

Uvod u SQL

D301



priručnik za polaznike © 2016 Srce

TEČAJEVI **srca**



srce

Sveučilište u Zagrebu
Sveučilišni računski centar

Ovu inačicu priručnika izradio je autorski tim Srca u sastavu:

Autor: Edin Mujadžević

Recenzent: Krešimir Tkalec

Urednica: Gorana Kurtović

Lektorica: Mia Kožul

TEČAJEVI Srca

Sveučilište u Zagrebu

Sveučilišni računski centar

Josipa Marohnića 5, 10000 Zagreb

edu@srce.hr

ISBN 978-953-7138-96-7 (meki uvez)

ISBN 978-953-7138-97-4 (PDF)

Verzija priručnika D301-20170223



Ovo djelo dano je na korištenje pod licencom *Creative Commons Imenovanje-Nekomercijalno-Dijeli pod istim uvjetima 4.0 međunarodna*.

Licenca je dostupna na stranici:

<http://creativecommons.org/licenses/by-nc-sa/4.0/>.

Sadržaj

Uvod.....	1
1. Stvaranje tablica	3
1.1. Uvod	3
1.2. Sustavi za upravljanje bazama podataka	3
1.3. Entiteti i atributi	4
1.4. Tipovi podataka	4
1.5. Vrijednost NULL	5
1.6. Primarni ključ tablice	5
1.7. Stvaranje baze podataka	6
1.8. Stvaranje tablice	7
1.9. Izmjena i dodavanje stupaca.....	9
1.10. Brisanje tablice	9
1.11. Brisanje baze podataka.....	10
2. Dohvaćanje podataka.....	11
2.1. Naredba SELECT.....	11
2.2. Razdvajanje naredbi u SQL-u	12
2.3. Komentari u SQL-u.....	12
2.4. Odabir svih stupaca.....	12
2.5. Aliasi	13
2.6. Konstantne vrijednosti	13
2.7. Spajanje stupaca i druge operacije	14
2.8. Sortiranje podataka	15
2.9. Dohvat prvih nekoliko redaka.....	16
2.10. Distinkcija – uklanjanje duplikata	16
2.11. Vježba: Dohvaćanje podataka	17
3. Postavljanje uvjeta	19
3.1. Klauzula WHERE	19
3.2. Operatori uspoređivanja	20
3.3. Operator LIKE	20
3.4. Usporedba s vrijednošću NULL	21
3.5. Logički operatori i kombiniranje uvjeta	22
3.6. Operator IN.....	23
3.7. Operator BETWEEN	24
3.8. Naredba SELECT INTO.....	24
4. Unos, izmjena i brisanje redaka	27
4.1. Naredba INSERT	27
4.2. Naredba UPDATE	28
4.3. Naredba DELETE.....	29
5. Pomoćne funkcije.....	31
5.1. Rad sa znakovnim nizovima	31
5.2. Rad s numeričkim podacima.....	33
5.3. Rad s datumskim podacima.....	33

5.4.	Pretvorba između tipova podataka	35
5.5.	Vježba: Postavljanje uvjeta, unos, izmjena i brisanje redaka, rad s pomoćnim funkcijama...	36
6.	Strani ključ i tipovi veza između tablica	37
6.1.	Strani ključ	37
6.2.	Veza jedan naprama više	38
6.3.	Veza više naprama više.....	38
6.4.	Veza jedan naprama jedan.....	39
6.5.	Dijagram baze podataka.....	40
7.	Spajanje tablica	41
7.1.	Kartezijev produkt tablica.....	41
7.2.	Unutarnji spoj	42
7.3.	Lijevi spoj	44
7.4.	Desni spoj	44
7.5.	Puni spoj	45
7.6.	Spoj tablice sa samom sobom.....	46
7.7.	Spoj više tablica	46
7.8.	Spoj po nejednakosti	46
7.9.	Vježba: Spajanje tablica	48
8.	Agregatne funkcije.....	49
8.1.	Funkcija COUNT.....	49
8.2.	Funkcija SUM	50
8.3.	Funkcija AVG	51
8.4.	Funkcija MAX.....	51
8.5.	Funkcija MIN.....	52
8.6.	Agregatne funkcije i vrijednost NULL	52
9.	Grupiranje podataka	55
9.1.	Grupiranje po jednom stupcu.....	55
9.2.	Grupiranje po više stupaca	57
9.3.	Grupiranje uz agregatne funkcije.....	59
9.4.	Postavljanje uvjeta nad grupom	59
9.5.	Grupiranje i vrijednost NULL.....	61
9.6.	Vježba: Grupiranje i agregatne funkcije.....	62
10.	Operacije nad skupovima	63
10.1.	Unija.....	63
10.2.	Unija s ponavljanjima.....	64
10.3.	Presjek	65
10.4.	Razlika	65
11.	Podupiti.....	67
11.1.	Podupit.....	67
11.2.	Podupit u listi za odabir.....	67
11.3.	Korelirani podupit.....	68
11.4.	Podupit u uvjetu	68
11.5.	Usporedba s rezultatom podupita.....	68
11.6.	Podupit s operatorom IN.....	69

11.7. Podupit s ključnom riječi EXISTS	70
11.8. Podupit u klauzulama FROM i JOIN	71
11.9. Podupiti i vrijednost NULL	72
11.10. Alternativa korištenju podupita	73
12. Pogledi	75
12.1. Pogled	75
12.2. Stvaranje pogleda	75
12.3. Korištenje pogleda	76
12.4. Brisanje pogleda	76
13. Indeksi	77
13.1. Indeksi	77
13.2. Stvaranje indeksa	78
13.3. Stvaranje indeksa nad više stupaca	78
13.4. Stvaranje UNIQUE indeksa	79
13.5. Brisanje indeksa	79
13.6. Vježba: Podupiti i operacije nad skupovima	80
14. Dodatak	81
14.1. Pregled sintakse naredbe SELECT	81
14.2. Dijagram baze podataka	83
14.3. Instalacija Microsoft SQL Server LocalDB sustava	84
14.4. Instalacija Microsoft SQL Server Management Studia	87

Uvod

Namjena ovog tečaja jest upoznavanje polaznika s osnovama SQL-a – jezika za postavljanje upita nad bazama podataka. Osim samim jezikom SQL, tečaj se bavi i izradom baze podataka pri čemu daje praktične preporuke za dizajn baze podataka.

Priručnik se sastoji od trinaest poglavlja i pet praktičnih vježbi za koje su potrebni dodatni materijali.

Ideja ovog tečaja jest da se uz čitanje objašnjenja svaki primjer odmah praktično isproba od strane polaznika. U tu svrhu uz tečaj dolazi i pokazna baza podataka, koja je napravljena na temelju modela obrazovnog centra koji održava tečajeve. Pokaznu bazu polaznici mogu i sami izraditi kod kuće, pokretanjem SQL skripte koja se može preuzeti na web stranici tečaja.

Tečaj je namijenjen za rad sa sustavom *Microsoft SQL Server*. Većina primjera (osim primjera iz poglavlja 5. *Pomoćne funkcije*) napisana je u standardnoj inačici jezika SQL i radit će na gotovo svim sustavima za upravljanje bazama podataka.

U priručniku se rabe ovi simboli:

- VELIKIM SLOVIMA JEDNAKE ŠIRINE su istaknute ključne riječi i nazivi funkcija u jeziku SQL (u primjerima kôda, ali i unutar teksta)
- sintaksa SQL naredbi opisana je u zasebnom odlomku, unutar **SIVOG OKVIRA** (u zasebnom odlomku)
- primjeri SQL naredbi ispisani su u zasebnom odlomku slovima jednake širine, tj. monospace fontom
- dijelovi teksta na koje je potrebno obratiti posebnu pozornost označeni su **masno**

U okvirima sa strane navedene su zanimljivosti i napomene.

Zanimljivosti i napomene

Svi su podaci o polaznicima i predavačima u pokaznoj bazi izmišljeni, odnosno nasumično generirani. Svaka sličnost sa stvarnim osobama je slučajna.

1. Stvaranje tablica

Po završetku ovog poglavlja polaznik će moći:

- navesti nekoliko sustava za upravljanje bazama podataka
- razumjeti osnovne principe modeliranja baza podataka
- razlikovati najčešće tipove podataka koji se koriste u bazama podataka
- objasniti što je to `NULL` vrijednost
- razumjeti što je primarni ključ tablice
- stvoriti i obrisati bazu podataka
- stvoriti i obrisati tablicu
- izmijeniti strukturu tablice.

Osnovni gradivni element u bazama podataka su tablice. Svaki redak u tablici predstavlja pojedini objekt (ili odnos) iz stvarnog svijeta.

U ovom poglavlju daje se uvod u rad s bazama podataka, objašnjavaju se naredbe za stvaranje baze podataka i tablica, te tipovi podataka koji se mogu koristiti.

1.1. Uvod

SQL je kratica za *Structured Query Language* (strukturirani upitni jezik) – jezik za postavljanje upita nad bazama podataka.

Originalno razvijen u IBM-u 70-ih godina prošlog stoljeća, SQL je postao glavni jezik za rad s bazama podataka.

Jezik SQL je napisan s namjerom da bude lako razumljiv i jednostavan za korištenje. On je jezik deklarativnog tipa – navodi se što se želi dobiti, a ne daju se konkretne instrukcije kako to dobiti (kao što je slučaj kod jezika proceduralnog tipa).

1.2. Sustavi za upravljanje bazama podataka

U upotrebi se nalazi više različitih sustava za upravljanje bazama podataka (*database management systems*).

Na tečaju se koristi sustav za upravljanje bazama podataka Microsoft SQL Server, koji je pogodan i za velike poslovne sustave kao i za male baze podataka. Radi se o komercijalnom sustavu za koji se plaća licenca, ali koji je dostupan i u besplatnoj (Express) verziji.

Osim njega, u širokoj su upotrebi i sustavi za upravljanje bazama podataka otvorenog koda - MySQL (najčešće se upotrebljava za web stranice i manje projekte) i PostgreSQL (koji je pogodan za ozbiljne primjene na velikim sustavima).

U velikim poslovnim sustavima često se koristi sustav za upravljanje bazama podataka Oracle, a u upotrebi su još i sustavi Informix, Ingres i DB2.

Iako je definiran SQL standard, svaki od sustava ima određene varijacije i nadogradnje na osnovni SQL jezik.

Zanimljivosti i napomene

SQL se može izgovarati kao akronim S.Q.L., ili kao engleska riječ *sequel*. Razlog tomu je što je originalni naziv ovog jezika bio SEQUEL (Structured English Query Language).

SQL koji se radi u okviru ovog tečaja primjenjiv je na većini sustava. U slučaju kad to nije tako bit će posebno naznačeno.

1.3. Entiteti i atributi

Objekt iz stvarnog svijeta koji se želi predstaviti u bazi podataka naziva se entitetom. Entitet može biti stvaran objekt ili osoba (kao npr. polaznik tečaja), ili pak apstraktni objekt ili događaj (pohađanje tečaja).

Tip entiteta određen je njegovim atributima. Tako je tip entiteta *Polaznik* određen atributima *Ime* i *Prezime*.

Za svaki tip entiteta koji postoji u modelu, stvara se tablica u bazi podataka. Stupci tablice odgovaraju atributima entiteta. Za svaki atribut potrebno je, osim imena atributa, definirati i koje vrijednosti on može poprimiti, tj. koji je njegov tip podatka.

Jedan redak u tablici predstavlja pojedini entitet iz stvarnog svijeta, dakle jedan redak u tablici *Polaznik* predstavlja točno određenog polaznika s njegovim imenom i prezimenom.

1.4. Tipovi podataka

Baze podataka podržavaju velik broj tipova podataka. Ovdje su navedeni oni koji se najčešće koriste:

Zanimljivosti i napomene

Tipovi podataka dani u ovom pregledu odnose se na Microsoft SQL Server. Drugi sustavi imaju slične tipove podataka, ali postoje određene razlike.

Tip podatka	Objašnjenje
char	Niz znakova fiksne duljine. Potrebno je navesti duljinu – npr. <code>char(10)</code>
nchar	Niz znakova fiksne duljine zapisan u kôdnoj stranici UTF-8. Potrebno je navesti duljinu – npr. <code>nchar(10)</code>
varchar	Niz znakova varijabilne duljine. Potrebno je navesti maksimalnu duljinu – npr. <code>varchar(100)</code>
nvarchar	Niz znakova fiksne duljine zapisan u kôdnoj stranici UTF-8. Potrebno je navesti maksimalnu duljinu – npr. <code>nvarchar(100)</code>
int	Cijeli broj u rasponu od -2 147 483 648 do 2 147 483 647
decimal	Decimalni broj. Moguće je definirati broj znamenaka uključujući decimale (defaultno 18, maksimalno 38) i broj decimala (defaultno 2)
float	Decimalni broj zapisan u formatu s pomičnim zarezom
bit	Logička istina (1) ili laž (0)
date	Datum u rasponu od 1.1.0001 do 31.12.9999
datetime	Datum zajedno s vremenskom komponentom, u rasponu od 1.1.1753 00:00:00 do 31.12.9999 23:59:59.997

Navedeni tipovi podatka odnose se na sustav za upravljanje bazama podataka Microsoft SQL Server, mada većina ovih tipova ima svoje ekvivalente na drugim sustavima.

Prilikom definicija stupca u tablici određuje se njegov tip podatka, odnosno kakve je vrijednosti moguće spremati u tom polju.

1.5. Vrijednost NULL

NULL je specijalna vrijednost koja se koristi u bazama podataka, i koja označava da neka vrijednost nije poznata, ili da nije primjenjiva.

U praksi, pri unosu podataka u bazu, može se dogoditi da neki od podataka nije dobiven ili je nepoznat.

Na primjer, za svakog se polaznika unosi njegova dob. Ako neki od polaznika ne želi dati taj podatak, onda se kod njega kao vrijednost za dob unosi NULL.

NULL vrijednost nije isto što i vrijednost 0. Vrijednost 0 je poznata vrijednost, dok u slučaju NULL-a vrijednost nije poznata. Ako bi za dob upisali 0, to bi imalo sasvim drugo značenje (da osoba ima 0 godina).

Drugi primjer za NULL vrijednost može biti OIB. Ako je polaznik tečaja stranac, on nema OIB, odnosno kolona OIB nije primjenjiva na njega, i kao vrijednost za OIB upisuje se NULL.

Prilikom definicije stupca u tablici navodi se smije li on sadržavati NULL vrijednosti.

1.6. Primarni ključ tablice

Primarni ključ tablice jest stupac čija vrijednost jednoznačno identificira određeni redak u tablici.

Kako je osobu moguće jednoznačno identificirati pomoću njezinog osobnog identifikacijskog broja ili OIB-a, kolona OIB bila bi prirodan izbor za primarni ključ tablice *Polaznik*.

U tablici ne možemo imati dva retka s istom vrijednošću primarnog ključa. OIB ispunjava i taj zahtjev, jer svaki građanin ima jedinstveni OIB.

No, pokazuje se da je u praksi preporučljivo za primarni ključ tablice koristiti podatak koji nema značenje u stvarnom svijetu (tzv. surogatni ključ).

Prilikom unosa OIB-a može doći do pogreške, pa je tu vrijednost potrebno ispraviti. Također, ako se radi o pravnim osobama, može doći do promjene OIB-a. Izmjena vrijednosti primarnog ključa u pravilu je problematična jer se njegova vrijednost koristi za referenciranje tog retka u drugim tablicama.

Zbog toga se preporuča korištenje autogeneriranog ključa (npr. niz brojeva od 1 do N). U tom slučaju sama baza automatski svakom unesenom retku dodjeljuje sljedeću raspoloživu vrijednost iz niza.

Na taj način nikad ne postoji potreba za izmjenom vrijednosti primarnog ključa jer ta vrijednost ne nosi stvarno značenje.

1.7. Stvaranje baze podataka

Za stvaranje baze podataka koristi se naredba `CREATE DATABASE`.

U najjednostavnijoj varijanti (bez navođenja dodatnih parametara), njezina sintaksa izgleda ovako:

```
CREATE DATABASE BazaPodataka
```

Sljedeća naredba stvorit će bazu podataka *Tecajevi*:

```
CREATE DATABASE Tecajevi
```

Prilikom poziva naredbe `CREATE DATABASE` moguće je navesti i dodatne parametre, odnosno postavke za stvaranje baze podataka.

Neki od osnovnih parametara baze podataka koje možemo podesiti na sustavu Microsoft SQL Server sljedeći su:

Naziv parametra	Objašnjenje
NAME	naziv datoteke u kojoj je pohranjena baza podataka
FILENAME	putanja do datoteke baze podataka
SIZE	početna veličina datoteke baze podataka
MAXSIZE	maksimalna veličina datoteke baze podataka
FILEGROWTH	veličina za koju će rasti datoteka baze podataka prilikom povećanja

Ako ne navedemo ove parametre, oni će biti postavljeni na predefimirane vrijednosti.

Osim same datoteke baze podataka (kod sustava Microsoft SQL Server to je datoteka s ekstenzijom *.mdf*), u kojoj su fizički zapisani struktura i podatci, za svaku bazu podataka stvara se i transakcijski dnevnik (*transaction log*) – datoteka u koju se zapisuje svaka promjena izvršena nad bazom.

Gore navedeni parametri mogu se podešavati i za datoteku transakcijskog dnevnika (kod sustava Microsoft SQL Server to je datoteka s ekstenzijom *.ldf*).

Proširena sintaksa naredbe `CREATE DATABASE` izgleda ovako (u prvom dijelu su navedeni parametri za datoteku baze podataka, dok su u drugom dijelu navedeni parametri za datoteku transakcijskog dnevnika):

```
CREATE DATABASE BazaPodataka ON (
  NAME = NazivDatoteke,
  FILEPATH = Putanja,
  SIZE = Velicina,
  MAXSIZE = MaksimalnaVelicina,
  FILEGROWTH = Povecanje
)
LOG ON (
  NAME = NazivDatoteke,
  FILEPATH = Putanja,
  SIZE = Velicina,
```

Zanimljivosti i napomene

Iz transakcijskog dnevnika moguće je doći do ranije verzije podataka te na taj način poništiti nehotične izmjene na podacima ili povratiti staru verziju baze podataka. No, budući da se u nju zapisuju sve promjene, ova datoteka s vremenom naraste te ju je potrebno s vremena na vrijeme smanjivati.

```

MAXSIZE = MaksimalnaVelicina,
FILEGROWTH = Povecanje
)

```

Primjer naredbe s kojom se stvara baza podataka uz postavljanje ovih parametara izgledao bi ovako:

```

CREATE DATABASE Tecajevi ON (
    NAME = 'Tecajevi',
    FILEPATH = 'C:\MSSQL\DATA\Tecajevi.mdf',
    SIZE = 3072KB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 1024KB
)
LOG ON (
    NAME = 'Tecajevi_log',
    FILEPATH = 'C:\MSSQL\DATA\Tecajevi_log.ldf',
    SIZE = 7616KB,
    MAXSIZE = 2048GB,
    FILEGROWTH = 10%
)

```

1.8. Stvaranje tablice

Prije navođenja naredbe kojom se stvara tablica (ili drugih naredbi koje mijenjaju strukturu baze), preporuča se navesti naredbu `USE`. Ovom se naredbom zadaje baza podataka na kojoj će izvršiti naredbe koje slijede iza nje, a njena sintaksa izgleda ovako:

```

USE BazaPodataka

```

Za stvaranje tablice koristi se naredba `CREATE TABLE`.

Njezina sintaksa izgleda ovako:

```

CREATE TABLE Tablica (
    Stupac1 Tip1,
    Stupac2 Tip2,
    ...
)

```

Nakon navođenja naziva tablice, unutar zagrada navodi se popis stupaca odvojenih zarezima. Za svaki stupac navodi se njegov naziv i tip, te eventualne dodatne postavke.

Sljedeće dvije naredbe stvorit će na bazi *Tecajevi* tablicu *Polaznik* u kojoj će se držati podaci o polaznicima tečajeva:

```

USE Tecajevi
CREATE TABLE Polaznik(
    Id int,
    OIB char(11),
    Ime nvarchar(50),
    Prezime nvarchar(50),
    Adresa nvarchar(250),
    Email nvarchar(250),
    Dob int
)

```

Zanimljivosti i napomene

Naredba `USE` specifična je za sustav Microsoft SQL Server. Ako se ne navede naredba `USE` (i ako baza nije odabrana pomoću sučelja SQL Server Management Studia), naredbe će se izvršiti na sistemskoj *master* bazi, što nikako nije poželjno.

Prvi stupac, *Id*, bit će cjelobrojnog tipa. On će sadržavati identifikator (šifru) polaznika.

Drugi stupac, *OIB*, bit će znakovnog tipa. Ovdje će se koristiti tip podatka *char* zato što OIB uvijek ima točno 11 znamenaka. Uz znakovne se tipove obavezno navodi širina polja (unutar zagrade).

Za stupce *Ime* i *Prezime* koristi se tip *nvarchar*, jer se mogu pojaviti nacionalni znakovi, a duljina podatka može varirati. Kao maksimalna duljina postavlja se 50 znakova.

Za stupac *Dob*, koji sadrži dob polaznika, koristi se cjelobrojni tip podatka.

Ako se ne navede drugačije, stupac smije sadržavati NULL vrijednost. Da bi se zabranilo da stupac sadrži NULL (nepoznatu) vrijednost, potrebno je u definiciji stupca navesti ključne riječi NOT NULL.

Na ovaj se način prilikom definiranja tablice određuje da su stupci *Id*, *Ime* i *Prezime* obavezni, odnosno da ne smiju sadržavati NULL vrijednost:

```
CREATE TABLE Polaznik(  
    Id int NOT NULL,  
    OIB char(11),  
    Ime nvarchar(50) NOT NULL,  
    Prezime nvarchar(50) NOT NULL,  
    Adresa nvarchar(250),  
    Email nvarchar(250),  
    Dob int  
)
```

Za stupac *Id* poželjno je naznačiti da se radi o primarnom ključu tablice, navođenjem ključnih riječi PRIMARY KEY:

```
CREATE TABLE Polaznik(  
    Id int NOT NULL PRIMARY KEY,  
    OIB char(11),  
    Ime nvarchar(50) NOT NULL,  
    Prezime nvarchar(50) NOT NULL,  
    Adresa nvarchar(250),  
    Email nvarchar(250),  
    Dob int  
)
```

Kako bi vrijednost identifikatora u stupcu *Id* bila automatski dodijeljena svakom unesenom retku, potrebno je navesti funkciju IDENTITY uz dva parametra. Prvi je početna vrijednost niza (najčešće se uzima 1), a drugi je vrijednost za koju se povećava (također 1).

```
CREATE TABLE Polaznik(  
    Id int NOT NULL PRIMARY KEY IDENTITY(1,1),  
    OIB char(11),  
    Ime nvarchar(50) NOT NULL,  
    Prezime nvarchar(50) NOT NULL,  
    Adresa nvarchar(250),  
    Email nvarchar(250),  
    Dob int  
)
```

Prvi redak u tablici *Polaznik* dobit će identifikator 1, drugi 2, treći 3 i tako dalje.

