedna od navedenih sa slike 3. Određena funkcija vrši operacije koje su zadane argumentom. Argumentom se određuje atribut po kome se vrši funkcija prozora. Atribut se određuje standardnim SQL-om u koje ne ulazi naredba DISTINCT[4]. Prozorska funkcija može, ali i ne mora nužno sadržavati filter-klauzulu[5]. Nužno nakon zadane funkcije slijedi klauzula OVER, koja je jedinstveni pokazatelj da se u primjeru radi o funkciji prozora. Unutar zagrada klauzule OVER nalazi se posebne klauzule, koje definiraju izračun i slijed funkcije prozora (u poglavlju 4.1. klauzule su detaljno prikazane). (SQLite, 2022-08-09)[6]

4.1. Klauzula OVER()

Glavna karakteristika prozorske funkcije je klauzula OVER(), koja konkretno govori da se u primjeru koristi prozorska funkcija. Ukoliko izraz sadrži klauzulu OVER(), to je pokazatelj da se radi o prozorskoj funkciji.

[4] Naredba DISTINCT nije podržana u funkcijama prozora.
[5] Filter ili WHERE klauzula, sadrži uvjete.
[6] Zadnja promjena web stranice dogodila se: 2022-08-09.

Slika 10. Struktura klauzule OVER()



Izvor: prijevod autor , izvorna slika na eng. jeziku preuzeta sa: SQLite, https://www.sqlite.org/windowfunctions.html

Glavni dijelovi klauzule OVER() su: naziv funkcije i definicija funkcije. Naziv funkcije koristi se ukoliko se želi imenovati funkcija. Definicija funkcije sadržana je u zagradama klauzule, zagrade moraju biti postojane, iako ne moraju nužno sadržavati argumente. Funkcija prozora koja je navedena kao ulaz na slici 9, sadržana je i u klauzuli OVER(). Funkcija prozora u tom dijelu bit će aktivirana, ukoliko se u upitu koristi klauzula PARTITION by. Klauzula PARTITION by nije uvjetovana, ali ako je nema sve n-torke postaju jedna particija. Klauzula ORDER uvjetovana je u korištenju samo ako se koristi redoslijedna funkcija.

4.1.1. Klauzula PARTITION by

Klauzula PARTITION by je podklauzula u klauzuli OVER(). Definira kriterije po kojima se n-torke grupiraju u particije. Ako je klauzula u upitu izostavljena, skup n-torki promatramo kao jednu particiju (Joe Celko, 2013.).

Klauzula se koristi za dijeljenje ili rastavljanje redaka na particije, a definiranje particije provodi se po proizvoljnom argumentu. Funkcija se izračunava za svaku particiju, a prelaskom na slijedeću particiju ponovno se pokreće funkcija izračunavanja. Prikaz rezultata korištenja klauzule PARTITION by prikazano je na slici 6 .

4.1.2. Klauzula ORDER by

Klauzula ORDER by definira logičan slijed n-torki, podijeljenih na particije ili jednostavni slijed podataka bez grupiranja. Klauzulu je moguće koristiti i bez upotrebe prozorskih funkcija. Klauzula ORDER by obavezna je ako koristimo redoslijedne funkcije, npr. ROW_NUMBER().

Na slici 13. prikazano je korištenje klauzule ORDER by, te pripadajuće sintakse. U primjeru korištena je relacija 'osoblje' s atributima: Ime_zaposlenika, Prezime_zaposlenika i Struka. Izvedeni atributi *redoslijed* i *obrnuti_redoslijed* dobiveni su funkcijom ROW_NUMBER(). Atribut *redoslijed* i *obrnuti_redoslijed* prikazuju poziciju retka unutar particije. U primjeru nije korištena klauzula PARTITION, prema tome - cijelu relaciju gledamo kao jednu particiju.

Slika 13. Prikaz klauzule ORDER BY Sintaksa klauzule ORDER BY

```
SELECT Ime_zaposlenika, Prezime_zaposlenika, struka,
      ROW_NUMBER() OVER (ORDER by Ime_zaposlenika ASC) as Redoslijed,
      ROW_NUMBER() OVER (ORDER by Ime_zaposlenika DESC) as Obrnuti_redoslijed
FROM osoblje
```

	Ime_zaposlenika	Prezime_zaposlenika	Struka	Redoslijed	Obrnuti_redoslijed
1	Alen	Nikić	Elektroinžinjer	1	13
2	Ana	Dokić	Trgovac	2	12
3	Carlos	Petrović	Programer	3	11
4	Dolores	Perić	Programer	4	10
5	Hrvoje	Nikolić	Programer	5	8
6	Hrvoje	Nad	Ekonomist	6	9
7	Ivan	Soldo	Elektroinžinjer	7	7
8	Ivana	Horvat	Ekonomist	8	6
9	Josip	Matić	Programer	9	5
10	Lorena	Varga	Programer	10	4
11	Monika	Veselovac	Ekonomist	11	3
12	Nikica	Polić	Ekonomist	12	2
13	Tomislav	Agatić	Ekonomist	13	1

izvor: autor

4.1.3. Specifikacija okvira ROWS, RANGE i GROUPS

„Okvir ROWS ograničava retke unutar particije određivanjem fiksnog broja redaka koji prethode ili slijede trenutni red. S druge strane, okvir RANGE logički ograničava retke unutar particije određivanjem raspona vrijednosti s obzirom na vrijednost u trenutnom retku. Prethodni i slijedeći redovi definirani su na temelju redoslijeda određenog naredbom ORDER BY.“(Amorim Fabiano 2011)

Na slici 14. prikazano je korištenje okvira ROWS i RANGE. U primjerima je korištena relacija 'osoblje' s atributima: *ID_zaposlenika, Ime_zaposlenika, Struka* i *Plaća*. Izvedeni atributi: *Kumulativna_rast_plaće* i *Suma_place_po_grupama* prikazuje djelovanje okvira ROWS i RANGE.

