Upiti nad bazom podataka

Upiti predstavljaju različite oblike naredbe *SELECT* i mogu da budu različite složenosti. Naredba *SELECT* uključuje veliki broj klauzula, koje određuju podatke koji žele da se dobiju. Važna činjenica je da se naredba *SELECT* može da koristi u ostalim DML naredbama.

Skraćena verzija naredbe *SELECT*:

```
SELECT [DISTINCT] [<kvalifikator>.]<ime_kolone> |*|<izraz>
                [AS <alias>],...
FROM  <ime_tabele_ili_pogleda> |
     <upit>
      [[AS] <alias>]
[WHERE <predikat>]
[GROUP BY [<qualifier>.]< ime_kolone>,...
 [HAVING <predikat>]
]
[ORDER_BY < ime_kolonee> |
         <broj_kolone>
          [ASC | DESC],...
];
```

Kratka interpretacija ove naredbe:

Iza ključne reči *SELECT* navode se imana kolona iz odgovarajućih tabela ili pogleda, zatim izrazi (na primer, kolona3*0.25, ili zbir kolone sum(kolona2)). Ključna reč *DISTINCT* eliminiše ponavljanje vrsta u rezultatu. Alias je drugo ime za tabelu. Kvalifikator je potreban u različitim slučajevima, naprimer, kada su nazivi kolona isti u različitim tabelama.
Iza ključne reči *FROM* se navode imena tabela ili pogleda iz kojih potiču kolone koje su navedene. To može da bude i upit.
Klauzula *WHERE* sa odgovarajućim predikatima definiše uslov pretraživanja u cilju dobijanja traženog rezultata.
Klauzula *GROUP BY* omogućava grupisanje vrsta, a klauzula *HAVING* postavljanje dodatnog uslova za klauzulu *GROUP BY*, odnosno prikaz rezultata.
Klauzula *ORDER BY* omogućava sortiranje podataka koje može da bude u rastućem ili opadajućem redosledu po jednoj ili po više kolona.

Prikazati sve nastavnike:

```
SELECT * FROM Nastavnici;
```

```
Snast  Imen
------ ------------------------
1      Petrović
2      Tomić
3      Marić
4      Marković
5      Nikolić
6      Lazić
7      Janković
8      Lukić
9      Ristić
```

10 Radović
11 Milić
12 Mitrović
13 Božić

Prikazati sva pojavljivanja mesta iz kojih su studenti (svaki student):

```
SELECT Mesto FROM Studenti;
```

```
Mesto
------------------------------
Kragujevac
Kragujevac
Jagodina
Lapovo
Kragujevac
Ćuprija
Kragujevac
Batočina
Kraljevo
Kruševac
Paraćin
Kragujevac
Užice
Čačak
Kragujevac
Rača
Aranđelovac
Kraljevo
Smederevo
Čačak
Kragujevac
Užice
Kragujevac
Jagodina
Kraljevo
Jagodina
Paraćin
```

Prikazati mesta iz kojih su studenti (samo različita mesta):

```
SELECT DISTINCT Mesto FROM Studenti;
```

```
Mesto
------------------------------
Aranđelovac
Batočina
Čačak
Ćuprija
Jagodina
Kragujevac
Kraljevo
Kruševac
Lapovo
Paraćin
Rača
Smederevo
Užice
```

```
SELECT Indeks, Upisan, Imes, Mesto FROM Studenti
WHERE Upisan = 2003
```

Indeks	Upisan	Imes	Mesto
5	2003	Mira	Kragujevac
8	2003	Jovan	Aranđelovac
16	2003	Tanja	Kragujevac
17	2003	Zoran	Kraljevo
18	2003	Saša	Jagodina

Prikazati broj indeksa, godinu upisa, ime studenta, smer na kome studiraju, srednju ocenu, najveću ocenu, najmanju ocenu i broj ocena pod uslovom svakog studenta koji je iz Kragujevca i koji ima najmanje 8 ocena.

```
SELECT s.Indeks, s.Upisan, Imes, Ssmer, AVG(Ocena), MAX(Ocena), MIN(Ocena),
COUNT(Ocena) FROM Studenti AS s, Prijave AS p
WHERE Mesto = 'Kragujevac'AND s.Indeks = p.Indeks AND s.Upisan = p.Upisan
GROUP BY Ssmer, Imes, s.Indeks, s.Upisan
HAVING COUNT(Ocena) > 7
ORDER BY Ssmer, s.Upisan, s.Indeks
```

Indeks	Upisan	Imes	Ssmer				
3	2000	Sava	1	7	9	5	19
1	2002	Nenad	1	7	9	5	15
1	2000	Ana	2	6	9	5	18
3	2002	Toma	4	8	10	7	18

Naredba za izvršavanje upita SELECT

Naredba *SELECT* nalazi (vadi, donosi) vrste, kolone i izvedene vrednosti iz jedne ili više tabela baze podataka.

Naredba *SELECT* je podržana, sa varijacijama, od svih proizvođača DBMS_a. Osnova naredbe je iskazana standardom SQL2003:

SELECT [{ALL | DISTINCT}] *select_item* [AS *alias*] [,...]

FROM [ONLY | OUTER]

{*table_name* [[AS] *alias*] | *view_name* [[AS] *alias*]} [,...]

[[*join_type*] JOIN *join_condition*]

[WHERE *search_condition*] [{AND | OR | NOT} *search_condition* [...]]

[GROUP BY *group_by_expression*{*group_by_columns* | ROLLUP *group_by_columns* |

CUBE *group_by_columns* | GROUPING SETS (*grouping_set_list*) |

( ) | *grouping_set* , *grouping_set_list*}

[HAVING *search_condition*]]

[ORDER BY {*order_expression* [ASC | DESC]} [,...]]

Ključne reči

Svaka od ključnih reči koje se prikazuju izuzev klauzule *select_item*, će se kasnije razmatrati sa više detalja.

[ALL | DISTINCT] *select_item*

> Vadi vrednosti koje sačinjavaju rezultujući skup upita. *select_item* može da bude literal, agregatna ili skalarna funkcija, matematičko izračunavanje, parametar ili promenljiva ili podupit, ali *select_item* je u osnovi najčešće kolona iz tabele ili upita(podsetimo se da je u osnovi pogleda upit). Zarez mora da odvoji svaku stavku u listi takvih stavki.

Šema ili ime vlasnika trebalo bi da bude prefiks (kvalifikator) za kolonu kada je izvučena iz konteksta izvan aktuelnog korisnika. Ako je tabela posedovana od drugog korisnika, onda taj korisnik mora da bude uključen u pozivanje kolone. Na primer, predpostavimo da korisnik Ivan treba da pristupi podacima u šemi Marko:

SELECT Indeks, Upisan FROM Marko.Studenti

Može da se koristi zvezdica (*) stenogrfski, kao skraćenica (džoker), za nalaženje svih kolona u svakoj tabeli ili pogledu koji su prikazani u klauzuli *FROM*. Dobra je praksa da se koristi ova skraćenica samo u upitima na jednoj tabeli.

ALL, podrazumevano ponašanje, vraća sve slogove koji zadovoljavaju kriterijum selekcije. *DISTINCT* govori bazi podataka da filtrira bilo koje duple vrste, tako da daje samo jednu instancu od više identičnih vrsta.

AS *alias*

> Zamenjuje zaglavlje kolone (kada je u klauzuli *select_item*) ili ime tabele ili ime pogleda (kada je u klauzuli *FROM*) sa kraćim zaglavljem ili imenom. Ova klauzula je posebno korisna za zamenu kriptovanih ili dugačkih imena sa kraćim, jasnim imenima za razumevanje ili simbolima, ili kada je kolona samo izveden podatak. Na taj način može da se izbegne ime kolone, na primer, ORA000189x7/0.02.To je takođe vrlo korisno u self-joins i korelacionim podupitima gde jedan upit poziva istu tabelu više od jedanput. Kada se pojavljuje više stavki u klauzuli *select_item* ili klauzuli *FROM*, proverite postavljanje zareza posle klauzula *AS alias*. Treba biti oprezan i uvek koristiti alias ujednačeno sa njenim prvim uvođenjem u upit.

