Optimizacija upita korištenjem pogleda i materijaliziranih pogleda

Skočibušić, Danijela

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

OPTIMIZACIJA UPITA KORIŠTENJEM POGLEDA I MATERIJALIZIRANIH POGLEDA

Danijela Skočibušić

Split, rujan 2024.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

OPTIMIZACIJA UPITA KORIŠTENJEM POGLEDA I MATERIJALIZIRANIH POGLEDA

Danijela Skočibušić

SAŽETAK

U ovom završnom radu, cilj je bio napraviti usporedbu korištenja pogleda i materijaliziranih pogleda te se osvrnuti na osnovne karakteristike korištenja pohranjenih procedura. U tu svrhu osmišljena je i realizirana relacijska baza podataka nazvana *Dostava hrane* koja sadrži sve informacije koje se vežu uz stvaranje jedne online narudžbe iz nekog restorana i koja bi se mogla učinkovito koristiti za aplikacije koje omogućavaju online narudžbe. Baza je kreirana uz pomoć Oracle alata. Za kreiranje logičkog i relacijskog modela korišten je Oracle SQL Data Modeler, dok je za izradu korisnika, pogleda, materijaliziranih pogleda i procedura korišten Oracle SQL Developer. Rad obuhvaća teorijski dio od same pojave baza podataka, razvoja relacijskih baza, osnove Oracle alata, postupak izrade baze podataka, pogleda, materijaliziranih pogleda i pohranjenih procedura.

Ključne riječi: relacijski model, pogled, materijalizirani pogled, pohranjena procedura

Rad sadrži: 35 stranica, 17 grafičkih prikaza, 14 literaturnih navoda.
Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Monika Mladenović**, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Dr. sc. Monika Mladenović,** *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Dr. sc. Divna Krpan, *docent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*
Antonela Prnjak, mag. educ. inf. *asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: rujan, 2024.

Basic documentation card

Thesis

University of Splitu

Faculty of science

Department of computer science

Ruđera Boškovića 33, 21000 Split, Hrvatska

QUERY OPTIMIZATION USING VIEWS AND MATERIALIZED VIEWS

Danijela Skočibušić

ABSTRACT

In this final thesis, the goal was to make a comparison of the use of views and materialized views and to look at the basic characteristics of using stored procedures. For this purpose, a relational database called Food Delivery was designed and implemented, which contains all the information related to the creation of an online order from a restaurant and which could be effectively used for applications that enable online orders. The database was created with the help of Oracle tools. Oracle SQL Data Modeler was used to create the logical and relational model, while Oracle SQL Developer was used to create user, views, materialized views and procedures. The work includes the theoretical part from the very emergence of the database, the development of relational databases, the basics of Oracle tools, the process of creating a database, views, materialized views and stored procedures.

Key words: relational model, view, materialized view, stored procedure

Thesis consists of: 35 pages, 17 figures and 14 references.

Original language: Croatian

Supervisor: **Monika Mladenović**, **Ph.D**. Assistant Professor of Faculty of Science, University of Split

Reviewers: **Monika Mladenović, Ph.D.** *Assistant Professor of Faculty of Science, University of Split*

Divna Krpan, Ph.D. *Assistant Professor of Faculty of Science, University of Split*

Antonela Prnjak, mag.educ.inf. *Instructor of Faculty of Science, University of Split*

Thesis accepted: September 2024.

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom „Optimizacija upita korištenjem pogleda i materijaliziranih pogleda“ izradila samostalno pod voditeljstvom Doc. Dr. sc. Monike Mladenović. U radu sam primijenila metodologiju znanstvenoistraživačkog rada i koristila literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući navela u završnom radu na uobičajen, standardan način citirala sam i povezala s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Studentica

Danijela Skočibušić

Sadržaj

Uvod

U današnjem svijetu, u kojem vlada sve veća digitalizacija, baze podataka su koncept koji svakodnevno koristimo, iako toga ponekad nismo ni svjesni. Upravo su baze podataka jedan od glavnih koncepata u gotovo svim poslovanjima i u tom području je zapravo njihova najvažnija primjena. One omogućavaju pohranjivanje velikih količina informacija na organiziran način što nam onda olakšava upravljanje njima brzo i jednostavno te je stoga važno odabrati bazu koja će najbolje odgovarati našim zahtjevima. Bez baza podataka podaci bi bili pohranjeni u raznim formatima i na različitim mjestima što bi znatno otežalo mogućnosti njihovom pristupanju.

Razvoj prvih sustava upravljanja bazama podataka započeo je u 1960-im, zahvaljujući Charlesu Bachmanu čiji se rad temeljio na stvaranju djelotvornije pohrane podataka u uređajima, koja se do tada temeljila na bušenim karticama i magnetskoj vrpci. Tijekom vremena razvijale su se različite tehnike za formiranje modela podataka, koje osim strukturiranja podataka uključuju i skupove operacija koje je moguće izvoditi na podacima.

U ovom radu kratko ćemo se osvrnuti na analizu povijesti razvoja baza podataka, a pobliže ćemo se upoznati s relacijskim bazama, posebice s izradom baze podataka te pogleda, materijaliziranih pogleda i pohranjenih procedura na Oracle alatu. Fokus će biti na izradi pogleda (eng. Views) i materijaliziranih pogleda (eng. Materialized Views) te optimizaciji upita njihovim korištenjem.

1. Razvojna tehnologija

Prva poznata uporaba termina *baza podataka* zabilježena je u lipnju 1963. U to vrijeme su se pojavila dva važna modela podataka, a to su mrežni model i hijerarhijski model, dok je odnosni model predložen 1970. godine. Od 1980-ih godina pa sve do danas prevladava relacijski model. Osnovna zamisao tehnologije baza podataka je u tome da pojedine aplikacije ne stvaraju vlastite datoteke na disku, nego da sve aplikacije koriste zajednički, međusobno povezani skup podataka. Taj skup podataka se naziva *baza podataka*, a specijalizirani softver koji je zadužen za komunikaciju između aplikacija i podataka naziva se *sustav za upravljanje bazom podataka* (eng. Data Base Management System – DBMS). Prvi sustavi upravljanja bazom podataka razvijeni su u 1960-im. Upravo sustav za upravljanje bazom podataka obavlja sve operacije u ime korisnika te se brine za sigurnost podataka. Podaci u bazi su organizirani tako da se mogu lako i brzo pronaći [1]. Pretraživanje podataka je temeljeno upravo na upitima korisnika, a rezultat, odnosno pronađeni skup podataka mora odgovarati zadanim svojstvima. Neke od karakteristika baza podataka su: spremanje velike količine podataka, dopuštanje velikog broja dnevnih promjena te omogućavanje različitih razina izvještaja kao što su detaljni podaci, sumarni podaci i izračunati podaci. S obzirom na način spremanja, održavanja, čitanja i pretraživanja podataka te načinu na koji su oni organizirani, baze dijelimo na: datotečne, hijerarhijske, mrežne, relacijske i objektne [2]. U nastavku ovog rada posebnu pažnju ćemo posvetiti relacijskim bazama, njihovom razvoju te principu oblikovanja modela u Oracle Database, koji je jedan od najpoznatijih i najkorištenijih alata za relacijske baze podataka.

1.1. Relacijske baze podataka

Britansko-američki matematičar Edgar Frank Codd tijekom 1960-ih i 1970-ih razvijao je svoje teorije slaganja podataka, a 1970., dok je radio za IBM objavio je dokument pod nazivom „A Relational Model of Data for Large Shared Data Banks“ u kojem je predložio novi model baza podataka kojeg je nazvao relacijski model (eng. relational model). Te godine se prvi put i javlja pojam relacijskih baza podataka. Njegov rad je imao značajan utjecaj na dotadašnje shvaćanje sustava baza podataka jer je promijenio način na koji se podaci pohranjuju, pretražuju i upravljaju u računalnim sustavima. Njegova zamisao je bila organizacija podataka u tablice koje bi bile povezane relacijama i sadržavale bi redove

(zapise) i stupce (atribute). Redovi bi sadržavali podatke koji predstavljaju jednu stavku, dok bi stupci predstavljali označeni element torke (torka obično predstavlja objekt i informacije o njemu). Prva tvrtka koja je implementirala takav relacijski model podataka je Oracle koji je danas sinonim za baze podataka. Codd je dugo vremena pokušavao usmjeravati proizvođače sustava za upravljanje bazom podataka kako da, u potpunosti ispravno, implementiraju relacijski model podataka te je kao rezultat tih pokušaja 1985. godine definirao 13 pravila (12 pravila i jedno nulto) koja navode što sve određuje relacijsku bazu podataka, a sustav mora podržavati minimalno 6 pravila kako bi se mogao definirati kao relacijski. Pravila su navedena u nastavku:

0. Nulto pravilo: Bilo koji sustav za upravljanje bazama podataka koji se smatra relacijskim, mora upravljati bazom na potpuno relacijski način.

1. Pravilo informacije: Podaci se predstavljaju na jedinstven način: kao vrijednosti u tablici; sve informacije u bazi podataka se nalaze u nekoj relaciji.

2. Pravilo pristupa: Svaki podatak mora biti logički dostupan preko: kombinacije imena relacije, vrijednosti primarnog ključa i imena atributa.

3. Tretiranje nepoznatih vrijednosti: Sustav mora dopuštati da polje može ostati prazno ukoliko je to potrebno. Nepoznata vrijednost se tretira kao NULL i neovisna je o tipu. NULL vrijednost nije isto što i nula (0). Kombinirana s bilo čime ponovno daje NULL vrijednost.

4. Dinamički online katalog: Rječnik baze podataka mora biti spremljen kao i drugi podaci u bazi kako bi mu korisnici mogli pristupiti koristeći isti jezik kao i za upite.

5. Sveobuhvatni jezik za upravljanje podacima: Postojanje jezika koji sadrži sve potrebno za komunikaciju s bazom podataka i koji podržava relacijske operatore (definiranje podataka, definiranje pogleda, upravljanje podacima)

6. Pravilo ažuriranja pogleda: Svi pogledi se moraju moći ažurirati i implementirati u model.

7. Pravila ažuriranja skupova: Sustav mora podržavati umetanje, ažuriranje i brisanje za sve skupove podataka (insert, update, delete operatori).

8. Fizička neovisnost podataka: Aplikacije koje pristupaju bazi podataka moraju biti potpuno neovisne o strukturi spremanja podataka.

9. Logička neovisnost podataka: Mijenjanje odnosa među tablicama ne smije utjecati na funkciju aplikacije. Teže je postići logičku nego fizičku neovisnost.

10. Nezavisnost integriteta podataka: Pravila integriteta moraju biti definirana u jeziku baze podataka i ne smiju ovisiti o aplikacijama. Njihove izmjene moraju biti omogućene u svakom trenutku tako da ne utječu na postojeće aplikacije.

11. Distribuirana nezavisnost: Ako se uvede distribuirana verzija sustava za upravljanje bazom podataka aplikacija mora nastaviti raditi.