Slika 14. Sintaksa i prikaz relacije okvira: ROWS i RANGE

```
1  SELECT Ime_zaposlenika, Prezime_zaposlenika, struka,
2      SUM(Placa) OVER (PARTITION by Struka ORDER by sektor
3          ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as Kumulativna_suma_plaće,
4      SUM(Placa) OVER (PARTITION by Struka ORDER by sektor
5          RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as Suma_plaće_po_grupama
6  FROM osoblje
```

	Ime_zaposlenika	Prezime_zaposlenika	Struka	Kumulativna_rast_plaće	Suma_place_po_grupama
1	Monika	Veselovac	Ekonomist	7500	26200
2	Ivana	Horvat	Ekonomist	13000	26200
3	Hrvoje	Nad	Ekonomist	19500	26200
4	Tomislav	Agatić	Ekonomist	26200	26200
5	Nikica	Polić	Ekonomist	36200	36200
6	Alen	Nikić	Elektroinžinjer	8200	22200
7	Ivan	Soldo	Elektroinžinjer	22200	22200
8	Josip	Matić	Programer	6800	37650
9	Carlos	Petrović	Programer	14600	37650
10	Dolores	Perić	Programer	24800	37650
11	Hrvoje	Nikolić	Programer	37650	37650
12	Lorena	Varga	Programer	46650	46650
13	Ana	Dokić	Trgovac	4500	4500

izvor: autor

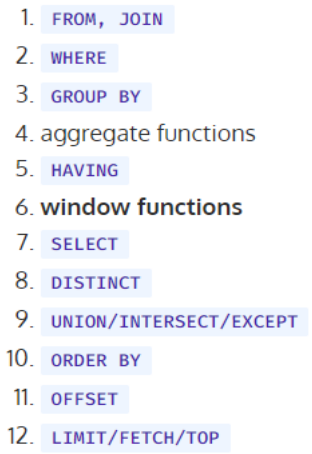
Iz navedenih primjera okvira ROWS i RANGE, izgledno je da je sintaksa identična, ali je prikaz relacije drukčiji. Relacija okvira ROWS prikazuje kumulativni rast, dok relacije okvira RANGE prikazuje sumiranu vrijednost n-torka.

Okvir GROUPS slične je sintakse kao navedeni okviri ROWS i RANGE, a rezultat okvira je relacija koja je identična kao kod okvira RANGE. Nije podržana od mnogih SUBP-a, a primjer okvira GROUPS naveden je u praktičnom dijelu rada.

4.2. Klauzule WHERE(), GROUP BY() i HAVING()

SQL je strogo hijerarhijski jezik u kojemu je bitan slijed operacija. Svaka funkcija u SQL-u obrađuju nekim logičkim slijedom, prema tome se prozorske funkcije ne mogu koristiti u klauzuli WHERE, zato jer nisu izračunate u trenutku odvijanja klauzule. Isto vrijedi i za ostale klauzule koje se učitavaju prije window funkcija. (Kozubek-Krycuń A., 2020.). Slijed prioriteta funkcija naveden je na slici 16.

Slika 16. Logički slijed operacija u SQL-u



Izvor: https://learnsql.com/blog/window-functions-not-allowed-in-where/

Ukoliko probamo iskoristiti prozorsku funkciju unutar WHERE, dobit ćemo odgovor: Greška: 4015. „Prozorske funkcije dopuštene su samo u SELECT listi i ORDER BY klauzuli.“ (eng. „Error Code: 4015. Window function is allowed only in SELECT list and ORDER BY clause.“)

4.2.1. Klauzula WITH

Klauzula WITH služi za uspostavljanje privremene relacije, koja služi za izvođenje glavnog upita. Prilikom izvršavanja klauzule WITH prvo se procjenjuje upit naveden unutar klauzule, a rezultat procijene sprema se u privremenoj relaciji. Potom, počinje izvršenje glavnog upita povezanog s klauzulom WITH, koji može koristiti rezultate privremene relacije. (geeksforgeeks.org, 2021)

Klauzula WITH korisna je pri radu sa složenijim SQL upitima, a ne s jednostavnima. Služi za rastavljanje složenih SQL upita na jednostavnije upite, što je jednostavnije za otklanjanje pogreške. U radu s funkcijom prozora omogućava korištenje klauzula koje su prioritetnije te tako zaobilazi prepreku korištenja. Prikaz prioriteta vidljiv je na slici 16.

Sintaksa i prikaz relacije sa WITH klauzulom prikazan je na slici 17. U primjeru je korištena relacija 'osoblje' s atributima: *ID_zaposlenika, Struka, Ime_zaposlenika* (dodijeljeno ime *prvi_predstavnik*), *Plaća* i *Radni_staz_u_godinama*. Atributi *suma_placa_struke* i *zaposlenik_iste_struke* izvedeni su funkcijama SUM() i RANK().

Slika 17. Klauzula WITH

```
WITH prvi_upit as
    (SELECT
        ID_zaposlenika, Struka, Ime_zaposlenika as Prvi_predstavnik,
        Placa, Radni_staz_u_godinama,
        SUM(Placa) OVER (PARTITION by Struka) as suma_place_po_grupama
    FROM osoblje
    WHERE Radni_staz_u_godinama>2
    GROUP by ID_zaposlenika)
        ,drugi_upit as
    (SELECT
        ID_zaposlenika, Struka, Prvi_predstavnik,
        Placa, Radni_staz_u_godinama, suma_place_po_grupama,
        DENSE_RANK() OVER(ORDER by Struka) as Zapolenik_iste_struke
        FROM prvi_upit)
    SELECT
        ID_zaposlenika, Struka, Prvi_predstavnik, Placa,
        Radni_staz_u_godinama, suma_place_po_grupama, Zapolenik_iste_struke
    from drugi_upit
```

	ID_zaposlenika	Struka	Prvi_predstavnik	Placa	Radni_staz_u_godinama	suma_place_po_grupama	Zapolenik_iste_struke
1	2	Ekonomist	Nikica	10000	10	10000	1
2	6	Elektroinžinjer	Alen	8200	3	22200	2
3	12	Elektroinžinjer	Ivan	14000	11	22200	2
4	3	Programer	Lorena	9000	5	32050	3
5	8	Programer	Dolores	10200	10	32050	3
6	9	Programer	Hrvoje	12850	7	32050	3

Izvor: autor

Pojašnjenje rezultata prikaza sa slike 17. U prvom upitu vrši se dohvaćanje podataka u privremenoj relaciji, pri tome korištena je funkcija SUM(), klauzula WHERE i GROUP by. Klauzulama WHERE i GROUP by vrše uvjete prema kojima će se izvoditi iduća dva upita. Upit dva koristi dohvaćene podatke iz privremene relacije i djeluje s funkcijom DENSE_RANK() na dohvaćenim podacima. Svi podaci se prikazuju u trećem upitu, koji prikazuju sve funkcije i atribute navedene u upitu dva.[7]

[7] Korištene funkcije su pojašnjene u odlomku 3.3., a sve korištene klauzule objašnjene su u odlomku 4.

