Uvod u relacione baze podataka — Tema 1: Uvod u SQL

Univerzitet u Beogradu · Matematički fakultet · Školska 2026/2027 ·

dr Nina Radojičić Matić

Uvodna napomena

Ovaj dokument sadrži primere za časove koji se odnose na temu: Uvod u upitni jezik SQL. Primeri su inicijalno preuzeti od profesora Saše Malkova, navedeni bez većih izmena, uz nešto detaljnija objašnjenja. Svi primeri se izvršavaju nad bazom podataka Stud2020 u sistemu IBM Db2.

Gde se primeri izvršavaju

Naredbe iz primera mogu se izvršavati na više načina:

- **u terminalu**, u procesoru komandne linije sistema Db2 (CLP): komandom `db2 -t` pokreće se interaktivni rad, u kome se naredbe kucaju jedna za drugom i završavaju znakom `;`;
- **iz datoteke sa ekstenzijom `.sql`**: naredbe se zapišu u datoteku (svaka završena znakom `;`), a zatim se cela datoteka izvrši komandom `db2 -tvf primeri.sql`;
- **u modernijim razvojnim okruženjima**, kao što su DataGrip ili IBM Data Studio, u kojima se uspostavi veza sa bazom podataka, a naredbe se kucaju i izvršavaju u SQL editoru.

Zbog toga su sve naredbe u primerima završene znakom `;`. Naredbe `LIST TABLES`, `DESCRIBE` i `REORG` nisu SQL naredbe, već komande procesora komandne linije sistema Db2: izvršavaju se u terminalu, a u razvojnim okruženjima se umesto njih koriste alternative navedene uz primere. Naredba `CONNECT` kuca se u terminalu kao i ostale, dok se u razvojnim okruženjima veza sa bazom podataka podešava pri pravljenju konekcije.

Kako su primeri prikazani

Svaki primer je prikazan u tri dela: **naredba**, **rezultat** njenog izvršavanja i **objašnjenje**. Prikazani rezultati dobijeni su izvršavanjem nad bazom Stud2020; način prikaza (npr. broj decimala realnih brojeva, zapis logičkih vrednosti i datuma) zavisi od alata u kome se naredba izvršava. Nedefinisane vrednosti označene su sa NULL. Kod veoma širokih ili dugačkih rezultata prikazan je samo deo kolona ili redova, što je navedeno ispod rezultata. Neki rezultati zavise od trenutka izvršavanja (npr. tekući datum), pa se pri ponovnom izvršavanju razlikuju od prikazanih.

Čas 1 — Uvod u SQL

1. Povezivanje sa bazom podataka

Primer 1.1 — Povezivanje sa bazom podataka

```
connect to stud2020 user student using abcdef;
set schema da;
```

Rezultat: veza sa bazom podataka je uspostavljena, a tekuća shema postavljena.

Objašnjenje: Pre izvršavanja bilo kakvih upita potrebno je uspostaviti vezu sa bazom podataka. Naredbom `CONNECT TO` navodi se ime baze podataka (`stud2020`), korisničko ime i lozinka; na računarima na Fakultetu podrazumevani studentski nalog je korisnik `student` sa lozinkom `abcdef` . Naredbom `SET SCHEMA` određuje se podrazumevana shema (videti odeljak 2). U razvojnim okruženjima (DataGrip, Data Studio) veza se obično podešava kroz dijalog za novu konekciju, pa se naredba `CONNECT` ne kuca.

2. Tabele i sheme

Osnovni pojam u relacionim bazama podataka jeste **relacija**, koja se na nivou implementacije i upotrebe naziva **tabela**. Formalno definisanje relacija i odgovarajuća terminologija biće predmet kasnijeg dela kursa; u ovom delu koristi se intuitivan pojam tabele.

Tabela se sastoji od **kolona** i **redova**. Svaka kolona ima *ime*, jedinstveno u okviru tabele, i *tip*, koji određuje skup dopuštenih vrednosti u toj koloni. Svaki red sadrži po jednu vrednost za svaku kolonu tabele.

Svaka trajna tabela u bazi podataka ima svoje ime. Radi logičke organizacije, tabele (i drugi objekti baze podataka) grupišu se u **sheme**, a puno (kvalifikovano) ime tabele zapisuje se u obliku:

```
<ime sheme>.<ime tabele>
```

Primer 1.2 — Postavljanje tekuće sheme

```
set schema da;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Ako se tabela navede bez imena sheme, podrazumeva se *tekuća shema*. Na početku rada tekuća shema ima ime jednako korisničkom imenu prijavljenog korisnika. Tekuća shema se menja naredbom sistema Db2 `SET SCHEMA` ; posle naredbe iz ovog primera, ime `dosije` označava tabelu `DA.DOSIJE` .

Napomena: Imena tabela, kolona i shema u SQL-u ne zavise od veličine slova, osim ako su zapisana pod navodnicima. Sistem ih interno čuva velikim slovima, zbog čega se u rezultatima prikazuju kao `DA`, `DOSIJE`, `SKGODINA` i sl.

3. Komentari

Primer 1.3 — Linijski komentar

```
set schema da; --ovo je komentar
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Komentari u SQL-u zapisuju se kao **linijski komentari**: komentar počinje simbolom `--` i proteže se do kraja reda. Sadržaj komentara se zanemaruje pri izvršavanju. Ista sintaksa važi i za komande procesora komandne linije sistema Db2. Sistem Db2 podržava i komentare oblika /* ... */, koji se mogu prostirati kroz više redova.

Komentari se u nastavku koriste za kratak opis namene upita, što je dobra praksa i pri samostalnom radu.

4. Upiti i upitne naredbe

Naredbe za čitanje (često se kaže i *izdvajanje*) podataka iz postojećeg sadržaja baze podataka nazivaju se **upitnim naredbama**. One čitaju podatke iz postojećih tabela baze podataka i daju rezultat u obliku privremene tabele. Može se reći da upitne naredbe **izračunavaju nove tabele na osnovu postojećih**. Upitne naredbe ne menjaju sadržaj baze podataka.

Potrebno je razlikovati **upit** i **upitnu naredbu**:

- **Upitna naredba** je naredba sistema za upravljanje bazama podataka (SUBP), tj. jedinica koja se šalje sistemu na izvršavanje.
- **Upit** je izraz koji čini glavni deo upitne naredbe i koji izračunava rezultat.
- **Podupit** je nešto manje uopšten upitni izraz, koji se može koristiti kao deo većeg upita ili čak većeg podupita.

Danas je razlika između upita i podupita relativno mala:

- u podupitima često ne mogu da se koriste skupovne operacije;
- neke stvari, poput uređivanja rezultata, nema smisla navoditi u podupitima;
- većina upita, ako se navede u okviru zagrada, može se iskoristiti kao podupit.

Prema složenosti rezultata razlikuju se tri vrste upita:

Vrsta	Rezultat
Tabelarni upit	tabela (proizvoljan broj kolona i redova) — najopštija vrsta upita
Vektorski upit	jedna kolona (proizvoljan broj redova)
Skalarni upit	jedna vrednost (jedna kolona i najviše jedan red)

Osnovna upitna naredba je naredba `SELECT`. Pre nje upoznaćemo naredbu `VALUES`, koja služi za zapisivanje **konstantnih tabela** ili **literala**. Kasnije će biti predstavljena i naredba `WITH` za zapisivanje složenijih upita, kao i drugi složeniji oblici upita.

5. Naredba VALUES

Naredba `VALUES` služi za navođenje **konstantnih tabela**, tj. tabela čiji je sadržaj u potpunosti zadat u samoj naredbi. Opšti oblik je:

```
VALUES <red_1>, <red_2>, ..., <red_n>
```

Svaki red se navodi kao lista vrednosti (kolona) unutar zagrada:

```
( <kol_1>, <kol_2>, ..., <kol_m> )
```

Pri tome važe sledeća pravila:

- svi redovi moraju imati isti broj kolona *m*;
- odgovarajuće kolone u svim redovima moraju biti **saglasnog tipa**, tj. vrednosti na istoj poziciji moraju pripadati tipovima koji se mogu svesti na zajednički tip;
- ako red ima samo jednu kolonu, može se navesti bez zagrada.

Primer 1.4 — Konstantna tabela sa dva reda i tri kolone

```
values
    (1, 'a', 15.7),
    (2, 'b', 20);
```

Rezultat:

1	2	3
1	a	15.7
2	b	20.0

Objašnjenje: Kolone rezultata naredbe `VALUES` nemaju imena, pa ih sistem Db2 označava rednim brojevima `1`, `2`, … U ovom primeru druga kolona je tipa niske, a treća kolona sadrži vrednosti `15.7` (decimalni broj) i `20` (ceo broj). Tip treće kolone rezultata je zajednički tip ovih vrednosti — decimalni broj — pa se vrednost `20` prikazuje kao `20.0`.

Primer 1.5 — Kolone nesaglasnog tipa

```
values
    (1, 'a', 15.7),
    (2, 'b', 'abc');
```

Rezultat:

```
SQL0420N  Invalid character found in a character string argument of the
function "DECFLOAT".  SQLSTATE=22018  SQLCODE=-420
```

Objašnjenje: Ako odgovarajuće kolone nisu saglasnog tipa, dolazi do greške. U ovom primeru treća kolona prvog reda je broj, a drugog reda niska `'abc'`. Sistem pokušava da nisku implicitno konvertuje u numerički tip (`DECFLOAT`); pošto `'abc'` nije zapis broja, konverzija ne uspeva i naredba se prekida sa greškom `SQL0420N`.

Primer 1.6 — Konstantna tabela sa jednom kolonom

```
values 1, 2, 3, 4, 5;
```

Rezultat:

1
1
2
3
4
5

Objašnjenje: Ako redovi imaju samo jednu kolonu, zagrade oko pojedinačnih redova mogu se izostaviti. Ovaj primer daje tabelu sa jednom kolonom i pet redova (vektor).

6. Naredba SELECT

Puna sintaksa naredbe `SELECT` je relativno složena, pa se upoznaje postepeno; pun oblik biće rezimiran kada budu predstavljeni svi njeni elementi. Osnovni oblik naredbe čine tri komponente (klauzule):

```
SELECT <lista imena kolona>
FROM   <tabela>
[WHERE <uslov>]
```

Uglaste zagrade označavaju opcioni deo. Značenje naredbe je sledeće: čita se tabela navedena u klauzuli `FROM`, odabiraju se samo redovi koji zadovoljavaju uslov iz klauzule `WHERE`, a iz njih se izdvajaju samo kolone navedene u klauzuli `SELECT`. Važno je uočiti da se klauzule *izračunavaju* redom `FROM` → `WHERE` → `SELECT`, iako se zapisuju drugim redom.

6.1. Projekcija

Primer 1.7 — Projekcija na dve kolone

```
select SkGodina, Datpocetka
from SkolskaGodina;
```

Rezultat:

SKGODINA	DATPOCETKA
2015	2015-10-01
2016	2016-10-01
2017	2017-10-01
2018	2018-10-01
2019	2019-10-01
2020	2020-10-01

Objašnjenje: Izdvajanje samo nekih kolona tabele naziva se **projekcija**. Kolone se navode u klauzuli `SELECT`, razdvojene zarezima; redosled kolona u rezultatu odgovara redosledu navođenja. Ovaj upit iz tabele `SkolskaGodina` izdvaja samo kolone `SkGodina` i `DatPocetka`.

Primer 1.8 — Izdvajanje svih kolona

```
select *
from SkolskaGodina;
```

Rezultat:

SKGODINA	DATPOCETKA	DATKRAJA
2015	2015-10-01	2016-09-30
2016	2016-10-01	2017-09-30
2017	2017-10-01	2018-09-30
2018	2018-10-01	2019-09-30
2019	2019-10-01	2020-09-30
2020	2020-10-01	2021-09-30

Objašnjenje: Ako se umesto liste imena kolona navede simbol `*`, izdvajaju se sve kolone tabele, i to redosledom kojim su definisane u tabeli.

Primer 1.9 — Sve kolone navedene tabele

```
select SkolskaGodina.*
from SkolskaGodina;
```

Rezultat:

SKGODINA	DATPOCETKA	DATKRAJA
2015	2015-10-01	2016-09-30
2016	2016-10-01	2017-09-30
2017	2017-10-01	2018-09-30
2018	2018-10-01	2019-09-30
2019	2019-10-01	2020-09-30
2020	2020-10-01	2021-09-30

Objašnjenje: Kasnije će biti pokazano da tabela iz koje se čita može da se izračunava iz više drugih tabela. Ako je potrebno izdvojiti sve kolone samo jedne od njih, umesto `*` navodi se `<ime tabele>.*`. U ovom primeru to ne menja rezultat, jer se čita samo jedna tabela, ali je navedeno radi ilustracije sintakse.

6.2. Restrikcija

Klauzula `WHERE` je opciona. Ako se ne navede, izdvajaju se svi redovi tabele. Ako se navede, izdvajaju se samo oni redovi za koje je uslov **tačan**. Takvo odabiranje redova naziva se **restrikcija** (ili selekcija). Uslov je logički izraz koji se izračunava posebno za svaki red tabele; u njemu se mogu koristiti operatori poređenja `=`, `<>`, `<`, `<=`, `>`, `>=` i logički operatori `AND`, `OR` i `NOT`.

Primer 1.10 — Restrikcija uslovom jednakosti

```
select *
from SkolskaGodina
where SkGodina = 2018;
```

Rezultat:

SKGODINA	DATPOCETKA	DATKRAJA
2018	2018-10-01	2019-09-30

Primer 1.11 — Restrikcija uslovom poređenja

```
select *
from SkolskaGodina
where SkGodina > 2018;
```

Rezultat:

SKGODINA	DATPOCETKA	DATKRAJA
2019	2019-10-01	2020-09-30
2020	2020-10-01	2021-09-30

Projekcija i restrikcija se mogu kombinovati u istom upitu: najpre se restrikcijom odaberu redovi, a zatim se iz njih projekcijom izdvoje tražene kolone.

Primer 1.12 — Restrikcija i projekcija

```
select SkGodina, DatPocetka
from SkolskaGodina
where SkGodina = 2018;
```

Rezultat:

SKGODINA	DATPOCETKA
2018	2018-10-01

Važno: Izbegavajte postavljanje upita bez restrikcije. Takvi upiti izdvajaju sve redove tabele, što kod velikih tabela može da ima za posledicu nepotrebno zauzeće resursa sistema (vremena izvršavanja, memorije i mrežnog prenosa).

Primer 1.13 — Studenti rođeni u Prijepolju

```
-- Izdvajanje imena i prezimena svih studenata rođenih u Prijepolju
select ime, prezime
from dosije
where mestorodjenja = 'Prijepolje';
```

Rezultat:

IME	PREZIME
Natalija	Velickovic
Anes	Mandic
Mateja	Gluscevic
Mirna	Baranin
Milica	Pavela
...	...
Stefan	Todorovic
Ivan	Kasapu
Milos	Vasiljevic
Milan	Panic
Kristijan	Prodan

Ukupno: 42 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: Ovaj primer izdvaja imena i prezimena svih studenata rođenih u Prijepolju. Niske se u SQL-u zapisuju između jednostrukih navodnika, a poređenje niski u ovoj bazi podataka razlikuje velika i mala slova (`'Prijepolje'` i `'prijepolje'` nisu jednake niske).

6.3. Zadatak: nazivi predmeta koji počinju sa „Prog”

Zadatak: izdvojiti sve različite nazive predmeta koji počinju sa „Prog”.

Primer 1.14 — Nazivi koji počinju sa „Prog”

```
-- Izdvajanje svih naziva predmeta koji počinju sa "Prog"
select naziv
from predmet
where naziv >= 'Prog'
  and naziv < 'Proh';
```

Rezultat:

NAZIV
Programiranje 1
Programiranje 2
Programiranje 1
Programiranje 2
Programski paketi u matematici
Programske paradigme
Programiranje baza podataka
Programske paradigme
Programiranje baza podataka
Programiranje ogranicenja
Programiranje za veb
Programiranje za veb

Napomena: prikazano je prvih 12 od 16 redova.

Objašnjenje: Niske se porede **leksikografski**, znak po znak, prema redosledu znakova u kodnoj tabeli (tačnije, prema pravilu poređenja baze podataka). Niska počinje prefiksom `Prog` ako i samo ako je leksikografski veća ili jednaka niski `'Prog'` i strogo manja od niske `'Proh'`, koja se dobija kada se poslednji znak prefiksa zameni narednim znakom (`g` → `h`). Zato se uslov „počinje sa Prog” može zapisati kao konjunkcija dva poređenja.

Rezultat sadrži ponovljene nazive (npr. „Programiranje 1” i „Programiranje 2” pojavljuju se po dva puta), jer u tabeli `predmet` postoji više različitih predmeta sa istim nazivom. Ovaj upit, dakle, još ne rešava zadatak u potpunosti: da bi se izdvojili samo *različiti* nazivi, potrebno je isključiti ponavljanja ključnom rečju `DISTINCT` (odeljak 9). Isti uslov se može zapisati i jednostavnije predikatom `LIKE`, koji će biti predstavljen kasnije.

7. Baza podataka Stud2020 i podaci o bazi podataka

Na predavanjima i vežbama za sve primere koristi se baza podataka **Stud2020**. To je pojednostavljen model baze podataka informacionog sistema fakulteta, sa podacima o studentima (tabela `DOSIJE`), studijskim programima, predmetima (`PREDMET`), školskim godinama (`SKOLSKAGODINA`), semestrima (`SEMESTAR`), ispitnim rokovima, ispitima (`ISPIT`) i dr. Detaljan opis baze podataka dostupan je na stranici kursa.

7.1. Naredbe LIST TABLES i DESCRIBE

Primer 1.15 — Spisak tabela

```
list tables;
```

Rezultat:

TABNAME	TABSCHEMA	DESCRIPTION
ADVISE_INDEX	STUDENT	Table
ADVISE_INDEX	SYSTOOLS	Table
ADVISE_INSTANCE	STUDENT	Table
ADVISE_INSTANCE	SYSTOOLS	Table
ADVISE_MQT	STUDENT	Table
ADVISE_MQT	SYSTOOLS	Table
ADVISE_PARTITION	STUDENT	Table
ADVISE_PARTITION	SYSTOOLS	Table
ADVISE_TABLE	STUDENT	Table
ADVISE_TABLE	SYSTOOLS	Table
ADVISE_WORKLOAD	STUDENT	Table
ADVISE_WORKLOAD	SYSTOOLS	Table

Napomena: prikazano je prvih 12 od 66 redova.

Objašnjenje: Sistem Db2 ima više naredbi koje pomažu da se bolje upozna baza podataka sa kojom se radi. Ove naredbe nisu deo standarda SQL, već su komande procesora komandne linije sistema Db2: izvršavaju se u terminalu (`db2 -t`), dok se u razvojnim okruženjima isti podaci dobijaju upitima nad sistemskim katalogom (odeljak 7.2) ili pozivom `CALL SYSPROC.ADMIN_CMD('DESCRIBE TABLE da.dosije');` . Naredba `LIST TABLES` izdvaja spisak tabela baze podataka; za svaku tabelu navodi se ime (`TABNAME`), shema (`TABSCHEMA`) i vrsta objekta (`DESCRIPTION` , npr. `Table` ili `View`).

Rezultati u ovom odeljku prikazani su u obliku koji daju upiti nad sistemskim katalogom, odnosno poziv procedure `SYSPROC.ADMIN_CMD` . U terminalu se isti podaci ispisuju u drugačijem obliku: npr. `LIST TABLES` ispisuje kolone `Table/View` , `Schema` , `Type` i `Creation time` , i to bez dodatnih opcija samo za tabele tekućeg korisnika; tabele drugih shema dobijaju se sa `LIST TABLES FOR SCHEMA <shema>` ili `LIST TABLES FOR ALL` .

Naredba `DESCRIBE` opisuje kolone izabrane tabele. Opis obuhvata ime kolone (`COLNAME`), shemu i ime tipa (`TYPESCHEMA` , `TYPENAME`), dužinu (`LENGTH`), broj decimala (`SCALE`) i podatak o tome da li kolona može da sadrži nedefinisane vrednosti (`NULLABLE` : `Y` — može, `N` — ne može). Za tip `VARCHAR` dužina je najveći dozvoljeni broj znakova, a za tipove `INTEGER` , `SMALLINT` i `DATE` broj bajtova interne reprezentacije.

Primer 1.16 — Opis kolona tabele DOSIJE

```
describe table dosije;
```

Rezultat:

COLNAME	TYPESCHEMA	TYPENAME	LENGTH	SCALE	NULLABLE
INDEKS	SYSIBM	INTEGER	4	0	N
IDPROGRAMA	SYSIBM	INTEGER	4	0	N
IME	SYSIBM	VARCHAR	50	0	N
PREZIME	SYSIBM	VARCHAR	50	0	N
POL	SYSIBM	CHARACTER	1	0	Y
MESTORODJENJA	SYSIBM	VARCHAR	50	0	Y
IDSTATUSA	SYSIBM	SMALLINT	2	0	N
DATUPISA	SYSIBM	DATE	4	0	N
DATDIPLOMIRANJA	SYSIBM	DATE	4	0	Y

Naredbom `DESCRIBE` može se opisati i rezultat upita, bez njegovog izvršavanja. Tada se za svaku kolonu rezultata navodi, između ostalog, ime (`SQLNAME_DATA`), tip (`SQLTYPE`) i dužina (`SQLLENGTH`). Kolona `SQLTYPE_ID` sadrži interni kôd tipa (npr. 496 za `INTEGER`, 448 za `VARCHAR`); paran kôd označava da kolona ne može da sadrži nedefinisane vrednosti, a neparan da može.

Primer 1.17 — Opis kolona rezultata upita

```
describe select indeks, ime, prezime from dosije;
```

Rezultat:

SQLTYPE_ID	SQLTYPE	SQLLENGTH	SQLSCALE	SQLNAME_DATA	SQLNAME_LENGTH	SQLDATATYPE_NAME_DATA	SQLDATATYPE_NAME_LENGTH
496	INTEGER	4	0	INDEKS	6		0
448	VARCHAR	50	0	IME	3		0
448	VARCHAR	50	0	PREZIME	7		0

7.2. Sistemski katalog

Po standardu SQL-a svi podaci o bazi podataka (tzv. *metapodaci*) moraju da budu sačuvani upravo u toj istoj bazi podataka. Štaviše, sve podatke o bazi podataka trebalo bi da je moguće pročitati (ali ne i neposredno izmeniti) istim upitnim naredbama kojima se čita i sadržaj korisničkih podataka. To se ostvaruje tako što se u samoj bazi podataka održavaju tzv. **sistemske tabele**, u kojima se nalaze podaci o organizaciji, ali i o implementaciji baze podataka. Sistem ih ažurira automatski, pri svakoj izmeni strukture baze podataka.

Primer 1.18 — Pogledi sistemskog kataloga

```
list tables for schema syscat;
```

Rezultat:

TABNAME	TABSCHEMA	DESCRIPTION
ATTRIBUTES	SYSCAT	View
AUDITPOLICIES	SYSCAT	View
AUDITUSE	SYSCAT	View
BUFFERPOOLDBPARTITIONS	SYSCAT	View
BUFFERPOOLEXCEPTIONS	SYSCAT	View

TABNAME	TABSCHEMA	DESCRIPTION
...	...	...
XSROBJECTCOMPONENTS	SYSCAT	View
XSROBJECTDEP	SYSCAT	View
XSROBJECTDETAILS	SYSCAT	View
XSROBJECTHIERARCHIES	SYSCAT	View
XSROBJECTS	SYSCAT	View

Ukupno: 155 redova × 3 kolona; prikaz je skraćen.

Objašnjenje: U sistemu Db2 ti podaci nalaze se u rezervisanim shemama `SYSIBM`, `SYSIBMADM`, `SYSPUBLIC`, `SYSCAT`, `SYSSTAT` i `SYSTOOLS`. Za korisnike su najznačajniji podaci u shemi `SYSCAT`, koja sadrži tzv. **sistemski katalog** — skup pogleda koji opisuju sve objekte baze podataka.

Primer 1.19 — Sve tabele sheme DA

```
select *
from syscat.tables
where tabschema = 'DA';
```

Rezultat: po jedan red za svaku tabelu sheme `DA` (rezultat nije prikazan zbog obima).

Objašnjenje: Na primer, tabela `SYSCAT.TABLES` (tačnije, u pitanju je *pogled*, o čemu će biti reči kasnije) sadrži po jedan red za svaku tabelu u bazi podataka, a `SYSCAT.COLUMNS` po jedan red za svaku kolonu svake tabele. Pošto su to obične tabele (pogledi), čitaju se istom naredbom `SELECT` kao i korisničke tabele. Imena shema i tabela u katalogu zapisana su velikim slovima, pa se u uslovima restrikcije navode kao `'DA'` i `'DOSIJE'`.

