Prijava na bazu podataka - SQLTools

Hostname: **???**
Username : **???**
Password: **???**
Host string: **etflab**

Manipulacija sa podacima

Vježbe koji su prezentirane u predhodnim lekcijama odnosile su se na dobivanje podataka iz baze podataka upotrebom upita po bilo kom kriteriju. Tako dobiveni podaci mogu se dalje koristiti od stane bilo koje aplikacije za prezentaciju i/ili evantualnu modifikaciju u zavisnosti od potreba i namjene aplikacije. Međutim, do sada nije bilo riječi o načinu unosa, promjene ili brisanja podataka u bazi podataka, što je cilj ove lekcije.

Dio SQL jezgre koji pruža mogućnosti promjene podataka putem SQL komandi naziva se *D*ata *M*anipulation *L*anguage – DML. Dakle, kada se dodaju, ažuriraju ili brišu podaci u bazi podataka izvršavaju se DML komande kao što su INSERT, UPDATE ili DELETE. Skup DML iskaza definisanih tako da se izvršavaju u formi logičke jedinice naziva se transakcija.

INSERT ... VALUES iskaz
INSERT ... VALUES iskaz unosi podatke u tabelu slog po slog. Osnovna struktura za dodavanje novog sloga u tabelu data je sljedećom strukturom:

```
SQL> INSERT INTO tabela (kolona1, kolona2, kolona3, ...)
                VALUES (vrijednost1, vrijednost2, vrijednost3, ...);
```

U sintaksi je:

tabela	- je naziv tabele u koju se dodaje novi slog
kolona 1..n	- je kolona tabele u koju će se dodati nova vrijednost
vrijednost 1..n	- je odgovarajuća vrijednost za kolonu

Prilikom unosa podataka u tabelu upotrebom INSERT ... VALUES iskaza moraju se poštovati sljedeća pravila:

- Vrijednosti u VALUES listi moraju odgovarati kolonama u listi kolona, tj. prva vrijednost se mora unijeti za prvu kolonu, druga vrijednost za drugu kolonu i tako redom.
- Vrijednosti koje se unose moraju biti istog tipa kao i kolone u koju će se unijeti ta vrijednost.
- Veličina podatka koji se unosi u tabelu mora odgovarati veličini kolone. Na primjer, ne može se unijeti podatak koji je veći od maksimalne veličine definisane za datu kolonu. Ako je maksimalna veličina 100 karaktera, a pokušava se unijeti podatak koji je veći od ove veličine, SQL će prijaviti grešku.

Napomena
Prilikom dodavanja novog sloga u tabelu, neohodno je provjeriti sljedeće tri stari:

- Stukturu tabele, tj. tipove podataka za svaku od kolona u koju će se dodati nova vrijednost.
- Za koje od kolona je obavezan unos vrijednosti.

- Ukoliko je u pitanju zavisna tabela, tj. postoje kolone u tabeli u koje se mogu upisati jedino vrijednosti iz druge (nezavisne) tabele da bi podatak bio prihvaćen od strane SQL-a (postoji strani ključ), provjeriti koje su to nezavisne tabele i vrijednosti koje je moguće unijeti kao novu vrijednost.

Na primjer, predpostavimo da je potrebno unijeti novi odjel «Informatika» u tabelu odjela, SQL iskaz za unos novog sloga u tabelu odjela bi izgledao kako slijedi:

```
SQL> INSERT INTO departments(department_id,
                             department_name,
                             manager_id,
                             location_id)
                     VALUES (222,
                             'Informatika',
                             100,
                             1000);
```

INSERT ... VALUES ne zahtjeva unos naziva kolona da bi se unio neki slog. Ukoliko imena kolona nisu navedena, SQL očekuje da se u VALUES dijelu INSERT iskaza navedu sve vrijednosti prema broju kolona koji postoje u tabeli respektivno, tj. SQL unosi prvu vrijednost za prvu kolonu, drugu vrijednost za drugu kolonu i tako redom. Ovo je ujedno prvi od krirerija za uspiješno dodavanje slogova u bazu podataka.

