Korištenje JSON formata u Oracle bazi podataka

Preskočil, Marko

VELEUČILIŠTE U BJELOVARU
PREDDIPLOMSKI STRUČNI STUDIJ RAČUNARSTVO

KORIŠTENJE JSON FORMATA U ORACLE BAZI PODATAKA

Završni rad br. 06/RAČ/2021

Marko Preskočil

Bjelovar, listopad 2021.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Kandidat: **Preskočil Marko** Datum: 27.08.2021. Matični broj: 001988

JMBAG: 0314019670

Kolegij: **BAZE PODATAKA**

Naslov rada (tema): **Korištenje JSON formata u Oracle bazi podataka**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Tomislav Adamović, mag.ing.el.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Krunoslav Husak, dipl.ing.rač., predsjednik**
2. **Tomislav Adamović, mag.ing.el., mentor**
3. **Ante Javor, struč.spec.ing.comp., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 06/RAČ/2021

U radu je potrebno izraditi Oracle procedure koje će izložiti servis za dohvaćajne podataka u JSON formatu iz relacijskog modela baze podataka. Servis treba imati veliku propusnost podataka te specifične organizacije podataka: liste vrijednosti, pregled jednog zapisa i pregled cijele tablice. Dohvaćanje podataka izraditi koristeći Oracle objekt "pogled". Testirati servis pomoću POSTMAN alata.

Zadatak uručen: 27.08.2021. Mentor: **Tomislav Adamović, mag.ing.el.**

REPUBLIKA HRVATSKA VELEUČILIŠTE U BJELOVARU BJELOVAR

SADRŽAJ

SLIKE

1. UVOD

Unutar Oracle baze podataka za pisanje poslovne logike koristi se PL/SQL programski jezik koji se sastoji od funkcija i procedura. Prilikom komunikacije između procedura i funkcija u PL/SQL programskom jeziku koristi se tip (engl. *type*). Tip je striktno definiran te se ne može lako mijenjati. PL/SQL programski jezik je strogo tipizirani proceduralni jezik, što znači da prije rada s varijablama ili konstantama potrebno ih je deklarirati s određenim tipom podatka. Iako postoje unaprijed definirani tipovi podataka, moguće je kreirati i vlastite tipove. Neki od tih tipova su zapisi, zbirke te tipovi objekata.

Zapisi se mogu vrlo lako definirati na temelju postojećih struktura poput pokazivača ili tablice. Vrsta zapisa složena je vrsta podataka koja programeru omogućuje stvaranje nove vrste podataka sa željenom strukturom podataka, to pojednostavljeno znači stvaranje novog tipa podataka. Nakon što će se vrsta zapisa stvoriti, ona će se pohraniti kao nova vrsta podataka u bazi podataka. Zbirke su skupine nizovi u PL/SQL programskom jeziku, dijele se na tri vrste nizova: asocijativne tablice, ugniježđene tablice i nizove. Zbirke su korisne kada su potrebni privremeni skupovi podatka, a ne koriste se tablice. Nakon što je novi tip definiran, moguće ga je koristiti za deklariranje drugih tipova podataka.

Od verzije Oracle baze podataka s Oracle 12c uvodi se JSON format podataka. Sa JSON formatom sve podatke je vrlo lako moguće pohranjivati te nije potrebno stvarati nove tipove. JSON (engl. *JavaScript Object Notation*) je format koji sadrži tekstualne podatke, čitljiv je i razumljiv računalima i ljudima te služi za pohranu i prijenos različitih podataka. JSON se najčešće koristi za slanje podataka s poslužitelja na Web stranicu. Ekstenzija datoteke koja sadrži podatke u JSON formatu je (.json). JSON format polako zamjenjuje XML (engl. *Extensible Markup Language*) zbog svojih prednosti: kraći i razumljiv kod, jednostavniji hijerarhijski poredak. Prema izvorima s interneta [2] JSON je brži i manje zahtjevan u pogledu količine koda koju je potrebno izraditi ukoliko je potrebna određena aktivnost. JSON se češće koristi za prijenos podataka između servera i web preglednika dok se XML koristi za pohranu informacija na serverskoj strani te najvažnija prednosti je to što se JSON u odnosu XML provodi kroz standardne JavaScript funkcije, dok XML kroz XML prevoditelj.

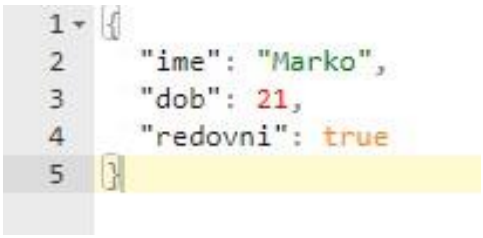
2. JSON (JavaScript Object Notation)

Sintaksa JSON formata je izvedena iz JavaScripta programskog jezika, ima podršku i u ostalim programskih jezicima zbog svoje jednostavnosti i praktičnosti.

Tipovi podataka u JSON formatu:

- Broj (engl. *number)*
- Tekst (engl. *string*)
- Logički tip podataka (engl. *boolean)* - vrijednosti točno ili netočno (engl. *true/false*)
- Polje(engl. *array)* - sortirano polje podataka, označava se uglatom zagradom [], vrijednosti se razdvajaju zarezom.
- *Null* vrijednost - predstavlja praznu vrijednost
- Objekt – iako je JSON format sam po sebi objekt on u samome sebi može sadržavati još jedan objekt. Za razliku od polja, objekt je nesortirano polje podataka, podaci se unose na način naziv/vrijednost te dvotočka dijeli naziv i vrijednost. Označava se vitičastim zagradama {}.

Slika 2.1. prikazuje način na koji je definiran i izrađen objekt u JSON formatu.

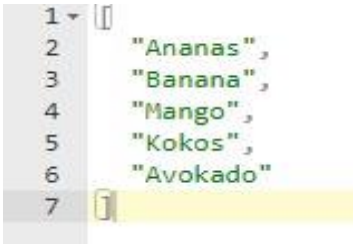


Slika 2.1: Prikaz objekta u JSON formatu

Za pohranu objekta koriste se vitičaste zagrade na početku i na kraju. Unutar njega nalazi se svojstvo "ime" koje je tip podatka tekst (engl. *string*) te se za unos teksta moraju koristiti dvostruki navodnici. U daljnjem prikazu slike 1 "dob" koja je broj (engl. *number)* ne smije biti unutar dvostrukih navodnika te vrijednost "redovni" koja je logički tip podataka (engl. *boolean)* također se ne smije nalaziti unutar dvostrukih navodnika.

JSON se definira na poredanoj listi vrijednosti i na skupu objekata. Te dvije strukture se koriste na isti način i u ostalim programskim jezicima, ali predstavljaju različite oblike podatkovnih struktura. Skup objekta se u ostalim programskim jezicima naziva: zapis (engl. *record),* struktura, rječnik (engl. *dictionary),* asocijativno polje (engl. *associative array*).

Na slici 2.2 prikazan je poredan niz vrijednosti koja počinje s lijevom uglatom zagradom, popisom vrijednosti te završava s desnom uglatom zagradom. Vrijednosti se odvajaju zarezom.



Slika 2.2: Polje s nekoliko pohranjenih vrijednosti

Na slici 2.3 prikazan je skup objekta s nekoliko vrijednosti koji počinje s lijevom uglatom zagradom te nakon nje slijedi lijeva vitičasta zagrada, popis *naziva:vrijednost* te završava s desnom vitičastom zagradom i uglatom zagradom.

Slika 2.3: Skup objekta s nekoliko vrijednosti

JSON je podskup notacija objektnih zapisa unutar JavaScript programskog jezika. Budući da se koristi za predstavljanje JavaScript objekata, JSON obično služi kao jezik za razmjenu podataka. JSON se često može koristiti bez potrebe za raščlanjivanjem ili serijalizacijom(koristi se za identifikaciju i generiranje kodova koji imaju jedinstvenu identifikaciju). To je tekstualni način predstavljanja JavaScript objekata, polja i različitih vrsta podataka. Iako je JSON definiran unutar JavaScript konteksta, on je zapravo neovisan format podataka.

Tekstualni JSON podaci kodiraju se pomoću *Unicode* UTF-8 ili UTF-16. Moguće je korištenje tekstualnih podataka koji su pohranjeni kao skup znakova koji nisu *Unicode* kao da su JSON podaci, ali Oracle baza podataka automatski prebacuje skupinu tih znakova u UTF-8 skupinu prilikom obrade podataka.

Prema izvoru [3] postoji i binarni JSON oblik dokumenta koji se naziva BSON (engl. *binary JSON*). Glavna razlika između JSON i BSON je u tome što se kod binarnog JSON dokumenta pohranjuju u binarnom obliku. Zapisi BSON dokumenta su kraći od JSON dokumenta, međutim JSON dokumentu se brže pristupa.

3. KORIŠTENJE JSON UNUTAR BAZE PODATAKA

SQL (engl. *Structured Query Language*) i relacijske baze podataka pružaju razne mogućnosti i veliku fleksibilnost podrške za složenu analizu podataka i izvještavanje. Također SQL osigurava integritet, zaštitu podataka te kontrolu pristupa podacima. Oracle baza podataka pruža sve mogućnosti korištenja SQL programskog jezika i relacijskih baza podataka sa JSON podacima. Ti podaci se pohranjuje i njime se manipulira na isti način te s istim povjerenjem kao i bilo koja druga vrsta podataka unutar baze podataka.

Oracle baza podataka podržava JavaScript notaciju objekata (engl. *Object Notation*). Ti podaci sadrže značajke relacijske baze podataka koja uključuje transakcije, indeksiranje, postavljanje upita i pregled podataka. Oznaka JavaScript objekta definirana je u standardima ECMA-404 i ECMA-262. *ECMAScript* je opći standard za programske jezike koji se široko koristi u web preglednicima i web poslužiteljima.

3.1 JSON u Oracle bazi podataka

Izvor [7] govori o SODA (engl. *Simple Oracle Document Access)* koja služi za razvoj aplikacije bez poznavanja sheme te bez znanja vezanih uz relacijske baze podataka ili poznavanje jezika poput SQL ili PL/SQL programskog jezika. SODA omogućava stvaranje i spremanje zbirki dokumenata u Oracle bazu podataka, mogućnost njihovog preuzimanja i postavljanja upita, bez potrebe da se zna kako su dokumenti pohranjeni u bazi podataka.

Podrška Oracle baze podataka za JSON osmišljena je tako da pruži najbolje usklađivanje između svijeta relacijskog pohranjivanja i postavljanja upita JSON podacima. Na taj način se omogućava relacijskim i JSON upitima da zajedno dobro funkcioniraju.

3.1.1 Implementacije SODA

SODA ima nekoliko vrsta implementacija:

- SODA for REST - zahtjevi za prijenos reprezentativnog stanja (REST) izvode operacije prikupljanja i dokumentiranja pomoću bilo kojeg programskog jezika koji može slati HTTP zahtjeve
- SODA for Java - Java sučelje predstavlja baze podataka, zbirke i dokumente.
- SODA for PL/SQL - PL/SQL tipovi objekata predstavljaju zbirke i dokumente.
- SODA for C - Oracle Call Interface (OCI) predstavlja zbirke i dokumente.

SODA for PL/SQL predstavlja određene zbirke i dokumente te podatke koji se nalaze u njima.

3.1.2 Pregled JSON formata u Oracle bazi podataka

XML i JSON podaci se mogu koristiti na sličan način unutar Oracle baze podataka, te dvije vrste podataka se mogu pohraniti, indeksirati i ispitivati bez potrebe za shemom koja definira određene podatke.

JSON podaci su često pohranjeni u NoSQL bazama podataka, poput Oracle NoSQL baze, Oracle Berkeley DB i MongoDB. Oracle NoSQL, Berkeley DB i MongoDB omogućuju pohranu i pronalazak podataka koji se temelje na bilo kojoj shemi, ali ne nude rigorozne modele relacijskih baza podataka. MongoDB sadrži model podataka koji podržava JSON te je on vrlo jednostavan za korištenje. MongoDB sadrži puno ugrađenih (engl. *built-in*) funkcija kao što je automatsko prikazivanje grešaka, horizontalno skaliranje i dodjeljivanja lokaciji gdje su određeni podatci pohranjeni. NoSQL pruža mehanizam za pohranu i pronalaženje podataka koji se modelira na druge načine osim standardnim tabličnim odnosima koji se koriste u relacijskim modelima baza podataka. Da bi se nadoknadio nedostatak relacijske baze, ponekad se relacijska baza koristi paralelno s NoSQL bazom podataka. Ukoliko se koristi JSON za pohranu podataka unutar NoSQL baze podataka, mora se osigurati integritet podataka.