1.9. Izmjena i dodavanje stupaca

Za izmjene na definiciji tablice koristi se naredba `ALTER TABLE`.

Ako se želi obrisati stupac iz tablice, naredba će izgledati ovako:

```
ALTER TABLE Tablica
DROP COLUMN Stupac
```

Sljedeća naredba obrisat će stupac *Dob* iz tablice *Polaznik*:

```
ALTER TABLE Polaznik
DROP COLUMN Dob
```

Ako se želi dodati stupac u tablicu, naredba će izgledati ovako:

```
ALTER TABLE Tablica
ADD Stupac Tip
```

Sljedeća naredba dodaje stupac *DatumRodjenja* (datumskog tipa) u tablicu *Polaznik*:

```
ALTER TABLE Polaznik
ADD DatumRodjenja date
```

Ako se želi izmijeniti stupac u tablici (promjena tipa ili izmjena dodatnih postavki), naredba će izgledati ovako:

```
ALTER TABLE Tablica
ALTER COLUMN Stupac Tip
```

Sljedeća naredba će povećati duljinu polja *Prezime* u tablici *Polaznik* na 100 znakova:

```
ALTER TABLE Polaznik
ALTER COLUMN Prezime nvarchar(100) NOT NULL
```

1.10. Brisanje tablice

Za brisanje tablice iz baze podataka služi naredba `DROP TABLE`.

Njezina sintaksa izgleda ovako:

```
DROP TABLE Tablica
```

Sljedeća naredba briše iz baze podataka tablicu *Polaznik*:

```
DROP TABLE Polaznik
```

Ova naredba nepovratno briše tablicu i sve podatke koji se nalaze u njoj.

1.11. Brisanje baze podataka

Za brisanje baze podataka služi naredba `DROP DATABASE`.

Njezina sintaksa izgleda ovako:

```
DROP DATABASE Baza
```

Sljedeća naredba briše bazu podataka *Tecajevi*:

```
DROP DATABASE Tecajevi
```

Ova naredba nepovratno briše bazu podataka i sve tablice koje se nalaze u njoj.

2. Dohvaćanje podataka

Po završetku ovog poglavlja polaznik će moći:

- navesti pojedine stupce u listi za odabir ili odabrati sve stupce
- pisati komentare u SQL-u
- koristiti aliase umjesto naziva stupaca
- kombinirati vrijednosti iz stupaca u složenije izraze
- poredati retke u rezultatu
- dohvatiti samo dio redaka iz rezultata
- vratiti retke s različitim vrijednostima.

U ovom poglavlju polaznik će se susresti sa naredbom `SELECT` – osnovnom naredbom u jeziku SQL. Ovo poglavlje bavi se raznim mogućnostima dohvaćanja podataka i oblikovanja koje su pritom moguće.

2.1. Naredba `SELECT`

Za dohvat podataka iz baze koristi se naredba `SELECT`. U svom najjednostavnijem obliku, ova naredba se sastoji iz dva dijela (tzv. klauzule) – klauzule `SELECT` u kojoj se navode stupci koje se želi ispisati (odvojeni zarezima), te klauzule `FROM`, u kojoj se navodi tablica iz koje se dohvaćaju podaci.

Lista stupaca u klauzuli `SELECT` naziva se još i listom za odabir (selekciju).

```
SELECT Stupac1, Stupac2, Stupac3...
FROM Tablica
```

Sljedeća naredba dohvaća stupce *Ime* i *Prezime* iz tablice *Polaznik*:

```
SELECT Ime, Prezime
FROM Polaznik
```

Stupce je moguće navesti proizvoljnim redoslijedom (ne mora se slijediti redoslijed kojim su stupci definirani pri stvaranju tablice):

```
SELECT Prezime, Ime, OIB
FROM Polaznik
```

Rezultat naredbe `SELECT` uvijek je u tabličnom obliku.

SQL nije osjetljiv na razliku između malih i velikih slova. Ključne riječi (npr. `SELECT`) mogu se pisati i malim slovima, no radi preglednosti uobičajen je način pisanja ključnih riječi velikim slovima. I nazive stupaca također je moguće pisati zanemarujući razliku između malih i velikih slova.

Jezik SQL je također neosjetljiv na praznine i prelazak u novi red te su sva uvlačenja i pisanje svake klauzule u novom redu napravljena isključivo radi povećanja čitljivosti naredbe.

Sljedeći upit je u potpunosti ekvivalentan upitu iz prethodnog primjera:

```
select
    prezime,
    ime,
    oib
from polaznik
```

2.2. Razdvajanje naredbi u SQL-u

U načelu, svaka SQL naredba stoji zasebno, no ako se želi navesti više SQL naredbi koje je potrebno izvršiti, na većini sustava za upravljanje bazama podataka potrebno ih je razdvojiti pomoću separatora `;`.

U sljedećem primjeru navedena su dva upita razdvojena separatorom:

```
SELECT Ime, Prezime
FROM Polaznik;

SELECT Sifra, Naziv, Cijena
FROM Tecaj;
```

Zanimljivosti i napomene

Više SQL naredbi nalazi se zajedno najčešće unutar datoteke. Datoteka koja sadrži SQL naredbe naziva se SQL skriptom.

Zanimljivosti i napomene

Na sustavu Microsoft SQL Server nije potrebno razdvajati naredbe separatorom `;`.

Zanimljivosti i napomene

Na sustavu MySQL jednolinijski komentar mora počinjati kombinacijom znakova `--`, tj potrebno je navesti razmak nakon dvije crtice.

2.3. Komentari u SQL-u

Radi pojašnjenja namjene nekog upita, ponekad je poželjno dodati komentare.

Komentari se mogu protezati kroz više redaka, i u tom slučaju ih je potrebno započeti kombinacijom znakova `/*`, a završiti kombinacijom znakova `*/`.

```
/* Ovaj upit dohvaća sve polaznike.
   Ispisuju se stupci ime i prezime. */

SELECT Ime, Prezime
FROM Polaznik
```

Ako se komentar nalazi samo u jednoj liniji, potrebno ga je započeti kombinacijom znakova `--`.

```
-- Ovaj upit dohvaća sve polaznike.

SELECT Ime, Prezime
FROM Polaznik
```

2.4. Odabir svih stupaca

Ako se u rezultatu želi prikazati sve stupce iz tablice, nije potrebno navoditi sve stupce pojedinačno. U popisu stupaca dovoljno je navesti samo znak `*`.

```
SELECT *
FROM Polaznik
```

U rezultatu će biti prikazani svi stupci iz tablice, i to onim redoslijedom kojim su definirani prilikom stvaranja tablice.

2.5. Aliasi

Aliasi se koriste da bi se neki od stupaca u rezultatu prikazao pod drugim nazivom.

Prilikom navođenja stupca, potrebno je uz naziv stupca navesti ključnu riječ **AS** i alias pod kojim želimo prikazati stupac.

```
SELECT Stupac AS Alias
FROM Tablica
```

Sljedeća naredba dohvaća stupce *Ime*, *Prezime* i *OIB* iz tablice *Polaznik*, s tim da se stupac *OIB* prikazuje pod nazivom *OsobniBroj*:

```
SELECT Ime, Prezime, OIB AS OsobniBroj
FROM Polaznik
```

2.6. Konstantne vrijednosti

Pri pisanju SQL naredbi može biti potrebno, npr. u različitim izrazima ili uvjetima, navesti konstantne vrijednosti. Za različite tipove podataka vrijednosti se navode na različit način.

U sljedećoj su tablici prikazani primjeri konstantnih vrijednosti za različite tipove podataka:

Tip podatka	Primjeri konstantnih vrijednosti
znakovni	'A' 'Ivica Ivić'
cjelobrojni	250 -15
decimalni	3.1452 -15.25
datumski	'2015-01-25'
datum i vrijeme	'2015-01-25 16:50:00'
logički	0 (laž) 1 (istina)

Kod znakovnih tipova podataka (*char*, *varchar* i drugi) najbitnije je pravilo da se znakovna konstanta uvijek navodi unutar jednostrukih navodnika ('). Ako se u samoj znakovnoj konstanti nalazi znak ', onda ga je potrebno navesti dva puta.

Sljedeći primjer ispisat će tekst „O' Brien“:

```
SELECT 'O'' Brien'
```

Kod cjelobrojnih podataka nema nekih posebnih napomena, ali kod decimalnih tipova podataka treba pripaziti da se kao decimalni separator uvijek koristi točka, a ne zarez.

Datum i vrijeme također se navode unutar jednostrukih navodnika. Postoji više formata za zapisivanje datuma i vremena, a u primjeru je dan standardni format SQL-a za datum (godine, mjeseci pa dani sa separatorom -) i vrijeme (sati, minute i sekunde sa separatorom :).

Zanimljivosti i napomene

U ovom primjeru koristi se **SELECT** naredba bez klauzule **FROM**, dakle podaci se ne dohvaćaju iz tablice, već se radi s konstantnim vrijednostima.

Rezultat ovakvog upita uvijek će biti samo jedan redak.

Kod logičkih vrijednosti (tip podatka `bit`) vrijednost 0 predstavlja laž, dok 1 predstavlja istinu.

Konstantne vrijednosti najčešće se koriste u operacijama s vrijednostima stupaca (npr. preračunavanje i sl.), te kod provjere vrijednosti stupaca u uvjetima. Ovaj prvi način korištenja bit će opisan u sljedećem poglavlju.

2.7. Spajanje stupaca i druge operacije

Pri navođenju stupaca u naredbi `SELECT` moguće su određene operacije nad stupcima. Najčešća operacija nad stupcima je spajanje dvaju ili više stupaca u jedan.

Za spajanje znakovnih nizova koristi se operator `+`. Sljedeća naredba spojiti će stupce *Ime* i *Prezime* u jedan stupac:

```
SELECT Ime + Prezime
FROM Polaznik
```

Kako bi ime i prezime u rezultatu bili odvojeni, između njih se može staviti razmak. U SQL-u se znakovni nizovi uvijek navode unutar jednostrukih navodnika, pa se razmak zapisuje kao `' '`.

```
SELECT Ime + ' ' + Prezime
FROM Polaznik
```

Stupac koji je dobiven spajanjem stupaca ili drugom operacijom nema svoj naziv, već mu ga je potrebno dodijeliti pomoću aliasa. Ova naredba dodjeljuje dobivenom stupcu alias *PunoIme*:

```
SELECT Ime + ' ' + Prezime AS PunoIme
FROM Polaznik
```

Na stupcima koji sadrže brojčane podatke moguće je raditi aritmetičke i druge operacije. Osnovne aritmetičke operacije u SQL-u su:

Operator	Operacija
+	zbrajanje
-	oduzimanje
*	množenje
/	dijeljenje
%	operacija modulo (određivanje ostatka pri dijeljenju)

U tablici *Tecaj* zapisani su podaci o tečajevima, između ostalog i njihova cijena. Da bi se prikazao iznos cijene s porezom, moguće je pomnožiti cijenu s faktorom 1.25 i prikazati dobiveni iznos u novom stupcu:

```
SELECT Naziv, Cijena * 1.25 AS CijenaSPorezom
FROM Tecaj
```

Osim aritmetičkih, nad stupcima je moguće izvoditi i logičke operacije te pozivati razne ugrađene funkcije.

Zanimljivosti i napomene

Na drugim sustavima operator za spajanje znakovnih nizova zapisuje se kao `||`.

2.8. Sortiranje podataka

Dobiveni retci mogu biti poredani po nekom kriteriju. U tu svrhu se naredbi `SELECT` dodaje rečenica `ORDER BY`, u kojoj se navodi jedan ili više stupaca po kojima se želi sortirati.

```
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica
ORDER BY Stupac1, Stupac2
```

Sljedeća će naredba dohvatiti retke iz tablice *Polaznik* sortirane po prezimenu:

```
SELECT Ime, Prezime, OIB
FROM Polaznik
ORDER BY Prezime
```

Ako se želi da polaznici koji imaju isto prezime budu poredani po imenu, potrebno je navesti da rezultat bude sortiran po prezimenu, a zatim po imenu:

```
SELECT Ime, Prezime, OIB
FROM Polaznik
ORDER BY Prezime, Ime
```

Smjer sortiranja može se definirati pomoću ključnih riječi `ASC` i `DESC`.

Ako se navede ključna riječ `ASC` (ili ako se uopće ne navede ključna riječ), smjer sortiranja bit će uzlazni (*ascending*) - od A do Z, odnosno od manjeg prema većem u slučaju brojčanih podataka.

Ova naredba ekvivalentna je prethodnoj:

```
SELECT Ime, Prezime, OIB
FROM Polaznik
ORDER BY Prezime ASC, Ime ASC
```

Ako se navede ključna riječ `DESC` smjer sortiranja će biti silazni (*descending*) – od Z do A, tj. od većeg prema manjem.

Ova će naredba imati obrnuti smjer sortiranja od prethodnog primjera:

```
SELECT Ime, Prezime, OIB
FROM Polaznik
ORDER BY Prezime DESC, Ime DESC
```

U klauzuli `ORDER BY` moguće je upotrebljavati aliase:

```
SELECT Ime + ' ' + Prezime AS PunoIme
FROM Polaznik
ORDER BY PunoIme
```

Osim pomoću naziva stupca ili aliasa, u klauzuli `ORDER BY` stupce možemo referencirati na temelju njihova rednog broja u listi za odabir:

```
SELECT Ime, Prezime, OIB
FROM Polaznik
ORDER BY 2 ASC, 1 DESC
```

Zanimljivosti i napomene

Klauzula `ORDER BY` uvijek dolazi kao posljednja u `SELECT` naredbi (iznimka su neke specijalne klauzule koje se ne rade na ovom tečaju).

Zanimljivosti i napomene

Ključna riječ `TOP` nije dio SQL standarda.

Na drugim sustavima (MySQL, PostgreSQL) umjesto ključne riječi `TOP` koristi se ključna riječ `LIMIT` koja dolazi na sam kraj naredbe `SELECT`.

2.9. Dohvat prvih nekoliko redaka

Ako se želi dohvatiti samo nekoliko redaka iz rezultata, koristi se ključna riječ `TOP`, koja dolazi prije navođenja stupaca.

```
SELECT TOP N Stupac1, Stupac2, Stupac3
FROM Tablica
```

Sljedeća će naredba dohvatiti prvih 10 redaka iz tablice *Polaznik*:

```
SELECT TOP 10 Ime, Prezime, OIB
FROM Polaznik
```

Ključna riječ `TOP` najkorisnija je u kombinaciji s klauzulom `ORDER BY`. Sljedeća naredba dohvatit će 5 najmlađih polaznika:

```
SELECT TOP 5 Ime, Prezime
FROM Polaznik
ORDER BY DatumRodjenja DESC
```

2.10. Distinkcija – uklanjanje duplikata

Ako u rezultatu postoji više istih redaka, moguće je ukloniti duplikate pomoću ključne riječi `DISTINCT`. Ona dolazi prije navođenja stupaca:

```
SELECT DISTINCT Stupac1, Stupac2, Stupac3
FROM Tablica
```

Ako se u popisu stupaca navede samo *Ime*, u rezultatu će sigurno biti više istih redaka, budući da u tablici *Polaznik* vjerojatno ima više polaznika s istim imenom.

```
SELECT Ime
FROM Polaznik
```

Da bi u se dobio rezultat bez duplikata, koristi se ključna riječ `DISTINCT`:

```
SELECT DISTINCT Ime
FROM Polaznik
```

2.11. Vježba: Dohvaćanje podataka

1. Prikažite sve stupce iz tablice *Tecaj*.
2. Prikažite šifru, naziv i cijenu tečaja iz tablice *Tecaj*.
3. Prikažite šifru i naziv spojene u jedan stupac (s razmakom između), te cijenu tečaja iz tablice *Tecaj*. Prvi stupac prikažite pod nazivom *PuniNaziv*.
4. Prikažite šifru, naziv i cijenu tečaja s popustom iz tablice *Tecaj*. Cijenu s popustom izračunajte oduzimanjem popusta od pune cijene. Posljednji stupac prikažite pod nazivom *CijenaSPopustom*.
5. Prikažite šifru, naziv i ukupno trajanje tečaja (ukupan broj sati) iz tablice *Tecaj*. Posljednji stupac prikažite pod nazivom *Trajanje*.
6. Prikažite retke iz tablice tečaj poredane po cijeni tečaja.
7. Prikažite retke iz tablice tečaj poredane po cijeni tečaja – od najveće cijene prema najmanjoj.
8. Prikažite 5 najskupljih tečajeva.
9. Prikažite popis svih mogućih trajanja tečajeva u danima. Svaka vrijednost u popisu smije biti navedena samo jednom.
10. Rezultat prethodnog upita poredajte od najveće prema najmanjoj vrijednosti.

3. Postavljanje uvjeta

Po završetku ovog poglavlja polaznik će moći:

- filtrirati retke iz rezultata na temelju usporedbe s drugim vrijednostima ili s vrijednošću NULL
- pronaći znakovni niz koji odgovara traženom uzorku
- kombinirati više uvjeta
- provjeriti nalazi li se vrijednost u skupu vrijednosti
- provjeriti nalazi li se vrijednost u zadanom rasponu.

U ovom poglavlju uvodi se klauzula `WHERE` naredbe `SELECT`. Pomoću nje postavlja se uvjet na skup redaka, odnosno filtriraju se retci iz tablice prema zadanom kriteriju.

3.1. Klauzula WHERE

Budući da se u tablicama često nalazi velik broj redaka, potrebno je moći dohvatiti točno određene retke. Da bi se filtrirali rezultati koje vraća upit, u naredbu `SELECT` dodaje se klauzula `WHERE`. Ona uvijek dolazi nakon klauzule `FROM`:

```
SELECT Stupac1, Stupac2, Stupac3...
FROM Tablica
WHERE Uvjet
```

Filtriranje redaka obavlja se na temelju nekog uvjeta. To može biti bilo koji logički izraz, a najčešće će se upotrebljavati operator jednakosti. Sljedeća naredba dohvaća samo one retke iz tablice *Polaznik* kod kojih je vrijednost stupca *Ime* jednaka „Josip“:

```
SELECT Ime, Prezime
FROM Polaznik
WHERE Ime = 'Josip'
```

Ako se uz klauzulu `WHERE` koristi i `ORDER BY`, potrebno je prisjetiti se da klauzula `ORDER BY` uvijek dolazi posljednja u naredbi `SELECT`.

```
SELECT Stupac1, Stupac2, Stupac3...
FROM Tablica
WHERE Uvjet
ORDER BY Stupac1, Stupac2
```

U sljedećoj će naredbi rezultat biti sortiran po prezimenu:

```
SELECT Ime, Prezime
FROM Polaznik
WHERE Ime = 'Josip'
ORDER BY Prezime
```

U klauzuli `WHERE` ne mogu se upotrebljavati aliasi stupaca. Sljedeći upit će javiti grešku:

```
SELECT Ime + ' ' + Prezime as PunoIme
FROM Polaznik
WHERE PunoIme = 'Josip Tomić'
```

Da bismo gornji upit napisali ispravno, moramo ponoviti cijeli izraz koji je korišten u listi za odabir:

```
SELECT Ime + ' ' + Prezime as PunoIme
FROM Polaznik
WHERE Ime + ' ' + Prezime = 'Josip Tomić'
```

3.2. Operatori uspoređivanja

U uvjetu `WHERE` klauzule najčešće se obavlja usporedba vrijednosti u nekom stupcu. Operatori uspoređivanja koji se mogu koristiti sljedeći su:

Zanimljivosti i napomene

Na većini sustava operator `<>` (različito od) može se pisati i kao `!=`.

Operator	Opis
=	jednako
>	veće od
<	manje od
>=	veće od ili jednako
<=	manje od ili jednako
<>	različito od

Sljedeći upit vratit će sve polaznike čiji je identifikator manji ili jednak broju 100:

```
SELECT *
FROM Polaznik
WHERE Id <= 100
```

Svi operatori uspoređivanja mogu se koristiti i nad tekstualnim podacima. U tom slučaju podaci se uspoređuju po abecednom redosljedju.

Ovaj upit vratit će sve polaznike čije je prezime po abecedi dolazi poslije „P“:

```
SELECT *
FROM Polaznik
WHERE Prezime > 'P'
```

3.3. Operator LIKE

Operator `LIKE` služi za usporedbu s tekstualnom vrijednošću, no na način da ta vrijednost samo djelomično odgovara zadanom znakovnom nizu.

Kao znak koji zamjenjuje druge znakove koristi se znak `%`. Ovaj znak zamjenjuje jedan ili više znakova.

Sljedeći upit vratit će sve polaznike čije ime počinje slovima 'Iv'. U skupu rezultata nalazit će se Ivica, Ivan, i tako dalje.

```
SELECT *
FROM Polaznik
WHERE Ime LIKE 'Iv%'
```

Dodavanjem modifikatora `NOT` može se obaviti provjera da neka vrijednost **nije** unutar zadanog skupa vrijednosti.

Sljedeći upit vratit će sve polaznike čije ime ne počinje slovima 'Iv':

```
SELECT *
FROM Polaznik
WHERE Ime NOT LIKE 'Iv%'
```

Znak `%` može zamijeniti više znakova, a ako se želi zamijeniti točno jedan znak, može se koristiti zamjenski znak `_`.

Sljedeći upit vratit će samo one polaznike čije je prezime Matković ili Marković, ali ne i one čije prezime je Mataković:

```
SELECT *
FROM Polaznik
WHERE Prezime LIKE 'Ma_ković'
```

3.4. Usporedba s vrijednošću NULL

Kod usporedbe s vrijednošću `NULL` potrebno je posvetiti posebnu pažnju. Budući da je vrijednost `NULL` nepoznata, usporedba uz pomoć standardnih operatora uspoređivanja nije moguća.

Iako u tablici *Polaznik* postoje retci kod kojih je vrijednost stupca `OIB` postavljena na `NULL`, sljedeći upit neće vratiti ni jedan redak:

```
SELECT *
FROM Polaznik
WHERE OIB = NULL
```

No, ni sljedeći upit također neće vratiti ni jedan redak:

```
SELECT *
FROM Polaznik
WHERE OIB <> NULL
```

Razlog je u tome što je vrijednost `NULL` nepoznata, te nije moguće reći je li neka poznata vrijednost jednaka ili nije jednaka toj vrijednosti. Svaka usporedba s vrijednošću `NULL` uvijek vraća logičku vrijednost `FALSE`.

Za usporedbu s vrijednošću `NULL` stoga je potrebno koristiti poseban izraz **`IS NULL`**. Sljedeći upit vratit će one polaznike koji nemaju `OIB`, dakle kod kojih je vrijednost stupca `OIB` postavljena na `NULL`:

```
SELECT *
FROM Polaznik
WHERE OIB IS NULL
```

Također je moguća i provjera da neka vrijednost nije `NULL`, korištenjem izraza **`IS NOT NULL`**. Sljedeći upit vratit će polaznike koji imaju `OIB`, tj. za koje vrijednost stupca `OIB` nije postavljena na `NULL`:

```
SELECT *
FROM Polaznik
WHERE OIB IS NOT NULL
```

3.5. Logički operatori i kombiniranje uvjeta

Uvjet klauzule `WHERE` može biti bilo koji logički izraz pa tako i kombinacija dvaju ili više logičkih izraza.