Primer 1.20 — Sve kolone tabele DA.DOSIJE

```
select *
from syscat.columns
where tabschema = 'DA'
  and tabname = 'DOSIJE';
```

Rezultat:

TABSCHEMA	TABNAME	COLNAME	COLNO	TYPESCHEMA	TYPENAME	LENGTH	SCALE
DA	DOSIJE	DATUPISA	7	SYSIBM	DATE	4	0
DA	DOSIJE	DATDIPLOMIRANJA	8	SYSIBM	DATE	4	0

TABSCHEMA	TABNAME	COLNAME	COLNO	TYPESCHEMA	TYPENAME	LENGTH	SCALE
DA	DOSIJE	INDEKS	0	SYSIBM	INTEGER	4	0
DA	DOSIJE	IDPROGRAMA	1	SYSIBM	INTEGER	4	0
DA	DOSIJE	IME	2	SYSIBM	VARCHAR	50	0
DA	DOSIJE	PREZIME	3	SYSIBM	VARCHAR	50	0
DA	DOSIJE	POL	4	SYSIBM	CHARACTER	1	0
DA	DOSIJE	MESTORODJENJA	5	SYSIBM	VARCHAR	50	0
DA	DOSIJE	IDSTATUSA	6	SYSIBM	SMALLINT	2	0

Napomena: prikazano je prvih 8 od 58 kolona.

Pogled `SYSCAT.COLUMNS` ima veliki broj kolona; među najvažnijima su `COLNAME` (ime kolone), `COLNO` (redni broj kolone u tabeli, počev od 0), `TYPENAME` (ime tipa), `LENGTH`, `SCALE`, `DEFAULT` (podrazumevana vrednost) i `NULLS` (da li su dozvoljene nedefinisane vrednosti).

8. Nedefinisane vrednosti

U osnovi relacionih baza podataka i jezika SQL nalaze se pojmovi skupa i relacije. Ipak, između formalnog relacionog modela i SQL-a postoje značajne razlike. Dva ključna odstupanja SQL-a od formalne definicije relacionog modela su:

1. **nedefinisane vrednosti** i
2. **ponavljanje redova**.

Oba odstupanja donose sa sobom i mnoštvo specijalnih slučajeva, koji su uspostavljeni više prema praktičnosti primene nego prema teorijskom modelu. Ovaj odeljak se bavi nedefinisanim vrednostima, a naredni ponavljanjem redova.

8.1. Pojam nedefinisane vrednosti

U nekim slučajevima korisno je da baza podataka može da sadrži tzv. **nedefinisane vrednosti**. Postoje i teorijski čistiji načini da se postigne isti cilj, o kojima će biti reči kasnije, ali postoje i opravdanja za njihovo postojanje.

Primer 1.21 — Sve ispitne prijave jednog studenta

```
-- Svi prijavljeni ispiti studenta sa brojem indeksa 20180050
select *
from ispit
where indeks = 20180050;
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2018	jan1	20180050	1594	n	NULL	NULL	NULL
2018	jan1	20180050	1595	n	NULL	NULL	NULL
2018	jan1	20180050	2168	o	2019-01-14	71	8
2018	jan2	20180050	1595	o	2019-02-09	11	5
2018	jun1	20180050	1595	o	2019-06-16	38	5
2018	jun1	20180050	2167	n	NULL	NULL	NULL
2018	jun2	20180050	2166	n	NULL	NULL	NULL
2018	jun2	20180050	2167	o	2019-07-09	49	5
2018	sep1	20180050	1595	o	2019-09-05	63	7
2018	sep1	20180050	2167	o	2019-09-12	55	6

Objašnjenje: Posmatrajmo sve ispitne prijave studenta sa brojem indeksa 20180050, obraćajući posebnu pažnju na redove sa statusom `n` (student nije izašao na ispit).

Ako student nije izašao na ispit, nema smisla govoriti o datumu polaganja, broju poena i oceni. Na svim tim mestima nalaze se nedefinisane vrednosti, koje se u SQL-u označavaju sa `NULL`. Vrednost `NULL` nije vrednost nijednog tipa u uobičajenom smislu, već oznaka da vrednost nedostaje; ona nema sopstveni tip, ali se tumači kao da je onog tipa koji zahteva kontekst u kome se nalazi. Za nedefinisanu vrednost logičkog tipa postoji posebna oznaka: `UNKNOWN` (*nepoznato*).

8.2. Nedefinisane vrednosti i restrikcija

Primer 1.22 — Položeni ispiti

```
-- Svi položeni ispiti studenta sa brojem indeksa 20180050
-- (nije do kraja ispravno, jer bi moralo da se proveri status='o')
select *
from ispit
where indeks = 20180050
  and ocena > 5;
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2018	jan1	20180050	2168	o	2019-01-14	71	8
2018	sep1	20180050	1595	o	2019-09-05	63	7
2018	sep1	20180050	2167	o	2019-09-12	55	6

Objašnjenje: Pokušajmo da izdvojimo sva uspešna, a zatim i sva neuspešna polaganja ispita ovog studenta.

Primer 1.23 — Nepoloženi ispiti

```
-- Svi nepoloženi ispiti studenta sa brojem indeksa 20180050
select *
from ispit
where indeks = 20180050
  and not ( ocena > 5 );
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2018	jan2	20180050	1595	o	2019-02-09	11	5
2018	jun1	20180050	1595	o	2019-06-16	38	5
2018	jun2	20180050	2167	o	2019-07-09	49	5

Objašnjenje: Komentar u upitu ukazuje na to da rešenje nije sasvim ispravno: ocena se upisuje samo za ispite na koje je student izašao, pa bi pravilno bilo proveriti i da je status jednak `'o'`. U ovim podacima to ne menja rezultat.

Student ima ukupno 10 prijava, od kojih su 3 izdvojene prvim, a 3 drugim upitom. Preostale 4 prijave (one sa nedefinisanom ocenom) nisu izdvojene ni jednim od ta dva upita, iako bi se moglo očekivati da svaki red zadovoljava ili uslov `ocena > 5` ili njegovu negaciju.

Objašnjenje je u tome što restrikcija izdvaja samo one redove za koje je uslov **tačan**. Postojanje nedefinisanih vrednosti uvodi **trovalentnu (ternarnu) logiku**, u kojoj osim vrednosti *tačno* (T) i *netačno* (F) postoji i vrednost *nepoznato* (U, `UNKNOWN`). Svako poređenje u kome učestvuje nedefinisana vrednost ima za rezultat *nepoznato*. Logički operatori su definisani sledećim tablicama:

AND	T	U	F
T	T	U	F
U	U	U	F
F	F	F	F

OR	T	U	F
T	T	T	T

OR	T	U	F
U	T	U	U
F	T	U	F

x	NOT x
T	F
U	U
F	T

Za red u kome je ocena nedefinisana, poređenje `ocena > 5` ima vrednost U, a njegova negacija `NOT (ocena > 5)` takođe vrednost U. Kako nijedan od ta dva uslova nije tačan, takav red se ne izdvaja ni u jednom od dva upita.

8.3. Predikati IS NULL i IS NOT NULL

Primer 1.24 — Ispiti u kojima ocena nije definisana

```
-- Svi ispiti u kojima ocena nije definisana
select *
from ispit
where indeks = 20180050
  and ocena is null;
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2018	jan1	20180050	1594	n	NULL	NULL	NULL
2018	jan1	20180050	1595	n	NULL	NULL	NULL
2018	jun1	20180050	2167	n	NULL	NULL	NULL
2018	jun2	20180050	2166	n	NULL	NULL	NULL

Objašnjenje: Jedini ispravan način da se utvrdi da li je neka vrednost nedefinisana jesu predikati `x IS NULL` i `x IS NOT NULL`. Za razliku od poređenja, ovi predikati uvek imaju vrednost *tačno* ili *netačno*, nikada *nepoznato*.

Primer 1.25 — Neispravno poređenje sa NULL

```
select *
from ispit
where indeks = 20180050
  and ocena = null;
```

Rezultat:

Upit ne izdvaja nijedan red (rezultat je prazan).

Objašnjenje: Poređenja oblika `x = NULL` nemaju smisla, jer je njihova logička vrednost uvek *nepoznato* — bez obzira na vrednost `x`, pa i kada je `x` nedefinisano. Zbog toga ovaj upit ne izdvaja nijedan red.

Primer 1.26 — Uslov (ocena > 5) IS NULL

```
-- ocena is null === ocena > 5 is null
select *
from ispit
where indeks = 20180050
  and ocena > 5 is null;
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2018	jan1	20180050	1594	n	NULL	NULL	NULL
2018	jan1	20180050	1595	n	NULL	NULL	NULL
2018	jun1	20180050	2167	n	NULL	NULL	NULL
2018	jun2	20180050	2166	n	NULL	NULL	NULL

Objašnjenje: Pošto je poređenje izraz logičkog tipa, a nepoznata logička vrednost je takođe nedefinisana vrednost, smisla ima čak i izraz oblika `(x > 5) IS NULL`. On je tačan ako i samo ako poređenje `x > 5` ima vrednost *nepoznato*, tj. ako i samo ako je `x` nedefinisano. Zato je uslov u ovom upitu ekvivalentan uslovu `ocena IS NULL`.

Primer 1.27 — Uslov (ocena = null) IS NULL

```
-- Ovde je drugi deo uslova uvek zadovoljen
select *
from ispit
where indeks = 20180050
  and ocena = null is null;
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2018	jan1	20180050	1594	n	NULL	NULL	NULL
2018	jan1	20180050	1595	n	NULL	NULL	NULL
2018	jan1	20180050	2168	o	2019-01-14	71	8
2018	jan2	20180050	1595	o	2019-02-09	11	5
2018	jun1	20180050	1595	o	2019-06-16	38	5
2018	jun1	20180050	2167	n	NULL	NULL	NULL
2018	jun2	20180050	2166	n	NULL	NULL	NULL
2018	jun2	20180050	2167	o	2019-07-09	49	5
2018	sep1	20180050	1595	o	2019-09-05	63	7
2018	sep1	20180050	2167	o	2019-09-12	55	6

Objašnjenje: Nasuprot tome, poređenje `ocena = null` uvek ima vrednost *nepoznato*, pa je izraz `(ocena = null) IS NULL` uvek tačan. Drugi deo uslova u ovom upitu je, dakle, zadovoljen za svaki red, i upit izdvaja sve prijave studenta.

9. Ponavljanje redova

Iako bi tabele u osnovi trebalo da budu **skupovi** redova, SQL je definisan tako da upiti izdvajaju *sve* redove koji zadovoljavaju uslov restrikcije, što može da uključi i više identičnih redova. Rezultat upita je, dakle, u opštem slučaju **multiskup** (skup sa ponavljanjem) redova.

Primer 1.28 — Svi studenti koji se zovu Milena

```
-- Svi studenti koji se zovu Milena
select Ime, Prezime
from Dosije
where Ime = 'Milena'
order by prezime;
```

Rezultat:

IME	PREZIME
Milena	Bosnjakovic
Milena	Brankovic
Milena	Dimitrijevic
Milena	Filipovic

IME	PREZIME
Milena	Marinkovic
Milena	Markovic
Milena	Markovic
Milena	Medvedev
Milena	Milovic
Milena	Miric
Milena	Mitrovic
Milena	Nenadic

Napomena: prikazano je prvih 12 od 22 redova.

Objašnjenje: Na primer, izdvojimo imena i prezimena svih studentkinja koje se zovu Milena. (Klauzula `ORDER BY`, koja uređuje rezultat, objašnjena je u odeljku 11; ovde je navedena da bi ponovljeni redovi bili jedan do drugog.)

U rezultatu postoje ponovljeni redovi: dva puta se pojavljuje Milena Marković i dva puta Milena Nikolić. To su različiti studenti (sa različitim brojevima indeksa), ali su izdvojeni redovi identični, jer projekcija nije obuhvatila kolonu po kojoj se razlikuju.

U nekim slučajevima ponavljanje je korisno, ali je često potrebno da se rezultat ponaša kao skup različitih redova. Tada se neposredno iza ključne reči `SELECT` navodi ključna reč `DISTINCT`, kojom se nalaže da se iz rezultata izbace ponovljeni redovi. Dva reda se smatraju ponovljenim ako imaju jednake vrednosti u svim kolonama rezultata:

```
SELECT DISTINCT <lista kolona>
FROM ...
```

Primer 1.29 — Različiti parovi (Ime, Prezime)

```
-- Svi razliciti parovi Ime,Prezime studenata koji se zovu Milena
select distinct Ime, Prezime
from Dosije
where Ime = 'Milena'
order by prezime;
```

Rezultat:

IME	PREZIME
Milena	Bosnjakovic
Milena	Brankovic
Milena	Dimitrijevic
Milena	Filipovic
Milena	Marinkovic
Milena	Markovic
Milena	Medvedev
Milena	Milovic
Milena	Miric
Milena	Mitrovic
Milena	Nenadic
Milena	Nikolic

Napomena: prikazano je prvih 12 od 20 redova.

Primer 1.30 — Sve ocene studenta

```
-- Sve ocene studenta
select ocena
from ispit
where indeks = 20180050;
```

Rezultat:

OCENA
NULL
NULL
8
5
5
NULL

OCENA
NULL
5
7
6

Objašnjenje: Izbacivanje ponovljenih redova radi na specifičan način kada postoje nedefinisane vrednosti. Kao što je pokazano, poređenje nedefinisane vrednosti sa bilo kojom vrednošću, pa i sa drugom nedefinisanom vrednošću, ima za rezultat *nepoznato*. Međutim, pri izbacivanju ponavljanja **sve nedefinisane vrednosti u okviru jedne kolone tretiraju se kao jednake**. Ako izdvojimo sve ocene studenta sa brojem indeksa 20180050, videćemo da ima ponavljanja.

Primer 1.31 — Različite ocene studenta

```
-- Sve razlicite ocene studenta
select distinct ocena
from ispit
where indeks = 20180050;
```

Rezultat:

OCENA
5
6
7
8
NULL

Objašnjenje: Ako se navede ključna reč `DISTINCT`, ponavljanja se izbacuju. Izbacuju se i ponovljene nedefinisane vrednosti, osim jedne, pa rezultat sadrži tačno jedan red sa vrednošću `NULL`.

Ključna reč `DISTINCT` ima i druge uloge (npr. u kolonskim funkcijama), o kojima će biti reči kasnije.

10. Izrazi i imenovanje kolona

U klauzuli `SELECT` se osim postojećih kolona mogu navoditi i **izrazi**, koji se izračunavaju za svaki izdvojeni red. SQL ima veliki broj ugrađenih operatora i funkcija, a svaki relacioni SUBP dodaje i neke svoje. Većina sistema omogućava i dodavanje tzv. *korisnički definisanih funkcija*.

Primer 1.32 — Izračunata kolona

```
-- Izracunavamo kolonu spajanjem imena i prezimena
select ime, prezime, ime || ' ' || prezime
from dosije;
--limit 5 -- o ovome kasnije...
```

Rezultat:

IME	PREZIME	3
Nemanja	Nicic	Nemanja Nicic
Andrijana	Beara	Andrijana Beara
Milos	Milenkovic	Milos Milenkovic
Valentina	Stojanov	Valentina Stojanov
Aida	Babic	Aida Babic
…	…	…
Tara	Bozinovic	Tara Bozinovic
Ana	Ilic	Ana Ilic
Tamara	Zejak	Tamara Zejak
Uros	Stanojkov	Uros Stanojkov
Milos	Djokic	Milos Djokic

Ukupno: 3496 redova × 3 kolona; prikaz je skraćen.

Objašnjenje: U ovom primeru izračunava se nova kolona spajanjem imena, razmaka i prezimena. Operator `||` označava **nadovezivanje** (konkatenaciju) niski. Izračunata kolona nema ime, pa je sistem označava njenim rednim brojem u rezultatu (`3`). Red `--limit 5` je komentar; ograničavanje broja redova rezultata biće obrađeno kasnije.

Kolone koje predstavljaju rezultat izračunavanja, ali i sve ostale izdvojene kolone, mogu se imenovati. Ime kolone navodi se iza izraza, uz opcionu ključnu reč `AS`:

```
<izraz> [AS] <ime kolone>
```

Primer 1.33 — Imenovanje izračunate kolone

```
-- Imenujemo izracunatu kolonu
select ime || ' ' || prezime as ime_i_prezime
from dosije;
-- limit 15
```

Rezultat:

IME_I_PREZIME
Nemanja Nicic
Andrijana Beara
Milos Milenkovic
Valentina Stojanov
Aida Babic
…
Tara Bozinovic
Ana Ilic
Tamara Zejak
Uros Stanojkov
Milos Djokic

Ukupno: 3496 redova × 1 kolona; prikaz je skraćen.

Objašnjenje: Dodeljeno ime postaje ime kolone u rezultatu i u njegovom opisu, ali ne menja nijednu tabelu u bazi podataka.

11. Uređivanje rezultata upita

Redovi u tabelama, pa ni u rezultatima upita, nemaju unapred određen redosled: ako se redosled izričito ne zahteva, sistem može da vrati redove bilo kojim redosledom. Željeni redosled zadaje se klauzulom `ORDER BY`, koja se navodi na kraju upita, posle klauzule `WHERE`:

```
ORDER BY {<ime> | <rbr> | <izraz>} [ASC | DESC] [NULLS {FIRST | LAST}] , ...
```

Kriterijum uređivanja može biti ime kolone rezultata, redni broj kolone u rezultatu (počev od 1) ili proizvoljan izraz. Za svaki kriterijum posebno se zadaje smer: `ASC` — rastući (podrazumevano) ili `DESC` — opadajući. Ako je navedeno više kriterijuma, redovi se najpre uređuju po prvom, redovi sa jednakom vrednošću prvog kriterijuma po drugom, i tako dalje.

Primer 1.34 — Uređivanje po koloni i po izrazu

```
-- Uređivanje rezultata upita
select *
from semestar
order by skgodina desc, -semestar;
```

Rezultat:

SKGODINA	SEMESTAR
2020	2
2020	1
2019	2
2019	1
2018	2
2018	1
2017	2
2017	1
2016	2
2016	1
2015	2
2015	1

Objašnjenje: U ovom primeru rezultat se uređuje najpre po koloni `SkGodina` u opadajućem redosledu, a zatim, unutar iste školske godine, po izrazu `-Semestar` u rastućem redosledu. Rastući redosled po vrednosti `-Semestar` isti je kao opadajući redosled po vrednosti `Semestar`, pa se u okviru svake godine najpre navodi drugi, a zatim prvi semestar.

Za svaki kriterijum uređivanja može se posebno odrediti i da li će redovi sa nedefinisanom vrednošću biti na početku (`NULLS FIRST`) ili na kraju (`NULLS LAST`). Ako se to ne navede, sistem Db2 smatra nedefinisanu vrednost većom od svih definisanih vrednosti, pa se ona pri rastućem uređivanju nalazi na kraju, a pri opadajućem na početku.

Primer 1.35 — NULLS FIRST i uređivanje po rednom broju kolone

```
select ocena, datpolaganja
from ispit
where indeks = 20180050
order by ocena nulls first, 2 desc;
```

Rezultat:

OCENA	DATPOLAGANJA
NULL	NULL
NULL	NULL
NULL	NULL
NULL	NULL
5	2019-07-09
5	2019-06-16
5	2019-02-09
6	2019-09-12
7	2019-09-05
8	2019-01-14

Objašnjenje: U ovom primeru redovi se uređuju po oceni rastuće, pri čemu redovi sa nedefinisanom ocenom dolaze prvi, a redovi sa jednakom ocenom uređuju se opadajuće po drugoj koloni rezultata (`datpolaganja`), navedenoj rednim brojem `2`. To se vidi kod tri ispita sa ocenom 5, koji su uređeni od najkasnijeg ka najranijem datumu.

Čas 2 — Ograničavanje rezultata, tipovi, izrazi i spajanje tabela

Na početku svakog časa potrebno je povezati se sa bazom podataka, na isti način kao na prvom času.

Primer 2.1 — Povezivanje sa bazom podataka

```
connect to stud2020 user student using abcdef;
set schema da;
```

Rezultat: veza sa bazom podataka je uspostavljena, a tekuća shema postavljena.

1. Ograničavanje broja izdvojenih redova

Standard SQL-a predviđa da se broj redova rezultata ograniči klauzulama `OFFSET` i `FETCH`, koje se navode na kraju upita:

```
SELECT ...
[OFFSET <m> ROWS]
FETCH {FIRST | NEXT} [<n>] {ROW | ROWS} ONLY
```

Klauzula `OFFSET <m> ROWS` nalaže da se preskoči prvih *m* redova rezultata, a klauzula `FETCH ... ONLY` da se izdvoji najviše *n* narednih redova. Ključne reči `FIRST` i `NEXT`, kao i `ROW` i `ROWS`, međusobno su sinonimi i služe samo čitljivosti. Sistem Db2 podržava i alternativne, nestandardne zapise istog ograničenja:

```
LIMIT <n> [OFFSET <m>]
LIMIT <m>, <n>
```

Primer 2.2 — Preskakanje i ograničavanje broja redova

```
select *
from dosije
offset 5 rows fetch next 10 rows only;

--offset 5 rows
--fetch next 10 rows only

--limit 10 offset 5
--limit 5, 10
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150071	101	Andjela	Beric	z	Beograd (Savski venac)	-2	2015-07-06
20150275	101	Lola	Zivic	z	Beograd (Zvezdara)	-1	2015-07-06
20150177	101	Blagoje	Kostic	m	Beograd (Savski venac)	2	2015-07-06

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150135	101	Vukasin	Zivkovic	m	Beograd (Zvezdara)	2	2015-07-06
20150357	101	Milos	Simic	m	Beograd (Savski venac)	-2	2015-07-06
20150236	101	Sara	Ruzic	z	Uzice	-2	2015-07-06

Napomena: prikazano je prvih 8 od 9 kolona; prikazano je prvih 6 od 10 redova.

Objašnjenje: Ovaj upit preskoči prvih 5 redova tabele `dosije` i izdvaja narednih 10. Zakomentarisani redovi prikazuju ekvivalentne alternativne zapise. Treba imati u vidu da bez klauzule `ORDER BY` redosled redova nije određen, pa nije određeno ni koji će redovi biti preskočeni, a koji izdvojeni; u praksi se ograničavanje broja redova gotovo uvek kombinuje sa uređivanjem.

2. Optimizacija za određen broj redova

U nekim slučajevima rezultat upita može biti veoma obiman, a da se pri njegovoj upotrebi čita samo nekoliko prvih redova. Tada se SUBP-u može sugerisati da će se verovatno čitati samo deo rezultata, što može da utiče na izbor načina izvršavanja upita. To se postiže klauzulom:

```
OPTIMIZE FOR <n> {ROW | ROWS}
```

Primer 2.3 — Sugestija optimizatoru

```
select *
from dosije
optimize for 5 rows;
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150069	101	Nemanja	Nicic	m	Sabac	-2	2015-07-06
20150090	103	Andrijana	Beara	z	Sabac	-2	2015-07-06
20150320	103	Milos	Milenkovic	m	Krusevac	2	2015-07-06
20150185	101	Valentina	Stojanov	z	Beograd (Savski venac)	2	2015-07-06

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150230	103	Aida	Babic	z	Cacak	2	2015-07-06
...	...	...	...	...	...	...	...

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 3496 redova × 9 kolona; prikaz je skraćen.

Objašnjenje: Ovom klauzulom sistem se obaveštava da će se čitati približno *n* redova i da plan izvršavanja treba optimizovati u skladu sa tom pretpostavkom (npr. tako da se prvi redovi dobiju što pre). Za razliku od klauzula `FETCH` i `LIMIT`, klauzula `OPTIMIZE FOR` **ne utiče na sadržaj rezultata**: i dalje se može pročitati manje ili više od *n* redova. Klauzula ne može da stoji u podupitu, već samo na kraju upitne naredbe. U ovom primeru rezultat i dalje sadrži sve redove tabele (3496).

3. Tipovi podataka

Tipovi podataka u SQL-u su veoma bogati, kako zbog raznovrsnosti podataka koje je potrebno podržati u različitim vrstama aplikacija, tako i zbog postojanja više varijanti pojedinih tipova. Pregled osnovnih tipova sistema Db2 dat je u narednoj tabeli; potpun pregled nalazi se u dokumentaciji sistema.