12. Pravilo zaštite podataka: Integritet podataka i relacijska sigurnost ne smiju biti narušeni [3].

Relacijska baza podataka organizira podatke na temelju relacijskog podatkovnog modela, a sustav za upravljanje relacijskim bazama podataka (eng. Relational Database Management System, u nastavku RDBMS) je referenca na osnovni softver baze podataka koji omogućuje njegovo korisničko održavanje. Kao što je već spomenuto u prethodnom poglavlju, najpoznatija i najkorištenija implementacija sustava za upravljanje relacijskim bazama podataka je Oracle Database, koji ima široku primjenu u velikim organizacijama, a o njemu će detaljnije pisati u nastavku ovoga rada. Neke od ostalih implementacija ovakvog sustava su: MySQL, Microsoft SQL Server, PostgreSQL, SQLite i drugi. Prednost relacijskih baza podataka je mogućnost stvaranja smislenih informacija spajanjem tablica i samim time razumijevanja odnosa između podataka ili načina na koji su tablice povezane. Također, prednost relacijskih baza je njihovo korištenje indeksa, čime je omogućen brzi pronalazak informacija. Kao što je ranije navedeno u uvodnom dijelu ovoga rada, osnovni sustav koji upravlja radom baza podataka naziva se Sustav za upravljanje bazom podataka (eng. Database Managment System, u nastavku DBMS). Radi se o softveru za spremanje i dohvaćanje podataka korisnika uz pridržavanje određenih sigurnosnih mjera. DBMS omogućuje korisnicima kreiranje baze podataka prema vlastitim potrebama te omogućuje istovremeni pristup i korištenje podataka od strane više korisnika. Razlikujemo četiri vrste DBMS-a:

- Hijerarhijska baza podataka
- Mrežna baza podataka
- Relacijska baza podataka
- Objektno orijentirana baza podataka [4]

Osnovna razlika između RDBMS i DBMS je u načinu pohrane podataka. DBMS pohranjuje podatke kao datoteke, dok RDBMS podatke pohranjuje u tablice [5].

1.2. Modeliranje podataka

Modeliranje podataka predstavlja fazu stvaranja podatkovnog modela za podatke koji se spremaju u bazu podataka. Struktura takvog modela omogućava jednostavno definiranje relacijskih tablica uz korištenje primarnih i stranih ključeva, a glavni rezultat modeliranja podataka bi trebala biti shema baze podataka koja pomaže vizualno predstaviti podatke i njihovu povezanost. Model podataka omogućuje stvaranje baze podataka na tri razine, a to su redom: konceptualna, logička i fizička razina. Svaki od tih modela ima vlastitu ulogu u procesu modeliranja baze podataka [4].

Konceptualno oblikovanje: predstavlja prvu fazu u izradi baze podataka i najčešće se reprezentira kao skica na papiru, a rezultat je shema cijele baze podataka koja je sastavljena od atributa, entiteta i veza, a nazivamo je konceptualnim modelom. Kod konceptualnog oblikovanja naglasak je na intuitivnom povezivanju informacija te na jednostavnosti modela kako bi ga korisnik mogao shvatiti na pravi način. Konceptualni model je jedna vrsta nepotpune verzije baze, no ipak je vrlo važno napraviti dobar model jer poslije može doći do promjena koje je lako implementirati na modelu, no problemi se mogu javiti tijekom implementacije u aplikaciji [1].

Entitet (eng. Entity): predstavlja stvar, biće ili pojavu o kojoj želim spremati podatke. Svaki entitet sadrži svoje atribute koji ga opisuju, a za naziv entiteta se uglavnom koristi imenica u jednini [6]. Npr. entitet može biti: KUĆA (skup svih kuća), STUDENT (skup svih studenata), AUTOMOBIL (skup svih automobila)…

Atribut (eng. Attribute): predstavlja određeno svojstvo entiteta. Svaka vrsta entiteta opisana je skupom svojih atributa (obilježja) [6]. Tako atributi za entitet STUDENT mogu biti: matični broj, ime, prezime, godina studija itd.

Veza: odnos između entiteta opisan je vezom ili relacijom. Veza se definira na razini tipova entiteta. Razlikujemo više vrsta veza, a najčešće korištene su binarne veze (eng. Binary Relationship) koje su najjednostavnije i imaju više funkcionalnosti, a funkcionalnost je svojstvo koje nam govori može li se veza interpretirati kao preslikavanje iz skupa primjeraka entiteta jednog tipa u skup primjeraka entiteta drugog tipa. Postoje tri vrste funkcionalnosti koje su navedene u sljedećoj tablici [6].

OZNAKA	NAZIV	OPIS
1:1	Jedan naprema jedan	Svaki element prvog skupa može biti povezan najviše s jednim elementom drugog skupa [6].
1:N	Jedan naprema više	Jedan element prvog skupa može biti povezan s više elemenata drugog skupa. Istovremeno jedan element drugog skupa može biti povezan s najviše jednim elementom prvog skupa [6].
M:N	Više naprema više	Jedan element iz jednog skupa može biti povezan s više elemenata drugog skupa. Također, jedan element drugog skupa može biti povezan s više elemenata prvog skupa. Ovakve veze nisu poželjne u bazama podataka, no s obzirom na to da se često pojavljuju, u tom slučaju se uvodi novi entitet čiji je primarni ključ uglavnom sastavljen od primarnih ključeva glavnog entiteta [6].

Tablica 1.1 Vrste binarnih veza

Za identifikaciju pojedinog entiteta koristimo primarne ključeve (eng. Primary key). *Primarni ključ* je jedinstveni atribut svakog entiteta koji je uglavnom predstavljen nekim brojem koji ne može biti dodijeljen drugom entitetu, kao npr. OIB osobe. Primarni ključ se može kreirati korištenjem više atributa i on se tada naziva *kompozitni primarni ključ* (eng. Composite primary key).

Logičko oblikovanje: predstavlja drugu fazu u izradi baze podataka. Odabere se odgovarajući DBMS (npr. Oracle) te se stvore tablice prema izrađenom konceptualnom modelu. Cilj ove faze je stvoriti logičku shemu baze koja opisuje logičku strukturu baze u skladu s pravilima relacijskog modela podataka. Konceptualni model pruža apstraktan pogled na podatke i njihove međusobne odnose, bez tehničkih detalja, dok logički model detaljnije definira strukturu podataka pripremajući je za fizičku implementaciju [1].

Fizičko oblikovanje: predstavlja treću fazu u procesu izrade baze podataka. Glavni rezultat ove faze je fizička shema cijele baze koju nazivamo relacijskim modelom. U principu je relacijska shema korisnicima manje razumljiva od konceptualne jer su u njoj entiteti i veze među njima pretvoreni u relacije. Ključ jedne relacije koji je prepisan u drugu relaciju naziva se *strani ključ*. Ta shema je zapravo niz naredbi kojima se relacije iz logičke sheme realiziraju kao tablice. Koristeći odabrani DBMS stvore se tablice prema specifikacijama iz logičkog modela uključujući stupce, tipove podataka, primarne i strane ključeve. Definiraju se ograničenja podataka kako bi se osigurala njihova točnost (npr. NOT NULL, UNIQUE, VARCHAR, NUMBER). Nakon izrade i povezivanja tablica izrađuje se DDL, dodjeljuju se prava korisnicima i omogućuje korisnički pristup i unos podataka u izrađene tablice [1].

1.3. Oracle

Larry Ellison, Bob Miner i Edd Oates 1977. godine su pokrenuli konzultantsku tvrtku Software Development Laboratories, koja je postala Relational Software, Inc. (RSI), koji 1983. postaje Oracle Systems Corporation, a kasnije samo Oracle Corporation. Oracle Database (često se naziva još i Oracle DBMS, Oracle Autonomus Database) je vlasnički višemodalni sustav za upravljanje bazom podataka koji razvija i prodaje Oracle Corporation. Ime Oracle je nastao od projekta tvrtke Software Development Laboratories na kojem su radili za američku obavještajnu agenciju CIA, kojem je ime bilo „Project Oracle“. 1979. Oracle je predstavljen kao prvi komercijalno dostupan RDBMS baziran na SQL-u. Postoji više verzija ovog sustava, a u ovom završnom radu je korištena verzija „Oracle Database 11g Express Edition“ koja je javno dostupna i besplatna. U Oracle Database, shema baze podataka predstavlja zbirku logičkih struktura podataka, a korisnik baze podataka posjeduje shemu baze podataka koja ima isto ime kao korisničko ime. Najvažniji objekti baze podataka su tablice i indeksi. Tablica se definira imenom i skupom redaka, dok je svaki redak definiran nazivom, vrstom podatka te veličinom koju taj podatak može imati. Tijekom formiranja redaka možemo koristiti posebna ograničenja integriteta poput ograničenja NOT NULL koje osigurava da taj redak sadrži neku vrijednost, odnosno da ne bude prazan. Indeksi nam omogućavaju učinkovito lociranje traženih redaka u bazi podataka što nam posebno olakšava pretraživanje kod upita za određeni redak ili skup redaka. Osim već spomenute korištene verzije sustava, u radu će još biti korištena dva alata: *Oracle SQL Developer* čija je svrha pružanje grafičkog sučelja za upravljanje objektima baza podataka i *Oracle SQL Data*

Modeler s pomoću kojeg korisnici kreiraju baze (crtaju, kreiraju, analiziraju modele baze podataka) [7].

1.4. Pogledi

Pogled (eng. View) nam zapravo predstavlja virtualnu tablicu koja se temelji na jednoj ili više tablica. Ta tablica fizički ne postoji i ne pohranjuje nikakve podatke. Može se izvršiti svaki put kada se pozove, a stvara se upitom koji povezuje jednu ili više tablica. Svaki pravilan SQL upit se može koristiti u pogledu [8]. Pogled se jednostavno može zamisliti kao upit spremljen pod jednim nazivom u bazi podataka. Također se naziva i nematerijalizirani pogled kako bi se razlikovao od materijaliziranog pogleda koji sadrži podatke. Pogledi se ne ažuriraju izravno, ali promjene u tablicama podataka se odražavaju u prikazu tako da pogled uvijek ispisuje najnovije rezultate podataka. Mogu se koristiti na svakom mjestu gdje se koriste i tablice, tako ih možemo koristiti u klauzuli FROM našeg SQL upita te u naredbama INSERT, UPDATE ili DELETE. Za izradu pogleda možemo koristit Schema Manager ili naredbu:

```
CREATE VIEW ViewName AS SelectQuery
```

ViewName označava ime koje želimo dodijeliti pogledu, a SelectQuery označava upit kojim ćemo povezati podatke jedne ili više tablica. Konkretni primjer stvaranja pogleda će biti prikazan u nastavku ovog rada.

Pogled se ne može odbaciti izjavom DROP TABLE nego moramo koristiti DROP VIEW izjavu.

```
DROP VIEW ViewName
```

Pogledi imaju brojne prednosti poput izolacije podataka i kontrole pristupa tako da korisnici nemaju direktan pristup osnovnim tablicama tj. dozvoljava im se pristup samo određenim podacima putem pogleda, zatim poboljšanje čitljivosti koda i pojednostavljenje složenih SQL upita tako što enkapsuliraju složene SQL upite i skrivaju ih iza jednostavnih naziva što nam je jako korisno kod upita koje je potrebno često izvršavati [9].

1.5. Materijalizirani pogledi

Materijalizirani pogledi (eng. Materialized views) su u principu vrlo slični pogledima, a njihova glavna razlika je ta da materijalizirani pogled, za razliku od pogleda, pohranjuje rezultate napisanog upita u fizičku tablicu u bazi podataka. To nam omogućava brži pristup bazi podataka u odnosu na korištenje pogleda. Podaci koji se prikazuju pozivanjem materijaliziranog pogleda već su unaprijed izračunati i pohranjeni i tu se javlja glavni problem korištenja materijaliziranih pogleda u sustavima u kojima se podaci u bazama podataka vrlo često mijenjaju. Materijalizirane poglede je potrebno često ažurirati ukoliko želimo biti sigurni da imamo najnovije podatke koje dohvaćamo željenim upitom. Neke baze podataka zahtijevaju njihovo ručno osvježavanje, dok je nekoliko njih implementiralo automatsko ažuriranje [10]. Što se tiče Oracle Database, koji je korišten u ovom radu, on ima mogućnost da se materijalizirani pogledi mogu postaviti na ručno osvježavanje, na osvježavanje prema rasporedu ili ako SQL ispunjava određene zahtjeve moguće je i automatsko osvježavanje. Najveće prednosti materijaliziranih pogleda su u slučaju kada je složeni SQL upit potrebno izvršavati više puta te kada je vrijeme izvršavanja samog upita iznimno sporo. Tada se materijaliziranim pogledom, s obzirom na to da se ponaša kao jedna tablica, podaci mogu puno brže dohvatit, a često se koriste za izradu izvještaja za koje nisu nužni trenutni podaci, nego zbirni podaci na razini npr. dana, mjeseca... Za izradu materijaliziranog pogleda također možemo koristiti ili Schema Manager ili naredbu:

```
CREATE MATERIALIZED VIEW MatViewName AS SelectQuery
```

MatViewName označava ime koje dodjeljujemo izrađenom materijaliziranom pogledu, a SelectQuery predstavlja upit kojim želimo dobiti podatke iz jedne ili više tablica.