FROM *table_name*

Služi za nabrajanje svih tabela i/ili pogleda iz koji upit nalazi podatke. Odvajanje tabela i pogleda se vrši korišćenjem zareza. Klauzula *FROM*, takođe, omogućava da se dodele usvojena imena (*alias*) umesto dužim imenima tabela/ili pogleda ili podupita korišćenjem kaluzule *AS*. Korišćenje kraćih usvojenih imena umesto dužih imena tabela ili pogleda pojednostavljuje kodiranje. (Naravno, ovo bi moglo da sukobi pažljivo planiranim konvencijama imenovanja DBA, ali usvojena imena ostaju samo za vreme trajanja upita. Klauzula *FROM* može da sadrži podupit.

ONLY

Specificira da će se u rezultujućem skupu naći samo vrste imenovane tabele ili pogleda i vrste u podtabelama ili pod pogledima. Kada se koristi *ONLY*, proverite da li ste stavili *table_name* ili *view_name* u okviru zagrada. *ONLY* se zanemaruje ako tabela ili pogled nemaju pod tabele ili pod poglede.

OUTER

Kada se dve tabele spajaju, prava tabela (koju nazivamo levom), može da ima vrste koje se ne uparuju sa vrstama u drugoj tabeli (koju nazivamo desnom). Obrnuto, tabela sa desne strane može da ima vrste koje se ne uparuju sa vrstama u levoj tabeli. Kod unutrašnjeg upita, u rezultatu se pojavljuju samo uparene vrste. Spoljašnji upiti ne isključuju ne uparene vrste. Postoje tri vrste spoljašnjeg spajanja: *levo spoljašnje spajanje* (left outer join), *desno spoljašnje spajanje* (right outer join) i *potpuno spoljašnje spajanje* (full outer join). U upitu koji sadrži spajanje, leva tabela je ona koja prethodi ključnoj reči *JOIN*. *Levo spoljašnje spajanje* uključuje sve ne uparene vrste leve tabele, a eliminiše ne uparene vrste desne tabele. *Desno spoljašnje spajanje* uključuje sve ne uparene vrsta desne tabele, a eliminiše ne uparene vrste leve tabele. *Potpuno spoljašnje spajanje* kombinuje funkcije *levog spoljašnjeg spajanja* i *desnog spoljašnjeg spajanja*. Ono sadrži neuparene vrste i leve i desne tabele.

JOIN *join_condition*

Relacioni operator *JOIN* omogućava kombinovanje podataka iz više tabela u jednu tabelu. Zajednički rezultujući skup predstavljaju odabrane kolone iz tabela navedenih u klauzuli *FROM*, pri čemu ove tabele imaju vezu koja se bazira na zajedničkom skupu vrednosti određenih kolona tabela navedenih u *OUTER* klauzuli. Ove kolone, ili moguće jedna kolona iz svake tabele, se nazivaju ključ spajanja ili zajednički ključ. Većinom, ali ne uvek, ključ spajanja je primarni ključ jedne tabele i spoljašnji ključ u drugoj tabeli. Ako se podaci u kolonama slažu, spajanje može da se izvrši. (zapazimo da spajanja takođe mogu da se izvrše korišćenjem klauzule *WHERE*. Ova tehnika se ponekad naziva teta spajanje.)

Join_conditions se najčešće uobiučajeno pojavljuje u obliku:

JOIN *table_name2* ON *table_name1.column1* <*comparison operator*>

table_name2.column2

JOIN *table_name3* ON *table_name1.columnA* <*comparison operator*>

table_name3.columnA

[...]

Kada je *comparison_operator* znak jednakosti (=), kaže se da je spajanje ekvi spajanje. Naravno, operator poređenja bi mogao da bude bilo koji prihvatljiv operator poređenja kao <, >, <=, >= ili <>.

Koristite operator *AND* za izdavanje višestrukih uslova. Može da se koristi operator *OR* za specificiranje alternatvnih uslova spajanja.

Ako se eksplicitno ozosatavi *join_type*, onda se podrazumeva *INNER JOIN*. Zapazimo da postoje mnogi tipovi spajanja, svako sa svojim sopstvenim pravilima. Takođe, zapazimo da postoji alternativni pristup uslovu spajanja preko klauzule *USING*:

USING (*column_name* [,...])

Radi kao alternativa klauzuli *ON*. Umesto opisivanja uslova spajanja, jednostavno obezbeđuje *column_name* (ili imena kolona odvojena sa zarezima,) koje se pojavljuje u obe tabele. Baza podataka tada izračunava spajanje, koje se bazira na koloni (ili kolonama) *column*_name, koja se pojavljuje u obe tabele. (imena kolona moraju da budu identična u obe tabele.).

Prikazati broj indeksa, godinu upisa, šifru predmeta i ocenu svakog upisanog studenta. U ovom primeru, dva upita proizvode iste rezultate:

```
SELECT s.Indeks, s.Upisan, p.Spred, p.Ocena FROM Studenti AS s LEFT OUTER JOIN Prijave AS p ON s.Indeks = p.Indeks AND s.Upisan = p.Upisan

SELECT s.Indeks, s.Upisan, p.Spred, p.Ocena from Studenti AS s LEFT OUTER JOIN Prijave AS p USING(Indeks, Upisan)
```

```
Indeks Upisan Spred  Ocena
------ ------ ------ ------
1      2000   1      7
1      2000   2      5
1      2000   2      6
1      2000   3      5
1      2000   3      8
1      2000   4      7
1      2000   5      8
1      2000   6      7
1      2000   8      9
1      2000   9      7
1      2000   10     5
1      2000   10     6
1      2000   11     8
1      2000   12     7
1      2000   13     6
1      2000   14     7
1      2000   15     9
1      2000   26     7
```

1	2002	1	8
1	2002	2	7
1	2002	17	9
1	2002	3	8
1	2002	4	7
1	2002	5	5
1	2002	5	8
1	2002	6	8
1	2002	7	7
1	2002	8	9
1	2002	9	7
1	2002	10	8
1	2002	11	7
1	2002	12	9
1	2002	13	8
2	2001	1	10
2	2001	2	9
2	2001	17	10
2	2001	3	9
2	2001	4	8
2	2001	5	10
2	2001	6	9
2	2001	7	9
2	2001	8	9
2	2001	9	10
2	2001	18	10
2	2001	11	9
2	2001	12	9
2	2001	19	10
2	2001	21	10
2	2001	22	9
2	2001	23	9
2	2001	24	10
2	2002	1	7
2	2002	2	7
2	2002	17	8
2	2002	3	5
2	2002	4	5
2	2002	3	7
2	2002	4	6
2	2002	5	8
2	2002	6	7
2	2002	7	8
2	2002	8	9
2	2002	9	7
2	2002	10	7
2	2002	11	8
2	2002	12	7
2	2002	13	6
3	2000	1	7
3	2000	2	8
3	2000	17	9
3	2000	3	7
3	2000	4	8
3	2000	5	7
3	2000	6	5
3	2000	6	7
3	2000	7	8
3	2000	8	7
3	2000	9	8

3	2000	10	9
3	2000	12	7
3	2000	13	8
3	2000	11	8
3	2000	19	8
3	2000	21	7
3	2000	23	7
3	2000	15	9
3	2001	1	8
3	2001	2	8
3	2001	17	9
3	2001	3	8
3	2001	4	8
3	2001	5	9
3	2001	6	9
3	2001	7	8
3	2001	8	8
3	2001	9	9
3	2001	10	10
3	2001	11	9
3	2001	12	10
3	2001	13	9
3	2001	14	8
3	2001	15	10
3	2001	20	10
3	2002	1	7
3	2002	2	8
3	2002	17	9
3	2002	3	8
3	2002	4	8
3	2002	5	8
3	2002	6	8
3	2002	7	9
3	2002	8	8
3	2002	9	9
3	2002	10	9
3	2002	19	10
3	2002	11	8
3	2002	12	9
3	2002	13	10
3	2002	21	10
3	2002	15	10
3	2002	23	9
4	2000	1	8
4	2000	2	9
4	2000	17	8
4	2000	3	9
4	2000	4	9
4	2000	5	10
4	2000	6	9
4	2000	7	10
4	2000	8	9
4	2000	9	10
4	2000	18	9
4	2000	11	9
4	2000	12	10
4	2000	19	10
4	2000	21	9
4	2000	22	10
4	2000	23	10

```
4    2000  24    9
4    2001  1     7
4    2001  17    8
4    2001  2     7
4    2001  3     8
4    2001  4     7
4    2001  5     7
5    2001  NULL  NULL
5    2002  NULL  NULL
5    2003  1     7
5    2003  2     7
5    2003  3     5
5    2003  3     7
5    2003  4     8
5    2003  6     8
6    2001  1     7
6    2002  NULL  NULL
7    2001  NULL  NULL
8    2001  NULL  NULL
8    2003  7     9
8    2003  8     10
8    2003  9     9
8    2003  10    10
8    2003  19    10
8    2003  11    9
8    2003  1     9
8    2003  2     8
8    2003  3     9
8    2003  4     9
9    2000  NULL  NULL
9    2001  NULL  NULL
12   2000  NULL  NULL
14   2000  NULL  NULL
15   2002  NULL  NULL
16   2003  NULL  NULL
17   2000  NULL  NULL
17   2003  NULL  NULL
18   2003  NULL  NULL
99   2000  17    8
```

Vidimo da smo spoljašnjim spajanjem dobili I podatak o studentima koji nisu položili još ni jedan predmet.