Grupa	Tip	Opis
Celobrojni	`SMALLINT`	16-bitni ceo broj
Celobrojni	`INTEGER`, `INT`	32-bitni ceo broj
Celobrojni	`BIGINT`	64-bitni ceo broj
Pokretni zarez	`REAL`	32-bitni broj u pokretnom zarezu
Pokretni zarez	`DOUBLE`	64-bitni broj u pokretnom zarezu
Pokretni zarez	`DECFLOAT(n)`	decimalni broj u pokretnom zarezu (IEEE 754r), preciznosti n = 16 ili 34 cifre; opseg do 10^6144 za n = 34
Fiksni zarez	`DECIMAL(n,m)`	n — ukupan broj cifara, m — broj cifara iza decimalnog zareza, 0 ⩽ m ⩽ n ⩽ 31
Jednobajtne niske	`CHAR(n)`	niska fiksne dužine, 1 ⩽ n ⩽ 255
Jednobajtne niske	`VARCHAR(n)`	niska promenljive dužine, n ⩽ 32672

Grupa	Tip	Opis
Jednobajtne niske	`CLOB(n)`	velika niska, do 2 GiB − 1
Dvobajtne niske	`GRAPHIC(n)`	niska fiksne dužine, 1 ⩽ n ⩽ 127
Dvobajtne niske	`VARGRAPHIC(n)`	niska promenljive dužine, n ⩽ 16336
Dvobajtne niske	`DBCLOB(n)`	velika niska, do 1 GiB − 1
Binarne niske	`BINARY(n)`	niska bajtova fiksne dužine, 1 ⩽ n ⩽ 255
Binarne niske	`VARBINARY(n)`	niska bajtova promenljive dužine, n ⩽ 32672
Binarne niske	`BLOB(n)`	velika niska bajtova, do 2 GiB − 1
Vremenski	`DATE`, `TIME`, `TIMESTAMP`	datum, vreme, datum i vreme
Logički	`BOOLEAN`	vrednosti `TRUE` i `FALSE`

4. Konverzije tipova

SQL je relativno strogo tipiziran jezik. Kada je to moguće, konverzije tipova izvode se automatski (implicitno), ali je u mnogim situacijama potrebno konverziju navesti eksplicitno. Postoji opšti izraz za konverziju, kao i veći broj funkcija za konverziju u pojedinačne tipove.

4.1. Opšta konverzija — izraz CAST

Opšti oblik konverzije je izraz:

```
CAST( <izraz> AS <tip> )
```

koji vrednost datog izraza konvertuje u navedeni tip. Pri konverziji realnog broja u ceo broj razlomljeni deo se odbacuje (ne zaokružuje se), pa se `42.987` konvertuje u `42`:

Primer 2.4 — Konverzija celog broja u DOUBLE

```
values cast( 1 as double );
```

Rezultat:

1
1.0

Primer 2.5 — Konverzija decimalnog broja u INT

```
values cast( 42.987 as int );
```

Rezultat:

1
42

4.2. Specifične konverzije

Primer 2.6 — Konverzija niski u brojeve

```
values (
    int('42'),
    int('42.987'),
    double('20'),
    dec('42')
);
```

Rezultat:

1	2	3	4
42	42	20.0	42.0

Objašnjenje: Za veliki broj tipova postoje funkcije istog ili sličnog imena kao tip, koje konvertuju argument u odgovarajući tip: `CHAR`, `DECIMAL` (`DEC`), `DATE`, `DOUBLE`, `FLOAT`, `INT`, `TIME` i mnoge druge. U ovom primeru niske se konvertuju u brojeve; i ovde se pri konverziji niske `'42.987'` u ceo broj razlomljeni deo odbacuje.

Funkcija `DATE` prihvata niske u više formata (`DD.MM.YYYY`, `MM/DD/YYYY` i `YYYY-MM-DD`), ali i ceo broj, koji se tumači kao redni broj dana računajući od 1. 1. 0001. (vrednost 1 odgovara tom datumu). Funkcija `CHAR` primenjena na vrednost tipa `TIMESTAMP` daje nisku u standardnom formatu sistema Db2 (`YYYY-MM-DD-HH.MM.SS.nnnnnn`); ovde je korišćena da bi se vrednost tipa `TIMESTAMP` prikazala kao niska u tom formatu.

Primer 2.7 — Konverzije u datum i vreme

```
values (
    date('15.3.2020'),
    date('3/15/2020'),
    date('2020-03-15'),
    date(737711),
    time('12:23'),
    char(timestamp('2023-09-17 12:25:10'))
);
```

Rezultat:

1	2	3	4	5	6
2020-03-15	2020-03-15	2020-03-15	2020-10-13	12:23:00	2023-09-17-12.25.10.000000

5. Izrazi

Izrazi u SQL-u koriste se na sličan način kao u programskim jezicima, ali sa nekim specifičnostima. Elementi izraza su, između ostalog:

- konstante;
- imena kolona;
- posebni registri;
- izrazi u zagradama;
- podupiti;
- izraz `CASE`;
- izraz `CAST`;
- OLAP izrazi.

Od nastanka SQL-a dodati su mnogi novi elementi (objektno-orijentisani elementi, podrška za XML i dr.), pa je svako proširenje dodatno usložnjavalo i jezik u celini i pojam izraza. Ovde nisu navedeni svi elementi izraza, jer ih ima mnogo i delimično zavise od implementacije, već samo one specifičnosti koje su najvažnije za dalji rad.

5.1. Realni brojevi

Realni brojevi mogu da se zapisuju i u obliku sa eksponentom:

```
<znak><ceo deo>[.[<decimalni deo>]][E<eksponent>]
```

Postoje i posebne realne vrednosti:

- `INFINITY` ili `INF` — beskonačna vrednost, može biti označena (`-INF`);
- `NAN` (*Not A Number*) — vrednost koja ne predstavlja broj, ali ne izaziva grešku;

- `SNAN` (*Signaling NaN*) — vrednost koja ne predstavlja broj i izaziva grešku pri upotrebi.

5.2. Niske

Niske se zapisuju između apostrofa. Apostrof unutar niske zapisuje se udvojeno (`'O''Brien'`). Postoje i posebni oblici zapisa:

- `X'...'` — sadržaj čini paran broj heksadekadnih cifara, pri čemu svaki par cifara predstavlja jedan bajt niske; praznine u zapisu su dopuštene i zanemaruju se;
- `U&'...' [UESCAPE '...']` — Unicode niska; može da sadrži znakove zapisane u UTF-8 ili kodove Unicode znakova u obliku `\<4 heksadekadne cifre>` ili `\+<6 heksadekadnih cifara>`;
- `G'...'` ili `N'...'` — niska tipa `GRAPHIC` (dvobajtni znakovi);
- `BX'...'` — binarna niska tipa `BINARY`, zapisana heksadekadnim kodovima bajtova.

Primer 2.8 — Različiti zapisi niski

```
values
    'Obicna niska',
    'Обична ниска може да садржи УТФ-8',
    X'404142616263',
    U&'Знак € може да се запише и овако: \20AC \000aovo je bio novi red,
\000aa ovo su neke note \266B\266A';
```

Rezultat:

1
Obicna niska
Обична ниска може да садржи УТФ-8
@ABabc
Знак € може да се запише и овако: € ovo je bi...

Objašnjenje: U ovom primeru niska `X'404142616263'` predstavlja bajtove sa kodovima 0x40, 0x41, 0x42, 0x61, 0x62 i 0x63, tj. nisku `@ABabc`. U Unicode niski kod `\20AC` predstavlja znak €, `\000a` znak za novi red, a `\266B` i `\266A` notne znakove ♫ i ♪. Prikaz poslednje vrednosti je skraćen, jer ona sadrži i dva znaka za novi red.

5.3. Datum i vreme

Konstanta tipa datuma zapisuje se u obliku `DATE 'YYYY-MM-DD'`. Pored toga, niska bez prefiksa `DATE` automatski se konvertuje u datum kada to zahteva kontekst, ako je zapisana u jednom od tri oblika: `'YYYY-MM-DD'`, `'DD.MM.YYYY'` ili `'MM/DD/YYYY'`.

Vreme se zapisuje kao `TIME 'HH:MM[:SS]'`, a datum i vreme (tip `TIMESTAMP`) kao `TIMESTAMP 'YYYY-MM-DD HH:MM:SS.nnnnnn'` ili `TIMESTAMP 'YYYY-MM-DD-HH.MM.SS.nnnnnn'`. I u ovim slučajevima niske odgovarajućeg oblika automatski se konvertuju kada to zahteva kontekst.

Primer 2.9 — Zapisi datuma

```
values
    -- bar jedan red mora da ima eksplicitno oznacen datum
    date '2020-10-15',
    '2020-10-15',
    '15.10.2020',
    '10/15/2020';
```

Rezultat:

1
2020-10-15
2020-10-15
2020-10-15
2020-10-15

Objašnjenje: U ovom primeru tip kolone određen je prvim redom (eksplicitno navedenim datumom), pa se niske u ostalim redovima konvertuju u datume. Kako navodi komentar, bar jedan red mora da sadrži eksplicitno označen datum; inače bi sve vrednosti bile niske.

5.4. Posebni registri

Posebni registri predstavljaju unapred definisane globalne podatke, koji se mogu koristiti u upitima i drugim naredbama SQL-a. Nisu promenljive, jer ih korisnik ne može menjati. Neki od najvažnijih su:

- `CURRENT_DATE` (`CURRENT DATE`) — tekući datum;
- `CURRENT_TIME` (`CURRENT TIME`) — tekuće vreme;
- `CURRENT_TIMESTAMP`, `NOW` — sinonimi, tekući datum i vreme;
- `CURRENT_USER`, `USER` — sinonimi, trenutno autorizovani korisnik.

Važno: koliko god puta se u jednoj naredbi upotrebe registri koji čitaju vreme, svaki put vraćaju istu vrednost, bez obzira na trajanje izvršavanja naredbe.

Primer 2.10 — Posebni registri

```
values( current_date, current_time, now, user );
```

Rezultat: vrednosti zavise od trenutka izvršavanja.

5.5. Operatori

SQL obuhvata sledeće skalarne operatore:

Operator	Značenje
`+`, `-`, `*`, `/`	osnovne aritmetičke operacije
`%`	ostatak pri celobrojnom deljenju
`**`	stepenovanje
`||`, `CONCAT`	nadovezivanje niski (sinonimi)
`&`, `|`, `^`	bitovske operacije konjunkcije, disjunkcije i ekskluzivne disjunkcije nad celim brojevima

Primer 2.11 — Skalarni operatori

```
values (
    'a' concat 'b',
    2 ** 3,
    19 % 4,
    19 | 6,
    5 > 3 and 2 < 7
);
```

Rezultat:

1	2	3	4	5
ab	8	3	23	True

Objašnjenje: U ovom primeru je $2^3 = 8$, ostatak pri deljenju 19 sa 4 jednak je 3, a bitovska disjunkcija 19 | 6 jednaka je 23 (binarno 10011 | 00110 = 10111). Poslednji izraz je logičkog tipa i ima vrednost *tačno*.

5.6. Operacije i funkcije sa datumom i vremenom

Neke od operacija sa datumom i vremenom koje se ovde navode specifične su za sistem Db2. Pri sabiranju i oduzimanju datuma i vremena brojevi se moraju *labelirati*, tj. mora se eksplicitno navesti da li predstavljaju dane, mesece, godine, sate, minute, sekunde ili mikrosekunde (`DAYS` , `MONTHS` , `YEARS` , `HOURS` , `MINUTES` , `SECONDS` , `MICROSECONDS`).

Primer 2.12 — Datum posle 1234 dana

```
-- koji datum će biti za 1234 dana
values current_date + 1234 days;
```

Rezultat:

1
2027-02-25

Primer 2.13 — Datum i vreme pre 123456 minuta

```
-- koji datum i vreme su bili pre 123456 minuta
values now - 123456 minutes;
```

Rezultat: vrednost zavisi od trenutka izvršavanja.

Razlika dva datuma je tzv. **trajanje datuma**: broj tipa DECIMAL(8,0) oblika

```
10000 * <broj celih godina> + 100 * <broj celih meseci> + <broj celih dana>
```

tako da, ako je `D1 - D2 = R` , važi:

```
D2 + (R / 10000) YEARS + (R / 100) % 100 MONTHS + R % 100 DAYS = D1
```

Na sličan način se računaju i razlike vremena i datuma i vremena. Za vreme je rezultat oblika `10000 * <sati> + 100 * <minuti> + <sekunde>` , a za datum i vreme oblika `YYYYMMDDHHMMSS.nnnnnn` , tj. `10000000000 * <godine> + 100000000 * <meseci> + 1000000 * <dani> + 10000 * <sati> + 100 * <minuti> + <sekunde>` , gde se mikrosekunde zapisuju kao razlomljeni deo.

Primer 2.14 — Razlika dva datuma

```
values current date - date '2017-02-01';
```

Rezultat:

1
60809.0

Objašnjenje: U ovom primeru vrednost `60809` znači da je od 1. 2. 2017. do dana izvršavanja upita proteklo 6 godina, 8 meseci i 9 dana.

Primer 2.15 — Razlika datuma i vremena i datuma

```
values now - date '2017-02-01';
```

Rezultat:

1
6.080920e+10

Objašnjenje: Kada se od tekućeg datuma i vremena oduzme datum, dobija se trajanje datuma i vremena. Da bi se lakše pročitali njegovi niži delovi, u Primeru 2.16 od rezultata je oduzeta vrednost 60802000000 (tj. 6 godina, 8 meseci i 2 dana), pa vrednost `7200235.825` znači još 7 dana, 20 sati, 2 minuta i 35,825 sekundi.

Primer 2.16 — Čitanje nižih delova trajanja

```
values now - date '2017-02-01' - 60802000000;
```

Rezultat:

1
7200235.825

Primer 2.17 — Oduzimanje godina, meseci i dana

```
values current date - 4 years - 8 months - 12 days;
```

Rezultat:

1
2019-01-29

Objašnjenje: Labelirani brojevi mogu se oduzimati i sabirati redom, pri čemu se operacije izvode sleva nadesno.

Postoji veći broj funkcija za rad sa datumom i vremenom:

Funkcija	Rezultat
`YEAR`, `QUARTER`, `MONTH`, `WEEK`, `DAY`	odgovarajuća komponenta datuma (za argumente tipa `DATE` i `TIMESTAMP`)
`AGE(d)`	slično kao `CURRENT_DATE - d`
`DAYNAME`, `DAYOFMONTH`, `DAYOFWEEK`, `DAYOFYEAR`	ime dana, redni broj dana u mesecu, u nedelji i u godini
`MONTHNAME`	ime meseca
`DAYS(d)`	broj dana od 1. 1. 0001. uvećan za 1
`DAYS_BETWEEN(d1, d2)`	broj punih dana od d2 do d1
`FIRST_DAY`, `LAST_DAY`	prvi i poslednji datum u mesecu
`NEXT_WEEK`, `NEXT_MONTH`, `NEXT_QUARTER`, `NEXT_YEAR`	prvi datum u narednoj nedelji, mesecu, tromesečju, godini
`HOUR`, `MINUTE`, `SECOND`, `MICROSECOND`	odgovarajuća komponenta vremena (za argumente tipa `TIME` i `TIMESTAMP`)
`HOURS_BETWEEN(t1, t2)`	broj punih sati od t2 do t1
`MIDNIGHT_SECONDS(t)`	broj punih sekundi od prethodne ponoći

Primer 2.18 — Funkcije za delove datuma

```
values( dayname(now), dayofmonth(now), dayofweek(now), dayofyear(now) );
```

Rezultat:

1	2	3	4
Tuesday	10	3	283

Objašnjenje: Funkcija `DAYOFWEEK` u sistemu Db2 numeriše dane od nedelje (1) do subote (7), pa vrednost 3 u ovom primeru odgovara utorku.

Primer 2.19 — Dužina studiranja diplomiranih studenata u danima

```
-- duzina studiranja diplomaca u danima
select
    days_between(datdiplomiranja, datupisa),
    days(datdiplomiranja) - days(datupisa)
from dosije
where datdiplomiranja is not null
order by 1;
```

Rezultat:

1	2
217	217
225	225
239	239
245	245
245	245
…	…
1911	1911
1913	1913
1913	1913
1913	1913
1913	1913

Ukupno: 405 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: Broj dana između dva datuma može se izračunati funkcijom `DAYS_BETWEEN` ili kao razlika vrednosti funkcije `DAYS` za ta dva datuma. Ovaj upit pokazuje da oba načina daju isti rezultat; klauzula `ORDER BY 1` uređuje rezultat po prvoj koloni.

5.7. Funkcije za rad sa niskama

Funkcija za rad sa niskama ima mnogo. Najvažnije su:

Funkcija	Rezultat
`LENGTH(s)`	dužina niske (u bajtovima, za niske tipa `CHAR` / `VARCHAR`)

Funkcija	Rezultat
`LENGTHB(s)`	dužina niske u bajtovima
`LENGTH2(s)`	dužina niske u 16-bitnim kodnim jedinicama (UTF-16)
`LENGTH4(s)`	dužina niske u 32-bitnim kodnim jedinicama (UTF-32)
`SUBSTR(s, p, n)`	podniska niske s od pozicije p (počev od 1), dužine n; bez n — do kraja niske
`SUBSTRB`, `SUBSTR2`, `SUBSTR4`	isto, sa pozicijom i dužinom u bajtovima, odnosno u 16-bitnim ili 32-bitnim kodnim jedinicama
`LEFT`, `RIGHT`, `STRLEFT`, `STRRIGHT`	početni, odnosno završni deo niske date dužine
`CONCAT(s1, s2)`	nadovezivanje niski (može se koristiti i kao operator)
`SPACE(n)`	niska od n razmaka
`POSSTR(s1, s2)`	pozicija prvog pojavljivanja niske s2 u niski s1
`REPEAT(s, n)`	niska s ponovljena n puta
`REPLACE(s, x, y)`	zamena svih pojavljivanja niske x u niski s niskom y
`LTRIM`, `RTRIM`, `TRIM`	uklanjanje razmaka sa početka, kraja ili oba kraja niske
`UCASE`, `LCASE`	konverzija u velika, odnosno mala slova

Za operacije čiji su argumenti ili rezultat pozicije ili dužine niski postoje verzije koje rade sa 8-bitnim, 16-bitnim i 32-bitnim kodnim jedinicama. Baza Stud2020 napravljena je sa kodiranjem UTF-8, u kome ASCII znakovi zauzimaju po jedan bajt, a ćirilični po dva. Zato niska `'abcdАБВГ'` ima 8 znakova, ali 12 bajtova: funkcije `LENGTH` i `LENGTHB` vraćaju 12, a `LENGTH2` i `LENGTH4` vraćaju 8.

Primer 2.20 — Dužina niske u različitim jedinicama

```
values
    -- konkretna baza je napravljena kao UTF-8
    -- zato dužina nije konstantna po karakteru
    length('abcdАБВГ'),
    lengthb('abcdАБВГ'),
    length2('abcdАБВГ'),
    length4('abcdАБВГ');
```

Rezultat:

1
12
12
8
8

Primer 2.21 — SUBSTR i SUBSTR2

```
values
    -- ovo radi ispravno zato sto su svi obuhvaćeni znaci ASCII
    substr('abcdАБВГ',2,3),
    -- ovo ne radi, zato što se uzima "pola" ćiriličnog slova A
    substr('abcdАБВГ',2,4),
    -- ovo ne radi, zato što se uzimaju dva bajta jednog karaktera
    substr('abcdАБВГ',2,5),
    -- svi naredni izrazi rade ispravno
    substr2('abcdАБВГ',2,3),
    substr2('abcdАБВГ',2,4),
    substr2('abcdАБВГ',2,5);
```

Rezultat:

1
bcd
bcd
bcdA
bcd
bcdA
bcdАБ

Objašnjenje: Funkcija `SUBSTR` računa pozicije i dužine u bajtovima. Prvi izraz radi ispravno jer obuhvata samo ASCII znakove. U drugom izrazu 4 bajta od druge pozicije obuhvataju `b`, `c`, `d` i samo prvi bajt znaka `A`, pa rezultat sadrži neispravan (nepotpun) znak. U trećem

izrazu 5 bajtova obuhvata `bcdA`, a ne 5 znakova kako bi se očekivalo. Funkcija `SUBSTR2` računa u 16-bitnim kodnim jedinicama, koje za ove znakove odgovaraju znakovima, pa svi njeni rezultati odgovaraju očekivanim (3, 4 i 5 znakova).

Primer 2.22 — Funkcija LEFT sa jedinicom dužine

```
values
    -- prva tri izraza ne rade ispravno
    left('abcdАБВГ',5),
    left('abcdАБВГ',6),
    left('abcdАБВГ',7),
    -- poslednja tri izraza rade ispravno
    left('abcdАБВГ',5,codeunits16),
    left('abcdАБВГ',6,codeunits16),
    left('abcdАБВГ',7,codeunits16);
```

Rezultat:

1
abcd
abcdA
abcdA
abcdA
abcdAБ
abcdAБВ

Objašnjenje: Za neke druge operacije ne postoje posebne verzije, već opcioni dodatni argument koji određuje jedinicu: `CODEUNITS16`, `CODEUNITS32` ili `OCTETS`. Ako se ne navede, podrazumevaju se bajtovi. Zato prva tri izraza u ovom primeru vraćaju 5, 6 i 7 bajtova (pa neki od rezultata sadrže nepotpun znak), a poslednja tri 5, 6 i 7 znakova.

Primer 2.23 — Zamena podniske

```
values replace( 'abcdАБВГ', 'dA', 'ШЂ' );
```

Rezultat:

1
abcШЂБВГ

Objašnjenje: Operacije koje ne koriste pozicije i dužine niski rade ispravno bez obzira na sadržaj niske.

Primer 2.24 — Konverzija u velika i mala slova

```
values ucase('abcABCшђжШЂЖšđžŠĐŽ'), lcase('abcABCшђжШЂЖšđžŠĐŽ');
```

Rezultat:

1
ABCABCШЂЖШЂЖŠĐŽŠĐŽ
abcabcшђжшђжšđžšđž

6. Predikati IN, BETWEEN i LIKE

U SQL-u se **predikatima** nazivaju izrazi koji izračunavaju logičku vrednost. Pored operatora poređenja i već predstavljenih predikata `IS NULL` i `IS NOT NULL`, na raspolaganju su i sledeći predikati:

- `<izraz> [NOT] IN ( <lista> )` — proverava da li vrednost izraza (ne) pripada datoj listi (skupu) vrednosti;
- `<izraz> [NOT] BETWEEN <izraz1> AND <izraz2>` — proverava da li vrednost izraza (ne) pripada zatvorenom intervalu; izraz `x BETWEEN a AND b` ekvivalentan je izrazu `a <= x AND x <= b`;
- `<izraz> [NOT] LIKE <maska> [ESCAPE <znak>]` — proverava da li niska odgovara šablonu opisanom maskom; maska može da sadrži obične znakove i specijalne simbole `%` (proizvoljna podniska, uključujući i praznu) i `_` (tačno jedan proizvoljan znak). Opcija `ESCAPE` određuje znak kojim se specijalni simboli mogu navesti kao obični.

Primer 2.25 — Predikat NOT IN

```
select * from dosije
where ime not in ('Milan','Ana');
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150069	101	Nemanja	Nicic	m	Sabac	-2	2015-07-06
20150090	103	Andrijana	Beara	z	Sabac	-2	2015-07-06

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150320	103	Milos	Milenkovic	m	Krusevac	2	2015-07-06
20150185	101	Valentina	Stojanov	z	Beograd (Savski venac)	2	2015-07-06
20150230	103	Aida	Babic	z	Cacak	2	2015-07-06
…	…	…	…	…	…	…	…

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 3375 redova × 9 kolona; prikaz je skraćen.

Primer 2.26 — Predikat BETWEEN

```
select * from dosije
where indeks between 20170100 and 20170105;
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20170100	101	Aleksandar	Laketic	m	Uzice	-5	2017-07-05
20170101	101	Gavrilo	Klima	m	Beograd (Savski venac)	2	2017-07-03
20170102	103	Dusan	Ilic	m	Cacak	-5	2017-07-04
20170103	101	David	Raznatovic	m	Valjevo	-1	2017-10-24
20170104	103	Tamara	Rasic	z	Beograd (Savski venac)	2	2017-07-04
20170105	101	Jovan	Vrecica	m	Beograd (Savski venac)	2	2017-07-03

Napomena: prikazano je prvih 8 od 9 kolona.

Primer 2.27 — Predikat LIKE sa simbolom %

```
select ime from dosije
where ime like 'Gor%';
```

Rezultat:

IME
Goran
Gordana
Goran

IME
Goran
Gorana
Goran
Goran
Gorica
Goran

Objašnjenje: U ovom primeru maska `'Gor%'` odgovara svim imenima koja počinju sa „Gor”. Rezultat sadrži ponavljanja, jer postoji više studenata sa istim imenom.

