POGLAVLJE 1

Preuzimanje zapisa

Ovo poglavlje se fokusira na osnovne SELECT izjave. Važno je dobro razumeti osnove, jer su mnoge teme ovde obrađene ne nalaze samo u težim receptima već i u svakodnevnom SQL-u.

1.1 Preuzimanje svih redova i kolona iz tabele

Zadatak

Imate tabelu i želite da vidite sve podatke u njoj.

Rešenje

Koristite specijalni * znak i zadajte SELECT na tabeli:

```
1 select *
2   from emp
```

Objašnjenje

Znak * ima posebno značenje u SQL-u. Njegova upotreba će vratiti sve kolone navedene tabele. Budući da nije navedena klauzula WHERE, vratiće se i svaki red. Alternativa bi bila da se svaka kolona navede pojedinačno:

```
select empno,ename,job,sal,mgr,hiredate,comm,deptno
  from emp
```

U kratkim upitima koje izvršavate interaktivno lakše je koristiti SELECT *. Međutim, prilikom pisanja programskog koda, bolje je navesti svaku kolonu pojedinačno. Učinak će biti isti, ali ako budete izričiti, uvek ćete znati koje kolone vraćate iz upita. Isto tako, takve upite lakše razumeju ljudi koji nisu vi (koji možda znaju ili ne znaju sve kolone u tabelama u upitu). Problemi sa SELECT * mogu se pojaviti i ako je vaš upit unutar koda, a program iz upita dobija drugačiji skup kolona nego što se očekivalo. Ako navedete sve kolone, a jedna ili više nedostaje, veća je verovatnoća da će se svaka greška voditi do određenih kolona koje nedostaju.

1.2 Preuzimanje podskupa redova iz tabele

Zadatak

Imate tabelu i želite da vidite samo redove koji zadovoljavaju određeni uslov.

Rešenje

Koristite WHERE klauzulu da zadate koje redove treba zadržati. Na primer, za pregled svih zaposlenih raspoređenih u odeljenje broj 10:

```
1 select *
2   from emp
3  where deptno = 10
```

Objašnjenje

WHERE klauzula vam omogućava da preuzmete samo redove koji vas zanimaju. Ako je izraz u WHERE klauzuli tačan za bilo koji red, tada se taj red vraća.

Većina dobavljača podržava uobičajene operatore kao što su =, <, >, <=, >=, ! i <>. Pored toga, možda ćete želeti redove koji zadovoljavaju više uslova; to se može učiniti navođenjem AND, OR i zagradama, kao što je prikazano u sledećem receptu.

1.3 Pronalaženje redova koji zadovoljavaju više uslova

Zadatak

Želite da vratite redove koji zadovoljavaju više uslova.

Rešenje

Koristite klauzulu WHERE zajedno sa klauzulama OR i AND. Na primer, ako želite da pronađete sve zaposlene u odeljenju 10, zajedno sa zaposlenima koji zarade proviziju, zajedno sa zaposlenima u odeljenju 20 koji zarađuju do 2.000 USD:

```
1 select *
2   from emp
3  where deptno = 10
4     or comm is not null
5     or sal <= 2000 and deptno=20
```

Objašnjenje

Možete da koristite kombinaciju AND, OR i zagrade da biste vratili redove koji zadovoljavaju više uslova. U primeru rešenja, klauzula WHERE pronalazi redove takve da:

- DEPTNO je 10
- COMM nije NULL
- Plata je 2.000 USD ili manje za bilo kog zaposlenog u DEPTNO 20.

Prisustvo zagrada dovodi do zajedničkog vrednovanja uslova u njima.

Na primer, razmotrite kako se skup rezultata menja ako je upit napisan u zagradama kao što je ovde prikazano:

```
select *
 from emp
where (     deptno = 10
       or comm is not null
       or sal <= 2000
      )
  and deptno=20

EMPNO ENAME  JOB    MGR   HIREDATE     SAL        COMM  DEPTNO
----- ------ ----- -----  ----------- ----- ----------  ------
7369  SMITH  CLERK  7902  17-DEC-1980   800                 20
7876  ADAMS  CLERK  7788  12-JAN-1983  1100                 20
```

1.4 Preuzimanje podskupa kolona iz tabele

Zadatak

Imate tabelu i želite da vidite vrednosti za određene kolone, a ne za sve kolone.