Logički operator **AND** spaja dva ili više uvjeta na način da je rezultat istina samo ako su svi uvjeti istiniti.

```
SELECT *
FROM Tablica
WHERE Uvjet1 AND Uvjet2 AND Uvjet3
```

Vrsta polaznika nalazi se u tablici *Tip*.

Sljedeći upit vratit će samo polaznike čije je ime Ivan, ali im je ujedno vrsta polaznika (stupac *TipId*) jednaka 1, što označava studente.

```
SELECT *
FROM Polaznik
WHERE Ime = 'Ivan' AND TipId = 1
```

Ako se pomoću operatora **AND** spoji tri ili više izraza, rezultat će biti istina samo ako su svi istiniti. Sljedeći upit vratit će samo one polaznike koji zadovoljavaju sva tri uvjeta (da im je ime Ivan, da im je vrsta 1 i da im je identifikator veći od 100).

```
SELECT *
FROM Polaznik
WHERE Ime = 'Ivan' AND TipId = 1 AND Id > 100
```

Logički operator **OR** spaja dva ili više uvjeta na način da je rezultat istina ako je jedan od uvjeta istinit (dok ostali ne moraju, ali i mogu biti istiniti).

```
SELECT *
FROM Tablica
WHERE Uvjet1 OR Uvjet2 OR Uvjet3
```

Sljedeći upit vratit će sve one polaznike čije je ime Ivan, ali i sve one kod kojih je vrsta (stupac *TipId*) jednaka 1.

```
SELECT *
FROM Polaznik
WHERE Ime = 'Ivan' OR TipId = 1
```

Ako se pomoću operatora **OR** spoji tri ili više izraza, rezultat će biti istina već ako je samo jedan od njih istinit. Sljedeći upit vratit će sve one polaznike koji zadovoljavaju bar jedan od ova tri uvjeta (ako im je ime Ivan, ili ako im je vrsta 1 ili ako im je identifikator veći od 100).

```
SELECT *
FROM Polaznik
WHERE Ime = 'Ivan' OR TipId = 1 OR Id > 100
```

Logički operator **NOT** služi za negaciju uvjeta.

```
SELECT *
FROM Polaznik
WHERE NOT Ime = 'Ivan'
```

Prethodni upit vraća sve one polaznike čije ime nije Ivan te je zapravo jednak ovom upitu (napisanom korištenjem operatora uspoređivanja `<>`):

```
SELECT *
FROM Polaznik
WHERE Ime <> 'Ivan'
```

Logički operatori AND, OR i NOT mogu se i međusobno kombinirati.

```
SELECT *
FROM Polaznik
WHERE NOT Ime = 'Ivan' AND TipId = 1 OR Id > 100
```

Redoslijed izvođenja operacija određen je prioritetom operatora.

Operator NOT ima najviši prioritet, zatim AND i na kraju OR.

Da bi se kombinacijom operatora dobio željeni rezultat, najbolje je uvijek precizirati redoslijed izvođenja operacija zagradama.

U sljedećem će primjeru zgrade promijeniti dobiveni rezultat.

```
SELECT *
FROM Polaznik
WHERE NOT (Ime = 'Ivan' AND (TipId = 1 OR Id >
100))
```

3.6. Operator IN

Operator IN služi za provjeru toga nalazi li se neka vrijednost unutar zadanog skupa vrijednosti.

Sljedeći upit vratit će sve polaznike čija je vrsta (stupac *TipId*) jednaka 1, 2 ili 3.

```
SELECT *
FROM Polaznik
WHERE TipId IN (1, 2, 3)
```

Prethodni upit zapravo je ekvivalentan ovome:

```
SELECT *
FROM Polaznik
WHERE TipId = 1 OR TipId = 2 OR TipId = 3
```

Operator IN može se koristiti i nad znakovnim nizovima. Sljedeći upit vratit će polaznike čije je ime Ivo, Ivica ili Ivan:

```
SELECT *
FROM Polaznik
WHERE Ime IN ('Ivo', 'Ivica', 'Ivan')
```

Dodavanjem modifikatora NOT može se obaviti provjera toga da neka vrijednost **nije** unutar zadanog skupa vrijednosti.

Sljedeći upit vratit će sve polaznike čija vrsta nije jednaka 1 niti 2:

```
SELECT *
FROM Polaznik
WHERE TipId NOT IN (1, 2)
```

3.7. Operator BETWEEN

Operator `BETWEEN` služi za provjeru toga nalazi li se zadana vrijednost u određenom rasponu vrijednosti. Koristi se u kombinaciji s ključnom riječju `AND` na sljedeći način:

```
SELECT *  
FROM Tablica  
WHERE Stupac BETWEEN Pocetak AND Kraj
```

Sljedeći upit vratit će sve polaznike za koje je vrijednost stupca `Id` između 100 i 200:

```
SELECT *  
FROM Polaznik  
WHERE Id BETWEEN 100 AND 200
```

U rezultatu će se naći i polaznik kojem je `Id` jednak 100, kao i polaznik s identifikatorom 200.

Operator `BETWEEN` zapravo služi tome da bi se ovakav upit napisao jednostavnije:

```
SELECT *  
FROM Polaznik  
WHERE Id >= 100 AND Id <= 200
```

Operator `BETWEEN` može se upotrebljavati i s drugim tipovima podataka. Sljedeći upit vratit će polaznike čiji je datum rođenja između 1.1.1990 i 31.12.1995:

```
SELECT *  
FROM Polaznik  
WHERE DatumRodjenja BETWEEN '1990-01-01' AND  
'1995-12-31'
```

Dodavanjem modifikatora `NOT` može se obaviti provjera toga da neka vrijednost **nije** unutar zadanog raspona vrijednosti.

Sljedeći upit vratit će sve polaznike za koje je vrijednost stupca `Id` nije između 100 i 200:

```
SELECT *  
FROM Polaznik  
WHERE Id NOT BETWEEN 100 AND 200
```

3.8. Naredba SELECT INTO

Pomoću naredbe `SELECT INTO` moguće je stvoriti novu tablicu koja će biti napunjena podacima koji su rezultat nekog upita. Nova tablica imat će točno one stupce koji su definirani upitom.

Sintaksa naredbe `SELECT INTO` izgleda ovako:

```
SELECT Stupac1, Stupac2, Stupac3  
INTO NovaTablica  
FROM Tablica  
WHERE Uvjet
```

Sljedeći upit stvorit će novu tablicu *Polaznik2000* u kojoj će se nalaziti samo oni polaznici koji su rođeni 2000. godine:

```
SELECT Ime, Prezime, DatumRodjenja  
INTO Polaznik2000  
FROM Polaznik  
WHERE DatumRodjenja BETWEEN '2000-01-01' AND  
'2000-12-31'
```


4. Unos, izmjena i brisanje redaka

Po završetku ovog poglavlja polaznik će moći:

- unijeti retke u tablicu pomoću naredbe `INSERT`
- izmijeniti postojeće retke u tablici pomoću naredbe `UPDATE`
- obrisati retke iz tablice pomoću naredbe `DELETE`.

U jeziku SQL postoje četiri glavne naredbe – `SELECT`, `INSERT`, `UPDATE` i `DELETE`. Veći dio tečaja bavi se naredbom `SELECT`, a u ovom će se poglavlju polaznik upoznati s preostale tri.

Naredba `INSERT` služi za unos redaka, naredba `UPDATE` za izmjenu i naredba `DELETE` za brisanje redaka iz tablice.

4.1. Naredba `INSERT`

Za unos novog retka u tablicu koristi se naredba `INSERT`. Potrebno je navesti naziv tablice u koju se redak unosi, zatim stupce tablice te nakon ključne riječi `VALUES` odgovarajuće vrijednosti i to istim redoslijedom kojim su navedeni stupci.

```
INSERT INTO Tablica
(Stupac1, Stupac2, Stupac3...)
VALUES
(Vrijednost1, Vrijednost2, Vrijednost3...)
```

Sljedeća naredba unijet će novi redak u tablicu *Polaznik*:

```
INSERT INTO Polaznik
(Ime, Prezime, OIB, Spol, DatumRodjenja, Adresa,
Email, GradId, TipId, UstanovaId)
VALUES
('Ivan', 'Ivančić', 'M', '18965408575', '1990-1-
15', 'Ulica profesora Baltazara 16',
'ivica@mail.com', 1, 2, 10)
```

Kako je stupac *Id* definiran kao `IDENTITY` koji se automatski povećava, njegova se vrijednost ne navodi u naredbi `INSERT`, već će ona biti automatski postavljena.

Stupci koji imaju definirane predodređene vrijednosti ili smiju sadržavati vrijednost `NULL` mogu se također izostaviti iz naredbe `INSERT`. U tom će slučaju novi redak u tim stupcima dobiti predodređene vrijednosti, odnosno vrijednost `NULL`.

Sljedeća naredba unijet će novi redak u tablicu *Polaznik*, ali će biti postavljene vrijednosti samo za obavezne stupce (one koji su definirani kao `NOT NULL`):

```
INSERT INTO Polaznik
(Ime, Prezime, Spol, TipId)
VALUES
('Tomislav', 'Matešić', 'M', 3)
```

Ako se u naredbi `INSERT` postavljaju vrijednosti za sve stupce, onda se može izostaviti navođenje samih stupaca. Redoslijed navođenja

vrijednosti treba slijediti redoslijed kojim su stupci navedeni pri stvaranju tablice.

Kad na tablici *Polaznik* stupac *Id* ne bi bio definiran kao `IDENTITY`, bila bi moguća ovakva varijanta naredbe `INSERT`, uz izostavljanje naziva stupaca:

```
INSERT INTO Polaznik
VALUES
(100, 'Ivan', 'Ivančić', 'M', '18965408575',
'1990-1-15', 'Ulica profesora Baltazara 16',
'ivica@mail.com', 1, 2, 10)
```

Pomoću naredbe `INSERT` moguće je odjednom unijeti više redaka:

```
INSERT INTO Polaznik
(Ime, Prezime, Spol, TipId)
VALUES
('Tomislav', 'Matešić', 'M', 2),
('Marija', 'Žuborić', 'F', 1),
('Egon', 'Bistrić', 'M', 3)
```

Kombinacijom naredbi `INSERT` i `SELECT` moguće je retke koji su rezultat nekog upita unijeti u tablicu.

```
INSERT INTO Tablica
(Stupac1, Stupac2, Stupac3...)
SELECT
Stupac1, Stupac2, Stupac3...
FROM Tablica2
```

Sljedeći upit unijet će prvih 5 redaka iz tablice *Predavac* u tablicu *Polaznik*:

```
INSERT INTO Polaznik
(Ime, Prezime, Spol, TipId)
SELECT TOP 5 Ime, Prezime, Spol, 1
FROM Predavac
```

4.2. Naredba UPDATE

Pomoću naredbe `UPDATE` ažuriraju se vrijednosti u jednom ili više redaka. Potrebno je navesti naziv tablice, te, poslije ključne riječi `SET`, stupce koji se ažuriraju u paru s novim vrijednostima.

```
UPDATE Tablica
SET Stupac1 = Vrijednost1,
     Stupac2 = Vrijednost2,
     Stupac3 = Vrijednost3
WHERE Uvjet
```

Klauzula `WHERE` u naredbi `UPDATE` opcionalna je. Pomoću nje ograničava se djelovanje naredbe `UPDATE` samo na one retke koji zadovoljavaju navedeni uvjet.

Ako se klauzula `WHERE` izostavi bit će ažurirani svi retci u tablici.

Zanimljivosti i napomene

Na starijim verzijama Microsoft SQL servera (2005 i 2000), kao i na nekim drugim sustavima, nije moguće unijeti više redaka pomoću jedne naredbe `INSERT`.

Sljedeća naredba ažurirat će stupac *Tip* na svim retcima u tablici *Polaznik*:

```
UPDATE Polaznik
SET TipId = 2
```

Sljedeća naredba ažurirat će stupce *TipId* i *DatumRodjenja* na svim retcima u tablici *Polaznik*:

```
UPDATE Polaznik
SET TipId = 2,
    DatumRodjenja = '1992-12-02'
```

Sljedeća naredba ažurirat će stupac *TipId* samo retku koji zadovoljava uvjet da mu je vrijednost stupca *Id* jednaka 100:

```
UPDATE Polaznik
SET TipId = 1
WHERE Id = 100
```

4.3. Naredba DELETE

Naredba `DELETE` služi za brisanje jednog ili više redaka iz tablice. Dovoljno je navesti samo naziv tablice. Klauzula `WHERE` u naredbi `DELETE` opcionalna je. Pomoću nje se ograničava djelovanje naredbe `DELETE` samo na one retke koji zadovoljavaju navedeni uvjet.

```
DELETE FROM Tablica
WHERE Uvjet
```

Sljedeća naredba obrisat će sve podatke iz tablice *Polaznik*:

```
DELETE FROM Polaznik
```

Sljedeća naredba obrisat će samo jedan redak iz tablice *Polaznik*, i to onaj redak kod kojega je vrijednost stupca *Id* jednaka 100:

```
DELETE FROM Polaznik
WHERE Id = 100
```

Zanimljivosti i napomene

Prilikom izvođenja naredbe `UPDATE` na bazi uvijek treba paziti na to da se ne izostavi `WHERE` dio naredbe, ako je potreban. Promjene koje su jednom napravljene na bazi ne mogu se poništiti.

Zanimljivosti i napomene

Na sustavu Microsoft SQL Server moguće je izostaviti ključnu riječ `FROM` iz naredbe `DELETE`.

Zanimljivosti i napomene

Kao i kod naredbe `UPDATE`, i kod izvođenja naredbe `DELETE` uvijek treba dobro paziti na to da se ne izostavi klauzula `WHERE`. Jednom izbrisane retke ne možemo više vratiti.

5. Pomoćne funkcije

Po završetku ovog poglavlja polaznik će moći:

- raditi sa znakovnim nizovima – uklanjati praznine, izdvajati podnizove i sl.
- pozivati matematičke funkcije u radu s numeričkim podacima
- izdvajati pojedine komponente iz datuma te računati s datumskim vrijednostima
- pretvoriti podatak iz jednog tipa u drugi.

U jeziku SQL moguće je koristiti razne pomoćne funkcije. Najčešće korištene funkcije su one koje služe za rad sa specifičnim tipovima podataka – znakovnim nizovima, numeričkim i datumskim podacima te na kraju za pretvorbu između različitih tipova podataka.

Neke od navedenih funkcija nisu dio SQL standarda, već se razlikuju od sustava od sustava.

5.1. Rad sa znakovnim nizovima

U radu s podacima često se pojavljuje potreba za manipulacijom sa znakovnim nizovima, radi pročišćavanja ili preoblikovanja podataka. U nastavku se nalazi pregled najčešće korištenih funkcija za rad sa znakovnim nizovima.

Pri unosu podataka može se dogoditi da korisnik nehotice unese i razmak. Kako bi se uklonio razmak i druge praznine s početka znakovnog niza, koristi se funkcija `LTRIM`. Ova funkcija uklanja praznine s lijeve strane niza (otuda L u nazivu funkcije). Funkcija kao argument prima znakovni niz koji se želi očistiti:

```
SELECT LTRIM(Ime)
FROM Polaznik
```

Za uklanjanje praznina s kraja niza, tj. s desne strane, koristi se funkcija `RTRIM`:

```
SELECT RTRIM(Ime)
FROM Polaznik
```

Kako bi se odjednom uklonile praznine i s lijeve i s desne strane znakovnog niza, može se prvo pozvati funkciju `LTRIM`, a zatim rezultat predati funkciji `RTRIM`:

```
SELECT RTRIM(LTRIM(Ime) )
FROM Polaznik
```

Ponekad je potrebno izdvojiti određene znakove iz znakovnog niza. Pomoću funkcije `LEFT` može se izdvojiti početni (lijevi) dio niza.

```
LEFT (ulaz, duljina)
```

Kao parametri se predaju početni znakovni niz te broj znakova koji se želi izdvojiti.

Zanimljivosti i napomene

Neke od navedenih funkcija nisu dostupne na svim sustavima za upravljanje bazama podataka.

U sljedećem upitu dohvaćamo prva tri znaka iz imena polaznika:

```
SELECT LEFT(Ime, 3)
FROM Polaznik
```

Ako je potrebno izdvojiti desni dio niza, odnosno posljednjih nekoliko znakova iz niza, može se koristiti funkcija `RIGHT`.

```
RIGHT (ulaz, duljina)
```

Kao argumenti se također predaju početni znakovni niz te broj znakova koji se želi izdvojiti. U sljedećem upitu dohvaćamo posljednja tri znaka iz imena polaznika:

```
SELECT RIGHT(Ime, 3)
FROM Polaznik
```

Funkcija `SUBSTRING` nudi dodatne mogućnosti za izdvajanje podniza iz niza znakova.

```
SUBSTRING (ulaz, početak, duljina)
```

Kao prvi argument predaje se početni niz, kao drugi argument pozicija u početnom nizu na kojoj počinje željeni podniz, a posljednji duljina podniza. Sljedeći upit kreće od pozicije broj 1 u početnom nizu i uzima 3 znaka.

```
SELECT SUBSTRING(Ime, 1, 3)
FROM Polaznik
```

Ovaj upit, dakle, izdvaja prva tri znaka iz stupca `Ime` te je ekvivalentan pozivu funkcije `LEFT(Ime, 3)`.

No, ako bi se željelo da podniz počne na nekoj drugoj poziciji (npr. na poziciji 2), to se ne bi moglo postići pomoću funkcije `LEFT`, već je nužno koristiti `SUBSTRING`:

```
SELECT SUBSTRING(Ime, 2, 3)
FROM Polaznik
```

Funkcija `LEN` vratit će duljinu predanog znakovnog niza:

```
SELECT LEN(Ime)
FROM Polaznik
```

Pomoću funkcije `REPLACE` može se napraviti izmjena dijela znakova unutar niza.

```
REPLACE (ulaz, trazeniNiz, noviNiz)
```

Kao prvi parametar funkcija prima ulazni niz, kao drugi niz znakova koji se traži, i kao treći novi niz kojim treba zamijeniti pronađeni.

```
SELECT REPLACE(Ime, 'a', 'aa')
FROM Polaznik
```

Gornji upit vratit će stupac `Ime` u kojem je znak 'a' zamijenjen s 'aa'.

Funkcija `UPPER` vraća ulazni niz ispisan svim velikim slovima.

```
SELECT UPPER(Ime)
FROM Polaznik
```

Zanimljivosti i napomene

Neke od navedenih funkcija nisu dostupne na svim sustavima za upravljanje bazama podataka.

Slična funkcija `LOWER` vraća ulazni niz ispisan svim malim slovima.

```
SELECT LOWER (Ime)
FROM Polaznik
```

5.2. Rad s numeričkim podacima

Osim osnovnih aritmetičkih operacija, prilikom rada s numeričkim podacima ponekad su potrebne i složenije operacije koje možemo izvesti pomoću sljedećih funkcija.

Sve navedene funkcije primaju numeričke vrijednosti.

Funkcija `SQUARE` vraća kvadrat ulazne vrijednosti.

```
SELECT SQUARE (4)
```

Funkcija `SQRT` vraća kvadratni korijen ulazne vrijednosti.

```
SELECT SQRT (16)
```

Kada je potrebno zaokružiti neku vrijednost, na raspolaganju je nekoliko funkcija. Funkcija `CEILING` zaokružuje decimalnu vrijednost na prvu višu cjelobrojnu vrijednost:

```
SELECT CEILING (2.45)
```

Funkcija `FLOOR` zaokružuje decimalnu vrijednost na prvu nižu cjelobrojnu vrijednost:

```
SELECT FLOOR (2.45)
```

Funkcija `ROUND` obavlja zaokruživanje vrijednosti u užem smislu riječi – ulazna vrijednost zaokružuje se na dolje ili na gore ovisno o tome kojoj je vrijednosti bliža. Funkcija prima ulaznu vrijednost i broj znamenaka na koji se želi zaokružiti:

```
ROUND (vrijednost, brojZnamenaka)
```

U sljedećem primjeru vrijednost će biti zaokružena na dvije decimale:

```
SELECT ROUND (12.359, 2)
```

Ako je drugi parametar (broj znamenaka) negativan, zaokruživanje se vrši s lijeve strane decimalne točke (tako da se zaokruži toliko znamenaka slijeva):

```
SELECT ROUND (245, -2)
```

Od korisnih numeričkih funkcija potrebno je spomenuti još i funkciju `RAND` koja vraća nasumičnu vrijednost između 0 i 1.

```
SELECT RAND ()
```

5.3. Rad s datumskim podacima

U radu s datumskim podacima često nam je potrebna pomoć neke od specijaliziranih datumskih funkcija. Pregled najčešće korištenih datumskih funkcija nalazi se u nastavku.

Trenutni datum i vrijeme može se dobiti pomoću funkcije `GETDATE`:

```
SELECT GETDATE ()
```

Zanimljivosti i napomene

U sljedećim primjerima koristi se `SELECT` naredba bez klauzule `FROM`, dakle podaci se ne dohvaćaju iz tablice, već se radi s konstantnim vrijednostima.

Rezultat ovakvog upita uvijek će biti samo jedan redak.

Iz datumske vrijednosti često je potrebno izdvojiti pojedinu komponentu (npr. godinu, mjesec ili dan).

Kako bi se iz datumske vrijednosti izdvojila godina, koristi se funkcija YEAR:

```
SELECT YEAR('2015-09-25')
```

Za izdvajanje mjeseca iz datuma koristi se funkcija MONTH:

```
SELECT MONTH('2015-09-25')
```

Kako bi se iz datumske vrijednosti dobio dan u mjesecu, koristi se funkcija DAY:

```
SELECT DAY('2015-09-25')
```

Funkcija DATEPART nudi više mogućnosti za izdvajanje pojedinog dijela iz datuma i vremena. Kao prvi argument predaje se naziv komponente koja će se izdvojiti iz datumske vrijednosti:

DATEPART (*komponenta*, *datum*)

Na ovaj bismo način izdvojili godinu iz datumske vrijednosti korištenjem funkcije DATEPART:

```
SELECT DATEPART(year, '2015-09-25')
```

Umjesto punog naziva moguće je koristiti i skraćenicu, a sve vrijednosti koje se može koristiti nalaze su u sljedećoj tablici:

Datumska komponenta	Skraćenica	Objašnjenje
year	yy ili yyyy	godina
quarter	qq ili qqq	kvartal
month	mm ili m	mjesec
dayofyear	dy ili y	dan u godini
day	dd ili d	dan u mjesecu
week	wk ili ww	tjedan u godini
weekday	dw	dan u tjednu
hour	hh	sat
minute	mi ili n	minuta
second	ss ili s	sekunda
milisecond	ms	milisekunda
microsecond	mcs	mikrosekunda
nanosecond	ns	nanosekunda

Funkcija DATENAME slična je funkciji DATEPART, s tim da, umjesto broja, vraća znakovni niz koji predstavlja određeni dio datuma. Ova je funkcija najkorisnija kad se radi o nazivu mjeseca ili dana u tjednu.