WHERE *search_condition*

> Filtrira neželjene podatke iz rezultujućeg skupa upita, vraćajući samo one vrste koje zadovoljavaju uslove pretraživanja. Loše pisanje klauzule *WHERE* može da upropasti performance inače korisne naredbe *SELECT*, tako da nijanse korišćenja kaluzule WHERE moraju da se potpuno savladaju. *Search_conditions* se sintaktički prikazuje u obliku:

WHERE [*schema.*[*table_name.*]]*column operator value*

Klauzule *WHERE* obično porede vrednosti koje se nalaze u *column* tabele. Vrednosti kolona se porede korišćenjem operatora određenog tipa. Na primer, kolona bi mogla da bude jednaka (=) datoj vrednosti, da bude veća od (>) date vrednosti ili da bude između opsega vrednosti.

Klauzule *WHERE* mogu da sadrže više uslova pretraživana zajedno, konkateniranih korišćenjem Boolean operatora *AND* ili *OR*. Zagrade mogu da se koriste radi uređenja prioriteta uslova pretraživanja.

GROUP BY *group_by_expression*
Koristi se u upitima koje upotrebljavaju agregatne funkcije kao što su *AVG*, *COUNT*, *COUNT DISTINCT*, *MAX*, *MIN* i *SUM* za grupisanje rezultujućih skupova u gategorije koje su definisane u *group_by_expression*.

Gde je *group_by_expression*:

{ (*grouping_column* [,...]) | ROLLUP (*grouping_column* [,...]) |

CUBE (*grouping_column* [,...]) | GROUPING SETS (*grouping_set_list*) |

( ) | *grouping_set* , *grouping_set_list*}

HAVING *search_condition*

Dodaje uslove pretraživanja na rezultate klauzule *GROUP BY* na način sličan klauzuli *WHERE*. *HAVING* nema uticaja na vrste koje se koriste za izračunavanje agregacija. Klauzula *HAVING* može da sadrži podupit.

ORDER BY *order_expression* [ASC | DESC]

Sortira rezultujući skup bilo u rastućem (ASC) bilo u opadajućem (DESC) uređenju korišćenjem specifikovanog *order_expression*. *Order_expression* je lista kolona, razdvojenih zarezima, po kojima se sortira.

Svaka klauzula u naredbi *SELECT* ima specifično korišćenje. Tako, moguće je pojedinačno govoriti o klauzuli *FROM*, klauzuli *WHERE*, klauzuli *GROUP BY* i tako dalje. Naravno, nisu u svakoj naredbi *SELECT* potrebne sve klauzule. Minimalno, za upit su potrebni *SELECT*, lista stavki i klauzula FROM. Zato što je klauzula *SELECT* tako značajna i nudi tako mnogo opcija, pravila korišćenja mogu da se razmatraju u nekoliko podsekcija:

- Usvojena imena i spajanja u klauzuli WHERE
- Klauzula *JOIN*
- Klauzula *WHERE*
- Klauzula *GROUP BY*
- Klauzula *HAVING*
- Klauzula *ORDER BY*

Usvojena imena i spajanja u kaluzuli WHERE

Može da bude potrebno da kolone pojedinačno imaju oznaku (prefiks) baze podataka, šeme, imena tabele, kada one imaju isto ime u jednoj ili više tabela u upitu.

Želimo da prikažemo sve nastavnike koji su angažovani u nastavi zajedno sa šiframa predmeta koje predaju, grupisano po imenima nastavnika i za svakog nastavnika po šifri predmeta:

```
SELECT n.Snast, Imen, Spred FROM Nastavnici AS n, Angazovanje AS a
WHERE n.Snast = a.Snast
ORDER BY Imen, Spred
```

Snast	Imen	Spred
13	Božić	23
13	Božić	25
7	Janković	12
7	Janković	15
6	Lazić	8
6	Lazić	10
8	Lukić	13
8	Lukić	14
8	Lukić	15
3	Marić	3
3	Marić	4
4	Marković	5
4	Marković	9
4	Marković	10
11	Milić	17
11	Milić	18
11	Milić	24
12	Mitrović	21
12	Mitrović	22
12	Mitrović	24
5	Nikolić	6
5	Nikolić	7
1	Petrović	1
1	Petrović	2
1	Petrović	19
10	Radović	26
10	Radović	27
9	Ristić	11
9	Ristić	16
9	Ristić	20
2	Tomić	2
2	Tomić	11

Ako sada želimo da prikažemo i nazive predmeta, dodaćemo još jedan uslov spajanja u WHERE klauzuli:

```
SELECT n.Snast, Imen, p.Spred, Nazivp FROM Nastavnici AS n, Angazovanje AS a,
Predmeti AS p
WHERE n.Snast = a.Snast AND a.Spred = p.Spred
ORDER BY Imen, Spred
```

Snast	Imen	Spred	Nazivp
13	Božić	23	Baze podataka
13	Božić	25	Projektovanje inf. sis.
7	Janković	12	Verovatnoća i statistika
7	Janković	15	Teorija funk. kom. pro.

6	Lazić	8	Diferencijalne jednačine
6	Lazić	10	Topologija
8	Lukić	13	Parcijalne i int. jed.
8	Lukić	14	Diferencijalna geometrija
8	Lukić	15	Teorija funk. kom. pro.
3	Marić	3	Analiza I
3	Marić	4	Analiza II
4	Marković	5	Analitička geometrija
4	Marković	9	Osnovi geometrije
4	Marković	10	Topologija
11	Milić	17	Računarstvo I
11	Milić	18	Računarstvo II
11	Milić	24	Računarstvo III
12	Mitrović	21	Veštačka int. i eks. sis.
12	Mitrović	22	Operativni sistemi
12	Mitrović	24	Računarstvo III
5	Nikolić	6	Diskretna matematika
5	Nikolić	7	Numerička analiza
1	Petrović	1	Logika
1	Petrović	2	Algebra
1	Petrović	19	Algebra i logika u rač.
10	Radović	26	Operaciona istraživanja
10	Radović	27	Finasijska matematika
9	Ristić	11	Funkcionalna analiza
9	Ristić	16	Metodika nastave
9	Ristić	20	Analiza III
2	Tomić	2	Algebra
2	Tomić	11	Funkcionalna analiza

Ova dva upita takođe prikazuju neka značajna pravila o spajanjima u klauzuli *WHERE*:

1. Korišćenje zareza za razdvajanje višestrukih elemenata u *select_item* listi, tabela u klauzuli FROM i stavki u *order_expression*.
2. Dorišćenje klauzule *AS* za definisanje usvojenih imena.
3. Dosledno korišćenje usvojenih imena kroz celu naredbu *SELECT* onako kako su prvi put definisana.

- U glavnom, trebalo bi da se favorizuje klauzula *JOIN* nad kaluzulom *WHERE* za opisivanje izraza spajanja. Funkcija klauzula *ON* i *WHERE* u različitim tipovima spajanja može da dovede do konfuzije. Sledeće činjenice mogu da pomognu u korišćenju ovih klauzula:
- *ON* je deo od inner, left, right i full joins. Cross join i union join nemaju klauzulu *ON* zato što nijedno od njih ne vrši filtriranje podataka.
- Klauzula *ON* u unutrašnjem spajanju je logički ekvivalentna klauzuli *WHERE*, isti uslov treba da se specificira u obe klauzule.
- Klauzula *ON* u spoljašnjim spajanjima(left, right, full) je različita od klauzule *WHERE*. Klauzula *WHERE* jednostavno filtrira vrste koje se vraćaju klauzulom *FROM*. Vrste koje se dobijaju filtriranjem nisu uključene u rezultat. Klauzula *ON* u spoljašnjem spajanju prvo filtrira vrste ukrštenog proizvoda, a onda uključuje odbijene vrste, proširene sa nula vrednostima.