Oracle baza podataka može se koristiti SQL za spajanje JSON podataka s relacijskim podacima. Same te podatke moguće je prikazivati na relacijski način, čineći ih dostupnim za relacijske procese i alate. JSON podatke moguće je tražiti i izvan Oracle baze podataka ukoliko su pohranjeni u vanjskoj tablici.

3.1.3 JSON stupci u tablicama baza podataka

JSON stupci u Oracle tablicama baza podataka ne postavlja ograničenja na tablice koje se mogu koristiti za pohranu JSON dokumenata. Stupac koji sadrži JSON dokumente može postojati s bilo kojom drugom vrstom podataka unutar određene baze podataka. Tablica također može imati više stupaca koji sadrže JSON dokumente. Ukoliko se koristi Oracle baza podataka, JSON služi kao spremište dokumenata, a tablice koje sadrže JSON stupce obično imaju i nekoliko stupaca za održavanje koji nisu u JSON formatu.

Ako se koristi JSON za dodavanje fleksibilnosti primarno relacijskoj bazi podataka, tada neke tablice vjerojatno imaju i stupac za JSON dokumente, koji se koriste za upravljanje podacima aplikacije koji ne pripadaju izravno relacijskom modelu.

3.1.4 Korištenje SQL programskog jezika sa JSON podacima

SQL programski jezik može pristupiti JSON podacima pohranjenim u Oracle bazi podataka korištenjem specijaliziranih funkcija i uvjeta ili korištenjem notacije. Većina SQL funkcija pripada SQL/JSON standardu ali postoji i nekoliko funkcija koje su predefinirane specifično za Oracle bazu podataka.

Funkcije koje služe za generiranje SQL/JSON upita:

- *JSON_OBJECT*,
- *JSON_ARRAY*,
- *JSON_OBJECTAGG*,
- *JSON_ARRAYAGG*

Navedene SQL funkcije prikupljaju podatke za kreiranje JSON podataka. Isto vrijedi i za Oracle SQL agregatnu funkciju *JSON_DATAGUIDE*, ali JSON podaci koje proizvodi su vodič kroz podatke pomoću kojih se otkrivaju informacije o strukturi i sadržaju JSON podataka u bazi podataka.

3.1.5 Korištenje PL/SQL programskog jezika sa JSON podacima

Moguće je koristiti SQL kod, uključujući kod koji pristupa JSON podacima unutar PL/SQL koda. Neke od sljedećih SQL/JSON funkcija i uvjeta također su dostupni kao ugrađene funkcije. Za razliku od Oracle SQL koji nema tip podataka *BOOLEAN*, PL/SQL programski jezik sadrži *BOOLEAN* tip podatka. Postoje i različiti PL/SQL tipovi objekata za JSON koji se koristite za izgradnju i manipulaciju podacima u JSON memoriji, ti pohranjeni podaci se mogu pregledavati, izmjenjivati i serijalizirati natrag u tekstualne JSON podatke.

3.1.6 JSON sintaksa i podaci koji se prikazuju

Neke implementacije unutar JSON formata, uključujući implementaciju Oracle baze podataka, podržavaju uporabu nenavedenih i jednostruko citiranih naziva svojstva. Niz u JSON formatu sastoji se od *Unicode* znakova, s kosom crtom (engl. *backslash*) (\). JSON broj predstavljen je u decimalnom zapisu te može sadržavati decimalni eksponent. Svojstvo objekta obično se naziva polje te sam redoslijed nije važan između članova objekata.

Postoje različiti tipovi podataka unutar JSON, sve vrijednosti osim objekta i polja su skalarne vrijednosti. Unutar JSON objekta ime polja ne mora nužno biti jedinstveno (engl. *unique*). SQL/JSON koristi uvijek samo jednog od članova objekata koji imaju dano ime polja. Svi ostali članovi s istim imenom se ignoriraju.

4. STVARANJE I POHRANA JSON PODATAKA U STUPCE UNUTAR BAZE PODATAKA

Pomoću SQL/JSON uvjeta pohranjuje se JSON podaci u Oracle bazu podataka pomoću stupaca čiji su tipovi podataka: *VARCHAR2, CLOB, BLOB*. Ovisno o veličini JSON podataka koje je potrebno pohraniti, odabire se jedan od navedenih tipova podataka. Kod pohrane *VARCHAR2* korisnik mora biti siguran da JSON dokument ne prelazi 4000 bajta (bajt je jedinica mjere u binarnom brojevnom sustavu; jedan bajt se sastoji od 8 bitova). Postoji mogućnost korištenja *Oracle Exadata* s *VARCHAR2* (4000).

Kod pohrane JSON dokument koji sadrži više od 4000 bajtova koristi se *VARCHAR2* (32767). To znači da se uporaba *VARCHAR2* (32767) odnosi samo na dokumente koju su veći od 3500 bajtova.

JSON podaci se u bazu podataka pohranjuju koristeći standardne SQL tipove podataka što znači da se može manipulirati JSON podacima kao što bi se manipulirali bilo kojim drugim podacima tih vrsta. Pohranjivanjem JSON podataka pomoću standardnih vrsta omogućava lakše korištenje i rad s tablicama. BLOB (binarnih velikih objekata) ili CLOB (veliki objekt znakova) se upotrebljava ukoliko dokument sadrži više od 32767 bajtova.

4.1 Upotreba LOB pohrane za JSON podatke

Oracle preporučuje uporabu BLOB formata. Prilikom korištenja LOB formata poželjno je koristiti *SecureFiles*. *SecureFiles* služi za uklanjanje razlike između strukturirane i nestrukturirane pohrane sadržaja te se pomoću njega pohranjuju različiti sadržaj. Potrebno je razmotriti i upotrebu *Oracle Advanced Compression* kako bi se smanjio prostor za pohranu potreban za JSON podatke. Loša strana korištenja BLOB pohrane podataka u odnosu na CLOB za JSON ili bilo koju drugu vrstu podataka je to što je ponekad zahtjevnije za raditi s BLOB sadržajem pomoću alata naredbenog retka kao što je npr. *SQL*Plus*.

4.2 Kreiranje tablica sa JSON stupcima

Kod umetanja podataka u stupce mora se koristiti SQL ograničenje, a to je *IS JSON* kako bi se osiguralo da umetnuti podaci sadrže JSON vrijednosti, a ujedno od strane Oracle je preporučeno korištenje ograničenja. SQL/JSON uvjeti *IS JSON* i *IS NOT JSON* vraćaju točnu ili netočnu tvrdnju (engl. *true/false*) za bilo koju SQL vrijednost koja nije *NULL*. Ukoliko je vrijednost *NULL*, povratna informacija neće biti niti točna niti netočna. Ukoliko se koristi ograničenje provjere to može smanjiti performanse prilikom umetanja podataka. Ukoliko je korisnik siguran da aplikacija u određeni stupac ubacuje samo dobro oblikovane JSON podatke, postoji mogućnost onemogućavanja ograničenja provjere.

U slučaju prikazivanja podataka postoje određeni kriteriji od kojih bilo koji treba biti zadovoljen kako bi se smatralo da su to stvarno JSON podaci.

Kriteriji su:

• osnovni podaci sadrže ograničenje provjere JSON.

• stupac je rezultat uporabne SQL/JSON funkcije *JSON_QUERY*

• stupac je rezultat upotrebe funkcije JSON generiranja, kao npr. *JSON_OBJECT*

• stupac je rezultat upotrebe SQL funkcije tretiranja s ključnim riječima *IS JSON*

Ukoliko se utvrdi da je to JSON podatak i ako se ograničenje kasnije ukloni stupac ostaje naveden u prikazima, to znači da će i podaci iz stupca kojemu su uklonjena ograničenja biti vidljiv u bazi podataka te će biti moguć prikaz njegovih podataka.

Ukoliko se argument ne može procijeniti je li ispravno oblikovan, ukoliko se tijekom raščlanjivanja (dijeljenja podataka na više manjih segmenata prilikom dohvaćanja podataka) dogodi pogreška, smatra se da podaci nisu pravilno oblikovani te onda *IS JSON* vraća netočno. Vrlo je važno da se svi podaci prije pohrane u bazu podataka provjere, te ukoliko su oni dobro oblikovani znači da su sintaksno ispravni. JSON podaci mogu se oblikovati u dva značenja, strogom i labavom sintaksom.

4.2.1 Stroga i labava JSON sintaksa

Izvor [8] prikazuje zadanu (engl. *default*) Oracle sintaksa za JSON je labava. To ukratko znači da Oracle sintaksa za JSON podržava sintaksu JavaScripta za polja objekata. Polja objekata koji sadrže tipove podataka *BOOLEAN* i *NULL* vrijednosti ne razlikuju mala i velika slova te je moguće koristiti brojeve, razmake i moguće je izbjegavati *Unicode* znakove. Polja koja se koriste u JavaScript notaciji objekti mogu, ali ne trebaju biti zatvoreni dvostrukim navodnicima. Umjesto dvostrukih navodnika mogu se upotrijebiti jednostruku navodnici(').

U JavaScript implementaciju moguće je korištenje sljedećeg:

• korištenje velikih i malih slova za ključne riječi *true*, *false* i *null* (npr. truE, NuLL, TRue, falSE).

• Dodatan zarez (,) nakon posljednjeg elementa niza ili posljednjeg člana objekta.

• Brojevi s jednom ili više početnih nula (npr. 0021.2)

• Razlomljeni brojevi kojima nedostaje 0 ispred decimalne točke (npr. .13 umjesto 0.13).

• Brojevi bez razlomljenog dijela nakon decimalne točke (npr. 215. ili 3.e29).

• Znak plus (+) ispred broja, što znači da broj nije negativan.

U labavoj JSON sintaksi, polje objekata koje nije navedeno ne može sadržavati bilo koji *Unicode* znak, osim razmaka i JSON strukturiranih znakova [], {}, (:), (,).

Stroga ili labava sintaksa bitna je samo za SQL/JSON uvjete *IS JSON* i *IS NOT JSON*. Sve ostale funkcije i uvjeti koriste labavu sintaksu za tumačenje unosa jesu li podaci koji se unose u bazu podataka ispravni te koriste strogu sintaksu prilikom vraćanja izlaza. Navedena stroga sintaksa koja je oblikovana prema JSON standardu treba sadržavati dodatni *STRICT* izraz *IS JSON* ili *IS NOT JSON*. Za pravilno umetanje podataka u stupce te da su svi podaci pravilno oblikovani prema JSON standardu potrebno je koristiti *STRICT* izraz.

4.2.2 Jedinstveni naspram dupliciranih polja u JSON objektima

Nazivi polja ne moraju biti jedinstveni, ali moguće je odrediti da se za određeni JSON podaci smatraju dobro oblikovanim samo ako niti jedan od njegovih objekata nema dvostruka imena polja. U pravilu, JSON standard ne određuje moraju li nazivi polja biti jedinstveni za zadani JSON objekt. Znači da dobro oblikovani JSON objekt može imati više članova koji imaju isto ime polja. Da bi se osiguralo da se nazivi polja ne ponavljaju koristi se *WITH UNIQUE KEYS* / jedinstvenih ključeva. Ako se ne navede jedinstveni ključevi ili ako se koriste ključne riječi *WITHOUT UNIQUE KEYS* / bez jedinstvenih ključeva, objekti mogu imati dvostruka imena polja i dalje se smatrati dobro oblikovanim. Ovisno o tome koristi li se stroga ili labava sintaksa postoji mogućnosti dupliciranja naziva polja u JSON podacima.