DATENAME (*komponenta*, *datum*)

Na ovaj bi se način dobio naziv mjeseca za zadani datum:

```
SELECT DATENAME(month, '2015-09-25')
```

Kako bi se uvećalo neku datumsku vrijednost za određeni iznos, može se koristiti funkcija `DATEADD`. I kod ove funkcije potrebno je navesti na koju komponentu se taj iznos odnosi:

```
DATEADD(komponenta, iznos, datum)
```

Na sljedeći način bi se zadani datum uvećao za jedan mjesec:

```
SELECT DATEADD(month, 1, '2015-09-25')
```

Funkcija `DATEDIFF` koristi se kako bi se dobila razlika između dva datuma. I kod ove je funkcije potrebno navesti na koji način želimo izraziti tu razliku:

```
DATEDIFF(komponenta, početniDatum, krajnjiDatum)
```

Sljedeći će upit odgovoriti na to koliko je proteklo dana između ova dva datuma:

```
SELECT DATEDIFF(day, '2015-09-25', '2015-10-25')
```

5.4. Pretvorba između tipova podataka

Kad je potrebno pretvoriti podatak iz jednog tipa u drugi koristi se funkcija `CAST`.

```
CAST(vrijednost AS tipPodatka)
```

Na sljedeći način pretvorili bismo znakovni niz '010' u broj 10:

```
SELECT CAST('010' AS integer)
```

Ova se funkcija često koristi kada se želi spojiti znakovni niz s nekom brojčanom vrijednošću. U sljedećem primjeru želi se ispisati identifikator, ime i prezime polaznika spojeni u istoj koloni.

```
SELECT Id + ' ' + Ime + ' ' + Prezime
FROM Polaznik
```

No, stupac `Id` je tipa `integer`, pa gornji upit uzrokuje grešku. Prije spajanja sa znakovnim nizovima, cjelobrojnu vrijednost moramo pretvoriti u znakovnu pomoću funkcije `CAST`:

```
SELECT CAST(Id AS varchar)
      + ' ' + Ime + ' ' + Prezime
FROM Polaznik
```

5.5. Vježba: Postavljanje uvjeta, unos, izmjena i brisanje redaka, rad s pomoćnim funkcijama

1. Prikažite sve polaznike koji se zovu Tihomir.
2. Prikažite sve polaznike koji stanuju u Zagrebu. Provjerite koja je odgovarajuća vrijednost za stupac *GradId* tako da provjerite vrijednost stupca *Id* u tablici *Grad* gdje je naziv grada „Zagreb“.
3. Prikažite sve polaznike koji stanuju u Ulici grada Vukovara. Obratite pažnju na različite mogućnosti pisanja naziva ulice – *Ulica grada Vukovara*, *Vukovarska*...
4. Prikažite sve polaznike koji stanuju u Ulici grada Vukovara, ali samo u Zagrebu.
5. Prikažite sve polaznike koji stanuju u Samoboru ili u Zaprešiću.
6. Prikažite sve polaznike koji imaju e-mail adresu s *.hr* domenom.
7. Dodajte provjeru da polaznici prikazani u prethodnom zadatku nemaju adresu s *.t-com.hr* domenom.
8. Prikažite sve polaznike koji su rođeni 1995. godine. Zadatak riješite na ova tri načina: pomoću operatora *AND*, pomoću operatora *BETWEEN* i pomoću funkcije *YEAR*.
9. Prikažite sve polaznike kod kojih nije postavljena vrijednost za polje *Email*.
10. U tablicu *Polaznik* unesite vlastite podatke pomoću naredbe *INSERT*. Postavite odgovarajuće vrijednosti za stupce *GradId*, *Tipld* i *Ustanovld*. (Provjerite koje su odgovarajuće vrijednosti u tablicama *Grad*, *Tip* i *Ustanova*).
11. Prikažite ime, prezime i inicijale za osobe iz tablice *Polaznik*.
12. Prikažite ime, prezime i generirano korisničko ime za osobe iz tablice *Polaznik*. Korisničko ime treba se sastojati od prvog slova imena spojenog s prezimenom. Sva slova u korisničkom imenu trebaju biti mala. Hrvatske znakove koji se pojavljuju treba zamijeniti s odgovarajućim znakovima bez dijakritika.
13. Za sve retke u tablici *Polaznik* koji nemaju postavljeno polje *Email*, postavite ga pomoću naredbe *UPDATE* na generirano korisničko ime iz prethodnog koraka kojem ćete dodati „@srce.hr“.
14. Uz svakog polaznika ispišite njegovu godinu rođenja.
15. Uz svakog polaznika ispišite njegovu dob u godinama. Broj godina proteklih od danas do datuma rođenja izračunajte pomoću funkcije *DATEDIFF*.

6. Strani ključ i tipovi veza između tablica

Po završetku ovog poglavlja polaznik će moći:

- objasniti što je strani ključ i koji su razlozi njegovog korištenja
- napraviti vezu između dvije tablice preko stranog ključa
- objasniti što je i kada se upotrebljava veza jedan naprama više
- objasniti što je i kada se upotrebljava veza više naprama više
- prepoznati tip veze između entiteta
- ostvariti vezu više naprama više pomoću vezne tablice
- izraditi i interpretirati dijagram baze podataka.

Veze između tablica ostvaruju se preko stranog ključa. U ovom poglavlju objašnjava se što je strani ključ i koji su mogući tipovi veza između tablica.

6.1. Strani ključ

Svi pripadajući podaci koji opisuju neki objekt iz stvarnog svijeta neće se nalaziti nužno u samo jednoj tablici. Između tablica u bazi podataka postoji veza kada one sadrže međusobno povezane podatke.

U primjeru baze podataka o tečajevima, podaci o predavačima nalaze se u tablici *Predavac*. Podaci o odjelu u kojem je predavač zaposlen nalaze se u tablici *Odjel*. Vidi se da između ove dvije tablice postoji veza.

U tablici *Predavac* nalazi se stupac *OdjelId*. Vrijednost u ovom stupcu zapravo predstavlja vrijednost stupca *Id* iz tablice *Odjel*.

U slučaju kad u jednoj tablici postoji stupac koji je identifikator, odnosno ključ u drugoj tablici i na taj način referencira podatke iz druge tablice, taj stupac se naziva stranim ključem.

Preporučljivo je eksplicitno definirati stupac kao strani ključ prilikom stvaranja tablice.

Prilikom stvaranja tablice *Predavac* na ovaj bi se način definiralo da je kolona *OdjelId* strani ključ koji se odnosi na stupac *Id* iz tablice *Odjel*:

```
CREATE TABLE Predavac (
    Id int NOT NULL PRIMARY KEY IDENTITY(1,1),
    OdjelId int FOREIGN KEY REFERENCES Odjel(Id),
    Ime nvarchar(50) NOT NULL,
    Prezime nvarchar(50) NOT NULL
)
```

Prednosti eksplicitnog navođenja stranog ključa višestruke su. Prvo, jasna je svrha stupca *OdjelId* i ne može doći do nedoumice na koju se tablicu on odnosi. Zatim, pri unosu ili ažuriranju redaka u tablici *Predavac* nije moguće postaviti vrijednost stupca *OdjelId* na vrijednost koja ne postoji u tablici *Odjel*. Na kraju, nije moguće obrisati redak iz tablice *Odjel* ako se ta vrijednost koristi u tablici *Predavac*.

Zanimljivosti i napomene

Kao što se i primarni ključ može sastojati od više stupaca, tako se i strani ključ može sastojati od više stupaca.

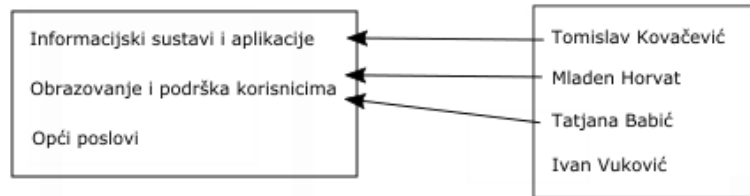
Međutim, u praksi se takvi strani ključevi rjeđe koriste, jer je rad s njima kompliciraniji.

6.2. Veza jedan naprama više

Glavni podatak koji opisuje vezu između tablica jest **brojnost veze**. Brojnost veze može se utvrditi navođenjem glagola koji opisuju vezu između pripadajućih entiteta u stvarnome svijetu. Naprimjer:

Jedan odjel zapošljava više predavača.

Jedan predavač zaposlen je u jednom odjelu.

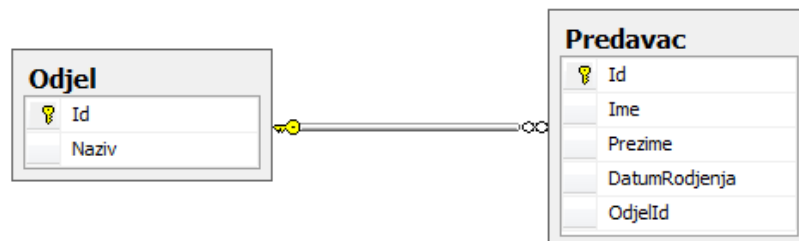


Na temelju ove dvije rečenice zaključuje se da je veza između tablica *Odjel* i *Predavac* veza **jedan naprama više**. To znači da za svaki redak u tablici *Odjel* možemo imati više redaka u tablici *Predavac*, tj. više predavača koji su zaposleni u jednom odjelu.

Veza jedan naprama više naziva se još i vezom jedan naprama n (ponekad i 1 naprama m). U dijagramima koji opisuju baze podataka brojnost veze se prikazuje oznakama 1 i N :



Veza jedan naprama više ostvaruje se pomoću stranog ključa. Stupac *OdjelId* je strani ključ u tablici *Predavac*, a odnosi se na podatak *Id* iz tablice *Odjel*.



U dijagramu baze koji generira *Microsoft SQL Server Management Studio* simbol ključa nalazi se na strani brojnosti *jedan* (odnosno na strani tablice u kojoj je to primarni ključ), dok simbol beskonačnosti predstavlja brojnost *više*.

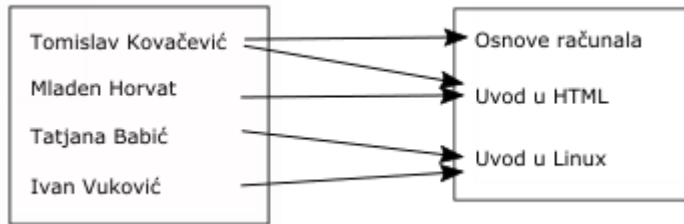
6.3. Veza više naprama više

Drugi osnovni tip veze koji može postojati u bazama podataka jest veza **više naprama više**.

U primjeru baze podataka o tečajevima, ovakva veza postoji između tablica *Predavac* i *Tecaj*. Isti predavač može držati više različitih tečajeva. S druge strane, isti tečaj mogu održavati različiti predavači (u različitim terminima).

Zanimljivosti i napomene

Veza jedan naprama više može se kraće zapisati kao 1:n (ponekad i 1:m)



Jedan predavač održava više tečajeva.

Jedan tečaj održava se od strane više predavača.

Iz navedenih rečenica vidi se da je brojnost veze više naprama više.

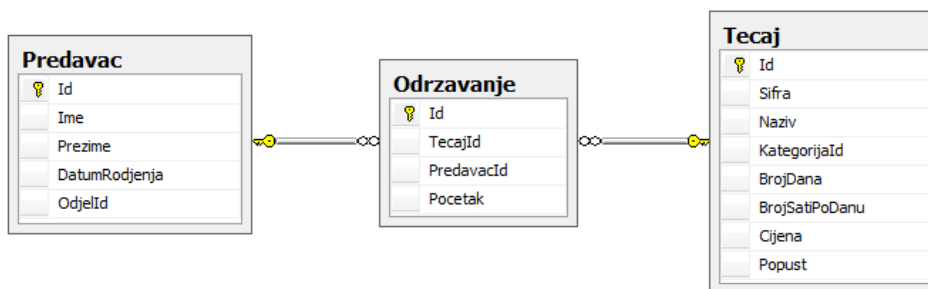
Ova veza naziva se još i vezom n naprama n (ponekad i m naprama n).



Zanimljivosti i napomene

Veza jedan naprama više može se kraće zapisati kao $n:n$ (ponekad i $m:n$).

Veza više naprama više ostvaruje se pomoću vezne tablice. Veza između tablica *Predavac* i *Tecaj* ostvarena je pomoću vezne tablice *Odrzavanje*.



Svaki redak u tablici *Odrzavanje* sadrži identifikator predavača koji je držao tečaj (u stupcu *PredavacId*) i identifikator tečaja koji je održan (*TecajId*).

Ova dva stupca su ujedno strani ključevi koji se referenciraju na tablice *Predavac* i *Tecaj*.

Osim ova dva stupca, tablica *Odrzavanje* može sadržavati i druge podatke (npr. datum početka održavanja tečaja).

Primarni ključ tablice *Odrzavanje* može biti kombinacija stupaca *PredavacId* i *TecajId* (kompozitni ključ). Ako se iz praktičnih razloga želi izbjeći komplikacije koje nosi upotreba kompozitnih ključeva, može se tablici *Odrzavanje* dodati stupac *Id* koji će biti primarni ključ te tablice.

Za veznu tablicu karakteristično je to da redak u njoj ne može postojati samostalno, odnosno ne može postojati bez odgovarajućeg retka u tablici *Predavac* i odgovarajućeg retka u tablici *Tecaj*.

6.4. Veza jedan naprama jedan

Uz ova dva osnovna tipa veze ponekad se može susresti i veza jedan naprama jedan. Za svaki redak iz prve tablice postojat će samo jedan redak iz druge tablice, i obrnuto.

Stupac preko kojeg se veza ostvaruje će u prvoj tablici biti strani ključ, a u drugoj tablici ujedno i strani i primarni ključ.

Primjer ovakve veze bi bio kad bi se željelo izdvojiti podatak o adresi u zasebnu tablicu. Za svakog polaznika bi postojala samo jedna adresa, i jedna adresa bi pripadala samo jednom polazniku.

Ovakva veza je potrebna samo u specijalnim slučajevima, jer je jednostavnije držati sve stupce koji se odnose na jedan entitet u istoj tablici.

6.5. Dijagram baze podataka

Dijagram baze podataka prikazuje tablice u bazi, veze između njih i njihovu brojnost. Uz svaku tablicu mogu biti prikazani stupci te označeni primarni i strani ključevi.

Microsoft SQL Server Management Studio nudi mogućnost automatskog izrade dijagrama baze podataka (opcija *Database Diagrams > New Database Diagram*). U dijagramu je moguće prikazati sve ili samo neke tablice iz baze podataka.

U Dodatku (poglavlje 14.2) je prikazan dijagram baze podataka o tečajevima izrađen pomoću aplikacije *Microsoft SQL Server Management Studio*.

7. Spajanje tablica

Po završetku ovog poglavlja polaznik će moći:

- razumjeti kako funkcionira spajanje tablica
- napraviti unutarnji spoj između tablica
- koristiti način spajanja pomoću klauzula `JOIN` i `ON` i način spajanja pomoću klauzule `WHERE`
- koristiti aliase prilikom spajanja tablica
- napraviti vanjski (lijevi, desni ili puni) spoj između tablica
- napraviti spoj tablice same sa sobom
- napraviti spoj između više od dvije tablice
- napraviti spoj po nejednakosti.

Kako bi se dobio željeni prikaz podataka, najčešće je potrebno dohvatiti podatke iz nekoliko tablica. Da bi se dohvatili podaci iz više tablica, potrebno je napraviti **spoj** (*join*) tablica. U ovom poglavlju bit će objašnjeni načini na koji je moguće spojiti dvije ili više tablice.

7.1. Kartezijev produkt tablica

Najjednostavniji primjer spoja tablica je Kartezijev produkt. Rezultat Kartezijevog produkta dviju tablica jest kombinacija svih redaka iz prve tablice sa svim retcima iz druge tablice.

Da bi se dobio Kartezijev produkt, u klauzuli `FROM` navode se imena obje tablice (odvojene zarezima).

```
SELECT *
FROM Tablica1, Tablica2
```

Neka tablica *Odjel* sadrži sljedeće retke:

Id	Naziv
1	Informacijski sustavi i aplikacije
2	Obrazovanje i podrška korisnicima
3	Opći poslovi

Neka tablica *Predavac* sadrži sljedeće retke:

Ime	Prezime	OdjelId
Mladen	Horvat	2
Tomislav	Kovačević	1
Tatjana	Babić	2
Ivan	Vuković	NULL

Zanimljivosti i napomene

Naziv Kartezijev produkt dolazi od latinskog imena matematičara Rene Descartesa, koji je definirao operaciju Kartezijevog produkta između dvaju skupova.

U bazama podataka Kartezijev produkt između tablica poznat je i kao *cross join*.

Sljedeći upit daje kao rezultat Kartezijev produkt tablica *Predavac* i *Odjel*:

```
SELECT *
FROM Predavac, Odjel
```

Rezultat ovog upita, ako tablice *Predavac* i *Odjel* imaju sadržaj kao u prethodnim primjerima, izgledat će ovako:

Ime	Prezime	OdjelId	Id	Naziv
Mladen	Horvat	2	1	Informacijski sustavi i aplikacije
Tomislav	Kovačević	1	1	Informacijski sustavi i aplikacije
Tatjana	Babić	2	1	Informacijski sustavi i aplikacije
Ivan	Vuković	NULL	1	Informacijski sustavi i aplikacije
Mladen	Horvat	2	2	Obrazovanje i podrška korisnicima
Tomislav	Kovačević	1	2	Obrazovanje i podrška korisnicima
Tatjana	Babić	2	2	Obrazovanje i podrška korisnicima
Ivan	Vuković	NULL	2	Obrazovanje i podrška korisnicima
Mladen	Horvat	2	3	Opći poslovi
Tomislav	Kovačević	1	3	Opći poslovi
Tatjana	Babić	2	3	Opći poslovi
Ivan	Vuković	NULL	3	Opći poslovi

Kao rezultat se dobiva kombinacija svih redaka iz tablice *Predavac* sa svim retcima iz tablice *Odjela*. Svaki pojedini predavač uparen je sa svim odjelima, te Kartezijev produkt ovih tablica zapravo ne nosi smislenu informaciju.

Kartezijev produkt uglavnom nema praktičnu primjenu u radu s bazama podataka. S obzirom na to da je rezultat uvijek velik broj redaka, ova operacija je dugotrajna za izvođenje u slučaju da početne tablice imaju veliki broj redaka.

7.2. Unutarnji spoj

Kako bi spoj tablica dao smislenu informaciju, potrebno je da veza između tablica bude ostvarena preko vrijednosti nekog stupca. Definiira se uvjet spajanja – da vrijednost jednog stupca iz prve tablice bude jednaka vrijednosti nekog stupca iz druge tablice. Taj će stupac najčešće biti strani ključ u jednoj od tablica.

Retci iz prve tablice uparaju se samo s onim retcima iz druge tablice za koje je taj uvjet ostvaren.

Ovakav spoj tablica naziva se spajanjem po jednakosti (*equi join*). Njegov je osnovni slučaj unutarnji spoj (tako nazvan za razliku od vanjskog spoja, koji će biti objašnjen u sljedećim poglavljima).

U klauzuli `FROM` navodi se samo prva tablica, dok se druga tablica navodi u klauzuli `JOIN`. Ključna riječ `INNER` označava da se radi o unutarnjem spoju. U klauzuli `ON` navodi se uvjet spajanja.

Zanimljivosti i napomene

Ključnu riječ `INNER` dopušteno je izostaviti. Ključna riječ `JOIN`, ako je sama, dovoljna je da bi bilo jednoznačno određeno da se radi o unutarnjem spoju.


```
SELECT *
FROM Tablica1
INNER JOIN Tablica2
ON Stupac1 = Stupac2
```

Spoj između tablica *Predavac* i *Odjel* može se napraviti preko stupca *OdjelId* u tablici *Predavac*. Ovaj je stupac ujedno strani ključ u tablici *Predavac* koji pokazuje na stupac *Id* u tablici *Odjel*.

Upit koji će napraviti spoj između tablica *Predavac* i *Odjel* izgledat će ovako:

```
SELECT *
FROM Predavac
INNER JOIN Odjel
ON Predavac.OdjelId = Odjel.Id
```

Retci iz tablice *Predavac* bit će upareni samo s odgovarajućim retkom iz tablice *Odjel* – onim odjelom u koji predavač pripada.

Ime	Prezime	OdjelId	Id	Naziv
Mladen	Horvat	2	2	Obrazovanje i podrška korisnicima
Tomislav	Kovačević	1	1	Informacijski sustavi i aplikacije
Tatjana	Babić	2	2	Obrazovanje i podrška korisnicima

Kako i tablica *Predavac* i tablica *Odjel* imaju stupac *Id*, u gornjem upitu je bilo potrebno dodatno specificirati o kojoj se tablici radi, navođenjem imena tablice (*Odjel.Id*).

Umjesto navođenja punog imena tablice, moguće je koristiti aliase. Sljedeći je upit napisan korištenjem aliasa *p* za tablicu *Predavac* te aliasa *o* za tablicu *Odjel*:

```
SELECT p.Id, p.Ime, p.Prezime, p.OdjelId, o.Naziv
FROM Predavac AS p
INNER JOIN Odjel AS o
ON p.OdjelId = o.Id
```

Spoj dviju tablica može se napisati i tako da se u klauzuli *FROM* navedu tablice koje se spajaju, a u klauzuli *WHERE* navede se uvjet spajanja. Ovaj (implicitni) način pisanja spojeva moguće je samo za unutarnji spoj.

```
SELECT *
FROM Predavac, Odjel
WHERE Predavac.OdjelId = Odjel.Id
```

Zanimljivosti i napomene

Spajanjem s drugom tablicom u rezultatu možemo dobiti više redaka nego što imamo u prvoj tablici. To ovisi o smjeru jedan naprama više veze preko koje spajamo tablice.

Zanimljivosti i napomene

Na sustavu Microsoft SQL server, kao i na nekim drugim sustavima, možemo izostaviti ključnu riječ *AS*.

Zanimljivosti i napomene

Preporučuje se upotreba načina spajanja pomoću klauzule *JOIN*. Ako se koristi klauzula *WHERE* i dodatni uvjet filtriranja, nije moguće na prvi pogled razlikovati uvjet spajanja od uvjeta filtriranja.

7.3. Lijevi spoj

Kod unutarnjeg spoja u rezultatu neće biti prisutni oni retci iz prve tablice koji se ne mogu upariti s retcima iz druge tablice, i obrnuto.

Kod vanjskog spoja i takvi će retci biti uključeni u rezultat. Vanjski spoj može biti lijevi, desni ili puni.

Kod lijevog spoja, ako u prvoj (lijevoj) tablici postoje retci koji se ne mogu upariti ni s jednim retkom iz druge tablice, takvi će retci biti uključeni u rezultat, a na mjestu stupaca iz druge tablice bit će vraćena vrijednost `NULL`.

Pomoću ključnih riječi `LEFT` i `OUTER` naznačuje se da se radi o lijevom (vanjskom) spoju.

```
SELECT *
FROM Tablica1
LEFT OUTER JOIN Tablica2
ON Stupac1 = Stupac2
```

Sljedeći upit kao rezultat daje lijevi spoj tablica *Predavac* i *Odjel*.

```
SELECT *
FROM Predavac
LEFT OUTER JOIN Odjel
ON Predavac.OdjelId = Odjel.Id
```

U rezultatu će biti prisutni i oni predavači koji ne spadaju ni u jedan odjel, odnosno kod kojih stupac *OdjelId* ima vrijednost `NULL`.

