7.6.6. Upotreba odredbe GROUP BY

Ova odredba omogućava grupisanje učitanih redova. Ona je korisna samo kada se upotrebi u kombinaciji sa funkcijama koje deluju na grupe redova (agregatne funkcije: MIN(), MAX(), SUM(), AVG(), COUNT(), …). Upit:

```
SELECT COUNT(*) AS broj_faktura, sifra_firme
FROM fakture
GROUP BY sifra_firme;
```

daje sledeće razultate:

broj_faktura	sifra_firme
3	1
2	2
3	3
3	4

Ovaj upit prebrojava fakture po firmama – tj. za svaku firmu utvrđuje broj faktura.
MySQL omogućava da se izabere redosled grupa kojim se prikazuju rezultati. Podrazumevani je rastući redosled. Sledeći upit je isti kao prethodni, ali se rezultati prikazuju opadajućim redosledom:

```
SELECT COUNT(*) AS broj_faktura, sifra_firme
FROM fakture
GROUP BY sifra_firme DESC;
```

broj_faktura	sifra_firme
3	4
3	3
2	2
3	1

Može se zadati opcija ASC (od *ascending*, rastući redosled), ali pošto se taj redosled podrazumeva nema potrebe da se izričito zadaje.
Izvršavanjem upita:

```
SELECT faktura, MIN(kolicina)
FROM detalji_fakture
GROUP BY faktura;
```

dobijaju se sledeći rezultati:

faktura	MIN(kolicina)
1	50.00
2	35.00
3	20.00
4	50.00
5	25.00
6	24.00
7	26.00
8	45.00
9	14.00
10	60.00
11	5.00

Ovaj upit nalazi minimalnu količinu na svakoj fakturi. Funkcija MIN() vraća minimalnu vrednost iz grupe prosleđenih vrednosti.

7.6.7. Izdvajanje određenih grupa podataka pomoću opcije HAVING

Odredba GROUP BY kojoj je dodata odredba HAVING deluje na sličan način kao komanda SELECT kojoj je dodata odredba WHERE. Izvršavanjem upita:

```
SELECT faktura, SUM(kolicina * dan_cena) AS iznos
FROM detalji_fakture
GROUP BY faktura
HAVING SUM(kolicina * dan_cena) > 10000;
```

dobijaju se sledeći rezultati:

faktura	iznos
2	12150.0000
3	32500.0000
4	13500.0000
5	19400.0000
6	45670.0000
8	15425.0000
10	81050.0000

Ovaj upit prikazuje sve fakture čiji je ukupan iznos veći od 10000. Iznos za svaku stavku na fakturi je *kolicina* * *dan_cena*. Ukupan iznos na fakturi je suma iznosa za sve stavke, tj. SUM(*kolicina* * *dan_cena*). Funkcija SUM() vrši sumiranje vrednosti.
Treba praviti razliku između odredbi WHERE i HAVING. Odredba WHERE se može upotrebiti u gotovo svakom upitu da bi se zadao uslov koji se odnosi na pojedinačne redove. Odredba HAVING se koristi kada određeni uslov treba da važi za celu grupu. U odredbi WHERE se ne mogu koristiti agregatne funkcije, a u odredbi HAVING mogu.

7.6.10. Upotreba spojeva u upitima koji obuhvataju više tabela

Može se proanalizirati sledeći upit:

```
SELECT fakture.sifra_fakture, firme.naziv_firme, fakture.datum, fakture.ulaz_izlaz
FROM fakture, firme
WHERE fakture.sifra_firme = firme.firma;
```

U ovom slučaju žele se prikazati podaci sa svih faktura, ali se umesto šifre firme za svaku fakturu želi prikazati naziv firme.
Izvršavanjem ovog upita dobijaju se sledeći rezultati:

sifra_fakture	naziv_firme	datum	ulaz_izlaz
3	BALKAN	2006-10-25	2
6	BALKAN	2006-11-06	2
9	BALKAN	2006-11-02	2
1	STIL	2006-10-25	1
8	STIL	2006-10-23	1
4	KOKOMAX	2006-10-29	2
5	KOKOMAX	2006-10-25	1
11	KOKOMAX	2006-10-15	1
2	HELIO	2006-10-28	1
7	HELIO	2006-11-04	2
10	HELIO	2006-10-08	1

