Prijava na bazu podataka - SQLTools

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

Agregatne funkcije

Agregante funkcije su funkcije koje daju kao rezultat vrijednosti bazirane na vrijednostima iz kolone, pa se zbog toga ove funkcije nazivaju još i grupnim funkcijama. Osnovne agregatne funkcije kao što su *COUNT*, *SUM*, *AVG*, *MAX* i *MIN* su definisane ANSI standardom, istina neke od implementacija SQL koriste nesto drugačije definicije imena, ali većina ih se pridržava ANSI standarda i većina ih je implementirana od strane baza podataka. Mnoge baze podatka posjeduju i jedan širi skup funkcija koje nisu definisane putem ANSI standard, i to proširenje funkcija daje neke dodatne mogućnosti koje se mogu dobiti kombinacijom dvije ili više ANSI funkcija.

Funkcija COUNT
Funkcija *COUNT* je jedna od SQL agregatnih funkcija koja kao rezultat vraća ukupan broj broj slogova koji zadovoljavaju uslov u *WHERE* klauzuli, ako postoji, ili u protivnom vraća broj svih slogova koji su dohvaćeni putem SQL upita. Tako na primjer, ako bi bilo potrebno saznati broj zaposlenih u odjelu 30, upit bi izgledao kako slijedi:

```
SQL> SELECT count(employee_id)
     FROM employees
     WHERE department_id = 30;
```

Pošto je cilj ove funkcije da vrati broj slogova koji zadovoljavaju uslov *WHERE* kaluzule, onda nije toliko bitno koja će se kolona navesti u datoj funkciji, pa se iz tih razloga u većini slučajeva umjesto naziva kolone, kao parametar datoj funkciji prosljeđuje znak "*". Imajući to u vidu prethodni upit se može napisati na sljedeći način:

```
SQL> SELECT count(*)
     FROM employees
     WHERE department_id = 30;
```

Funkcija SUM
Kao što i samo ime kaže ova funkcija vrši sabiranje, tj. kao rezultat vraća zbir svih vrijednosti iz navedene kolone. Na primjer, neka je potrebno napisati upit koji će vratiti mjesečnu platu svih zaposlenih.

```
SQL> SELECT sum(salary)
     FROM employees;
```

Funkcija AVG
Funkcija *AVG* izračunava srednju vrijednost vrijednosti iz kolone. Tako na primjer, ako bi trebalo saznati koja je prosječna plata u odjelu 90, upit bi izgledao kako slijedi:

```
SQL> SELECT avg(salary)
     FROM employees
     WHERE department_id = 90;
```

Funkcije MIN i MAX
Funkcije MIN i MAX se koriste za vraćanje minimalne odnosno maksimalne vrijednosti u okviru jedne kolone. Tako na primjer, ako se treba saznati minimalna i maksimalna plata iz odjela 30, upit bi izgledao kako slijedi:

```
SQL> SELECT min(salary) min_plata, max(salary) max_plata
     FROM employees
     WHERE department_id = 30;
```

Napomena
U slučaju da pokušate koristiti u WHERE klauzuli neku od gore pobrojanih funkcija SQL će vam prijaviti grešku.

```
SQL> SELECT count(*), sum(salary)
     FROM employees
     WHERE min(salary) < 2900;
```

Greška: ***ORA-00934: group function is not allowed here***

GROUP BY i HAVING klauzule

GROUP BY klauzula se najčešće koristi kod upotrebe agregatnih (grupnih) funkcija kao što su COUNT, SUM, AVG, MIN, MAX. Ako se pokušavaju u SQL-u napisati upiti koji koriste kolone tabele i agregante (grupne) funkcije nad kolonama, SQL će javiti grešku kod izvršavanja takvog upita, ako se predhodno nije unijela i GROUP BY klauzula za kolone upita nad kojima se vrši grupisanje u SELECT klauzuli. GROUP BY klauzula pokreće agregatnu/e funkciju/e navedene u SELECT klauzuli za svaku grupnu vrijednost kolone navene u GROUP BY klauzuli. Neka je potrebno napisati upit koji će prikazati naziv odjela i mjesečnu platu svih zaposlenih po odjelima.

Ako pokušamo riješiti ovaj zadatak bez upotrebe GROUP BY klauzule, kao što slijedi:

```
SQL> SELECT d.department_name "naziv odjela",
            sum(e.salary) "mjesečna plata odjela"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id;
```

SQL će prijaviti grešku: ***ORA-00937: not a single-group group function***

Kao što se može vidjeti bez upotebe GROUP BY klauzule SQL prijavljuje grešku prilikom pokretanja SQL upita, jer se ima situacija da se treba ispisati naziv odjela za svaki slog u tabeli departmenta i sumarno plate po odjelima svih zaposlenih. U normalnim okolnostima da se je izostavio naziv odjela i pokrenuo upit, kao rezultat izvršenja upita dobio bi samo jedan slog sa sumarnim iznosom svih zaposlenih, odnosno svih odjela, što se nije tražilo kao rezultat izvršenja upita.