Primer 2.28 — Predikat LIKE sa simbolima _ i %

```
select distinct ime from dosije
where ime like '_ora%';
```

Rezultat:

IME
Goran
Gorana
Zora
Zoran
Zorana

Objašnjenje: Maska `'_ora%'` odgovara imenima čiji su drugi, treći i četvrti znak redom `o`, `r` i `a`, a prvi znak proizvoljan; ključnom rečju `DISTINCT` isključena su ponavljanja.

7. Izraz CASE

Izraz `CASE` je uslovni izraz sa više grana. Ima dva oblika. Prvi oblik podseća na naredbu `switch` programskog jezika C, ali sa izrazima:

```
CASE <izraz0>
    WHEN <izraz1> THEN <rezultat1>
    [ WHEN <izraz2> THEN <rezultat2> ]*
    [ ELSE <rezultatE> ]
END
```

Najpre se izračunava `<izraz0>`, a zatim se njegova vrednost redom poredi sa izrazima iza ključnih reči `WHEN`. Rezultat je vrednost izraza iza `THEN` koji odgovara prvom jednakom izrazu. Ako nijedan nije jednak, rezultat je vrednost izraza iza `ELSE`, a ako grana `ELSE` nije navedena, rezultat je `NULL`.

Drugi oblik predstavlja uopštenje višestrukog uslovnog izraza:

```
CASE
    WHEN <uslov1> THEN <rezultat1>
    [ WHEN <uslov2> THEN <rezultat2> ]*
    [ ELSE <rezultatE> ]
END
```

Uslovi iza ključnih reči `WHEN` izračunavaju se redom, a rezultat je vrednost izraza iza `THEN` koji odgovara **prvom tačnom** uslovu. Ako nijedan uslov nije tačan, rezultat je vrednost izraza iza `ELSE`, odnosno `NULL`. Izraz `CASE` se najčešće koristi za jednostavna diskretna preslikavanja, ali i za složenije namene, npr. za primenu različitih načina izračunavanja u različitim slučajevima.

Primer 2.29 — Prvi oblik izraza CASE

```
-- Izlistati st.programe sa oznakom da li je program osnovni ili ne
select
    id,
    case idnivoa
        when 1 then 'Osnovne'
        else 'Poslediplomske'
    end,
    naziv
from studijskiprogram;
```

Rezultat:

ID	2	NAZIV
101	Osnovne	Matematika
103	Osnovne	Informatika
104	Osnovne	Astronomija i astrofizika

ID	2	NAZIV
201	Poslediplomske	Matematika
202	Poslediplomske	Informatika
203	Poslediplomske	Informatika
204	Poslediplomske	Astronomija i astrofizika
301	Poslediplomske	Matematika
302	Poslediplomske	Informatika
303	Poslediplomske	Astronomija i astrofizika

Objašnjenje: Ovaj upit za svaki studijski program, prema nivou studija, izračunava oznaku „Osnovne” ili „Poslediplomske”.

Primer 2.30 — Drugi oblik izraza CASE

```
select
    indeks,
    days_between(
        case when datdiplomiranja is null then current_date
            else datdiplomiranja
        end, datupisa )
from dosije;
```

Rezultat:

INDEKS	2
20150069	1529
20150090	1523
20150320	3018
20150185	3018
20150230	3018
...	...
20180100	1926
20190011	1562
20190254	1561

INDEKS	2
20200213	1182
20151029	2909

Ukupno: 3496 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit za svakog studenta računa dužinu studiranja u danima: za diplomirane studente od datuma upisa do datuma diplomiranja, a za ostale od datuma upisa do tekućeg datuma.

Primer 2.31 — Ista vrednost pomoću funkcije COALESCE

```
select
    indeks,
    days_between( coalesce(datdiplomiranja,current_date), datupisa )
from dosije;
```

Rezultat:

INDEKS	2
20150069	1529
20150090	1523
20150320	3018
20150185	3018
20150230	3018
…	…
20180100	1926
20190011	1562
20190254	1561
20200213	1182
20151029	2909

Ukupno: 3496 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: Isto se može izračunati i funkcijom `COALESCE`, koja je opisana u odeljku 9.

8. Brojač redova

U nekim slučajevima potrebno je izračunati redni broj reda u odnosu na neko uređenje. Jedan način je urediti rezultat i brojati redove (ili izdvojiti deo rezultata klauzulama `LIMIT` i `OFFSET`), ali to često nije dovoljno. Zbog toga postoji funkcija `ROWNUMBER` (sinonim `ROW_NUMBER`), koja izračunava redni broj reda prema izabranom kriterijumu:

```
{ ROWNUMBER | ROW_NUMBER } () OVER (
    [PARTITION BY <lista kolona>]
    [ORDER BY <lista kolona ili izraza>]
)
```

Primer 2.32 — Redni broj semestra

```
select
    semestar.*,
    rownumber() over (order by semestar, skgodina) as rbr
from semestar;
-- order by skgodina, semestar
-- offset 5 rows
```

Rezultat:

SKGODINA	SEMESTAR	RBR
2015	1	1
2016	1	2
2017	1	3
2018	1	4
2019	1	5
2020	1	6
2015	2	7
2016	2	8
2017	2	9
2018	2	10
2019	2	11
2020	2	12

Objašnjenje: Klauzula `ORDER BY` određuje uređenje prema kome se redovi broje i ima isti oblik kao pri uređivanju rezultata upita. Ona ne uređuje sam rezultat upita: redosled izdvojenih redova i dalje određuje (samo) klauzula `ORDER BY` upita. U ovom primeru svakom semestru dodeljuje se redni broj prema uređenju po semestru, a zatim po školskoj godini.

Primer 2.33 — Brojanje po grupama (PARTITION BY)

```
select
    indeks, ime, prezime, idprograma,
    rownumber() over( order by datdiplomiranja ) as rbr1,
    rownumber() over( partition by idprograma order by datdiplomiranja ) as
rbr2
from dosije
where datdiplomiranja is not null
order by indeks;
--order by datdiplomiranja
--order by idprograma, datdiplomiranja
```

Rezultat:

INDEKS	IME	PREZIME	IDPROGRAMA	RBR1	RBR2
20150006	Nikola	Milojkovic	101	184	13
20150011	Ana	Pantovic	103	177	10
20150015	Milica	Stefanovic	101	328	47
20150016	Andjela	Nakaradic	101	327	46
20150018	Miodrag	Kuburovic	103	236	25
…	…	…	…	…	…

Ukupno: 405 redova × 6 kolona; prikaz je skraćen.

Objašnjenje: Opciona klauzula `PARTITION BY` određuje da se redovi ne broje na celom skupu, već posebno za svaku grupu redova koji imaju jednake vrednosti navedenih kolona. U ovom upitu kolona `RBR1` sadrži redni broj studenta među svim diplomiranim studentima prema datumu diplomiranja, a kolona `RBR2` njegov redni broj među diplomiranim studentima istog studijskog programa: studenti se najpre podele u grupe prema studijskom programu, a zatim se u svakoj grupi posebno urede i numerišu.

Primer 2.34 — Redni broj godine za svaki semestar

```
select
    semestar.*,
    rownumber() over (partition by semestar order by skgodina) as rbr
from semestar
order by skgodina, semestar
limit 5,5;
```

Rezultat:

SKGODINA	SEMESTAR	RBR
2017	2	3
2018	1	4
2018	2	4
2019	1	5
2019	2	5

Objašnjenje: Ovaj upit za svaki semestar izračunava u kojoj je, po redu održavanja, školskoj godini bio održan. Klauzula `LIMIT 5,5` preskače prvih 5 redova i izdvaja narednih 5.

9. Funkcije COALESCE i NULLIF

U programskim jezicima i alatima nedefinisane vrednosti često nisu podržane. Tada je pri čitanju podataka iz baze podataka potrebno nedefinisanu vrednost preslikati u neku posebnu vrednost, a pri upisivanju posebnu vrednost preslikati u nedefinisanu. Tome služe funkcije `COALESCE` i `NULLIF`:

- `COALESCE(x1, ..., xn)` izračunava argumente redom i vraća prvi koji je definisan, odnosno `NULL` ako su svi nedefinisani. Koristi se za preslikavanje nedefinisanih vrednosti u definisane; retko se koristi sa više od dva argumenta.
- `NULLIF(x, y)` vraća `x` ako su vrednosti `x` i `y` različite, a `NULL` ako su jednake. Koristi se za preslikavanje neke posebne vrednosti u nedefinisanu.

Primer 2.35 — Dužina studiranja aktivnih studenata

```
-- Izračunati dužinu studiranja aktivnih studenata, od upisa do datuma
diplomiranja ili današnjeg datuma.
select
    indeks,
    days( coalesce( datdiplomiranja, current_date )) - days(datupisa)
from dosije
where idstatusa > 0
order by 2 desc;
```

Rezultat:

INDEKS	2
20150320	3018
20150185	3018
20150230	3018
20150177	3018
20150135	3018
…	…
20200112	1107
20200015	1107
20200022	1105
20200011	1104
20200239	1103

Ukupno: 2224 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit računa dužinu studiranja aktivnih studenata (onih sa pozitivnim identifikatorom statusa), od upisa do datuma diplomiranja ili, ako student nije diplomirao, do tekućeg datuma.

Na primer, ako se broj čita iz baze podataka u celobrojnu promenljivu programskog jezika koji ne poznaje nedefinisane vrednosti, nedefinisana vrednost se može predstaviti posebnom vrednošću −1. Pri čitanju bi se koristio izraz `coalesce(..., -1)`, a pri upisivanju izraz `nullif(..., -1)`.

10. Alijasi tabela

Kao što se u klauzuli `SELECT` može uvesti novo ime kolone, tako se u klauzuli `FROM` može uvesti novo ime (alijas) tabele:

```
<tabela> [AS] <ime>
```

Alijas važi u celom upitu i koristi se umesto imena tabele. To je posebno korisno kada se u upitu koristi više tabela, pa je potrebno eksplicitno navesti kojoj tabeli kolona pripada, ili kada se ista tabela koristi više puta.

Primer 2.36 — Alijas tabele

```
select d.ime, d.prezime
from dosije as d
limit 5;
```

Rezultat:

IME	PREZIME
Nemanja	Nicic
Andrijana	Beara
Milos	Milenkovic
Valentina	Stojanov
Aida	Babic

11. Spajanje tabela

Do sada opisani oblik upita omogućava izdvajanje podataka iz jedne tabele. Da bi se jednim upitom izdvojili podaci iz više tabela, te tabele je potrebno spojiti u jednu, pa tek zatim izdvojiti podatke. Formalno, u klauzuli `FROM` uvek stoji jedna tabela, ali ona može biti ili tabela baze podataka ili tabela izračunata spajanjem drugih tabela. Spajanje se semantički može posmatrati kao prvi korak pri izračunavanju rezultata upita (iako stvaran redosled operacija ne mora da prati taj koncept). Nije pogrešno reći ni da se u klauzuli `FROM` navodi više tabela.

Postoje tri vrste spajanja: **unutrašnje**, **spoljašnje** i **unakrsno**. Za svako spajanje važi sledeće:

- spajanje je izraz nad tabelama čiji je rezultat nova tabela;

- rezultat spajanja sadrži sve kolone prve i sve kolone druge tabele (tim redom, mada to retko ima značaja);
- jedan red rezultata gradi se od najviše jednog reda prve i najviše jednog reda druge tabele;
- ako jedna tabela ima *N*, a druga *M* redova, rezultat ima najviše *M* × *N* redova, osim u slučaju spoljašnjeg spajanja kada je jedna od tabela prazna.

Spajanje više tabela izvodi se kao niz spajanja po dve tabele. Spajanje je levo asocijativno, a redosled se može promeniti zagradama.

11.1. Unutrašnje spajanje

Unutrašnje spajanje spaja redove jedne tabele sa odgovarajućim redovima druge tabele prema datom uslovu spajanja. U rezultatu se pojavljuju samo oni redovi svake od tabela za koje u drugoj tabeli postoji bar jedan odgovarajući red; ako za neki red ne postoji odgovarajući red druge tabele, podaci iz tog reda se ne pojavljuju u rezultatu. Naziv *unutrašnje* odgovara konceptu preseka skupova. Sintaksa ima dva oblika:

```
<tab1> [INNER] JOIN <tab2> ON <uslov>
<tab1> [INNER] JOIN <tab2> USING ( <lista kolona> )
```

U prvom obliku spajaju se parovi redova koji zadovoljavaju uslov iza `ON`. U drugom obliku spajaju se parovi redova koji imaju jednake vrednosti svih navedenih kolona, tj. važi:

```
... USING (kol1, ..., koln)
==
... ON tab1.kol1 = tab2.kol1 AND ... AND tab1.koln = tab2.koln
```

Ako unutrašnje spajanje podrazumeva i odabir redova neke od tabela, taj uslov može da se navede bilo u uslovu spajanja, bilo u klauzuli `WHERE`; rezultat je isti. Uslov je dobro navesti tamo gde će biti razumljiviji pri održavanju upita.

Primer 2.37 — Studenti i nazivi studijskih programa

```
select ime, prezime, sp.naziv
from dosije d
    inner join studijskiprogram sp
        on d.idprograma = sp.id;
```

Rezultat:

IME	PREZIME	NAZIV
Svetozar	Micic	Informatika

IME	PREZIME	NAZIV
Nemanja	Nicic	Matematika
Andrijana	Beara	Informatika
Milos	Milenkovic	Informatika
Valentina	Stojanov	Matematika
…	…	…
Andrea	Novakovic	Informatika
Tara	Bozinovic	Informatika
Ana	Ilic	Matematika
Tamara	Zejak	Matematika
Milos	Djokic	Matematika

Ukupno: 3496 redova × 3 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit izdvaja imena i prezimena studenata sa nazivom studijskog programa na koji su upisani. Imena se nalaze u tabeli `Dosije`, a nazivi programa u tabeli `StudijskiProgram`, pa se podaci o svakom studentu spajaju sa podacima o programu na koji je upisan.

Primer 2.38 — Sadržaj tabele StudijskiProgram

```
select * from studijskiprogram;
```

Rezultat:

ID	OZNAKA	NAZIV	IDNIVOA	OBIMESPB	ZVANJE	OPIS
101	M	Matematika	1	240	Diplomirani matematicar	Studijski program Matematika, osnovne akademsk…
103	I	Informatika	1	240	Diplomirani informaticar	Studije na studijskom programu osnovnih akadem…
104	A	Astronomija i astrofizika	1	240	Diplomirani astronom	Studijski program Astronomija i astrofizika, o…
201	M2	Matematika	2	60	Master matematicar	Studijski program Matematika, master akademske…

ID	OZNAKA	NAZIV	IDNIVOA	OBIMESPB	ZVANJE	OPIS
202	I2.2	Informatika	2	120	Master informaticar	Studije na studijskom programu master akademsk...
203	I2.1	Informatika	2	60	Master informaticar	Studije na studijskom programu master akademsk...

Napomena: prikazano je prvih 6 od 10 redova.

Objašnjenje: U ovom primeru spojićemo, bez restrikcije i projekcije, sve podatke iz tabela `StudijskiProgram` i `NivoKvalifikacije`, tako da se svaki studijski program poveže sa odgovarajućim nivoom kvalifikacije. Najpre su prikazani sadržaji obe tabele.

Primer 2.39 — Sadržaj tabele NivoKvalifikacije

```
select * from nivokvalifikacije;
```

Rezultat:

ID	NAZIV
1	Osnovne akademske studije
3	Doktorske akademske studije
2	Master akademske studije

Primer 2.40 — Unutrašnje spajanje programa i nivoa

```
-- unutrašnje spajanje programa i nivoa ...
select *
from studijskiprogram sp
    join nivokvalifikacije nk
        on sp.idnivoa = nk.id;
```

Rezultat:

ID	OZNAKA	NAZIV	IDNIVOA	OBIMESPB	ZVANJE	OPIS	ID
101	M	Matematika	1	240	Diplomirani matematicar	Studijski program Matematika, osnovne akademsk...	1
103	I	Informatika	1	240	Diplomirani informaticar	Studije na studijskom programu osnovnih akadem...	1

ID	OZNAKA	NAZIV	IDNIVOA	OBIMESPB	ZVANJE	OPIS	ID
104	A	Astronomija i astrofizika	1	240	Diplomirani astronom	Studijski program Astronomija i astrofizika, o...	1
201	M2	Matematika	2	60	Master matematicar	Studijski program Matematika, master akademske...	2
202	I2.2	Informatika	2	120	Master informaticar	Studije na studijskom programu master akademsk...	2
203	I2.1	Informatika	2	60	Master informaticar	Studije na studijskom programu master akademsk...	2

Napomena: prikazano je prvih 8 od 9 kolona; prikazano je prvih 6 od 10 redova.

Rezultat sadrži sve kolone obe tabele, pa i dve kolone sa imenom `ID` (prva pripada tabeli `StudijskiProgram`, a druga tabeli `NivoKvalifikacije`).

Sledi složeniji primer. Za studenta sa brojem indeksa 20180054 potrebno je izdvojiti indeks, ime i prezime, kao i nazive svih položenih obaveznih predmeta sa brojem osvojenih poena i ocenom. Podaci o polaganjima su u tabeli `Ispit`, ime i prezime u tabeli `Dosije`, podaci o obaveznim predmetima programa u tabeli `PredmetPrograma`, a nazivi predmeta u tabeli `Predmet`. Sve četiri tabele potrebno je spojiti na odgovarajući način, postepeno, dodajući jednu po jednu tabelu.

Važno: svaki uslov spajanja mora biti potpuno i tačno definisan. Ako neki uslov spajanja nije dovoljno precizan, dobija se neispravan rezultat, koji u nekim slučajevima može da bude ogroman.

Primer 2.41 — Položeni obavezni predmeti jednog studenta

```
-- za studenta 20180054 izdvojiti podatke o položenim obaveznim predmetima
select
    d.indeks, d.ime, d.prezime,
    p.naziv,
    i.poeni, i.ocena
from dosije d
    join predmetprograma pp
        using (idprograma)
    join predmet p
        on p.id = pp.idpredmeta
    join ispit i
        on d.indeks = i.indeks
        and p.id = i.idpredmeta
        and i.status = 'o'
        and i.ocena > 5
where d.indeks = 20180054
  and pp.vrsta = 'obavezan';
```

Rezultat:

INDEKS	IME	PREZIME	NAZIV	POENI	OCENA
20180054	Milos	Djurisic	Programiranje 1	99	10
20180054	Milos	Djurisic	Diskretne strukture 1	100	10
20180054	Milos	Djurisic	Programiranje 2	98	10
20180054	Milos	Djurisic	Diskretne strukture 2	93	10
20180054	Milos	Djurisic	Geometrija	88	9
…	…	…	…	…	…

Ukupno: 17 redova × 6 kolona; prikaz je skraćen.

11.2. Spoljašnje spajanje

Spoljašnje spajanje je proširenje unutrašnjeg spajanja koje obezbeđuje da se svaki red tabele pojavi bar jednom u rezultatu. Ako za neki red jedne tabele ne postoji nijedan odgovarajući red druge tabele, pravi se red rezultata koji sadrži taj red i *prazan red* druge tabele, tj. sve vrednosti koje odgovaraju drugoj tabeli su nedefinisane. Dok unutrašnje spajanje odgovara konceptu preseka, spoljašnje spajanje odgovara konceptu unije skupova. Postoje tri vrste spoljašnjeg spajanja:

- **levo** — samo redovi prve (leve) tabele mogu se spajati sa praznim redovima;
- **desno** — samo redovi druge (desne) tabele mogu se spajati sa praznim redovima;
- **puno** — redovi obe tabele mogu se spajati sa praznim redovima.

```
<tab1> {LEFT | RIGHT | FULL} [OUTER] JOIN <tab2> ON <uslov>
<tab1> {LEFT | RIGHT | FULL} [OUTER] JOIN <tab2> USING ( <lista kolona> )
```

Primer 2.42 — Položeni i nepoloženi obavezni predmeti

```
-- za studenta 20180054 izdvojiti podatke o položenim i nepoloženim obaveznim
predmetima
select
    d.indeks, d.ime, d.prezime,
    p.naziv,
    i.poeni, i.ocena
from dosije d
    join predmetprograma pp
        using (idprograma)
    join predmet p
        on p.id = pp.idpredmeta
    left join ispit i
        on d.indeks = i.indeks
        and p.id = i.idpredmeta
        and i.status = 'o'
        and i.ocena > 5
where d.indeks = 20180054
  and pp.vrsta = 'obavezan';
```

Rezultat:

INDEKS	IME	PREZIME	NAZIV	POENI	OCENA
20180054	Milos	Djurisic	Programiranje 1	99	10
20180054	Milos	Djurisic	Diskretne strukture 1	100	10
20180054	Milos	Djurisic	Programiranje 2	98	10
20180054	Milos	Djurisic	Diskretne strukture 2	93	10
20180054	Milos	Djurisic	Geometrija	88	9
…	…	…	…	…	…

Ukupno: 29 redova × 6 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit je sličan prethodnom, ali izdvaja i nepoložene obavezne predmete, uključujući i one koji nisu ni prijavljeni. Najpre se unutrašnjim spajanjem tabela `Dosije`, `PredmetPrograma` i `Predmet` dobija spisak svih obaveznih predmeta studenta. Zatim se taj međurezultat levim spoljašnjim spajanjem spaja sa tabelom `Ispit`, tako da se svaki obavezan predmet poveže sa podatkom o polaganju, ako je položen, odnosno sa praznim

redom, ako nije. Uslovi o statusu i oceni namerno su navedeni u uslovu spajanja, a ne u klauzuli `WHERE`: u klauzuli `WHERE` bi odbacili prazne redove i spoljašnje spajanje bi se ponašalo kao unutrašnje.

11.3. „Spajanje" razlike

Primer 2.43 — Nepoloženi obavezni predmeti jednog studenta

```
-- za studenta 20180054 izdvojiti podatke o nepoloženim obaveznim predmetima
select
    d.indeks, d.ime, d.prezime,
    p.naziv,
    i.poeni, i.ocena
from dosije d
    join predmetprograma pp
        using (idprograma)
    join predmet p
        on p.id = pp.idpredmeta
    left join ispit i
        on d.indeks = i.indeks
        and p.id = i.idpredmeta
        and i.status = 'o'
        and i.ocena > 5
where d.indeks = 20180054
  and pp.vrsta = 'obavezan'
  and i.indeks is null;
```

Rezultat:

INDEKS	IME	PREZIME	NAZIV	POENI	OCENA
20180054	Milos	Djurisic	Relacione baze podataka	NULL	NULL
20180054	Milos	Djurisic	Prevodjenje programskih jezika	NULL	NULL
20180054	Milos	Djurisic	Vestacka inteligencija	NULL	NULL
20180054	Milos	Djurisic	Uvod u numericku matematiku	NULL	NULL
20180054	Milos	Djurisic	Programske paradigme	NULL	NULL
…	…	…	…	…	…

Ukupno: 12 redova × 6 kolona; prikaz je skraćen.

Objašnjenje: Unutrašnje spajanje konceptualno odgovara preseku, a spoljašnje uniji skupova. Izdvajanje razlike skupova ponekad se naziva **„spajanje" razlike**, iako nije reč o spajanju, već o izdvajanju onih redova jedne tabele za koje ne postoje odgovarajući redovi druge tabele. To se postiže spoljašnjim spajanjem uz restrikciju kojom se zahteva da

ključne kolone druge tabele budu nedefinisane (tj. da je red spojen sa praznim redom). Drugi način je upotreba podupita sa predikatom `NOT EXISTS`, koji je obrađen na narednom času.

11.4. Unakrsno spajanje

Unakrsno spajanje odgovara Dekartovom proizvodu tabela: svaki red jedne tabele spaja se sa svakim redom druge tabele. Ono je ekvivalentno unutrašnjem spajanju čiji je uslov uvek tačan. Ako jedna tabela ima *N*, a druga *M* redova, rezultat ima tačno *M* × *N* redova.

```
<tab1> CROSS JOIN <tab2>
```

Primer 2.44 — Unakrsno spajanje dve konstantne tabele

```
select *
from (values 1, 2, 3)
        cross join (values 'a', 'b');
```

Rezultat:

1	2
1	a
2	a
3	a
1	b
2	b
3	b

Objašnjenje: U ovom primeru spajaju se dve konstantne tabele, sa 3 i 2 reda, pa rezultat ima 6 redova.

