Prijava na bazu podataka - SQLTools

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

Ugniježdeni SQL izkazi - podupiti

Podupit je upit čiji se rezultat prenosi kao parametar drugom upitu, ili bolje rečeno, to je SELECT iskaz koji je ugniježden u neku od klauzula vodećeg SELECT izkaza. Podupit je veoma koristan kada je potrebno dobiti podatak iz tabele pod uslovom koji zavisi od podatka iste te tabele, ili veoma komplikovanog rezultata upotrebom agregatnih funkcija. Podupiti omogućavaju povezivanje dvaju ili više upita u jednu cjelinu – jedan osnovni (glavni) upit.

Podupit se može pojaviti u bilo kojoj od klauzula osnovnog (glavnog) upita, tj. u klauzuli:
- select,
- from,
- where i/ili
- having.

Tipovi podupita:

Tip podupita	Opis
Podupit – jednog sloga	Podupit koji vraća samo jedan slog kao parametar osnovnom upitu.
Podupit – više slogova	Podupit koji vraća više od jednog sloga kao parametar osnovnom upitu.
Podupit – više kolona	Podupit koji vraća kao parametar osnovnom upitu više od jedne kolone.

Dalja podjela ove osnovne podjele podupita mogla bi se podijeliti na:

Tip podupita	Opis
Nekolerisani podupit	To je vrsta podupita koja je samostalna i za čije izvršenje, tj. dohvatanje podataka iz baze podataka se ne koriste podaci iz osnovnog (glavnog) upita.
Kolerisani podupit	Ova vrsta podupita nije nezavisna i vraća podatake iz baze podataka koristeći podatke osnovnog upita i to najčešće u where i/ili having klauzuli. Ova vrsta podupita koristi podatak osnovnog upita kao podatak za filtriranje svojih vrijednosti (podataka).

Kao što je predhodno napomenuto, podupit se može nalaziti u bilo kojoj od klauzula osnovnog upita, ali je najčešći način upotrebe podupita kod WHERE klauzule. Generalizirana struktura ovog podupita data je kao što slijedi:

```
SQL>  SELECT kolona1, kolona2, kolona3, ....
      FROM tabela1, tabela2, ...
      WHERE kolona1 <operator> (SELECT kolonaA
                                FROM tabelaA
                                WHERE uslov)
```

```
GROUP BY kolona
HAVING grupni uslov
ORDER BY grupna funkcija / kolona;
```

Podupit – jednog sloga
Kao što se vidi na datoj strukturi između kolone osnovnog upita i podupita mora postojati operator. To može biti bilo koji od operatora koji je korišten do sada za poređenje vrijednosti kolona dosadanjih upita. Za podupite – jednog sloga najčešće korišteni operatori su dati sljedećom tabelom:

Operator	Značenje
=	Jednako
>	Veće od
>=	Veće od ili jednako
<	Manje od
<=	Manje od ili jednako
<> (!=)	Nije jednako (različito)

Kod ove vrste podupita specifično je to što vraća samo jedan slog, odnosno samo jednu vrijednost. To je i razlog zašto je moguće koristiti operatore koji se koriste za poređenje jedne vrijednosti. Na primjer, neka je potrebno napisati upit koji će prikazati šifru, ime, prezime, platu, i šifru odjela za sve zaposlene koji rade u istom odjelu kao i Sarah.

```
SQL> SELECT employee_id,
            last_name,
            first_name,
            salary,
            department_id
     FROM employees
     WHERE department_id = (SELECT department_id
                            FROM employees
                            WHERE first_name = 'Sarah');
```

Podupit koji je dat u ovom primjeru je nekolerisani podupit, zato što ne koristi podatke osnovnog upita za svoje izvršenje. Plan izvršenja ovog upita je da se najprije izvrši podupit, tj. dohvati određena vrijednost (podatak) iz baze podataka, a potom se taj podatak, kao parametar osnovom upitu, koristi za filtriranje podataka vodećeg upita.