Ime	Prezime	OdjelId	Id	Naziv
Mladen	Horvat	2	2	Obrazovanje i podrška korisnicima
Tomislav	Kovačević	1	1	Informacijski sustavi i aplikacije
Tatjana	Babić	2	2	Obrazovanje i podrška korisnicima
Ivan	Vuković	NULL	NULL	NULL

Tablica navedena s lijeve strane ključne riječi `JOIN` smatra se „glavnom tablicom“, te njezini retci moraju završiti u rezultatu bez obzira na to što u drugoj tablici nema odgovarajućih redaka s kojima se oni mogu upariti.

7.4. Desni spoj

Desni spoj sličan je lijevom spoju, s tom razlikom što ulogu „glavne“ tablice ima tablica s desne strane ključne riječi `JOIN`. Svi retci iz desne tablice bit će prisutni u rezultatu, a odgovarajući stupci iz lijeve tablice za taj redak imat će vrijednost `NULL`.

Pomoću ključnih riječi `RIGHT` i `OUTER` naznačuje se da se radi o desnom (vanjskom) spoju.

```
SELECT *
FROM Tablica1
RIGHT OUTER JOIN Tablica2
ON Stupac1 = Stupac2
```

U desnom spoju između tablica *Predavac* i *Odjel*, u rezultatu će se naći svi odjeli, pa tako i oni u kojima nije zaposlen nijedan predavač.

Zanimljivosti i napomene

Ključnu riječ `OUTER` dopušteno je izostaviti. Dovoljne su ključne riječi `LEFT JOIN` koje određuju da se radi o lijevom spoju.

```
SELECT *
FROM Predavac
RIGHT OUTER JOIN Odjel
ON Predavac.OdjelId = Odjel.Id
```

Ime	Prezime	OdjelId	Id	Naziv
Mladen	Horvat	2	2	Obrazovanje i podrška korisnicima
Tomislav	Kovačević	1	1	Informacijski sustavi i aplikacije
Tatjana	Babić	2	2	Obrazovanje i podrška korisnicima
NULL	NULL	NULL	3	Opći poslovi

Desni spoj se u praksi rjeđe koristi od lijevog spoja, jer se zapravo isti rezultat može dobiti pomoću lijevog spoja (tako da se obrne redoslijed tablica).

```
SELECT *
FROM Odjel
LEFT OUTER JOIN Predavac
ON Predavac.OdjelId = Odjel.Id
```

7.5. Puni spoj

Puni spoj je također vanjski spoj. Kod punog spoja obje tablice imaju ulogu „glavne“ tablice. U rezultatu će se naći svi retci iz prve, ali i svi retci iz druge tablice. Kod redaka koji ne mogu biti upareni, na mjestu odgovarajućih stupaca bit će vrijednost `NULL`.

Pomoću ključnih riječi `FULL` i `OUTER` naznačuje se da se radi o punom (vanjskom) spoju.

```
SELECT *
FROM Tablica1
FULL OUTER JOIN Tablica2
ON Stupac1 = Stupac2
```

U punom spoju između tablica *Predavac* i *Odjel*, u rezultatu će se naći i predavači koji nisu zaposleni ni u jednom odjelu, ali i odjeli u kojima nije zaposlen ni jedan predavač.

```
SELECT *
FROM Predavac
FULL OUTER JOIN Odjel
ON Predavac.OdjelId = Odjel.Id
```

Ime	Prezime	OdjelId	Id	Naziv
Mladen	Horvat	2	2	Obrazovanje i podrška korisnicima
Tomislav	Kovačević	1	1	Informacijski sustavi i aplikacije
Tatjana	Babić	2	2	Obrazovanje i podrška korisnicima
Ivan	Vuković	NULL	NULL	NULL
NULL	NULL	NULL	3	Opći poslovi

Puni spoj se također u praksi rjeđe koristi od lijevog spoja.

7.6. Spoj tablice sa samom sobom

U nekim slučajevima može postojati samoreferencirajuća veza, odnosno tablica može imati strani ključ koji pokazuje na tu istu tablicu. Najčešće se radi o hijerarhijskoj vezi, kao što je to slučaj u tablici *Kategorija*, gdje je preko stupca *NadKategorijaId* kategorija povezana sa svojom nadređenom kategorijom.

Sljedeći upit napraviti će spoj tablice *Kategorija* same sa sobom preko kolone *NadKategorijaId*. Kako se tablica spaja sama sa sobom, nužno je koristiti aliase.

```
SELECT k.Id, k.Naziv AS Kategorija,
       k.NadKategorijaId, n.Naziv AS NadKategorija
FROM Kategorija AS k
INNER JOIN Kategorija AS n
ON k.NadKategorijaId = n.Id
```

7.7. Spoj više tablica

Da bi se došlo do željenog rezultata, često je u upitu potrebno napraviti spoj više tablica. Klauzule `JOIN` i `ON` u tom slučaju ponavljamo više puta – po jedan put svaki put za svaki spoj između tablica.

Da bi se odgovorilo na pitanje koji predavač drži koji tečaj, moramo spojiti tablice *Predavac* i *Tecaj*. No, to je moguće napraviti samo preko vezne tablice *Odrzavanje*.

```
SELECT *
FROM Predavac
INNER JOIN Odrzavanje
ON Predavac.Id = Odrzavanje.PredavacId
INNER JOIN Tecaj
ON Odrzavanje.TecajId = Tecaj.Id
```

Ako se želi u rezultatu imati i predavače koji nisu (još) održali nijedan tečaj, pa ih nema u tablici *Odrzavanje*, spoj između tablica *Predavac* i *Odrzavanje* treba biti lijevi spoj:

```
SELECT *
FROM Predavac
LEFT OUTER JOIN Odrzavanje
ON Predavac.Id = PredavacId
INNER JOIN Tecaj
ON Odrzavanje.TecajId = Tecaj.Id
```

Spoj između ove tri tablice može se promatrati i kao da se rezultat spoja tablice *Predavac* s tablicom *Odrzavanje* spaja s tablicom *Tecaj*.

7.8. Spoj po nejednakosti

Kod spoja po nejednakosti uvjet spajanja nije jednakost dvaju stupaca, već možemo koristiti druge operatore uspoređivanja.

Primjer ovakvog spoja bio bi sljedeći upit koji spaja tablice *Tecaj* i *Lokacija*. U tablici *Lokacija* nalaze se podatci o učionicama u kojima se održavaju tečajevi, a u stupcu *BrojMjesta* nalazi se podatak o tome koliko učionica može primiti polaznika. U tablici *Tecaj* u stupcu *MinimalnoPolaznika* nalazi se podatak o minimalnom broju polaznika koji je potreban da bi se tečaj održao. Ako neka učionica ima manje

Zanimljivosti i napomene

Bitno je zapamtiti da je uvijek potrebno navesti onoliko uvjeta spajanja koliko imamo spojeva između tablica.

Ako se spajaju dvije tablice, postoji jedan uvjet spajanja, a ako se spajaju tri tablice, prisutna su dva uvjeta spajanja između njih.

Kod spoja četiri tablice, broj uvjeta bio bi tri.

Zanimljivosti i napomene

Spoj po nejednakosti poznat je i pod imenom *non-equi join*.

mjesta od minimalnog broja polaznika, taj se tečaj nikada ne može održati u toj učionici.

Sljedeći upit vraća tečajeve i učionice kod kojih je to slučaj:

```
SELECT *  
FROM Teca  
INNER JOIN Lokacija  
ON Teca.MinimalnoPolaznika > Lokacija.BrojMjesta
```

7.9. Vježba: Spajanje tablica

1. Napravite Kartezijev produkt tablica *Ustanova* i *Polaznik*.
2. Napravite unutarnji spoj tablica *Ustanova* i *Polaznik*.
3. Napravite unutarnji spoj tablica *Ustanova* i *Polaznik*, ali u rezultatu prikazite samo one ustanove koje u nazivu sadrže riječ „fakultet“. Poredajte rezultat po nazivu ustanove, a zatim po imenu i prezimenu polaznika.
4. Napravite lijevi spoj tablica *Ustanova* i *Polaznik*, tako da budu prikazane i ustanove kojima ne pripada nijedan polaznik.
5. Napravite desni spoj tablica *Ustanova* i *Polaznik*, tako da budu prikazani i polaznici koji ne pripadaju nijednoj ustanovi.
6. Napravite lijevi spoj tablica *Ustanova* i *Polaznik*, tako da budu prikazani i polaznici koji ne pripadaju nijednoj ustanovi. (Rezultat treba biti ekvivalentan onome iz točke 5.)
7. Napravite unutarnji spoj između tablica *Kategorija* i *Tecaj*.
8. Napravite unutarnji spoj tablice *Kategorija* same sa sobom, na način da budu prikazani sljedeći stupci: *Id* i *Naziv* nadređene kategorije, te zatim *Id* i *Naziv* podređene kategorije.
9. Napravite unutarnji spoj tablice *Kategorija* same sa sobom te zatim s tablicom *Tecaj*. U rezultatu bi trebali biti sljedeći stupci: *Id* i *Naziv* nadređene kategorije, zatim *Id* i *Naziv* podređene kategorije, te na kraju *Id*, *Sifra* i *Naziv* tečaja. Poredajte rezultat po nazivu nadkategorije, zatim po nazivu podkategorije te zatim po nazivu tečaja.
10. Napravite unutarnji spoj između tablica *Tecaj*, *Odrzavanje* i *Lokacija* koji će odgovoriti na pitanje koji se tečajevi održavaju u kojoj učionici. Od stupaca prikazite samo stupce iz tablica *Tecaj* i *Lokacija*. Pomoću ključne riječi `DISTINCT` u rezultatu prikazite samo jedinstvene retke.

8. Agregatne funkcije

Po završetku ovog poglavlja polaznik će moći:

- koristiti agregatne funkcije nad cijelom tablicom ili samo nad retcima koji zadovoljavaju neki uvjet
- prikazati ukupan broj redaka u tablici ili u skupu redaka
- izračunati sumu vrijednosti nekog stupca
- izračunati prosječnu vrijednost nekog stupca
- prikazati maksimalnu ili minimalnu vrijednost nekog stupca.

Agregatne funkcije služe za dobivanje (agregiranje) jednog podatka na temelju više redaka. Agregatne funkcije uvijek vraćaju samo jednu vrijednost. Primjer agregatnih funkcija jest računanje sume vrijednosti nekog stupca u nekoj tablici.

8.1. Funkcija COUNT

Agregatna funkcija `COUNT` služi za prebrojavanje redaka u tablici ili u skupu redaka (npr. redaka koji zadovoljavaju neki uvjet).

Najčešći način pozivanja funkcije `COUNT` je sa zvjezdicom kao argumentom. U tom načinu rada funkcija vraća ukupni broj redaka.

```
SELECT COUNT (*)
FROM Tablica
```

Sljedeći upit vratit će ukupni broj redaka u tablici *Polaznik*:

```
SELECT COUNT (*)
FROM Polaznik
```

Poziv funkcije `COUNT` (kao i drugih agregatnih funkcija) ne vraća više redaka, već samo jednu vrijednost – ukupni broj redaka.

Kako bi stupac u kojem se nalazi rezultat dobio neki naziv, može se upotrijebiti alias:

```
SELECT COUNT (*) AS Ukupno
FROM Polaznik
```

U drugom načinu pozivanja, funkciji `COUNT` se kao argument predaje jedan od stupaca. Rezultat funkcije je ukupni broj redaka za koje taj stupac nije `NULL`.

```
SELECT COUNT (Stupac)
FROM Tablica
```

Sljedeći upit vratit će ukupni broj redaka u tablici *Polaznik* za koje stupac *Ustanovald* nije `NULL`:

```
SELECT COUNT (UstanovaId)
FROM Polaznik
```

Osim nad cijelom tablicom, funkciju `COUNT` možemo pozvati i nad samo nekim retcima, te tako prebrojati retke koji zadovoljavaju neki uvjet. Sljedeći upit vratit će broj redaka u tablici *Polaznik* za koje je *Ustanovald* jednak 2.

Zanimljivosti i napomene

Agregatne funkcije možemo pozivati samo u klauzuli `SELECT` – ne možemo ih koristiti npr. u uvjetu u klauzuli `WHERE`.

Također, kad koristimo agregatne funkcije, u klauzuli `SELECT` ne možemo istovremeno navoditi i nazive stupaca.

Sljedeći upit **nije ispravan**:

```
SELECT COUNT (*), Ime
FROM Predavac
```

```
SELECT COUNT(*)
FROM Polaznik
WHERE UstanovaId = 2
```

Funkcija `COUNT` može se pozvati i uz ključnu riječ `DISTINCT`. Na taj se način prebrojavaju samo različite vrijednosti za zadani stupac. Sintaksa za ovakvo korištenje izgleda ovako:

```
SELECT COUNT(DISTINCT Stupac)
FROM Tablica
```

Sljedeći upit vratit će ukupni broj različitih vrijednosti koje ima stupac *UstanovaId* u tablici *Polaznik*.

```
SELECT COUNT(DISTINCT UstanovaId)
FROM Polaznik
```

8.2. Funkcija SUM

Agregatna funkcija `SUM` izračunava ukupan zbroj svih vrijednosti nekog stupca u tablici ili u skupu redaka.

Funkciji `SUM` kao parametar se predaje naziv stupca. Stupac mora biti numeričkog podatkovnog tipa.

```
SELECT SUM(Stupac)
FROM Tablica
```

Sljedeći upit vratit će sumu svih vrijednosti stupca *Cijena* u tablici *Tecaj*.

```
SELECT SUM(Cijena)
FROM Tecaj
```

Osim naziva stupaca, agregatnim funkcijama možemo predati i složenije izraze. U sljedećem primjeru računa se suma razlike stupaca *Cijena* i *Popust*.

```
SELECT SUM(Cijena - Popust)
FROM Tecaj
```

Sumu možemo računati i samo nad retcima koji zadovoljavaju neki uvjet. U sljedećem primjeru računa se suma samo za one tečajeve čija je *KategorijaId* jednaka 5:

```
SELECT SUM(Cijena)
FROM Tecaj
WHERE KategorijaId = 5
```

I uz funkciju `SUM` može se koristiti ključna riječ `DISTINCT`. U tom se slučaju u zbroj pribrajaju samo različite vrijednosti. Sintaksa takvog načina korištenja izgleda ovako:

```
SELECT SUM(DISTINCT Stupac)
FROM Tablica
```

U sljedećem primjeru računa se zbroj različitih vrijednosti stupca *Cijena* u tablici *Tecaj*:

```
SELECT SUM(DISTINCT Cijena)
FROM Tecaj
```


U klauzuli `SELECT` možemo pozvati više agregatnih funkcija (koje djeluju nad istim retcima):

```
SELECT COUNT(*) AS Broj, SUM(Cijena) AS Suma
FROM Tecaj
WHERE KategorijaId = 5
```

8.3. Funkcija AVG

Agregatna funkcija `AVG` izračunava prosjek svih vrijednosti nekog stupca u tablici ili u skupu redaka.

Funkciji `AVG` kao parametar se predaje naziv stupca. Stupac mora biti numeričkog podatkovnog tipa.

```
SELECT AVG(Stupac)
FROM Tablica
```

Sljedeći upit vratit će prosječnu vrijednost stupca `Cijena` u tablici `Tecaj`.

```
SELECT AVG(Cijena)
FROM Tecaj
```

Osim naziva stupaca, funkciji `AVG` (kao i drugim agregatnim funkcijama) možemo predati i složenije izraze. U sljedećem primjeru računa se prosječna vrijednost razlike stupaca `Cijena` i `Popust`.

```
SELECT AVG(Cijena - Popust)
FROM Tecaj
```

U sljedećem primjeru računa se prosjek cijene samo za one tečajeve čija je `KategorijaId` jednaka 5:

```
SELECT AVG(Cijena)
FROM Tecaj
WHERE KategorijaId = 5
```

I funkciju `AVG` može se pozvati uz ključnu riječ `DISTINCT`. U tom slučaju u računanje prosjeka ulaze samo različite vrijednosti.

```
SELECT AVG(DISTINCT Stupac)
FROM Tablica
```

Sljedeći upit vratit će prosjek različitih vrijednosti stupca `Cijena` u tablici `Tecaj`.

```
SELECT AVG(DISTINCT Cijena)
FROM Tecaj
```

8.4. Funkcija MAX

Agregatna funkcija `MAX` pronalazi najveću vrijednost nekog stupca u tablici ili u skupu redaka.

Funkciji `MAX` kao parametar se predaje naziv stupca. Stupac može biti bilo kojeg podatkovnog tipa. Ako se radi o tekstualnom tipu podataka, najvećom vrijednosti se smatra vrijednost koja dolazi posljednja po abecednom poretku.

```
SELECT MAX(Stupac)
FROM Tablica
```

Sljedeći upit vratit će maksimalnu vrijednost stupca *Cijena* u tablici *Tecaj*.

```
SELECT MAX(Cijena)
FROM Tecaj
```

U sljedećem primjeru traži se maksimum cijene samo za one tečajeve čija je *KategorijaId* jednaka 5:

```
SELECT MAX(Cijena)
FROM Tecaj
WHERE KategorijaId = 5
```

8.5. Funkcija MIN

Agregatna funkcija `MIN` pronalazi najveću vrijednost nekog stupca u tablici ili u skupu redaka.

Funkciji `MIN` kao parametar se predaje naziv stupca. Stupac može biti bilo kojeg podatkovnog tipa. Ako se radi o tekstualnom tipu podataka, najmanjom vrijednosti se smatra vrijednost koja dolazi prva po abecednom poretku.

```
SELECT MIN(Stupac)
FROM Tablica
```

Sljedeći upit vratit će minimalnu vrijednost stupca *Cijena* u tablici *Tecaj*.

```
SELECT MIN(Cijena)
FROM Tecaj
```

U sljedećem primjeru traži se minimum cijene samo za one tečajeve čija je *KategorijaId* jednaka 5:

```
SELECT MIN(Cijena)
FROM Tecaj
WHERE KategorijaId = 5
```

8.6. Agregatne funkcije i vrijednost NULL

Agregatne funkcije ignoriraju retke u kojim je vrijednost zadanog stupca `NULL`.

Funkcija `SUM` neće uključiti retke kod kojih je *Cijena* jednaka `NULL` u izračun:

```
SELECT SUM(Cijena)
FROM Tecaj
```

Ovako se ponašaju sve agregatne funkcije. I funkcija `COUNT` također neće brojati retke kod kojih je zadani stupac jednak `NULL` (ako se navede naziv stupca, a ne zvjezdica):

```
SELECT COUNT(Cijena)
FROM Tecaj
```

Samo u jednom slučaju stupci s vrijednostima `NULL` utječu na rezultat - ako je kod svih redaka nad kojima funkcija djeluje vrijednost zadanog stupca `NULL`, funkcija `SUM` vratit će kao rezultat `NULL`.

```
SELECT SUM(Cijena)
FROM Tecaj
WHERE Cijena IS NULL
```

Ovako se ponašaju sve agregatne funkcije osim funkcije `COUNT`.

Funkcija `COUNT` i dalje neće brojati retke kod kojih je vrijednost zadanog stupca `NULL`, ali neće kao rezultat vratiti `NULL`, već broj redaka, a to će biti 0.

```
SELECT COUNT(Cijena)
FROM Tecaj
WHERE Cijena IS NULL
```

Moguć je još jedan poseban slučaj. Ako u skupu redaka nema niti jednog retka (uvjet filtriranja ne vraća niti jedan redak), rezultat funkcije `SUM` bit će `NULL`.

```
SELECT SUM(Cijena)
FROM Tecaj
WHERE Id = 0
```

Opet, ovako se ponašaju sve agregatne funkcije osim funkcije `COUNT`.

Funkcija `COUNT` opet će vratiti broj redaka – 0.

```
SELECT COUNT(Cijena)
FROM Tecaj
WHERE Cijena IS NULL
```


9. Grupiranje podataka

Po završetku ovog poglavlja polaznik će moći:

- grupirati retke po jednom ili više stupaca
- koristiti agregatne funkcije nad grupiranim rezultatima
- filtrirati retke nad kojima se vrši grupiranje
- filtrirati dobivene grupe na temelju rezultata agregatnih funkcija.

Grupiranjem redaka može se dobiti izvrstan uvid u sadržaj i raspodjelu vrijednosti u podacima, te se zbog toga grupiranje često koristi u raznim izvještajima iz baza podataka. Grupiranje je iznimno korisno u kombinaciji s agregatnim funkcijama.

9.1. Grupiranje po jednom stupcu

Retci u tablici mogu se grupirati po vrijednosti nekog stupca. Za svaku jedinstvenu vrijednost tog stupca koja se pojavljuje u tablici formira se jedna grupa redaka u koju spadaju oni retci koji imaju tu vrijednost u promatranom stupcu.

Ako u tablici *Tecaj* postoje sljedeći retci:

Naziv	Kategorijald
Osnove računala (Windows)	3
Obrada teksta (Word)	3
Prezentacije (PowerPoint)	3
Uvod u HTML	4
Uvod u CSS	4
Uvod u Linux - rad s naredbama	5
Uvod u PHP i MySQL	5
Uvod u programski jezik Ruby	5
Wordionica	6
Exceliranje	6

Grupiranjem po identifikatoru kategorije (stupac *Kategorijald*) nastaju sljedeće četiri grupe redaka:

Naziv	Kategorijald
Osnove računala (Windows)	3
Obrada teksta (Word)	3
Prezentacije (PowerPoint)	3

Naziv	KategorijaId
Uvod u HTML	4
Uvod u CSS	4

Naziv	KategorijaId
Uvod u Linux - rad s naredbama	5
Uvod u PHP i MySQL	5
Uvod u programski jezik Ruby	5

Naziv	KategorijaId
Wordionica	6
Exceliranje	6

Rezultat upita s grupiranjem jest po jedan redak za svaku grupu.

KategorijaId
3
4
5
6

Zanimljivosti i napomene

U klauzuli `SELECT` je dozvoljeno je navesti samo one stupce po kojima se vrši grupiranje te agregatne funkcije.

Sljedeći upit **nije ispravan**:

```
SELECT Naziv
FROM Tecaj
GROUP BY KategorijaId
```

Grupiranje se obavlja pomoću klauzule `GROUP BY` u kojoj se navodi stupac (ili više stupaca) po kojima se grupira. Klauzula `GROUP BY` dolazi na kraju naredbe `SELECT`.

```
SELECT Stupac1, Stupac2
FROM Tablica
GROUP BY Stupac1, Stupac2
```

Sljedeći upit grupirat će retke u tablici *Tecaj* po stupcu *KategorijaId*:

```
SELECT KategorijaId
FROM Tecaj
GROUP BY KategorijaId
```

9.2. Grupiranje po više stupaca

Osim po jednom stupcu, grupiranje redaka može se obaviti i po više stupaca. U tom slučaju grupe se formiraju za svaku jedinstvenu kombinaciju vrijednosti tih stupaca u tablici.