JOIN klauzula

Za izvršavanje istog upita kao u prethodnom primeru korišćenjem spajanja prema ANSI načinu, nabraja se prvo tabela i ključna reč *JOIN*, koja prethodi tabeli koja će biti spojena.

Pošto se navede druga tabela, navodi se ključna reč *ON* i uslov spajanja koji se koristi u starom načinu upita.

Ovaj primer prikazuje upit kao u primeru *7.8*, ali sada u ANSI načinu:

```
SELECT n.Snast, Imen, p.Spred, Nazivp FROM Nastavnici AS n LEFT OUTER JOIN
Angazovanje AS a ON n.Snast = a.Snast
LEFT OUTER JOIN Predmeti AS p ON a.Spred = p.Spred
ORDER BY Imen, Spred
```

Alternativno može da se koristi kaluzula *USING*. Umesto opisivanja uslova spajanja, jednostavno daje jedno ili više *column_names* (razdvojenih zarezima), koji se pojavljuju u obe tabele koje se spajaju. *column_names* koje se pojavljuju u obe tabele. (imena kolona moraju da budu identična u obe tabele.)

U ovom primeru, dva upita proizvode iste rezultate, jedan korišćenjem kaluzule *ON* a drugi korišćenjem kaluzule *USING*:

```
SELECT n.Snast, Imen, p.Spred, Nazivp FROM Nastavnici AS n LEFT OUTER JOIN
Angazovanje AS a USING(n.Snast = a.Snast)
LEFT OUTER JOIN Predmeti AS p ON a.Spred = p.Spred
ORDER BY Imen, Spred
```

Može da se specificira nekoliko različitih tipova spajanja po ANSI standardu:

Cross join

> Specificira potpuni ukršteni proizvod dve tabele. Za svaku vrstu u prvoj tabeli, spojene su sve vrste u drugoj tabeli, kreirajući, moguće, ogroman rezultujući skup (naravno, rezultujući skup će biti mali ako obe tabele imaju samo po četiri vrste, ali zamislite ako one imaju po četiri miliona!). Ova naredba ima isti efekat kao da je izostavljen uslov spajanja i poznat je takođe kao "Cartesian Product." Ne preporučuje se korišćenje ukrštenih proizvoda.

Inner join

> Specificira da će ne uparene vrste u bilo kojoj od tabela biti odbačene. Ako se eksplicitno ne navede tip spajanja, u ANSI načinu se podrazumeva ovaj tip spajanja.

Left [outer] join

> Specificira da će biti vraćene sve vrste tabele na levoj strani naredbe spajanja. Ako vrste vraćene iz leve tabele nemaju uparene vrste u tabeli na desnoj strani spajanja, one se uvek vraćaju. Vrste iz desne tabele vraćaju NULL vrednosti. Mnogi profesionalci predlažu konfigurisanje spolajšnjih spajanja kao levih spajanja gde god je moguća konzistencija.

Right [outer] join

Specificira da će biti vraćene sve vrste iz tabele na desnoj strani naredbe spajanja, čak i ako tabela na levoj strani nema uparene vrste. Vrste iz leve tabele vraćaju NULL vrednosti.

Full [outer] join

Specificira da će biti vraćene sve vrste iz jedne ili iz druge tabele, nezavisno od uparenih vrsta u drugoj tabeli. Rezultujući skup prikazuje NULL vrednosti gde ne postoje upareni podaci u spajanju. (napomena: nisu sva spajanja podržana od svih platformi).

Klauzula WHERE

Loše napisana klauzula *WHERE* može da upropasti inače divnu naredbu *SELECT*, tako da se nijansama u korišćenju klauzule *WHERE* mora potpuno ovladati. (Podsećamo da se ova klauzula koristi i u naredbama *INSERT*, *UPDATE* i *DELETE*)

Treba prikazati šifu nastavnika, ime nastavnika, šifru predmeta i naziv predmeta za sve nastavnike koji na prvom mestu u imenu imaju slovo j ili slovo *t* na trećem mestu, a u nazivu predmeta slovo *k* ili *o* na bilo kom mestu u nazivu predmeta.Ovaj primer je tipičan upit i prikazuje više uslova:

```
SELECT n.Snast, Imen, p.Spred, Nazivp FROM Nastavnici AS n LEFT OUTER JOIN
Angazovanje AS a ON n.Snast = a.Snast
LEFT OUTER JOIN Predmeti AS p ON a.Spred = p.Spred
WHERE (Imen LIKE 'j%' OR Imen LIKE '__t%') AND (Nazivp LIKE '%k%'OR Nazivp
LIKE '%o%')
ORDER BY Imen, Spred
```

```
Snast  Imen                 Spred  Nazivp
------ ------------------------ ------ -------------------------
7      Janković             12     Verovatnoća i statistika
7      Janković             15     Teorija funk. kom. pro.
12     Mitrović             21     Veštačka int. i eks. sis.
12     Mitrović             22     Operativni sistemi
12     Mitrović             24     Računarstvo III
1      Petrović             1      Logika
1      Petrović             19     Algebra i logika u rač.
```

U izvršavanju ovog upita, značaj zagrada je u redosledu obrade uslova pretraživanja. Koristite zagrade da premestite uslov pretraživanja naviše ilii naniže po prioritetu sličnom algebarskim jednačinama. Postoji razlika između platformi kako kaluzula *WHERE* filtrira rezultate upita. Na primer, SQL Server, podrazumeva, uređenje po rečniku (dictionary-order) i neosetljivost na veličinu slova, tako da ne pravi razliku između "Smith," "smith," i "SMITH." Oracle, takođe, podrazumeva uređenje po rečniku, ali je osetljiv na veličinu slova.

Kao što prethodni primer pokazuje klauzula *WHERE* pruža mnogo mogućnosti. Slede samo neke najosnovnije mogućnosti klauzule *WHERE*.

NOT

Preinačuje operator poređenja korišćenjem sintakse *WHERE NOT* `expression`. Takoda se u upitu može da koristi *WHERE NOT LIKE*... or *WHERE NOT IN*...

Prikazati podatke o studentima koji nisu na smeru sa šifrom 1:

```
SELECT Indeks, Upisan, Imes, Mesto FROM Studenti
WHERE not Ssmer = 1
```

Indeks	Upisan	Imes	Mesto
1	2000	Ana	Kragujevac
2	2001	Sanja	Jagodina
2	2002	Voja	Lapovo
3	2001	Tanja	Ćuprija
3	2002	Toma	Kragujevac
4	2000	Pavle	Batočina
4	2001	Ranko	Kraljevo
5	2001	Marko	Kruševac
5	2002	Sima	Paraćin
5	2003	Mira	Kragujevac
6	2001	Ivan	Užice
6	2002	Nina	Čačak
7	2001	Irena	Kragujevac
8	2001	Kaća	Rača
8	2003	Jovan	Aranđelovac
9	2000	Aca	Kraljevo
9	2001	Milan	Smederevo
15	2002	Ana	Užice
17	2000	Maja	Jagodina
17	2003	Zoran	Kraljevo
18	2003	Saša	Jagodina

Operatori poređenja (*Comparison operators)*

Poredi bilo koji skup vrednosti korišćenjem operatora <, >, <>, >=, <= i =.

Prikazati podatke o studentima koji su upisani posle 2002 godine:

```
SELECT Indeks, Upisan, Imes FROM Studenti
 WHERE Upisan > 2002;
```

Indeks	Upisan	Imes
5	2003	Mira
8	2003	Jovan
16	2003	Tanja
17	2003	Zoran
18	2003	Saša

Traži se spisak studenata koji studiraju na smeru sa šifrom 4 i većom.:

```
SELECT indeks, Upisan, imes, Ssmer FROM studenti WHERE Ssmer >= 4
```

Indeks	Upisan	Imes	Ssmer
2	2002	Voja	4

```
3     2002   Toma                4
4     2001   Ranko               4
5     2003   Mira                4
6     2002   Nina                4
7     2001   Irena               4
8     2001   Kaća                4
8     2003   Jovan               4
9     2000   Aca                 4
17    2000   Maja                4
```

IS NULL or IS NOT NULL *conditions*

> Pretražuje bilo koje NULL ili NOT NULL vrednosti korišćenjem sintakse *WHERE expression IS [NOT] NULL*.