Sljedeći primjer predstavlja upotrebu kolerisanog podupita, tj. kada se vrijednost/i osnovnog upita koristi/e za filtriranje podataka podupita. Na primjer, neka je potrebno napisati upit koji će prikazati naziv zaposlenog i platu za sve zaposlene koji u svom odjelu ne pripadaju onim zaposlenim koji dobivaju minimalnu ili maksimalnu platu, sortirano po iznosu plate i nazivu zaposlenika.

```
SQL> SELECT o.employee_id,
            o.last_name || ' ' || o.first_name naziv,
            o.salary
     FROM employees o
     WHERE o.salary <> (SELECT min(p.salary)
                        FROM employees p
                        WHERE p.department_id = o.department_id)
           and o.salary <> (SELECT max(p.salary)
                            FROM employees p
                            WHERE p.department_id = o.department_id)
     ORDER BY o.salary, o.last_name || ' ' || o.first_name;
```

Kao što se da vidjeti iz priloženog primjera, upotreba podupita u WHERE klauzuli, rješava poprilično brzo komplikovanu strukturu postavljenog upita na veoma jednostavan način. Uporeba podupita može biti jako korisna i u SELECT klauzuli, što se može vidjeti kroz sljedeći primjer. Neka je za prethodni upit potrebno ispisati još i naziv odjela i broj zaposlenih u kojem dotični zaposlenik radi.

```
SQL> SELECT o.employee_id,
          o.last_name || ' ' || o.first_name naziv,
          o.salary,
          (SELECT d.department_name
           FROM departments d
           WHERE d.department_id = o.department_id) "odjel",
          (SELECT count(*)
           FROM employees t
           WHERE t.department_id = o.department_id) "broj uposlenih"
     FROM employees o
     WHERE o.salary <> (SELECT min(p.salary)
                        FROM employees p
                        WHERE p.department_id = o.department_id)
           and o.salary <> (SELECT max(p.salary)
                            FROM employees p
                            WHERE p.department_id = o.department_id)
     ORDER BY o.salary, o.last_name || ' ' || o.first_name;
```

Podupit – više slogova

Ova vrsta podupita se koristi kada je potrebno izvršiti poređenje u osnovnom upitu sa skupom mogućih vrijednosti. Za razliku od operatora podupita jednog sloga, kod podupita više slogova koriste se operatori koji očekuju jednu ili više vrijednosti, a to su:

Operator	Značenje
IN	Jednako nekoj od vrijednosti iz liste mogućih vrijednosti.
<*operator*> ANY	Poredi vrijednost sa svakom od vrijednosti u podupitu i provjerava da li postoji bar jedna vrijednost iz podupita koja zadovoljava uslov. Operator «=ANY» ima isto značenje kao i operator «IN».
<*operator*> ALL	Poredi vrijednost sa svakom od vrijednosti iz podupita, i provjerava da li je uslov zadovljen za sve vrijednosti podupita.
EXISTS	Vraća TRUE ako postoji vraćen bar jedan slog za zadati ulov u podupitu, u protivnom vraća FALSE. Operator EXISTS je jako specifičan po tome što ne postoji poredbena vrijednost, niti se uzima u obzir broj vraćenih kolona u SELECT klauzuli podupita.

Na primjer, predpostavimo da je potrebno napisati upit koji će prikazati sve zaposlene koji primaju platu veću od prosječne plate bilo kojeg odjela.

```
SQL> SELECT e.employee_id,
          e.last_name || ' ' || e.first_name naziv,
          e.salary
     FROM employees e
     WHERE e.salary > ANY (SELECT avg(z.salary)
                           FROM employees z
                           GROUP BY z.department_id);
```

Kao što se može vidjeti na navedenom primjeru ovo je nekolerisani podupit koji se izvršava prvi, i za čije izvršavanje se ne koriste podaci osnovnog upita. Međutim, mnogo češća situacija kod ovih vrsta podupita je da se koriste vrijednosti osnovnog upita na osnovu kojeg se filtriraju podaci u podupitu, a potom njegov rezultat koristi kao argument (parameter) za filtriranje vrijednosti osnovnog upita.