Ako u tablici *Tecaj* postoje sljedeći retci:

Naziv	Kategorijald	BrojDana
Osnove računala (Windows)	3	4
Obrada teksta (Word)	3	4
Prezentacije (PowerPoint)	3	3
Uvod u HTML	4	3
Uvod u CSS	4	3
Uvod u Linux - rad s naredbama	5	4
Uvod u PHP i MySQL	5	5
Uvod u programski jezik Ruby	5	5
Wordionica	6	1
Exceliranje	6	1

Grupiranjem po identifikatoru kategorije (stupac *Kategorijald*) i broju dana koliko tečaj traje (stupac *BrojDana*) nastaje sljedećih 6 grupa redaka:

Naziv	Kategorijald	BrojDana
Osnove računala (Windows)	3	4
Obrada teksta (Word)	3	4

Naziv	Kategorijald	BrojDana
Prezentacije (PowerPoint)	3	3

Naziv	Kategorijald	BrojDana
Uvod u HTML	4	3
Uvod u CSS	4	3

Naziv	Kategorijald	BrojDana
Uvod u Linux - rad s naredbama	5	4

Naziv	Kategorijald	BrojDana
Uvod u PHP i MySQL	5	5
Uvod u programski jezik Ruby	5	5

Naziv	Kategorijald	BrojDana
Wordionica	6	1
Exceliranje	6	1

Kao rezultat upita s grupiranjem dobiva se po jedan redak za svaku grupu.

Kategorijald	BrojDana
3	4
3	3
4	3
5	4
5	5
6	1

Sljedeći upit grupirat će retke u tablici *Tecaj* po stupcima *Kategorijald* i *BrojDana*:

```
SELECT KategorijaId, BrojDana
FROM Tecaj
GROUP BY KategorijaId, BrojDana
```

Grupiranje se može obaviti i nad vrijednosti izraza (izračunate na temelju vrijednosti više stupaca). Sljedeći upit grupirat će retke u tablici *Tecaj* na temelju ukupnog trajanja tečaja, koje se računa kao broj dana puta broj sati po danu:

```
SELECT BrojDana * BrojSatiPoDanu
FROM Tecaj
GROUP BY BrojDana * BrojSatiPoDanu
```

Grupiranje se može obaviti i nad spojem tablica. Ako se rezultatu umjesto identifikatora želi ispisati naziv kategorije, potrebno je napraviti spoj s tablicom *Kategorija*, pa nakon toga napraviti grupiranje:

```
SELECT Kategorija.Naziv as KategorijaNaziv
FROM Tecaj
INNER JOIN Kategorija
ON Kategorija.Id = KategorijaId
GROUP BY KategorijaId, Kategorija.Naziv
```

Međutim, kako se u klauzuli `SELECT` smiju nalaziti samo stupci po kojima se radi grupiranje, stoga i naziv kategorije treba biti dodan u klauzulu `GROUP BY`. Budući da je identifikator kategorije jedinstven, dodavanje ovog stupca u klauzulu `GROUP BY` neće povećati broj grupa.

U klauzuli `GROUP BY` ne mogu se koristiti aliasi.

9.3. Grupiranje uz agregatne funkcije

Grupiranje redaka samo po sebi nije osobito korisno. Za primjer se može uzeti upit kojim se grupiraju retci iz tablice *Tecaj* po vrijednosti stupca *KategorijaId*:

```
SELECT KategorijaId
FROM Tecaj
GROUP BY KategorijaId
```

Prethodni upit daje isti rezultat kao i ovaj, jednostavniji upit:

```
SELECT DISTINCT KategorijaId
FROM Tecaj
```

Međutim, kod grupiranja redaka možemo koristiti agregatne funkcije, što grupiranje čini vrlo korisnim. U klauzuli *SELECT* možemo navesti bilo koju od agregatnih funkcija, a agregiranje podataka obavlja se nad pojedinom grupom redaka (a ne nad cijelom tablicom). Ako se upotrijebi funkcija *COUNT*, bit će prebrojani retci u svakoj pojedinoj grupi.

Sljedeći upit odgovorit će na pitanje koliko svaka kategorija sadrži tečajeva:

```
SELECT KategorijaId, COUNT(*)
FROM Tecaj
GROUP BY KategorijaId
```

Sljedeći upit odgovorit će na pitanje koliki je prosječan broj dana tečaja u pojedinoj kategoriji:

```
SELECT KategorijaId, AVG(BrojDana) AS ProsjDana
FROM Tecaj
GROUP BY KategorijaId
```

9.4. Postavljanje uvjeta nad grupom

Grupiranje se može obaviti i nad skupom redaka koji zadovoljavaju neki uvjet. Klauzula *GROUP BY* dolazi poslije klauzule *WHERE*.

U sljedećem primjeru grupiramo samo one retke iz tablice *Tecaj* za koje je vrijednost stupca *Cijena* veća od 400.

```
SELECT KategorijaId
FROM Tecaj
WHERE Cijena > 400
GROUP BY KategorijaId
```

No, filtriranje pomoću klauzule *WHERE* događa se na retcima prije grupiranja.

Filtriranje na retcima nakon grupiranja, tj. na samim grupama obavlja se pomoću klauzule *HAVING*. Postavljanje uvjeta nad grupom postiže se upotrebom rezultata agregatnih funkcija nad grupom, tako da se u klauzuli *HAVING* koriste agregatne funkcije (dok se u klauzuli *WHERE* one ne mogu koristiti).

Klauzula *HAVING* uvijek dolazi nakon klauzule *GROUP BY*:

```
SELECT Stupac1, Stupac2
FROM Tablica
```

```
GROUP BY Stupa1, Stupa2
HAVING Uvjet
```

Primjer postavljanja uvjeta nad grupom redaka bilo bi grupiranje tečajeva po cijeni, ali uz uvjet da se u rezultatu ne trebaju vidjeti grupe koje imaju samo jedan redak.

```
SELECT Cijena, COUNT(*) AS UkupnoTecajeva
FROM Tecaj
GROUP BY Cijena
HAVING COUNT(*) > 1
```

Razliku između djelovanja klauzule `WHERE` i klauzule `HAVING` može ilustrirati sljedeći primjer:

```
SELECT Cijena, COUNT(*) AS UkupnoTecajeva
FROM Tecaj
WHERE BrojDana > 1
GROUP BY Cijena
HAVING COUNT(*) > 1
```

U gornjem upitu prvo se primjenjuje uvjet filtriranja iz klauzule `WHERE` – uzimaju se samo oni tečajevi koji traju dulje od jednog dana. Nakon toga se obavlja grupiranje po stupcu `Cijena`. Na kraju, iz rezultata se izbacuju grupe koje sadrže samo jedan redak.

Naziv	BrojDana	Cijena
Osnove računala (Windows 7)	3	500
Obrada teksta (Word 2010)	3	500
Prezentacije (PowerPoint 2010)	3	400
Uvod u HTML	4	400
Uvod u CSS	4	400
Uvod u Linux - rad s naredbama	5	500
Uvod u PHP i MySQL	5	750
Uvod u programski jezik Ruby	5	600
Wordionica	1	0
Exceliranje	1	0



Naziv	BrojDana	Cijena
Osnove računala (Windows 7)	3	500
Obrada teksta (Word 2010)	3	500
Prezentacije (PowerPoint 2010)	3	400
Uvod u HTML	4	400
Uvod u CSS	4	400
Uvod u Linux - rad s naredbama	5	500
Uvod u PHP i MySQL	5	750
Uvod u programski jezik Ruby	5	600

GROUP BY Cijena

Cijena	Count
500	3
400	3
750	1
600	1

HAVING COUNT (*) > 1

Cijena	Count
500	3
400	3

9.5. Grupiranje i vrijednost NULL

Ako je za neke retke vrijednost stupca po kojem se grupira jednaka NULL, svi takvi retci nalazit će se u jednoj grupi.

Kako vrijednost stupca *Cijena* za neke retke u tablici *Tecaj* može biti NULL, sljedeći upit ilustrirat će da se za vrijednost NULL dobije jedan redak u rezultatu (tj. jedna grupa).

```
SELECT Cijena, COUNT(*)
FROM Tecaj
GROUP BY Cijena
```

9.6. Vježba: Grupiranje i agregatne funkcije

1. Izračunajte prosječni kapacitet učionice u kojoj se tečaj održava. (Tablica *Lokacija*)
2. Izračunajte ukupan kapacitet svih učionica zajedno.
3. Ispišite ukupan broj polaznika u tablici *Polaznik*.
4. Ispišite ukupan broj polaznika ženskog spola.
5. Izračunajte prosječnu dob polaznika (dob izračunajte na temelju stupca *DatumRodjenja*).
6. Grupirajte polaznike po spolu. Ispišite ukupan broj polaznika koji pripadaju pojedinom spolu. Ispišite i prosječnu dob polaznika koji pripadaju pojedinom spolu.
7. Grupirajte polaznike po dobi. Ispišite ukupan broj polaznika za svaku dob. Poredajte rezultat po dobi (od najmanje do najveće).
8. Grupirajte polaznike po spolu i po dobi. Ispišite ukupan broj polaznika za svaku grupu.
9. Grupirajte polaznike po ustanovi (stupac *UstanovaId*). Za svaku ustanovu ispišite broj polaznika i prosječnu dob polaznika. Poredajte rezultat po broju polaznika (od najvećeg prema najmanjem broju).
10. Grupirajte polaznike po ustanovi. U rezultatu ispišite broj polaznika i prosječnu dob. Umjesto identifikatora ustanove, u rezultatu ispišite naziv ustanove.
11. Za svaku godinu u kojoj su održani tečajevi, ispišite ukupan broj održanih tečajeva.
12. Za svaku godinu u kojoj su održani tečajevi, ispišite ukupan broj polaznika.

10. Operacije nad skupovima

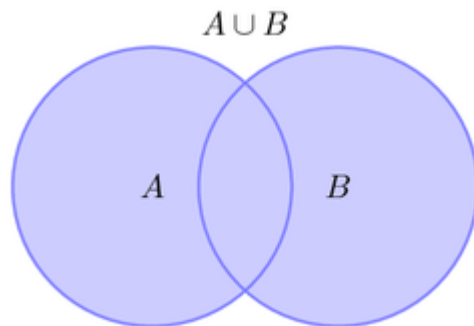
Po završetku ovog poglavlja polaznik će moći:

- napraviti uniju između skupova redaka
- napraviti uniju s ponavljanjem između skupova redaka
- napraviti presjek između skupova redaka
- napraviti razliku između skupova redaka.

U ovom poglavlju prikazan je pregled operacija nad skupovima, čiji je temelj u algebri skupova. Rezultate upita može se promatrati kao skupove redaka. U SQL-u su nad skupovima redaka moguće operacije unije, presjeka i razlike.

10.1. Unija

Unija dvaju skupova takav je skup koji sadrži sve elemente iz prvog i sve elemente iz drugog skupa.



Unija između rezultata dvaju upita u SQL-u dobiva se povezivanjem dvaju upita pomoću ključne riječi `UNION`. Sintaksa takvog upita izgleda ovako:

```
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica1
UNION
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica2
```

Broj stupaca u oba upita mora biti podjednak. Stupci koji se nalaze na istom mjestu moraju odgovarati jedni drugima po tipu podatka. Nazivi stupaca ne moraju biti isti u oba upita (u rezultatu će biti prikazani nazivi stupaca iz prvog upita).

Sljedeći upit vratit će uniju između tablice *Predavac* i tablice *Polaznik*.

```
SELECT Ime, Prezime
FROM Predavac
UNION
SELECT Ime, Prezime
FROM Polaznik
```

Zanimljivosti i napomene

Odgovarajući stupci ne moraju biti sasvim jednaki po podatkovnom tipu – moguća je unija između stupaca različitih numeričkih tipova, (npr. *int* i *decima*).

Rezultat ovog upita bit će lista svih predavača iz tablice *Predavac* i svih polaznika iz tablice *Polaznik*.

Ako u tablici *Polaznik* postoje polaznici koji imaju isto ime i prezime kao i neki predavači, ta imena i prezimena će se u rezultatu pojaviti samo jednom. Dva retka koji imaju iste vrijednosti svih stupaca smatraju se istim elementom skupa, te u uniji neće biti navedeni dva puta.

Ako se rezultat unije želi poredati po nekom od stupaca, na kraj upita je potrebno dodati klauzulu `ORDER BY`.

```
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica1

UNION

SELECT Stupac1, Stupac2, Stupac3
FROM Tablica2

ORDER BY Stupac1, Stupac2
```

Zanimljivosti i napomene

Prilikom usporedbe vrijednosti u operacija nad skupovima, dvije vrijednosti `NULL` se smatraju jednakima.

Zanimljivosti i napomene

U klauzuli `ORDER BY` moraju se navoditi nazivi stupaca ili aliasi navedeni u prvom upitu u uniji. Moguće je koristiti i redne brojeve.

Sljedeći upit vratit će uniju između predavača i polaznika poredanu po imenu i prezimenu:

```
SELECT Ime, Prezime
FROM Predavac

UNION

SELECT Ime, Prezime
FROM Polaznik

ORDER BY Ime, Prezime
```

Unija je moguća i između više upita:

```
SELECT Naziv
FROM Ustanova

UNION

SELECT Naziv
FROM Odjel

UNION

SELECT Naziv
FROM Lokacija
```

10.2. Unija s ponavljanjima

Kod unije s ponavljanjima, dva retka koja imaju jednake vrijednosti svih stupaca ne smatraju se istim elementom, i bit će navedena dvaput u rezultatu.

Unija s ponavljanjem ne postoji u algebri skupova, već je to nadgradnja dodana u jeziku SQL.

Unija s ponavljanjem dobiva se pomoću ključnih riječi `UNION ALL`:

```
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica1

UNION ALL

SELECT Stupac1, Stupac2, Stupac3
FROM Tablica2
```

Sljedećim upitom napravljena je unija s ponavljanjima između tablica *Predavac* i *Polaznik*.

```
SELECT Ime, Prezime
FROM Predavac
```

UNION ALL

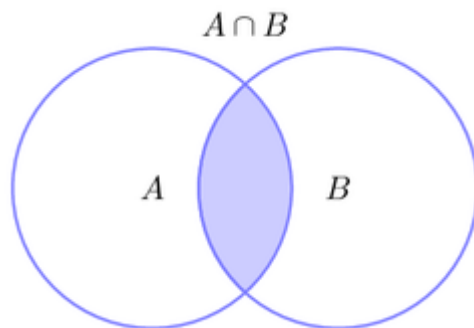
```
SELECT Ime, Prezime
FROM Polaznik
```

```
ORDER BY Ime, Prezime
```

Rezultat je poredan po imenu i prezimenu kako bi se lakše uočila ponavljanja, odnosno duple vrijednosti za ime i prezime.

10.3. Presjek

Presjek dvaju skupova takav je skup koji sadrži one elemente prvog skupa koji se nalaze i u drugom skupu.



Presjek dvaju skupova dobiva se upotrebom ključne riječi `INTERSECT`. Ako se koristi klauzula `ORDER BY`, ona se (kao i kod unije) dodaje na kraju cijelog upita.

```
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica1
```

INTERSECT

```
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica2
```

```
ORDER BY Stupac1, Stupac2
```

Sljedeći primjer vratit će presjek između tablica *Predavac* i *Polaznik*.

```
SELECT Ime, Prezime
FROM Predavac
```

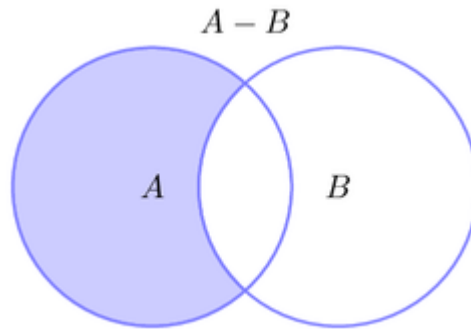
INTERSECT

```
SELECT Ime, Prezime
FROM Polaznik
```

U rezultatu će biti prisutne samo one osobe čije se ime i prezime pojavljuju u obje tablice.

10.4. Razlika

Razlika dvaju skupova takav je skup koji sadrži one elemente prvog skupa koji se **ne nalaze** u drugom skupu.



Razlika dvaju skupova dobiva se upotrebom ključne riječi `EXCEPT`. Ako se koristi klauzula `ORDER BY`, ona se (kao i kod unije) dodaje na kraju cijelog upita.

Zanimljivosti i napomene

Na nekim sustavima (Oracle) umjesto ključne riječi `EXCEPT` koristi se ključna riječ `MINUS`.

```
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica1

EXCEPT

SELECT Stupac1, Stupac2, Stupac3
FROM Tablica2

ORDER BY Stupac1, Stupac2
```

Sljedeći primjer vratit će razliku između tablica *Predavac* i *Polaznik*.

```
SELECT Ime, Prezime
FROM Predavac

EXCEPT

SELECT Ime, Prezime
FROM Polaznik
```

U rezultatu će biti prisutni samo oni predavači čije se ime i prezime ne pojavljuje u tablici *Polaznik*.

Ako se zamijeni redoslijed upita između kojih se radi razlika, rezultat će se promijeniti. Sljedeći upit daje posve drugi rezultat – polaznike koji se ne pojavljuju u tablici *Predavac*:

```
SELECT Ime, Prezime
FROM Polaznik

EXCEPT

SELECT Ime, Prezime
FROM Predavac
```

Za razliku od operacije razlike, kod operacija unije, unije s ponavljanjem i presjeka redoslijed upita nije imao utjecaj na rezultat (osim eventualno na poredak redaka u rezultatu).

11. Podupiti

Po završetku ovog poglavlja polaznik će moći:

- koristiti podupit unutar klauzule `SELECT`
- razumjeti razliku između koreliranog i nekoreliranog podupita
- koristiti podupite unutar klauzule `WHERE`
- ugnježdživati više podupita
- kada je moguće, koristiti spoj tablica umjesto podupita.

U ovom poglavlju objašnjava se što je to podupit i daje se pregled načina korištenja podupita.

11.1. Podupit

Unutar upita moguće je pozvati drugi upit. Glavni upit naziva se vanjskim upitom, dok se upit koji se poziva iz glavnog upita naziva **podupitom**. Kao podupit može se koristiti samo naredba `SELECT`.

Podupit se može pozvati iz liste za odabir (tj. u klauzuli `SELECT` vanjskog upita), ili unutar uvjeta filtriranja (u klauzuli `WHERE`).

Osim u naredbi `SELECT`, podupiti se mogu koristiti i u naredbama `UPDATE` i `DELETE` (samo unutar klauzule `WHERE`).

11.2. Podupit u listi za odabir

Na mjestu jednog od stupaca u listi za odabir može se navesti podupit. Na taj se način u rezultatu upita dobiva novi stupac u kojem će biti prikazan rezultat podupita.

Podupit u listi za odabir potrebno je staviti unutar zagrada, a dobivenom stupcu moguće je zadati naziv pomoću aliasa.

```
SELECT Stupac1, Stupac2,
       (SELECT Izraz FROM Tablica2) AS Alias
FROM Tablica1
```

Podupit naveden u klauzuli `SELECT` kao rezultat smije vraćati samo jednu vrijednost. Zbog toga, takav podupit u vlastitoj listi za odabir smije imati naveden samo jedan stupac, a kao rezultat smije vratiti samo jedan redak.

Sljedeći upit ispisuje ime i prezime predavača te uz njih ispisuje i ukupan broj održanih tečajeva (od strane svih predavača):

```
SELECT Ime, Prezime,
       (SELECT COUNT(*) FROM Odrzavanje) AS Ukupno
FROM Predavac
```

Broj ukupno održanih tečajeva bit će isti za sve predavače.

Zanimljivosti i napomene

Podupit se uvijek navodi unutar zagrada.

11.3. Korelirani podupit

U gornjem primjeru ne postoji veza između vanjskog upita i podupita. Podupit se, kao i vanjski upit, izvršava samo jednom. Broj ukupno održanih tečajeva isti je za sve predavače, te se podupit izvršava samo jednom.

Zanimljivosti i napomene

Kao i nekorelirani, i korelirani podupit koji se koristi u klauzuli `SELECT` također smije vraćati samo jednu vrijednost – smije imati samo jedan stupac u listi za odabir, i za svaki redak vanjskog upita smije vratiti samo jedan redak.

Kod **koreliranog podupita** postoji veza između vanjskog upita i podupita.

U slučaju da se za svakog predavača želi prikazati broj tečajeva koje je on održao, postoji veza između vanjskog upita i podupita i tu se radi o koreliranom podupitu. Vanjski upit se i dalje izvršava samo jednom, a podupit se izvršava više puta – po jednom za svaki redak.

```
SELECT Ime, Prezime,
       (SELECT COUNT(*) FROM Odrzavanje
        WHERE PredavacId = Predavac.Id) AS Odrzao
FROM Predavac
```

Budući da se podupit izvršava po jednom za svaki redak, izvršavanje ovakvog upita može biti osjetno sporije ako vanjski upit vraća velik broj redaka. U praksi, sustav za upravljanje bazama podataka pokušat će optimizirati korelirani podupit te se on neće uvijek zaista izvršavati na ovaj način.

11.4. Podupit u uvjetu

Podupit je moguće koristiti i unutar uvjeta, odnosno unutar klauzule `WHERE`. Podupit unutar uvjeta se može koristiti na tri načina:

- usporedba s rezultatom podupita - bez ključne riječi, ili s ključnim riječima `ALL` i `ANY`
- provjera nalazi li se neka vrijednost u rezultatu podupita - podupit s ključnom riječi `IN`
- provjera postoji li neki redak – podupit s ključnom riječi `EXISTS`

11.5. Usporedba s rezultatom podupita

Prvi način upotrebe podupita unutar uvjeta je usporedba neke vrijednosti s rezultatom podupita. Unutar klauzule `WHERE`, vrijednost iz vanjskog upita uspoređuje se pomoću operatora uspoređivanja s rezultatom koji vraća podupit. Primjer sintakse takvog upita izgleda ovako:

```
SELECT *
FROM Tablica1
WHERE Stupac1 > (SELECT Stupac2
                 FROM Tablica2)
```

Na mjestu operatora `<` može biti bilo koji od operatora uspoređivanja (`=`, `<`, `>`, `<=`, `>=`, `<>`).

Kod ovakve upotrebe, podupit smije vratiti samo jednu vrijednost – dakle smije imati samo jedan stupac u listi za odabir, i smije vratiti samo jedan redak.

Primjer ovakvog upita bi bio upit koji vraća sve tečajeve čija je cijena veća od prosječne cijene:

```
SELECT *
FROM Tecaj
WHERE Cijena > (SELECT AVG(Cijena) FROM Tecaj)
```

Kod ovakvog načina upotrebe upita, uz upit možemo dodati ključne riječi ALL ili ANY. Primjer sintakse takvog upita izgleda ovako:

```
SELECT *
FROM Tablica1
WHERE Stupac1 < ALL (SELECT Stupac2
                     FROM Tablica2)
```

Na mjestu operatora < može biti bilo koji od operatora uspoređivanja (=, <, >, <=, >=, <>).

