Prijava na bazu podataka - SQLTools

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

Poboljšano pretraživanje podataka

Poboljšano pretraživanje podataka u bazi podataka podrazumijeva upotrebu dvije ili više različitih tabela u okviru iste/različite šeme baze podataka nad kojom se vrši pretraživanje. Da bi se napisao traženi upit, tj. upit koji će vratiti sve potrebne informacije, najčešće je potrebno napisati upit koji će i u *select* klauzuli imati kolone iz više različitih tabela. Ovo objedinjavanje tabela u okviru upita postaju nešto složenije kada se pojave isti nazivi kolona u različitim tabelama. S druge strane, upotreba nekih grupnih funkcija u istoj *select* klauzuli može znatno povećati složenost upita.

Spajanje (Join)

Šta je spajanje (*join*)? Pa najednostavnije rečeno, spajanje (*join*) je povezivanje više od jedne tabele u bazi podataka. Prilikom povezivanja istih/različitih tabela mora se voditi računa o dodatnim uslovima pod kojim se vrši spajanje datih tabela. Slogovi u jednoj tabeli moraju biti povezani sa slogovima druge tabele u skladu sa odgovarajućim (korespodentnim) vrijednostima datih kolona, gdje su date kolone obično primarni ili strani ključ navedenih tabela.

Prilikom pisanja upita za slučaj dvije ili više tabela koje se nalaze u *from* klauzuli potrebno je voditi računa o sljedećem:

- ako postoje isti nazivi kolona u više od jedne table, potrebno je ispred naziva kolone pisati i naziv tabele na koju se odnosi data kolona (npr. <tabela.kolona> - zaposleni.sifra).

- za spajanje N tabela potrebno je minimalno N-1 uslova spajanja u *where* klauzuli dotičnog upita. Tako na primjer, ako se ima situacija da se u *from* klauzuli nalazi šest tabela, tada se u where klauzuli očekuje minimalno pet uslova spajanja. Po nekoj logici spajanja, ako je baza podataka dizajnirana kako treba, tada se zavisne tabele spajaju na nezavisne preko stranih ključeva, odnosno na primarne ključeve nezavisnih tabela. Naravno postoje situacije kada je ovo narušeno, ali i u tom slučaju broj uslova spajanja ne smije biti manji on N-1 tabele koje se nalaze u *from* klauzuli upita.

Kartezijanski proizvod

Kartezijski proizvod nastaje kao posljedica spajanja dvije ili više tablela kod kojih je izostavljen uslov spajanja datih tabela, a kao rezultat se očituje u spajanju svakog sloga jedne tabele sa svakim slogom druge/drugih tabela. Dakle, svi slogovi prve tabele su povezani sa svim slogovima druge/drugih tabela i tako redom.

Problem s kartezijanskim proizvodom je taj što generiše preveliki broj slogova iz baze podataka, čak i za relativno mali broj slogova u tabelama nad kojim je izvršen navedeni kartezijanski proizvod. Drugi problem koji se može pojaviti kod upotrebe kartezijankosg prizvoda je znatno zauzimaju resurse u bazi podataka. Obično je slučaj da dobijeni podaci iz baze podataka nisu upotrebljivi i korisni za bilo kakvo dalje procesiranje, pa je iz tih razloga neophodno voditi računa o tome da se napiše ispravan uslov spajanja datih tabela u *where* klauzuli, kako bi se izbjegao kartezijski proizvod.

Tako na primjer, u svrhu demonstracije kartezijanskog proizvoda, predpostavimo da je potrebno napisati upit koji će prikazati šifru zaposlenog, ime, prezime, šifru i naziv odjela za sve zaposlene.

```
SQL> SELECT employees.employee_id,
            employees.first_name,
            employees.last_name,
            departments.department_id,
            departments.department_name
     FROM employees, departments;
```

Koliko slogova imaju tabele employees i departments?
Koliko je broj dobivenih slogova u kartezijanskom proizvodu tabela employees i departments?