V. IMPLEMENTACIJA U ČETIRI SUBP-a (razlike i sličnosti)

U ovom dijelu rada prikazana je implementacija baze podataka u četiri SUBP-a putem primjera koji će prikazivati sličnosti i razlike. Baza je implementirana u: SQL server, PostgreSQL, SQLite i MySQL. Izbor SUBP-a za implementiranje baze vršen je prema podržavanju funkcija prozora. Kratke karakteristike korištenih SUBP-a navedene su u poglavlju 5.1.

5.1. Karakteristike korištenih SUBP-a

Temeljem zadataka bit će prikazano korištenje funkcija prozora na četiri poznatija SUBP-a. Oni su besplatno dostupni za korištenje i preuzimanje. To su: PostgreSQL verzije 15.0, SQL server verzije 15.0.2095.3, SQLite verzije 3.35.5, MySQL verzije 10.4.24-MariaDB. SUBP-i su dostupni široj javnosti te su besplatni za korištenje u privatne svrhe.

SQL Server je SUBP za upravljanje relacijskim bazama podataka, razvio ga je Microsoft. Podržava ANSI SQL, koji je standardni jezik SQL. Međutim, SQL Server dolazi s vlastitom implementacijom SQL jezika, odnosno T-SQL[8].

PostgeSQL sustav je otvorenog koda za upravljanje relacijskim bazama podataka, a razvili su ga programeri. PostgreSQL nije pod kontrolom nikakve korporacije ili drugog privatnog subjekta, a izvorni kod dostupan je besplatno. Podržava četiri standarda proceduralna jezika, a to su: PL/pgSQL, PL/Tcl, PL/Perl i PL/Python. Najsličniji je standardnom SQL jeziku.

SQLite sustav je otvorenog koda za upravljanje relacijskim bazama podataka, a razvio ga je programer D. Richard Hipp. Za svoj pravilan rad potrebna je samo uključena biblioteka imena sqlite3, te ljuska koja pokreće biblioteku. Zbog jednostavnijeg pregleda korišten je grafički prikaz u aplikaciji 'DB Browser for SQLite'.

MySQL sustav je otvorenog koda za upravljanje relacijskim bazama podataka. Razvila ga je švedska tvrtka MySQL AB, 2010. godine, a kupila ga je kompanija Oracle Corporation.

[8] Skraćenica od Transaction-SQL.

VI. CJELOVITI PRIMJER

Kroz zadatke biti će prikazana odstupanja u sintaksi i odstupanja rezultata u prikazu upita. Navedeni zadaci biti će postavljeni u obliku tekstualnog zadatka, potom slijedi praktična primjena kroz četiri SUBP-a, te na kraju zaključak. Zadaci će biti navođeni prema težini.

Primijenjena baza je 'osoblje', a ona je jednaka u svim SUBP-ovima.

Slika 18. Relacija 'osoblje'

ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	Radni_staž_u_godinama	Struka	datum_rodenja
1	Josip	Matić	6800	0	Programer	1997-05-05
2	Nikica	Polić	10000	10	Ekonomist	1987-12-03
3	Lorena	Varga	9000	5	Programer	1992-12-30
4	Carlos	Petrović	7800	1	Programer	1999-01-04
5	Monika	Veselovac	7500	2	Ekonomist	1997-04-08
6	Alen	Nikić	8200	3	Elektroinžinjer	1989-12-24
7	Ivana	Horvat	5500	2	Ekonomist	1997-04-08
8	Dolores	Perić	10200	10	Programer	1989-09-20
9	Hrvoje	Nikolić	12850	7	Programer	1991-03-02
10	Ana	Dokić	4500	0	Trgovac	1999-12-11
11	Hrvoje	Nad	6500	1	Ekonomist	1999-08-25
12	Ivan	Soldo	14000	11	Elektroinžinjer	1990-05-12
13	Tomislav	Agatić	6700	1	Ekonomist	1999-08-08

Izvor: autor

6.1. Prvi zadatak

Zadatak 1: na relaciji 'osoblje' s atributima: Ime_zaposlenika i placa . Treba pokazati koliko iznosi postotni udio u sumi plaća svakog pojedinca individualno.

Slika 19. Zadatak 1: PostgreSQL

Query

```
1  Select ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika, Placa,
2      SUM(Placa) OVER (PARTITION by Ime_zaposlenika) / SUM(placa) OVER() *100 AS Postotak_udjela
3      FROM osoblje
```

Query History | Data Output | Messages | Notifications

	id_zaposlenika integer	ime_zaposlenika character varying (45)	prezime_zaposlenika character varying (45)	placa integer	postotak_udjela bigint
1	6	Alen	Nikić	8200	0
2	10	Ana	Dokić	4500	0
3	4	Carlos	Petrović	7800	0
4	8	Dolores	Perić	10200	0
5	11	Hrvoje	Nad	6500	0
6	9	Hrvoje	Nikolić	12850	0
7	12	Ivan	Soldo	14000	0
8	7	Ivana	Horvat	5500	0
9	1	Josip	Matić	6800	0
10	3	Lorena	Varga	9000	0
11	5	Monika	Veselovac	7500	0
12	2	Nikica	Polić	10000	0
13	13	Tomislav	Agatić	6700	0

Izvor: autor

Slika 20. Zadatak 1: SQL server

```
Select ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika, Placa,
    SUM(Placa) OVER (PARTITION by Ime_zaposlenika) / SUM(placa) OVER() *100 AS Postotak_udjela
    FROM osoblje
    order by Postotak_udjela desc
```

90 %

Results | Messages

	ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	Postotak_udjela
1	9	Hrvoje	Nikolić	12850	17.663100
2	11	Hrvoje	Nad	6500	17.663100
3	12	Ivan	Soldo	14000	12.779500
4	8	Dolores	Perić	10200	9.310800
5	2	Nikica	Polić	10000	9.128200
6	3	Lorena	Varga	9000	8.215400
7	6	Alen	Nikić	8200	7.485100
8	4	Carlos	Petrović	7800	7.120000
9	5	Monika	Veselovac	7500	6.846100
10	1	Josip	Matić	6800	6.207200
11	13	Tomislav	Agatić	6700	6.115900
12	7	Ivana	Horvat	5500	5.020500
13	10	Ana	Dokić	4500	4.107700