5. PREGLED, UMETANJE, AŽURIRANJE I UČITAVANJE JSON PODATAKA

Za umetanje ili ažuriranje JSON podataka u Oracle bazi podataka postoji mogućnost korištenja API (engl. *Application Programming Interface*) baze podataka. Moguće je izravno raditi sa JSON podacima koji se nalaze u datotekama datotečnog sustava.

JSON podaci se pohranjuju korištenjem standardnih SQL tipova podataka. Svi standardni API baze podataka koriste se za umetanje ili ažuriranje *VARCHAR2* i *LOB* stupaca (stupac velikih objekata). Za ovu vrstu API podataka JSON format je ništa drugo nego pohranjeni niz znakova. Stupci koji sadrže tipove podataka *VARCHAR2*, *BLOB* ili *CLOB* koji sadrže JSON dokument ne razlikuju se od rada s bilo kojim drugim stupcem toga tipa.

Ukoliko se koristi operacija ažuriranja nad dokumentom koji sadrži JSON podatke, cijeli dokument se mora mijenjati. Moguće je napraviti male izmjene na dokumentu, ali kad je potrebno spremiti promjene na disk zapisuje se cijeli dokument neovisno o tome o kojoj veličini izmjene se radi.

Ukoliko se koristi alat naredbenog retka, kao npr. *SQL*Plus* za umetanje podataka u JSON stupac podatkovnog tipa BLOB ili za ažuriranje određenih podataka, tada se mora ispravno pretvoriti JSON podaci u binarni format.

Postoji mogućnost korištenja vanjskih tablica koje se ne nalaze unutar Oracle baze podataka kako bi se olakšao rad i pristup JSON dokumentima koji su pohranjeni kao zasebna datoteka. Vanjske tablice JSON dokumenta se mogu koristiti za izravno postavljanje upita u datotekama. Navedena mogućnost je korisna ukoliko se treba obraditi podatak iz svih datoteka u jednoj operaciji. Ukoliko se mora postaviti više upita prema dokumentima te ako različiti upiti utječu na različite podatke iz redaka. Poželjno je kopirati podatke iz vanjskih tablica u obične tablice zbog mogućnosti lošijih performansi. Nakon što se JSON podaci učitaju u stupac obične tablice, ti podaci se mogu indeksirati.

Moguće je učitati vanjske JSON podatke na način da se tablica koja sadrži JSON podatke unese u baza podataka te se na taj način može stvoriti sadržaja te datoteke ispisan u JSON formatu. JSON format datoteke kompatibilan je s izvornim formatom koji proizvodi uobičajene baze podataka NoSQL, uključujući i samu Oracle NoSQL bazu podataka. Svaki redak datoteke sadrži jedan JSON dokument kao JSON objekt. Tu vrstu vanjske tablice moguće je učitati izravno, ili ukoliko su potrebne što bolje performanse, može se učitati u običnu tablicu baze podataka iz podataka u vanjskoj tablici.

JSON API je sučelje koja radi s HTTP. Pomoću API se skicira izgled kako klijenti trebaju zatražiti ili urediti podatke s poslužitelja i kako poslužitelj treba odgovoriti na navedene zahtjeve. Vrlo je praktičan zbog oblikovanja JSON odgovora tj. povratnih informacija, glavni cilj mu je povećanje efikasnosti i produktivnosti. JSON API je vrlo koristan zbog pred memorije u kojoj se pohranjuju često korišteni zahtjevi te se na taj način lako zaobilaze suvišni zahtjevi prema poslužitelju. Osim običnih načina umetanja, ažuriranja i učitavanja podataka, moguće je koristiti i API za jednostavan pristup *SODA*. *SODA* je dizajnirana za razvoj aplikacija bez sheme i bez poznavanja značajki ili jezika relacijske baze podataka. Njena najveća pogodnost je to što je moguće stvarati i pohranjivati zbirke podataka bilo koje vrste, a njihovo preuzimanje i postavljanje upita moguće je napraviti bez poznavanja procesa pohrane dokumenata u bazi podataka.

6. UPITI SA JSON PODACIMA

Podaci [9] opisuju upite nad JSON podacima moguće je postavljati pomoću točkaste notacije (engl. *dot notation*), ili ukoliko je potrebno koristiti ugrađene funkcije. Prilikom korištenja ugrađenih funkcija koriste se SQL/JSON funkcije i uvjeti. Moguće je stvoriti i dohvatiti podatke koji sažimaju strukture i podatke iz JSON dokumentima. JSON podaci pohranjuju se u bazu podataka koristeći standardne tipove podataka.

Za postavljanje upita o određenim JSON poljima ili za mapiranje određenih JSON polja u SQL stupcima, može se koristiti SQL/JSON jezik putanje. Složeniji izrazi putanje mogu sadržavati određene filtere i indekse polja. Oracle nudi dva načina na koje je moguće postavljati upite za JSON sadržaj;

• Sintaksa označavanja točkom – pseudonim tablica, iza nje slijedi naziv stupca koji sadržava JSON podatke, nakon čega slijedi jedan ili više naziva polja (obavezno se odvaja točkom (.)). Ova sintaksa je osmišljena tako da je jednostavna za upotrebu i da vraća JSON vrijednost kada god je to moguće.

• SQL/JSON funkcije i uvjeti – za razliku od sintakse oznake točkom, funkcije i uvjeti nude puno više mogućnosti i fleksibilnosti, moguće ju je koristiti za stvaranje, postavljanje upita i upravljanje samim JSON podacima koji su pohranjeni u Oracle bazu podataka.

Budući da je jezik putanje (engl. *path language*) dio jezika upita (engl. *query language*) podaci nemaju fiksnu shemu. Sama shema se definira prilikom izrade upita, tako što se navodi dana putanja. To se razlikuje od standardnog SQL programskog jezika koji definira shemu (skup redaka i stupaca tablice).

Oracle SQL uvjet *JSON_EQUAL* ne prihvaća argument putanje-izraza. Uvjet *JSON_EQUAL* uspoređuje dvije JSON vrijednosti i vraća točno ukoliko su te vrijednosti jednake, ili netočno ukoliko vrijednosti nisu jednake. Moguće je generirati i postavljati upit za JSON podatkovni vodič koji će pomoći u razvoju izraza za navigaciju JSON sadržajem. Vodič s podacima može prikazivati način na koji se može bolje razumjeti struktura i podaci koji se nalaze u JSON dokumentima. Često se ti podaci mogu automatski ažurirati zbog praćenja novih dokumenata koji se dodaju u JSON dokument.

6.1 Pristup JSON podacima

Točkasta notacija je dizajniran za jednostavnu, opću upotrebu i uobičajene slučajeve uporabe u kombinaciji sa JSON podacima. Korištenjem JSON podataka prilikom svakog upita pomoću točkaste notacije, pokušava se vratiti JSON vrijednost kada god je to moguće. Povratna vrijednost koja se prikazuje za upite točkastom notacijom, uvijek je polje vrijednosnog tipa *VARCHAR2*.

Sadržaj niza ovisno o ciljanim JSON podacima dijeli se (po uvjetima):

• jedna JSON vrijednost, tada je ta vrijednost sadržaja niza, bilo da se radi o JSON skalarnoj vrijednosti, objektu ili nizu.

• više JSON vrijednosti, tada je sadržaj niza JSON niz čiji su elementi te vrijednosti.

Sadržaj niza ovisno o ciljanim podacima se znatno razlikuje od klasičnih SQL/JSON funkcija *JSON_VALUE* i *JSON_QUERY* koje se inače koriste za kompleksnije upite. Ukoliko JSON podaci ne odgovaraju poslanom upitu, moguć je prikaz greške ili povratna vrijednost *NULL*.

Sadržaj niza prihvaća izbornu klauzulu za navođenje vrste podataka povratnih vrijednosti (*RETURNING*) bez obzira na to hoće li više vrijednosti prelomiti niz, kako općenito postupati s pogreškama i kako postupati s nedostajućim JSON formatom polja.

Svako polje *JSON_FIELD* mora sadržavati ispravan SQL identifikator, ukoliko JSON polje ne sadržava ispravan identifikator i ako nije postavljeno ograničenje postoji mogućnost da će se pojaviti pogreška prilikom pisanja upita. Za JSON upite pomoću točkaste notacije, za razliku od općenito slučaja kod SQL programskog jezika, identifikator koji se nalazi nakon stupca, osjetljiv je na velika i mala slova, to jest ponaša se kao da je citiran. To znači da se imena JSON polja mogu koristit kao identifikatori bez navođenja. Umjesto *jcolumn.prijatelji* može se napisati *jcolumn.prijatelji*. Također to znači da ukoliko se JSON objekt imenuje velikim slovima, na primjer *PRIJATELJI*, tada je potrebno napisati *jcolumn.PRIJATELJI*, a ne *jsolumn.prijatelji*. Usklađivanje JSON izraza točkaste notacije s točkom prema JSON podacima isto je kao i podudaranje izraza putanje SQL/JSON.

6.2 SQL/JSON izraz putanje

Oracle baza podataka omogućava SQL pristup JSON podacima pomoću izraza putanje SQL/JSON. Kada su JSON podaci pohranjeni u bazi podataka, moguće je postali upit pomoću izraza putanje koji su slični *XQuery* ili *XPath* izrazima za XML podatke. Slično načinu na koji SQL/XML dopušta SQL programskom jeziku pristup XML podacima pomoću *Xquery* izraza.

SQL/JSON izrazi putanje imaju jednostavnu sintaksu, izraz bira nula ili više JSON vrijednosti koje mu odgovaraju. Ukoliko se ni jedna vrijednost ne postoji, tada funkcija *JSON_VALUE* vraća SQL *NULL*.

SQL/JSON funkcija *JSON_QUERY* vraća sve odgovarajuće vrijednosti, što znači da može vratiti više vrijednosti. U svim slučajevima, podudaranje izraza puta pokušava zauzvrat odgovoriti svakom koraku izraza putanje. Ako podudaranje bilo kojeg koraka ne uspije, ne pokušava se uskladiti sljedeći korak, a podudaranje izraza putanje ne uspijeva. Ako podudaranje svakog koraka uspije, tada uspijeva podudaranje izraza putanje.

Izrazi putanje mogu koristiti zamjenske znakove i raspone polja. Podudaranje razlikuje velika i mala slova. Izraz putanje se podudara s podacima, a odgovarajući podaci se obrađuju pomoću određenih SQL/JSON funkcija ili uvjeta. Taj proces podudaranja moguće je zamisliti u smislu izraza putanje koji vraća usklađene podatke u funkciju ili stanje.

Osnovna sintaksa izraza putanje SQL/JSON sastoji se od stavke konteksta nakon koje slijedi nula ili više koraka objekata niza, ovisno o prirodi stavke konteksta, nakon čega po izboru slijedi korak funkcije. Izborni izraz filtera pristupa samo kada se izraz putanje koristi u SQL uvjetu *JSON_EXISTS*. Usklađivanje podataka s izrazima putanje SQL/JSON razlikuje velika i mala slova. Osnovni izraz putanje SQL/JSON apsolutno je jednostavan izraz putanje, iza kojega slijedi izborni izraz filtera (engl. *optional filter expression*).

Apsolutni izraz putanje započinje znakom dolara ($), koji predstavlja stavku konteksta izraza putanje, to jest JSON podatke koje treba uskladiti. Znak dolar ($) slijedi iza nula ili više koraka na putu.

Svaki korak može biti korak objekta ili korak niza, ovisno o tome predstavlja li stavka konteksta JSON objekt ili JSON niz. Posljednji korak jednostavnog izraza putanje može biti pojedinačni, izborni korak funkcije. Korak objekta je točka (.), a ona se ponekad čita kao točka (engl. *dot*) nakon čega slijedi naziv polja objekta ili zamjenska zvjezdica (*) koja označava vrijednost svih polja. Naziv samog polja može biti prazan ali mora biti napisan kao " ".

Korak niza označava lijevu zagradu ([) iza koje slijedi zamjenska zvjezdica (*), koja označava sve elemente niza, ili jedan ili više specifičnih indeksa niza ili specifikacija raspona odvojenih zarezima, iza kojih slijedi desna zagrada (]). Prilikom korištenja indeksa ili specifikacije raspona, elementi niza koje oni zajedno navode moraju biti navedeni ulaznim redoslijedom, bez ponavljanja ili se u suprotnom pojavljuje greška za vrijeme kompajliranja.