Napraviti spisak studenata za koje nije poznat smer:

```
SELECT Indeks, Upisan, Ime, Smer FROM Studenti WHERE Ssmer IS NULL
```

```
Indeks Upisan Imes                Ssmer
------ ------ ------------------------- ------
99     2000   Stevan              NULL
```

Napomena, obratiti pažnju da nepoznata vrednost nije isto što i blank kod karakter tipa podataka, odnosno 0 kod celobrojnog tipa podataka.

AND

> Spaja (integriše) višestruke uslove, vraćajući samo one vrste, koje udovoljavaju svim uslovima, korišćenjem operatora *AND*. Maksimalni broj uslova zavisi od platforme.

Dati pregled studenata upisanih 2002, koji studiraju na smeru sa šifrom 2:

```
SELECT * FROM Studenti WHERE Upisan = 2002 AND Ssmer = 3
```

```
Indeks Upisan Imes        Mesto               Datr                    Ssmer
------ ------ ------------ ------------------ ----------------------- ------
5      2002   Sima         Paraćin        1984-07-12 00:00:00.000 3
15     2002   Ana          Užice          1984-07-28 00:00:00.000 3
```

OR

> Spaja alternativne uslove, vraćajući vrste koje udovoljavaju bilo kom od uslova, korišćenem operatora *OR*.

Prikazati tražene podatke o studentima koji su upisani 2001 ili 2002 godine:

```
SELECT Indeks, Upisan, Imes, Ssmer FROM Studenti
 WHERE Upisan = 2001 OR Upisan = 2002;
```

```
Indeks Upisan Imes                Ssmer
------ ------ ------------------------- ------
1      2002   Nenad               1
```

```
2     2001  Sanja              3
2     2002  Voja               4
3     2001  Tanja              3
3     2002  Toma               4
4     2001  Ranko              4
5     2001  Marko              3
5     2002  Sima               3
6     2001  Ivan               3
6     2002  Nina               4
7     2001  Irena              4
8     2001  Kaća               4
9     2001  Milan              2
15    2002  Ana                3
```

Dati pregled specificiranih podataka o studentima koji su upisani 2002 godine ili su iz Jagodine:

```
SELECT Indeks, Upisan, Imes, Ssmer FROM Studenti WHERE Upisan = 2002 OR Mesto = 'Jagodina'
```

```
Indeks Upisan Imes                 Ssmer
------ ------ ------------------------ ------
1      2002   Nenad               1
2      2001   Sanja               3
2      2002   Voja                4
3      2002   Toma                4
5      2002   Sima                3
6      2002   Nina                4
15     2002   Ana                 3
17     2000   Maja                4
18     2003   Saša                3
```

LIKE

> Govori upitu da koristi šablon uparivanja nizova koji se nalazi u okviru zagrada. Simbol džokera (wildcar) je podržan kod svake platforme. Sve platforme podržavaju znak za procenat (%) za džoker.

Naći studente koji u imenu imaju *v* na drugom mestu ili *k* na bilo kom mestu u imenu:

```
SELECT Indeks, Upisan, Imes FROM Studenti
 WHERE Imes LIKE '_v%'  OR  Imes LIKE '%k%';
```

```
Indeks Upisan Imes
------ ------ ------------------------
4      2001   Ranko
5      2001   Marko
6      2001   Ivan
8      2001   Kaća
```

EXISTS

> Koristeći se samo sa podupitima, *EXISTS* ispituje da li postoje podaci u podupitu. To je obično mnogo brže nego u *WHERE IN* podupitu.

Prikazati broj indeksa, godinu upisa i ime studenata koji imaju ocenu 10 iz bilo kog predmeta:

```
SELECT DISTINCT Indeks, Upisan, Imes FROM Studenti
WHERE EXISTS (SELECT Indeks, Upisan FROM Prijave
                WHERE Ocena = 10);
```

```
Indeks Upisan Imes
------ ------ ------------------------
1      2000   Ana
1      2002   Nenad
2      2001   Sanja
2      2002   Voja
3      2000   Sava
3      2001   Tanja
3      2002   Toma
4      2000   Pavle
4      2001   Ranko
5      2001   Marko
5      2002   Sima
5      2003   Mira
6      2001   Ivan
6      2002   Nina
7      2001   Irena
8      2001   Kaća
8      2003   Jovan
9      2000   Aca
9      2001   Milan
12     2000   Saša
14     2000   Mira
15     2002   Ana
16     2003   Tanja
17     2000   Maja
17     2003   Zoran
18     2003   Saša
99     2000   Stevan
```

BETWEEN

> Izvršava proveru dometa (prostiranja) da vidi da li se vrednost nalazi između dve zadate vrednosti (uključujući i ove dve vrednosti).

Prikazati podatke o studentima, koji imaju ocene između 8 i 10 iz predmeta sa šifrom 5:

```
SELECT * FROM Prijave WHERE Ocena BETWEEN 8 AND 10 AND Spred = 5;
```

```
Spred  Indeks Upisan Snast  Datump                  Ocena
------ ------ ------ ------ ----------------------- ------
5      1      2000   4      2002-02-03 00:00:00.000 8
5      1      2002   4      2004-06-03 00:00:00.000 8
5      2      2001   4      2003-06-03 00:00:00.000 10
5      2      2002   4      2004-06-03 00:00:00.000 8
5      3      2001   4      2003-06-03 00:00:00.000 9
5      3      2002   4      2004-06-03 00:00:00.000 8
5      4      2000   4      2002-06-03 00:00:00.000 10
```

Isto se dobija i sledećom naredbom:

```
SELECT * FROM Prijave WHERE Ocena >= 8 AND Ocena <= 10 AND Spred = 5;
```

Na zahtev kao u prethodnom primeru dodati još i uslov da datum polaganja bude u zadatom vremenskom periodu:

```
SELECT * FROM Prijave WHERE Ocena >= 8 AND Ocena <= 10 AND Spred = 5 AND
Datump BETWEEN '01-JAN-2003' AND '31-DEC-2003';
```

```
Spred  Indeks Upisan Snast  Datump                  Ocena
------ ------ ------ ------ ----------------------- ------
5      2      2001   4      2003-06-03 00:00:00.000 10
5      3      2001   4      2003-06-03 00:00:00.000 9
```

IN

> Izvršava test da vidi da li izraz odgovara bilo kojoj jednoj vrednosti izvan navedenih vrednosti.

Prikazati podatke o studentima koji su upisani 2001 i 2002.(Isti zadatak je u primeru *7.17* urađen na drugi način).

```
SELECT Indeks, Upisan, Imes, Ssmer FROM Studenti  WHERE Upisan IN(2001, 2002);
```

```
Indeks Upisan Imes                    Ssmer
------ ------ ------------------------ ------
1      2002   Nenad                     1
2      2001   Sanja                    3
2      2002   Voja                     4
3      2001   Tanja                    3
3      2002   Toma                      4
4      2001   Ranko                     4
5      2001   Marko                     3
5      2002   Sima                     3
6      2001   Ivan                    3
6      2002   Nina                     4
7      2001   Irena                   4
8      2001   Kaća                      4
9      2001   Milan                     2
15     2002   Ana                       3
```

Navedena vrednost može da bude literal, kao u *WHERE* Imes IN ('Saša', 'Ana','Mira','Nada').

SOME | ANY

> Funkcioniše isto kao operacija *EXIST*, mada sa neznatnom razlikom u sintaksi.

Prikazati podatke o onim studentima koji su polagali bar jedan ispit:

```
SELECT * FROM Studenti
WHERE Studenti.Indeks = ANY (SELECT Indeks FROM Prijave
      WHERE Studenti.Indeks = Prijave.Indeks AND
                        Studenti.Upisan = Prijave.Upisan);
```

```
Indeks Upisan Imes      Mesto                        Datr                 Ssmer
------ ------ --------- ---------------------------- -------------------- ------
```

1	2000	Ana	Kragujevac	1982-05-21 00:00:00.000	2
1	2002	Nenad	Kragujevac	1984-06-15 00:00:00.000	1
2	2001	Sanja	Jagodina	1983-04-20 00:00:00.000	3
2	2002	Voja	Lapovo	1984-08-15 00:00:00.000	4
3	2000	Sava	Kragujevac	1982-08-03 00:00:00.000	1
3	2001	Tanja	Ćuprija	1983-01-08 00:00:00.000	3
3	2002	Toma	Kragujevac	1984-12-01 00:00:00.000	4
4	2000	Pavle	Batočina	1982-04-07 00:00:00.000	3
4	2001	Ranko	Kraljevo	1983-05-05 00:00:00.000	4
5	2003	Mira	Kragujevac	1984-01-23 00:00:00.000	4
6	2001	Ivan	Užice	1983-03-13 00:00:00.000	3
8	2003	Jovan	Aranđelovac	1984-03-21 00:00:00.000	4
99	2000	Stevan	Paraćin	NULL	NULL

ALL

Vrši proveru da vidi ako sve vrste u upitu uparuju kriterijum izračunavanja. Vraća TRUE kada podupit vraća nula vrsta.