Izvor: autor

Slika 21. Zadatak 1: SQLite

```
1  Select ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika, Placa,
2      SUM(Placa) OVER (PARTITION by Ime_zaposlenika) / SUM(placa) OVER() *100 AS Postotak_udjela
3      FROM osoblje
```

	ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	Postotak_udjela
1	6	Alen	Nikić	8200	0
2	10	Ana	Dokić	4500	0
3	4	Carlos	Petrović	7800	0
4	8	Dolores	Perić	10200	0
5	9	Hrvoje	Nikolić	12850	0
6	11	Hrvoje	Nad	6500	0
7	12	Ivan	Soldo	14000	0
8	7	Ivana	Horvat	5500	0
9	1	Josip	Matić	6800	0
10	3	Lorena	Varga	9000	0
11	5	Monika	Veselovac	7500	0
12	2	Nikica	Polić	10000	0
13	13	Tomislav	Agatić	6700	0

Izvor: autor

Slika 22. Zadatak 1: MySQL

```
1 •  Select ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika, Placa,
2        SUM(Placa) OVER (PARTITION by Ime_zaposlenika) / SUM(placa) OVER() *100 AS Postotak_udjela
3        FROM osoblje
4        order by Postotak_udjela desc
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

ID_zaposlenika	Ime_zaposlenika	Prezime_zaposlenika	Placa	Postotak_udjela
11	Hrvoje	Nad	6500	17.6632
9	Hrvoje	Nikolić	12850	17.6632
12	Ivan	Soldo	14000	12.7796
8	Dolores	Perić	10200	9.3108
2	Nikica	Polić	10000	9.1283
3	Lorena	Varga	9000	8.2154
6	Alen	Nikić	8200	7.4852
4	Carlos	Petrović	7800	7.1200
5	Monika	Veselovac	7500	6.8462
1	Josip	Matić	6800	6.2072
13	Tomislav	Agatić	6700	6.1159
7	Ivana	Horvat	5500	5.0205
10	Ana	Dokić	4500	4.1077

Izvor: autor

Rezultati prvog zadatka prikazani su slikama 19.-22. Sintaksa je jednaka u svim SUBP-ovima, ali prikaz relacije pokazuje odstupanje u PostgreSQL-u i SQLite. Odstupanje u prikazu podataka dogodilo se zbog netočnog deklariranja varijable. Slika 22. prikazuje ispravljen prikaz u PostgeSQL-u.[9]

[9] Funkcija CAST() služi za promjenu varijabli, u primjeru: iz INT u decimal.

Slika 23. Zadatak 1: Ispravljeno PostgreSQL

```
Select ID_zaposlenika, Ime_zaposlenika, Prezime_zaposlenika, Placa,
    CAST(SUM(Placa) OVER (PARTITION by Ime_zaposlenika) as decimal (8,2)) / SUM(placa) OVER() *100 AS Postotak_udjela
    FROM osoblje
    order by Postotak_udjela DESC
```

Query History | Data Output | Messages | Notifications

	id_zaposlenika integer	ime_zaposlenika character varying (45)	prezime_zaposlenika character varying (45)	placa integer	postotak_udjela numeric
1	11	Hrvoje	Nad	6500	17.66316750342309447700
2	9	Hrvoje	Nikolić	12850	17.66316750342309447700
3	12	Ivan	Soldo	14000	12.77955271565495207700
4	8	Dolores	Perić	10200	9.31081697854860794200
5	2	Nikica	Polić	10000	9.12825193975353719800
6	3	Lorena	Varga	9000	8.21542674577818347800
7	6	Alen	Nikić	8200	7.48516659059790050200
8	4	Carlos	Petrović	7800	7.12003651300775901400
9	5	Monika	Veselovac	7500	6.84618895481515289800
10	1	Josip	Matić	6800	6.20721131903240529400
11	13	Tomislav	Agatić	6700	6.11592879963486992200
12	7	Ivana	Horvat	5500	5.02053856686444545900
13	10	Ana	Dokić	4500	4.10771337288909173900

Izvor: autor

6.2. Drugi zadatak

Zadatak 2: na relaciji 'osoblje' sa atributima ID_zaposlenika, Ime_zaposlenika, Struka i Placa. Potrebno je prikazati sumiranu plaću svih zaposlenika po strukama, ali se rezultat mora prikazati korištenjem okvira GROUPS.

Slika 24. Zadatak 2: PostgreSQL

```
1  SELECT ID_zaposlenika, Ime_zaposlenika, Struka, Placa,
2  SUM(Placa) OVER (PARTITION by Struka ORDER by Struka
3  GROUPS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as Sumirana_plaća_struke
4  from osoblje
```

Query History | Data Output | Messages | Notifications

	id_zaposlenika integer	ime_zaposlenika character varying (45)	struka character varying (45)	placa integer	sumirana_plaća_struke bigint
1	7	Ivana	Ekonomist	5500	36200
2	2	Nikica	Ekonomist	10000	36200
3	5	Monika	Ekonomist	7500	36200
4	13	Tomislav	Ekonomist	6700	36200
5	11	Hrvoje	Ekonomist	6500	36200
6	6	Alen	Elektroinžinjer	8200	22200
7	12	Ivan	Elektroinžinjer	14000	22200
8	1	Josip	Programer	6800	46650
9	4	Carlos	Programer	7800	46650
10	8	Dolores	Programer	10200	46650
11	9	Hrvoje	Programer	12850	46650
12	3	Lorena	Programer	9000	46650
13	10	Ana	Trgovac	4500	4500

Izvor: autor

Slika 25. Zadatak 2: SQL server

```
SELECT ID_zaposlenika, Ime_zaposlenika, Struka, Placa,
SUM(Placa) OVER (PARTITION by Struka ORDER by Struka
GROUPS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as Sumirana_plaća_struke
from osoblje
```

90 %

Messages

```
Msg 102, Level 15, State 1, Line 3
Incorrect syntax near 'GROUPS'.
```

Izvor: autor

Slika 26. Zadatak 2: SQLite