11.5. Implicitno spajanje

Starija sintaksa SQL-a nije imala operatore spajanja, već se spajanje izvodilo implicitno, pomoću klauzula `FROM` i `WHERE`. Klauzula `FROM` može da sadrži više tabela razdvojenih zapetama, što je ekvivalentno njihovom unakrsnom spajanju. Ako se uslov restrikcije u klauzuli `WHERE` (ili njegov deo) posmatra kao uslov spajanja, kombinacija tog uslova i unakrsnog spajanja daje unutrašnje spajanje sa tim uslovom. Spoljašnje spajanje ne može se na ovaj način zapisati neposredno, mada se može izraziti kao unija unutrašnjeg spajanja i odgovarajućih redova bez para.

Važno: ne preporučuje se upotreba implicitne sintakse spajanja. Prvo, u složenim upitima teško se uočava šta je uslov kog spajanja, a šta uslov restrikcije, što otežava pisanje i održavanje upita. Drugo, kao posledica toga mnogo lakše se dešava da uslov spajanja izostane ili bude nepotpun, što dovodi do neispravnog rezultata, koji se često manifestuje „eksplozijom" broja redova i nepotrebnim opterećenjem sistema.

Primer 2.45 — Implicitno spajanje programa i nivoa

```
-- unutrašnje spajanje programa i nivoa ...
select *
from studijskiprogram sp,
     nivokvalifikacije nk
where idnivoa = nk.id;
```

Rezultat:

ID	OZNAKA	NAZIV	IDNIVOA	OBIMESPB	ZVANJE	OPIS	ID
101	M	Matematika	1	240	Diplomirani matematicar	Studijski program Matematika, osnovne akademsk...	1
103	I	Informatika	1	240	Diplomirani informaticar	Studije na studijskom programu osnovnih akadem...	1
104	A	Astronomija i astrofizika	1	240	Diplomirani astronom	Studijski program Astronomija i astrofizika, o...	1
201	M2	Matematika	2	60	Master matematicar	Studijski program Matematika, master akademske...	2
202	I2.2	Informatika	2	120	Master informaticar	Studije na studijskom programu master akademsk...	2
203	I2.1	Informatika	2	60	Master informaticar	Studije na studijskom programu master akademsk...	2

Napomena: prikazano je prvih 8 od 9 kolona; prikazano je prvih 6 od 10 redova.

Primer 2.46 — Isto spajanje sa operatorom JOIN

```
-- unutrašnje spajanje programa i nivoa ...
select *
from studijskiprogram sp
        join nivokvalifikacije nk
            on idnivoa = nk.id;
```

Rezultat:

ID	OZNAKA	NAZIV	IDNIVOA	OBIMESPB	ZVANJE	OPIS	ID
101	M	Matematika	1	240	Diplomirani matematicar	Studijski program Matematika, osnovne akademsk...	1
103	I	Informatika	1	240	Diplomirani informaticar	Studije na studijskom programu osnovnih akadem...	1
104	A	Astronomija i astrofizika	1	240	Diplomirani astronom	Studijski program Astronomija i astrofizika, o...	1
201	M2	Matematika	2	60	Master matematicar	Studijski program Matematika, master akademske...	2
202	I2.2	Informatika	2	120	Master informaticar	Studije na studijskom programu master akademsk...	2
203	I2.1	Informatika	2	60	Master informaticar	Studije na studijskom programu master akademsk...	2

Napomena: prikazano je prvih 8 od 9 kolona; prikazano je prvih 6 od 10 redova.

Objašnjenje: Odgovarajući upit sa eksplicitnom sintaksom unutrašnjeg spajanja daje isti rezultat.

12. Podupiti

Podupit je svaki upitni izraz koji se koristi kao deo većeg upita. Prema složenosti rezultata postoje tri vrste podupita:

- **tabelarni podupit** — najopštiji; izračunava tabelu;
- **vektorski podupit** — poseban slučaj tabelarnog; izračunava tačno jednu kolonu;
- **skalarni podupit** — poseban slučaj vektorskog; izračunava tačno jednu vrednost (tabelu sa jednom kolonom i najviše jednim redom).

12.1. Skalarni podupiti

Skalarni podupiti mogu se koristiti na svim mestima na kojima se koriste skalarne vrednosti, a navode se između zagrada. Ako podupit izračuna više od jednog reda, dolazi do greške; ako ne izračuna nijedan red, njegova vrednost je `NULL`.

Primer 2.47 — Studenti sa istim imenom kao zadati student

```
select *
from dosije
where ime = (select ime from dosije where indeks=20170121);
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150320	103	Milos	Milenkovic	m	Krusevac	2	2015-07-06
20150357	101	Milos	Simic	m	Beograd (Savski venac)	-2	2015-07-06
20150204	101	Milos	Radenovic	m	Loznica	1	2015-07-06
20150034	101	Milos	Mijalkovic	m	Beograd (Savski venac)	-5	2015-07-07
20151101	201	Milos	Mladenovic	m	Uzice	-2	2015-10-22
…	…	…	…	…	…	…	…

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 75 redova × 9 kolona; prikaz je skraćen.

Objašnjenje: U ovom primeru izdvajaju se svi studenti koji se zovu isto kao student sa indeksom 20170121. Podupitom se izračunava ime tog studenta, a zatim se u glavnom upitu izdvajaju svi studenti sa tim imenom.

12.2. Vektorski podupiti

Vektorski podupiti se uobičajeno koriste sa operatorima `IN`, `ANY`, `SOME` i `ALL`:

- `<izraz> IN ( <podupit> )` — tačno ako vrednost izraza pripada skupu vrednosti podupita; negacija se zapisuje kao `<izraz> NOT IN ( <podupit> )`;
- `<izraz> <op> ANY ( <podupit> )` (sinonim `SOME`) — tačno ako je poređenje operatorom `<op>` tačno za **bar jednu** vrednost podupita; `= ANY` je isto što i `IN`;

- `<izraz> <op> ALL ( <podupit> )` — tačno ako je poređenje operatorom `<op>` tačno za **sve** vrednosti podupita; `<> ALL` je isto što i `NOT IN`. Ako je rezultat podupita prazan, predikat `ALL` je tačan.

Primer 2.48 — Predikat IN sa podupitom

```
select indeks, ime, prezime
from dosije
where indeks in (
    -- podupitom računamo spisak brojeva indeksa studenata koji su položili
ispit
    select indeks
    from ispit
    where idpredmeta = 2348
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

INDEKS	IME	PREZIME
20150112	Slobodan	Mijatovic
20150128	David	Jovovic
20180038	Bogdan	Lipovac
20150075	Mirko	Radojkovic
20170205	Nikola	Erakovic

Objašnjenje: Ovaj upit izdvaja indeks, ime i prezime studenata koji su položili ispit iz predmeta 2348. Podupit izračunava skup brojeva indeksa studenata koji su položili taj ispit, a glavni upit izdvaja studente čiji je broj indeksa u tom skupu.

Primer 2.49 — Predikat ANY

```
select *
from ispit i
where indeks = 20190087
  and ocena > any (
    select ocena
    from ispit i2
    where idpredmeta = i.idpredmeta
      and ocena > 5
);
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2019	jan1	20190087	2174	o	2020-01-14	88	9
2019	jan2	20190087	1578	o	2020-01-31	72	8
2019	jun1	20190087	2176	o	2020-06-24	89	9

Objašnjenje: Ovaj upit izdvaja položene ispite studenta 20190087 iz kojih ta ocena nije najmanja među položenim ocenama iz istog predmeta. Za svaki ispit studenta podupit izračunava sve dobijene prolazne ocene iz tog predmeta, a predikat `> ANY` proverava da li je ocena studenta veća od bar jedne od njih.

Primer 2.50 — Predikat ALL

```
select *
from ispit i
where indeks = 20190087
  and ocena >5
  and ocena <= all (
    select ocena
    from ispit i2
    where idpredmeta = i.idpredmeta
      and ocena >5
);
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2019	jan1	20190087	2170	o	2020-01-12	55	6
2019	jun2	20190087	1590	o	2020-07-20	52	6
2019	sep1	20190087	1580	o	2020-09-09	51	6
2019	sep3	20190087	2171	o	2020-09-29	55	6
2019	sep3	20190087	2172	o	2020-09-27	55	6

Objašnjenje: Obrnuto, ovaj upit izdvaja one ispite iz kojih student 20190087 ima najmanju prolaznu ocenu: predikat `<= ALL` proverava da li je ocena studenta manja ili jednaka od svih prolaznih ocena iz tog predmeta.

12.3. Korelisani i nekorelisani podupiti

Primer 2.51 — Nekorelisani podupit

```
-- imena studenata koji su polozili ispit iz predmeta 2348
select ime, prezime
from dosije
where indeks in (
    select indeks
    from ispit
    where idpredmeta = 2348
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

IME	PREZIME
Slobodan	Mijatovic
David	Jovovic
Bogdan	Lipovac
Mirko	Radojkovic
Nikola	Erakovic

Objašnjenje: Podupit je **korelisan** ako ne može da stoji kao samostalan upit, tj. ako se u njemu referišu kolone šireg upita. U suprotnom je **nekorelisan**. Ovaj upit sadrži nekorelisani podupit.

Primer 2.52 — Korelisani podupit

```
-- imena studenata koji su polozili ispit iz predmeta 2348
select ime, prezime
from dosije
where 2348 in (
    -- skup predmeta koje je položio student dosije.indeks
    select idpredmeta
    from ispit
    where indeks = dosije.indeks
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

IME	PREZIME
Slobodan	Mijatovic
David	Jovovic
Bogdan	Lipovac
Mirko	Radojkovic
Nikola	Erakovic

Objašnjenje: Semantika prethodnog upita može se opisati tako što se najpre izračuna nekorelisani podupit (skup brojeva indeksa studenata koji su položili ispit), a zatim se izdvoje imena studenata čiji je indeks u tom skupu. Isti rezultat može se dobiti i drugačije: umesto da se najpre izračuna skup indeksa, za svakog studenta se redom proverava da li je traženi predmet u skupu predmeta koje je on položio. Podupit tada referiše kolonu `dosije.indeks` spoljašnjeg upita, pa je korelisan.

Konceptualno, nekorelisani podupit izračunava se jednom, a njegov rezultat se upotrebljava više puta, dok se korelisani podupit izračunava ponovo za svaki red spoljašnjeg upita. Zato su nekorelisani podupiti potencijalno efikasniji. U praksi optimizator upita često uspešno svodi ovakve upite na isti oblik, pa razlike u efikasnosti nema, ali u nekim slučajevima to ne uspeva i razlike mogu biti značajne. Zato je dobro koristiti nekorelisane podupite kada god je to moguće. Obrada podupita nastavlja se na narednom času.

Čas 3 — Podupiti, kolonske funkcije, grupisanje i skupovne operacije

Primer 3.1 — Povezivanje sa bazom podataka

```
connect to stud2020 user student using abcdef;
set schema da;
```

Rezultat: veza sa bazom podataka je uspostavljena, a tekuća shema postavljena.

1. Podupiti (nastavak)

Podsetimo se: podupit je svaki upitni izraz koji se koristi kao deo većeg upita. Prema složenosti rezultata razlikuju se **tabelarni** (izračunavaju tabelu), **vektorski** (izračunavaju jednu kolonu) i **skalarni** podupiti (izračunavaju jednu vrednost). Najpre ponavljamo primere vektorskih podupita sa prethodnog časa, a zatim nastavljamo sa tabelarnim podupitima.

1.1. Vektorski podupiti — ponavljanje

Primer 3.2 — Predikat IN sa podupitom

```
select indeks, ime, prezime
from dosije
where indeks in (
    -- podupitom računamo spisak brojeva indeksa studenata koji su položili
ispit
    select indeks
    from ispit
    where idpredmeta = 2348
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

INDEKS	IME	PREZIME
20150112	Slobodan	Mijatovic
20150128	David	Jovovic
20180038	Bogdan	Lipovac
20150075	Mirko	Radojkovic
20170205	Nikola	Erakovic

Objašnjenje: Ovaj upit izdvaja indeks, ime i prezime studenata koji su položili ispit iz predmeta 2348. Podupit izračunava skup brojeva indeksa studenata koji su položili taj ispit, a glavni upit izdvaja studente čiji je indeks u tom skupu.

Primer 3.3 — Predikat ANY

```
select *
from ispit i
where indeks = 20190087
  and ocena > any (
    select ocena
    from ispit i2
    where idpredmeta = i.idpredmeta
      and ocena > 5
);
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2019	jan1	20190087	2174	o	2020-01-14	88	9
2019	jan2	20190087	1578	o	2020-01-31	72	8
2019	jun1	20190087	2176	o	2020-06-24	89	9

Objašnjenje: Ovaj upit izdvaja položene ispite studenta 20190087 iz kojih njegova ocena nije najmanja među prolaznim ocenama iz istog predmeta. Za svaki ispit podupit izračunava sve prolazne ocene iz tog predmeta, a predikat `> ANY` proverava da li je ocena studenta veća od bar jedne od njih.

Primer 3.4 — Predikat ALL

```
select *
from ispit i
where indeks = 20190087
  and ocena >5
  and ocena <= all (
    select ocena
    from ispit i2
    where idpredmeta = i.idpredmeta
      and ocena >5
);
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2019	jan1	20190087	2170	o	2020-01-12	55	6
2019	jun2	20190087	1590	o	2020-07-20	52	6
2019	sep1	20190087	1580	o	2020-09-09	51	6
2019	sep3	20190087	2171	o	2020-09-29	55	6
2019	sep3	20190087	2172	o	2020-09-27	55	6

Objašnjenje: Obrnuto, ovaj upit izdvaja ispite iz kojih student 20190087 ima najmanju prolaznu ocenu, jer predikat `<= ALL` zahteva da ocena studenta bude manja ili jednaka od svih prolaznih ocena iz tog predmeta.

1.2. Korelisani i nekorelisani podupiti — ponavljanje

Primer 3.5 — Nekorelisani podupit

```
-- imena studenata koji su polozili ispit iz predmeta 2348
select ime, prezime
from dosije
where indeks in (
    select indeks
    from ispit
    where idpredmeta = 2348
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

IME	PREZIME
Slobodan	Mijatovic
David	Jovovic
Bogdan	Lipovac
Mirko	Radojkovic
Nikola	Erakovic

Objašnjenje: Podupit je **korelisan** ako se u njemu referišu kolone šireg upita, pa ne može da stoji kao samostalan upit; u suprotnom je **nekorelisan**. Ovaj upit sadrži nekorelisani podupit.

Primer 3.6 — Korelisani podupit

```
-- imena studenata koji su polozili ispit iz predmeta 2348
select ime, prezime
from dosije
where 2348 in (
    -- skup predmeta koje je položio student dosije.indeks
    select idpredmeta
    from ispit
    where indeks = dosije.indeks
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

IME	PREZIME
Slobodan	Mijatovic
David	Jovovic
Bogdan	Lipovac
Mirko	Radojkovic
Nikola	Erakovic

Objašnjenje: Isti rezultat dobija se i korelisanim podupitom, koji za svakog studenta izračunava skup predmeta koje je položio i proverava da li je traženi predmet u tom skupu.

Konceptualno, nekorelisani podupit se izračunava jednom, a korelisani ponovo za svaki red spoljašnjeg upita, pa su nekorelisani podupiti potencijalno efikasniji. Optimizator upita često svodi oba oblika na isti plan izvršavanja, ali ne uvek; zato je dobro koristiti nekorelisane podupite kada god je to moguće.

2. Tabelarni podupiti

Tabelarni podupiti koriste se sa predikatom `EXISTS`, u klauzuli `FROM` i u naredbi `WITH`.

2.1. Predikat EXISTS

Predikat `EXISTS ( <podupit> )` tačan je ako i samo ako rezultat podupita nije prazan, a predikat `NOT EXISTS ( <podupit> )` ako i samo ako jeste prazan. Vrednost ovih predikata nikada nije *nepoznato*. Pošto se proverava samo postojanje redova, lista kolona podupita nije bitna i obično se piše `SELECT *`. Podupiti uz `EXISTS` po pravilu su korelisani, jer je skup čija se (ne)praznost proverava povezan sa redom šireg upita.

Primer 3.7 — Predikat EXISTS

```
-- studenti upisani 2019. koji su položili bar jedan predmet
select * from dosije
-- biramo one studente koji su upisani 2019 godine i za koje...
where indeks / 10000 = 2019
  -- postoji makar jedan predmet koji je položen
  and exists (
    select * from ispit
    where indeks = dosije.indeks
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20191097	201	Stefan	Mihajlovic	m	Beograd (Savski venac)	1	2019-10-11
20190178	101	Jelena	Vukovic	z	Beograd (Savski venac)	1	2019-07-01
20190371	101	Dimitrije	Vukadinovic	m	Loznica	1	2019-07-01
20190075	104	Nikolina	Miljojkovic	z	Beograd (Savski venac)	1	2019-07-01
20190211	103	Vukasin	Kalinic	m	Sabac	1	2019-07-01
…	…	…	…	…	…	…	…

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 479 redova × 9 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit izdvaja studente upisane 2019. godine koji su položili bar jedan predmet. Godina upisa dobija se iz broja indeksa celobrojnim deljenjem sa 10000 (npr. 20191097 / 10000 = 2019).

Primer 3.8 — Isti zahtev pomoću spajanja i DISTINCT

```
-- studenti upisani 2019. koji su položili bar jedan predmet
select distinct dosije.*
from dosije join ispit
    on dosije.indeks = ispit.indeks
      and status = 'o'
      and ocena > 5
where dosije.indeks / 10000 = 2019;
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20191097	201	Stefan	Mihajlovic	m	Beograd (Savski venac)	1	2019-10-11
20190178	101	Jelena	Vukovic	z	Beograd (Savski venac)	1	2019-07-01
20190371	101	Dimitrije	Vukadinovic	m	Loznica	1	2019-07-01

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20190075	104	Nikolina	Miljojkovic	z	Beograd (Savski venac)	1	2019-07-01
20190211	103	Vukasin	Kalinic	m	Sabac	1	2019-07-01
…	…	…	…	…	…	…	…

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 479 redova × 9 kolona; prikaz je skraćen.

Objašnjenje: Sličan rezultat može se dobiti i unutrašnjim spajanjem. Tada je neophodno upotrebiti `DISTINCT`, jer bi se inače student koji je položio više ispita pojavio više puta.

Iako dva prethodna upita daju isti rezultat, upit sa predikatom `EXISTS` je mnogo jasniji. Kod spajanja često nije očigledno zašto se tabele spajaju i potrebno je detaljno analizirati uslove, dok je kod predikata `EXISTS` jasno da se samo proverava postojanje odgovarajućih podataka.

Dodatni problem je izbacivanje ponavljanja. Upit sa spajanjem i `DISTINCT` ovde daje isti rezultat samo zato što izdvojene kolone (među njima i broj indeksa) jednoznačno određuju studenta. Kada bi se u oba upita izdvajale samo kolone `ime` i `prezime`, rezultati bi mogli da se razlikuju: upit sa `EXISTS` izdvojio bi po jedan red za svakog studenta, pa i ponovljene parove imena i prezimena različitih studenata, dok bi upit sa spajanjem i `DISTINCT` izdvojio samo različite parove imena i prezimena.

2.2. Predikat NOT EXISTS

Primer 3.9 — Predikat NOT EXISTS

```
-- studenti upisani 2019. koji nisu položili nijedan predmet
select * from dosije
where indeks / 10000 = 2019
  and not exists (
    select * from ispit
    where indeks = dosije.indeks
      and status = 'o'
      and ocena > 5
);
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20190330	101	Sara	Zakic	z	Smederevo	-1	2019-07-01

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20190440	103	Viseslav	Cupovic	m	Beograd (Savski venac)	1	2019-07-01
20190104	103	Milan	Joksimovic	m	Cacak	1	2019-07-01
20190160	101	Veljko	Jelic	m	Beograd (Savski venac)	1	2019-07-01
20190072	103	Aleksa	Mrvos	m	Smederevo	1	2019-07-01
…	…	…	…	…	…	…	…

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 117 redova × 9 kolona; prikaz je skraćen.

Objašnjenje: Semantički, razlika između `EXISTS` i `NOT EXISTS` je jednostavna: `NOT EXISTS` se koristi kada je potrebno proveriti da odgovarajući podatak **ne postoji**. Na primer, za studente upisane 2019. koji nisu položili nijedan ispit, prethodnom upitu dodaje se samo `NOT`.

Primer 3.10 — Nepoloženi obavezni predmeti pomoću NOT EXISTS

```
-- za studenta 20180054 izdvojiti podatke o nepoloženim obaveznim predmetima
select
    d.indeks, d.ime, d.prezime,
    p.naziv
from dosije d
    join predmetprograma pp
        using (idprograma)
    join predmet p
        on p.id = pp.idpredmeta
where d.indeks = 20180054
  and pp.vrsta = 'obavezan'
  and not exists (
      select * from ispit i
      where i.indeks = d.indeks
        and i.idpredmeta = p.id
        and i.status = 'o'
        and i.ocena > 5
  );
```

Rezultat:

INDEKS	IME	PREZIME	NAZIV
20180054	Milos	Djurisic	Relacione baze podataka
20180054	Milos	Djurisic	Prevodjenje programskih jezika
20180054	Milos	Djurisic	Vestacka inteligencija
20180054	Milos	Djurisic	Uvod u numericku matematiku
20180054	Milos	Djurisic	Programske paradigme
…	…	…	…
20180054	Milos	Djurisic	Verovatnoca
20180054	Milos	Djurisic	Istrazivanje podataka 1
20180054	Milos	Djurisic	Statistika
20180054	Milos	Djurisic	Racunarske mreze
20180054	Milos	Djurisic	Projektovanje baza podataka

Ukupno: 12 redova × 4 kolona; prikaz je skraćen.

Objašnjenje: Za razliku od `EXISTS`, izračunavanje predikata `NOT EXISTS` ne može se zameniti (unutrašnjim) spajanjem. Kako je već najavljeno, `NOT EXISTS` je ekvivalentan „spajanju razlike”. Ovaj upit izdvaja obavezne predmete studenta 20180054 za koje ne postoji podatak o polaganju.

Primer 3.11 — Nepoloženi obavezni predmeti pomoću spoljašnjeg spajanja

```
-- za studenta 20180054 izdvojiti podatke o nepoloženim obaveznim predmetima
select
    d.indeks, d.ime, d.prezime,
    p.naziv
from dosije d
    join predmetprograma pp
        using (idprograma)
    join predmet p
        on p.id = pp.idpredmeta
    left outer join ispit i
        on i.indeks = d.indeks
        and i.idpredmeta = p.id
        and i.status = 'o'
        and i.ocena > 5
where d.indeks = 20180054
  and pp.vrsta = 'obavezan'
  and i.indeks is null;
```

Rezultat:

INDEKS	IME	PREZIME	NAZIV
20180054	Milos	Djurisic	Relacione baze podataka
20180054	Milos	Djurisic	Prevodjenje programskih jezika
20180054	Milos	Djurisic	Vestacka inteligencija
20180054	Milos	Djurisic	Uvod u numericku matematiku
20180054	Milos	Djurisic	Programske paradigme
…	…	…	…
20180054	Milos	Djurisic	Verovatnoca
20180054	Milos	Djurisic	Istrazivanje podataka 1
20180054	Milos	Djurisic	Statistika
20180054	Milos	Djurisic	Racunarske mreze
20180054	Milos	Djurisic	Projektovanje baza podataka

Ukupno: 12 redova × 4 kolona; prikaz je skraćen.

Objašnjenje: Isti rezultat dobija se i „spajanjem razlike”, tj. levim spoljašnjim spajanjem sa tabelom `Ispit` i restrikcijom na redove spojene sa praznim redom.

2.3. Tabelarni podupiti u klauzuli FROM

Podupit se može navesti i umesto tabele u klauzuli `FROM`. Tada se piše između zagrada i obavezno mu se dodeljuje ime:

```
( <podupit> ) [AS] <ime>
```

Primer 3.12 — Podupiti u klauzuli FROM

```
-- izdvojiti za dva studenta 20170054, 20170059 uporedne rezultate na svim
polozenim predmetima
select
    p.naziv,
    prvi.indeks, prvi.poeni, prvi.ocena,
    drugi.indeks, drugi.poeni, drugi.ocena
from (
        select *
        from ispit i
        where i.indeks = '20170054'
          and status = 'o'
          and ocena > 5
    ) as prvi
    full outer join (
        select *
        from ispit i
        where i.indeks = '20170059'
          and status = 'o'
          and ocena > 5
    ) as drugi
    on prvi.idpredmeta = drugi.idpredmeta
    join predmet p
        on p.id = coalesce(prvi.idpredmeta,drugi.idpredmeta);
```

Rezultat:

NAZIV	INDEKS	POENI	OCENA	INDEKS	POENI	OCENA
Strani jezik	20170054	89	9	20170059	100	10
Programiranje 1	20170054	84	9	20170059	61	7
Linearna algebra	20170054	85	9	20170059	85	9
Uvod u matematicku logiku	20170054	82	9	20170059	95	10
Geometrija 1	20170054	83	9	20170059	88	9
Programiranje 2	20170054	95	10	20170059	78	8

Napomena: prikazano je prvih 6 od 21 redova.