U slučaju kad upotrebljavamo ključnu riječ ALL ili ANY, podupit smije vratiti više redaka (iako i dalje smije imati samo jedan stupac naveden u listi za odabir).

Ključna riječ **ALL** označava da će uvjet usporedbe biti zadovoljen samo ako je istinit za **sve** retke iz podupita.

Sljedeći upit vratit će one polaznike koji su stariji od svih predavača:

```
SELECT *
FROM Polaznik
WHERE DatumRodjenja < ALL (SELECT DatumRodjenja
                           FROM Predavac)
```

Ključna riječ **ANY** označava da će uvjet usporedbe biti zadovoljen već ako je istinit za **barem jedan** redak iz podupita.

Sljedeći upit vratit će one polaznike koji su stariji od barem jednog predavača:

```
SELECT *
FROM Polaznik
WHERE DatumRodjenja < ANY (SELECT DatumRodjenja
                            FROM Predavac)
```

Zanimljivosti i napomene

Umjesto ključne riječi ANY može se upotrijebiti i njen sinonim – ključna riječ SOME.

11.6. Podupit s operatorom IN

Drugi način upotrebe upita unutar klauzule WHERE je uz operator IN. Na ovaj način moguće je provjeriti postoji li neka vrijednost unutar skupa vrijednosti koje vraća podupit.

```
SELECT *
FROM Tablica1
WHERE Stupac1 IN (SELECT Stupac2
                 FROM Tablica2)
```

Podupit koji se koristi na ovaj način može vraćati više redaka, ali u listi za odabir smije imati samo jedan stupac.

Sljedeći upit vratit će sve predavače koji su održali bar jedan tečaj u 2015. godini:

```
SELECT *
FROM Predavac
WHERE Id IN (SELECT PredavacId
```

```
FROM Odrzavanje
WHERE YEAR(Pocetak) = 2015)
```

U uvjetu vanjskog upita provjerava se nalazi li se identifikator predavača u skupu vrijednosti koje vraća podupit, a to su identifikatori svih predavača koji su održavali tečajeve u 2015. godini.

Dodavanjem modifikatora `NOT` moguće je provjeriti da se neka vrijednost ne nalazi unutar skupa vrijednosti koje vraća podupit.

Sljedeći upit vraća sve predavače koji nisu održali nijedan tečaj u 2015. godini:

```
SELECT *
FROM Predavac
WHERE Id NOT IN (SELECT PredavacId
FROM Odrzavanje
WHERE YEAR(Pocetak) = 2015)
```

11.7. Podupit s ključnom riječi EXISTS

Treći način upotrebe podupita unutar uvjeta je provjera dobiva li se kao rezultat podupita ijedan redak, odnosno postoji li ijedan redak koji zadovoljava određeni uvjet. Pritom se koristi ključna riječ `EXISTS`, a sintaksa takvog upita je sljedeća:

```
SELECT *
FROM Tablica1
WHERE EXISTS (SELECT *
FROM Tablica2
WHERE Uvjet)
```

Podupit koji se koristi na ovaj način može vraćati više redaka, a također može imati i proizvoljan broj stupaca navedenih u listi za odabir (nije bitno što je navedeno u listi za odabir, već je bitan samo broj redaka koje podupit vraća).

Sljedeći upit vratit će one odjele u kojima je zaposlen bar jedan predavač (tj. za koje postoji bar jedan redak u tablici *Predavac*):

```
SELECT *
FROM Odjel
WHERE EXISTS (SELECT *
FROM Predavac
WHERE OdjelId = Odjel.Id)
```

Dodavanjem modifikatora `NOT` moguće je provjeriti da neki redak **ne** postoji.

Sljedeći upit vratit će one odjele u kojima nije zaposlen nijedan predavač (tj. za koje ne postoji nijedan redak u tablici *Predavac*):

```
SELECT *
FROM Odjel
WHERE NOT EXISTS (SELECT *
FROM Predavac
WHERE OdjelId = Odjel.Id)
```

Zanimljivosti i napomene

Podupiti s ključnom riječi `EXISTS` su u praksi uvijek korelirani upiti. Za svaki redak iz vanjskog upita provjerava se vraća li podupit ijedan rezultat.

11.8. Podupit u klauzulama FROM i JOIN

Rezultat podupita može se tretirati kao tablica u smislu da ga možemo koristiti u klauzuli FROM. Sintaksa za takav način korištenja izgleda ovako:

```
SELECT *
FROM
  (SELECT Stupac1, Stupac2...
   FROM Tablica) AS Alias
```

Prilikom takvog korištenja moramo obavezno zadati alias za skup redaka koji je rezultat podupita.

Primjer takvog upita bilo bi grupiranje koje se želi napraviti nad rezultatom unije. U sljedećem primjeru radi se unija između redaka u tablicama *Polaznik* i *Predavac* te se nakon toga rezultat te unije grupira po stupcu *Ime*.

```
SELECT Ime, COUNT(*) AS BrojPojavljivanja
FROM
  (SELECT Ime, Prezime
   FROM Polaznik
  UNION
   SELECT Ime, Prezime
   FROM Predavac) AS Osoba
GROUP BY Ime
```

Rezultat gornjeg upita je lista svih imena koja se pojavljuju u tablicama *Polaznik* i *Predavac* te broj pojavljivanja za svako ime.

Između rezultata podupita i druge tablice moguće je napraviti i spoj. Pritom se podupit može naći u klauzuli FROM, ali i u klauzuli JOIN.

Primjer takvog upita bilo bi spajanje s rezultatom grupiranja. U sljedećem upitu se na drugačiji način dolazi do broja održanih tečajeva ispisanih uz predavača:

```
SELECT Ime, Prezime, BrojOdrzavanja
FROM Predavac
JOIN
  (SELECT PredavacId, COUNT(*) AS BrojOdrzavanja
   FROM Odrzavanje
   GROUP BY PredavacId) AS OdrzavanjePoPredavacu
ON Id = PredavacId
```

11.9. Podupiti i vrijednost NULL

Ako među vrijednostima koje vraća podupit može biti i vrijednost `NULL`, treba na to posebno pripaziti kod podupita koji sadrže ključne riječi `ALL` i `NOT IN`.

U tablici *Lokacija* nalaze se učionice u kojima se održavaju tečajevi. Stupac *BrojMjesta* sadrži podatak o tome koliko ima mjesta u učionici. U tablici *Tecaj* stupac *MinimalnoPolaznika* sadrži podatak o minimalnom broju polaznika koji je potreban da bi se tečaj mogao održati. Ako je minimalan broj polaznika veći od broja mjesta u učionici, taj se tečaj nikad ne može održati u toj učionici.

Sljedeći upit vratit će one tečajeve koji se mogu održati u **barem jednoj** od učionica:

```
SELECT *
FROM Tecaj
WHERE MinimalnoPolaznika < ANY
      (SELECT BrojMjesta
       FROM Lokacija)
```

Naravno, za svaki tečaj postoji barem jedna učionica dovoljno velika da bi se on mogao održati u njoj. Gornji upit vraća stoga sve retke iz tablice.

Ali, sličan upit koji koristi ključnu riječ `ALL` neće vratiti očekivani rezultat:

```
SELECT *
FROM Tecaj
WHERE MinimalnoPolaznika < ALL
      (SELECT BrojMjesta
       FROM Lokacija)
```

Prethodni upit trebao bi vratiti one tečajeve koji se mogu održati u svim učionicama. Međutim, nijedan redak neće biti vraćen kao rezultat.

Razlog je u tome što neki retci u tablici *Lokacija* imaju vrijednost `NULL` u stupcu *BrojMjesta* te se u skupu vrijednosti koje vraća podupit nalazi i vrijednost `NULL`. Uvjet vanjskog upita evaluira se kao laž za svaki redak iz vanjskog upita, zbog toga što se usporedba s vrijednosti `NULL` uvijek evaluira kao laž.

Kako u tablici *Lokacija* stupac *BrojMjesta* može sadržavati vrijednost `NULL`, podupit iz sljedećeg primjera vraća i vrijednosti `NULL`.

Da bi upit vratio očekivani rezultat, potrebno je ukloniti vrijednosti `NULL` iz rezultata podupita:

```
SELECT *
FROM Tecaj
WHERE MinimalnoPolaznika < ALL
      (SELECT BrojMjesta
       FROM Lokacija
       WHERE BrojMjesta IS NOT NULL)
```

Slični problemi s vrijednostima `NULL` javljaju se i kod podupita sa ključnim riječima **NOT IN**. Sljedeći upit vratit će sve odjele u kojima postoje predavači:

```
SELECT *
FROM Odjel
WHERE Id IN (SELECT OdjelId
            FROM Predavac)
```

Važno je primijetiti da će podupit vratiti i vrijednosti `NULL`, jer ih stupac *OdjelId* u tablici *Predavac* sadrži.

Međutim, upravo zbog toga što se u rezultatu podupita nalaze i vrijednosti `NULL`, sljedeći upit neće dati očekivani rezultat:

```
SELECT *
FROM Odjel
WHERE Id NOT IN (SELECT OdjelId
                 FROM Predavac)
```

Bilo bi očekivano da upit vrati sve preostale odjele, dakle one u kojima ne postoje predavači, ali ovaj upit neće vratiti nijedan rezultat.

Razlog tomu je što se ovako napisan uvjet evaluira kao laž za svaki redak iz vanjskog upita, zbog toga što se usporedba s vrijednosti `NULL` uvijek evaluira kao laž.

Kako bi upit vratio očekivani rezultat, potrebno je ukloniti vrijednosti `NULL` iz rezultata podupita:

```
SELECT *
FROM Odjel
WHERE Id NOT IN (SELECT OdjelId
                FROM Predavac
                WHERE OdjelId IS NOT NULL)
```

11.10. Alternativa korištenju podupita

U nekim slučajevima nije nužno koristiti podupit već se zadani problem može riješiti na drugi način. Kako su korelirani podupiti zahtjevniji za izvođenje, kad je to moguće bolje je napisati upit na drugi način, korištenjem spoja tablica.

Naprimjer, popis svih predavača i naziva odjela u kojima su oni zaposleni moguće je ispisati pomoću podupita:

```
SELECT Ime, Prezime,
       (SELECT Naziv
        FROM Odjel
        WHERE Odjel.Id = Predavac.OdjelId) AS Odjel
FROM Predavac
```

Međutim, isti rezultat kao i s gornjim (koreliranim) podupitom možemo dobiti pomoću spoja tablica, što je efikasnije i jednostavnije rješenje. Kako postoje predavači za koje je *OdjelId* jednak `NULL`, potrebno je upotrijebiti lijevi spoj:

```
SELECT Ime, Prezime, Naziv AS Odjel
FROM Predavac
LEFT JOIN Odjel
ON Predavac.OdjelId = Odjel.Id
```

Zanimljivosti i napomene

U praksi, nekad će i sustav za upravljanje bazama podataka sam optimizirati upit s podupitom, tako da će se on interno zapravo izvršavati kao spoj tablica.

12. Pogledi

Po završetku ovog poglavlja polaznik će moći:

- razumjeti razloge korištenja pogleda
- stvoriti pogled
- koristiti pogled u upitu
- obrisati pogled iz baze podataka.

U ovom poglavlju objašnjava se što je pogled i koji su razlozi za njegovo korištenje.

12.1. Pogled

Pogled (*view*) je objekt u bazi podataka koji predstavlja virtualnu tablicu. Kao i tablica, pogled prikazuje podatke u tabličnom obliku.

Pogled se stvara zadavanjem upita. Podaci koje taj upit vraća su podaci koji će biti dostupni putem pogleda. Stupci pogleda – njihov redoslijed, nazivi i tipovi podataka, kao i retci koje pogled prikazuje, definirani su upitom pomoću kojeg je pogled stvoren.

Pogled može prikazivati podatke koji su kombinacija više tablica, podatke koji su dobiveni grupiranjem redaka, odnosno rezultat proizvoljno složenog upita.

Podaci u pogledu uvijek su ažurni i odgovaraju trenutnim podacima u tablicama iz kojih su uzeti.

Pogledi u prvom redu služe da bi se složeni upiti mogli definirati na jednom mjestu te da bi se na taj način podaci iz baze mogli lakše i jednostavnije koristiti. Poslovnu logiku dovoljno je definirati na jednom mjestu – u pogledu, a potom se pogled može pozivati s više mjesta. Na taj je način lakše pisati i razumjeti složene upite.

Drugi razlog korištenja pogleda je ograničenje prava pristupa. Nekim korisnicima baze moguće je ograničiti pristup samo na neke poglede, a u tim pogledima moguće je ograničiti dostupne stupce i retke na temelju proizvoljnog upita.

12.2. Stvaranje pogleda

Pogled se stvara pomoću naredbe `CREATE VIEW`, iza koje slijedi upit koji definira pogled:

```
CREATE VIEW Pogled AS
SELECT Stupac1, Stupac2, Stupac3
FROM Tablica
```

U sljedećem primjeru definira se pogled koji vraća polaznike koji su studenti (polaznici kod kojih je stupac *TipId* jednak 1):

```
CREATE VIEW Student AS
SELECT *
FROM Polaznik
WHERE TipId = 1
```

Primjer malo složenijeg pogleda bio bi pogled koji ispisuje detaljne podatke o održavanju tečaja – naziv tečaja, datum održavanja, trajanje tečaja i učionicu u kojoj se tečaj održava.

```
CREATE VIEW Raspored AS
SELECT TecajId,
       PredavacId,
       LokacijaId,
       Tecaj.Naziv AS Tecaj,
       Tecaj.Sifra,
       Odrzavanje.Pocetak,
       Tecaj.BrojDana,
       Tecaj.BrojSatiPoDanu,
       Lokacija.Naziv AS Lokacija
FROM Odrzavanje
INNER JOIN Tecaj
ON TecajId = Tecaj.Id
INNER JOIN Lokacija
ON LokacijaId = Lokacija.Id
```

12.3. Korištenje pogleda

Pogled možemo koristiti na isti način kao i običnu tablicu. Možemo raditi dohvat podataka iz pogleda, dodatno filtriranje nad rezultatima pogleda, koristiti pogled u spoju, podupitu i tako dalje.

Sljedeći upit daje primjer upotrebe pogleda stvorenog u prethodnom primjeru. Ako se želi ispisati raspored tečajeva za određeni mjesec dovoljno je napisati sljedeće:

```
SELECT *
FROM Raspored
WHERE MONTH(Pocetak) = 1
      AND YEAR(Pocetak) = 2016
```

Ako uz podatke koji već postoje u pogledu *Raspored* želimo ispisati i ime predavača, možemo spojiti pogled *Raspored* s tablicom *Predavac*. Pri tome se pogled ponaša isto kao i bilo koja tablica - spajanje pogleda s tablicom je potpuno istovjetno spajanju dviju tablica.

```
SELECT Raspored.*, Ime, Prezime
FROM Raspored
INNER JOIN Predavac
ON PredavacId = Predavac.Id
```

12.4. Brisanje pogleda

Pogled se briše iz baze podataka pomoću naredbe `DROP VIEW` navođenjem naziva pogleda (slično kao što se tablica briše pomoću naredbe `DROP TABLE`).

```
DROP VIEW Pogled
```

U sljedećem primjeru briše se pogled *Raspored*.

```
DROP VIEW Raspored
```

13. Indeksi

Po završetku ovog poglavlja polaznik će moći:

- razumjeti koja je svrha indeksa
- stvoriti indeks nad jednim stupcem
- stvoriti indeks nad više stupaca
- stvoriti `UNIQUE` indeks
- obrisati indeks iz baze podataka.

Ovo poglavlje objašnjava što su to indeksi, čemu služe i kada ih koristiti.

13.1. Indeksi

Indeks je struktura u bazi podataka koja sadrži podatke o raspodjeli vrijednosti pojedinog stupca ili stupaca u tablici, zapisanih na način koji pogoduje pretraživanju. Namjena indeksa je ubrzati pronalaženje redaka u kojima se nalazi tražena vrijednost.

Indeks se postavlja na pojedini stupac u tablici, tako da se ubrza pretraživanje te tablice po vrijednosti tog stupca.

Kod većine sustava za upravljanje bazama podataka automatski se stvara indeks nad stupcem koji je primarni ključ tablice. Zbog toga je pronalaženje retka u tablici preko vrijednosti primarnog ključa uvijek brzo.

Ako nad stupcem postoji indeks, dodavanjem novih vrijednosti u tablicu ili ažuriranjem postojećih i indeks se automatski ažurira. Ako je broj redaka velik, to može usporiti dodavanje ili ažuriranje retka.

Potrebno je dobro odabrati nad kojim stupcima treba biti indeks. Evo nekoliko preporuka vezanih uz stvaranje indeksa:

- Indeks nije potrebno postaviti ako tablica ima mali broj redaka (do nekoliko stotina). U tom slučaju brže je sekvencijalno pretraživanje.
- Indekse nije uputno postaviti nad stupcima čiji je raspon vrijednosti malen – npr. na stupcu *Spol* u tablici *Polaznik* koji može poprimiti samo dvije moguće vrijednosti.
- Indeks je preporučljivo postaviti na stupcima preko kojih se vrši spajanje tablica, što su u većini slučajeva strani ključevi.
- Indeks je preporučljivo postaviti na stupcima po čijoj se vrijednosti često pretražuje.

Na tablicama s vrlo velikim brojem podataka, pametno postavljeni indeksi donose dramatična ubrzanja u izvršavanju upita.

Kod tablica i baza koje imaju mali broj podataka, stvaranje indeksa najčešće nije potrebno.

13.2. Stvaranje indeksa

Indeks se stvara pomoću naredbe `CREATE INDEX`, uz navođenje tablice i stupca koji se indeksira. Prilikom stvaranja, indeksu je potrebno zadati i naziv:

```
CREATE INDEX Indeks
ON Tablica (Stupac)
```

Naziv indeksa treba biti jedinstven unutar jedne tablice.

Sljedeća naredba postavit će indeks nad stupcem *PolaznikId* u tablici *Pohadjanje*:

```
CREATE INDEX Indeks_PolaznikId
ON Pohadjanje(PolaznikId)
```

Postojanje ovog indeksa ubrzat će pretraživanje tablice *Pohadjanje* po stupcu *PolaznikId*:

```
SELECT *
FROM Pohadjanje
WHERE PolaznikId = 99
```

Postojanje ovog indeksa također će ubrzati i ovaj upit u kojem se spajaju tablice *Polaznik* i *Pohadjanje* preko stupca *PolaznikId*:

```
SELECT *
FROM Polaznik
INNER JOIN Pohadjanje
ON Polaznik.Id = Pohadjanje.PolaznikId
```

13.3. Stvaranje indeksa nad više stupaca

Indeks se može postaviti i nad više stupaca, ali je bitno je znati da se time ne ubrzava pretraživanje po pojedinim vrijednostima, nego samo po kombinaciji vrijednosti.

Prilikom stvaranja indeksa navode se svi stupci koji su dio indeksa: Indeks se stvara pomoću naredbe `CREATE INDEX`, uz navođenje tablice i stupca koji se indeksira. Pri stvaranju, indeksu je potrebno zadati i naziv:

```
CREATE INDEX Indeks
ON Tablica (Stupac1, Stupac2)
```

U sljedećem primjeru indeks bi se postavio nad kombinacijom vrijednosti *PredavacId* i *TecajId* u tablici *Odrzavanje*:

```
CREATE INDEX Indeks_PredavacId_TecajId
ON Odrzavanje(PredavacId, TecajId)
```

Ovakav bi indeks imao smisla u slučaju da su česti upiti nad tablicom *Odrzavanje* u kojima se traži redak za poznati *PredavacId* i *TecajId*.

```
SELECT *
FROM Odrzavanje
WHERE PredavacId = 15
AND TecajId = 25
```

13.4. Stvaranje UNIQUE indeksa

Kod stupaca za koje se želi osigurati da ne mogu biti unesena dva retka s istom vrijednošću preporučljivo je postaviti `UNIQUE` indeks. Ovaj indeks postavlja se dodavanjem ključne riječi `UNIQUE` u naredbu `CREATE INDEX`.

`UNIQUE` indeks osigurat će da ne mogu postojati duplikati, a dodatno će biti ubrzano pretraživanje po tom stupcu.

```
CREATE UNIQUE INDEX Indeks  
ON Tablica (Stupac)
```

Dobar kandidat za ovaj indeks bio bi stupac `OIB` u tablici `Predavac`.

```
CREATE UNIQUE INDEX Indeks_OIB  
ON Predavac(OIB)
```

Na ovaj način bit će onemogućen unos dva predavača s istim `OIB`-om, a i pretraživanje po `OIB`-u će biti brzo.

13.5. Brisanje indeksa

Indeks se briše iz baze podataka pomoću naredbe `DROP INDEX` navođenjem naziva pogleda (slično kao što se tablica briše pomoću naredbe `DROP TABLE`).

```
DROP INDEX Indeks  
ON Tablica
```

U sljedećem primjeru briše se indeks `Indeks_PolaznikId` s tablice `Pohadjanje`.

```
DROP INDEX Indeks_PolaznikId  
ON Pohadjanje
```

13.6. Vježba: Podupiti i operacije nad skupovima

1. Uz svaku lokaciju ispišite broj održavanja tečajeva na toj lokaciji.
2. Ispišite sve lokacije na kojima su održavani tečajevi.
3. Uz svaki tečaj ispišite broj puta koliko je tečaj održan.
4. Uz svaki tečaj ispišite ukupan broj polaznika koji su pohađali taj tečaj.
5. Ispišite naziv tečaja i datum održavanja za sva održavanja na kojima je bilo prisutno više od minimalnog broja polaznika.
6. Ispišite polaznike koji su pohađali više od 2 tečaja.
7. Ispišite polaznike koji nemaju ni jedan izostanak. (Prisutnost se bilježi pomoću stupca *Prisutan* u tablici *Dolazak*).
8. Ispišite tečajeve čija je prosječna ocjena veća od 4,75. Potrebno je računati prosječnu vrijednost stupca *OcjenaTecaja* u tablici , no prije računanja prosjeka potrebno je pretvoriti tu vrijednost u decimalni tip podatka (`decimal`).
9. Uz svaki tečaj ispišite broj godina u kojima se tečaj održava. (Upotrijebite agregatnu funkciju `COUNT` uz ključnu riječ `DISTINCT`.)
10. Napravite uniju između tablice *Ustanova* i onih redaka iz tablice *Lokacija* za koje je stupac *NadredjenaLokacijaId* jednak `NULL`. Napravite i uniju s ponavljanjem i usporedite dobiveni broj redaka.

14. Dodatak

14.1. Pregled sintakse naredbe `SELECT`

Radi ponavljanja, ali i za referencu polazniku, u nastavku se nalazi pregled sintakse naredbe `SELECT` koji uključuje sve klauzule ove naredbe i njihov redoslijed.

```
SELECT Stupac1, Stupac2, Stupac3...
FROM Tablica1
INNER/LEFT OUTER JOIN Tablica2
ON UvjetSpajanja1
INNER/LEFT OUTER JOIN Tablica3
ON UvjetSpajanja2
WHERE Uvjet1 AND/OR Uvjet2
GROUP BY Stupac1, Stupac2...
HAVING Uvjet3 AND/OR Uvjet4
ORDER BY Stupac1 ASC/DESC,
         Stupac2 ASC/DESC...
```

Prva dolazi klauzula `SELECT`, u kojoj se navodi lista stupaca koji se selektiraju. Ova lista naziva se još i listom za odabir. Osim stupaca iz tablice, u njoj je moguće navesti i složeni izrazi, ali i podupite.