Za brisanje materijaliziranog pogleda koristimo naredbu:

```
DROP MATERIALIZED VIEW MaterializedViewName
```

Konkretni primjer stvaranja materijaliziranog pogleda na oba načina bit će prikazan u nastavku rada.

1.6. Pohranjene procedure

Pohranjena procedura (eng. Stored pocedure) predstavlja potprogram dostupan aplikacijama koje pristupaju sustavu upravljanja relacijskom bazom podataka. Pohranjene procedure imaju tri dijela:

- deklaracija (declaration) – definira ime pohranjene procedure
- tijelo (body) – definira gdje se piše kod
- iznimka (exception) – definira vrstu pogrešaka koje se mogu dogoditi

Također, one mogu primati parametre prilikom pozivanja, a razlikujemo dva tipa parametara:

- Formalni (engl. formal) - predstavljaju dio deklaracije pohranjene procedure i vraćaju vrijednosti u stvarne parametre u potprogramu
- Stvarni (engl. actual) - argumenti iz bloka potprograma koji daju podatke pohranjenoj proceduri; njihova prisutnost ima značenje samo u kontekstu bloka potprograma

Formalni parametri mogu imati tri moda:

- IN – definira formalni parametar kao ulaznu varijablu za pohranjeni potprogram; ovaj tip se može samo čitati
- OUT – definira formalni parametar kao spremnik za vraćanje vrijednosti u njemu odgovarajući parametar; može biti varijabla
- IN OUT – definira formalni parametar kao spremnik varijablu koja se može koristiti kako unutar izraza, tako i kao dodijeljene vrijednosti koje on vraća odgovarajućem stvarnom parametar

Za stvaranje pohranjene procedure koristimo izraz: *CREATE PROCEDURE* nakon kojeg definiramo naziv procedure, njene parametre, lokalne varijable te begin-end blok koji sadrži njen kod i obrađuje eventualne iznimke. Za modifikaciju pohranjene procedure postoje dva načina. U prvom načinu je svaki put moramo odbaciti s pomoću naredbe *DROP PROCEDURE*. Taj način izmjene se koristi kako bi se spriječile slučajne modifikacije pohranjenih procedura. Drugi način je korištenjem izraza *CREATE OR REPLACE* kojim će pohranjene procedure, ako postoje, biti odbačene i ponovno stvorene [8]. Oracle Database

Express Edition (u nastavku Oracle Database XE) omogućava pohranjivanje programa u bazu podataka. To svojstvo nam daje mogućnost pisanja i testiranja često potrebnog koda jednom, a zatim pristupanje tom kodu iz bilo koje aplikacije kojoj je taj kod potreban. Takav kod u programskim jezicima nazivamo procedurom. U kontekstu baze podataka, poput Oracle Database, procedure su dio PL/SQL jezika i koriste se za izvršavanje specifičnih operacija unutar baze podataka. Pohranjene procedure mogu se kompajlirati i pohraniti u Oracle Database XE te će tako spremljene biti spremne za izvršavanje kad to bude potrebno. Tako pohranjene procedure mogu se referencirati ili pozivati neograničen broj puta od strane različitih aplikacija. Procedure i funkcije stvorene izvan paketa nazivaju se pohranjeni ili samostalni potprogrami, odnosno paketirani potprogrami ukoliko su definirane unutar paketa. Najveća prednost procedura je njihovo organiziranje koda u modularne dijelove, što nam olakšava čitanje, održavanje i ponovno korištenje koda [11].

Konkretni primjer stvaranja pohranjene procedure bit će opisan u drugom poglavlju ovog rada.

2. Izrada projekta

Nakon što smo se upoznali s teorijskim dijelom vezanim za relacijske baze podataka te radom s Oracle alatom, možemo krenuti na izradu praktičnog dijela. Za potrebe ovog projekta izrađena je relacijska baza podataka pomoću Oracle alata. Nakon izrade modela u Oracle SQL Developer Data Modeler-u (u nastavku Data Modeler) slijedilo je pisanje upita u Oracle SQL Developer-u (u nastavku SQL Developer) te optimizacija upita korištenjem pogleda i materijaliziranih pogleda i na kraju izrada pohranjene procedure. Fokus je bio na usporedbi vremena izvršavanja korištenjem pogleda i materijaliziranih pogleda. Izrađene su dvije verzije baze podataka. U prvoj verziji korišteno je 1000 podataka, dok je u drugoj korišteno 1 000 000 podataka. Svaki od korištenih upita izvršen je minimalno pet puta kako bismo bili sigurni da sporedna izvršavanja nisu utjecala na vrijeme potrebno za izvršavanje upita baze podataka koje je dobiveno kao rezultat.

2.1. Specifikacije sustava

U današnjem svijetu, mnogi ljudi biraju da naruče jelo putem dostave hrane, nazivajući restoran ili koristeći aplikaciju za naručivanje. Izrađena baza bi omogućavala praćenje online narudžbi iz raznih restorana i dostavu te narudžbe. Tijekom procesa realizacije baze fokus je bio izraditi bazu podataka koja će voditi podatke vezane za dostavljačke tvrtke, klijente, pojedine restorane te za narudžbe poslane iz tih restorana. Kako bi korisnik mogao obaviti online narudžbu prvo mora odabrati željenu dostavljačku tvrtku. Nakon odabira tvrtke, korisniku su ponuđeni restorani za koje su navedene određene karakteristike ovisno o ponudi jelovnika. Korisnik bira iz kojega od ponuđenih restorana želi naručiti hranu te iz ponuđenog jelovnika bira koju vrstu hrane želi i želi li uz hranu odabrati i neko piće ili dodatak jelu. Odabirom željenih stavki stvara se narudžba na kojoj može biti više artikala. Ukoliko je narudžba uspješno odabrana korisniku se mora omogućiti da odabere koji način plaćanja želi.

2.2. Modeliranje podataka

Prilikom izrade modela podataka korištene su informacije iz stvarnoga svijeta te su na temelju njih formirani entiteti, a za svaki entitet su napisana svojstva, odnosno atributi. Proces modeliranja podataka imao je tri faze koje su uključivale izradu konceptualnog modela, logičkog modela i na kraju izradu relacijskog modela. Prije prelaska na formiranje modela navest ću entitete korištene u izradi baze, a zatim prikazati izradu pojedinog modela.

2.2.1. Entiteti

Narudžba

Ovaj entitet nam daje informacije o konkretnoj narudžbi, središnjoj točki ovoga modela. Jedinstvena je i sadrži informacije potrebne da bi se narudžba mogla uredno izvršiti, poput identifikacijskog broja narudžbe, datuma izvršene narudžbe, njezine cijene, adresu dostave, moguće napomene kupca te informacije o plaćanju i isporuci narudžbe. Tablica olakšava proces obrade narudžbi i upravljanje svakom od njih. U relaciji je s entitetima *Kupac*, *Vrsta plaćanja*, *Mjesto, Dostavljač*, *Restoran* i *Stavke*.

Kupac

Ovaj entitet označava korisnika koji pokreće narudžbu te daje potrebne informacije o njemu koje su važne za dostavu, poput imena, prezimena i telefonskog broja. Također, ovisno o željama kupca entitet može sadržavati određenu napomenu vezanu za njega, kao što bi primjerice bila napomena o alergijama koje kupac ima. Tablica omogućava praćenje informacija o svakom kupcu. U relaciji je s entitetom *Narudžba*.

Restoran

Restoran je entitet koji sadrži glavne informacije o dostupnim restoranima. Atributi sadržani u ovoj tablici su naziv i adresa restorana. Tablica omogućava korisnicima pregled mogućih opcija te uz pomoć dostupnih informacija olakšava njihov odabir restorana. Tablica je u relaciji s entitetom *Recenzija* kako bi bile vidljive recenzije kojima su kupci vrednovali usluge restorana te s entitetom *Karakteristika* u kojem je navedena glavna karakteristika restorana.

Karakteristika

Svaki restoran ima svoju glavnu karakteristiku s obzirom na jelovnike koje spremaju npr. restoran kineske hrane, restoran brze hrane, restoran bezglutenske hrane itd. Ova tablica omogućuje korisnicima da pregledaju i filtriraju restorane na temelju njihovih karakteristika, što olakšava odabir restorana prema osobnim preferencijama ili zahtjevima. Povezivanjem tablice *Karakteristika* s tablicom *Restoran* omogućuje se praćenje karakteristike određenog restorana.

Mjesto

Entitet Mjesto sadrži informacije o mjestu dostave narudžbe i mjestu u kojem se nalazi restoran. Tablica sadržava atribut naziv, a u relaciji je s entitetima *Narudžba* i *Restoran*.

Vrsta plaćanja

S obzirom na to kako se radi o online naručivanju korisniku je također omogućeno da izabere želi li plaćanje također obaviti online ili pak platiti gotovinom prilikom preuzimanja narudžbe. Tablica sadrži atribut naziv plaćanja. Entitet je povezan s entitetom *Narudžba*.

Stavke

Entitet *Stavke* omogućuje praćenje detalja o pojedinačnim artiklima koje kupci odabiru. S obzirom na to da jedna narudžba može sadržavati više jela, pića i dodataka, a svi oni mogu biti na više narudžbi, ovaj entitet ih povezuje te tvori jednu tablicu i omogućuje dodavanje više artikala iste vrste. Tablica sadrži informaciju količine odabranih artikala. U relaciji je s entitetima *Dodatak*, *Jelo*, *Piće i Narudžba*.

Jelo

Entitet koji sadrži informacije o različitim jelima koja su dostupna za narudžbu u pojedinom restoranu. Tablica omogućuje korisnicima pregled dostupnih jela te olakšava njihov odabir. Sadrži atribute o nazivu jela i njegovoj cijeni. U relaciji je s entitetima *Tip jela* i *Stavke.*

Tip jela

Kako bi se korisniku olakšao odabir jela, jelovnik je podijeljen prema tipu jela. Entitet sadrži atribut s kategorijom jela, a u relaciji je s entitetom *Jelo*.

Piće

Entitet koji olakšava odabir pića koje kupac želi uz svoj obrok te pruža informacije o nazivu i cijeni pojedinog pića kojeg restoran nudi. U relaciji je s entitetom *Stavke*.

Dodatak

Ovaj entitet omogućava korisniku opciju dodavanja priloga uz naručeno jelo. Sadrži informacije o nazivu i cijeni pojedinog dodatka. Entitet je u relaciji s entitetom *Stavke*.

Dostavljač

U entitetu *Dostavljač* nalaze se podaci o osobi koja obavlja dostavu narudžbe. Povezivanjem entiteta Dostavljač s entitetom Narudžba omogućava se praćenje kojem dostavljaču je dodijeljena određena narudžba. Entitet je povezan s entitetima *Narudžba, Tvrtka i Vozilo.*

Tvrtka

Entitet *Tvrtka* je entitet koji sadrži informacije o dostavljajućim tvrtkama koje pružaju usluge dostave hrane putem online narudžbe. Ova tablica omogućuje praćenje dostupnih dostavljačkih tvrtki navodeći njihov naziv kao atribut. Također, omogućuje pridruživanje dostavljača određenoj tvrtki. Povezivanjem entiteta *Tvrtka* s entitetom *Dostavljač* omogućava se praćenje u kojoj tvrtki je zaposlen određeni dostavljač.