Prikazati skup podataka o prijavama, koji zadovoljava uslove: ocena manja od bilo koje ocene, koju je dao nastavnik, koji ima u imenu bar jedno slovo *s*:

```
SELECT Indeks, Upisan, Ocena FROM Prijave WHERE Ocena < ALL (SELECT Ocena
FROM Prijave WHERE Snast = ANY(SELECT Snast FROM Nastavnici
        WHERE Imen LIKE'%s%'))
```

Indeks	Upisan	Ocena
1	2000	5
1	2000	6
1	2000	5
1	2000	5
1	2000	6
1	2000	6
1	2002	5
2	2002	5
2	2002	5
2	2002	6
2	2002	6
3	2000	5
5	2003	5

Ako sad promenimo *ALL* u *ANY* u prethodnom upitu dobija se:

```
SELECT Indeks, Upisan, Ocena FROM Prijave WHERE Ocena < ANY (SELECT Ocena
FROM Prijave WHERE Snast = ANY(SELECT Snast FROM Nastavnici
        WHERE Imen LIKE'%s%'))
```

Indeks	Upisan	Ocena
1	2000	9
1	2000	7
1	2002	8
1	2002	7
1	2000	7
1	2000	5
1	2000	6

1	2000	5
1	2000	8
1	2000	7
1	2000	8
1	2000	7
1	2000	9
1	2000	7
1	2000	5
1	2000	6
1	2000	8
1	2000	7
1	2000	6
1	2000	7
1	2002	9
1	2002	8
1	2002	7
1	2002	5
1	2002	8
1	2002	8
1	2002	7
1	2002	9
1	2002	7
1	2002	8
1	2002	7
1	2002	9
1	2002	8
2	2001	9
2	2001	9
2	2001	8
2	2001	9
2	2001	9
2	2001	9
2	2001	9
2	2001	9
2	2001	9
2	2001	9
2	2002	7
2	2002	7
2	2002	8
2	2002	5
2	2002	5
2	2002	7
2	2002	6
2	2002	8
2	2002	7
2	2002	8
2	2002	9
2	2002	7
2	2002	7
2	2002	8
2	2002	7
2	2002	6
3	2000	7
3	2000	8
3	2000	9
3	2000	7
3	2000	8
3	2000	7
3	2000	5
3	2000	7

3	2000	8
3	2000	7
3	2000	8
3	2000	9
3	2000	7
3	2000	8
3	2000	8
3	2000	8
3	2000	7
3	2000	7
3	2000	9
3	2001	8
3	2001	8
3	2001	9
3	2001	8
3	2001	8
3	2001	9
3	2001	9
3	2001	8
3	2001	8
3	2001	9
3	2001	9
3	2001	9
3	2001	8
3	2002	7
3	2002	8
3	2002	9
3	2002	8
3	2002	8
3	2002	8
3	2002	8
3	2002	9
3	2002	8
3	2002	9
3	2002	9
3	2002	8
3	2002	9
3	2002	9
4	2000	8
4	2000	9
4	2000	8
4	2000	9
4	2000	9
4	2000	9
4	2000	9
4	2000	9
4	2000	9
4	2000	9
4	2000	9
4	2001	7
4	2001	8
4	2001	7
4	2001	8
4	2001	7
4	2001	7
5	2003	7
5	2003	7
5	2003	5
5	2003	7
5	2003	8

```
8     2003   9
8     2003   8
8     2003   9
8     2003   9
5     2003   8
8     2003   9
8     2003   9
8     2003   9
6     2001   7
99    2000   8
```

Klauzula GROUP BY

Klauzula *GROUP BY* (i klauzula *HAVING*) je potrebna samo u upitima koji koriste agregatne funkcije. Ona deli tabelu u grupe, koje mogu da imaju svoje agregatne vrednosti. Klauzula *GROUP BY* se koristi za izveštavanje o agregatnim vrednostima za jednu ili više vrsta vraćenih naredbom *SELECT* baziranoj na jednoj ili više ne-agregiranih kolona koje se nazivaju kolone za grupisanje.

Prikazati ukupan broj studenata , minimalni broj indeksa i maksimalni broj indeksa za svaku godinu upisa:

```
SELECT COUNT (*), MIN(Indeks), MAX(Indeks) FROM Studenti
GROUP BY Upisan;
```

```
----------- ------ ------
8         1    99
8         2    9
6         1    15
5         5    18
```

Grupisanje je izvršeno prema ne-agregiranoj koloni Upisan. U zaglavlju nisu prikazani nazivi kolona, pošto se radi o izvedenim vrednostima, anisu data posebna imena.

Sledeća naredba je neispravna:

```
SELECT COUNT (*), MIN(Indeks), MAX(Indeks), Indeks, Upisan FROM Studenti GROUP
BY Upisan;
```

```
Msg 8120, Level 16, State 1, Line 1
Column 'Studenti.Indeks' is invalid in the select list because it is not contained in either an aggregate function or
the GROUP BY clause.
```

Razlog ne ispravnosti je, kao što kaže poruka, u navođenju kolone Indeks po kojoj se ne vrši grupisanje. Pošto su vrednosti kolone Indeks različite za svaku godinu upisa, koja vrednost bi se navela u rezultatu!

Ispravno je:

```
SELECT COUNT (*), MIN(Indeks), MAX(Indeks), Upisan FROM Studenti
GROUP BY Upisan;
```

```
                Upisan
```

---------- ------ ------ ------
8 1 99 2000
8 2 9 2001
6 1 15 2002
5 5 18 2003

Upiti koji koriste agregatne funkcije obezbeđuju mnogo tipova sumarnih informacija. Najčešće korišćene agregatne funkcije su:

AVG
: Vraća srednju vrednost svih ne-NULL vrednosti u specificiranoj koloni (kolonama).

AVG DISTINCT
: Vraća srednju vrednost svih jedinstvenih, ne-NULL vrednosti u specificiranoj koloni (kolonama).

COUNT
: Prebrojava pojavljivanje svih ne-NULL vrednosti u specificiranoj koloni (kolonama).

COUNT DISTINCT
: Prebrojava pojavljivanje svih jedinstvenih, ne-NULL vrednosti u specificiranoj koloni (kolonama).

COUNT(*)
: Prebrojava svaku vrstu u tabeli.

MAX
: Vraća najveću ne-NULL vrednost u u specificiranoj koloni (kolonama).

MIN
: Vraća najmanju ne-NULL vrednost u specificiranoj koloni (kolonama).

SUM
: Sabira sve ne-NULL vrednost u specificiranoj koloni (kolonama).

SUM DISTINCT
: Sabira sve jedinstvene, ne-NULL vrednosti u specificiranoj koloni (kolonama).

Neki upiti koji koriste agregate vraćaju same vrenosti. Agregati pojedinačne vrednosti su poznati kao skalarni agregati. Za skalarne agregate nije potrebna klauzula *GROUP BY*.

Prikazati prosečnu ocenu svih studenata iz svih predmeta:

```
SELECT AVG(Ocena) AS 'Ukupna srednja ocena'FROM Prijave
```

```
Ukupna srednja ocena
--------------------
8
```

Ovde je agregatna vrednost imenovana. Prosečna ocena je zaokružena.

Upiti koji vraćaju i regularne vrednosti kolona i vrednosti agregatnih funkcija se uobičajeno nazivaju vektor agregati. Vektor agregati koriste klauzulu *GROUP BY* i vraćaju jednu ili više vrsta. Postoji nekoliko pravila koje teba slediti kada se koristi *GROUP BY*:

- Postavite *GROUP BY* u pravilan redosled – posle klauzule *WHERE* a pre klauzule *ORDER BY*.
- Uključite sve ne-agregatne kolone u klauzulu *GROUP BY*.

- Ne koristite usvojeno ime kolone u kaluzuli *GROUP BY*, mada su usvojena imena dopustiva.