```
SELECT ID_zaposlenika, Ime_zaposlenika, Struka, Placa,
SUM(Placa) OVER (PARTITION by Struka ORDER by Struka
GROUPS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as Sumirana_plaća_struke
from osoblje
```

	ID_zaposlenika	Ime_zaposlenika	Struka	Placa	Sumirana_plaća_struke
1	2	Nikica	Ekonomist	10000	36200
2	5	Monika	Ekonomist	7500	36200
3	7	Ivana	Ekonomist	5500	36200
4	11	Hrvoje	Ekonomist	6500	36200
5	13	Tomislav	Ekonomist	6700	36200
6	6	Alen	Elektroinžinjer	8200	22200
7	12	Ivan	Elektroinžinjer	14000	22200
8	1	Josip	Programer	6800	46650
9	3	Lorena	Programer	9000	46650
10	4	Carlos	Programer	7800	46650
11	8	Dolores	Programer	10200	46650
12	9	Hrvoje	Programer	12850	46650
13	10	Ana	Trgovac	4500	4500

Izvor: autor

Slika 27. Zadatak 2: MySQL

```
SELECT ID_zaposlenika, Ime_zaposlenika, Struka, Placa,
SUM(Placa) OVER (PARTITION by Struka ORDER by Struka
GROUPS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as Sumirana_plaća_struke
from osoblje
```

```
Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'Groups BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as 'Groups'
from osoblje' at line 3
```

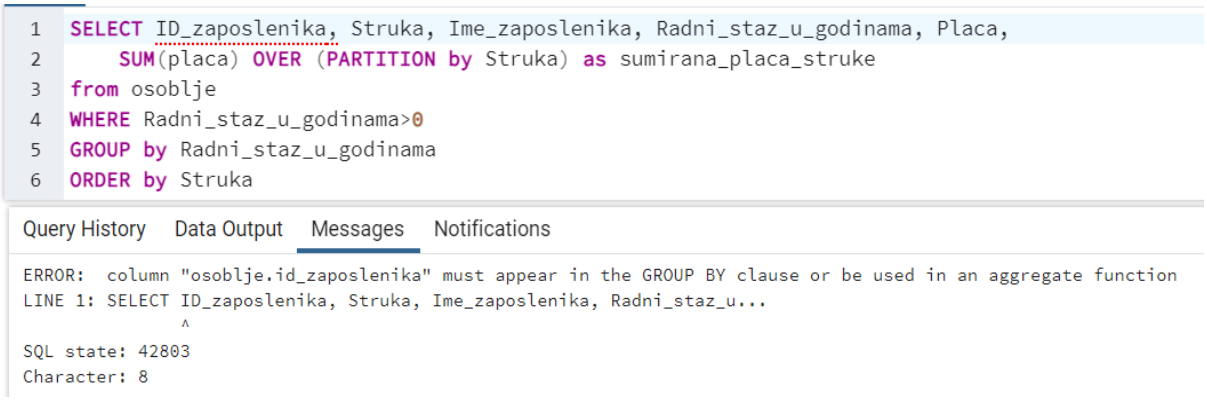
Izvor: autor

Rezultati drugog zadatka prikazani su slikama 24.-27. Sintaksa je jednaka u svim SUBP-ovima, ali prikaz upita pokazuje odstupanja u SQL server i MySQL-u. Razlog pokazivanja greške je što okvir Groups nije definiran u SQL server-u i MySQL-u.

6.3. Treći zadatak

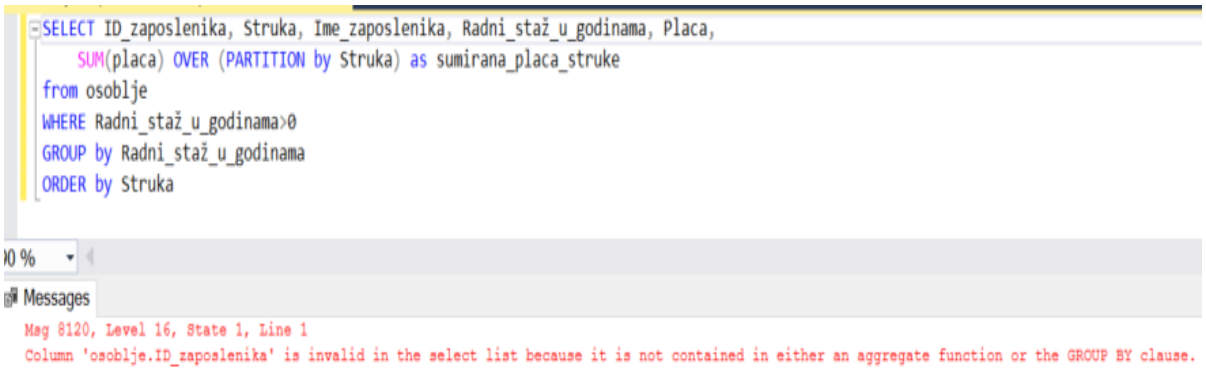
Zadatak 3: na relaciji 'osoblje' sa atributima ID_zaposlenika, Ime_zaposlenika, Struka, radni_staz_u_godinama i Plaća. Treba prikazati sumiranu plaću po struci za svakog zaposlenika individualno, uz uvjet da radni staž zaposlenika ne smije biti manji od jedne godine i treba prikazati samo jednog zaposlenika po godini staža.

Slika 28. Zadatak 3: PostgreSQL



Izvor: autor

Slika 29. Zadatak 3: SQL server



Izvor: autor

Slika 30. Zadatak 3: SQLite

```
SELECT
    ID_zaposlenika, Struka, Ime_zaposlenika, Radni_staz_u_godinama, Placa,
    sum(placa) OVER (PARTITION by Struka) as suma_place_po_struci
FROM osoblje
WHERE Radni_staz_u_godinama>0
GROUP by Radni_staz_u_godinama
order by Struka
```

	ID_zaposlenika	Struka	Ime_zaposlenika	Radni_staz_u_godinama	Placa	suma_place_po_struci
1	5	Ekonomist	Monika	2	7500	17500
2	2	Ekonomist	Nikica	10	10000	17500
3	17	Elektroinžinjer	Alen	3	8200	22200
4	12	Elektroinžinjer	Ivan	11	14000	22200
5	4	Programer	Carlos	1	7800	29650
6	3	Programer	Lorena	5	9000	29650
7	9	Programer	Hrvoje	7	12850	29650

Izvor: autor

Slika 31. Zadatak 3: MySQL