Kao primjer, predpostavimo da je potrebno napisati upit koji će prikazati šifru, naziv, platu, i platu uvećanu za dodatak na platu, ako ne postoji dodatak na platu napisati «nema dodatka na platu», za sve zaposlene čiji šefovi u tim organizacijskim jedinicama primaju prosječnu platu veću od prosječne plate svih zaposlenih ostalih organizacijskih jedinica.

```
SQL> SELECT e.employee_id šifra,
     e.last_name || ' ' || e.first_name naziv,
     e.salary plata,
     nvl(to_char((SELECT t.salary * (1 + t.commission_pct)
                  FROM employees t
                  WHERE e.employee_id = t.employee_id)),
        'Nema dodatka na platu') dodatak
     FROM employees e
     WHERE (SELECT AVG(t.salary)
            FROM employees t
            WHERE employee_id IN (SELECT t1.manager_id
                                  FROM employees t1)
                  and t.department_id = e.department_id)  >
                 (SELECT avg(t.salary)
                  FROM employees t
                  WHERE t.department_id <> e.department_id);
```

Na predstavljenom primjeru upotrijebljen je i operator IN. Namjera je da se izdvoje šifre svih šefova u onim organizacijskom jedinicama kojem pripada i dotični zaposlenik koji se trenutno obrađuje kroz osnovni upit. Interesantno je primjetiti da će podupit «SELECT t1.manager_id FROM employees t1» vratiti sve menadžere iz tabele zaposlenih, a potom će operator IN u podupitu osigurati provjeru po jednakosti, šifre zaposlenika i šifre menadžera, po istoj organizacijskoj jedinici trenutnog zaposlenog koji se obrađuje kroz osnovni upit.

Sljedeći primjer ilustruje upotrebu operatora ALL. Na primjer, neka je potrebno modificirati predhodni upit da prikazuje sve zaposlene koji dobivaju platu veću od svakog zaposlenog iz odjela 60 i 90, izuzimajući zaposlene iz ovih odjela. Upit će izgledati kako slijedi:

```
SQL> SELECT e.employee_id šifra,
     e.last_name || ' ' || e.first_name naziv,
     e.salary plata,
     nvl(to_char((SELECT t.salary * (1 + t.commission_pct)
                  FROM employees t
                  WHERE e.employee_id = t.employee_id)),
        'Nema dodatka na platu') dodatak
     FROM employees e
     WHERE e.salary > ALL (SELECT t.salary
                           FROM employees t
                           WHERE t.department_id IN (60,90))
           and e.department_id NOT IN (60,90);
```

Operator EXISTS

EXISTS posmatra podupit kao parametar i kao rezultat daje TRUE, ako je podupit vratio bar jedan slog, a FALSE ukoliko je rezultat upita prazan, tj. nema vraćenih vrijednosti kao rezultata obrade podupita. Kako EXISTS operator posmatra kao parametar rezultat podupita, tako i broj kolona koje se mogu naći u listi SELECT klauzule nije bitan, niti se uzima u obzir, pa se često puta mogu pronaći ove vrste upita napisane tako da se u podupitu SELECT klauzule nalaze konstantne vrijednosti tipa: karakter, broj i sl., umjesto stvarne kolone i/ili kolona.

Kao primjer predpostavimo da je potrebno napisati upit koji će prikazati naziv zaposlenog, naziv odjela, naziv posla, grad i platu za sve zaposlene iz odjela 10, 30, 50, 60 i 90, ako postoji bar jedan zaposleni iz navedenih odjela koji radi u Americi.

```
SQL> SELECT e.last_name || e.first_name naziv,
          d.department_name odjel,
          l.city grad,
          j.job_title posao,
          e.salary plata
     FROM employees e, departments d, jobs j, locations l
     WHERE e.department_id = d.department_id
          and e.job_id = j.job_id
          and d.location_id = l.location_id
          and EXISTS (SELECT e1.employee_id, d1.department_id,
                             l1.location_id, c1.country_id,
                             r1.region_id
                      FROM employees e1,departments d1,
                           locations l1,countries c1,
                           regions r1
                      WHERE e1.department_id = d1.department_id
                           and d1.location_id = l1.location_id
                           and l1.country_id = c1.country_id
                           and c1.region_id = r1.region_id
                           and d1.department_id IN (10,30,50,60,90)
                           and lower(substr(r1.region_name,1,7)) =
                               'america')
          and d.department_id IN (10,30,50,60,90);
```