Korak s jednom funkcijom nije obavezan, ako je prisutan, to je posljednji korak apsolutnog jednostavnog izraza puta, pojavljuje se neposredno prije izraza filtera ukoliko postoji. Izraz filtera je upitnik (?) iza kojega slijedi uvjet koji se nalazi u zagradama (()). Filter je zadovoljen ako je ispunjen njegov uvjet, odnosno vraća vrijednost točno.

Relativno jednostavan izraz puta je predznak (@) iza kojeg slijedi nula ili više koraka puta. Znak (@) predstavlja trenutnu stavku filtera izraza putanje, to jest JSON podatke koji odgovaraju dijelu izraza putanje koji prethodi filteru. Jednostavan izraz puta uspoređuje se s trenutnom stavkom filtera na isti način na koji se izraz putanje podudara sa stavkom konteksta.

Osnovna sintaksa izraza putanje SQL/JSON labava je kako bi omogućila implicitno omatanje i odmotavanje niza. To znači da se ne mora mijenjati izraz puta u kodu ukoliko se podaci razvijaju kako bi zamijenili JSON vrijednosti s nizom takvih vrijednosti ili obrnuto.

6.2.1 Metoda stavki izraza SQL/JSON putanje

Metoda stavke primjenjuje se na JSON podatke koji su ciljani ostatkom izraza putanje koji je završio tom metodom, a ona služi za transformaciju podataka. Određeni SQL uvjeti i funkcije kojima se prosljeđuju izrazi putanje koriste transformirane podatke umjesto cijelih podataka. U nekim slučajevima primjena metode stavke djeluje kao filter, uklanjajući ciljane podatke iz skupa rezultata. Za određene ciljane JSON vrijednosti vrijedi da se tumače kao vrijednost danog SQL tipa podatka. Metode pretvorbe tipa podataka sa *samo* (engl. *only*) u svom nazivu je isti kao i odgovarajuće metode s imenima bez *samo*, osim što pretvaraju samo JSON vrijednosti danog tipa u povezanu SQL vrstu podataka. Metode bez *samo* u nazivu dopuštaju pretvaranje, kad god je to moguće te nije bitno u koju JSON vrijednost se pretvara zadani SQL tip podatka.

Neke od metoda su:

• *abs* () – apsolutna vrijednost za željeni JSON broj

• *boolean* () – interpretacija ciljane JSON vrijednosti

• *ceiling* () – zaokružuje JSON broj na najbliži cijeli broj

• *lenght* () – prikazuje broj znakova za određeni JSON niz.

ISO standard 8601 opisuje način međunarodnog prihvaćanja datuma i vremena. Prilikom pohrane datuma u JSON format postoje dva načina unosa datuma:

• Samo datum: YYYY-MM-DD, na primjer 2021-05-21

• Datum s vremenom: YYYY-MM-DD hh:mm:ss,

na primjer 2021-05-21 18:32:12

Značenje navedenih kratica:

• YYYY označuje godinu, zapisuje se u obliku četiri cijela broja

• MM označuje mjesec, zapis se u obliku dva cijela broja (00 do 12)

• DD označuje dan, zapisuje se u obliku dva cijela broja (00 do 31)

• hh označuje sate, mm označuje minute, ss označuje sekunde

6.3 Klauzule korištene u SQL/JSON upitima, funkcijama i uvjetima

SQL/JSON funkcije koje sadrže upit *JSON_VALUE* i *JSON_QUERY* prihvaćaju izbornu *RETURNING* klauzulu koja specificira tip podataka vrijednosti koju funkcija vraća. Za *JSON_VALUE* moguće je korištenje bilo koja od navedenih SQL tipova podataka u klauzuli *RETURNING*: *VARCHAR2, NUMBER, DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, SDO_GEOMETRY* i *CLOB*. Tip podatka *SDO_GEOMETRY* pruža mogućnost korištenja *JSON_VALUE* s podacima *GeoJSON* koji je format za kodiranje geografskih podataka u JSON podatke.

Klauzula *RETURNING* prihvaća dvije izborne ključne riječi; *PRETTY* i *ASCII*. Ukoliko su obje izborne riječi odabrane, tada *PRETTY* dolazi prije *ASCII*. *ASCII* se koristi samo za SQL/JSON funkciju *JSON_VALUE* i *JSON_QUERY*, a *PRETTY* je dopušten za korištenje samo za *JSON_QUERY*. Ključna riječ *PRETTY* služi za ispisivanje podataka umetanjem znakova novog retka i uvlačenjem. Učinak ključne riječi *ASCII* je automatsko izbjegavanje svih znakova *Unicode* koji nisu *ASCII* u vraćenim podacima koristeći standardne *ASCII Unicode* izlazne sekvence.

SQL/JSON upitne funkcije *JSON_QUERY* i *JSON_TABLE* prihvaćaju izbornu klauzulu omotača (engl. *wrapper*). Omotač služi za oblikovanje vrijednosti koju *JSON_QUERY* vraća ili se koristi za podatke u stupcu *JSON_TABLE*.

Klauzula omotača ima sljedeće oblike:

• *WITH WRAPPER* – koriste se vrijednosti niza koje predstavljaju JSON niz koji sadrži sve JSON vrijednosti koje odgovaraju izrazu putanje.

• *WITHOUT WRAPPER* – koriste vrijednosti niza koja predstavlja pojedinačni JSON objekt ili niz koji odgovara izrazu putanje. Ukoliko dođe do pogreške on ju pojasni.

• *WITH CONDITIONAL WRAPPER* - koristite vrijednost niza koja predstavlja sve JSON vrijednosti koje odgovaraju izrazu putanje.

U Oracle bazi podataka je zadano ponašanje bez omotača. Opcijsku ključnu riječ *UNCONDITIONAL* se može dodati odmah nakon ključne riječi *WRAPPER*. *WITH WRAPPER* i *WITH CONDITIONAL WRAPPER* imaju isto značenje. Opcijsku ključnu riječ *ARRAY* moguće je dodati neposredno prije ključne riječi *WRAPPER*. *WRAPPER* i *ARRAY WRAPPER* imaju isto značenje.

6.3.1 Klauzula pogreške za SQL/JSON funkcije i uvjete

Neke od SQL/JSON funkcija i uvjeta prihvaćaju izbornu klauzulu o pogrešci, koja točno specificira o kojoj pogrešci se radi. SQL funkcije i uvjeti izbjegavaju izazivati greške tijekom izvođenja - ukoliko su neki JSON podaci neispravni, *JSON_EXISTS* te *JSON_EQUAL* vraćaju netočne vrijednost dok *JSON_VALUE* vraća NULL vrijednost. U nekim slučajevima može se navesti klauzula pogreške koja ima veću ovlast od zadanog (engl. *default*) ponašanja. Postoji više različitih grešaka, ali svaka SQL/JSON funkcija i uvjet omogućavaju navođenje rukovanja pogreškama te podržava barem ponašanje *ERROR ON ERROR* prilikom prikaza pogreške.

Postoji više oblika opcijske klauzule nad pogreškama:

• *ERROR ON ERROR*– prikazuje pogrešku

• NULL ON ERROR– umjesto pogreške prikazuje *NULL* vrijednost

• *FALSE ON ERROR* – umjesto pogreške prikazuje *FALSE*, vrijedi samo za *JSON_EXISTS* i *JSON_EQUAL*

• *TRUE ON ERROR*– umjesto pogreške prikazuje *TRUE*, vrijedi samo za *JSON_EXISTS* i *JSON_EQUAL*

• *EMPTY OBJECT/ARRAY ON ERROR*– umjesto pogreške prikazuje prazan objekt ({}) / niz ([]).

NULL ON EMPTY se koristi u slučaju funkcionalnog indeksa stvorenog za izraz *JSON_VALUE*. Naredba nema utjecaja na to hoće li se indeks pokupiti ili kada, ali je učinkovita u dopuštanju indeksiranja nekih podataka koji inače ne bi bili jer nedostaje polje ciljano izrazom *JSON_VALUE*. Općenito se predlaže korištenje *ERROR ON ERROR* za upite koji popunjavaju indeks, tako da izraz putanje upita koji rezultira s više vrijednosti ili složenim vrijednostima izaziva pogrešku. No ponekad je jednostavno bolje preskočiti prikaz pogreške samo zato što nedostaje polje ciljano izrazom putanje ukoliko postoji potreba da se ti podaci indeksiraju.

6.4 SQL/JSON uvjet JSON_EXISTS

SQL/JSON uvjet *JSON_EXISTS* omogućuje korištenje izraza putanje SQL/JSON kao filtera redaka za odabir redaka na temelju sadržaja JSON dokumenta. Uvjet *JSON_EXISTS* može se koristiti u izrazu *CASE* ili *WHERE* klauzuli unutar *SELECT* izraza. Uvjet *JSON_EXISTS* provjerava postoji li određena vrijednost unutar JSON podataka. Ukoliko postoji tražena vrijednost, korisnik dobiva točnu povratnu informaciju, a ako tražena vrijednost ne postoji, korisnik dobiva povratnu informaciju netočno.

Ukoliko je potrebno, *JSON_EXISTS* je pogodan za kreiranje bitmape za upotrebu sa JSON podacima zato što vraća samo dvije vrijednosti. Nositelji pogrešaka (engl. *error handlers*) POGREŠKA NAD POGREŠKOM, FALSE ON ERROR i TRUE ON ERROR se primjenjuju za *JSON_EXISTS*. Najčešće do greške dolazi zbog lošeg oblikovanja podataka zbog korištenja labave sintakse. SQL/JSON uvjet *JSON_EXISTS* moguće je koristiti za jedan ili više izraza filtera za odabir dokumenta koji sadrže podudarane podatke. Filteri omogućavaju testiranje postojanih dokumenata koji imaju određena polja koja zadovoljavaju različite uvjete. SQL/JSON uvjet *JSON_EXISTS* vraća točno za dokumente koji sadrže podatke koji odgovaraju izrazu putanje SQL/JSON. Ukoliko izraz putanje sadrži filter, tada i ti podaci moraju zadovoljiti filter kako bi *JSON_EXISTS* vratio točno za dokument koji sadrži podatke. Filter se primjenjuje na putanju koja prethodi, a provjerava ima li:

• dani dokument neke podatke koji odgovaraju tom putu

• da li odgovarajući podaci zadovoljavaju filter.

Ako su oba uvjeta zadovoljena, onda *JSON_EXISTS* vraća vrijednost za dokument. Izraz puta koji neposredno prethodi filteru definira opseg uzroka koji se u njemu koriste.

SQL/JSON uvjet *JSON_EXISTS* može se promatrati kao poseban slučaj SQL/JSON funkcije *JSON_TABLE*. Ako se *JSON_EXISTS* koristi više puta ili se koristi u kombinaciji sa *JSON_VALUE* ili *JSON_QUERY*, za pristup istim podacima pozivanje *JSON_TABLE* predstavlja prednost u tome što su podaci raščlanjeni samo jednom. Zbog toga optimizator često automatski prepisuje više poziva *JSON_EXISTS, JSON_VALUE* i *JSON_QUERY*.

6.5 SQL/JSON funkcije JSON_VALUE

SQL/JSON funkcija *JSON_VALUE* odabire skalarnu vrijednost iz JSON podataka i vraća je u obliku SQL vrijednosti. Funkcija prima dva argumenta i podržava klauzule o pogreškama i povratnim informacijama. Prvi *JSON_VALUE* argument je SQL izraz koji vraća instancu skalarne SQL vrste podataka. Tip podataka koji se koristi s funkcijom *JSON_VALUE* može biti *VARCHAR2, BLOB* ili *CLOB*. Sami rezultat SQL izraza koristi se kao stavka konteksta za procjenu izraza putanje. Drugi argument *JSON_VALUE* funkcije je izraz putanje SQL/JSON iza kojega slijedi izborna klauzula *RETURNING, ON ERROR* i *ON EMPTY*. Izraz putanje mora ciljati jednu vrijednost u protivnome dolazi do pogreške. Zadano (engl. *default*) ponašanje prilikom pojave pogrešaka je *NULL ON ERROR*, što znači da ukoliko dođe do pogreške ne vraća se nikakva vrijednost odnosno pogreška se ne pojavljuje. Ukoliko korisnik želi imati na uvid ukoliko dođe do pogreške potrebno je koristiti *ERROR ON ERROR*.