Prvi kreterij za uspiješno dodavanje slogova u bazu podataka je da vrijednosti koje se unose moraju biti istog tipa kao i navedene kolone. Najednostavniji način da se provjeri tip podataka za svaku od kolona u navadenoj tabeli je upotreba komande za opis strukture tabela – **DESC**RIBE. Tako na primjer, da bi smo dobili strukturu tabele zaposlenih, u SQL-u bi otipkali:

```
SQL> DESC employees;
```

Drugi kriterij za uspiješno dodavanje novog sloga je da se provjere koje od kolona smiju, a koje ne, imate NULL vrijednosti prilikom unosa. To se jednostavno da provjeriti kroz DESC komandu, nakon čijeg izvršenja se vidi da li neka od kolona može sadržavati NULL vrijednost. Ukoliko kolona ne može sadržavati NULL vrijednost u opisu strukture tabele na tim pozicijama će se pojaviti oznaka NOT NULL.

Treći kriterij su dozvoljene vrijednosti koje se mogu dodavati za pojedine kolone tabele, a koje zavise od podataka iz drugih tabela. O ovome će biti riječi mnogo više u sekciji za kreiranje tabela.

INSERT ... SELECT iskaz

INSERT ... VALUES iskaz dodaje nove podatke u tabelu, ali očigledno je da ima jedno značajno ograničenje, a to je da se na ovaj način dodaje samo po jedan slog tabeli u bazi podataka. Da je bilo potrebno kopirati sadržaj tabele koja ima 500,000 ili više slogova, onda je više nego očigledno da bi takav poduhvat kopiranja trajao čitavu vječnost preko INSERT ... VALUES iskaza. U ovakvim situacijam, mnogo bolja opcija je korištenje INSERT ... SELECT iskaza. Osnovna struktura ovog iskaza data je sa sljedećom strukturom:

```
SQL> INSERT INTO tabela (kolona1, kolona2, kolona3, ...)
                SELECT kolona1, kolona2, kolona3, ...
                FROM tabela1, tabela2, ...
                WHERE tabela1.kolona1 = tabela2.kolona2
```

```
                    ...
                    and dodatni_uslov
                    ...;
```

U navednoj strukturi, kao što se da vidjeti, neophodno je napisati odgovarajući podupit koji će obezbijetiti podatke iz jedne ili više tabela s istim kriterijalnim pravilima kao što postoje i za INSERT ... VALUES iskaz.

Na primjer, neka je potrebno napraviti arhivu slogova svih zaposlenih koji su zaposleni poslije 1996 i koji imaju platu veću od 2000 KM i zaposleni su odjelima 10,30, 60 i 80.

```
SQL> INSERT INTO employees_history
     SELECT *
     FROM employees
     WHERE to_number(to_char(hire_date, 'yyyy')) > 1996
           and salary > 2000
           and department_id IN (10,30,60,80);
```

Ovdje se predpostavlja da je u tabeli employees_history raspored i broj kolona isti kao u tabeli employees. Da to nije bilo slučaj bilo bi neophodno navesti sve nazive kolona u INSERT INTO dijelu za davedenu tabelu historije, a potom u SELECT klauzuli navesti sve nazive kolona koje korespodentno odgovaraju kolonama INSERT INTO iskaza. Na ovaj način smo kopirali sve zaposlene koji zadovoljavaju definisani kriterij.

INSERT ... SELECT zahtjeva još tri dodatna pravila za uspiješno dodavanje slogova u tabelu navedenu u INSERT INTO dijelu izkaza, a to su:

- Podupit ne može filtrirati slogove u tabeli u koju se unose slogovi.
- Broj kolona u upitu mora odgovarati broju kolona koja se nalaze u INSERT INTO iskazu.
- Tipovi podataka, za navedene kolone u INSERT INTO iskazu, moraju biti istog tipa kao i u navedenom podupitu.