Može se videti da su iza SELECT komande zadate kolone koje postoje u različitim tabelama. Korišćena su apsolutna imena kolona zbog razumljivosti. Da se je desilo da u ove dve tabele postoje kolone sa istim nazivom, a u rezultatima se žele prikazati vrednosti iz obe kolone, apsolutna imena kolona bi bila neophodna. Da bi sve ovo funkcionisalo bilo je neophodno da se iza odredbe FROM navedu nazivi obe tabele.
Najzanimljiviji deo ovog upita je odredba WHERE. Ako se ovaj upit izvrši bez odredbe WHERE u sledećem obliku:

SELECT *fakture*.*sifra_fakture*, *firme*.*naziv_firme*, *fakture*.*datum*, *fakture*.*ulaz_izlaz*
FROM *fakture*, *firme*;

dobiće se sledeći rezultati:

sifra_fakture	naziv_firme	datum	ulaz_izlaz
1	BALKAN	2006-10-25	1
1	STIL	2006-10-25	1
1	KOKOMAX	2006-10-25	1
1	HELIO	2006-10-25	1
2	BALKAN	2006-10-28	1
2	STIL	2006-10-28	1
2	KOKOMAX	2006-10-28	1
2	HELIO	2006-10-28	1
3	BALKAN	2006-10-25	2
3	STIL	2006-10-25	2
3	KOKOMAX	2006-10-25	2
3	HELIO	2006-10-25	2
4	BALKAN	2006-10-29	2
4	STIL	2006-10-29	2
4	KOKOMAX	2006-10-29	2
4	HELIO	2006-10-29	2
5	BALKAN	2006-10-25	1
5	STIL	2006-10-25	1
5	KOKOMAX	2006-10-25	1
5	HELIO	2006-10-25	1
6	BALKAN	2006-11-06	2
6	STIL	2006-11-06	2
6	KOKOMAX	2006-11-06	2
6	HELIO	2006-11-06	2
7	BALKAN	2006-11-04	2
7	STIL	2006-11-04	2
7	KOKOMAX	2006-11-04	2
7	HELIO	2006-11-04	2
8	BALKAN	2006-10-23	1
8	STIL	2006-10-23	1
8	KOKOMAX	2006-10-23	1
8	HELIO	2006-10-23	1
9	BALKAN	2006-11-02	2
9	STIL	2006-11-02	2
9	KOKOMAX	2006-11-02	2
9	HELIO	2006-11-02	2
10	BALKAN	2006-10-08	1
10	STIL	2006-10-08	1
10	KOKOMAX	2006-10-08	1
10	HELIO	2006-10-08	1
11	BALKAN	2006-10-15	1
11	STIL	2006-10-15	1
11	KOKOMAX	2006-10-15	1

Prvi upit, kojem je pridružena odredba WHERE, prikazuje podatke sa svake fakture sa tačnim podatkom za naziv firme, a drugi upit prikazuje sve kombinacije faktura i firmi pri čemu nije moguće utvrditi koji redovi rezultata sadrže tačne podatke, a koji su besmisleni.