Ispravan upit za predhodni zadatak bio bi:

```
SQL> SELECT d.department_name "naziv odjela",
            sum(e.salary) "mjesečna plata odjela"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id
     GROUP BY s.department_name;
```

Struktura SQL upita uvođenjem GROUP BY klauzule ima sada sljedeću strukturu:

```
SQL> SELECT kolona, grupna_funkcija(kolona)
     FROM tabela
     WHERE uslov
     GROUP BY kolona
     ORDER BY grupna funkcija / kolona;
```

Napomena
Sve grupne funkcije izuzev count(*) ignorišu NULL vrijednosti. COUNT funkcija posjeduje dva formata:

- COUNT(*)
- COUNT(kolona),

gdje COUNT(*) vraća broj slogova u tabeli, uključujući duple slogove i slogove koji posjeduju NULL vrijednost, dok COUNT(kolona) vraća broj slogova za sve slogove koji ne posjeduju null vrijednost.

Zamjena za NULL vijednost je NVL funkcija, koja mijenja NULL vijednost u neku predefinisanu vijednost. Kao primjer, predpostavimo da trebamo napisati upit koji će prikazati sumu dodataka na platu i broj uposlenih koji dobivaju dodatak na platu. Upit bi izgledao kao što slijedi:

```
SQL> SELECT sum(salary*commission_pct) "suma dodataka na platu",
            count(commission_pct) "broj uposlenih"
     FROM employees;
```

Ovaj zadatak se mogao riješiti i upotrebom funkcije count(*), ali bi se u tom slučaju mora dodati i uslov da se žele prebrojati samo oni zaposleni koji dobivaju dodatak na platu.

```
SQL> SELECT sum(salary*commission_pct) "suma dodataka na platu",
            count(*) "broj uposlenih"
     FROM employees
     WHERE commission_pct is not null;
```

Ako bi se za prethodni upit upotrijebila NVL funkcija nad kolonom commission_pct i zamijenila commission_pct sa NVL(commission_pct,0) dobili bi smo broj svih zaposlenih i, naravno, sumu dodataka na platu koja je ista kao i u predhodnom zadatku.

```
SQL> SELECT sum(salary*nvl(commission_pct,0)) "suma dodataka na platu",
            count(*) "broj uposlenih"
     FROM employees
     WHERE nvl(commission_pct,0) is not null;
```

Međutim, predhodni upit se mogao napisati i kao upit koji daje isti rezultat, bez upotrebe nvl funkcije, zbog gore navedene specifičnosti count(*) i sum funkcija na sljedeći način:

```
SQL> SELECT sum(salary*commission_pct) "suma dodataka na platu",
            count(*) "broj uposlenih"
     FROM employees;
```

Pravilo
Ne koristiti SELECT izkaz nad kolonama koje imaju različite vrijednosti od kolona u GROUP BY kaluzuli. Obrnuto ne vrijedi, osnosno, mogu se koristi vrijednosti kolona u GROUP BY klauzuli nad kolonama koje se ne navode u SELECT klauzuli. Na primjer, neka treba napisati upit koji će prikazati samo broj zaposlenih po odjelima.

```
SQL> SELECT  d.department_name "naziv odjela",
            count(*) "broj uposlenih"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id
     GROUP BY d.department_name;
```

Ali da se pokušalo izvršiti grupisanje podataka samo preko šifre organizacionog dijela, što nije pogrešno ako se logično posmatra, ipak bi se dobila greška jer je u SELECT klauzuli navedena kolona department_name, a ne department_id.

```
SQL> SELECT  d.department_name "naziv odjela",
            count(*) "broj uposlenih"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id
     GROUP BY d.department_id;
```

Greška: ***ORA-00979: not a GROUP BY expression***

Da smo stavili u GROUP BY kaluzuli department_name i department_id ne bi bilo pogrešno, a rezultat bi i dalje bio tačan:

```
SQL> SELECT  d.department_name "naziv odjela",
            count(*) "broj uposlenih"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id
     GROUP BY d.department_name, d.department_id;
```

Grupisanje podataka po više kolona
Ponekad postoji potreba za gupisanje podataka po više različitih kolona, odnosno da se vidi rezultat upita za grupe unutar grupe. To praktično znači da će se pojaviti više različitih kolona u GRUOP BY klauzuli pobrojanih ili ne pobrojanih kolona iz SELECT klauzule. Na primjer, predpostavimo da trebamo napisati upit koji će prikazati broj zaposlenih po nazivu posla i šifri organizacionog dijela.

```
SQL> SELECT  j.job_title "naziv posla",
            e.department_id "šifra odjela",
            count(*) "broj uposlenih po odjelima"
     FROM employees e, jobs j
     WHERE e.job_id = j.job_id
     GROUP BY j.job_title, e.department_id;
```

HAVING klauzula
HAVING klauzula omogućava upotrebu agregatnih funkcija u izkazu poređenja, obezbijeđujući za agregatne funkcije ono što WHERE klauzula obezbijeđuje za pojedinačne redove, tj. kolone tabele. U ovoj klauzuli su dozvoljeni svi logički operatori i operatori poređenja agregarnih funkcija kao i u WHERE klauzli kada se vrši poređenje kolona sa kolonama ili kolona sa konstantnim vrijednostima i slično.