Vozilo

Entitet Vozilo sadrži informacije o vozilima koja se koriste za dostavu hrane. Obuhvaća vrstu vozila (npr. automobil, bicikl, skuter) te je u relaciji s entitetom *Dostavljač* čime je omogućeno pridruživanje vozila određenim dostavljačima kako bi se olakšalo praćenje dostave.

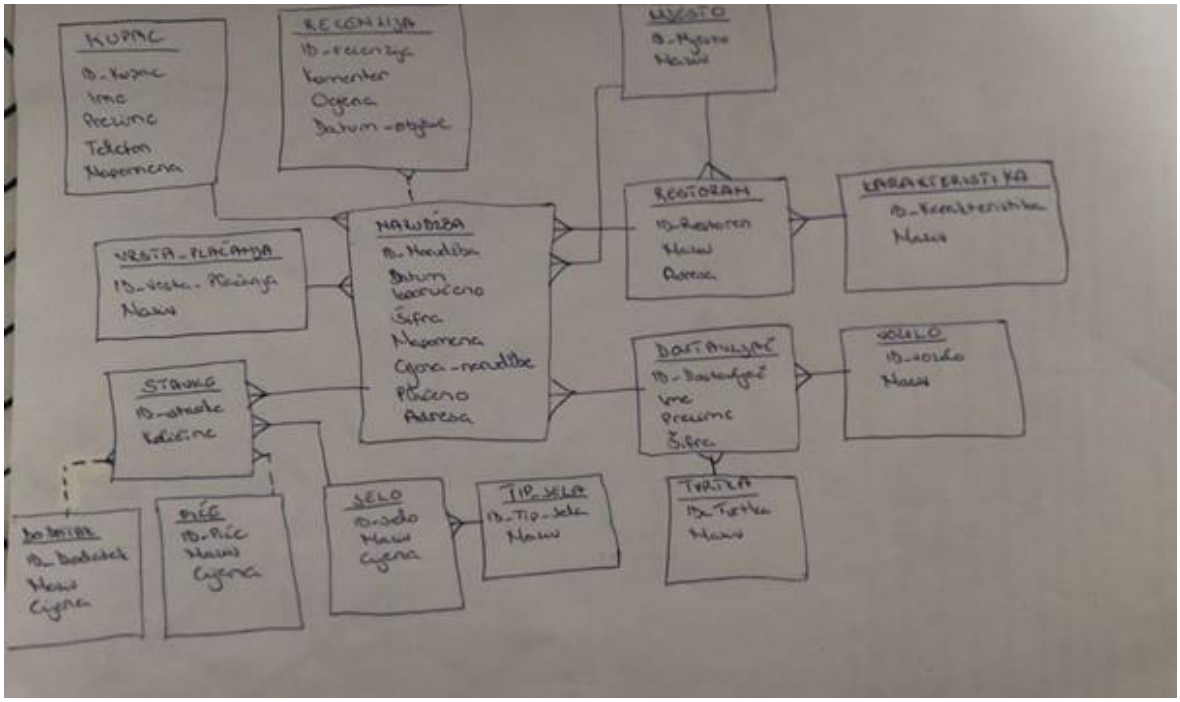
Recenzija

Svaki kupac ima mogućnost ostaviti recenziju za narudžbu u obliku ocjene ili komentara. Sadrži atribute komentar, ocjena i datum objave. Povezana je s entitetom *Narudžba.*

2.2.2. Konceptualni model

Početak realizacije projekta zapravo se odnosio na izradu konceptualnog modela baze podataka koji je izrađen na papiru i u principu predstavlja shemu baze sa svim potrebnim entitetima, atributima te vezama između svih entiteta. Daje nam dobar pregled u sustav kojeg

želimo što jednostavnije i intuitivnije izraditi. Na slici su prikazani svi prethodno navedeni entiteti, njihovi atributi i međusobna povezanost entiteta.

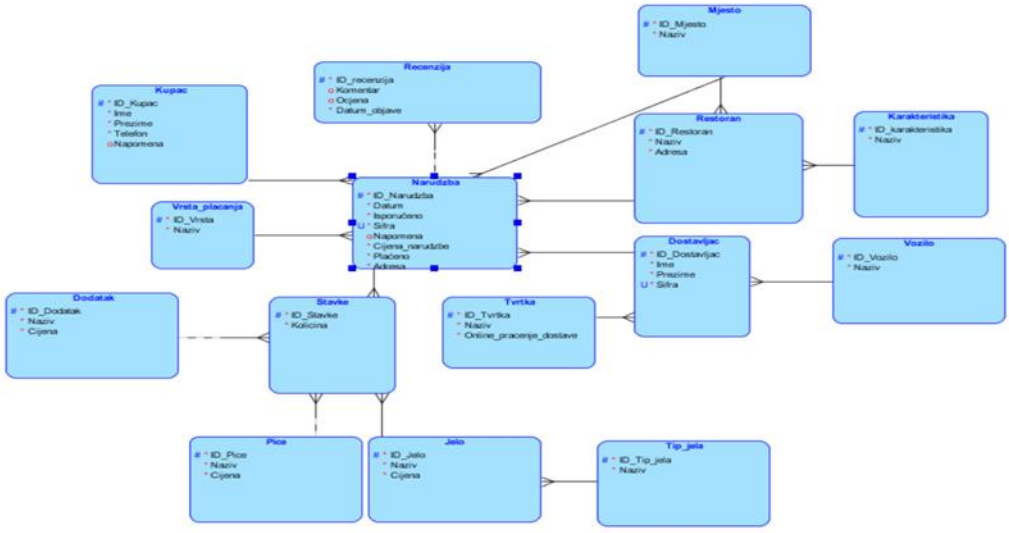


Slika 2.1 Konceptualni model baze podataka *Dostava hrane*

Na slici se vidi isprekidana veza između entiteta Dodatak i entiteta Stavke, čime je postavljeno da je dodatak opcionalan za kupca. Isto vrijedi i za entitete Piće i Recenzija .

2.2.3. Logički model

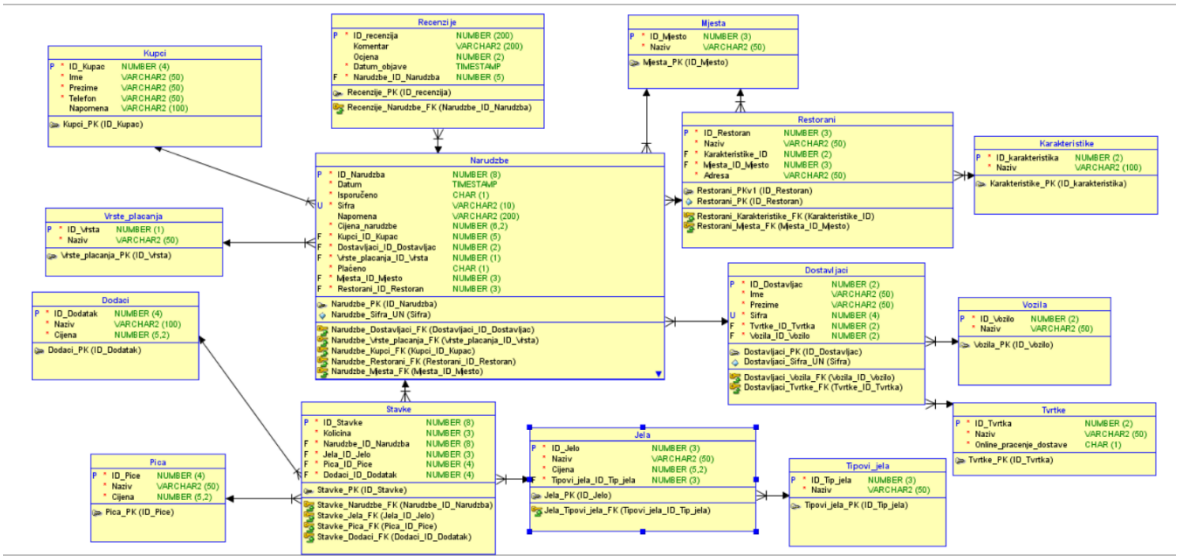
Nakon izrade konceptualnog modela, po uzoru na njega slijedila je izrada logičkog modela koji nam prikazuje detaljniju strukturu podataka. Logički model predstavlja tablice povezane određenim vezama. Za razliku od konceptualnog modela, logički model nam prikazuje tablice iz kojih možemo iščitati primarne ključeve, atribute i njihova svojstva, no nisu vidljivi strani ključevi kao ni tipovi podataka. Primarni ključevi imaju oznaku #. Možemo vidjeti oznake za opcionalne parametre kao što su parametri za atribut *Napomena* u tablici *Kupac*, zatim za atribute *Komentar* i *Ocjena* u tablici *Recenzija* itd.



Slika 2.2 Logički model baze podataka *Dostava hrane*

2.2.4. Relacijski model

Završna faza u razvoju modela baze podataka bila je izrada relacijskog modela. Relacijski model najdetaljnije prikazuje strukturu baze podataka. Omogućava nam uvid u tip podatka pojedinog atributa te njegova ograničenja. Također, vidljivi su nam primarni i strani ključevi za pojedinu tablicu, tako primjerice vidimo da centralna tablica *Narudžbe* ima najviše stranih ključeva zbog svoje povezanosti s ostalim tablicama. Primarni ključevi imaju oznaku P, dok strani ključevi imaju oznaku F. U tablici *Narudžbe* je također prikazana oznaka jedinstvenosti atributa *Sifra,* a time je spriječeno da dvije narudžbe imaju istu šifru kako ne bi došlo do problema zbog dostave pogrešne narudžbe. Tipovi podataka koje sam koristila su: numeric, datatime, bit, varchar i timestamp.



Slika 2.3 Relacijski model baze podataka *Dostava hrane*

2.3. Izrada baze podataka

2.3.1. Izrada DDL-a

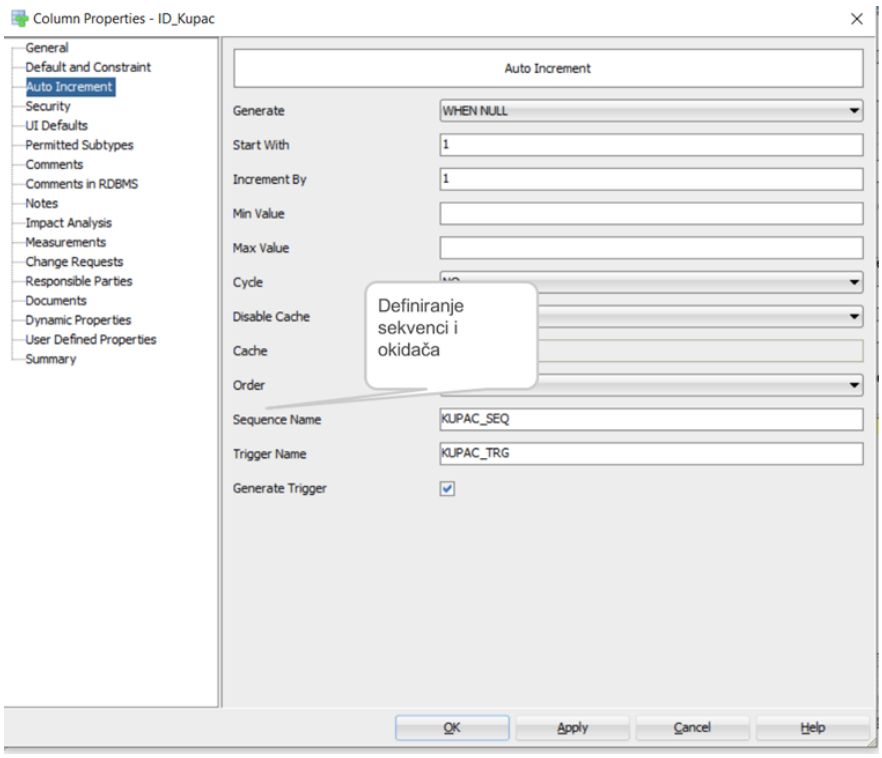
Nakon kreiranja svih modela idući korak bio je izrada DDL-a (Data Definition Language) koji će se izvršiti u SQL Developeru kako bismo iz modela dobili relacijsku bazu podataka. Time se omogućuje unos podataka u bazu, pisanje i izvršavanje različitih upita nad tim podacima. Budući da u Oracle-u ne postoji ugrađeni tip podataka za primarne ključeve, prije kreiranja DDL-a potrebno je svim primarnim ključevima napraviti sekvencu (eng. sequence) i okidač (eng. trigger).