Rešenje

Navedite kolone koje vas zanimaju. Na primer, da biste videli samo ime, broj odeljenja i platu zaposlenih:

```
1 select ename,deptno,sal
2   from emp
```

Objašnjenje

Zadavanjem kolona u klauzuli SELECT obezbeđujete da se ne vraćaju nikakvi suvišni podaci. Ovo može biti posebno važno pri preuzimanju podataka preko mreže, jer se izbegava gubljenje vremena svojstvenog preuzimanju podataka koji vam nisu potrebni.

1.5 Davanje smislenih imena za kolone

Zadatak

Želeli biste da promenite imena kolona koje se vraćaju vašim upitom tako da budu čitljivija i razumljivija. Razmotrite ovaj upit koji vraća zarade i provizije za svakog zaposlenog:

```
1 select sal,comm
2   from emp
```

Šta je SAL? Da li je skraćenica od *sale* (srp. *prodaja*)? Je li to nečije ime? Šta je COMM? Da li je to komunikacija? Želite da rezultati imaju značajnije oznake.

Rešenje

Da biste promenili imena rezultata upita, koristite AS ključnu reč u obliku *`original_name AS ovo_ime`*. Neke baze podataka ne zahtevaju AS, ali sve ga prihvataju:

```
 1 select sal as salary, comm as commission
 2   from emp

SALARY   COMMISSION
-------  ----------
    800
   1600         300
   1250         500
   2975
   1250        1400
   2850
   2450
   3000
   5000
   1500           0
   1100
    950
   3000
   1300
```

Objašnjenje

Korišćenje ključne reči AS za davanje novih imena kolonama koje vraća vaš upit poznato je kao davanje *aliasa (rezrevnog imena, pseudoimena)* kolonama. Stvaranje dobrih pseudonima može mnogo da pomogne da upit i njegovi rezultati budu razumljivi drugima.

1.6 Referenciranje kolone sa alijasom u klauzuli WHERE

Zadatak

Koristili ste pseudonime da biste obezbedili smislenija imena kolona za svoj rezultat i želeli biste da izuzmete neke redove pomoću klauzule WHERE. Međutim, vaš pokušaj pozivanja pseudoimena u klauzuli WHERE ne uspeva:

```
select sal as salary, comm as commission
from emp
where salary < 5000
```

Rešenje

Umotavanjem upita u umetnuti prikaz možete postaviti upit kolonama sa alijasima:

```
1 select *
2   from (
3 select sal as salary, comm as commission
4   from emp
5        ) x
6  where salary < 5000
```

Objašnjenje

U ovom jednostavnom primeru možete da izbegnete umetnuti prikaz i referencu COMM ili SAL direktno u klauzuli WHERE da biste postigli isti rezultat. Ovo rešenje vas upoznaje sa onim što biste trebali učiniti kada pokušavate da uputite bilo šta od sledećeg u klauzulu WHERE:

- Zbirne funkcije
- Skalarni podupiti
- Funkcije prozora
- Aliasi

Postavljanje vašeg upita, onoga koji daje pseudonime, u umetnuti prikaz daje vam mogućnost referenciranja na alias kolone u vašem spoljnom upitu. Zašto to treba da uradite? Klauzula WHERE procenjuje se pre SELECT; tako da SALARY i COMMISSION još uvek ne postoje kada se procenjuje WHERE klauzula upita „Zadatak". Ti pseudonimi se primenjuju tek nakon završetka obrade klauzule WHERE. Međutim, klauzula FROM procenjuje se pre WHERE. Postavljanjem originalnog upita u klauzulu FROM, rezultati tog upita generišu se ispred najudaljenije klauzule WHERE, a vaša krajnja klauzula WHERE „vidi" imena aliasa. Ova tehnika je posebno korisna kada stupci u tabeli nisu posebno dobro imenovani.

Umetnuti prikaz u ovom rešenju ima alias X. Ne zahtevaju sve baze podataka da umetnuti prikaz bude eksplicitno alias, ali neke to zahtevaju. Svi oni to prihvataju.

1.7 Nadovezivanje vrednosti kolona

Zadatak

Želite da vratite vrednosti iz više kolona kao jednu kolonu. Na primer, želeli biste da napravite ovaj skup rezultata iz upita prema EMP tabeli:

```
CLARK WORKS AS A MANAGER
KING WORKS AS A PRESIDENT
MILLER WORKS AS A CLERK
```

Međutim, podaci koji su vam potrebni za generisanje ovog skupa rezultata dolaze iz dve različite kolone, ENAME i JOB kolone u EMP tabeli:

```
 select ename, job
    from emp
    where deptno = 10

ENAME      JOB
---------- ---------
CLARK      MANAGER
KING       PRESIDENT
MILLER     CLERK</pre>
```

Rešenje

Pronađite i koristite ugrađenu funkciju koju pruža vaš DBMS za nadovezivanje vrednosti iz više kolona.