UPDATE iskaz

UPDATE iskaz se koristi za modifikaciju postojećih slogova u bazi podataka. Osnovna sintaksa za ažuriranje podataka u bazi podataka preko UPDATE DML komade predstavljena je sa sljedećom strukturom:

```
SQL> UPDATE tabela
     SET kolona1 = vrijednost1,
         Kolona2 = vrijednost2,
         Kolona3 = vrijednost3,
         ...
     WHERE uslov;
```

U sintaksi je:

tabela	- je naziv tabele u kojoj se ažuriraju slog/vi
kolona 1..n	- je kolona tabele za koju se ažurira vrijednost
vrijednost 1..n	- je odgovarajuća vrijednost za kolonu
uslov	- je uslov na osnovu kojeg se određuje koji će se slogovi ažurirati u bazi podataka. Ako uslov ne postoji, nakon izvršenja ove DML komande, ažurirati će se svi slogovi u tabeli.

Na primjer, neka je potrebno povećati platu svim zaposlenim za broj mjeseci od dana njihovog zaposlenja:

```
SQL> UPDATE employees
     SET salary = salary + ABS(ROUND(MONTHS_BETWEEN(hire_date,
                  sysdate),0));
```

Kao što se vidi u predhodnom primjeru u UPDATE iskazu mogu se ažurirati kolone bazirane na rezultatima aritmetičkih funkcija. Kada se koristi ova tehnika za povećanje, ili eventualno spajanje više različitih vrijednosti, neophodno je voditi računa da je tip podataka isti kao i kolona koja se ažurira. U slučaju pojave greške ili rezultata koji se nije očekivao može se izvršiti ROLLBACK komanda za vraćanje podataka na staro stanje,tj. prije konačnog prihvatanja ažuriranih podataka.

U slučaju da nije postojao uslov za ažuriranje slogova, kao u prethodnim primjeru, ažurirat će se svi slogovi. Česta situacija kod ažuriranja slogova tabele je da se koristi vrijednost ili vrijednosti iz drugih tabela koje su na neki način povezane sa osnovnom tabelom koja se ažurira. U takvim slučajevima koriste se podupiti u okviru UPDATE iskaza koji kao argument predaje vrijednost UPDATE iskazu.

Na primjer, neka je potrebno ažurirati platu svim zaposlenim na iznos plate kao i kod njihovih šefova. Za one koji nemaju šefa platu ostaviti nepromijenjenu.

```
SQL> UPDATE employees z
     SET z.salary = (SELECT s.salary
                     FROM employees s
                     WHERE s.employee_id = z.manager_id)
     WHERE z.manager_id IS NOT NULL;
```

Drugi primjer bi bio kada bi se trebao ažurirati posao i plata zaposlenog sa šifrom 195 na iste vrijednosti kao i kod zaposlenog sa šifrom 150.

```
SQL> UPDATE employees
     SET (job_id, salary) = (SELECT job_id, salary
                             FROM employees
                             WHERE employee_id = 150)
     WHERE employee_id = 195;
```

DELETE iskaz

DELETE iskaz se koristi za brisanje slogova iz baze podataka. Osnovna struktura DELETE iskaza data je sljedećom strukturom:

```
SQL> DELETE (FROM) tabela
     WHERE uslov;
```

U sintaksi je:

tabela	- je naziv tabele iz koje se briše/u slog/vi
uslov	- je uslov na osnovu kojeg se određuje koji će se slogovi brisati u bazi podataka. Ako uslov ne postoji nakon izvršenja ove DML komande izbrisati će se svi slogovi u tabeli.

Jedna od specifičnih stvari kod DELETE DML komande je ta što prilikom brisanja slogova iz tabele SQL neće izvršiti nikakvo upozorenje, pa iz tih razloga treba biti posebno pažljiv u onim situacijama kada imamo komandu spašavanja za svaku DML opraciju izvršenu nad određenim slogovima. Kod DELETE iskaza riječ FROM je opcionalna može se, a i ne mora navoditi.

Na primjer, neka je potrebno izbrisati sve zaposlene koji primaju platu veću od 5000 KM.

```
SQL> DELETE FROM employees
     WHERE salary > 5000;
```

Kod upotrebe DELETE iskaza treba imati na umu sljedeće:

- DELETE iskazom se ne može obrisati vrijednost jedne kolone.
- DELETE iskaz briše cijeli slog, a ne samo njegove pojedine kolone.
- Prilikom brisanja slogova nad pojedinim (nezavisnim) tabelama može se pojaviti problem referencijalnog integriteta u odnosu na druge (zavisne) tabele.
- DELETE iskazom se brišu samo slogovi, a ne i sama tabela.