Sekvence su sheme objekata u bazi podataka koji se koriste za generiranje primarnog ključa svaki put kada se sekvenca pozove. Na taj način se osigurava jedinstvenost svakog ključa [1].

Okidači se pokreću pokretanjem DML akcija u tablici: ažuriranje (UPDATE), upisivanje (INSERT) i brisanje (DELETE). Osiguravaju kontrolu nad bazom podataka i održavaju sinkronizirane tablice [1].

Odaberemo opciju autoincrement i definiramo vrijednost kojom započinje označavanje primarnih ključeva te vrijednost za koju će se oni povećavati (u ovom primjeru označavanje

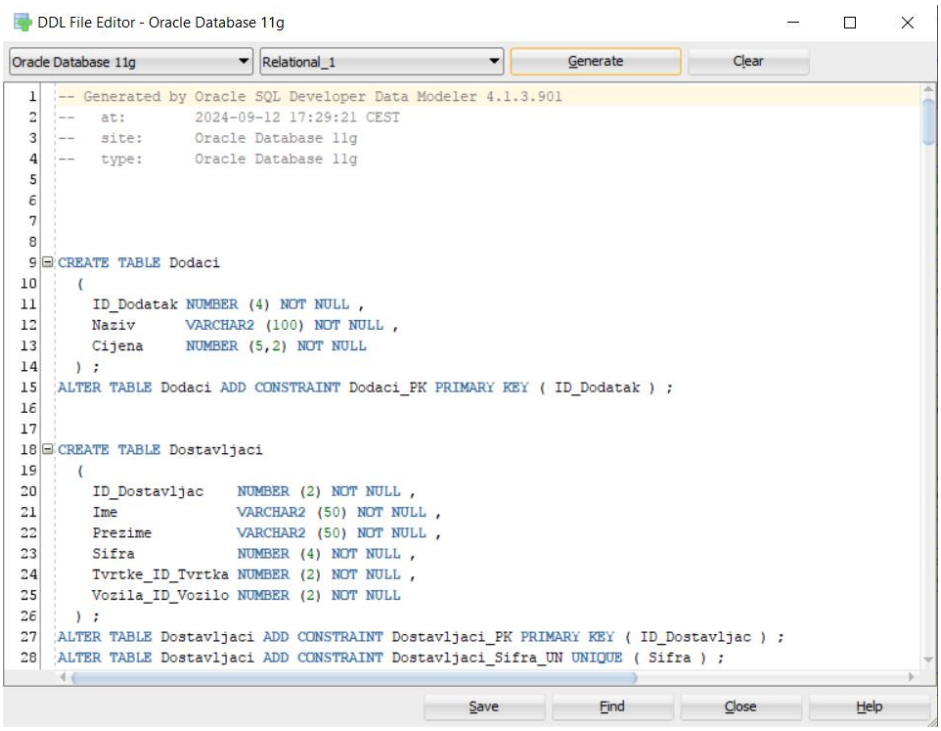
započinje s 1 i svaki put kada se pozove sekvenca ta vrijednost će se povećati za 1) (Slika 2.4).



Slika 2.4 Definiranje sekvenci i okidača

Svaka sekvenca i okidač moraju imati svoje ime, a u ovom slučaju imena su definirana tako da prvi dio imena sadrži ime entiteta i nastavak „_SEQ“ za sekvencu, odnosno „_TRG“ za okidač.

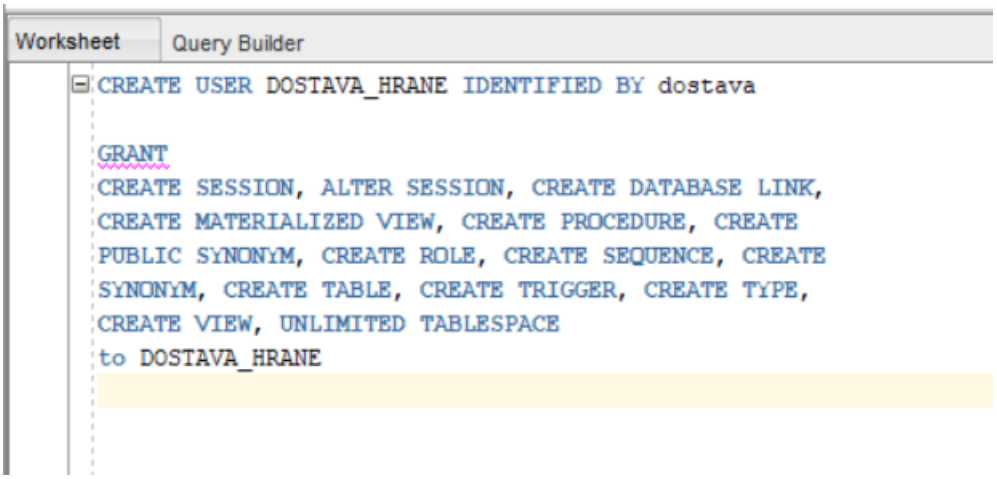
DDL datoteku kreiramo klikom na ikonu Generate DDL u SQL Modeleru. Generiramo traženi DDL koji predstavlja kod za kreiranje svih entiteta, atributa, sekvenci i okidača naše baze podataka (Slika 2.5).



Slika 2.5 Izvršavanje DDL-a

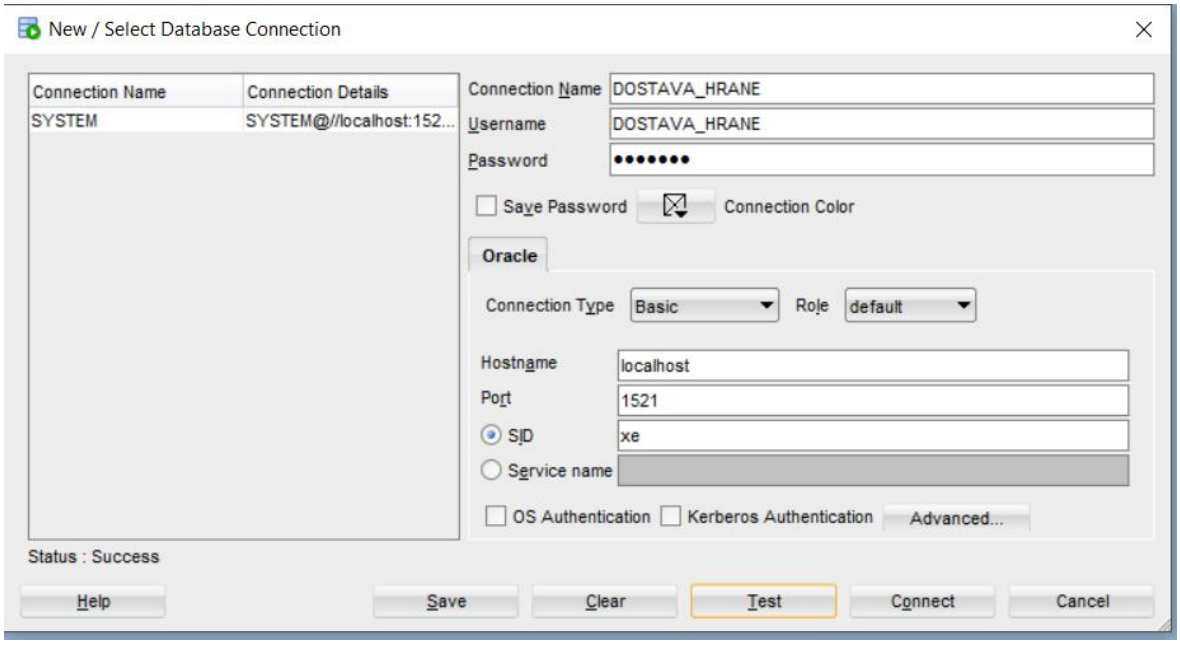
2.3.2. Izrada korisnika i veze

Nakon što je u SQL Modeleru kreiran DDL, u SQL Developeru stvaramo novog korisnika i dajemo mu ovlasti kojima će mu biti dopušteno izvršavanje operacija nad bazom podataka. Novom korisniku dodjeljujemo njegovo korisničko ime, u ovom slučaju to je *DOSTAVA_HRANE* i lozinku *dostava* koji će mu biti potrebni prilikom spajanja da bazu.



Slika 2.6 Kreiranje korisnika

Nakon kreiranja korisnika i dodjele ovlasti, potrebno je izvršiti prijavu na novog korisnika.



Slika 2.7 Prijava na novog korisnika

Nakon prijave novog korisnika slijedi izvršavanje prethodno generiranog DDL-a tako da kod iz generirane datoteke DDL-a kopiramo, a zatim zalijepimo u Worksheet novog korisnika.

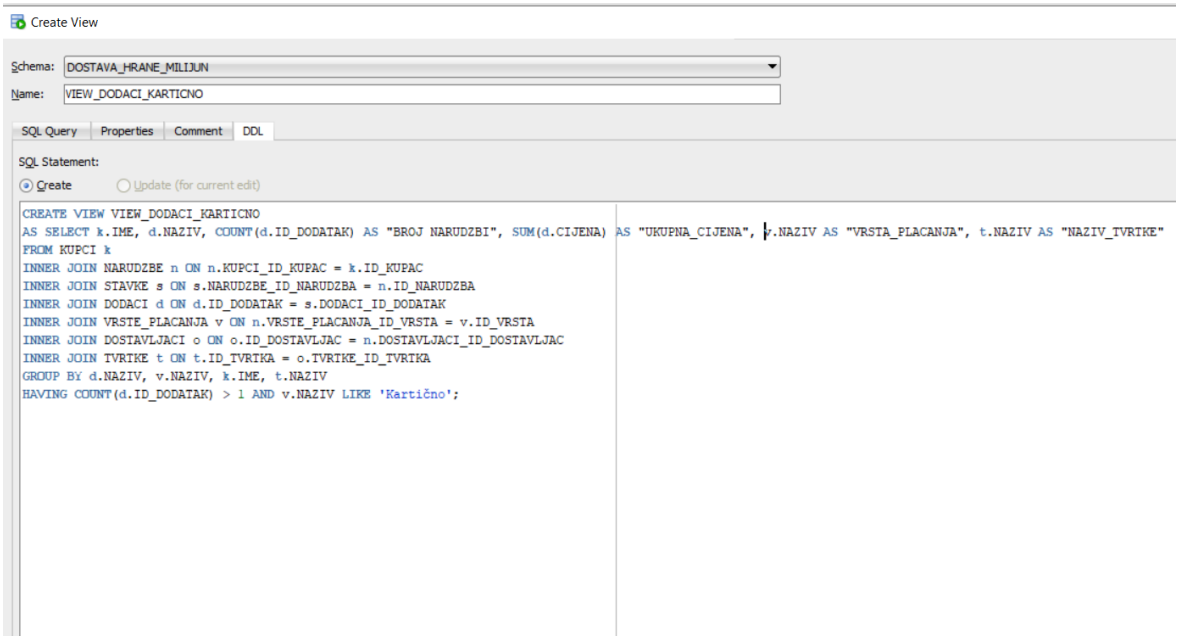
Slika 2.8 Izvršavanje DDL-a

Nakon svih ovih koraka baza podataka je konačno izrađena te još preostaje popuniti bazu podacima. S obzirom na to da je fokus izrade projekta na optimizaciji, odnosno usporedbi vremena potrebnog za izvršavanje upita korištenjem pogleda (eng. Views) i materijaliziranih pogleda (eng. Materialized Views), potrebne su različite količine podataka koje će biti spremljene u bazi i nad kojima ćemo testirati brzinu izvršavanja. Iz tog razloga su napravljene dvije kopije baze te je za svaku uvezena csv datoteka s različitim brojem

podataka. S obzirom na to da je tablica *Narudzbe* centralna tablica, pri uvozu podataka najveći fokus je bio na popunjavanju upravo te tablice, pa tako jedna verzija baze podataka sadrži 1000 narudžbi, dok druga verzija sadrži 1 000 000 narudžbi.