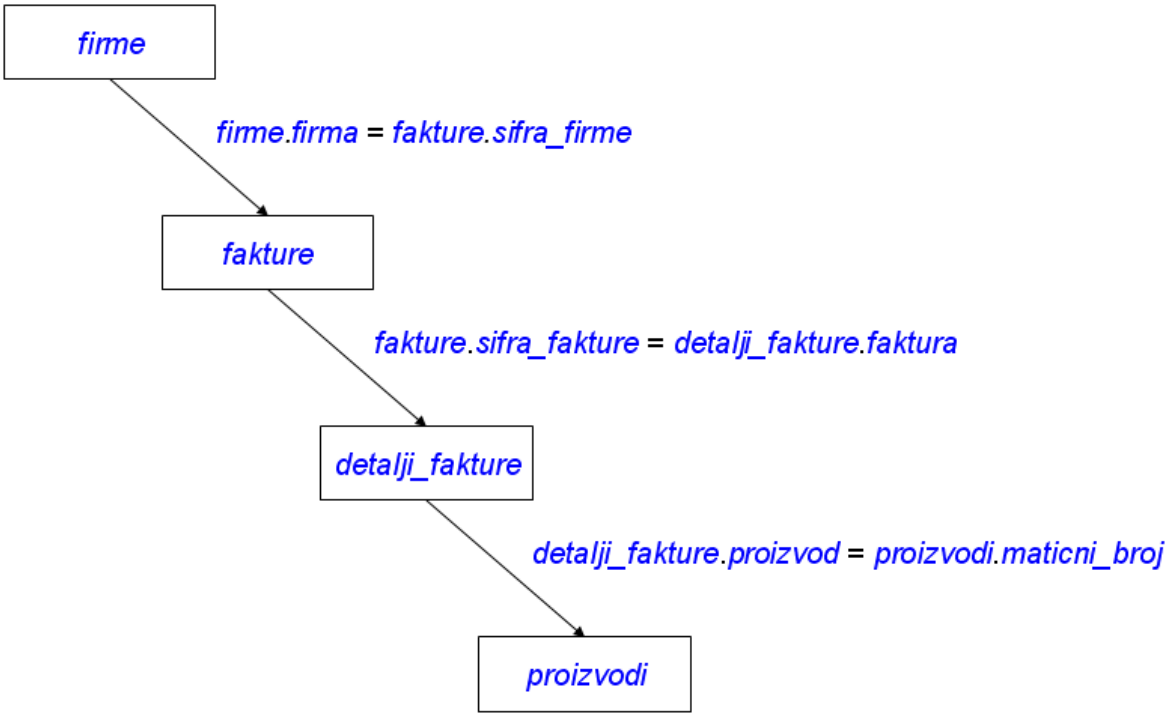
Ovaj skup rezultata koji se sastoji od svih mogućih kombinacija rezultata iz dve tabele, zove se Dekartov proizvod (*Cartesian product*) dveju tabela.
Sasvim je očigledno da je odredba WHERE ključna za dobijanje željenih rezultata. Kada se u upitu spajaju dve tabele, uslov ili grupa uslova pomoću kojih se povezuju dve tabele zove se **spojni uslov**. U konkretnom slučaju uslov je: *fakture*.*sifra_firme* = *firme*.*firma*, što je veza između tabela koja je definisana preko ključeva još prilikom definisanja strukture baze podataka.
Kada se želi da se istovremeno učitaju podaci koji se nalaze u više tabela, moraju se upotrebiti veze između tih tabela.
Spajanje više tabela se ne razlikuje od spajanja samo dve tabele.
Recimo da treba pronaći sve proizvode koji su kupljeni od ili prodati firmi KOKOMAX.
Pošto se zna naziv firme, u koloni *firma* tabele *firme* se može naći njena šifra. Pomoću tog podatka mogu se naći šifre svih faktura koje su formirane za tu firmu (kolona *sifra_fakture* tabele *fakture*). Pomoću šifri faktura u tabeli *detalji_fakture* se mogu naći matični brojevi proizvoda za svaku stavku svake fakture (kolona *proizvod* tabele *detalji_fakture*). Na osnovu matičnih brojeva proizvoda moguće je naći nazive proizvoda u tabeli *proizvodi* (kolona *ime* tabele *proizvodi*). Potrebno je samo na kraju prikazati različite proizvode korišćenjem ključne reči DISTINCT.
Upit bi izgledao ovako:

```
SELECT DISTINCT proizvodi.ime AS naziv_proizvoda
FROM firme, fakture, detalji_fakture, proizvodi
WHERE firme.naziv_firme = 'KOKOMAX'
AND firme.firma = fakture.sifra_firme
AND fakture.sifra_fakture = detalji_fakture.faktura
AND detalji_fakture.proizvod = proizvodi.maticni_broj;
```