Objašnjenje: Ovaj upit poredi rezultate dva studenta, 20170054 i 20170059, na svim položenim predmetima. Svaki od dva podupita izdvaja položene ispite jednog studenta. Punim spoljašnjim spajanjem dobijaju se i predmeti koje je položio samo jedan od njih (kolone drugog studenta su tada nedefinisane), a funkcija `COALESCE` daje identifikator predmeta iz onog podupita u kome je definisan, pa se naziv predmeta može pronaći u oba slučaja.

Ovakva primena je veoma korisna, ali upit može postati nepregledan. Zato se umesto toga obično koristi naredba `WITH` (odeljak 7).

2.4. Univerzalni kvantifikator

U SQL-u ne postoji neposredan ekvivalent univerzalnog kvantifikatora (∀). Zahtevi koji ga sadrže izražavaju se pomoću egzistencijalnog kvantifikatora, tj. predikata `EXISTS`, na osnovu ekvivalencije ∀x P(x) ⇔ ¬∃x ¬P(x). Na primer, zahtev „student je položio sve obavezne predmete” transformiše se ovako:

1. „student je položio sve obavezne predmete”
2. = „ne postoji obavezan predmet koji student nije položio”
3. = „ne postoji obavezan predmet za koji ne postoji podatak da ga je student položio”.

Primer 3.13 — Studenti koji su položili sve obavezne predmete

```
select *
from dosije d
where not exists (
    select *
    from predmetprograma pp
    where pp.idprograma = d.idprograma
      and pp.vrsta = 'obavezan'
      and not exists (
          select * from ispit i
          where i.indeks = d.indeks
            and i.idpredmeta = pp.idpredmeta
            and i.status = 'o'
            and i.ocena > 5
      )
);
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150090	103	Andrijana	Beara	z	Sabac	-2	2015-07-06

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150154	103	Zarija	Milanovic	m	Beograd (Savski venac)	-2	2015-07-06
20150300	103	Filip	Selakovic	m	Beograd (Savski venac)	1	2015-07-06
20150030	103	Mladen	Panic	m	Pancevo	-2	2015-07-06
20150011	103	Ana	Pantovic	z	Smederevo	-2	2015-07-06
...	...	...	...	...	...	...	...

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 103 redova × 9 kolona; prikaz je skraćen.

Objašnjenje: Poslednji oblik zapisuje se sa dva ugnežđena predikata `NOT EXISTS`.

3. Kolonske funkcije

Za razliku od **skalarnih funkcija**, koje se izračunavaju nad pojedinačnim vrednostima, **kolonske funkcije** (tzv. *agregatne funkcije*) izračunavaju se nad svim, ili nad izabranim, vrednostima jedne ili više kolona iz redova koje izdvaja upit. Argument kolonske funkcije može biti i izraz: tada se izraz izračunava za svaki red, a kolonska funkcija nad dobijenim vrednostima. Opšti oblik je:

```
<funkcija>( [DISTINCT] <izraz> [, <izraz>]* )
```

Ključna reč `DISTINCT` nalaže da se funkcija izračuna samo nad različitim vrednostima. Sve kolonske funkcije, osim `COUNT(*)`, zanemaruju nedefinisane vrednosti.

3.1. Funkcija COUNT

Funkcija `COUNT` broji elemente skupa, odnosno redove rezultata upita:

- `COUNT(*)` — broj redova;
- `COUNT(<izraz>)` — broj redova u kojima izraz ima definisanu vrednost;
- `COUNT(DISTINCT <izraz>)` — broj različitih definisanih vrednosti izraza.

Rezultat funkcije `COUNT` je tipa `INTEGER`; za tabele sa više od oko dve milijarde redova koristi se funkcija `COUNT_BIG`, čiji je rezultat tipa `DECIMAL(31,0)`.

Primer 3.14 — Ukupan broj prijava

```
-- Ukupan broj evidentiranih prijavljenih ispita.
select count(*)
from ispit;
```

Rezultat:

1
119143

Primer 3.15 — Broj prijava sa upisanom ocenom

```
-- Ukupan broj ispita sa upisanom ocenom, tj. sa ocenom koja nije NULL.
select count(ocena)
from ispit;
```

Rezultat:

1
52874

Primer 3.16 — Broj prijava sa ocenom najmanje 5

```
-- Ukupan broj ispita sa upisanom ocenom, tj. sa ocenom koja je bar 5.
-- Efektivno isto kao prethodni upit.
select count(*)
from ispit
where ocena >= 5;
```

Rezultat:

1
52874

Objašnjenje: Pošto je najmanja upisana ocena 5, broj redova u kojima je `ocena >= 5` jednak je broju redova sa definisanom ocenom, pa ovaj upit daje isti rezultat kao prethodni.

Primer 3.17 — Broj različitih ocena

```
-- Ukupan broj različitih ocena.
select count(distinct ocena)
from ispit;
```

Rezultat:

1
6

3.2. Funkcije MIN i MAX

Funkcije `MIN` i `MAX` izračunavaju najmanju, odnosno najveću definisanu vrednost izraza. Primenjuju se na sve tipove čije se vrednosti mogu porediti (brojeve, niske, datume). Ključna reč `DISTINCT` nema uticaja na njihov rezultat.

Primer 3.18 — Najraniji i najkasniji datum polaganja

```
select min(datpolaganja), max(datpolaganja)
from ispit;
```

Rezultat:

1	2
2016-01-15	2020-09-30

Primer 3.19 — Najmanja i najveća ocena

```
select min(ocena), max(ocena)
from ispit;
```

Rezultat:

1	2
5	10

3.3. Funkcije SUM, AVG i MEDIAN

Funkcije `SUM`, `AVG` i `MEDIAN` izračunavaju redom zbir, aritmetičku sredinu i medijanu definisanih vrednosti. Uz ključnu reč `DISTINCT` izračunavaju se samo nad različitim vrednostima, što u opštem slučaju menja rezultat (u narednom primeru medijana je

slučajno ista sa i bez `DISTINCT`).

Primer 3.20 — SUM, AVG, MEDIAN i COUNT sa i bez DISTINCT

```
select
    sum( ocena ) s,
    sum( distinct ocena ) sd,
    avg (ocena + 0.0 ) a,
    avg( distinct ocena + 0.0 ) ad,
    median( ocena ) m,
    median( distinct ocena ) md,
    count( ocena ) c,
    count( distinct ocena ) cd
from ispit
where status = 'o'
  and ocena > 5;
```

Rezultat:

S	SD	A	AD	M	MD	C	CD
263440	40	8.026079	8.0	8.0	8.0	32823	5

Objašnjenje: Funkcija `AVG` izračunava rezultat istog tipa kao argument, pa bi prosek celih brojeva bio ceo broj. Zato se ocenama dodaje `0.0`, čime izraz postaje decimalnog tipa i prosek se računa sa decimalama. U ovom primeru vidi se uticaj `DISTINCT`: zbir svih prolaznih ocena je 263440, a zbir različitih prolaznih ocena 6 + 7 + 8 + 9 + 10 = 40.

3.4. Kolonske funkcije na više skupova redova

Primer 3.21 — Broj ocena pomoću COUNT i CASE

```
-- koliko je kojih ocena dobijeno na ispitima
select
    count(ocena) as ocene,
    count(case when ocena=5 then 1 else null end) as ocena_5,
    count(case when ocena=6 then 1 else null end) as ocena_6,
    count(case when ocena=7 then 1 else null end) as ocena_7,
    count(case when ocena=8 then 1 else null end) as ocena_8,
    count(case when ocena=9 then 1 else null end) as ocena_9,
    count(case when ocena=10 then 1 else null end) as ocena_10
from ispit
where status = 'o';
```

Rezultat:

OCENE	OCENA_5	OCENA_6	OCENA_7	OCENA_8	OCENA_9	OCENA_10
51227	18404	6608	6638	6268	5908	7401

Objašnjenje: Da bi se jednim upitom prebrojali podaci prema više različitih kriterijuma, ispunjeni uslovi mogu se izrazom `CASE` preslikati u definisane, a neispunjeni u nedefinisane vrednosti. Funkcija `COUNT` tada broji samo redove u kojima je uslov ispunjen.

Primer 3.22 — Broj ocena pomoću SUM i CASE

```
-- koliko je kojih ocena dobijeno na ispitima
select
    count(ocena) as ocene,
    sum(case when ocena=5 then 1 else 0 end) as ocena_5,
    sum(case when ocena=6 then 1 else 0 end) as ocena_6,
    sum(case when ocena=7 then 1 else 0 end) as ocena_7,
    sum(case when ocena=8 then 1 else 0 end) as ocena_8,
    sum(case when ocena=9 then 1 else 0 end) as ocena_9,
    sum(case when ocena=10 then 1 else 0 end) as ocena_10
from ispit
where status = 'o';
```

Rezultat:

OCENE	OCENA_5	OCENA_6	OCENA_7	OCENA_8	OCENA_9	OCENA_10
51227	18404	6608	6638	6268	5908	7401

Objašnjenje: Alternativno, ispunjeni uslovi mogu se preslikati u 1, a neispunjeni u 0, pa se primenjuje funkcija `SUM`.

Primer 3.23 — Udeo svake ocene u procentima

```
-- zastupljenost svake od ocena
select
    avg(case when ocena=5 then 1. else 0 end) * 100 as procenat_5,
    avg(case when ocena=6 then 1. else 0 end) * 100 as procenat_6,
    avg(case when ocena=7 then 1. else 0 end) * 100 as procenat_7,
    avg(case when ocena=8 then 1. else 0 end) * 100 as procenat_8,
    avg(case when ocena=9 then 1. else 0 end) * 100 as procenat_9,
    avg(case when ocena=10 then 1. else 0 end) * 100 as procenat_10
from ispit
where status = 'o';
```

Rezultat:

PROCENAT_5	PROCENAT_6	PROCENAT_7	PROCENAT_8	PROCENAT_9	PROCENAT_10
35.926367	12.899448	12.95801	12.235735	11.532981	14.447459

Objašnjenje: Na sličan način se funkcijom `AVG` izračunava udeo nekog slučaja u celom skupu: prosek vrednosti 1 i 0 jednak je udelu jedinica. Konstanta `1.` je decimalna, pa je i prosek decimalan; množenjem sa 100 dobija se procenat.

3.5. Druge kolonske funkcije

Primer 3.24 — Funkcija LISTAGG

```
select listagg( naziv, ', ')
from da.nivokvalifikacije;
```

Rezultat:

1
Osnovne akademske studije, Doktorske akademske studije, Master akademske studije

Objašnjenje: Postoji još mnogo kolonskih funkcija: statističke (`STDDEV` , `VARIANCE` , `CORRELATION` , `STDDEV_SAMP` , `VARIANCE_SAMP`), regresione funkcije i druge. Funkcija `LISTAGG(<izraz>, <separator>)` nadovezuje vrednosti izraza u jednu nisku, razdvajajući ih datim separatorom.

4. Grupisanje redova

Često je potrebno primeniti kolonsku funkciju više puta, na različitim skupovima redova. Na primer, broj studenata sa nekim imenom može se izračunati upitom:

```
select count(*) from dosije
where ime = '...'
```

Ponavljanje ovog upita za svako ime bilo bi zamorno. Zato postoji klauzula `GROUP BY` , kojom se nalaže da se redovi podele u grupe prema vrednostima navedenih izraza i da se kolonske funkcije izračunaju posebno za svaku grupu:

```
GROUP BY <izraz> [, <izraz>]*
```

U navedenom slučaju grupiše se po imenu: `GROUP BY ime` . Rezultat sadrži po jedan red za svaku grupu. Zbog toga se u klauzuli `SELECT` mogu navoditi samo izrazi koji imaju jednu vrednost za celu grupu: kolone i izrazi navedeni u klauzuli grupisanja mogu se koristiti

slobodno, a sve ostale kolone samo kao argumenti kolonskih funkcija.

Primer 3.25 — Imena i broj njihovih pojavljivanja

```
-- imena i broj ponavljanja
select ime, count(*) as n
from dosije
group by ime
order by n desc;
```

Rezultat:

IME	N
Nikola	155
Milica	99
Jelena	90
Aleksandar	81
Marija	81
…	…
Zeljana	1
Zlata	1
Zona	1
Zorka	1
Zudana	1

Ukupno: 450 redova × 2 kolona; prikaz je skraćen.

Primer 3.26 — Broj polaganja i prosečna ocena po ispitnim rokovima

```
select skgodina, oznakaroka, avg(ocena+0.0) ocena, count(*) n
from ispit
where status = 'o'
  and skgodina between 2016 and 2017
group by skgodina, oznakaroka;
```

Rezultat:

SKGODINA	OZNAKAROKA	OCENA	N
2016	jan1	7.575136	1657
2017	jan1	7.468765	2129
2016	jan2	6.810833	1200
2017	jan2	6.787234	1598
2016	janps	8.100000	10
…	…	…	…
2016	kom	8.750000	12
2016	sep1	6.514360	1149
2017	sep1	6.510858	1842
2016	sep2	6.803430	758
2017	sep2	6.572222	1080

Ukupno: 15 redova × 4 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit izračunava broj polaganja i prosečnu ocenu u svakom ispitnom roku školskih godina 2016 i 2017. Ispitni rok je određen parom kolona `skgodina` i `oznakaroka`, pa se grupiše po obe kolone.

4.1. Odabir grupa — klauzula HAVING

Pored restrikcije redova, često je potrebna i restrikcija grupa. Klauzula `WHERE` bira redove **pre** grupisanja: ako se pre grupisanja izbace svi redovi neke grupe, izbačena je i grupa. Međutim, uslov koji se odnosi na grupu u celini (npr. na vrednost kolonske funkcije) može se proveriti tek **posle** grupisanja, pa se ne može navesti u klauzuli `WHERE`. Tome služi klauzula `HAVING`.

Primer 3.27 — Klauzula HAVING

```
select ime, count(*) as n
from dosije
group by ime
having count(*) >= 50
order by n desc;
```

Rezultat:

IME	N
Nikola	155
Milica	99
Jelena	90
Aleksandar	81
Marija	81
...	...
Aleksandra	59
Jovana	57
Aleksa	53
Djordje	50
Lazar	50

Ukupno: 17 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: U klauzuli `HAVING` navode se samo uslovi koji imaju smisla na nivou grupe: kao i u klauzuli `SELECT`, slobodno se mogu koristiti samo kolone i izrazi po kojima se grupiše, a ostale kolone samo kao argumenti kolonskih funkcija. Kolonske funkcije u uslovu ne moraju biti navedene i u klauzuli `SELECT`. Ovaj upit izdvaja imena koja se pojavljuju najmanje 50 puta.

Redosled izračunavanja klauzula je: `FROM` → `WHERE` → `GROUP BY` → `HAVING` → `SELECT` → `ORDER BY`.

4.2. Particionisanje kolonskih funkcija

Sličan efekat kao grupisanje može se postići navođenjem particionisanja neposredno uz kolonsku funkciju:

```
<funkcija>(...) OVER ( PARTITION BY <lista kolona> )
```

Ova tehnika se od grupisanja razlikuje u nekoliko bitnih pogleda:

- u klauzuli `SELECT` mogu se navesti i kolone i izrazi koji nisu navedeni u uslovu particionisanja;
- upit se ne izračunava po grupama, već po redovima, pa rezultat ima isti broj redova kao odgovarajući upit bez takvih funkcija;

- različite kolonske funkcije u istom upitu mogu koristiti različite uslove particionisanja.

Primer 3.28 — COUNT sa particionisanjem

```
-- imena i broj ponavljanja
select ime, count(*) over ( partition by ime ) as n
from dosije
order by n desc;
```

Rezultat:

IME	N
Nikola	155
Nikola	155
Nikola	155
Nikola	155
Nikola	155
…	…
Zeljana	1
Zlata	1
Zona	1
Zorka	1
Zudana	1

Ukupno: 3496 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: U ovom primeru za svaki red tabele `dosije` izračunava se broj studenata sa istim imenom, pa rezultat ima onoliko redova koliko ima studenata (3496).

Primer 3.29 — Particionisanje i DISTINCT

```
-- imena i broj ponavljanja
select distinct ime, count(*) over ( partition by ime ) as n
from dosije
order by n desc;
```

Rezultat:

IME	N
Nikola	155
Milica	99
Jelena	90
Aleksandar	81
Marija	81
…	…
Zeljana	1
Zlata	1
Zona	1
Zorka	1
Zudana	1

Ukupno: 450 redova × 2 kolona; prikaz je skraćen.

Objašnjenje: Ako se ne koristi nijedna od navedenih posebnosti, a navede se `DISTINCT`, upit je ekvivalentan odgovarajućem upitu sa grupisanjem (rezultat ponovo ima 450 redova).

Primer 3.30 — COUNT, ROWNUMBER i RANK sa particionisanjem

```
select
    ime,
    prezime,
    count(*) over ( partition by ime ) as broj_imena,
    rownumber() over (partition by ime order by indeks) as indeks_rbr,
    rank() over (order by ime) as ime_rank
from dosije
order by ime, broj_imena desc;
```

Rezultat:

IME	PREZIME	BROJ_IMENA	INDEKS_RBR	IME_RANK
Aco	Krsmanovic	1	1	1
Ada	Ilic	1	1	2

IME	PREZIME	BROJ_IMENA	INDEKS_RBR	IME_RANK
Adam	Dimitrijevic	3	1	3
Adam	Rankovic	3	2	3
Adam	Kolcic	3	3	3
…	…	…	…	…

Ukupno: 3496 redova × 5 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit izdvaja podatke o studentima uređene po imenima. Za svakog studenta izračunava se koliko ukupno studenata ima isto ime, koji je njegov redni broj među studentima sa istim imenom (prema broju indeksa) i rang imena u leksikografskom uređenju. Funkcija `RANK` svim redovima sa jednakom vrednošću kriterijuma dodeljuje isti rang, a naredni različiti red dobija rang jednak broju prethodnih redova uvećanom za 1.

5. Operatori grupisanja

Pored osnovnog grupisanja postoje i napredni oblici, kojima se jednim upitom izvodi analiza za više različitih grupisanja. Nazivaju se **operatori grupisanja** ili *OLAP grupisanja*. Koriste se u obliku `GROUP BY <operator grupisanja>`, a postoje tri:

```
GROUPING SETS ( <lista skupova> )
ROLLUP ( <lista> )
CUBE ( <lista> )
```

5.1. GROUPING SETS

U operatoru `GROUPING SETS` navodi se jedan ili više skupova kolona po kojima se grupiše. Rezultat je unija rezultata upita sa pojedinačnim grupisanjima: ako je navedeno pet skupova, rezultat je unija pet upita sa odgovarajućim grupisanjima. Ako je naveden samo jedan skup, upit se ne razlikuje od upita sa klauzulom `GROUP BY` u kojoj su navedeni elementi tog skupa. Prazan skup `()` označava jednu grupu koju čine svi redovi.

U klauzuli `SELECT` mogu se navesti sve kolone i izrazi koji se pojavljuju u bilo kom od skupova. Ako kolona ne pripada skupu po kome je dobijen neki red rezultata, u tom redu ima nedefinisanu vrednost. Unutar skupova mogu se navesti i operatori `ROLLUP` i `CUBE`.

Primer 3.31 — GROUPING SETS sa jednim skupom

```
select
    skgodina,
    oznakaroka,
    count(*) brojprijava,
    avg(ocena+0.0) prosek
from ispit
group by grouping sets( (skgodina, oznakaroka) );
-- ovaj upit je isti kao da je napisano:
-- group by skgodina, oznakaroka
```

Rezultat:

SKGODINA	OZNAKAROKA	BROJPRIJAVA	PROSEK
2015	jan1	1640	7.746206
2015	jan2	1182	6.741030
2015	jun1	1848	7.127389
2015	jun2	1550	6.644717
2015	kom	88	8.409091
…	…	…	…
2019	jun2	4995	6.665320
2019	sep1	5198	6.635577
2019	sep2	5808	6.422013
2019	sep3	5739	6.521135
2019	sep4	435	7.447257

Ukupno: 38 redova × 4 kolona; prikaz je skraćen.

Primer 3.32 — GROUPING SETS sa tri skupa

```
-- sada grupisemo po tri skupa, treći je skup svih prijava
select
    skgodina,
    oznakaroka,
    count(*) brojprijava,
    avg(ocena+0.0) prosek
from ispit
group by grouping sets( (skgodina, oznakaroka), (skgodina), () );
```

Rezultat:

SKGODINA	OZNAKAROKA	BROJPRIJAVA	PROSEK
NULL	NULL	119143	6.882135
2015	NULL	8251	7.046416
2016	NULL	16325	7.007736
2017	NULL	23383	6.875428
2018	NULL	30901	6.857216
2019	NULL	40283	6.796705
2015	jan1	1640	7.746206
2015	jan2	1182	6.741030
2015	jun1	1848	7.127389
2015	jun2	1550	6.644717
2015	kom	88	8.409091
2015	sep1	1131	6.600318

Napomena: prikazano je prvih 12 od 44 redova.

Objašnjenje: U ovom primeru grupiše se po tri skupa: po ispitnom roku, po školskoj godini i po praznom skupu (svi redovi). Prvi red rezultata, sa nedefinisanim vrednostima obe kolone, sadrži podatke o svim prijavama.

Primer 3.33 — Mogući problemi sa nedefinisanim vrednostima

```
-- potencijalni problemi sa NULL
select ocena, skgodina, count(*) n
from ispit
group by grouping sets ((ocena, skgodina), (skgodina), ())
order by ocena nulls first, skgodina;
```

Rezultat:

OCENA	SKGODINA	N
NULL	2015	8251
NULL	2015	3619

OCENA	SKGODINA	N
NULL	2016	16325
NULL	2016	8440
NULL	2017	23383
...	...	...
10	2015	762
10	2016	1235
10	2017	1475
10	2018	1855
10	2019	2077

Ukupno: 41 redova × 3 kolona; prikaz je skraćen.

Objašnjenje: Nedefinisane vrednosti mogu dovesti do nejasnoća. U ovom primeru prvi red rezultata nastaje grupisanjem po drugom skupu (sve prijave iz 2015. godine), a drugi grupisanjem po prvom skupu za prijave bez upisane ocene (prijave iz 2015. godine kod kojih je ocena nedefinisana). Iz samog rezultata nije očigledno koji red predstavlja koji podatak, sve dok se ne analiziraju podaci. Sistem Db2 za razlikovanje ovih slučajeva nudi funkciju `GROUPING(<kolona>)`, koja ima vrednost 1 ako je nedefinisana vrednost kolone posledica grupisanja, a 0 inače.

5.2. ROLLUP

Operator `ROLLUP(<lista>)` izračunava grupisanja po svim **prefiksima** date liste:

```
ROLLUP( c1, c2, ..., cn-1, cn )
==
GROUPING SETS( (c1, c2, ..., cn-1, cn), (c1, c2, ..., cn-1), ..., (c1, c2),
(c1), () )
```

Operator je pogodan za hijerarhijske podatke, kao što su školska godina i ispitni rok: dobijaju se podaci po rokovima, zbirni podaci po godinama i ukupni zbir.

Primer 3.34 — ROLLUP

```
-- grupisanje po (skgodina, oznakaroka), (skgodina) i ()
select
    skgodina,
    oznakaroka,
    count(*) brojprijava,
    avg(ocena+0.0) prosek
from ispit
group by rollup( skgodina, oznakaroka );
```

Rezultat:

SKGODINA	OZNAKAROKA	BROJPRIJAVA	PROSEK
NULL	NULL	119143	6.882135
2015	NULL	8251	7.046416
2016	NULL	16325	7.007736
2017	NULL	23383	6.875428
2018	NULL	30901	6.857216
…	…	…	…
2019	jun2	4995	6.665320
2019	sep1	5198	6.635577
2019	sep2	5808	6.422013
2019	sep3	5739	6.521135
2019	sep4	435	7.447257

Ukupno: 44 redova × 4 kolona; prikaz je skraćen.

Dopuštene su i kombinacije kolona i operatora `ROLLUP`. Na primer:

```
GROUPING SETS( (c1, ROLLUP(c2, c3), c4) )
==
GROUPING SETS( (c1, c2, c3, c4), (c1, c2, c4), (c1, c4) )
```

Primer 3.35 — Kombinacija kolona i operatora ROLLUP