JSON ima *BOOLEAN* vrijednosti, kada SQL/JSON funkcija *JSON_VALUE* procjenjuje izraz puta SQL/JSON i rezultat je JSON točno ili netočno, moguće je povratnu informaciju prikazati kao PL/SQL tip podatka *BOOLEAN*. SQL/JSON funkcija *JSON_VALUE* može se promatrati kao poseban slučaj funkcije *JSON_TABLE*. Postoji mogućnost korištenja *JSON_VALUE* više puta, ili se može koristiti u kombinaciji sa *JSON_EXISTS* ili *JSON_QUERY*. Obje funkcije se mogu prikazati pomoću *JSON_TABLE*.

Za pristup povezanim podacima povezivanje *JSON_TABLE* ima prednost u tome što se podaci rasčlanjuju samo jednom. SQL/JSON funkcija *JSON_TABLE* generalizira ostale SQL/JSON upite poput *JSON_VALUE*. Prilikom korištenja JSON *BOOLEAN* vrijednosti, *JSON_VALUE* se koristi implicitno, a rezultirajuće SQL vrijednosti se vraćaju kao *VARCHAR2* vrijednosti.

6.6 SQL/JSON funkcija JSON_QUERY

SQL/JSON funkcija *JSON_QUERY* odabire jednu ili više vrijednosti iz JSON podataka i vraća polje (*VARCHAR2, CLOB ili BLOB*) koji predstavlja JSON vrijednosti. Na taj način moguće je koristiti *JSON_QUERY* za dohvaćanje JSON dokumenata. Prvi argument *JSON_QUERY* je SQL izraz koji vraća instancu skalarne SQL vrste podataka. Drugi argument *JSON_QUERY* je izraz putanje SQL/JSON iza kojeg slijede izborne klauzule *RETURNING*, *WRAPPER*, *ON ERROR* i *ON EMPTY*. Izraz putanje može ciljati bilo koji broj JSON vrijednosti. U *RETURNING* klauzuli moguće je navesti tri tipa podataka. Vraćena vrijednost uvijek sadrži dobro oblikovane JSON podatke. To uključuje osiguravanje da se znakovi koji nisu *ASCII* u vrijednostima niza izbjegavaju prema potrebi. Korištenje ključnih riječi formata JSON nisu potrebni za *JSON_QUERY*. Klauzula omotača određuje oblik vraćene vrijednosti niza. Klauzula o pogrešci za *JSON_QUERY* može navesti *EMPTY ON ERROR*, što znači da se u slučaju pogreške vraća prazan niz ([]). SQL/JSON funkcija *JSON_QUERY* može se promatrati kao poseban slučaj funkcije *JSON_TABLE*.

6.7 SQL/JSON funkcija JSON_TABLE

SQL/JSON funkcija *JSON_TABLE* služi za prikaz JSON podataka unutar stupaca koji prikazuju različite vrste SQL podataka. Koristi se za mapiranje dijelova JSON dokumenta u retke i stupce novih tablica. Glavna uporaba *JSON_TABLE* funkcije je prikaz JSON podataka te na taj način se omogućava rad s podacima bez razmatranja sintakse izraza JSON putanje. Prvi argument za *JSON_TABLE* je SQL izraz. Rezultat vrednovanja izraza koristi se kao stavka konteksta za procjenu izraza putanje retka. Drugi argument *JSON_TABLE* je izraz putanje SQL/JSON nakon kojega slijedi izborna klauzula pogreške za rukovanje retkom i *COLUMNS* klauzula. Umjesto prvog argumenta i izraza putanje retka moguće je koristiti sintaksu oznaka točkama. Oznaka točkama označava tablicu ili stupac prikaza zajedno s jednostavnim putem do ciljanih JSON podataka.

Postoje dvije razine rukovanja pogreškama za *JSON_TABLE,* kojom odgovaraju dvije razine izraza putanje: redak i stupac. Kada je prisutan, rukovatelj pogreškom stupca nadjačava rukovanje pogreškom na razini retka. Zadani (engl. *default*) rukovoditelj pogreškama za obje razine je *NULL ON ERROR*. Obvezna klauzula *COLUMNS* definira stupce virtualne tablice koje će stvoriti *JSON_TABLE*.

Sastoji se od ključne riječi *COLUMNS* nakon čega slijede unosi zatvoreni u zagradi:

• najviše jedan unos u klauzuli *COLUMNS* može biti naziv stupca iza kojega slijedi *FOR ORDINALITY*. Specificira stupac generiranih brojeva redaka (SQL tip podataka *number*).

• redovita specifikacija stupca sastoji se od naziva stupca iza kojeg slijedi izborni skalarni tip podataka za stupac nakon čega slijedi opcionalna klauzula PATH.

• tip podataka *SDO_GEOMETRY* koristi se za *Oracle Spatial* i *Graph* podatke. Postoji mogućnost korištenja *JSON_TABLE* s GeoJSON podacima.

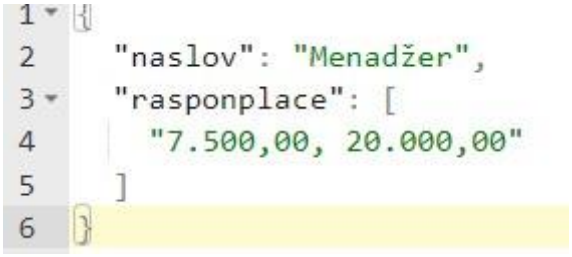
• ugniježđeni stupci sastoje se od ključne riječi *NESTED*, nakon koje slijedi po izboru ključna riječ *PATH*. Klauzula *COLUMNS* navodi stupce koji predstavljaju ugniježđene podatke.

SQL/JSON funkcija *JSON_TABLE* generira SQL/JSON uvjet *JSON_EXISTS* i SQL/JSON funkcije *JSON_VALUE* i *JSON_QUERY*. Sve mogućnosti funkcija *JSON_VALUE* i *JSON_QUERY* moguće je napraviti i pomoću *JSON_TABLE*. Ukoliko se više puta koriste neke od JSON funkcija, moguće je riješiti jednim pozivanjem *JSON_TABLE* funkcije.

JSON vrijednost može biti polje ili uključivati jedan ili više ugniježđenih polja na bilo koji broj razina unutar drugih JSON polja ili objekata. Zbog poboljšanih performansi upita, moguće je izraditi prikaz nad JSON podacima koje unutar stupaca prikazuju SQL/JSON funkcije *JSON_TABLE*. Također, za još bolji rad i poboljšanje performansi moguće je stvoriti materijalizirani prikaz i postaviti JSON podatke u memoriju.

6.8 SQL/JSON funkcija JSON_ARRAY

SQL/JSON funkcija *JSON_ARRAY* konstruira JSON polje iz rezultata procjene njegovih argumenata SQL izrazom. Svaki argument može biti bilo koji SQL izraz. Rezultirajuće polje ima element za svaki argument koji je naveden. Vrijednost argumenta koja nije NULL i nije broj pretvara se u JSON polje. Slika 6.4 prikazuje JSON_ARRAY prikaz podataka o radnom mjestu te minimalnoj i maksimalnoj plaći.



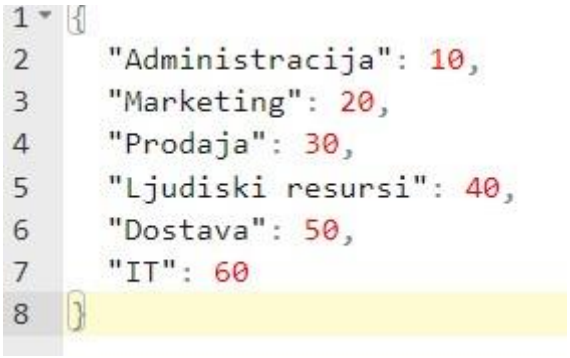
Slika 6.4: Prikaz podataka pomoću json_array funkcije

SQL/JSON funkcija *JSON_ARRAYAGG* konstruira JSON niz agregiranjem informacija iz više redaka grupiranog SQL upita kao elemenata niza. Redoslijed elemenata polja prema zadanim postavkama održava redoslijed rezultata upita, ali moguće je koristiti klauzulu *ORDER BY* za nametanje redoslijeda elemenata polja. Glavna razlika između *JSON_ARRAY* funkcije i *JSON_ARRAYAGG* je da *JSON_ARRAY* broj elemenata u nizu izravno održava broj argumenata, dok *JSON_ARRAYAGG* veličine rezultirajućeg niza održava trenutne upitane podatke.

6.9 SQL/JSON funkcija JSON_OBJECT

SQL/JSON funkcija *JSON_OBJECT* konstruira JSON objekt iz parova naziv-vrijednost. Svaki naziv para mora imati SQL identifikator te vrijednost para može biti bilo koji SQL izraz. Naziv i vrijednost odvojeni su ključnom riječi *VALUE*. Procijenjeni argumenti koje *JSON_OBJECT* omogućava su eksplicitni nazivi polja, objekata i vrijednosti polja.

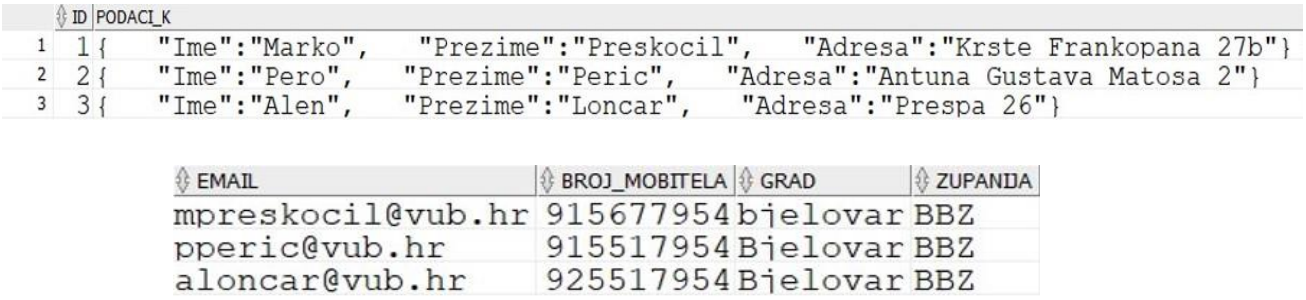
Za razliku od slučaja za SQL/JSON funkcije *JSON_OBJECT*, gdje broj članova u rezultirajućem objektu izravno održava broj argumenata, za *JSON_OBJECTAGG* veličina rezultirajućeg objekta održava trenutne upitane podatke. Stoga se može razlikovati, ovisno o podacima koji se traže. Slika 6.5 prikazuje konstruirani JSON objekt iz tablice.



Slika 6.5: Prikaz podataka korištenjem JSON_OBJECTAGG funkcije

7. PRIMJERI PRIKAZA JSON PODATAKA

Korištene tablice za prikaz podataka su *KORISNIK* te *PODUZECE* koje sadrže različite tipove podataka te JSON tip podatka. U tablici *KORISNIK* stupac koji sadrži JSON podatke naziva se *PODACI_K*, a tablica *PODUZECE* sadrži u stupcu *PODACI_P* JSON podatke. Upiti su pisani unutar Oracle baze podataka (pomoću VIEW-ova), a podaci se prikazuju unutar Postman aplikacije zbog veće preglednost, jednostavnosti i ljepšeg prikaza podataka. Slika 7.6 prikazuju tablice *KORISNIK*. Slika 7.7 prikazuju tablicu *PODUZECE* koje je korišteno za dohvaćanje podataka.