U podupitu gore navedenog upita, definisana je lista od pet kolona koji se predaju kao argument osnovnom upitu. Međutim, kao što je navedeno i ranije operator EXISTS ne gleda broj niti raspored kolona u podupitu, pa iz tih razloga gore navedeni upit može se napisati tako da se u SELECT klauzuli upiše neka konstanta vrijednost, kao na primjer 'x', što se najčeće i radi:

```
SQL> SELECT e.last_name || ' ' || e.first_name naziv,
          d.department_name odjel,
          l.city grad,
          j.job_title posao,
          e.salary plata
     FROM employees e, departments d, jobs j, locations l
     WHERE e.department_id = d.department_id
          and e.job_id = j.job_id
          and d.location_id = l.location_id
          and EXISTS (SELECT 'x'
                      FROM employees e1,departments d1,
                           locations l1,countries c1,
                           regions r1
```

```
                    WHERE e1.department_id = d1.department_id
                          and d1.location_id = l1.location_id
                          and l1.country_id = c1.country_id
                          and c1.region_id = r1.region_id
                          and d1.department_id IN (10,30,50,60,90)
                          and lower(substr(r1.region_name,1,7)) =
                              'america')
          and d.department_id IN (10,30,50,60,90);
```

Podupit – više kolona

Do sada objašnjeni podupiti su sadržavali samo po jednu kolonu, osim EXISTS operatora koji je po svojoj prirodi specifičan, koji su kao argument za poređenje korišteni u WHERE ili HAVING klauzuli osnovnog upita. Međutim, često puta situacija nalaže poređenje dvije ili više kolona u okviru WHERE klauzule osnovnog upita, što je moguće uraditi preko podupita koji vraća dvije ili više kolona. Istina, ove situacije se mogu riješiti upotrebom dva ili više podupita, tj. onoliko podupita koliko ima kolona za poređenje, koji vraćaju po jednu vrijednost (kolonu). Nažalost, to nije toliko fleksibilno i troši znatno više resursa baze podataka za razliku od jednog podupita koji to sve može riješiti u jednom prolazu. Operator koji se koristi za poređenje dvaju ili više kolona osnovnog upita i podupita je operator IN. Ovaj operator je objašnjen do sada u prethodnom tekstu, tako da nema potrebe za njegovim ponovnim objašnjenjem. Generalizirana struktura ove vrste upita data je kako slijedi:

```
SQL> SELECT kolona1, kolona2, kolona3, ....
     FROM tabela1, tabela2, ...
     WHERE (kolona1, kolona2, ...) [NOT] IN  (SELECT kolona1, kolona2, ...
                                              FROM tabela1, tabela2, ...
                                              WHERE uslov)
     GROUP BY kolona
     HAVING grupni uslov
     ORDER BY grupna funkcija / kolona;
```

Na primjer, neka je potrebno napisati upit koji će vratiti šifru zaposlenog, naziv zaposlenog, naziv posla i platu za sve zaposlene koji rade u odjelu 50 i imaju još samo jednog kolegu koji dobiva istu platu kao i dati zaposlenik.

```
SQL> SELECT e.employee_id šifra,
            e.last_name || ' ' || e.first_name naziv,
            j.job_title posao,
            e.salary plata
     FROM employees e, jobs j
     WHERE e.job_id = j.job_id
           and (e.department_id,
                 e.salary, 2) IN (SELECT t.department_id,
                                         t.salary, count(*)
                                  FROM employees t
                                  WHERE t.department_id = 50
                                  GROUP BY t.department_id, t.salary);
```

NULL vrijednosti kao povratne vrijednosti iz podupita

Postoje situacije kada je moguće očekivati da pojave NULL vrijednosti kao povratne vrijednosti kolone/kolona iz podupita. U takvim situacijama u zavisnosti od korištenog operatora, dovoljno je da postoji samo jedna NULL vrijednost u skupu vraćenih vrijednosti, i da osnovi upit kao rezultat ne vrati ni jedan slog. Operator NOT IN je ekvivalentan sa operatorom !=ALL, ali i pored toga, ako je vrijednost NULL preostala vrijednost za koju bi trebalo vratiti slog u osnovnom upitu, to se neće desiti.