```
SELECT ID_zaposlenika, Struka, Ime_zaposlenika, Radni_staž_u_godinama, Placa,
    SUM(placa) OVER (PARTITION by Struka) as sumirana_placa_struke
from osoblje
WHERE Radni_staž_u_godinama>0
GROUP by Radni_staž_u_godinama
ORDER by Struka
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ID_zaposlenika	Struka	Ime_zaposlenika	Radni_staž_u_godinama	Placa	sumirana_placa_struke
2	Ekonomist	Nikica	10	10000	17500
5	Ekonomist	Monika	2	7500	17500
6	Elektroinžinjer	Alen	3	8200	22200
12	Elektroinžinjer	Ivan	11	14000	22200
9	Programer	Hrvoje	7	12850	29650
4	Programer	Carlos	1	7800	29650
3	Programer	Lorena	5	9000	29650

Izvor: autor

Rezultati trećeg zadatka prikazani su slikama 28.-31. Sintaksa je jednaka u svim SUBP-ovima, ali prikaz relacije pokazuje odstupanja kod PostgreSQL-a i SQL servera. Razlog prikazivanja greške je kršenje strukture koju implementiraju PostgreSQL i SQL server. Kršenje strukture se dogodilo zbog navođenja prozorske funkcije, a potom je iskorištena klauzula GROUP by. Klauzula GROUP by vraća skalar kao svoj izlazni rezultat, a prozorska funkcija vraća rezultat za svaki individualni redak.[10] Relacija je uredno prikazana u SQLite-u i MySQL-u.

6.4. Četvrti zadatak

Zadatak 4: na relaciji 'osoblje' potreban je atribut: Datum_rođenja. Treba pokazati godinu rođenja bez ostalih informacija, mjesec rođenja bez ostalih informacija, koliko je osoba rođeno u istoj godini i mjesecu, brojač rođenih s prethodna dva retka i trenutnim, brojač rođenih sa srednjom vrijednosti s prethodna dva retka i trenutnim i broj rođenih osoba do trenutnog retka.

Slika 32. Zadatak 4: PostgreSQL

Query

```
SELECT EXTRACT(YEAR from datum_rodenja) AS Godina_rodenja, EXTRACT(MONTH from datum_rodenja) AS Mjesec_rodenja,
COUNT(*) AS Isti_mjesec_i_godina,
SUM(COUNT(*)) OVER(ORDER BY EXTRACT(YEAR from datum_rodenja),
EXTRACT(MONTH from datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS Suma_rođenih_u_3_mj,
AVG(COUNT(*)) OVER(ORDER BY EXTRACT(YEAR from datum_rodenja),
EXTRACT(MONTH from datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS AVG_rođenih_u_3_mj,
SUM(COUNT(*)) OVER(ORDER BY EXTRACT(YEAR from datum_rodenja), EXTRACT(MONTH from datum_rodenja)) as broj_rođenih
FROM osoblje
GROUP BY EXTRACT(YEAR from datum_rodenja), EXTRACT(MONTH from datum_rodenja)
```

Query History | Data Output | Messages | Notifications

	godina_rodenja numeric	mjesec_rodenja numeric	isti_mjesec_i_godina bigint	suma_rođenih_u_3_mj numeric	avg_rođenih_u_3_mj numeric	broj_rođenih numeric
1	1987	12	1	1	1.00000000000000000000	1
2	1989	9	1	2	1.00000000000000000000	2
3	1989	12	1	3	1.00000000000000000000	3
4	1990	5	1	3	1.00000000000000000000	4
5	1991	3	1	3	1.00000000000000000000	5
6	1992	12	1	3	1.00000000000000000000	6
7	1997	4	2	4	1.3333333333333333	8
8	1997	5	1	4	1.3333333333333333	9
9	1999	1	1	4	1.3333333333333333	10
10	1999	8	2	4	1.3333333333333333	12
11	1999	12	1	4	1.3333333333333333	13

Izvor: autor

[10] Vidi poglavlje 3.3.1.

Slika 33. Zadatak 4: SQL server

```
SELECT
YEAR(datum_rodenja) AS Godina_rodenja,
MONTH(datum_rodenja) AS Mjesec_rodenja,
COUNT(*) AS Isti_mjesec_i_godina,
SUM(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja),
MONTH(datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS Suma_rođenih_u_3_mj,
AVG(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja),
MONTH(datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS AVG_rođenih_u_3_mj,
SUM(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja), MONTH(datum_rodenja)) as broj_rođenih
FROM osoblje
GROUP BY YEAR(datum_rodenja), MONTH(datum_rodenja)
```