Izvršavanjem upita dobijaju se sledeći rezultati:

naziv_proizvoda
Gips
Četka
Destilovana voda
Šmirgla

U upitu se vidi da je bilo neophodno navesti sve tabele u putanji koja je sleđena i zadati spojne uslove koji povezuju jednu tabelu sa drugom. U ovom slučaju postoji jedan običan uslov (*firme*.*naziv_firme* = 'KOKOMAX') i više spojnih uslova. Treba primetiti da su spojene četiri tabele pomoću tri spojna uslova.
Kada se spaja n tabela, u većini slučajeva, biće potrebna po jedna veza između svakog para tabela, što znači da će postojati n-1 spojnih uslova. Spojevi definisani u ovom primeru prikazani su na slici.



Kao što se jedna tabela može spojiti sa drugom, tako se može spojiti i sa samom sobom. Ovakav način spajanja će se koristiti kada se traže veze između redova u istoj tabeli. Recimo da žele da se pronađu sve firme iz istog mesta kao i firma BALKAN. Da bi se došlo do ovih podataka prvo u tabeli *firme* za firmu BALKAN treba pronaći vrednost u koloni *mesto*, a zatim u istoj tabeli treba pronaći i sve ostale firme koje u koloni *mesto* imaju istu vrednost.
Upit bi izgledao ovako:

```
SELECT f2.naziv_firme
FROM firme AS f1, firme AS f2
WHERE f1.naziv_firme = 'BALKAN'
AND f1.mesto = f2.mesto;
```

Izvršavanjem upita dobijaju se sledeći rezultati:

naziv_firme
BALKAN
KOKOMAX

Kao što se vidi, u ovom upitu deklarisana su dva različita alijasa za tabelu *firme*. Time je rečeno MySQL-u da će se raditi kao da postoje dve zasebne tabele, *f1* i *f2*, koje slučajno sadrže iste podatke. Zatim su te tabele bile spojene na isti način kao što bi se to uradilo sa bilo kojim drugim dvema tabelama. Najpre je u tabeli *f1* potražen red u kome je vrednost u koloni *naziv_firme* BALKAN (tj. u kome je ispunjen uslov *f1.naziv_firme* = 'BALKAN'), a zatim su u tabeli *f2* pronađeni redovi koji u koloni *mesto* sadrže istu vrednost kao pronađen red u tabeli *f1* u istoj koloni (*mesto*).
Ovde je samo potrebno zamisliti da se radi sa dve tabele.
Vidi se da se u rezultatima prethodnog upita nalaze sve firme koje su iz istog mesta kao i firma BALKAN, ali se nalazi i firma BALKAN. Upitu se može dodati novi uslov koji će firmu BALKAN isključiti iz skupa rezultata:

```
SELECT f2.naziv_firme
```

```
FROM firme AS f1, firme AS f2
WHERE f1.naziv_firme = 'BALKAN'
AND f1.mesto = f2.mesto
AND f2.naziv_firme != 'BALKAN';
```

naziv_firme
KOKOMAX

7.6.11. Vrste spojeva između tabela

Dekartov proizvod se ponekad naziva **punim spojem** (*full join*) ili **unakrsnim spojem** (*cross join*), ali bez obzira na ime sastoji se od svih mogućih kombinacija redova tabela. Kada se tom spoju doda određeni uslov (kao što je bio *fakture.sifra_firme* = *firme.firma*) dobija se nešto što se ponekad naziva **jednakovredni spoj** (*equijoin*), koji ograničava broj redova u skupu rezultata.
Do sada je u odredbi FROM zadavana lista tabela koje su razdvojene zarezima. Time se dobija unakrsni spoj, koji se pretvara u jednakovredni spoj kada mu se doda odredba WHERE. MySQL podržava više oblika sintakse za ovu vrstu spoja.
Izvorni upit je glasio:

```
SELECT fakture.sifra_fakture, firme.naziv_firme, fakture.datum, fakture.ulaz_izlaz
FROM fakture, firme
WHERE fakture.sifra_firme = firme.firma;
```