Pretpostavimo da je potrebno da znamo ukupni iznos porudžbina u tabeli koja se naziva Porudžbine i čija je instanca prikazan ispod:

BrojPor	ŠifraPr	Cena	Količina
10248	11	14.0000	12
10248	42	9.8000	10
10248	72	34.8000	5
10249	14	18.6000	9
10249	51	42.4000	40
10250	41	7.7000	10
10250	51	42.4000	35
10250	65	16.8000	15

...

```
SELECT BrojPor, SUM(Cena * Količina) AS 'Iznos porudžbine'

FROM Porudžbine

WHERE BrojPor IN (10248, 10249, 10250)

GROUP BY BrojPor
```

Rezultati je:

BrojPor	Iznos porudžbine
10248	440.0000
10249	1863.4000
10250	1813.0000

Mogli bi uz to da se usavrše (prerade) agregacije korišćenjem više od jedne kolone za grupisanje.

Prikazati broj studenata po godinama upisa i smeru na koji su upisani:

```
SELECT Upisan, Ssmer, COUNT (*) FROM Studenti
GROUP BY Upisan, Ssmer;
```

Upisan	Ssmer	
2000	NULL	1
2000	1	3
2002	1	1
2003	1	1
2000	2	1
2001	2	1
2003	2	1
2000	3	1
2001	3	4
2002	3	2
2003	3	1
2000	4	2

```
2001  4     3
2002  4     3
2003  4     2
```

Slično prethodnom primeru, prikazati broj upisanih studenata po smerovima i godini upisa:

```
SELECT Upisan, Ssmer, COUNT (*) FROM Studenti
GROUP BY Ssmer, Upisan;
```

```
Upisan Ssmer
------ ------ -----------
2000   NULL   1
2000   1      3
2000   2      1
2000   3      1
2000   4      2
2001   2      1
2001   3      4
2001   4      3
2002   1      1
2002   3      2
2002   4      3
2003   1      1
2003   2      1
2003   3      1
2003   4      2
```

Napomena: jedan student nije imao podatak o upisanom smeru.

Prikazati prosečne ocene studenata po godinama kada su polagali i prema godini upisa:

```
SELECT YEAR(datump) AS 'Godina', Upisan, SUM(Ocena)/COUNT(*) AS 'Srednja ocena
za godinu' FROM Prijave
GROUP BY YEAR(Datump),Upisan
ORDER BY YEAR(Datump),Upisan
```

```
Godina      Upisan Srednja ocena za godinu
----------- ------ ----------------------
2001        2000   7
2002        2000   7
2002        2001   8
2003        2000   8
2003        2001   8
2003        2002   7
2004        2000   6
2004        2001   9
2004        2002   7
2004        2003   7
2005        2000   8
2005        2001   9
2005        2002   8
2005        2003   8
2006        2000   8
2006        2001   7
2006        2002   9
2006        2003   9
```

U dodatku, kaluzula *GROUP BY* podržava jedan broj vrlo značajnih pod klasa:

GROUP BY [{ROLLUP | CUBE}] ([*grouping_column* [,...]]) [, *grouping set list*]

> Grupiše agregatne vrednosti rezultujućeg skupa po jednoj ili više kolona za grupisanje. (Bez *ROLLUP* ili *CUBE*, klauzula *GROUP BY (grouping_column [,...])* je najednostavnija i osnovna forma kaluzule *GROUP BY*.)
>
> *ROLLUP*
>
> Daje pod zbirove za svaki skup kolona koje se grupišu kao hijerarhijski rezultujući skup. Ona dodaje pod zbir i ukupni zbir vrsta u rezultujuće skupove u hijerarhijskom obliku. Operacije *ROLLUP* vraćaju jednu vrstu po grupišućoj koloni gde se NULL pojavljuje u grupišućoj koloni da prikaže pod zbirovne ili zbirove agregatne vrednosti.
>
> *CUBE*
>
> Daje pod zbirove i poprečno-tabelisane (cross-tabulated) zbirove za sve grupišuće kolone. U nekom smislu, klauzula *CUBE* omogućava da se brzo izvadi višedimenzijalni rezultujući skup iz standardnih relacionih tabela bez mnogo programerskog posla. *CUBE* je posebno korisna kada se radi sa velikom količinom poadataka. Slično *ROLLUP*, *CUBE* obezbeđuje pod zbirove kolona za grupisanje ali isto tako uključuje pod zbirove vrsta za sve moguće kombinacije kolona za grupisanje, koje su specificirane u upitu.

GROUP BY GROUPING SETS [{ROLLUP | CUBE}] ([*grouping_column* [,...]]) [, *grouping set list*]

> Omogućava agregirane grupe na različitim skupovima kolona za grupisanje u okviru istog upita. Ovo je posebno korisno kada se želi vraćanje samo delova agregiranog rezultujućeg skupa. Kaluzula *GROUPING SETS* takođe dozvoljava da se selektuju grupišuće kolone za poređenje, dok *CUBE* vraća sve grupišuće kolone, a *ROLLUP* vraća hijerarhijske podskupove kolona za grupisanje. Kao što sintaksa prikazuje, ANSI standard, takođe, omogućava da se *GROUPING SETS* sparuje (spaja) sa *ROLLUP* ili *CUBE*.

Prikazati broj studenata za godine upisa 2000 i 2002 po smerovima, u redosledu po godini upisa i smeru:

```
SELECT Upisan As 'Godina upisa', Ssmer AS 'Smer', COUNT (*) from Studenti
WHERE Upisan IN(2000, 2001)
GROUP BY Ssmer, Upisan WITH ROLLUP
ORDER BY Ssmer, Upisan
```

```
Godina upisa Smer
------------ ------ -----------
NULL         NULL   16
NULL         1       3
2000         1       3
NULL         2       3
2000         2       2
```

```
2001        2       1
NULL        3       5
2000        3       1
2001        3       4
NULL        4       5
2000        4       2
2001        4       3
```

Klauzula *GROUP BY CUBE* je korisna za izvođenje višedimenzionalnih analiza na agregiranim podacima. Slično *GROUP BY ROLLUP*, ona vraća podzbirove. Ali za razliku od *GROUP BY ROLLUP*, ona vraća podzbirove kombinivane od svih imenovanih kolona za grupisanje u upitu. (Kao što će se videti, ona ima, takođe, mogućnost da poveća broj vrsta koje su vraćene u rezultujućem skupu.).

Prikazati ukupan broj studenata po smerovima i godini upisa, za godine upisa 2000 i 2001:

```
SELECT Ssmer AS 'Smer', Upisan As 'Godina upisa',  COUNT (*) AS 'Ukupan broj' FROM
Studenti
WHERE Upisan IN(2000, 2001)
GROUP BY Ssmer, Upisan WITH CUBE
ORDER BY Ssmer, Upisan
```

```
Smer   Godina upisa
------ ------------ -----------
NULL   NULL         16
NULL   2000         8
NULL   2001         8
1      NULL         3
1      2000         3
2      NULL         3
2      2000         2
2      2001         1
3      NULL         5
3      2000         1
3      2001         4
4      NULL         5
4      2000         2
4      2001         3
```

Klauzula *GROUP BY GROUPING SETS* dopušta agregate na više od jedne grupe u jednom upitu. Za svaki skup grupe, upit vraća podzbirove sa grupisanim kolonama koje su označene sa NULL. Dok klauzulule *CUBE* i *ROLLUP GROUPING SETS* postavljaju podzbirove u rezultujući skup, klauzula *GROUPING SETS* omogućava da se kontroliše koje podzbirove dodati upitu. Klauzula *GROUPING SETS* ne vraća veliki zbir.

Drugi način da se razmišlja o *GROUPING SETS* je da se on razmotri slično *UNION ALL* sa više od jednog *GROUP BY* upita koji poziva različite delove istig podatka. Može da se kaže bazi podataka da doda pod zbirove u *GROUPING SET* jednostavnim dodavanjem u klauzuli *ROLLUP* ili *CUBE* u zavisnosti kako se želi da se izračunavanje pod zbirova pojavljuje.

Zadatak kao u jednom od prethodnih primera uz korišćenje *GROUP BY GROUPING SETS*.

```
SELECT Upisan As 'Godina upisa', Ssmer AS 'Smer', COUNT (*), GROUPING(Upisan)
AS'Po godini upisa', GROUPING(Ssmer) AS 'Po smeru'  FROM Studenti
WHERE Upisan IN(2000, 2001)
GROUP BY Ssmer, Upisan WITH ROLLUP
ORDER BY Ssmer, Upisan
```

Napomena: primer urađen u SQL serveru 2005.