```
select
    skgodina,
    oznakaroka,
    status,
    count(*) brojprijava,
    avg(ocena+0.0) prosek
from ispit
group by grouping sets(( status, rollup(skgodina), oznakaroka ));
```

Rezultat:

SKGODINA	OZNAKAROKA	STATUS	BROJPRIJAVA	PROSEK
NULL	jan1	d	1	5.00
NULL	jan2	d	3	5.00
NULL	jun1	d	1	5.00
NULL	jun2	d	3	5.00
NULL	sep1	d	1	5.00
…	…	…	…	…

Ukupno: 199 redova × 5 kolona; prikaz je skraćen.

5.3. CUBE

Operator `CUBE(<lista>)` izračunava grupisanja po **svim podskupovima** date liste, pa za listu od *n* elemenata daje 2^n grupisanja:

```
CUBE( c1, c2, c3 )
==
GROUPING SETS( (c1, c2, c3), (c1, c2), (c1, c3), (c2, c3), (c1), (c2), (c3),
() )
```

Primer 3.36 — CUBE

```
-- pregled upisivanja studenata po sk.godinama, programu i polu
select
    sp.naziv,
    pol,
    year(datupisa) as godupisa,
    count(*) n
from dosije d
    join studijskiprogram sp
        on d.idprograma = sp.id
        and sp.idnivoa = 1
group by cube( sp.naziv, pol, year(datupisa))
order by 3 nulls first, 2 nulls first, 1 nulls first;
```

Rezultat:

NAZIV	POL	GODUPISA	N
NULL	NULL	NULL	2690
Astronomija i astrofizika	NULL	NULL	159
Informatika	NULL	NULL	1061
Matematika	NULL	NULL	1470
NULL	m	NULL	1504
Astronomija i astrofizika	m	NULL	94
Informatika	m	NULL	699
Matematika	m	NULL	711
NULL	z	NULL	1186
Astronomija i astrofizika	z	NULL	65
Informatika	z	NULL	362
Matematika	z	NULL	759

Napomena: prikazano je prvih 12 od 84 redova.

Objašnjenje: Ovaj upit daje pregled upisa studenata osnovnih studija po studijskim programima, polu i godini upisa, uključujući sve zbirne kombinacije.

Primer 3.37 — CUBE u naredbi WITH — prolaznost po ispitnim rokovima

```
-- prolaznost po ispitnim rokovima
with
data as (
    select
        skgodina,
        oznakaroka,
        count(*) prijavljeno,
        sum(case when status='p' then 1 else 0 end) neodrzano,
        count(*) - sum(case when status='n' then 1 else 0 end) izaslo,
        sum(case when status='o' and ocena>5 then 1 else 0 end) polozilo
    from ispit
    group by cube( skgodina, oznakaroka )
)
select
    data.*,
    (izaslo+0.0)/prijavljeno as izlaznost,
    (polozilo+0.0)/prijavljeno as prolaznost,
    (polozilo+0.0)/izaslo as prolaznost_uk
from data
order by skgodina, oznakaroka;
```

Rezultat:

SKGODINA	OZNAKAROKA	PRIJAVLJENO	NEODRZANO	IZASLO	POLOZILO	IZLAZNOST	PROLAZNOST
2015	jan1	1640	0	1186	884	0.723171	0.539024
2015	jan2	1182	0	641	409	0.542301	0.346024
2015	jun1	1848	0	1099	722	0.594697	0.390693
2015	jun2	1550	0	653	360	0.421290	0.232258
2015	kom	88	0	88	88	1.000000	1.000000
2015	sep1	1131	0	628	369	0.555261	0.326260

Napomena: prikazano je prvih 8 od 9 kolona; prikazano je prvih 6 od 54 redova.

Objašnjenje: Operatori grupisanja mogu se kombinovati i sa naredbom `WITH` (odeljak 7). Ovaj upit za svaki ispitni rok, za svaku školsku godinu, za svaku oznaku roka i ukupno izračunava broj prijava, neodržanih ispita, izlazaka i položenih ispita, a zatim izlaznost i prolaznost (u odnosu na prijavljene i u odnosu na izašle).

6. Skupovne operacije

SQL podržava osnovne skupovne operacije nad tabelama. Svaka operacija ima dva oblika, bez ponavljanja i sa ponavljanjem:

Operacija	Bez ponavljanja	Sa ponavljanjem
unija	`UNION`	`UNION ALL`
razlika	`EXCEPT`	`EXCEPT ALL`
presek	`INTERSECT`	`INTERSECT ALL`

Primer 3.38 — UNION

```
values 1,1,1,2,2,2,3,4,4,5
union
values 1,1,3,3,3,3,4;
```

Rezultat:

1
1
2
3
4
5

Objašnjenje: Tabele nad kojima se primenjuje skupovna operacija moraju imati isti broj kolona saglasnih tipova. Ako se operacija koristi bez opcije `ALL`, to je ekvivalentno tome da se iz svake tabele izdvoji skup različitih redova, a zatim primeni uobičajena matematička skupovna operacija. U ovim primerima prva tabela sadrži vrednosti 1, 1, 1, 2, 2, 2, 3, 4, 4, 5, a druga vrednosti 1, 1, 3, 3, 3, 3, 4.

Primer 3.39 — EXCEPT

```
values 1,1,1,2,2,2,3,4,4,5
except
values 1,1,3,3,3,3,4;
```

Rezultat:

1
2

1
5

Primer 3.40 — INTERSECT

```
values 1,1,1,2,2,2,3,4,4,5
intersect
values 1,1,3,3,3,3,4;
```

Rezultat:

1
1
3
4

Ako se operacija koristi sa opcijom `ALL`, semantika zavisi od operacije. Kod unije se svi redovi smatraju različitim, pa rezultat sadrži sve redove obe tabele: ako prva ima *N*, a druga *M* redova, rezultat ima *N* + *M* redova (ovde 10 + 7 = 17).

Primer 3.41 — UNION ALL

```
values 1,1,1,2,2,2,3,4,4,5
union all
values 1,1,3,3,3,3,4;
```

Rezultat:

1
1
1
3
3
3
3
4

1
1
1
1
2
2

Napomena: prikazano je prvih 12 od 17 redova.

Primer 3.42 — EXCEPT ALL

```
values 1,1,1,2,2,2,3,4,4,5
except all
values 1,1,3,3,3,3,4;
```

Rezultat:

1
1
2
2
2
4
5

Objašnjenje: Kod razlike i preseka semantika je kao da je (1) svaka tabela uređena, (2) proširena dodatnom kolonom u kojoj su redovi svake grupe jednakih redova numerisani od 1 nadalje, (3) zatim primenjena skupovna operacija nad proširenim tabelama i (4) na kraju uklonjena dodatna kolona. Posledica je da se jednaki redovi unutar jedne tabele smatraju različitim, a između dve tabele mogu da se poklope. Ako se neki red pojavljuje a puta u prvoj i b puta u drugoj tabeli, u rezultatu operacije `EXCEPT ALL` pojavljuje se $\max(a - b, 0)$ puta, a u rezultatu operacije `INTERSECT ALL` $\min(a, b)$ puta. Na primer, vrednost 1 pojavljuje se 3 puta u prvoj i 2 puta u drugoj tabeli, pa je u razlici jednom, a u preseku dva puta.

Primer 3.43 — INTERSECT ALL

```
values 1,1,1,2,2,2,3,4,4,5
intersect all
values 1,1,3,3,3,3,4;
```

Rezultat:

1
1
1
3
4

7. Naredba WITH

Naredba `WITH` služi za zapisivanje složenijih upita sa podupitima:

```
WITH <ime> [( <lista kolona> )] AS ( <upit> )
  [, <ime> [( <lista kolona> )] AS ( <upit> )]*
<upit>
```

U osnovnom obliku ekvivalentna je glavnom upitu u čijoj su klauzuli `FROM` zapisani i imenovani odgovarajući podupiti, ali je njen zapis čitljiviji i olakšava pisanje i održavanje složenih upita. Naredba `WITH` ima i značajne dodatne mogućnosti:

- u telu svakog imenovanog podupita mogu se referisati svi prethodno definisani podupiti, pa se veoma složeni upiti mogu graditi korak po korak;
- u telu podupita može se referisati i sam taj podupit, čime se omogućava pisanje rekurzivnih upita (Čas 4).

Primer 3.44 — Uporedni rezultati dva studenta pomoću WITH

```
-- izdvojiti za dva studenta 20170054, 20170059 uporedne rezultate na svim
polozenim predmetima
with
prvi as (
    select *
    from ispit i
    where i.indeks = '20170054'
      and status = 'o'
      and ocena > 5
),
drugi as (
```

```
    select *
    from ispit i
    where i.indeks = '20170059'
      and status = 'o'
      and ocena > 5
)
select
    p.naziv,
    prvi.indeks, prvi.poeni, prvi.ocena,
    drugi.indeks, drugi.poeni, drugi.ocena
from prvi
    full outer join drugi
        on prvi.idpredmeta = drugi.idpredmeta
    join predmet p
        on p.id = coalesce(prvi.idpredmeta,drugi.idpredmeta);
```

Rezultat:

NAZIV	INDEKS	POENI	OCENA	INDEKS	POENI	OCENA
Strani jezik	20170054	89	9	20170059	100	10
Programiranje 1	20170054	84	9	20170059	61	7
Linearna algebra	20170054	85	9	20170059	85	9
Uvod u matematicku logiku	20170054	82	9	20170059	95	10
Geometrija 1	20170054	83	9	20170059	88	9
Programiranje 2	20170054	95	10	20170059	78	8

Napomena: prikazano je prvih 6 od 21 redova.

Objašnjenje: Ovaj upit je isti kao upit iz odeljka 2.3, ali zapisan naredbom `WITH`.

Primer 3.45 — Rangiranje studenata upisanih na izborni predmet

```
-- rangirati studente upisane u 2020. na izborni predmet 'Programiranje za
veb'
-- na osnovu broja polozenih ESPB i ostvarene prosecne ocene
with
studenti as (
    select
        i.indeks,
        sum( p.espb ) as espb,
        avg( ocena + 0. ) as ocena
    from ispit i
        join predmet p
            on i.idpredmeta = p.id
```

```
    where i.status = 'o'
      and ocena > 5
    group by i.indeks
)
select
    p.id, p.oznaka, p.naziv,
    d.indeks, d.ime, d.prezime,
    s.espb, s.ocena
from predmet p
    join upisankurs uk
        on p.id = uk.idpredmeta
        and uk.skgodina = 2020
    join dosije d
        using( indeks )
    join studenti s
        using( indeks )
where p.naziv like 'Pro% za _eb'
  and d.datdiplomiranja is null
order by 1, 7 desc, 8 desc;
```

Rezultat:

ID	OZNAKA	NAZIV	INDEKS	IME	PREZIME	ESPB	OCENA
2343	R291	Programiranje za veb	20160086	Natalija	Nikolic	216	7.615385
2343	R291	Programiranje za veb	20150429	Rada	Djoric	205	7.222222
2343	R291	Programiranje za veb	20160423	Mina	Zivanovic	193	8.142857
2343	R291	Programiranje za veb	20150012	Kristina	Savatovic	193	7.057143
2343	R291	Programiranje za veb	20160301	Jana	Albijanic	186	8.750000
…	…	…	…	…	…	…	…

Ukupno: 109 redova × 8 kolona; prikaz je skraćen.

Objašnjenje: Ovaj upit rangira studente upisane 2020. godine na izborni predmet „Programiranje za veb” prema broju položenih ESPB i prosečnoj oceni. Imenovani podupit `studenti` za svakog studenta izračunava zbir ESPB i prosek položenih predmeta, a glavni upit povezuje te podatke sa upisom na predmet. Maska `'Pro% za _eb'` obuhvata različite zapise naziva predmeta (npr. „veb” i „Veb”).

Primer 3.46 — Rangiranje izbornih predmeta

```
-- rangirati izborne predmete upisane u 2020. prema broju studenata
-- i prosecnoj oceni studenata koji su ih upisali
-- za predmete sa bar 25 upisanih studenata
with
```

```
studenti as (
    select
        indeks,
        avg( ocena + 0. ) as prosek
    from ispit
    where status = 'o'
      and ocena > 5
    group by indeks
),
predmeti as (
    select
        pp.idpredmeta,
        count(*) n,
        avg(prosek) prosek
    from predmetprograma pp
        join upisankurs uk
            on pp.idpredmeta = uk.idpredmeta
            and uk.skgodina = 2020
        join dosije d
            on d.indeks = uk.indeks
            and d.idprograma = pp.idprograma
        join studenti s
            on s.indeks = d.indeks
    where vrsta = 'izborni'
    group by pp.idpredmeta
    having count(*) >= 25
)
select p.*, n, prosek
from predmet p
    join predmeti ps
        on p.id = ps.idpredmeta
order by n desc, prosek desc;
```

Rezultat:

ID	OZNAKA	NAZIV	ESPB	N	PROSEK
1894	M3.04	Geometrija 4	5	123	7.850414
2343	R291	Programiranje za veb	6	109	7.683580
2180	O62	Osnovi mehanike	6	104	7.939383
1907	AM02	Odabrana poglavlja astronomije	5	99	8.030590
2341	R269	Racunarska inteligencija	6	92	7.745577
2337	R246	Funkcionalno programiranje	6	91	7.726705

Napomena: prikazano je prvih 6 od 25 redova.

Objašnjenje: Ovaj upit rangira izborne predmete upisane 2020. godine prema broju studenata i prosečnoj oceni studenata koji su ih upisali, za predmete sa najmanje 25 upisanih studenata. Podupit `studenti` izračunava prosek svakog studenta, a podupit `predmeti`, koji koristi prethodni podupit, broj studenata i prosek njihovih proseka za svaki izborni predmet.

Čas 4 — Rekurzivni upiti, pogledi, menjanje podataka i pravljenje tabela

Primer 4.1 — Povezivanje sa bazom podataka

```
connect to stud2020 user student using abcdef;
set schema da;
```

Rezultat: veza sa bazom podataka je uspostavljena, a tekuća shema postavljena.

1. Rekurzivni upiti

Rekurzivni upiti koriste se za izračunavanje **tranzitivnog zatvorenja** relacija. Na primer, ako tabela sadrži grane grafa, rekurzivnim upitom mogu se pronaći svi parovi posredno povezanih čvorova.

Rekurzivni upit zapisuje se kao imenovani podupit naredbe `WITH`, koji je unija (obavezno `UNION ALL`) dva dela. Prvi deo je **početni korak** (iteracija 0, npr. neposredne veze), a drugi je **rekurzivni korak**, koji opisuje kako se na osnovu rezultata iteracije *N* (referencom na sam podupit) gradi dopuna do iteracije *N* + 1:

```
WITH ...
rekurzivan_upit AS (
    <početni korak, ne koristi rekurzivan_upit>
    UNION ALL
    <rekurzivni korak, koristi rekurzivan_upit>
)
...
```

Početni korak je obično jednostavan. Rekurzivni korak obavezno sadrži spajanje nekih podataka sa prethodnom iteracijom podupita. Pri tome važe sledeća pravila:

- rekurzivni korak ne sme da sadrži spajanje operatorom `JOIN`, već se koristi implicitno spajanje: tabele se navode u klauzuli `FROM`, a uslov spajanja u klauzuli `WHERE`;
- početni i rekurzivni korak moraju imati isti broj kolona, istih tipova, pa čak i istih imena; zato je često jednostavnije navesti imena kolona pri imenovanju podupita:

```
rekurzivan_upit( <imena kolona> ) AS ( <telo upita> )
```

Rekurzivni upit se izračunava na sledeći način:

1. izračuna se početni (nerekurzivni) korak;
2. izračuna se rekurzivni korak, pri čemu se kao sadržaj rekurzivno referisanog podupita koristi **samo rezultat prethodnog koraka**;
3. korak 2 se ponavlja sve dok njegov rezultat ne bude prazan.

Rezultat upita je unija rezultata svih koraka.

1.1. Primer: tranzitivne uslovnosti predmeta

Naredni primer izračunava sve uslovnosti predmeta, i neposredne i posredne. Neposredne uslovnosti su zapisane u tabeli `USLOVNIPREDMET`: predmet `idpredmeta` na programu `idprograma` uslovljen je predmetom `iduslovnogpredmeta`. Posredne uslovnosti izvode se iz neposrednih:

- početni korak izdvaja neposredne uslovnosti, sa udaljenošću 1;
- rekurzivni korak produžava svaku do tada izračunatu uslovnost za jedan korak, spajanjem sa neposrednim uslovnostima: ako je predmet A uslovljen predmetom B (sa udaljenošću *d*), a B neposredno predmetom C, onda je A uslovljen predmetom C sa udaljenošću *d* + 1.

Primer 4.2 — Sve uslovnosti predmeta

```
with
uslovljenost( idprograma, idpredmeta, iduslovnogpredmeta, udaljenost ) as (
    -- neposredno ustanovljene uslovnosti
    select idprograma, idpredmeta, iduslovnogpredmeta, 1
    from uslovnipredmet

    union all

    -- nadovezivanje up na u, gradjenje nove iteracije
    select u.idprograma, u.idpredmeta, up.iduslovnogpredmeta, u.udaljenost+1
    from uslovljenost u, uslovnipredmet up
    where u.idprograma = up.idprograma
      and u.iduslovnogpredmeta = up.idpredmeta
)
select u.*, p1.naziv, p2.naziv
from uslovljenost u
    join predmet p1
```

```
        on u.idpredmeta = p1.id
    join predmet p2
        on u.iduslovnogpredmeta = p2.id;
--order by p1.naziv, p2. naziv, udaljenost
```

Rezultat:

IDPROGRAMA	IDPREDMETA	IDUSLOVNOGPREDMETA	UDALJENOST	NAZIV	NAZIV
101	1600	1595	1	Programiranje 2	Programiranje 1
101	1883	1594	1	Diskretna matematika	Uvod u matematicku logiku
101	1889	1878	1	Algebra 2	Algebra 1
101	1890	2166	1	Diferencijalne jednacine A	Analiza 1
101	1890	2167	1	Diferencijalne jednacine A	Linearna algebra
…	…	…	…	…	…

Ukupno: 648 redova × 6 kolona; prikaz je skraćen.

Objašnjenje: U prvom koraku izračunavaju se uslovnosti udaljenosti 1, u drugom udaljenosti 2, zatim 3 i tako redom, sve dok rezultat koraka ne bude prazan. U ovim podacima ne postoje uslovnosti udaljenosti 5, pa se izračunavanje završava posle petog koraka. Glavni upit povezuje dobijene uslovnosti sa nazivima predmeta.

Ako se ne navede uređivanje, rezultat obično sadrži redove onim redom kojim su izračunati, što u slučaju rekurzivnog upita lepo ilustruje opisani redosled izračunavanja.

Primer 4.3 — Najkraće udaljenosti uslovnosti

```
with
uslovljenost( idprograma, idpredmeta, iduslovnogpredmeta, udaljenost ) as (
    -- neposredno ustanovljene uslovnosti
    select idprograma, idpredmeta, iduslovnogpredmeta, 1
    from uslovnipredmet

    union all

    -- nadovezivanje up na u, gradjenje nove iteracije
    select u.idprograma, u.idpredmeta, up.iduslovnogpredmeta, u.udaljenost+1
    from uslovljenost u, uslovnipredmet up
    where u.idprograma = up.idprograma
      and u.iduslovnogpredmeta = up.idpredmeta
),
najkrace as (
    select idprograma, idpredmeta, iduslovnogpredmeta, min(udaljenost) as
```

```
udaljenost
    from uslovljenost
    group by idprograma, idpredmeta, iduslovnogpredmeta
)
select u.*, p1.naziv, p2.naziv
from najkrace u
    join predmet p1
        on u.idpredmeta = p1.id
    join predmet p2
        on u.iduslovnogpredmeta = p2.id
order by p1.naziv, p2. naziv, udaljenost;
```

Rezultat:

IDPROGRAMA	IDPREDMETA	IDUSLOVNOGPREDMETA	UDALJENOST	NAZIV	NAZIV
103	2492	1578	1	Alati za razvoj softvera	Programiranje 1
103	2492	1588	1	Alati za razvoj softvera	Programiranje 2
301	1684	1680	1	Algebarska geometrija	Algebra 4
301	1684	2295	1	Algebarska geometrija	Komutativna algebra
103	2328	1580	1	Algebra 1	Diskretne strukture 1
…	…	…	…	…	…

Ukupno: 449 redova × 6 kolona; prikaz je skraćen.

Objašnjenje: Rezultat može da sadrži ponavljanja: neke uslovnosti su istovremeno i neposredne i posredno izvodive, a neke se mogu izvesti na više različitih načina (sa različitim udaljenostima). Zato ni `DISTINCT` ne bi u potpunosti rešio problem. Rešenje je da se podaci grupišu po paru predmeta (i programu) i da se izdvoji najmanja udaljenost. To se postiže novim imenovanim podupitom `najkrace`, koji se u glavnom upitu koristi umesto podupita `uslovljenost`.

Primer 4.4 — Samo posredne uslovnosti

```
with
uslovljenost( idprograma, idpredmeta, iduslovnogpredmeta, udaljenost ) as (
    -- neposredno ustanovljene uslovnosti
    select idprograma, idpredmeta, iduslovnogpredmeta, 1
```

```
    from uslovnipredmet

    union all

    -- nadovezivanje up na u, gradjenje nove iteracije
    select u.idprograma, u.idpredmeta, up.iduslovnogpredmeta, u.udaljenost+1
    from uslovljenost u, uslovnipredmet up
    where u.idprograma = up.idprograma
      and u.iduslovnogpredmeta = up.idpredmeta
),
najkrace as (
    select idprograma, idpredmeta, iduslovnogpredmeta, min(udaljenost) as
udaljenost
    from uslovljenost
    group by idprograma, idpredmeta, iduslovnogpredmeta
)
select u.*, p1.naziv, p2.naziv
from najkrace u
    join predmet p1
        on u.idpredmeta = p1.id
    join predmet p2
        on u.iduslovnogpredmeta = p2.id
where udaljenost > 1
order by p1.naziv, p2. naziv, udaljenost;
```

Rezultat:

IDPROGRAMA	IDPREDMETA	IDUSLOVNOGPREDMETA	UDALJENOST	NAZIV	NAZIV
101	1915	2166	2	Analiza 3A	Analiza 1
101	1957	2166	2	Analiza 3A	Analiza 1
101	1972	2166	2	Analiza 3A	Analiza 1
101	1918	2166	2	Analiza 3B	Analiza 1
101	1958	2166	2	Analiza 3B	Analiza 1
…	…	…	…	…	…

Ukupno: 51 redova × 6 kolona; prikaz je skraćen.

Objašnjenje: Da bi se izdvojile samo uslovnosti koje nisu neposredne, dodaje se uslov da najmanja udaljenost bude veća od 1.

2. Pogledi

Ako se neki upit često koristi, on se može sačuvati kao logički objekat baze podataka koji se naziva **pogled**. Pogled se koristi kao tabela, ali fizički ne sadrži podatke: on je *virtuelna tabela*, čiji se sadržaj izračunava pri svakoj upotrebi. Upotreba pogleda olakšava pisanje upita, ali ne doprinosi performansama. Pogled se pravi naredbom:

```
CREATE [OR REPLACE] VIEW <ime> [( <lista imena kolona> )]
AS <upit>
...
```

Opcija `OR REPLACE` nalaže da se postojeći pogled istog imena zameni novom definicijom. Neke druge opcije biće predstavljene kasnije.

Primer 4.5 — Pogled sa studentima osnovnih studija

```
-- pravimo pogled sa svim studentima osnovnih studija
create or replace view dosije_osnovne as
select d.*
from dosije d
    join studijskiprogram sp
        on d.idprograma = sp.id
where sp.idnivoa = 1;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Ovaj pogled sadrži samo studente osnovnih studija, tj. sve podatke iz tabele `DOSIJE` o studentima upisanim na neki program osnovnih studija (nivo 1).

Primer 4.6 — Čitanje pogleda

```
select *
from dosije_osnovne;
```

Rezultat:

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150369	101	Jelena	Mandic	z	Jagodina	-1	2015-07-07
20150069	101	Nemanja	Nicic	m	Sabac	-2	2015-07-06
20150090	103	Andrijana	Beara	z	Sabac	-2	2015-07-06
20150320	103	Milos	Milenkovic	m	Krusevac	2	2015-07-06

INDEKS	IDPROGRAMA	IME	PREZIME	POL	MESTORODJENJA	IDSTATUSA	DATUPISA
20150185	101	Valentina	Stojanov	z	Beograd (Savski venac)	2	2015-07-06
…	…	…	…	…	…	…	…

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 2690 redova × 9 kolona; prikaz je skraćen.