2.4. Izrada pogleda

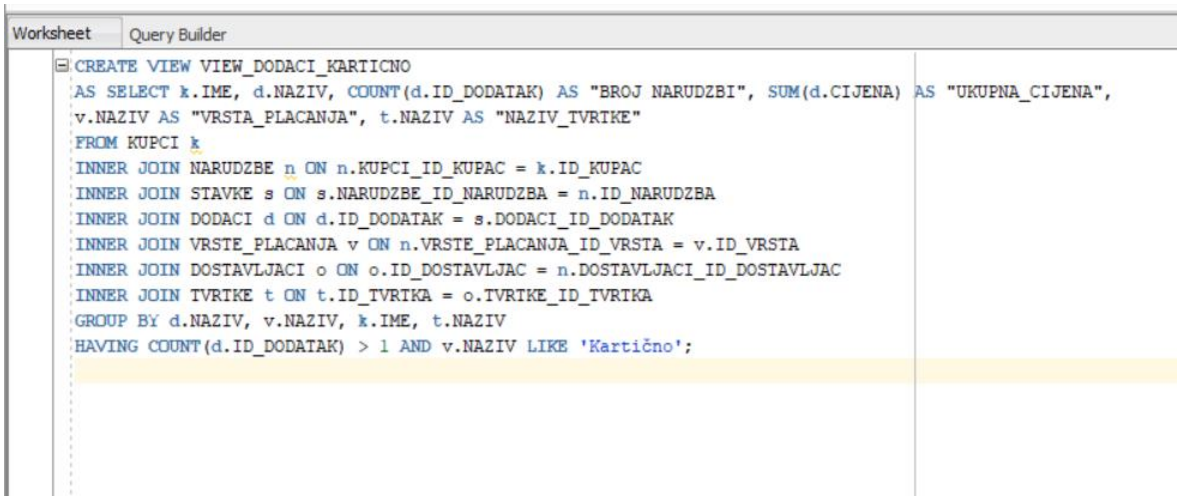
Nakon kreiranja baze podataka i uvoza podataka slijedila je izrada pogleda. Pogledi se mogu kreirati na dva načina. Prvi način ćemo prikazati kreirajući pogled pomoću grafičkog sučelja. Padajući izbornik u Developeru sadrži popis objekata koje možemo stvoriti. Odabirom Views otvara se prozor Create View u kojem kreiramo pogled unošenjem imena u okvir Name, a zatim upisujemo upit u okvir za tekst (Slika 2.9).



Slika 2.9 Kreiranje pogleda- grafičko sučelje

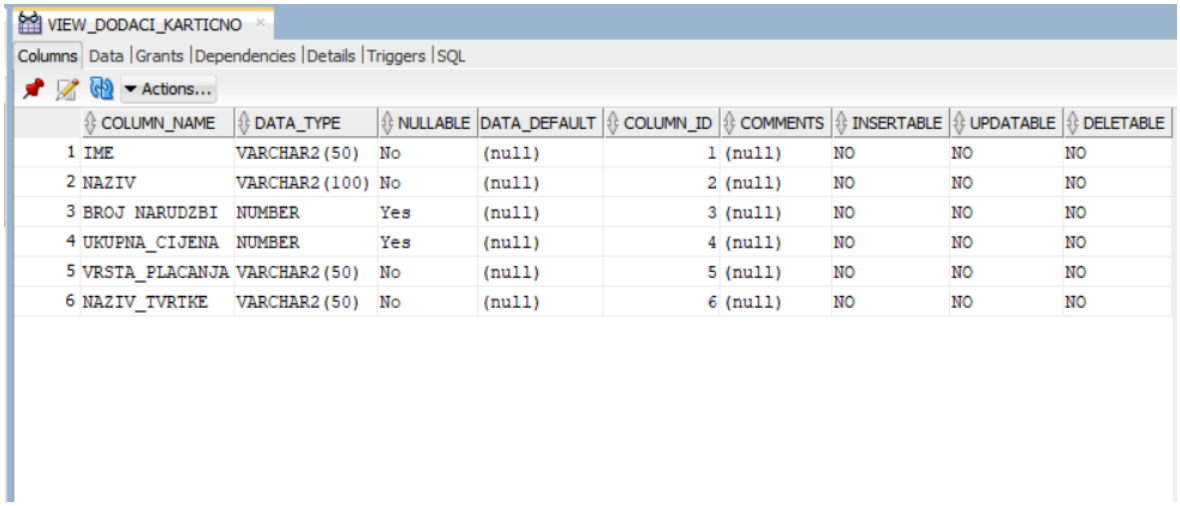
U ovom primjeru napisan je upit pod nazivom VIEW_DODACI_KARTICNO koji spaja ukupno 6 tablica, a daje nam popis kupaca koji su neki dodatak naručili više od jednoga puta i narudžbu platili online, odnosno kartično. Također se ispisuje ukupna cijena tih dodataka i naziv tvrtke koja je obavila dostavu.

Drugi način izrade ovakvog pogleda bio bi korištenjem naredbe 'CREATE VIEW' u SQL Worksheet-u. Nakon nje slijedi navođenje imena pogleda te pisanje željenog upita (Slika 2.10).



Slika 2.10 Kreiranje pogleda-Worksheet

Kreiranjem pogleda smo zapravo dobili virtualnu tablicu koja se sastoji od rezultata napisanog upita koji dohvaća podatke iz 6 različitih tablica. Tablica također sadrži stupce koji nam daju informacije može li se pomoću pogleda nešto nadodati, obrisati ili ažurirati originalnim tablicama (Slika 2.11).

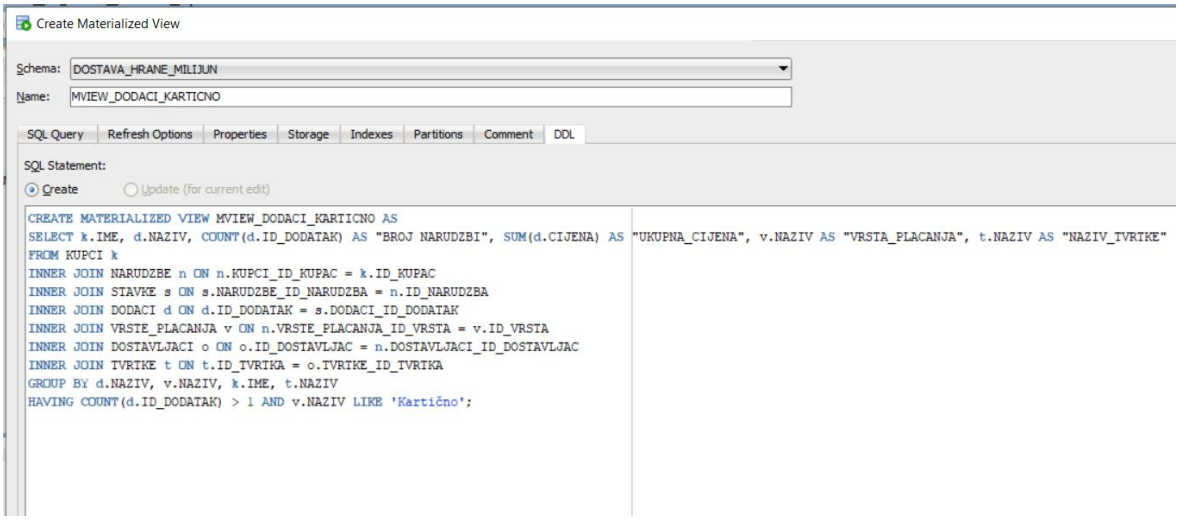


	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS	INSERTABLE	UPDATABLE	DELETABLE
1	IME	VARCHAR2(50)	No	(null)	1	(null)	NO	NO	NO
2	NAZIV	VARCHAR2(100)	No	(null)	2	(null)	NO	NO	NO
3	BROJ NARUDZBI	NUMBER	Yes	(null)	3	(null)	NO	NO	NO
4	UKUPNA_CIJENA	NUMBER	Yes	(null)	4	(null)	NO	NO	NO
5	VRSTA_PLACANJA	VARCHAR2(50)	No	(null)	5	(null)	NO	NO	NO
6	NAZIV_TVRTKE	VARCHAR2(50)	No	(null)	6	(null)	NO	NO	NO

Slika 2.11 Tablica nastala kreiranjem pogleda

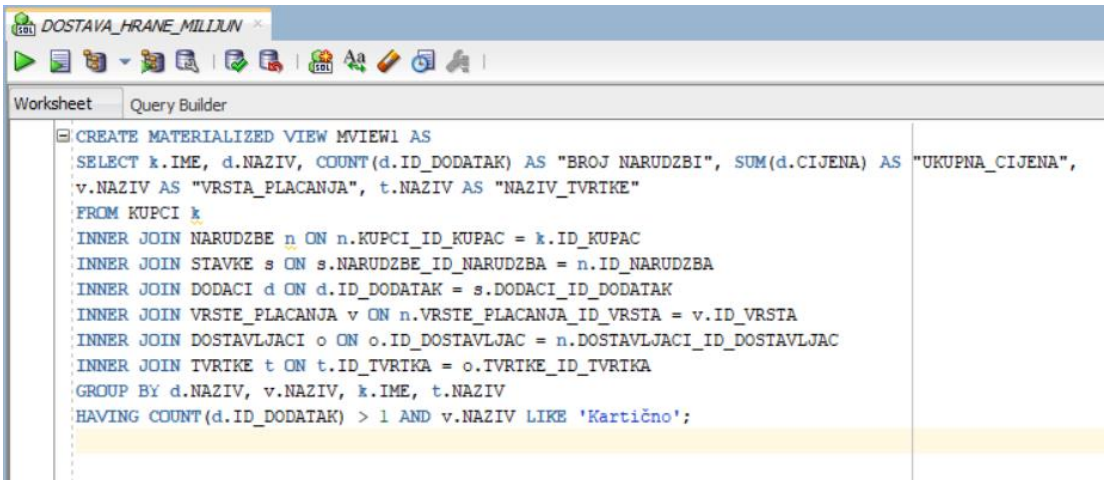
2.5. Izrada materijaliziranih pogleda

Za izradu materijaliziranih pogleda također koristimo dva načina. Izrađujemo ih na sličan način kao i poglede. Prvi način je pomoću grafičkog sučelja i odabirom Materialized Views. Otvara nam se prozor poput onoga za izradu pogleda te unosimo ime u okvir Name, a u okvir za tekst upisujemo željeni upit (Slika 2.12).