Sinonimi za tabele

U prethodnom primjeru dat je upit kojim je bilo potrebno prikazati šifru zaposlenog, ime, prezime, šifru i naziv odjela u kojem zaposleni radi. Da nije bilo prefiksa u nazivima kolona, upit bi prijavio grešku na koloni department_id, zato što kolona department_id postoji i u tabeli zaposlenih i u tabeli odjela, tako da se ne bi znalo na koju se tabelu odnosi dotična kolona šifre odjela.

Naime, ukoliko postoji kolona koja egzistira u obe tabele, mora se koristiti sinonim u *SELECT* klauzuli da bi se odredila pripadnost kolone. Uobičajna tehnika je da se dodijeli jedan karakter svakoj tabeli u *FROM* klauzuli na osnovu kojeg će se raspozanavati kolone. Nepisano pravilo je da se za sinonime tabele koristi prvo slovo imena tabele. Na primjer, za slučaj tabele zaposlenih i odjela sinonimi će biti **e** i **d** respektivno. Prethodni upit uz upotrebu sinonimima za tabela izgleda kako slijedi:

```
SQL> SELECT e.employee_id,
            e.first_name,
            e.last_name,
            d.department_id,
            d.department_name
     FROM employees e, departments d;
```

Tipovi spajanja tabela

Postoje više različitih tipova spajanja i oni se uglavnom razlikuju po načinu upotrebe logičkih operatora između kolona nad kojima se vrši spajanje. Generalno gledajući spajanja se mogu podijeliti na sljedeća:

- spajanje po uslovu jednakosti,
- spajanje po bilo kom uslovu, osim po jednakosti kolona,

- spoljašnje nasuprot unutrašnjeg spajanja,
- spajanje tabele sa samom sobom.

Spajanje po uslovu jednakosti

Predpostavimo da je prethodni primjer napisan na sljedeći način:

```
SQL> SELECT e.employee_id,
            e.first_name,
            e.last_name,
            d.department_id,
            d.department_name
     FROM employees e, departments d
     WHERE e.department_id = d.department_id;
```

Koristeći kolonu department_id, koja postoji u obe prethodne tabele, kombinovana je informacija koja postoji u tabeli employees i informacija iz tabele departments. Na ovaj način prikazani su svi zaposlenici koji rade u dogovarajućim odjelima s njihovim nazivima. Spajanje iz prethodnog primjera zove se spajanje po uslovu jednakosti. Cilj ovog spajanja je pronaći iste vrijednosti slogova jedne i druge tabele preko zajedničke kolone department_id.

Spajanje po bilo kom uslovu, osim po jednakosti kolona

S obzirom da SQL podržava spajanje po uslovu jednakosti kolona može se intuitivno predpostaviti da se može izvršiti i bilo koja druga vrstu spajanja osim jednakosti. Dok spajanje po uslovu jednakosti kolona isključivo koristi operator poređenja "=" u *WHERE* klauzuli, upiti po bilo kom uslovu spajanja, osim po uslovu jednakosti, koriste sve druge operatore poređenja osim "=".

```
SQL> SELECT e.employee_id,
            e.first_name,
            e.last_name,
            d.department_id,
            d.department_name
     FROM employees e, departments d
     WHERE e.department_id <> d.department_id;
```

U svakodnevnoj praksi upiti spajanja po uslovu jednakosti se mnogo češće koriste nego upiti spajanja po bilo kom uslovu. Međutim, realno je za očekivati da će se pojavljivati potreba i za ovaj vid spajanja.