	ID	PODACI_K
1	1	{"Ime":"Marko", "Prezime":"Preskocil", "Adresa":"Krste Frankopana 27b"}
2	2	{"Ime":"Pero", "Prezime":"Peric", "Adresa":"Antuna Gustava Matosa 2"}
3	3	{"Ime":"Alen", "Prezime":"Loncar", "Adresa":"Prespa 26"}

EMAIL	BROJ_MOBITELA	GRAD	ZUPANIJA
mpreskocil@vub.hr	915677954	bjelovar	BBZ
pperic@vub.hr	915517954	Bjelovar	BBZ
aloncar@vub.hr	925517954	Bjelovar	BBZ

Slika 7.6: Prikazuje tablicu KORISNIK koja sadrži JSON podatke

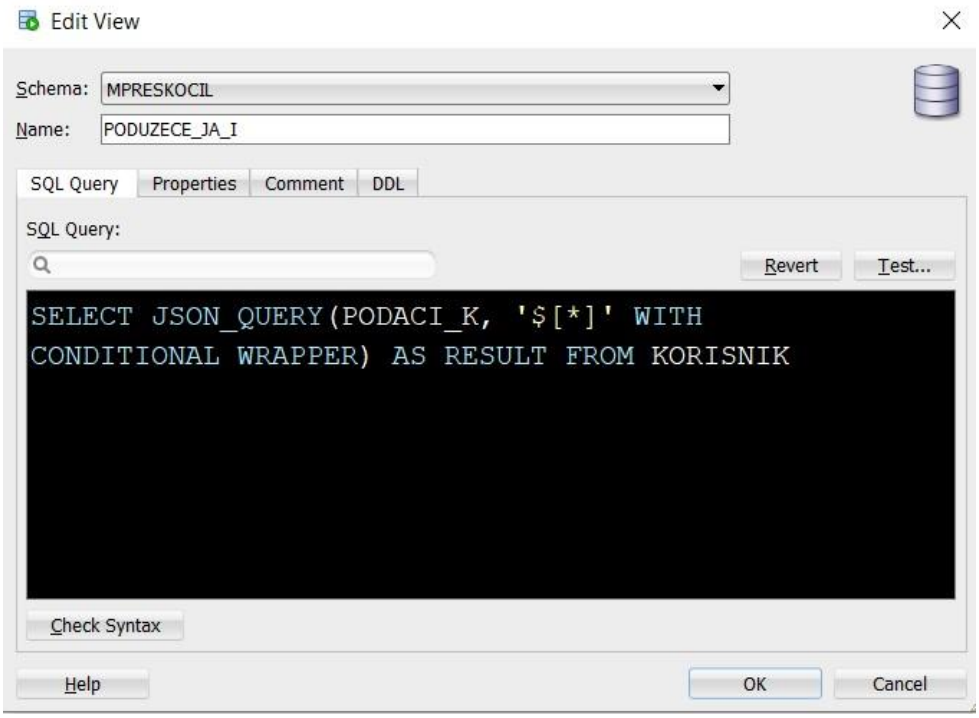
	ID	PODACI_P
1	1	{"Naziv":"TISAK", "Adresa":"Trg Bana Jelačića 104, Zagreb"}
2	2	{"Naziv":"GLS", "Adresa":"Ilica 32, Zagreb"}
3	3	{"Naziv":"Posta", "Adresa":"Ilica 241, Zagreb"}

IBAN	OIB	WEB
234758392818293948272 1	12684752718	https://www.tisak.hr/
943837262782918382211	73839283922	https://gls-group.eu/HR/hr
674289301748390165923 4	93728192827	https://www.posta.hr/

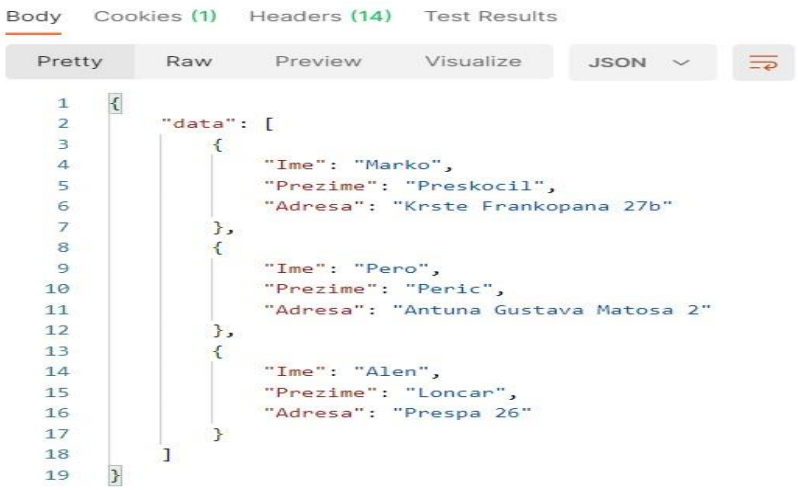
Slika 7.7: Prikaz tablice PODUZECE koje sadrži JSON podatke

7.1 PRIMJER KORIŠTENJA JSON_QUERY

JSON_QUERY prikazuje sve odgovarajuće vrijednosti koje su tražene u određenom upitu. Slika 7.8 prikazuje VIEW, a slika 7.9 prikazuju rezultat primjera korištenja *JSON_QUERY* funkcije koja iz stupca *PODACI_K* iz tablice *KORISNIK* prikazuje sve podatke koji sadrži JSON podatke s uglatim zagradama ([]).

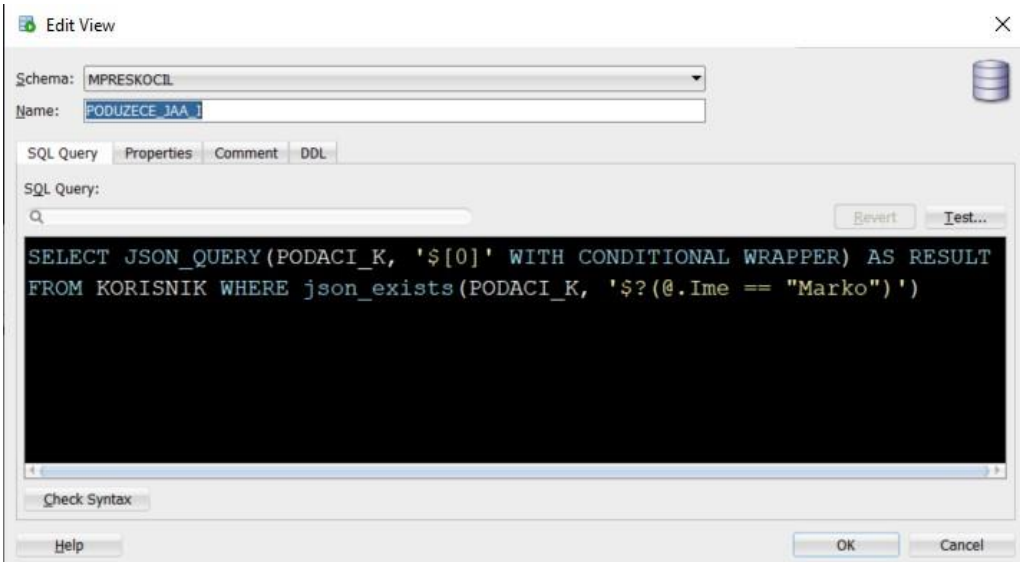


Slika 7.8: Prikaz VIEW-a za dohvaćanje svih podataka pomoću JSON_QUERY



Slika 7.9: Prikaz podataka dohvaćenih iz VIEW-a pomoću Postmana

Drugi primjer *JSON_QUERY* funkcije služi za prikaz samo jednog korisnika, u primjeru je korišten *JSON_EXISTS* koji služi za prikaz *Ime == Marko*. Služi za prikaz samo podatka koji u sebi sadrži *Ime* koje se podudara s *Marko*. Slika 7.10 prikazuje VIEW, a slika 7.11 prikazuje prikaz podataka unutar Postman aplikacije.

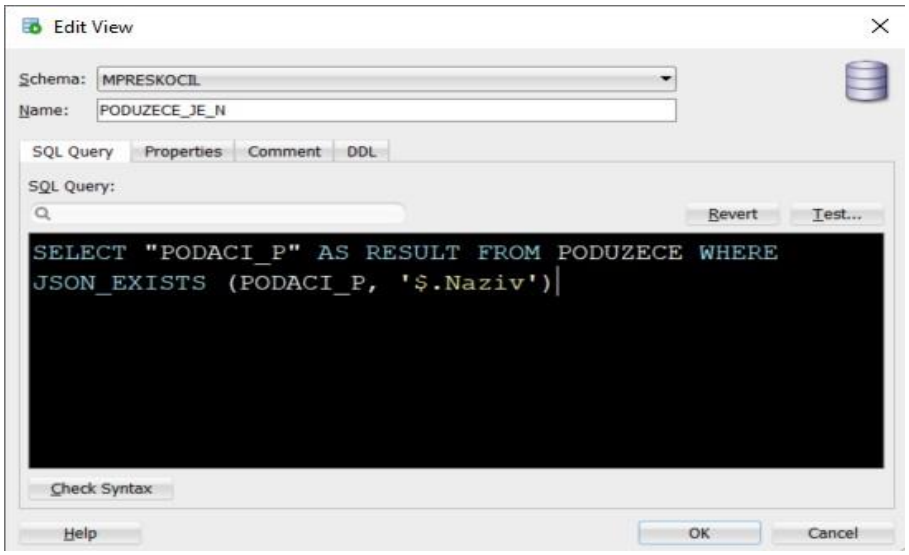


Slika 7.10: Prikaz VIEW-a koji služi za prikaz određenih podataka

Slika 7.11: Prikazuje dohvaćene podatke koji sadrže podatak 'Marko' unutar podatka 'Ime' u tablici

7.2 PRIMJER KORIŠTENJA JSON_EXISTS

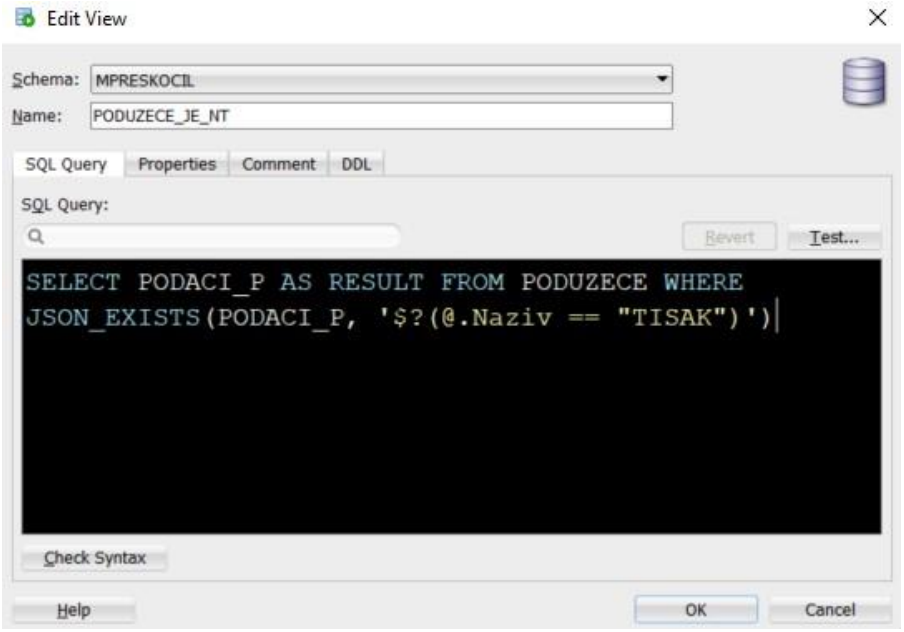
JSON_EXISTS služi za provjeru postoje li određeni podaci unutar te tablice/stupca. Slika 7.12 prikazuje VIEW koji iz *PODACI_P* prikazuje sve podatke koji sadrže *Nazive*, slika 7.13 prikazuje prikaz podataka u Postman aplikaciji.



Slika 7.12: Prikaz VIEW-a koji dohvaća sve podatke koji sadrže Naziv

Slika 7.13: Prikazuje dohvaćene podatke koji sadrže Naziv u sebi unutar Postman aplikacije

Drugi primjer *JSON_EXISTS* funkcije prikazuje dohvaćanje jednog određenoga podatka, tj. provjerava postoji li unutar *PODACI_P* naziv koji je jednak s *TISAK*. Slika 7.14 prikazuje VIEW, dok slika 7.15 prikazuje prikaz podatka unutar Postman aplikacije.