Imajući prethodno u vidu i u zavisnosti od toga kako je definisan WHERE uslov u DELETE iskazu SQL može uraditi sljedeće:

- Izvršiti brisanje samo jednog sloga.
- Izvršiti brisanje više slogova.
- Izvršiti brisanje svih slogova tabele.
- Da se ne izvrši brisanje ni jednog slog, jer je uslov i suviše restriktivan.

Prilikom brisanja slogova neke od tabela, moguće je imati i podupite koji se koriste kao argumenti za poređenje vrijednosti u WHERE klauzuli. Kao primjer toga, neka je potrebno izbrisati sve zaposlene koji rade u odjelu marketinga i imaju platu veću o prosječne plate ostalih zaposlenih u tom odjelu.

```
SQL> DELETE FROM employees e
     WHERE e.department_id IN (SELECT d.department_id
                               FROM departments d
                               WHERE d.department_name = 'Marketing')
           and e.salary > (SELECT avg(t.salary)
                           FROM employees t
                           WHERE t.department_id = e.department_id);
```

Zadaci

1. Kreirati tabelu zaposlenih u okviru šeme baze na koju ste trenutno logirani. Za naziv tabele koriste slovo «z» i broj vašeg indeksa (na primjer z14004):

   CREATE TABLE z14004 AS SELECT * from employees;
2. Opišite strukturu vaše tabele i identifirajte nazive kolona. Da li postoje neka ograničenja vezana za pojedine kolone tabele? Ako postoje koja su, ako ne zašto ne postoje?
3. U vašu tabelu zaposlenih dodajte 5 novih slogova za odjel marketinga i šefom sa šifrom 100.
4. Promijenite dodatak na platu za sve one zaposlene koji imaju platu manju od 3000 KM.
5. Promijenite platu za sve one zapslene koji rade u New Yorku tako da im je plata uvećana za dodatak na platu ako ga imaju, a ako ne onda smanjiti platu za 10% i dodatak na platu uvećati za 15%.
6. Modificirati šifru odjela za sve one zaposlene, u vašoj tabeli zaposlenih, koji rade u Americi i imaju platu manju od prosječne plate svih zaposlenih u dotičnom odjelu, osim datog zaposlenog, tako da pripada odjelu Makretinga, i nemaju platu jednaku minimalnoj i maksimalnoj plati na nivou svih organizacijskih jednica.
7. Modificirati šifru šefa, u vašoj tabeli zaposlenih, za sve one zaposlene koji su nadređeni onim šefovima koji posjeduju veći broj zaposlenih od prosječnog broja zaposlenih kod svih preostalih šefova, onom šefu koji posjeduje minimalan broj zaposlenih.
8. Na osnovu prvog primjera kreirati file koji sadrži komande za kreiranje nove vaše tabele odjela koja će se zvati slično kao tabela u prvom primjeru, samo što će se umjesto slova «z» sada koristiti slovo «o» i broj vašeg indeksa.
9. Modificirati sve nazive odjela, u vašoj tabeli odjela, tako što će te ispred imena odjela staviti «US -», ako se odjel nalazi u Americi, u protivnom staviti «OS -» za sve ostale odjele.
10. Iz vaše tabele zaposlenih izbrisati sve one zaposlene koji rade u onim odjelima koji u imenu sadrže, na bilo kojoj poziciji, slovo 'a' ili 'A'.
11. Iz tabele odjela izbrisati sve odjele u kojim ne radi ni jedan zaposlenik.
12. Izbrisati sve one zaposlene, iz vaše tabele zaposlenih, koji ne rade u Aziji i imaju šefa koji je nadređen bar trojici zaposlenih, i gdje taj šef ima šefa koji prima platu veću od plate onog šefa koji u okviru firme ima minimalan broj zaposlenih kojim je nadređen.