Struktura SQL upita uvođenjem HAVING klauzule ima sada sljedeću strukturu:

```
SQL> SELECT kolona, grupna_funkcija(kolona)
     FROM tabela
     WHERE uslov
     GROUP BY kolona
     HAVING grupni uslov
     ORDER BY grupna funkcija / kolona ;
```

Primjer, napisati upit koji će prikazati naziv odjela, sumarnu i prosjećnu platu za sve odjele koji primaju prosjećnu mjesečnu platu veću od 7000 KM.

```
SQL> SELECT  d.department_name "naziv odjela",
             avg(salary) "prosječna plata odjela",
             sum(salary) "sumarna plata odjela"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id
     GROUP BY d.department_name
     HAVING avg(salary) > 7000;
```

U HAVING klauzuli mogu se uključiti i poređenja za kolone, ali u tom slučaju treba voditi računa da se mogu dobiti rezultati drugačiji od očekivanih. S druge strane, neke implementacije SQL-a ukoliko se upotrijebi nešto drugo izuzev agregatnih funkcija u HAVING klauzuli će prijaviti grešku. U svakom slučaju, restrikcije koje je neophodno imati za kolone tabele najbolje je ostaviti u WHERE klauzuli gdje se postiže mnogo jasniji i prirodniji upiti za ostale programere koji koriste dati upit za dalju upotrebu.

Primjer, da je potrebno napisati prethodni upit samo za organizacione dijelove koji u nazivu, na bilo kojoj poziciji, imaju slovo 'a'.

```
SQL> SELECT  d.department_name "naziv odjela",
             avg(e.salary) "prosječna plata odjela",
             sum(e.salary) "sumarna plata odjela"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id
     GROUP BY d.department_name
     HAVING avg(e.salary) > 7000
            and lower(d.department_name) like '%a%';
```

Predhodni upit bolje napisan, izgleda kao što slijedi:

```
SQL> SELECT  d.department_name "naziv odjela",
             avg(e.salary) "prosječna plata odjela",
             sum(e.salary) "sumarna plata odjela"
     FROM employees e, departments d
     WHERE e.department_id = d.department_id
           and lower(d.department_name) like '%a%'
     GROUP BY d.department_name
     HAVING avg(e.salary) > 7000;
```

Sortiranje podataka po grupnim funkcijama je dozvoljeno u okviru ORDER BY klauzule i nije potrebno uvoditi nikakav dodatni kod da bi se upit izvršio. Osim toga u ORDER BY klauzuli moguće je imati kombinacije agregatnih funkcija i kolona, po kojima je potrebno sortirati podatke.

Na primjer, neka je potrebno sortirati podatke po prosječnoj plati organizacionog dijela i nazivu odjela iz prethodnog upita.

```
SQL> SELECT  d.department_name "naziv odjela",
            avg(e.salary) "prosječna plata odjela",
            sum(e.salary) "sumarna plata odjela"
FROM employees e, departments d
WHERE e.department_id = d.department_id
GROUP BY d.department_name
HAVING avg(e.salary) > 7000
ORDER BY avg(e.salary), d.department_name;
```

Zadaci

1. Napisati upit koji će prikazati sumu iznosa datataka na platu, broj zaposlenih koji dobivaju dodatak na platu, kao i ukupan broj zaposlenih.
2. Napisati upit koji će prikazati broj zaposlenih po poslovima i organizacionim jedinicama. Za labele uzeti naziv posla, naziv organizacione jedinice i broj uposlenih respektivno.
3. Napisati upit koji će prikazati najveću, najmanju, sumarnu i prosječnu platu za sve zaposlene. Vrijednosti zaokružiti na šest decimalnih mjesta.
4. Modificirati prethodni upit tako da pokazuje maksimalnu, minimalnu i prosječnu platu po poslovima.
5. Napisati upit koji će prikazati broj zaposlenih po poslovima.
6. Napisati upit koji će prikazati broj menadžera, bez njihovog prikazivanja.
7. Napisati upit koji će prikazati naziv menadžera i platu samo za one menadžere koji u okviru date organizacione jedinice dobivaju minimalnu platu u odnosu na sve ostale menadžere ostalih odjela.
8. Napisati upit koji će prikazati naziv odjela, naziv grada, broj zaposlenih i prosječnu platu za sve zaposlene u dotičnom odjelu.
9. Napisati upit koji će prikazati broj zaposlnih koji su bili zaposleni u 1995, 1996, 1997 i 1998, kao i ukupan broj zaposlenih u ovim godinama. Za labele uzeti 1995g, 1996g, 1997g, 1998g i ukupan broj zaposlenih respektivno.
10. Napisati matrični izvještaj koli će prikazati naziv posla i sumarnu platu po odjelima, kao i ukupnu platu po datim poslovima i odjelima. Za labele uzeti kao što je prikazano na tabeli:

Posao	Odjel 10	Odjel 30	Odjel 50	Odjel 90	Ukupno
Menadžer			60		60
Programer	20		55	10	85
…	…	…	…	…	…

Tabela 1. Primjer rasporeda redova i kolona za matrični izvještaj.