DB2, Oracle, PostgreSQL

Ove baze podataka koriste dvostruku vertikalnu traku kao operater nadovezivanja:

```
1 select ename||' WORKS AS A '||job as msg
2   from emp
3  where deptno=10
```

MiSQL

Ova baza podataka podržava funkciju CONCAT:

```
1 select concat(ename, ' WORKS AS A ',job) as msg
2   from emp
3  where deptno=10
```

SQL Server

Koristite + operator za nadovezivanje:

```
1 select ename + ' WORKS AS A ' + job as msg
2   from emp
3  where deptno=10
```

Objašnjenje

Koristite funkciju CONCAT za nadovezivanje vrednosti iz više kolona. Znak || je prečica za funkciju CONCAT u DB2, Oracle i PostgreSQL, dok je + prečica za SQL Server.

1.8 Korišćenje uslovne logike u SELECT izjavi

Zadatak

Želite da izvršite IF-ELSE operacije nad vrednostima u vašem SELECT izrazu. Na primer, želeli biste da napravite skup rezultata tako da ako je zaposleni plaćen 2.000 USD ili manje, vraća se poruka „UNDERPAID" (srp. *slabo plaćen*); ako je zaposlenom plaćeno 4.000 USD ili više, vraća se poruka „OVERPAID" (srp. *preplaćeno*); a ako su plaćeni negde između, vraća se „OK". Skup rezultata trebao bi izgledati ovako:

```
 ENAME            SAL  STATUS
---------- ----------  ---------
SMITH             800  UNDERPAID
ALLEN            1600  UNDERPAID
WARD             1250  UNDERPAID
JONES            2975  OK
MARTIN           1250  UNDERPAID
BLAKE            2850  OK
CLARK            2450  OK
SCOTT            3000  OK
KING             5000  OVERPAID
TURNER           1500  UNDERPAID
ADAMS            1100  UNDERPAID
JAMES             950  UNDERPAID
FORD             3000  OK
MILLER           1300  UNDERPAID</pre>
```

Rešenje

Koristite CASE izraz za izvođenje uslovne logike direktno u vašem SELECT izrazu:

```
1 select ename,sal,
2        case when sal <= 2000 then 'UNDERPAID'
3             when sal >= 4000 then 'OVERPAID'
4             else 'OK'
5        end as status
6   from emp
```

Objašnjenje

CASE izraz vam omogućava da izvodite logiku uslova na vrednostima koje vraća upit. Možete dodati pseudonim za izraz CASE da biste vratili čitljiviji skup rezultata. U rešenju ćete videti pseudonim STATUS dat rezultatu izraza CASE. Klauzula ELSE nije obavezna. Izostavite ELSE i izraz CASE će vratiti NULL za bilo koji red koji ne zadovoljava uslov testa.

1.9 Ograničavanje broja vraćenih redova

Zadatak

Želite da ograničite broj redova vraćenih vašim upitom. Ne zanima vas redosled; bilo kojih *n* redova vam je potrebo.

Rešenje

Koristite ugrađenu funkciju vaše baze podataka za kontrolu broja vraćenih redova.

DB2

U DB2 koristite klauzulu FETCH FIRST:

```
1 select *
2   from emp fetch first 5 rows only
```

MiSQL i PostgreSQL

Uradite isto u MiSQL-u i PostgreSQL-u koristeći LIMIT:

```
1 select *
2   from emp limit 5
```

Oracle

U Oracleu postavite ograničenje na broj vraćenih redova upotrebom ROWNUM u klauzuli WHERE:

```
1 select *
2   from emp
3  where rownum <= 5
```

SQL Server

Koristite rezervisanu reč TOP da biste ograničili broj vraćenih redova:

```
1 select top 5 *
2   from emp
```

Objašnjenje

Mnogi dobavljači pružaju klauzule kao što su FETCH FIRST i LIMIT koje vam omogućavaju da zadate broj redova koji se vraćaju upitom. Oracle je drugačiji, jer morate koristiti funkciju ROWNUM koja vraća broj za svaki vraćeni red (rastuće vrednosti počev od 1).