GROUPING SETS može takođe da se konkatenira u sažeto stvorene velike kombinacije grupisanja. Konkatenirani *GROUPING SETS* daje ukršteni-proizvod grupisanja iz svih skupova u okviru navođenja *GROUPING SET*. Konkatenirani *GROUPING SET* su kompatibilni sa *CUBE* i *ROLLUP*. Konkatenirani *GROUPING SETS*, pošto izvršavaju ukršteni-proizvod svih *GROUPING SETS*, će da generišu vrlo veliki broj konačnih grupisanja iz čak malog broja konkateniranih grupisanja.

Klauzula HAVING

Klauzula *HAVING* dodaje uslov pretraživanja na rezultat klauzule GROUP BY. *HAVING* radi vrlo slično kaluzuli *WHERE*, ali primenjeno na klauzulu *GROUP BY*. Klauzula *HAVING* radi kao filter za ograničavanje grupa koje su formiranje klauzulom *GROUP BY*. Grupe koje ne zadovoljavaju uslov klauzule *HAVING* ne ulaze u rezultat. Ako se izostavi *GROUP BY*, *HAVING* podrazumeva samo jednu grupu.

Prikazati ukupan broj studenata po godinama upisa i smeru, pod uslovom da je upisano više od jednog studenta.

```
SELECT Upisan, Ssmer, COUNT (*) FROM Studenti
GROUP BY Upisan, Ssmer
HAVING COUNT(*) > 1;
```

```
Upisan Ssmer
------ ------ -----------
2000   1      3
2001   3      4
2002   3      2
2000   4      2
2001   4      3
2002   4      3
2003   4      2
```

Dati podatke kao i u prethodnom primeru, ali uz uslov da je u mestu studenta zastupljeno bar jedno slovo *a*, bilo malo ili veliko:

```
SELECT Upisan, Ssmer, COUNT (*) from Studenti
WHERE Mesto LIKE '%a%'
GROUP BY Upisan, Ssmer
HAVING COUNT(*) > 1;
```

```
Upisan Ssmer
------ ------ -----------
2000   1      3
2001   3      3
```

```
2000  4    2
2001  4    3
2002  4    3
2003  4    2
```

Zapazimo da ANSI standard ne zahteva da se eksplicitno klauzula *GROUP BY* pojavi sa kaluzulom *HAVING*. (Ovo važi i za većinu implementacija)

Prikazati ukupan broj studenata, koji su iz mesta u čijem nazivu postoji bar jedno slovo *a*, bilo veliko ili malo ako je taj broj veći od 10:

```
SELECT COUNT(*) AS 'Ukupno studenata' from studenti
WHERE Mesto LIKE '%a%'
HAVING COUNT(*) > 10
```

```
Ukupno studenata
----------------
24
```

Mada je prethodni primer valjan, njegova primena za klauzulu *HAVING* je prilično retka.

Moguće su pod upiti u klauzuli *HAVING* (kao i u klauzuli *WHERE*).

Primer je sličan prethodnom primeru, ali je u klauzuli *HAVING* dodat podupit – da je ukupan broj studenata koji zadovoljavaju dati uslov veći od prosečne ocene svih studenata:

```
SELECT COUNT(*) AS 'Ukupno studenata' FROM Studenti
WHERE Mesto LIKE '%a%'
HAVING COUNT(*) > (SELECT SUM(Ocena)/COUNT(*) FROM Prijave)
```

```
Ukupno studenata
----------------
24
```

Upit sa pod upitom u klauzuli *HAVING* radi na sledeći način:

- Prvo se izvršava spoljašnji upit (*SELECT, FROM I GROUP BY*)
- Klauzula HAVIG filtrira dobijene grupe
- Izvršava se unutrašnji upit. Ako ona radi sa istim tabelama kao i spoljašnji koriste se alternativna imena.

Klauzula ORDER BY

Rezultujući skup može da se sortira kroz kaluzulu *ORDER BY*, u saglasnosti sa redosledom (uređenjem) sortiranja baze podataka. Svaka kolona rezultujućeg skupa može da se sortira bilo u rastućem (ASC) ili opadajućem (DESC) redosledu. (Rastuće uređenje je podrazumevano.) Ako se klauzula *ORDER BY* specificira, većina implementacija vraća podatke zavisno od fizičkog uređenja podataka u okviru tabele, ili u zavisnosti od uređenja indeksa koji se koristi u upitu. Naravno, kada kluzula *ORDER BY* nije specificirana, ne garantuje se uređenje rezultujućeg skupa.

Ako se vrši sortiranje po više kolona, redosled sortiranja će biti s leva na desno, odnossno najviši nivo sortiranja ima prva kolona posmatran s leva na desno

Može da se napiše klauzula *ORDER BY* korišćenjem kolona u tabeli koje se ne pojavljuju u *SELECT* listi.

Kada se jednom usvojeno ime dodeli tabeli ili pogledu u kaluzuli *FROM*, treba ga ekskluzivno koristiti za sva druga pozivanja ove tabele ili pogleda u okviru upita (u klauzuli *WHERE*, na primer). Ne treba mešati puno ime tabele i usvojeno ime u okviru jednog upita. Trebalo bi da se izbegne mešanje pozivanja iz nekoliko razloga. Prvo, jednostavno je nekonzistentno i čini održavanje koda teškim. Drugo, neke platforme baza podataka vraćaju greške na naredbi *SELECT*, koje sadrže mešovita pozivanja (reference). (Pogledajte sekciju za pod naredbu *SUBQUERY* za posebne instrukcije i stepenast efekat u okviru pod upita.).

MySQL, PostgreSQL i SQL Server podržavaju određene tipove upita kojima nije potrebna kaluzula *FROM*. Treba koristiti ove tipove upita sa opreznošću, pošto ANSI standard zahteva kaluzulu *FROM*. Upiti bez *FROM* klauzule moraju ručno da migriraju (presele se) bilo u oblik ANSI standarda ili u obliku koji, takođe, radi na ciljnoj bazi podataka. Određene platforme ne podržavaju ANSI stil klauzule *JOIN*.

Prikazati broj indeksa, godinu upisa, šifru predmeta i ocenu za studente koji nisu upisani 2000 i 2001 godine sa ocenom većom od 7 po rastućoj šifri predmeta i najvećoj oceni:

```
SELECT Indeks, Upisan, Spred, Ocena FROM Prijave
WHERE Upisan != 2000 AND Upisan != 2001 AND Ocena > 7
ORDER BY Spred ASC, Ocena DESC
```

```
Indeks Upisan Spred  Ocena
------ ------ ------ ------
8      2003   1      9
1      2002   1      8
3      2002   2      8
8      2003   2      8
8      2003   3      9
3      2002   3      8
1      2002   3      8
8      2003   4      9
3      2002   4      8
5      2003   4      8
3      2002   5      8
1      2002   5      8
2      2002   5      8
1      2002   6      8
3      2002   6      8
5      2003   6      8
8      2003   7      9
3      2002   7      9
2      2002   7      8
8      2003   8      10
2      2002   8      9
1      2002   8      9
3      2002   8      8
3      2002   9      9
8      2003   9      9
8      2003   10     10
```

3	2002	10	9
1	2002	10	8
8	2003	11	9
3	2002	11	8
2	2002	11	8
1	2002	12	9
3	2002	12	9
3	2002	13	10
1	2002	13	8
3	2002	15	10
3	2002	17	9
1	2002	17	9
2	2002	17	8
3	2002	19	10
8	2003	19	10
3	2002	21	10
3	2002	23	9

Prikazati indeks, upisan, ime i mesto studenata koji su na smerovima *3* i *4* uređeno po mestu i smeru (primer uređenja po koloni koja nije u *SELECT* listi):

```
SELECT Indeks, Upisan, Imes, Mesto FROM Studenti
WHERE Ssmer IN(3,4)
ORDER BY Mesto, Ssmer
```

Indeks	Upisan	Imes	Mesto
8	2003	Jovan	Aranđelovac
4	2000	Pavle	Batočina
6	2002	Nina	Čačak
3	2001	Tanja	Ćuprija
2	2001	Sanja	Jagodina
18	2003	Saša	Jagodina
17	2000	Maja	Jagodina
7	2001	Irena	Kragujevac
5	2003	Mira	Kragujevac
3	2002	Toma	Kragujevac
4	2001	Ranko	Kraljevo
9	2000	Aca	Kraljevo
5	2001	Marko	Kruševac
2	2002	Voja	Lapovo
5	2002	Sima	Paraćin
8	2001	Kaća	Rača
6	2001	Ivan	Užice
15	2002	Ana	Užice