Na primjer, kako u okviru tabele zaposlenih postoji jedan zaposleni koji nema nadređenog, direktor firme, i ako bi se pokušao napisati upit koji će vratiti sve one zaposlene koji nemaju nadređenog, upotrebom podupita, upit bi izgledao kako slijedi:

```
SQL> SELECT *
     FROM employees
     WHERE employee_id NOT IN (SELECT manager_id
                               FROM employees
                               WHERE manager_id IS NULL);
```

Međutim, upit napisan na ovakav način naće vratiti ni jedan slog, a za očekivati je bilo da će vratiti bar jedan.

Podupit u FROM klauzuli

Podupiti se mogu koristiti i u FROM klauzuli select izkaza, koji je veoma sličan načinu upotrebe pogleda. Podupit napisan u FROM klauzuli se posmatra kao tabela, i ako je u okviru date klauzule definisano više tabela i dati podupit, onda je potrebno povezati tabele sa podupitom na isti način kao što se povezuju tabele međusobno.

Predpostavimo da je potrebno napisati upit koji će prikazati šifru i naziv zaposlenog, šifru i naziv odjela, platu i prosječnu platu odjela za sve zaposlene koji primaju platu veću od prosječne plate odjela u kojem dotični zaposleni rade.

```
SQL> SELECT e.employee_id "šifra zaposlenog",
            e.last_name || ' ' || e.first_name "naziv zaposlenog",
            a.odjel "šifra odjela",
            d.department_name "naziv odjela",
            e.salary plata,
            a.pros_plata "prosječna plata odjela"
     FROM employees e, departments d, (SELECT e1.department_id odjel,
                                              avg(e1.salary) pros_plata
                                       FROM employees e1
                                       GROUP BY e1.department_id) a
     WHERE e.department_id = d.department_id
           and e.department_id = a.odjel
           and e.salary > a.pros_plata;
```

Zadaci

1. Napisati upit koji će prikazati naziv zaposlenog, naziv odjela i naziv posla za sve zaposlene koji rade u istom odjelu kao i Susan, isključujući Susan.
2. Napisati upit koji će prikazati šifru, ime, prezime, platu za sve zaposlene koji zarađuju platu veću od prosječne plate svih zaposlenih iz odjela 30 i 90.
3. Napisati upit koji će prikazati sve podatke o zaposlenim za sve zaposlene koji rade u istom odjelu kao i neki od zaposlenih koji u imenu, na bilo kom mjestu, sadrže slovo «C».
4. Napisati upit koji će prikazati šifru i naziv zaposlenog, kao i naziv posla za sve zaposlene koji rade u odjelu koji je locairan u Torontu.
5. Napisati upit koji će prikazati sve podatke o zaposlenim koji izvještavaju King-a.
6. Modificirati upit pod rednim brojem 3 tako da prikazuje samo one zaposlene koji dobivaju platu veću od prosječne plate svih zaposlenih iz dotičnog odjela u kojem dati zaposlenik radi.
7. Napisati upit koji će prikazati naziv zaposlenog, naziv odjela i platu za sve one zaposlene koji pripadaju istom odjelu i zarađuju istu platu kao i neki od zaposlenih koji dobiva dodatak na platu, isključujući one zaposlene koji dobivaju dodatak na platu.
8. Napisati upit koji će prikazati naziv zaposlenog, naziv odjela, platu i grad za svakog zaposlenog koji ima istu platu i dodatak na platu kao i neki od zaposlenih koji rade u Rimu.
9. Napisati upit koji će prikazati naziv zaposlenog, datum zaposlenja i platu za sve zaposlene koji imaju istu platu i dodatak na platu kao i Scott.
10. Napisati upit koji će prikazati samo one zaposlene koji zarađuju platu veću od plate svih iz odjala za prodaju. Rezultat sortirati po plati od najveće do najmanje.
11. Napisati upit koji će prikazati naziv zaposlenog, naziv odjela, naziv posla i grad za sve zaposlene koji primaju platu veću od prosjećne plate svojih svih šefova koji imaju dodatak na platu i rade u istom odjelu kao i dotični zaposlenik.
12. Napisati upit koji će prikazati šifru i naziv zaposlenog, šifru i naziv odjela, platu, prosječnu, minimalnu i maksimalnu platu odjela u kojem zaposlenik radi, kao i minimalnu, maksimalnu i prosječnu platu na nivou firme za sve zaposlene koji imaju platu veću od minimalne prosječne plate svih šefova u odjelu u kojim dati zaposlenik radi.