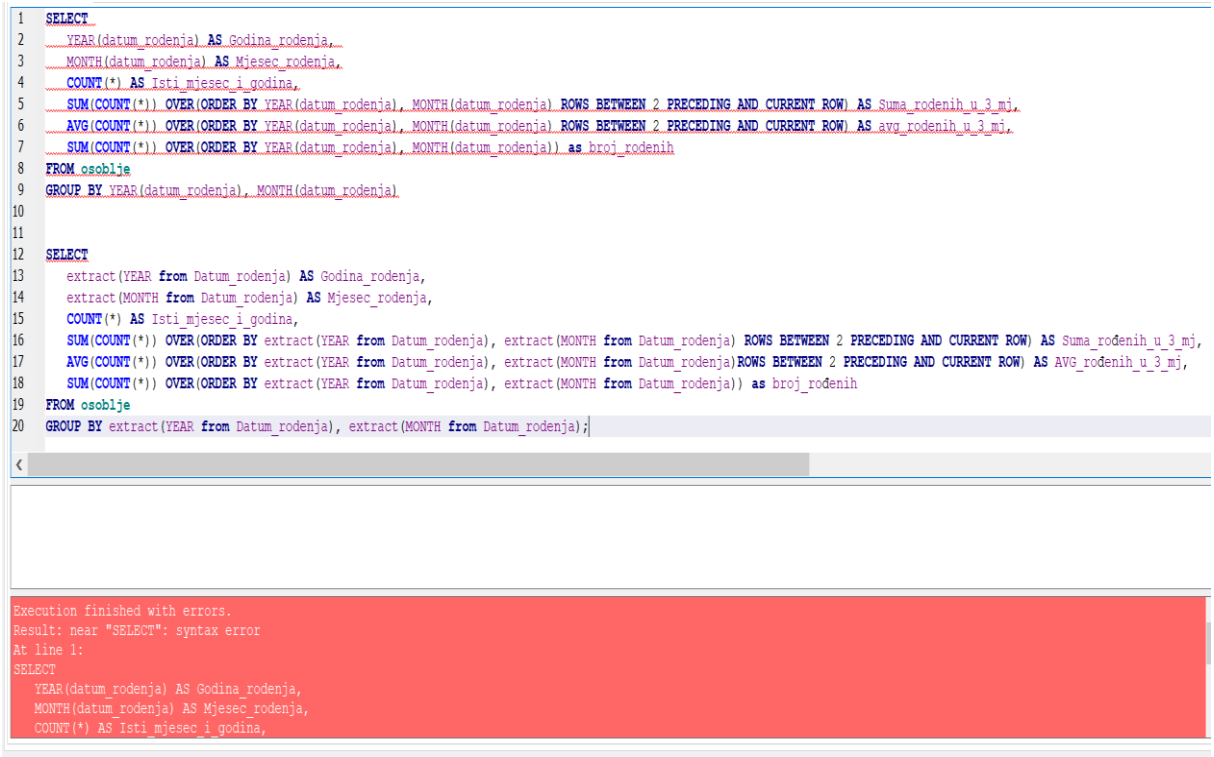
90 %

Results | Messages

	Godina_rodenja	Mjesec_rodenja	Isti_mjesec_i_godina	Suma_rođenih_u_3_mj	AVG_rođenih_u_3_mj	broj_rođenih
1	1987	12	1	1	1	1
2	1989	9	1	2	1	2
3	1989	12	1	3	1	3
4	1990	5	1	3	1	4
5	1991	3	1	3	1	5
6	1992	12	1	3	1	6
7	1997	4	2	4	1	8
8	1997	5	1	4	1	9
9	1999	1	1	4	1	10
10	1999	8	2	4	1	12
11	1999	12	1	4	1	13

Izvor: autor

Slika 34. Zadatak 4: SQLite



```
SELECT
   YEAR(datum_rodenja) AS Godina_rodenja,
   MONTH(datum_rodenja) AS Mjesec_rodenja,
   COUNT(*) AS Isti_mjesec_i_godina,
   SUM(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja), MONTH(datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS Suma_rodenih_u_3_mj,
   AVG(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja), MONTH(datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS avg_rodenih_u_3_mj,
   SUM(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja), MONTH(datum_rodenja)) as broj_rodenih
FROM osoblje
GROUP BY YEAR(datum_rodenja), MONTH(datum_rodenja)


SELECT
   extract(YEAR from Datum_rodenja) AS Godina_rodenja,
   extract(MONTH from Datum_rodenja) AS Mjesec_rodenja,
   COUNT(*) AS Isti_mjesec_i_godina,
   SUM(COUNT(*)) OVER(ORDER BY extract(YEAR from Datum_rodenja), extract(MONTH from Datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS Suma_rođenih_u_3_mj,
   AVG(COUNT(*)) OVER(ORDER BY extract(YEAR from Datum_rodenja), extract(MONTH from Datum_rodenja)ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS AVG_rođenih_u_3_mj,
   SUM(COUNT(*)) OVER(ORDER BY extract(YEAR from Datum_rodenja), extract(MONTH from Datum_rodenja)) as broj_rođenih
FROM osoblje
GROUP BY extract(YEAR from Datum_rodenja), extract(MONTH from Datum_rodenja);
```

```
Execution finished with errors.
Result: near "SELECT": syntax error
At line 1:
SELECT
   YEAR(datum_rodenja) AS Godina_rodenja,
   MONTH(datum_rodenja) AS Mjesec_rodenja,
   COUNT(*) AS Isti_mjesec_i_godina,
```