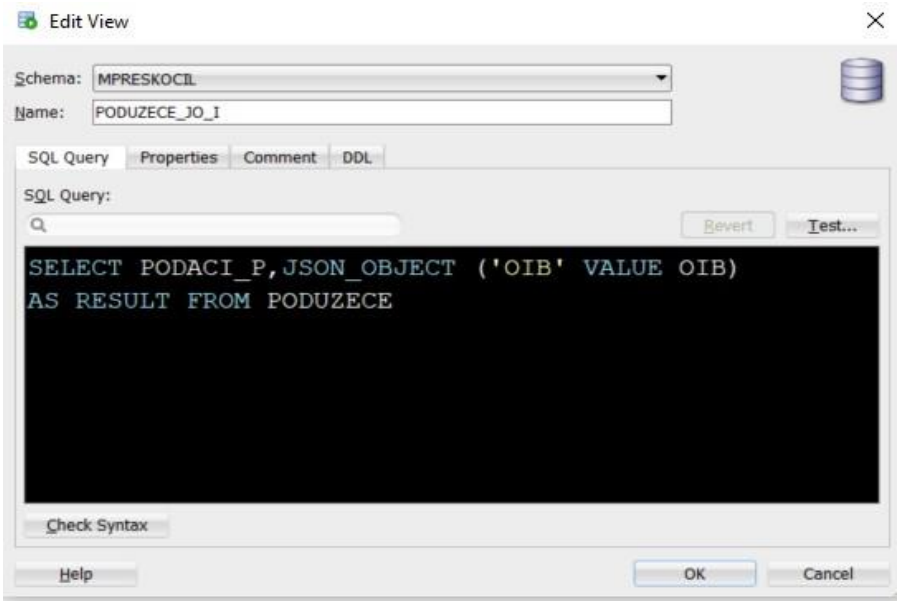


Slika 7.14: Prikaz VIEW-a koji prikazuje podataka ukoliko je 'Naziv' jednak TISAK

Slika 7.15: Prikazuje prikaz podatka koji ima Naziv Tisak unutar Postman aplikacije

7.3 PRIMJER KORIŠTENJA JSON_OBJECT

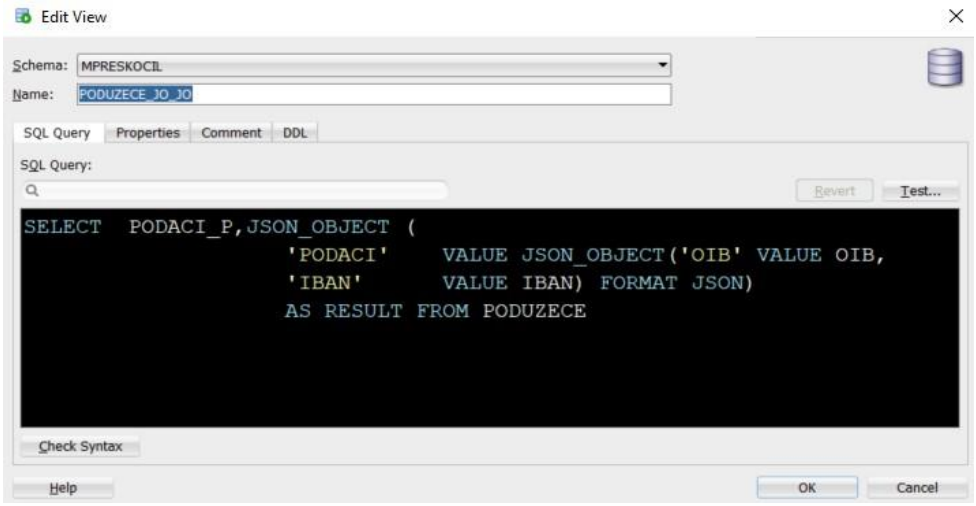
JSON_OBJECT funkcija uzima jedan ili više parova ključ/vrijednosti te vraća JSON objekt koji sadrži člana tih objekata. Slika 7.16 prikazuje dohvaćanje jednog podatka *OIB* iz tablice PODUZECE. Slika 7.17 prikazuje dohvaćene podatke pomoću Postman aplikacije.



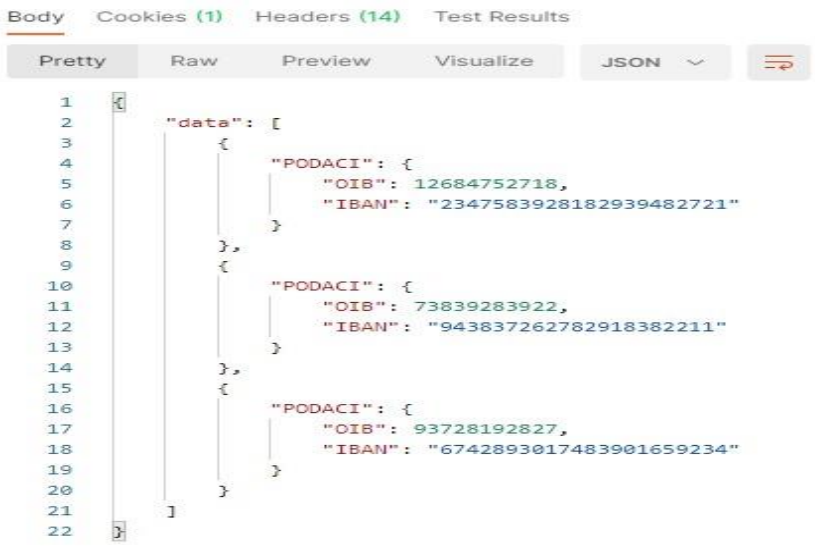
Slika 7.16: Prikaz VIEW-a koji dohvaća sve OIB podatke iz tablice PODUZECE

Slika 7.17: Prikaz dohvaćenih podataka iz VIEW-a pomoću Postman aplikacije

Drugi primjer *JSON_OBJECT* funkcije prikazuje objekt unutar objekta, unutar *PODACI* se prikazuju vrijednosti *OIB* te *IBAN*. Ti podaci se povlače iz tablice *PODUZECE*. Slika 7.18 prikazuje VIEW koji sadrži objekt *PODACI* te se unutar njega nalazi vrijednosti *OIB* i *IBAN*. Slika 7.19 pomoću Postman aplikacije prikazuje podatke dohvaćene pomoću *JSON_OBJECT*.



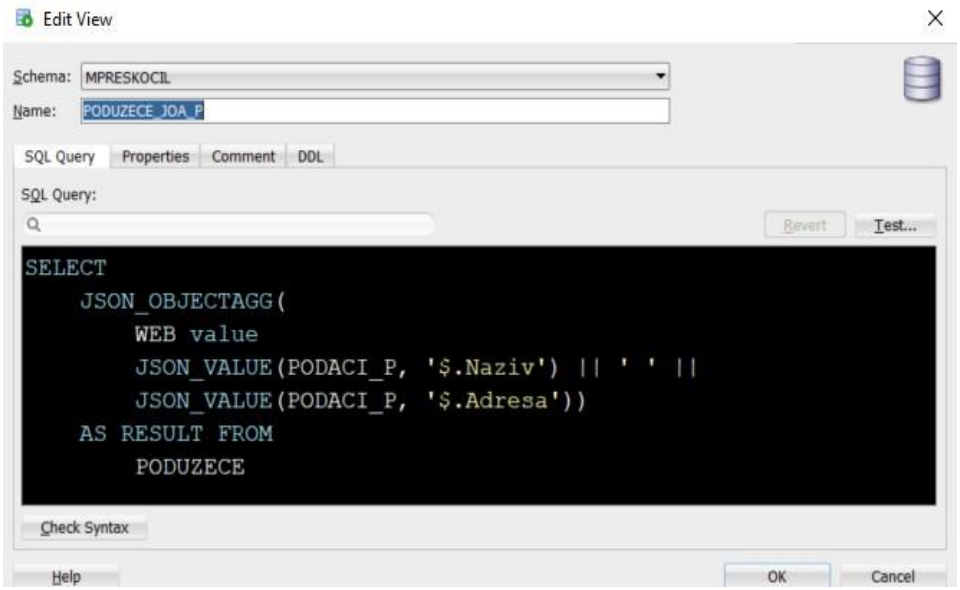
Slika 7.18:Prikaz VIEW-a u kojemu se nalazi objekt PODACI koji sadrži IBAN i OIB podatke iz tablice PODUZECE



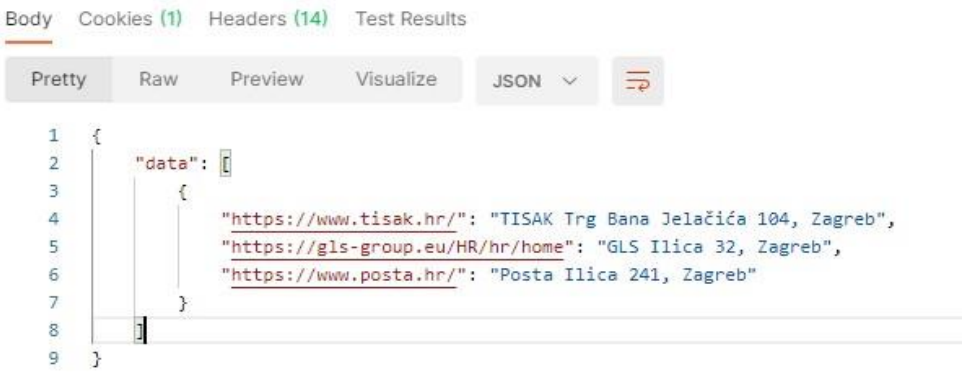
Slika 7.19: Prikaz objekta PODACI koji sadrži OIB te IBAN podatke unutar Postam aplikacije

7.4 PRIMJER KORIŠTENJA JSON_OBJECTAGG

JSON_OBJECTAGG je agregatna funkcija koja kao i *JSON_OBJECT* koja za ulaz podataka koristi ključ/vrijednost. *JSON_OBJECTAGG* vraća jedan JSON objekt sa zadanim ključem/vrijednosti. Slika 7.20 prikazuje VIEW u kojemu *WEB* vrijednost iz tablice *PODUZECE* prikazuje *Naziv* i *Adresa* iz stupca *PODACI_P*. Slika 7.21 prikazuje podatke *Naziv* i *Adresa* iz tablice *PODUZECE* pomoću *JSON_OBJECTAGG* u Postman aplikaciji.