Evo šta se dešava kada koristite ROWNUM <= 5 za vraćanje prvih pet redova:

1. Oracle izvršava vaš upit.
2. Oracle preuzima prvi red i naziva ga red broj 1.

3. Da li smo prešli red broj 5? Ako nismo, tada Oracle vraća red jer ispunjava kriterijume da bude numerisan manje ili jednako 5. Ako je odgovor da, Oracle ne vraća red.
4. Oracle preuzima sledeći red i povećava broj reda (na 2, zatim na 3, pa na 4 i tako dalje).
5. Ide na korak 3.

Kao što ovaj proces pokazuje, vrednosti Oracleovog ROWNUM dodeljuju se *posle* preuzimanja svakog reda. Ovo je važna i ključna tačka. Mnogi Oracle programeri pokušavaju da vrate samo, recimo, peti red navodeći ROWNUM = 5.

Korišćenje uslova jednakosti u sprezi sa ROWNUM-om je loša ideja. Evo šta se dešava kada pokušate da vratite, recimo, peti red koristeći ROWNUM = 5:

1. Oracle izvršava vaš upit.
2. Oracle preuzima prvi red i naziva ga red broj jedan.
3. Jesmo li već stigli do petog reda? Ako nismo, tada Oracle odbacuje red jer ne ispunjava kriterijume. Ako je odgovor da, Oracle vraća red. Ali odgovor nikada neće biti da!
4. Oracle preuzima sledeći red i naziva ga red broj jedan. To je zato što prvi red koji se vraća iz upita mora biti numerisan sa 1.
5. Ide na korak 3.

Pažljivo proučite ovaj proces i videćete zašto upotreba ROWNUM = 5 za vraćanje petog reda ne uspeva. Ne možete imati peti red ako prvo ne vratite redove jedan do četiri!

Možda ćete primetiti da ROWNUM = 1 u stvari radi na vraćanju prvog reda, što može izgledati u suprotnosti sa dosadašnjim objašnjenjem. Razlog zašto ROWNUM = 1 vraćan prvi red je taj što Oracle mora da pokuša da preuzme barem jednom da bi utvrdio da li postoje redovi u tabeli. Pažljivo pročitajte prethodni postupak, zamenjujući 1 za 5, i razumećete zašto je u redu zadati ROWNUM = 1 kao uslov (za vraćanje jednog reda).

1.10 Vraćanje n nasumičnih zapisa iz tabele

Zadatak

Iz tabele želite da vratite određeni broj nasumičnih zapisa. Sledeću izjavu želite da izmenite tako da će uzastopna izvršavanja proizvesti drugačiji skup od pet redova:

```
select ename, job
from emp
```

Rešenje

Uzmite bilo koju ugrađenu funkciju koju vaš DBMS podržava za vraćanje nasumičnih vrednosti. Koristite tu funkciju u klauzuli ORDER BY da biste nasumično sortirali redove. Zatim, koristite tehniku prethodnog recepta da biste ograničili broj nasumično sortiranih redova koji se vraćaju.

DB2

Koristite ugrađenu funkciju RAND u saradnji sa ORDER BY i FETCH:

```
1 select ename,job
2   from emp
3  order by rand() fetch first 5 rows only
```

MySQL

Koristite ugrađenu funkciju RAND u kombinaciji sa LIMIT i ORDER BY:

```
1 select ename,job
2   from emp
3  order by rand() limit 5
```

PostgreSQL

Koristite ugrađenu funkciju RANDOM u kombinaciji sa LIMIT i ORDER BY:

```
1 select ename,job
2   from emp
3  order by random() limit 5</pre>
```

Oracle

Koristite ugrađenu funkciju VALUE, koja se nalazi u ugrađenom paketu DBMS_RANDOM, zajedno sa ORDER BY i ugrađenom funkcijom ROWNUM:

```
1 select *
2   from (
3  select ename, job
4    from emp
6   order by dbms_random.value()
7        )
8   where rownum <= 5
```

SQL Server

Koristite ugrađenu funkciju NEWID zajedno sa TOP i ORDER BY da biste vratili slučajni skup rezultata:

```
1 select top 5 ename,job
2   from emp
3  order by newid()
```

Objašnjenje

Klauzula ORDER BY može prihvatiti povratnu vrednost funkcije i koristiti je za promenu redosleda skupa rezultata. Sva ova rešenja ograničavaju broj redova za vraćanje *nakon* što se izvrši funkcija u klauzuli ORDER BY. Korisnicima koji nisu Oracle može biti korisno pogledati Oracle rešenje jer ono pokazuje (konceptualno) šta se dešava ispod haube ostalih rešenja.