Spoljašnje nasuprot unutrašnjeg spajanja

Unutrašnje spajanje se postiže kada se slogovi kombinuju međusobno dajući kao rezultat broj novih slogova koji je jednak proizvodu broja slogova svake tabele. Takođe, unutrašnje spajanje koristi ove slogove da bi odredilo koji su slogovi rezultat WHERE kaluzule.

```
SQL>  SELECT e.employee_id,
             e.first_name,
             e.last_name,
             d.department_id,
             d.department_name
      FROM employees e JOIN departments d ON d.department_id = 10;
```

Kao rezultat ovog upita dobili smo sve slogove iz tabele employees (employee_id, first_name, last_name), a iz tabele departmants (department_id, department_name) samo jednu vrijednost (10, Administration) za sve kolone tabele employees.

Za slučaj kada želimo da "natjeramo" SQL da prikaže sve kolone iz desne tabele, a da postavi null vrijednost u polja za koja je department_id <> 10, upit bi izgledao kako slijedi:

```
SQL>  SELECT e.employee_id,
             e.first_name,
             e.last_name,
             d.department_id,
             d.department_name
      FROM employees e RIGHT OUTER JOIN departments d
            ON  d.department_id = 10;
```

Također, postoji i lijevo spoljašnje spajanje (LEFT OUTER JOIN), kao i unutrašnje spajanje (INNER JOIN).

Oracle implementacija SQL koristi znak + (prikaži sve iako nešto nedostaje) umjesto OUTER JOIN kaluzule, tako da bi izmijenjeni prethodni primjer mogao izgledati kako slijedi:

```
SQL>  SELECT e.employee_id,
             e.first_name,
             e.last_name,
             d.department_id,
             d.department_name
      FROM employees e, departments d
      WHERE e.department_id(+) = d.department_id;
```

Spajanje tabele sa samom sobom

Spajanje tabele sa samom sobom je kao i bilo koje drugo spajanje dvije tabele. Potrebno je samo sinonimima tabele osigurati razliku među kolona jedne i druge tabele i na taj način izvršiti spojanje u *WHERE* klauzuli.

Neka je potrebno napisati upit koji će prikazati naziv zaposlenog i naziv njegovog šefa za sve zaposlene.

```
SQL>  SELECT a.first_name || ' ' || a.last_name Zaposlenik,
             b.first_name || ' ' || b.last_name Šef
      FROM employees a, employees b
      WHERE a.manager_id = b.employee_id;
```

Zadaci

1. Napisati upit koji će prikazati naziv zaposlenog , šifru i naziv odjela za sve zaposlene.
2. Napisati jedinstvenu listu svih poslova iz odjela 30.
3. Napisati upit koji će prikazati naziv zaposlenog, naziv odjela i lokaciju za sve zaposlene koji ne primaju dodataka na platu.
4. Napisati upit koji će prikazati naziv zaposlenog i naziv odjela za sve zaposlene koji u imenu sadrže slovo A na bilo kom mjestu.
5. Napisati upit koji će prikazati naziv, posao, broj i naziv odjela za sve zaposlene koji rade u DALLAS-u.
6. Napisati upit koji će prikazati naziv zaposlenog, naziv šefa i grad šefa u kojem radi. Za labele kolona uzeti Naziv zaposlenog, Šifra zaposlenog, Naziv šefa, Šifra šefa, Grad šefa, respektivno.
7. Modificirati upit pod rednim brojem šest, da prikazuje i manager-a King-a koji nema predpostavljenog.
8. Napisati upit koji će prikazati naziv zaposlenog, šifru odjela, i sve zaposlene koji rade u istom odjelu kao i uzeti zaposlenik. Za kolone uzeti odgovarajuće labele.
9. Napisati upit koji će prikazati naziv, posao, naziv odjela, platu i stepene plate za sve zaposlene kod kojih stepen plate nije u rasponu kada se na platu zaposlenog doda dodatak na platu.
10. Napisati upit koji će prikazati naziv i datum zaposlenja za sve radnike koji su zaposeleni poslije Blake-a.
11. Napisati upit koji će prikazati naziv i datum zaposlenja zaposlenog, naziv i datum zaposlenja šefa zaposlenog, za sve zaposlene koji su se zaposlili prije svog šefa.