Izvor: autor

Slika 35. Zadatak 4: MySQL

```
SELECT
   YEAR(datum_rodenja) AS Godina_rodenja,
   MONTH(datum_rodenja) AS Mjesec_rodenja,
   COUNT(*) AS Isti_mjesec_i_godina,
   SUM(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja), MONTH(datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS Suma_rođenih_u_3_mj,
   AVG(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja), MONTH(datum_rodenja) ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS AVG_rođenih_u_3_mj,
   SUM(COUNT(*)) OVER(ORDER BY YEAR(datum_rodenja), MONTH(datum_rodenja)) as broj_rođenih
   FROM osoblje
   GROUP BY YEAR(datum_rodenja), MONTH(datum_rodenja)
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Godina_rodenja	Mjesec_rodenja	Isti_mjesec_i_godina	Suma_rođenih_u_3_mj	AVG_rođenil	broj_rođenih
1987	12	1	1	1.0000	1
1989	9	1	2	1.0000	2
1989	12	1	3	1.0000	3
1990	5	1	3	1.0000	4
1991	3	1	3	1.0000	5
1992	12	1	3	1.0000	6
1997	4	2	4	1.3333	8
1997	5	1	4	1.3333	9
1999	1	1	4	1.3333	10
1999	8	2	4	1.3333	12
1999	12	1	4	1.3333	13

Izvor: autor

Rezultati četvrtog zadatka prikazani su slikama 32.-35. Sintaksa i rezultat upita jednaki su u SQL server i MySQL-u, dok se sintaksa mijenja u PostgreSQL-u, a prikaz relacije je ostao isti. Sintaksa u PostgreSQL je modificirana prema njegovoj implementaciji SQL jezika. U SQLite dolazi do greške, a razlog prikazivanja greške je što nema podršku funkcije EXTRACT, te ne može izdvojiti podatak iz atributa.

Tablica 1. Prozorska funkcija-pregled sposobnosti izvršavanja

	PostgreSQL verzija	SQL server	SQLite	MySQL
Klauzula OVER()	*DA*	DA	DA	DA
Klauzula PARTITION by	DA	DA	DA	DA
Klauzula ORDER by	DA	DA	DA	DA
Okvir ROWS	DA	DA	DA	DA
Okvir RANGE	DA	DA	DA	DA
Okvir GROUPS	DA	NE	DA	NE
Klauzula WITH	DA	DA	DA	DA
Klauzula GROUP by	Podržana unutar klauzule WITH	Podržana unutar klauzule WITH	DA	DA
Naredba DISTINCT	NE	NE	NE	NE
Naredba EXTRACT	DA	NEMA, drugačija definicija	NE	NEMA, drugačija definicija

Uspješno riješeni zadatci

	PostgreSQL	SQL server	SQLite	MySQL
Zadatak 1.	DA, stroži zahtjev sintakse tipa podataka	DA	DA, stroži zahtjev sintakse tipa podataka	DA
Zadatak 2.	DA	NE	DA	NE
Zadatak 3.	NE	NE	DA	DA
Zadatak 4.	DA, stroži zahtjev sintakse tipa podataka	DA	NE	DA
Ukupno	3	2	3	3

Izvor: završni rad

VII. ZAKLJUČNA RAZMATRANJA

Prema podacima koji su predstavljeni u radu, zaključujem da su svrha i cilj ovog rada ispunjeni. Svrha rada bila je analiza potencijala prozorske funkcije u SQL-u koje su relativni novitet u analitičkom prikazu podataka. Cilj ovog rada bio je prikazati utjecaj alata za manipulaciju baza podataka na prozorskim funkcijama u SQL-u.

Opširnim prikazom ovoga rada možemo zaključiti da su funkcije prozora u SQL-u vrlo djelotvorne u analitičkom prikazivanju podataka, ali i dalje su prilično neistražena tema. Alternativni pristupi za izvođenje prozorske funkcije su vrlo zahtjevni, ali ju je ipak moguće izvesti. Prozorske funkcije dobivaju veliku prednost kod analitičkog prikazivanja podataka i same jednostavnosti u izradi takvih upita. Implementirane su na široki spektar funkcija, koje mogu raditi unutar funkcije prozora ili samostalno. Sintaksa prozorske funkcije prepoznatljiva je po klauzuli OVER(), koja je glavni pokazatelj da se u primjeru koristi funkcija prozora. Iako odstupa od standardne sintakse, vrlo je pamtljiva. Dodatne klauzule koje se koriste uz klauzulu OVER() su: filter klauzula, klauzula PARTITION by i klauzula ORDER by. Moguće je korištenje okvira: ROWS, RANGE i GROUP.

Putem više primjera prikazano je odstupanje primjene prozorske funkcije u alatima: SQL server, PostgreSQL, SQLite i MySQL. Utvrđeno je da PostgreSQL najbolje implementira prozorske funkcije, te zadržava svoj integritet prema realizaciji vlastitog cilja da održi strukturu prozorske funkcije i samog SQL jezika prema SQL standardu. SQL server dolazi s vlastitom implementacijom SQL jezika, a implementacija se naziva T-SQL. Pokazuje odstupanja prema podržavanju funkcija koje su sastavni dio prozorske funkcije, u prikazanom slučaju bio je okvir GROUPS za koji nije sadržavao implementiranu podršku. SQL server kao i PostgreSQL održava strukturu prozorske funkcije i SQL jezika. SQLite i MySQL pokazuju značajnija odstupanja u strukturi prozorske funkcije, ali to doprinosi fleksibilnijoj primjeni.

LITERATURA

Knjige:

1. Kramberger T., Duk S., Kovačević S. (2018) Baze podataka: udžbenik za kolegij 'Baze podataka' i 'Napredne baze podataka'. Zagreb: tehničko veleučilište u Zagrebu, Vrbnik 8, Zagreb

Internetske stranice:

1. Celko J. (2013) Window function in SQL. Dostupno na: https://www.red-gate.com/simple-talk/databases/sql-server/t-sql-programming-sql-server/window-functions-in-sql/ (Datum pristupa 15.10.2022.)
2. Fabiano A. (2011) Window Functions in SQL Server: Part 2- The Frame Dostupno na: https://www.red-gate.com/simple-talk/databases/sql-server/learn/window-functions-in-sql-server-part-2-the-frame/ (Datum pristupa 15.10.2022.)
3. geeksforgeeks.org (2021). SQL WITH clause. Dostupno na: https://www.geeksforgeeks.org/sql-with-clause/ (Datum pristupa 22.10.2022.)
4. Koidan K. (2020) SQL Window functions vs. SQL aggregate functions: Similarities and differences. Dostupno na: https://learnsql.com/blog/window-functions-vs-aggregate-functions/ (Datum pristupa 22.10.2022.)
5. Kozubek-Krycuń A. (2020) Why Window Function Are Not Allowed in WHERE Clause. Dostupno na: https://learnsql.com/blog/window-functions-not-allowed-in-where/ (Datum pristupa 22.10.2022.)
6. SQLite. (2022) Window function in SQL. Dostupno na: https://www.sqlite.org/windowfunctions.html (Datum pristup: 22.10.2022.)
7. William L. Hosch (-) Edgar Frank Codd, American computer scientist and mathematician. Dostupno na: https://www.britannica.com/biography/Edgar-Frank-Codd (Datum pristupa 22.10.2022.)

Nastavna predavanja:

1. mr. sc. Vuk D. (2020./2021.) Nastavna predavanja kolegija 'Baze podataka'.
2. ZPR-FER-Zagreb (2018./2019.) Napredni modeli i baze podataka 2018./2019, str. 20-20. Preuzeto sa: https://www.fer.unizg.hr/_download/repository/2._Napredni_SQL[5].pdf (Datum pristupa 17.10.2022.)

POPIS TABLICA

POPIS ILUSTRACIJA

Veleučilište u Virovitici

OBRAZAC 5

IZJAVA O AUTORSTVU

Ja, JOSIP MATIĆ

izjavljujem da sam autor/ica završnog/diplomskog rada pod nazivom

PROZORSKE FUNKCIJE U SQL-U

Svojim vlastoručnim potpisom jamčim sljedeće:

- da je predani završni/diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija,
- da su radovi i mišljenja drugih autora/ica, koje sam u svom radu koristio/la, jasno navedeni i označeni u tekstu te u popisu literature,
- da sam u radu poštivao/la pravila znanstvenog i akademskog rada.

Potpis studenta/ice

J. Matić

Veleučilište u Virovitici

OBRAZAC 6

ODOBRENJE ZA POHRANU I OBJAVU ZAVRŠNOG/DIPLOMSKOG RADA

Ja Josip Matić

dajem odobrenje za objavljivanje mog autorskog završnog/diplomskog rada u javno dostupnom digitalnom repozitoriju Veleučilišta u Virovitici te u javnoj internetskoj bazi završnih radova Nacionalne i sveučilišne knjižnice bez vremenskog ograničenja i novčane nadoknade, a u skladu s odredbama članka 83. stavka 11. Zakona o znanstvenoj djelatnosti i visokom obrazovanju (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15, 131/17).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog završnog/diplomskog rada. Ovom izjavom, kao autor navedenog rada dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti
b) studentima i djelatnicima ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

Potpis studenta/ice

Jmatić

U Virovitici, 09.11.2023.

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem završnom/diplomskom radu, podnosi se pisani obrazloženi zahtjev.*