Važno je da ne pomešate upotrebu funkcije u klauzuli ORDER BY sa upotrebom numeričke konstante. Kada zadate numeričku konstantu u klauzuli ORDER BY, zahtevate da se sortiranje izvrši prema koloni u na tom rednom položaju na listi SELECT. Kada zadate funkciju u klauzuli ORDER BY, sortiranje se vrši na rezultatu iz funkcije, kako se ona izračunava za svaki red.

1.11 Pronalaženje null vrednosti

Zadatak

Želite da pronađete sve redove koji su null u određenoj koloni.

Rešenje

Da biste utvrdili da li je vrednost null, morate koristiti IS NULL:

```
1 select *
2   from emp
3  where comm is null
```

Objašnjenje

NULL nikada nije jednak a nije ni nejednak nečemu, čak ni sebi; stoga ne možete koristiti = ili != za testiranje da li je u koloni NULL. Da biste utvrdili da li red ima NULL vrednosti, morate koristiti IS NULL. Takođe možete da koristite IS NOT NULL da biste pronašli redove bez null u datoj koloni.

1.12 Pretvaranje null u stvarne vrednosti

Zadatak

Imate redove koji sadrže null i želeli bi da vrate nenull vrednosti umesto tih null.

Rešenja

Koristite funkciju COALESCE da biste null zamenili realnim vrednostima:

```
1 select coalesce(comm,0)
2   from emp
```

Objašnjenje

Funkcija COALESCE uzima jednu ili više vrednosti kao argumente. Funkcija vraća prvu nenull vrednost sa liste. U rešenju vrednost COMM se vraća kad god COMM nije null. U suprotnom, vraća se nula (0).

Kada radite sa null, najbolje je iskoristiti ugrađenu funkcionalnost koju pruža vaš DBMS; u mnogim slučajevima naći ćete da nekoliko funkcija jednako dobro radi i ovaj zadatak. COALESCE funkcioniše za sve DBMS-ove. Pored toga, CASE se može koristiti i za sve DBMS-ove:

```
select case
       when comm is not null then comm
       else 0
       end
  from emp
```

Iako možete da koristite CASE za prevođenje nullova u vrednosti, možete videti da je mnogo lakše i sažetije koristiti COALESCE.

1.13 Traženje obrazaca

Zadatak

Želite da vratite redove koji se podudaraju sa određenim podnizom ili šablonom. Uzmite u obzir sledeći upit i skup rezultata:

```
select ename, job
  from emp
 where deptno in (10,20)

ENAME      JOB
---------- ---------
SMITH      CLERK
JONES      MANAGER
CLARK      MANAGER
SCOTT      ANALYST
KING       PRESIDENT
ADAMS      CLERK
FORD       ANALYST
MILLER     CLERK
```

Od zaposlenih u odeljenjima 10 i 20 želite da vratite samo one koji negde u svom imenu imaju ili „I” ili naziv posla koji se završava sa „ER”:

```
ENAME      JOB
---------- ---------
SMITH      CLERK
JONES      MANAGER
CLARK      MANAGER
KING       PRESIDENT
MILLER     CLERK
```

Rešenje

Koristite operator LIKE zajedno sa SQL džoker operatorom (%):

```
1 select ename, job
2   from emp
3  where deptno in (10,20)
4    and (ename like '%I%' or job like '%ER')
```

Objašnjenje

Kada se LIKE koristi u operaciji podudaranja šablona, operator procenat (%) se podudara sa bilo kojim nizom znakova. Većina SQL implementacija takođe obezbeđuje donju crtu ("_") za podudaranje sa jednim znakom. Zatvaranjem šablona za pretragu „I" sa operatorom %, biće vraćen bilo koji niz koji sadrži „I" (na bilo kojoj poziciji). Ako šablon pretraživanja ne zatvorite sa %, tada će mesto na kojem postavite operator uticati na rezultate upita. Na primer, da biste pronašli naslove poslova koji se završavaju sa „ER", dodajte operatoru % ispred „ER"; ako je zahtev da se pretražuju svi poslova koji počinju sa „ER", dodajte operator % iza „ER".

1.14 Rezime

Ovi recepti su možda jednostavni, ali su i osnovni. Nalaženje informacija je srž upita za baze podataka, a to znači da su ovi recepti u osnovi gotovo svega o čemu se raspravlja u knjizi.