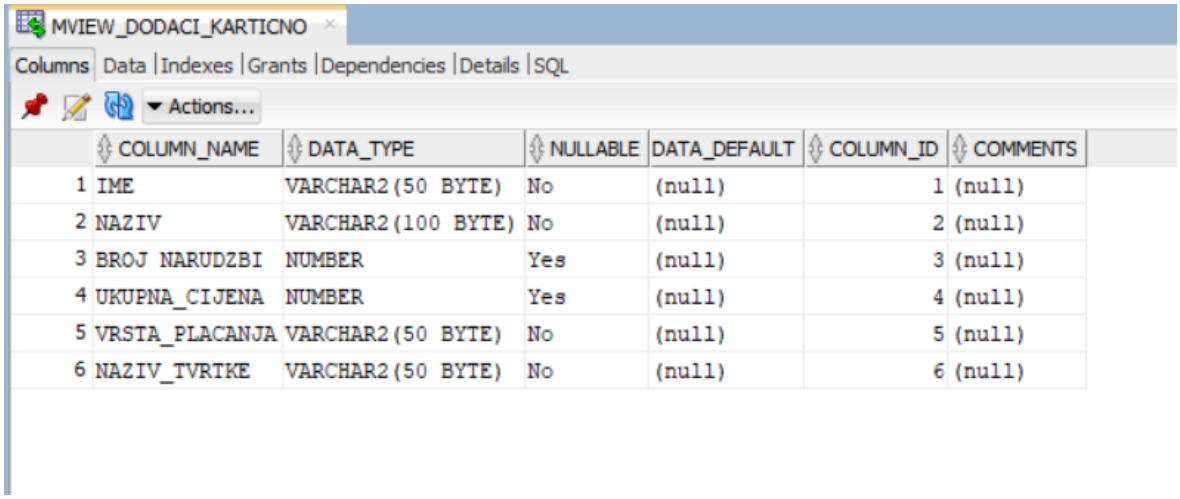
Slika 2.12 Kreiranje materijaliziranog pogleda- grafičko sučelje

Drugi način je gotovo isti kao kod izrade pogleda, samo uz naredbu 'CREATE MATERIALIZED VIEW' u SQL Worksheet-u, nakon koje slijedi navođenje imena za materijalizirani pogled te pisanje željenog upita (Slika 2.13).



Slika 2.13 Kreiranje materijaliziranog pogleda- Woksheet

Izradom materijaliziranog pogleda dobijemo gotovo istu tablicu kakvu smo dobili i izradom pogleda (Slika 2.14). Glavna razlika u ovim tablicama je zapravo prethodno spomenuto osvježavanje podataka koje te tablice prikazuju. Materijalizirani pogledi pohranjuju rezultate upita, što omogućuje brzi pristup podacima bez potrebe za njihovim ponovnim izračunavanjem, dok pogledi svakim pozivanjem ponovno izvršavaju upit i daju najnovije podatke iz tablica.


MVIEW_DODACI_KARTICNO

Columns | Data | Indexes | Grants | Dependencies | Details | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	IME	VARCHAR2(50 BYTE)	No	(null)	1	(null)
2	NAZIV	VARCHAR2(100 BYTE)	No	(null)	2	(null)
3	BROJ_NARUDZBI	NUMBER	Yes	(null)	3	(null)
4	UKUPNA_CIJENA	NUMBER	Yes	(null)	4	(null)
5	VRSTA_PLACANJA	VARCHAR2(50 BYTE)	No	(null)	5	(null)
6	NAZIV_TVRTKE	VARCHAR2(50 BYTE)	No	(null)	6	(null)

Slika 2.14 Tablica nastala kreiranjem materijaliziranog pogleda

2.6. Izvršavanje pogleda i materijaliziranih pogleda

2.6.1. Specifikacije računala

Na izvršavanje upita u Oracle SQL Developeru mogu utjecati i hardverske specifikacije računala na kojem se baza izvršava, kao što su tip i brzina procesora, količina RAM-a itd. Na primjer, procesori visoke brzine mogu ubrzati obradu zahtjevnih upita, dok veće količine RAM-a omogućuju bazi da efikasnije manipulira velikim skupovima podataka. Prije same usporedbe izvršavanja vidjet ćemo neke od specifikacija korištenog računala u sljedećoj tablici:

Naziv proizvoda	HP EliteBook 850 G4
Ukupna memorija	8.00 GB (7.79 GB iskoristivo)
Procesor	Intel™ Core™ i5-7300U CPU @ 2.60GHz

Memorija	8192MB RAM ~ 8GB
Naziv operacijskog sustava	Microsoft Windows 10 Pro

Tablica 2.1 Tablica specifikacija računala

Različite vrste upita mogu imati različite zahtjeve u pogledu resursa procesora, memorije i pohrane. U konačnici, dostupna hardverska specifikacija je solidna za mnoge osnovne i srednje zahtjevne operacije u Oracle bazi podataka, no važno je pažljivo pratiti opterećenje sustava i prilagoditi konfiguraciju baze podataka kako bi se postigle najbolje performanse s obzirom na resurse dostupne na računalu.

2.6.2. Usporedba izvršavanja

Usporedbu vremena izvršavanja različitih upita korištenjem pogleda i materijaliziranih pogleda provela sam praćenjem i zapisivanjem vremena prilikom izvršavanja po četiri različita upita, odnosno izvršena su četiri pogleda i četiri materijalizirana pogleda na obje verzije baze podataka. S obzirom na to da na vrijeme izvršavanja mogu utjecati i drugi faktori, kako bismo bili sigurni u točnost dobivenih rezultata, svaki upit je izvršen pet puta te je kao konačan rezultat zabilježeno srednje vrijeme tih pet izvršavanja. Svi pogledi i materijalizirani pogledi su izrađeni na jedan od prethodno prikazanih načina, koristeći upite koje ću navoditi u nastavku ovog rada.

Prvi upit koji je korišten za mjerenje i usporedbu vremena bio je jednostavan select upit koji nam dohvaća popis svih narudžbi uključujući i sve informacije o njima:

```
SELECT
"ID_NARUDZBA","DATUM","ISPORUČENO","SIFRA","NAPOMENA","CIJENA_NARUDZBE"
,"KUPCI_ID_KUPAC","DOSTAVLJACI_ID_DOSTAVLJAC","VRSTE_PLACANJA_ID_VRSTA"
  ,"PLAĆENO","MJESTA_ID_MJESTO","RESTORANI_ID_RESTORAN" FROM NARUDZBE;
```

Na temelju njega izrađen je pogled i materijalizirani pogled koji su najprije izvršeni nad bazom koja ima 1000 zapisanih narudžbi, a zatim nad bazom s 1 000 000 narudžbi. Dobiveni podaci su prikazani u sljedećoj tablici, a vrijeme je izraženo u sekundama.

BROJ PODATAKA	1000	1 000 000
POGLED	0.0852	75.8926
MATERIJALIZIRANI POGLED	0.0578	74.5798

Tablica 2.2 Usporedba izvršavanja prvog upita

Drugi upit na kojem je mjereno vrijeme bio je jednostavni select upit koji nam vraća popis narudžbi gdje je id kupca manji od broja 9500:

```
SELECT
"ID_NARUDZBA","DATUM","ISPORUČENO","SIFRA","NAPOMENA","CIJENA_NARUDZBE"
,"KUPCI_ID_KUPAC","DOSTAVLJACI_ID_DOSTAVLJAC","VRSTE_PLACANJA_ID_VRSTA"
  ,"PLAĆENO","MJESTA_ID_MJESTO","RESTORANI_ID_RESTORAN" FROM NARUDZBE
                    where KUPCI ID KUPAC<9500;
```

Izrađeni su pogled i materijalizirani pogled koji su najprije izvršeni nad bazom koja ima 1000 zapisanih narudžbi, a zatim nad tablicom s 1 000 000 narudžbi. Dobiveni podaci su prikazani u sljedećoj tablici, a vrijeme je izraženo u sekundama.

BROJ PODATAKA	1000	1 000 000
POGLED	0.0912	74.0812
MATERIJALIZIRANI POGLED	0.0852	69.3422

Tablica 2.3 Usporedba izvršavanja drugog upita

Treći upit na kojem je izvršena usporedba vremena izvršavanja pomoću pogleda i materijaliziranog pogleda je upit koji nam pronalazi sve narudžbe čija je cijena manja od 1000 ili je narudžba plaćena kartično:

```
SELECT
k.IME || ' ' || k.PREZIME as "KUPAC"
,n.SIFRA,n.DATUM,n.ISPORUČENO,n.NAPOMENA,d.IME as
"DOSTAVLJAC",n.CIJENA_NARUDZBE, v.NAZIV AS "VRSTA PLACANJA"
FROM NARUDZBE n INNER JOIN  VRSTE_PLACANJA v on
v.ID_VRSTA=n.VRSTE_PLACANJA_ID_VRSTA
INNER JOIN KUPCI k on k.ID_KUPAC=n.KUPCI_ID_KUPAC
INNER JOIN  DOSTAVLJACI d ON d.ID_DOSTAVLJAC=n.DOSTAVLJACI_ID_DOSTAVLJAC
WHERE n.CIJENA_NARUDZBE<1000 OR v.NAZIV LIKE 'Kartično';
```

Podaci dobiveni izvršavanjem pogleda i materijaliziranog pogleda za taj upit, nakon izvršavanja na obje verzije baze podataka prikazani su u sljedećoj tablici, a vrijeme je izraženo u sekundama.

BROJ PODATAKA	**1000**	**1 000 000**
POGLED	0.1478	54.866
MATERIJALIZIRANI POGLED	0.0692	52.564

Tablica 2.4 Usporedba izvršavanja trećeg upita

Četvrti upit na osnovu kojega su izrađeni pogled i materijalizirani pogled je upit kojim dohvaćamo narudžbe pod uvjetom da im je datum u zadanom rasponu.

```
SELECT
"ID_NARUDZBA","DATUM","ISPORUČENO","SIFRA","NAPOMENA","CIJENA_NARU
DZBE","KUPCI_ID_KUPAC","DOSTAVLJACI_ID_DOSTAVLJAC","VRSTE_PLACANJA
_ID_VRSTA","PLAĆENO","MJESTA_ID_MJESTO","RESTORANI_ID_RESTORAN"
FROM NARUDZBE WHERE DATUM >= '01.01.2020' AND DATUM <='30.6.2024';
```

Usporedba izvršavanja pogleda i materijaliziranog pogleda navedena je u sljedećoj tablici, a rezultati su navedeni u sekundama.

BROJ PODATAKA	1000	1 000 000
POGLED	0.079	71.441
MATERIJALIZIRANI POGLED	0.0672	70.7316

Tablica 2.5 Usporedba izvršavanja četvrtog upita

Iz svih navedenih podataka mjerenja i usporedbe vremena izvršavanja možemo primijetiti da se vrijeme izvršavanja mijenja s obzirom na korištenje različitih upita i njihovih uvjeta, no razlike nisu značajne. Možemo zaključiti da količina uvjeta ne utječe uvelike na vrijeme izvršavanja te da je izvršavanje materijaliziranog pogleda u svim testiranjima brže u odnosu na izvršavanje pogleda.

2.7. Izrada pohranjene procedure

Pohranjene procedure imaju jednostavnu definiciju sheme koja sadrži tri glavna bloka, a to su: blok deklaracije, izvršni blok i blok za upravljanje iznimkama:

```
CREATE Procedure <ime procedure> <argumenti popis>
Odjeljak deklaracije varijabli
BEGIN
Odjeljak za pokretanje PL/SQL-a ili SQL-a
EXCEPTION
Odjeljak za pokretanje PL/SQL-a ili SQL-a
END;
```

Deklaracija pohranjene procedure definira njeno ime, parametre za njeno pokretanje i varijable koje se koriste unutar te procedure. U tijelu se definira kod PL/SQL izraza, a iznimka definira vrstu pogrešaka koje se mogu pojaviti i ona se poziva svaki put kada se dogodi pogreška koja je definirana u PL/SQL grupi.