Objašnjenje: Pogled se zatim čita kao bilo koja tabela.

Primer 4.7 — Pogled sa položenim predmetima

```
-- pravimo pogled sa svim polozenim predmetima
create or replace view polozenpredmet as
select indeks, idpredmeta, datpolaganja, poeni, ocena
from ispit
where status = 'o'
  and ocena > 5;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Ovaj pogled sadrži samo one podatke o ispitima koji predstavljaju potvrdu da je ispit položen. Upiti nad tabelom i nad pogledom pokazuju da pogled obuhvata samo deo redova tabele.

Primer 4.8 — Broj redova tabele ISPIT

```
select count(*) from ispit;
```

Rezultat:

1
119143

Primer 4.9 — Broj redova pogleda POLOZENPREDMET

```
select count(*) from polozenpredmet;
```

Rezultat:

1
32823

2.1. Brisanje pogleda

Pogled se briše naredbom:

```
DROP VIEW <ime pogleda>
```

Brisanjem pogleda ne brišu se podaci iz tabela na kojima pogled počiva. Naredba `DROP` koristi se i za brisanje drugih vrsta objekata baze podataka.

Primer 4.10 — Brisanje pogleda DOSIJE_OSNOVNE

```
drop view dosije_osnovne;
```

Rezultat: naredba je uspešno izvršena.

Primer 4.11 — Brisanje pogleda POLOZENPREDMET

```
drop view polozenpredmet;
```

Rezultat: naredba je uspešno izvršena.

3. Naredbe za menjanje podataka

Osnovne naredbe za menjanje podataka su:

- `INSERT` — dodavanje novih redova u tabelu;
- `UPDATE` — menjanje sadržaja postojećih redova tabele;
- `DELETE` — brisanje redova iz tabele.

U osnovnim oblicima ove naredbe su relativno jednostavne, a složenije oblike dobijaju u kombinaciji sa upitima i njihovim delovima.

3.1. Naredba INSERT

Naredba `INSERT` dodaje nove redove u izabranu tabelu:

```
INSERT INTO <ime tabele> [( <lista imena kolona> )]
<upit>
```

Lista kolona je opciona. Ako se ne navede, podrazumevaju se sve kolone tabele, redosledom kojim su navedene pri definisanju tabele. Ako se navede, kolone koje nisu navedene u dodatim redovima dobijaju podrazumevane vrednosti (ako su definisane), odnosno nedefinisane vrednosti. Zato lista mora da sadrži sve kolone koje ne smeju imati nedefinisane vrednosti, a nemaju definisanu podrazumevanu vrednost.

Primer 4.12 — Dodavanje redova bez liste kolona

```
-- upisujemo izborni predmet Teorija izracunljivosti (2022)
-- u prvom semestru 2020/21
-- studentima 20180008, 20180012, 20180057 i 20180145
insert into upisankurs
values
    ( 20180008, 2020, 1, 2022 ),
    ( 20180012, 2020, 1, 2022 ),
    ( 20180057, 2020, 1, 2022 ),
    ( 20180145, 2020, 1, 2022 );
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: U osnovnom obliku upit je izraz `VALUES` sa spiskom redova, a u složenijim slučajevima proizvoljan upit. U oba slučaja broj i redosled kolona upita moraju odgovarati navedenoj listi kolona, odnosno kolonama tabele. Prvi primer ne navodi imena kolona; tabela `UPISANKURS` ima kolone `INDEKS`, `SKGODINA`, `SEMESTAR` i `IDPREDMETA`.

Primer 4.13 — Dodavanje redova sa listom kolona

```
-- upisujemo izborni predmet Teorija izracunljivosti (2022)
-- u prvom semestru 2020/21
-- studentima 20180008, 20180012, 20180057 i 20180145
insert into upisankurs( indeks, skgodina, semestar, idpredmeta )
values
    ( 20180008, 2020, 1, 2022 ),
    ( 20180012, 2020, 1, 2022 ),
    ( 20180057, 2020, 1, 2022 ),
    ( 20180145, 2020, 1, 2022 );
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Ovaj primer je ista naredba sa eksplicitno navedenim imenima kolona. Ako su redovi iz prethodnog primera već dodati, naredba se neće izvršiti, jer bi se u tabeli pojavili redovi sa vrednostima primarnog ključa koje već postoje. (Prikazani rezultat dobijen je izvršavanjem u trenutku kada ti redovi nisu postojali.)

Primer 4.14 — Sadržaj tabele UPISANKURS

```
select * from upisankurs;
```

Rezultat:

INDEKS	SKGODINA	SEMESTAR	IDPREDMETA
20150069	2015	1	1594

INDEKS	SKGODINA	SEMESTAR	IDPREDMETA
20150069	2015	1	1595
20150069	2015	1	2166
20150069	2015	1	2167
20150069	2015	1	2168
…	…	…	…
20200165	2020	1	2168
20180008	2020	1	2022
20180012	2020	1	2022
20180057	2020	1	2022
20180145	2020	1	2022

Ukupno: 74044 redova × 4 kolona; prikaz je skraćen.

Primer 4.15 — Brisanje dodatih redova

```
delete from upisankurs
where indeks in (20180008, 20180012, 20180057, 20180145)
  and idpredmeta = 2022
  and skgodina = 2020 and semestar = 1;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Prethodno dodati redovi brišu se naredbom `DELETE` (odeljak 3.2).

Primer 4.16 — Dodavanje redova pomoću upita

```
-- upisujemo izborni predmet Teorija izracunljivosti (2022)
-- u prvom semestru 2020/21
-- studentima 20180008, 20180012, 20180057 i 20180145
insert into upisankurs
with indeksi(indeks) as ( values 20180008, 20180012, 20180057, 20180145 )
select indeks, 2020, 1, 2022
from indeksi;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Isti redovi mogli bi se dodati i pomoću upita (ali ne nakon što su već dodati). Upit može biti i naredba `WITH`.

3.2. Naredba DELETE

Naredba `DELETE` briše postojeće redove iz izabrane tabele:

```
DELETE FROM <ime tabele>
[ WHERE <uslov> ]
```

Brišu se svi redovi za koje je uslov tačan. Navođenje uslova je opciono, ali ga ne treba izostavljati bez jasne namere: bez uslova brišu se **svi** redovi tabele, bez ikakvog upozorenja.

Primer 4.17 — Brisanje upisanog predmeta

```
-- Ispisujemo upisan predmet 2022 navedenim studentima u tekućoj šk.godini
delete from upisankurs
where indeks in  ( 20180008, 20180012, 20180057, 20180145 )
  and idpredmeta = 2022
  and skgodina = 2020;
```

Rezultat: naredba je uspešno izvršena.

3.3. Naredba UPDATE

Naredba `UPDATE` menja vrednosti u postojećim redovima tabele:

```
UPDATE <ime tabele>
SET { <pojedinačne dodele> | <grupna dodela> }
[ WHERE <uslov> ]
```

Pojedinačne dodele imaju oblik `<ime kolone> = <izraz> [, <ime kolone> = <izraz>]*`, a grupna dodela oblik `( <lista kolona> ) = <upit>`, gde upit izračunava tačno jedan red sa odgovarajućim brojem kolona. U oba slučaja mogu se koristiti podupiti. Izrazi na desnoj strani izračunavaju se nad vrednostima reda **pre** izmene, pa izraz `poeni + 10` koristi staru vrednost kolone `poeni`. Menjaju se svi redovi za koje je uslov tačan; bez uslova menjaju se svi redovi tabele.

Primer 4.18 — Podatak koji će biti izmenjen

```
select *
from ispit
where indeks = 20150185
  and idpredmeta = 1907
  and status = 'o';
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2019	jan1	20150185	1907	o	2020-02-01	71	8

Objašnjenje: Najpre proverimo podatak o polaganju koji ćemo menjati.

Primer 4.19 — Izmena pojedinačnim dodelama

```
-- studentu 20150185 ispraviti pogresno upisanu ocenu 8 i upisati 9
-- i povecati broj poena za 10
-- iz predmeta 'Odabrana poglavlja astronomije' (ID=1907)
-- u ispitnom roku jan1/2020 (sk.godina 2019)
update ispit
set ocena = 9,
    poeni = poeni + 10
where indeks = 20150185
  and idpredmeta = 1907
  and status = 'o'
  and ocena = 8;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Naredba iz ovog primera menja ocenu i broj poena tom polaganju, a naredba iz Primera 4.20 ga vraća u prvobitno stanje.

Primer 4.20 — Vraćanje prvobitnih vrednosti

```
-- Poništavamo izmenu i vraćamo stare podatke:
update ispit
set ocena = 8,
    poeni = 71
where indeks = 20150185
  and idpredmeta = 1907
  and status = 'o';
```

Rezultat: naredba je uspešno izvršena.

Primer 4.21 — Izmena dodelom listi kolona

```
-- isto kao prethodno ažuriranje, ali sa drugom sintaksom
update ispit
set ( ocena, poeni ) = ( 9, poeni + 10 )
where indeks = 20150185
  and idpredmeta = 1907
  and status = 'o'
  and ocena = 8;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Iste izmene mogu se zapisati i u obliku dodele vrednosti listi kolona.

Primer 4.22 — Vraćanje prvobitnih vrednosti dodelom listi kolona

```
-- Poništavamo izmenu ali sa drugom sintaksom:
update ispit
set ( ocena, poeni ) = ( 8, 71 )
where indeks = 20150185
  and idpredmeta = 1907
  and status = 'o';
```

Rezultat: naredba je uspešno izvršena.

3.4. Naredba MERGE

Naredba `MERGE` ažurira ciljnu tabelu na osnovu izvorne tabele: redovi ciljne tabele kojima odgovaraju redovi izvorne tabele se menjaju ili brišu, a redovi koji postoje samo u izvornoj tabeli dodaju se u ciljnu.

```
MERGE INTO <ime tabele>
USING <upit> ON <uslov povezivanja>
{ WHEN [NOT] MATCHED [AND <uslov>] THEN <akcija> }+
[ELSE IGNORE]
```

gde akcija može biti:

```
UPDATE ... SET <pojedinačne dodele ili grupna dodela>
DELETE ...
INSERT [(<lista kolona>)] VALUES <izraz>
```

a `...` označava dodatne opcije. Grana `WHEN MATCHED` primenjuje se na parove redova koji zadovoljavaju uslov povezivanja, a grana `WHEN NOT MATCHED` na redove izvorne tabele kojima ne odgovara nijedan red ciljne tabele.

Primer 4.23 — Naredba MERGE

```
merge into ispit i
using (select * from sep2020) sep20
on i.skgodina = sep20.skgodina
    and i.oznakaroka = sep20.oznakaroka
    and i.indeks = sep20.indeks
    and i.idpredmeta = sep20.idpredmeta
when matched then
    update set
        i.status = sep20.status,
```

```
        i.datpolaganja = sep20.datpolaganja,
        i.poeni = sep20.poeni,
        i.ocena = sep20.ocena
when not matched then
    insert values (
        sep20.skgodina,
        sep20.oznakaroka,
        sep20.indeks,
        sep20.idpredmeta,
        sep20.status,
        sep20.datpolaganja,
        sep20.poeni,
        sep20.ocena
    );
```

Rezultat:

```
SQL0204N  "DA.SEP2020" is an undefined name.  SQLSTATE=42704  SQLCODE=-204
```

Objašnjenje: Na primer, ako bi tabela `SEP2020` sadržala ažurne rezultate ispita iz septembarskog roka, a tabela `ISPIT` neke od tih podataka, moguće i neažurne, prepisivanje svih rezultata u tabelu `ISPIT` moglo bi da se izvede ovom naredbom. Tabela `SEP2020` ne postoji u bazi Stud2020, pa je naredba prijavila grešku `SQL0204N` (nedefinisano ime); primer ilustruje sintaksu.

4. Menjanje podataka i pogledi

Naredbe za menjanje podataka mogu se primeniti i na poglede, ali samo kada je nedvosmisleno jasno šta se i kako menja. Ako se pogled dobija iole složenijom operacijom od čitanja iz jedne tabele, obično se ne može koristiti za menjanje podataka. Pogled ne može da se koristi za menjanje podataka ako se u njegovoj definiciji:

- spaja više tabela;
- koristi grupisanje;
- koriste skupovne operacije;
- neke kolone dobijaju kao rezultat izraza.

Primer 4.24 — Pogled zasnovan na spajanju

```
-- pravimo pogled sa svim studentima osnovnih studija
create or replace view dosije_osnovne as
select d.*
from dosije d
    join studijskiprogram sp
        on d.idprograma = sp.id
where sp.idnivoa = 1;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Ovo je ranije napravljeni pogled sa studentima osnovnih studija. Pošto se u njegovoj definiciji spajaju dve tabele, nije nedvosmisleno određeno koje podatke treba promeniti: u opštem slučaju neki redovi mogu ostati nespojeni ili biti spojeni više puta. Zato se ovaj pogled ne može koristiti za menjanje podataka.

Primer 4.25 — Pogled nad jednom tabelom, sa podupitom

```
-- pravimo pogled sa svim studentima osnovnih studija
create or replace view dosije_osnovne as
select *
from dosije d
where d.idprograma in (
    select id
    from studijskiprogram
    where idnivoa = 1
);
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Problem se ponekad može rešiti drugačijom definicijom pogleda. Ovaj pogled ima isti sadržaj kao prethodni, ali je definisan upitom nad tačno jednom tabelom (DOSIJE), dok se druga tabela koristi samo u podupitu. Takav pogled može se koristiti za menjanje podataka.

Menjanje podataka u pogledima može se rešavati i pomoću okidača (eng. *triggers*), kojima se eksplicitno opisuje kako se izvode operacije menjanja podataka posredstvom pogleda. O tome će biti reči na jednom od narednih časova.

4.1. Ograničavanje promena putem pogleda

Menjanje podataka putem pogleda može imati i neočekivane posledice: može se dogoditi da novi ili izmenjeni red ne zadovoljava uslov pogleda, pa se ne vidi kroz pogled, iako je nastao ili izmenjen putem njega. Prethodni pogled obuhvata samo studente osnovnih studija. Ako bi se putem njega dodao red koji se odnosi na studenta master studija, dodavanje bi uspelo, ali se novi red ne bi mogao pročitati istim pogledom. Slično, ako bi se naredbom `UPDATE` na tom pogledu nekom studentu studijski program promenio u

program master studija, izmena bi uspela, ali se izmenjeni red više ne bi video kroz pogled — kao da je obrisan iz pogleda, iako je ostao u tabeli `DOSIJE`. Takve promene ponekad imaju smisla, ali najčešće je poželjno da problematične operacije dodavanja i menjanja proizvedu grešku.

Da bi se pri radu sa pogledom proveravalo da li izmene zadovoljavaju uslov pogleda, pogled se pravi sa tzv. **opcijom provere**, koja se navodi na kraju definicije pogleda:

```
...
[ WITH [ LOCAL | CASCADED ] CHECK OPTION ]
```

Provera može biti lokalna ili kaskadna (podrazumevano). **Lokalna** provera koristi uslov iz definicije samog pogleda, kao i uslove onih pogleda na kojima on počiva koji i sami imaju opciju provere. **Kaskadna** provera koristi i uslov tog pogleda i (konjunkcijom) uslove svih pogleda na kojima on počiva, bez obzira na to da li je u njima navedena opcija provere. U svakom slučaju, ako pogled V ima opciju provere, a pogled Q je definisan nad pogledom V, svaki pokušaj menjanja podataka putem Q proverava i uslov pogleda V. Opcija provere ne sme se navesti ako je pogled samo za čitanje (tj. ako nije dopušteno menjanje, kao što je opisano u ovom odeljku) ili ako su za pogled definisani okidači za menjanje.

Primer 4.26 — Pogled sa opcijom provere

```
create or replace view dosije_osnovne as
select *
from dosije d
where d.idprograma in (
    select id
    from studijskiprogram
    where idnivoa = 1
)
with check option;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Ovaj primer pravi pogled sa studentima osnovnih studija, sa opcijom provere.

5. Pravljenje objekata baze podataka

Za pravljenje, menjanje i brisanje objekata baze podataka postoji veći broj naredbi, oblika:

```
CREATE <vrsta objekta> ...
ALTER <vrsta objekta> ...
DROP <vrsta objekta> ...
```

5.1. Pravljenje tabela

Tabela se pravi naredbom:

```
CREATE TABLE [ IF NOT EXISTS ] <ime tabele>
( <lista elemenata> )
[ <opcije> ]
```

Elementi se razdvajaju zapetama. Element tabele može biti definicija kolone, primarnog ključa, stranog ključa, jedinstvenog ključa ili uslova integriteta. Opcija `IF NOT EXISTS` nalaže da se naredba ne izvrši (i ne prijavi grešku) ako tabela tog imena već postoji.

Definicija kolone ima oblik:

```
<ime kolone> <tip> [<opcije>]
```

gde su najvažnije opcije:

- `NOT NULL` — kolona ne sme sadržati nedefinisane vrednosti;
- `[CONSTRAINT <ime>] <uslov integriteta>` — uslov integriteta kolone;
- `[WITH] DEFAULT <vrednost>` — podrazumevana vrednost.

Uslovi integriteta kolone mogu biti:

- `PRIMARY KEY` — kolona je primarni ključ;
- `UNIQUE` — nikoja dva reda ne smeju imati istu vrednost u koloni;
- uslov referisanja (stranog ključa), oblika:

```
REFERENCES <ime tabele> [( <ime kolone> )]
    [ ON DELETE { NO ACTION | RESTRICT | CASCADE | SET NULL } ]
    [ ON UPDATE { NO ACTION | RESTRICT } ]
```

- `CHECK ( <uslov> )` — uslov ograničenja domena, u kome se ne smeju neposredno pojavljivati druge kolone.

Primarni ključ, **jedinstveni ključ** i **strani ključ** koji obuhvataju više kolona definišu se kao posebni elementi tabele:

```
[CONSTRAINT <ime>] PRIMARY KEY ( <lista imena kolona> )
[CONSTRAINT <ime>] UNIQUE ( <lista imena kolona> )
[CONSTRAINT <ime>] FOREIGN KEY ( <lista imena kolona> )
    REFERENCES <ime tabele> [( <lista kolona> )]
    [ ON DELETE { NO ACTION | RESTRICT | CASCADE | SET NULL } ]
    [ ON UPDATE { NO ACTION | RESTRICT } ]
```

Opcije `ON DELETE` i `ON UPDATE` određuju šta se dešava kada se obriše ili izmeni red na koji se strani ključ odnosi: `NO ACTION` i `RESTRICT` sprečavaju operaciju, `CASCADE` briše i redove koji se na njega odnose, a `SET NULL` postavlja strani ključ tih redova na `NULL`.

Uslov integriteta kao element tabele zadaje uslov koji svaki red tabele mora da zadovolji; u njemu se mogu pojaviti sve kolone reda:

```
[CONSTRAINT <ime>] CHECK ( <uslov> )
```

Primeri pravljenja tabela nalaze se u skriptovima za pravljenje baze podataka Stud2020.

5.2. Alternativni načini pravljenja tabela

Nove tabele često je jednostavnije praviti po uzoru na postojeće. Sistem Db2 omogućava dva načina. Tabela iste strukture (sa istim kolonama) kao neka postojeća pravi se naredbom:

```
CREATE TABLE [ IF NOT EXISTS ] <ime tabele>
LIKE <ime postojeće tabele>
[...]
```

Tabela čija je struktura određena rezultatom upita (imenima i tipovima njegovih kolona) pravi se naredbom:

```
CREATE TABLE [ IF NOT EXISTS ] <ime tabele> [( <lista imena kolona> )]
AS ( <upit> )
WITH [NO] DATA
```

Opcija `WITH DATA` nalaže da se nova tabela odmah popuni rezultatom upita, a `WITH NO DATA` da se napravi prazna tabela.

Primer 4.27 — Tabela iste strukture kao ISPIT

```
create table ispit_a like ispit;
```

Rezultat: naredba je uspešno izvršena.

Primer 4.28 — Tabela napravljena na osnovu upita, sa podacima

```
create table ispit_b as (
    select i.*, p.naziv
    from ispit i
        join predmet p
            on i.idpredmeta = p.id
)
with data;
```

Rezultat: naredba je uspešno izvršena.

Primer 4.29 — Čitanje nove tabele

```
select * from ispit_b
limit 100;
```

Rezultat:

SKGODINA	OZNAKAROKA	INDEKS	IDPREDMETA	STATUS	DATPOLAGANJA	POENI	OCENA
2015	jan1	20150112	2170	o	2016-01-18	73	8
2015	jan1	20150069	1594	o	2016-02-03	100	10
2015	jan1	20150069	1595	o	2016-02-02	96	10
2015	jan1	20150069	2168	o	2016-01-30	92	10
2015	jan1	20150109	1594	o	2016-02-17	100	10
…	…	…	…	…	…	…	…

Napomena: prikazano je prvih 8 od 9 kolona.

Ukupno: 100 redova × 9 kolona; prikaz je skraćen.

5.3. Menjanje strukture tabele

Struktura tabele menja se naredbom:

```
ALTER TABLE <ime tabele>
{ ADD <element> | ALTER <element> | DROP <element> }*
[ <opcije> ]
```

Primer 4.30 — Dodavanje kolone i stranog ključa

```
alter table predmet
add column nalik_na int
add constraint fk_predmet foreign key(nalik_na)
    references predmet
    on delete set null
    on update restrict;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Sintaksa je veoma bogata; ovde je prikazana samo za kolone i uslove integriteta. Ovaj primer dodaje tabeli `PREDMET` kolonu `nalik_na` i strani ključ koji se odnosi na samu tabelu `PREDMET` (na njen primarni ključ). Pri brisanju predmeta na koji se

neki red odnosi, vrednost kolone u tom redu postavlja se na `NULL`; menjanje primarnog ključa takvog predmeta nije dopušteno.

Primer 4.31 — Tabela PREDMET sa novom kolonom

```
select * from predmet
limit 20;
```

Rezultat:

ID	OZNAKA	NAZIV	ESPB	NALIK_NA
1578	P100	Programiranje 1	8	NULL
1580	M105	Diskretne strukture 1	6	NULL
1588	P101	Programiranje 2	8	NULL
1590	M106	Diskretne strukture 2	6	NULL
1594	M1.02	Uvod u matematicku logiku	5	NULL
...	...	...	...	...

Ukupno: 20 redova × 5 kolona; prikaz je skraćen.

Primer 4.32 — Uklanjanje kolone

```
alter table predmet
--drop constraint fk_predmet
drop column nalik_na;
```

Rezultat: naredba je uspešno izvršena.

Objašnjenje: Kolona se uklanja elementom `DROP COLUMN`. Uz nju se uklanjaju i uslovi integriteta koji je obuhvataju; zakomentarisani red pokazuje kako bi se strani ključ uklonio eksplicitno.

Primer 4.33 — Pokušaj reorganizacije tabele

```
reorg table da.predmet;
```

Rezultat:

```
SQL0104N  An unexpected token "da" was found following "reorg table ".
Expected tokens may include:  "JOIN".  SQLSTATE=42601  SQLCODE=-104
```

Objašnjenje: Posle promene strukture tabele, zavisno od vrste promene i popunjenosti tabele, neke operacije i upiti nad tabelom mogu biti onemogućeni dok se tabela ne reorganizuje. Tabela se reorganizuje naredbom `REORG TABLE <tabela>`. To je komanda procesora komandne linije sistema Db2, a ne SQL naredba: u terminalu (`db2 -t`) izvršava se neposredno, dok se u alatima koji naredbe prosleđuju kao SQL (npr. DataGrip) dobija greška `SQL0104N`, kao u prikazanom rezultatu. Iz SQL-a se reorganizacija poziva posredstvom procedure `SYSPROC.ADMIN_CMD`, npr. `CALL SYSPROC.ADMIN_CMD('REORG TABLE da.predmet');`.

5.4. Brisanje tabele

Tabela se briše naredbom:

```
DROP TABLE <ime tabele>
```

Brisanjem tabele brišu se i svi njeni podaci.

Primer 4.34 — Brisanje tabele ISPIT_A

```
drop table ispit_a;
```

Rezultat: naredba je uspešno izvršena.

Primer 4.35 — Brisanje tabele ISPIT_B

```
drop table ispit_b;
```

Rezultat: naredba je uspešno izvršena.

Izvor primera

Primeri u ovom dokumentu preuzeti su iz materijala prof. dr Saše Malkova za kurs „Uvod u relacione baze podataka” (Matematički fakultet, Univerzitet u Beogradu). Navedeni su bez većih izmena, a objašnjenja uz njih su nešto detaljnija od izvornih.