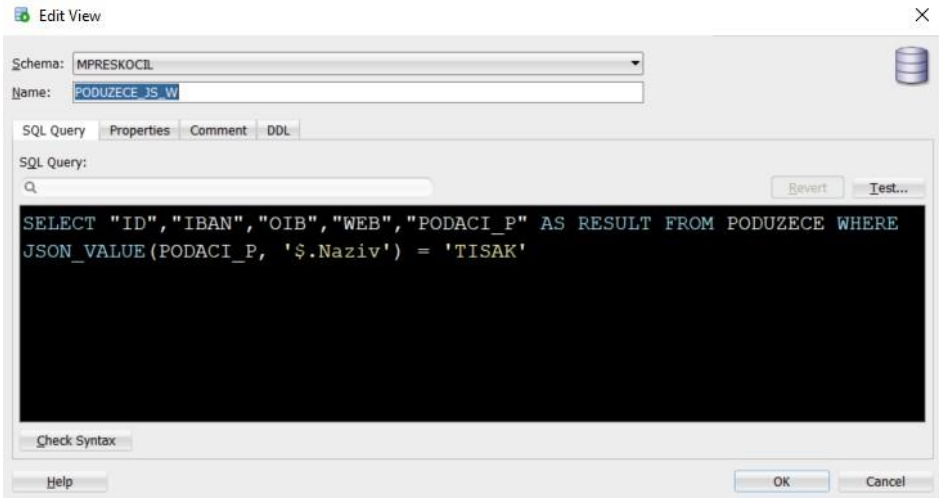
Slika 7.20: Prikazuje VIEW sa JSON_OBJECTAGG podacima



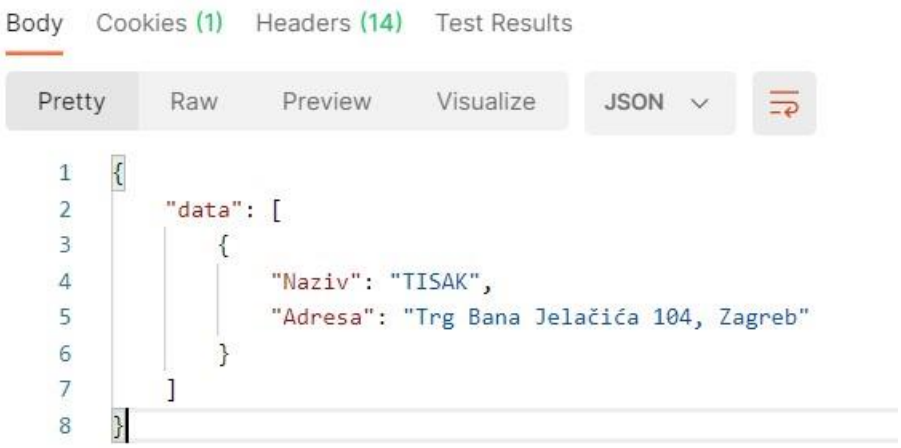
Slika 7.21: Prikazuje podatke Naziv i Adresa unutar Postman aplikacije

7.5 PRIMJER KORIŠTENJA JSON_VALUE

JSON_VALUE odabire određenu JSON vrijednost i vraća je u obliku SQL vrijednosti. Slika 7.22 prikazuje funkciju koja dohvaća iz tablice *PODUZECE* sve podatke pomoću *JSON_VALUE* iz stupca *PODACI_P* gdje je *Naziv* jednak *TISAK*. Slika 7.23 prikazuje dohvaćene podatke gdje se iz *PODACI_P* stupca dohvatila vrijednost u kojoj je Naziv jednak *TISAK*.

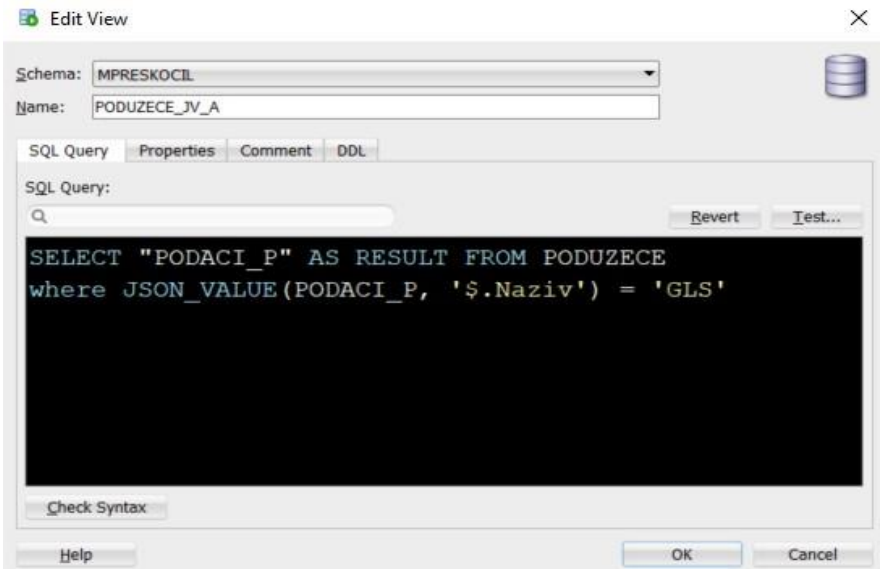


Slika 7.22: Prikaz VIEW-a sa JSON_VALUE podacima



Slika 7.23: Prikaz dohvaćenih podataka pomoću JSON_VALUE unutar Postman aplikacije

Drugi primjer prikazuje sliku 7.24 koja koristi *JSON_VALUE* funkciju koja odabire iz stupca *PODACI_P Naziv* koji je jednak vrijednosti *GLS* iz tablice *PODUZECE*. Slika 7.25 prikazuje dohvaćeni podatak u kojemu je *Naziv* jednak *GLS* u Postman aplikaciji.



Slika 7.24: Prikaz VIEW-a koji dohvaća određene podatke unutar

Slika 7.25: Prikazuje dohvaćene podatke Naziv i Adresa pomoću Postman aplikacije

8. ZAKLJUČAK

PL/SQL je proceduralni programski jezik pa JSON formatu daje puno više slobode i različitih mogućnosti. S obzirom da je JSON objekt, moguće je primijeniti objektno orijentiranu paradigmu. On omogućava jednostavnije korištenje i lakše snalaženje za korisnika prilikom rada s JSON formatom. JSON ima veliku prednost nad korištenjem i uporabom jer je razumljiv ljudima i računalima. Pogodan je za pohranu velikih količina podataka neovisno o tome kojeg tipa podataka su. Moguće je pohrana brojeva, posebnih znakova, slova, datuma. Navedeni primjeri su prikazani unutar Postman aplikacije zbog boljeg i preglednijeg prikaza JSON podatka. Ukoliko postoje različiti tipovi podataka unutar tablice, u Postman aplikaciji je moguće samo dohvatiti i prikazati samo one podatke koji su JSON. Osim same jednostavnosti JSON formata, SQL programski jezik i relacijske baze podataka pružaju različite mogućnosti korištenja s JSON podacima, fleksibilnost i podržava različite analize podataka te izvještaje o podacima. Stupci koji sadrže JSON podatke mogu imati ograničenja spremanja podataka te provjere ispravnosti podataka.

Primjeri koji su prikazani u ovome radu moguće je izmjenjivati te prikazivati iste podatke pomoću različitih funkcija. Pomoću nekih funkcija moguće je prikazivati sve podatke unutar tablice - određeni stupac unutar tablice te samo jedan podatak. Jedno od ograničenja u prikazu primjera je ograničenje s Postman aplikacijom jer prikazuje samo JSON podatke, u Oracle Database aplikaciji su dohvaćeni podaci raznovrsni i moguće je izraditi više kombiniranih funkcija.

JSON se može koristiti u bilo kojem većem sustavu kojemu je potrebna pohrana podataka; u bankarskim sustavima baza podataka gdje se pohranjuju podaci od korisnika u kojima se nalaze različiti tipovi podataka (brojevi, slova, specijalni znakovi...), u obrazovnim sustavima isto kao što je navedeno za bankarske sustave, zbog velike količine pohrane podataka od učenika, web servisi koji sadrže pohranu podataka od svojih korisnika... Neke od izrađenih primjera moguće je primijeniti za prikazivanje puno većih, kompleksnijih i raznovrsnijih JSON podataka.

9. LITERATURA

[1] vipinyadav15799: Difference Between JSON and BSON, s interneta, https://www.geeksforgeeks.org/difference-between-json-and-bson/, 15.07.2021.

[2] JSON Developer's Guide, s interneta, https://docs.oracle.com/en/database/oracle/oracle-database/18/adjsn/intro-to-json-data-and-oracle-database.html#GUID-17642E43-7D87-4590-8870-06E9FDE9A6E9, 16.07.2021.

[3] IBM Corporation: BSON and JSON built-in opaque data types, https://www.ibm.com/docs/en/informix-servers/12.10?topic=types-bson-json-built-in-opaque-data, 23.07.2021.

[4] SODA for PL/SQL Developer's Guide, s interneta, https://docs.oracle.com/en/database/oracle/simple-oracle-document-access/plsql/19/adsdp/using-soda-pl-sql.html#GUID-C553B14B-376D-43B6-BF72-CE42F605B2D9, 04.08.2021.

[5] Steven Feuerstein: Quick guide to User-Defined Types in Oracle PL/SQL, s interneta https://dzone.com/articles/quick-guide-to-user-defined-types-in-oracle-plsql, 24.08.2021.

[6] Oracle PL/SQL records type with examples, s interneta https://www.guru99.com/pl-sql-record-type.html, 24.08.2021.

[7] JSON Developer's Guide, s interneta, https://docs.oracle.com/en/database/oracle/oracle-database/18/adjsn/json-in-oracle-database.html#GUID-A8A58B49-13A5-4F42-8EA0-508951DAE0BB, 27.10.2021.

[8] JSON Developer's Guide, s interneta, https://docs.oracle.com/en/database/oracle/oracle-database/18/adjsn/conditions-is-json-and-is-not-json.html#GUID-1B6CFFBE-85FE-41DD-BA14-DD1DE73EAB20,. 27.10.2021.

[9] JSON Developer's Guide, https://docs.oracle.com/en/database/oracle/oracle-database/18/adjsn/simple-dot-notation-access-to-json-data.html#GUID-7249417B-A337-4854-8040-192D5CEFD576, 27.10.2021.

10.OZNAKE I KRATICE

API – application programming interface
BLOB - binary large object
CLOB – character large object
ECMA - European Computer Manufacturers Association
HTTP - Hypertext Transfer Protocol
JSON - JavaScript Object Notation
LOB – Large Objects
NoSQL - not only Structured Query Language
NPR - na primjer
OCI - Oracle Call Interface
PL/SQL - Procedural Language for Structured Query Language
REST - Representational state transfer
SODA - Simple Oracle Document Access
SQL - Structured Query Language
UTF - Unicode Transformation Format
XML - Extensible Markup Language

11.SAŽETAK

Ovaj završni rad temelji se na dohvaćanju **JSON** podataka pomoću **Postman aplikacije** iz **Oracle baze podataka**. Prije samih primjera teoretski su obrazložene sve osnove tematike završnoga rada te su obrazložene sve funkcije koje su se koristile za prikaz podataka. Osim obrazloženih funkcija, navedeno su sve mogućnosti i funkcionalnosti korištenja JSON podataka, način na koji se postavljaju ograničenja, ažuriraju, izmjenjuju i dodaju novi podaci, objašnjeni su tipovi podataka te **CLOB** i **BLOB** tip podataka koji se posebno odnose na pohranu JSON podataka.

Podaci se nalaze u Oracle Database aplikaciji, bilo je potrebno izraditi tablice koje sadrže JSON podatke, unutar njih umetnuti te podatke, izraditi sekvence i okidače za tablice, izraditi router te get_data funkciju za dohvaćanje podataka te na taj način povezati Oracle Database i Postman aplikaciju. Prije početka dohvaćanja tih podataka u Postmanu je bilo potrebno napraviti 'login' GET poziv pomoću kojega se spajamo s bazom podataka.

Ključne riječi: JSON, Postman aplikacija, Oracle baza podataka, CLOB, BLOB

12.ABSTRACT

This final paper is based on retrieving **JSON** data using the **Postman application** from **Oracle Database**. Before the examples themselves, all the basics of the topic of the final paper are theoretically explained, and all the functions that were used to present the data are explained. In addition to the explained functions, all possibilities and functionalities of using JSON data are listed, how restrictions are set, updated, modified and new data are added, data types are explained and **CLOB** and **BLOB** data type which is especially related to JSON data storage.

The data is in the Oracle Database application, it was necessary to create tables containing JSON data, insert the data within them, create sequences and triggers for tables, create a router and get_data functions for retrieving data and thus connect Oracle Database and Postman application. Before retrieving this data, postman had to make a 'login' GET call to connect to the database.

Keywords: JSON, Postman application, Oracle Database, CLOB, BLOB

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 25.10.2021.	MARKO PREŠKOČIL	Preškočil

Prema Odluci Veleučilišta u Bjelovaru, a u skladu sa Zakonom o znanstvenoj djelatnosti i visokom obrazovanju, elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru bit će pohranjene i javno dostupne u internetskoj bazi Nacionalne i sveučilišne knjižnice u Zagrebu. Ukoliko ste suglasni da tekst Vašeg završnog rada u cijelosti bude javno objavljen, molimo Vas da to potvrdite potpisom.

Suglasnost za objavljivanje elektroničke inačice završnog rada u javno dostupnom nacionalnom repozitoriju

MARKO PRESKOČIL

ime i prezime studenta/ice

Dajem suglasnost da se radi promicanja otvorenog i slobodnog pristupa znanju i informacijama cjeloviti tekst mojeg završnog rada pohrani u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu i time učini javno dostupnim.

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 25. 10. 2021.

Preskočil

potpis studenta/ice