Općenita sintaksa stvaranja procedure:

```
CREATE [OR REPLACE] PROCEDURE ime_procedure
[(parametar [IN | OUT | IN OUT] tip_podataka [{ := | DEFAULT }
izraz],,
...
[(parametar [IN | OUT | IN OUT | tip_podataka)] {IS | AS}
BEGIN
<tijelo_procedure>
END ime_procedure;
```

Naredba CREATE OR REPLACE automatski odbacuje i ponovno stvara pohranjenu proceduru.

Naredba kojom pozivamo proceduru je:

```
BEGIN
    ime_procedure ;
 END
```

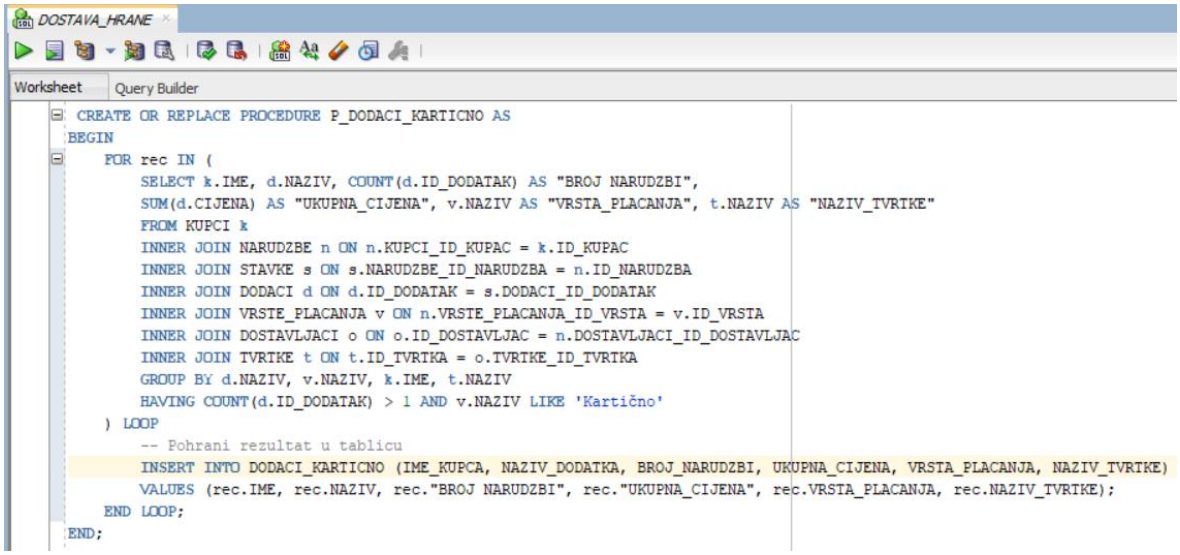
Na ovakav način proceduru možemo pozivati iz glavnog programa sve dok je prisutna na serveru [12]. U nastavku ću kreirati jednostavnu proceduru koja će zahtijevati novu tablicu koja će se popuniti izvršavanjem procedure i u kojoj će biti zapravo rezultati upita napisanog u proceduri koji nam dohvaća podatke vezane za narudžbe dodataka koji su naručeni više od jednom i za koje je narudžba plaćena karticom. Najprije je bilo potrebno kreirati odgovarajuću tablicu:

Worksheet | Query Builder

```
CREATE TABLE DODACI_KARTICNO(
    IME_KUPCA VARCHAR2(100),
    NAZIV_DODATKA VARCHAR2(100),
    BROJ_NARUDZBI NUMBER,
    UKUPNA_CIJENA NUMBER,
    VRSTA_PLACANJA VARCHAR2(100),
    NAZIV_TVRTKE VARCHAR2(100)
);
```

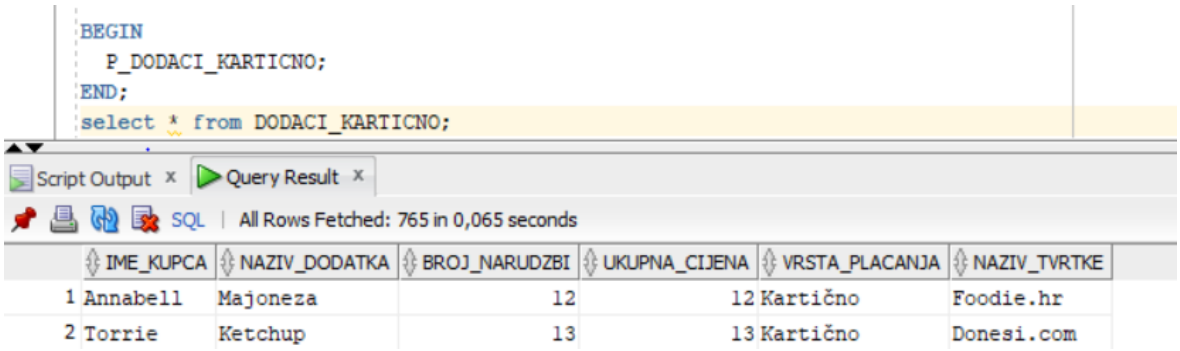
Slika 2.15 Kreiranje tablice za pohranu rezultata procedure

Zatim slijedi kreiranje procedure kojom će se ta tablica popuniti odgovarajućim podacima:



Slika 2.16 Kreiranje procedure

Pozivanje i rezultat kreirane procedure:



Slika 2.17 Pozivanje procedure

Zaključak

Baze podataka predstavljaju ključan dio svake aplikacije. Svaka baza podataka mora proći kroz nekoliko faza razvoja i svaka od tih faza mora biti smisleno i razumljivo odrađena kako bi konačna baza podataka bila jednostavna za korištenje. Relacijske baze podataka su pogodne za organizaciju strukturiranih podataka te su dobar izbor za aplikacije kojima je potrebna baza podataka koja sadrži cjelovite informacije. Također, jedna od ključnih stvari koje su važne za bazu podataka je brzina dohvaćanja traženih podataka i važno je korištenje najbolje opcije koja omogućuje brzinu i efikasnost dohvaćanja podataka.

U ovom radu opisan je rad Oracle Database alata koji je i korišten za izradu baze podataka. Baza podataka izrađena u svrhu ovoga rada može biti korištena za aplikacije koje se koriste za online naručivanje hrane jer omogućava pohranjivanje svih informacija koje su potrebne za izradu narudžbe iz jednog restorana. Sam proces izrade baze podataka bio je vrlo zanimljiv. Na samom početku je trebalo razmisliti koje to entitete takva baza podataka treba sadržavati. Potom je izrađen konceptualni model baze podataka, odnosno model na papiru koji sadrži sve entitete i atribute, a među glavnim entitetima su tablice Narudzba i Kupac. To je prva i osnovna faza razvoja baze podataka u kojoj treba pomno razmišljati kako organizirati entitete i njihovu međusobnu povezanost kako bi se izbjegli mogući kasniji problemi. Tako je primjerice nastao problem prilikom izrade veze između entiteta pojedinih stavki restorana (piće, hrana, dodatak…) i entiteta Narudzba. S obzirom na to da jedna stavka može biti naručena više puta u jednoj narudžbi, bilo je potrebno uvesti još jednu tablicu, u ovom slučaju tablicu pod nazivom Stavke, koja će omogućavati ispravno spremanje takvih podataka.

Nakon izrade konceptualnog modela, na osnovu njega, izrađen je logički model, a zatim i relacijski model korištenjem Data Modelera. Kako bismo relacijski model pretvorili u stvarnu bazu podataka korišten je DDL koji je izvršen u SQL Developeru. Nakon toga je bilo potrebno unijeti podatke u kreirane tablice i dobili smo funkcionalnu bazu podataka.

Nakon kreiranja baze slijedio je ujedno i glavni dio ovoga rada, a to je kreiranje pogleda, materijaliziranih pogleda i procedura. Nakon kreiranja pogleda i materijaliziranih pogleda slijedila je usporedba njihovoga rada na nekoliko različitih upita. Svaki od njih ima svoje prednosti i nedostatke. Materijalizirani pogledi su vrlo pogodni za aplikacije koje imaju

velike količine podataka i kojima nisu nužni trenutni podaci, nego primjerice podaci na razini dana, mjeseca, godine, kakvi su uglavnom potrebni i u restoranima za praćenje narudžbi obavljenih uz pomoć aplikacija sa sličnom bazom podataka kakva je izrađena i u ovom radu. Također su praktični za izvršavanje složenih upita jer omogućavaju mnogo brže izvršavanje nego što bi trajalo izvršavanje upita na uobičajen način. S druge strane, kod pogleda nije potrebno brinuti o točnosti dobivenih rezultata, jer za razliku od materijaliziranih pogleda, pogledi nam uvijek daju najnovije podatke iz baze. Pohranjene procedure su vrlo korisne u složenim sustavima kada želimo postići veću organiziranost koda jer omogućavaju njegovu raspodjelu u modularne dijelove, što nam olakšava njegovo razumijevanje, održavanje i ponovno korištenje.

Uzevši u obzir ove metode, njihove prednosti i nedostatke, zaključujemo da je prilikom odabira metode bitno najprije odrediti koji su nam ciljevi, odnosno što nam je važno prilikom dohvata podataka (brzina, točnost, zauzeće memorije..). Baze podataka su ključan dio svakog sustava i svake aplikacije i vrlo je važno znati upravljati njima i njihovim podacima.

Literatura

[1] R. Manger, Baze podataka, Zagreb: Element d.o.o., 2012.

[2] T. Dadić, Baze podataka, Split: Sveučilište u Splitu, 2012..

[3] G. Čuljak, »Prezi,« 21 3 2016. [Mrežno]. Available: https://prezi.com/eunad9-q84o7/12-coddovih-pravila/.

[4] R. Peterson, »GURU99,« 9 ožujak 2024. [Mrežno]. Available: https://www.guru99.com/what-is-dbms.html.

[5] »IBM,« [Mrežno]. Available: https://www.ibm.com/topics/relational-databases.

[6] M. Varga, Baze podataka, Zagreb: Vlastita naklada, 2022.

[7] »Oracle,« [Mrežno]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/introduction-to-oracle-database.html#GUID-A42A6EF0-20F8-4F4B-AFF7-09C100AE581E. [Pokušaj pristupa 2 7 2024].

[8] C. McCullough-Dieter, Oracle 8 biblija, Zagreb: Znak, 1998.

[9] »Oracle,« 2 7 2024. [Mrežno]. Available: https://docs.oracle.com/en/database/other-databases/timesten/22.1/operations/understanding-materialized-views.html#GUID-72B7472A-6990-4C10-A897-86913C067023.

[10] »aws,« [Mrežno]. Available: https://aws.amazon.com/what-is/materialized-view/.

[11] »Oracle Help Center,« [Mrežno]. Available: https://docs.oracle.com/cd/B25329_01/doc/appdev.102/b25108/xedev_programs.htm. [Pokušaj pristupa 7 srpanj 2024].

[12] T. Ladan, Hrvatska enciklopedija, Zagreb: Leksikografski zavod Miroslav Krleža, 2009..

[13] B. Beljo, »UNIRI,« rujan 2018. [Mrežno]. Available: https://repository.inf.uniri.hr/islandora/object/infri%3A333/datastream/PDF/view.

[14] D. Carter, »GURU99,« 2024 lipanj 20. [Mrežno]. Available: https://www.guru99.com/hr/data-modelling-conceptual-logical.html.[Pokušaj pristupa 28 lipanj 2024].

Popis slika