Umesto zareza može se zadati neobavezna rezervisana reč JOIN:

```
SELECT fakture.sifra_fakture, firme.naziv_firme, fakture.datum, fakture.ulaz_izlaz
FROM fakture JOIN firme
WHERE fakture.sifra_firme = firme.firma;
```

Osim reči JOIN može se zadati i CROSS JOIN ili INNER JOIN.
Kada se zada ova vrsta spoja, MySQL pretražuje sve tabele koje su zadate i pokušava da pronađe najefikasniji način spajanja, pri čemu ne spaja tabele obavezno redom koji je naveden. Ako se želi da se naloži MySQL-u da spoji tabele redosledom koji je naveden treba reč JOIN zameniti rečima STRAIGHT_JOIN.
Da bi bile opisane vrste spojeva između tabela, tabeli *firme* biće dodat novi red, pri čemu za novododatu firmu neće postojati podaci za fakture i detalje faktura. Iskaz bi izgledao ovako:

```
INSERT INTO firme VALUES
 (5, 'ADRIA', 'Beograd', 'Ustanička 39', '011-224-299');
```

Ako se želi da se pronađu nazivi firmi za koje nema formiranih faktura, potrebno je iskoristiti levi spoj, odnosno operator LEFT JOIN, na sledeći nečin:

```
SELECT firme.naziv_firme
FROM firme LEFT JOIN fakture
ON firme.firma = fakture.sifra_firme
WHERE sifra_fakture IS NULL;
```

Izvršavanjem upita dobijaju se sledeći rezultati:

naziv_firme
ADRIA

Ako se pogleda sadržaj tabela lako se može videti da su rezultati tačni.
Levi spoj radi tako što za sve redove tabele na levoj strani spoja (u ovom primeru to je tabela *firme*) traži odgovarajuće redove u tabeli na desnoj strani spoja. Pronađeni redovi se postavljaju pored leve tabele. Za svaki red iz leve tabele koji nema parnjaka u desnoj tabeli, operator LEFT JOIN dodaje red vrednosti NULL. Redovi iz leve tabele bez parnjaka u desnoj se mogu naći ako se zada uslov da je vrednost primarnog ključa u desnoj tabeli NULL. Ako se izvrši sledeći upit:

SELECT *firme*.*naziv_firme*, *firme*.*mesto*, *firme*.*adresa*, *fakture*.*sifra_fakture*, *fakture*.*datum*, *fakture*.*ulaz_izlaz*
FROM *firme* LEFT JOIN *fakture*
ON *firme*.*firma* = *fakture*.*sifra_firme*;

dobijaju se sledeći rezultati:

naziv_firme	mesto	adresa	sifra_fakture	datum	ulaz_izlaz
BALKAN	Niš	Mokranjčeva 13	3	2006-10-25	2
BALKAN	Niš	Mokranjčeva 13	6	2006-11-06	2
BALKAN	Niš	Mokranjčeva 13	9	2006-11-02	2
STIL	Beograd	Takovska 10	1	2006-10-25	1
STIL	Beograd	Takovska 10	8	2006-10-23	1
KOKOMAX	Niš	Dušanova 33	4	2006-10-29	2
KOKOMAX	Niš	Dušanova 33	5	2006-10-25	1
KOKOMAX	Niš	Dušanova 33	11	2006-10-15	1
HELIO	Subotica	Nikole Tesle 55	2	2006-10-28	1
HELIO	Subotica	Nikole Tesle 55	7	2006-11-04	2
HELIO	Subotica	Nikole Tesle 55	10	2006-10-08	1
ADRIA	Beograd	Ustanička 39	NULL	NULL	NULL

U ovom primeru upotrebljen je operator LEFT JOIN, ali isto tako je mogao da bude upotrebljen i operator RIGHT JOIN, koji deluje na isti način, s tom razlikom što je desna tabela osnova, a nedostajući redovi iz tabele sa leve strane dopunjuju se vrednostima NULL.