Ako se žele prikazati svi stupci, moguće je umjesto navođenja liste stupaca samo navesti zvjezdicu (*).

Zatim dolazi klauzula `FROM`, u kojoj se navodi tablica iz koje se selektiraju podaci. Na mjestu tablice može se nalaziti i pogled, ali i podupit (unutar zagrada).

Sve druge klauzule opcionalne su, odnosno mogu se, ali i ne moraju navesti (klauzula `FROM` je također opcionalna, međutim upiti bez nje nemaju puno smisla).

Ako se dohvat podataka radi iz više tablica, za svaku dodatnu tablicu potrebno je navesti po jednu klauzulu `JOIN` i `ON`.

U klauzuli `JOIN` navodi se tablica (ili pogled ili podupit) s kojom se vrši spajanje. Spoj će najčešće biti unutrašnji ili lijevi (vanjski). Desni i puni vanjski spoj rjeđe se upotrebljavaju, pa radi preglednosti nisu navedeni.

Kod unutrašnjeg spoja može se, radi jasnoće, dodatno navesti ključna riječ `INNER` prije ključne riječi `JOIN`. Kod lijevog spoja obavezno se navodi ključna riječ `LEFT`, dok je ključna riječ `OUTER` opcionalna.

U klauzuli `ON` navodi se uvjet spajanja. U najvećem broju slučajeva radit će se o spajanju po jednakosti, i to po stupcu koji je strani ključ u jednoj, a primarni ključ u drugoj tablici.

Bitno je zapamtiti da **sve druge klauzule mogu doći tek nakon klauzule `FROM` i nakon klauzula `JOIN` i `ON`** ako su one prisutne, dakle tek nakon što je navedeno iz kojih se tablica selektiraju podatci.

Ako je potrebno napraviti određeno filtriranje rezultata, potrebno je navesti klauzulu `WHERE`. Dakle, klauzula `WHERE` uvijek dolazi tek nakon klauzule `FROM` i nakon klauzula `JOIN` i `ON` ako su one prisutne.

U klauzuli `WHERE` navodi se jedan ili kombinacija više uvjeta koje retci moraju zadovoljiti kako bi se našli u konačnom rezultatu. U klauzuli `WHERE` moguće je navoditi i podupite.

Ako je potrebno napraviti grupiranje rezultata, koristi se klauzula `GROUP BY`. Grupiranje dolazi tek nakon što su navedene tablice iz kojih dolaze podatci i nakon što je navedeno filtriranje, dakle nakon klauzula `FROM`, `JOIN`, `ON` i `WHERE`. U upitu s grupiranjem klauzule `JOIN`, `ON` i `WHERE` mogu se izostaviti, ali su klauzule `SELECT` i `FROM` obavezne.

Kod upita s grupiranjem u listi za odabir moguće je navesti samo stupce po kojima se grupira, ili agregatne funkcije.

Klauzula `HAVING` služi da bi se postavio uvjet na rezultate grupiranja. Ova klauzula mora doći poslije klauzule `GROUP BY` (i ne može doći bez nje). U uvjetu klauzule `HAVING` može se koristiti samo stupac po kojem se grupira, odnosno rezultat agregatne funkcije.

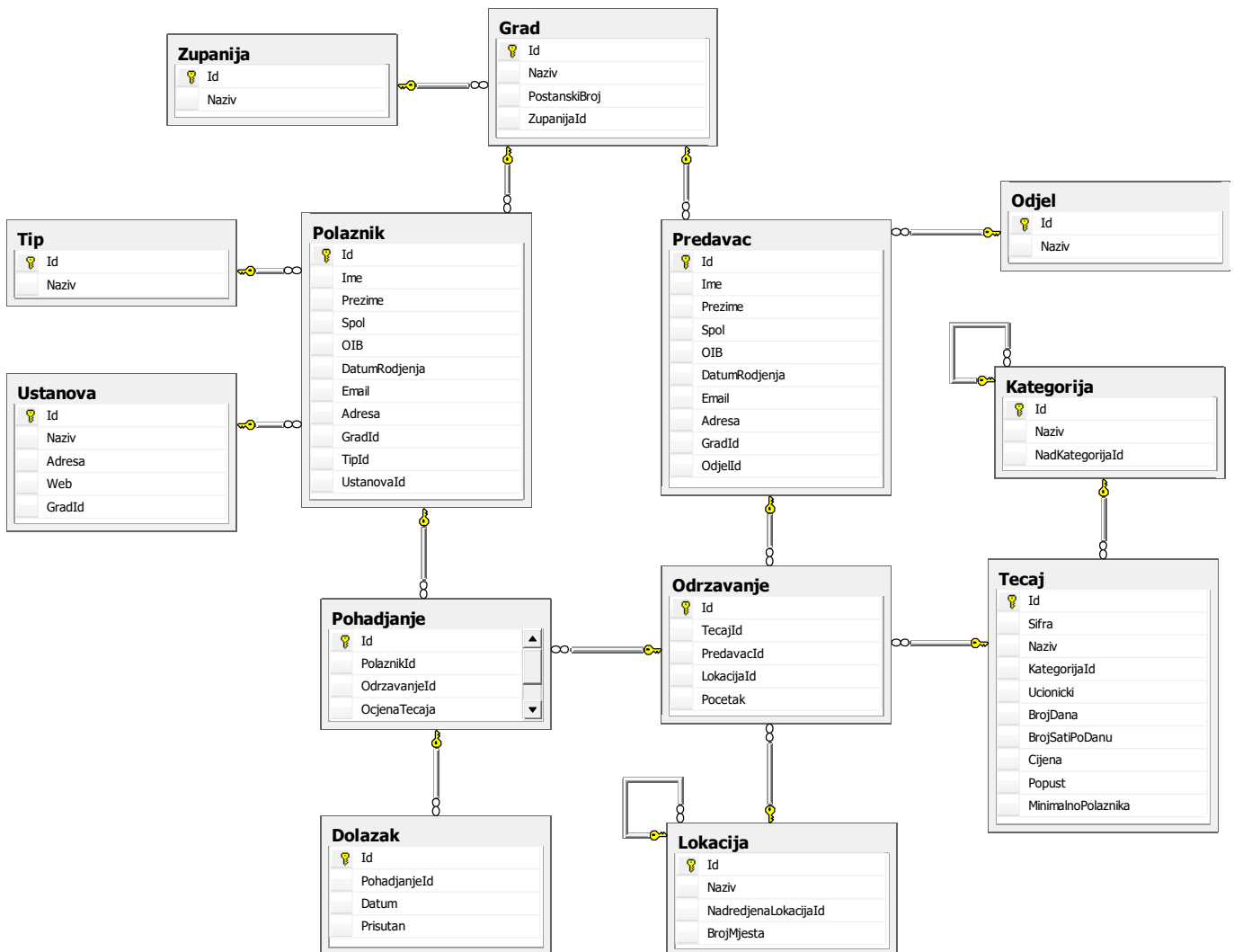
Na kraju, retke iz rezultata moguće je poredati. **Klauzula `ORDER BY` dolazi posljednja u naredbi `SELECT`.**

U klauzuli `ORDER BY` navode se stupci (ili izrazi) po kojima se želi poredati rezultat. Smjer poretka određuje se pomoću ključnih riječi `ASC` (uzlazni poredak, tj. od najmanje prema najvećoj vrijednosti) ili `DESC` (silazni poredak). Ako se ne navede smjer poretka, pretpostavlja se uzlazni poredak (`ASC`).

Zanimljivosti i napomene

Postoji iznimka od ovog pravila. Poslije klauzule `ORDER BY` može doći klauzula za straničenje rezultata koja nije dio SQL standarda i razlikuje se na različitim sustavima.

14.2. Dijagram baze podataka



14.3. Instalacija Microsoft SQL Server LocalDB sustava

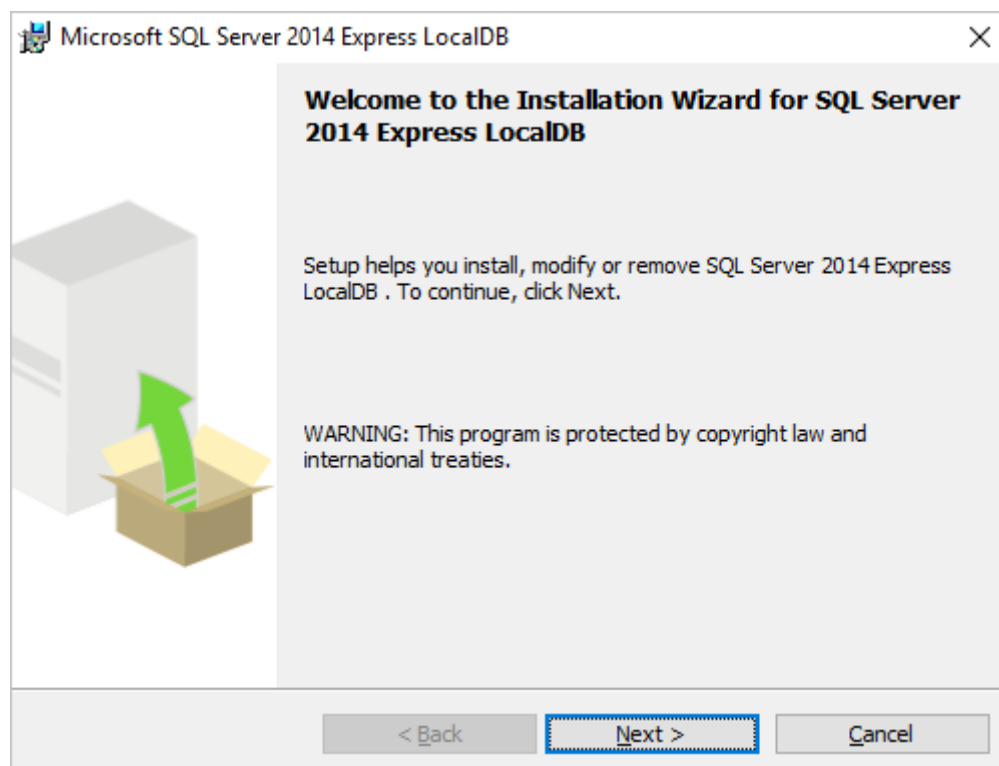
Za učenje SQL-a i drugu upotrebu na vlastitom računalu preporuča se instalacija LocalDB verzije Microsoft SQL Servera, koja je namijenjena za lokalnu upotrebu i zahtijeva manje resursa od poslužiteljske SQL Server Express verzije.

Na Microsoftovoj stranici za preuzimanje odaberite **SQL Server LocalDB Express** – 64 bitnu ili 32 bitnu verziju, ovisno o vašem računalu.

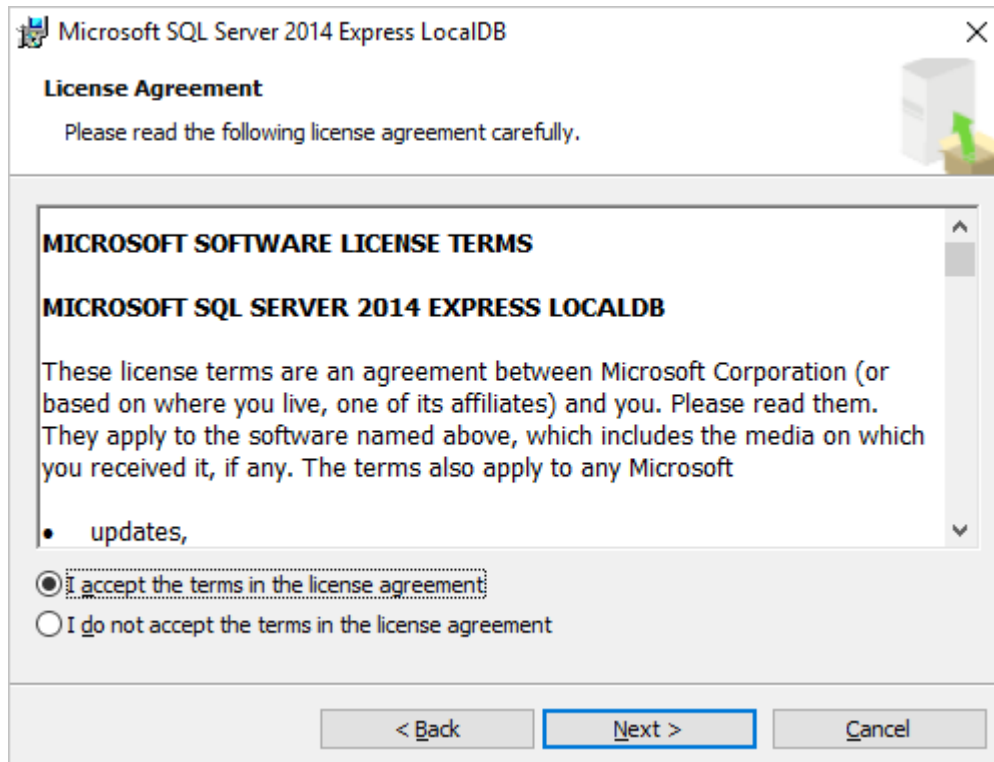
* **Please select the version of SQL Server 2014 Express you'd like to download**

- SQL Server 2014 LocalDB Express 32bit
- SQL Server 2014 LocalDB Express 64 Bit
- SQL Server 2014 Express 32 Bit
- SQL Server 2014 Express 64 Bit
- SQL Server 2014 Express with Tools 32 Bit
- SQL Server 2014 Express with Tools 64 Bit
- SQL Server 2014 Management Studio Express 32 Bit
- SQL Server 2014 Management Studio Express 64 Bit
- SQL Server 2014 Express with Advanced Services 32 Bit
- SQL Server 2014 Express with Advanced Services 64 Bit

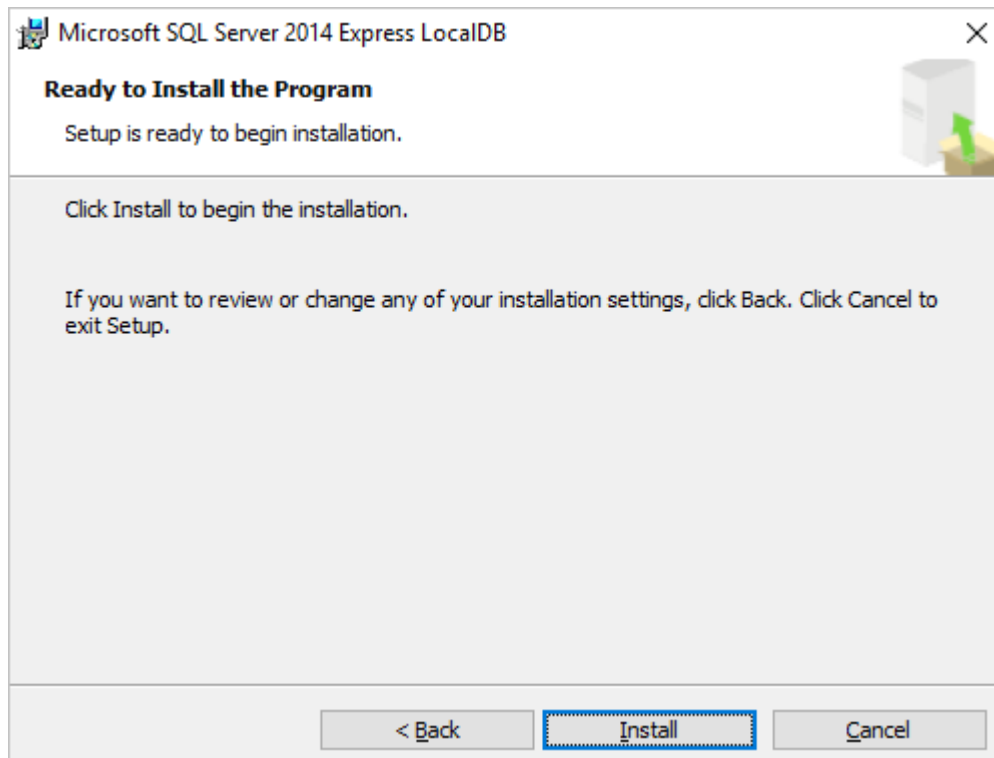
Nakon preuzimanja, pokrenite *.msi* datoteku. Pojavit će se sljedeći ekran, na kojem kliknite *Next*.



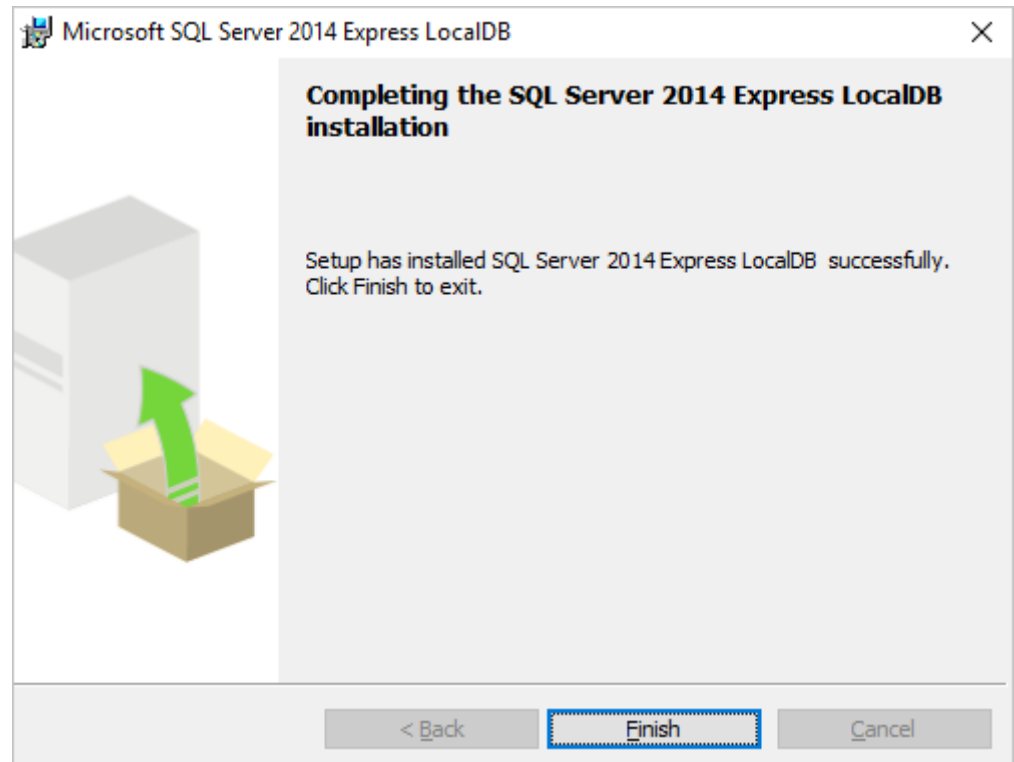
Odaberite *I accept...* i kliknite na *Next*.



Na sljedećem ekranu kliknite na *Install*.



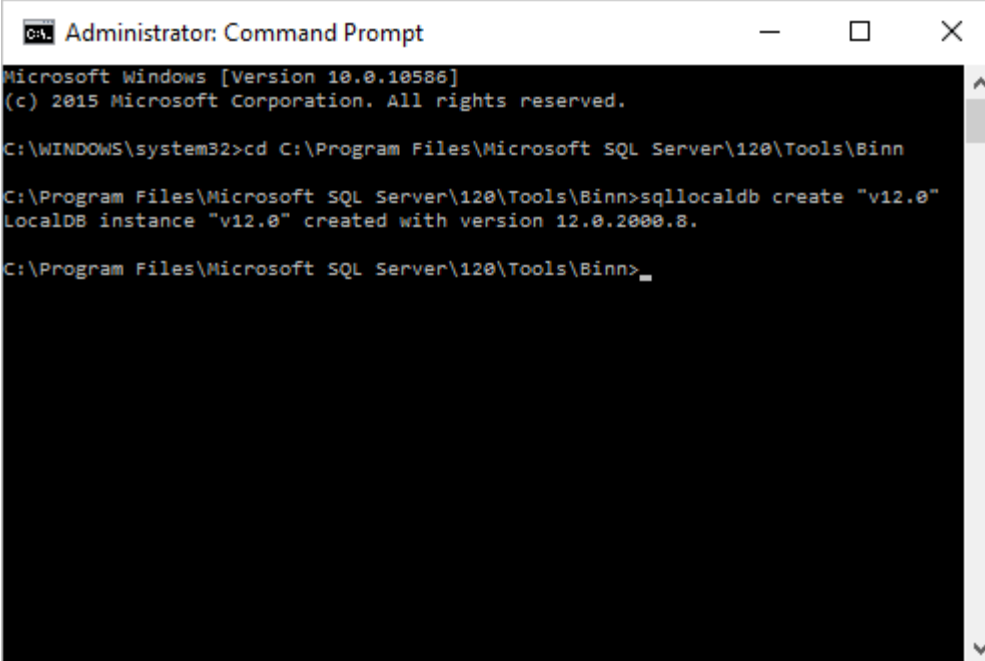
Pričekajte da se izvrši instalacija i kliknite na *Finish*.



Ako se radi o verziji SQL Server **2014** LocalDB, potreban je još i ovaj korak:

U komandnom retku pozicionirajte se u direktorij u kojem se nalazi alat *SqLocalDb.exe* (tipično *C:\Program Files\Microsoft SQL Server\120\Tools\Binn*). Pokrenite sljedeću naredbu kako bi se stvorila nova LocalDB instanca pod imenom *v12.0*:

```
sqllocaldb create "v12.0"
```



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Program Files\Microsoft SQL Server\120\Tools\Binn

C:\Program Files\Microsoft SQL Server\120\Tools\Binn>sqllocaldb create "v12.0"
LocalDB instance "v12.0" created with version 12.0.2000.8.

C:\Program Files\Microsoft SQL Server\120\Tools\Binn>
```

14.4. Instalacija Microsoft SQL Server Management Studio

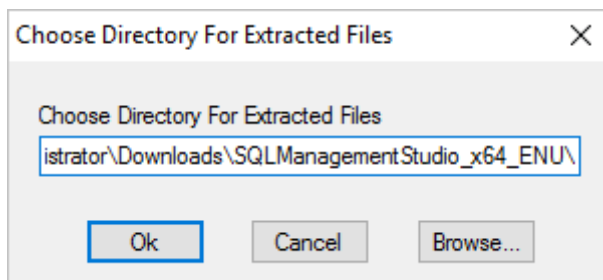
Nakon što je instaliran sustav za upravljanje bazama podataka Microsoft SQL Server Local DB, potrebno je instalirati i korisnički alat za spajanje na bazu, odnosno Microsoft SQL Server Management Studio.

Na Microsoftovoj stranici za preuzimanje odaberite **SQL Server Management Studio Express** – 64 bitnu ili 32 bitnu verziju, ovisno o vašem računalu.

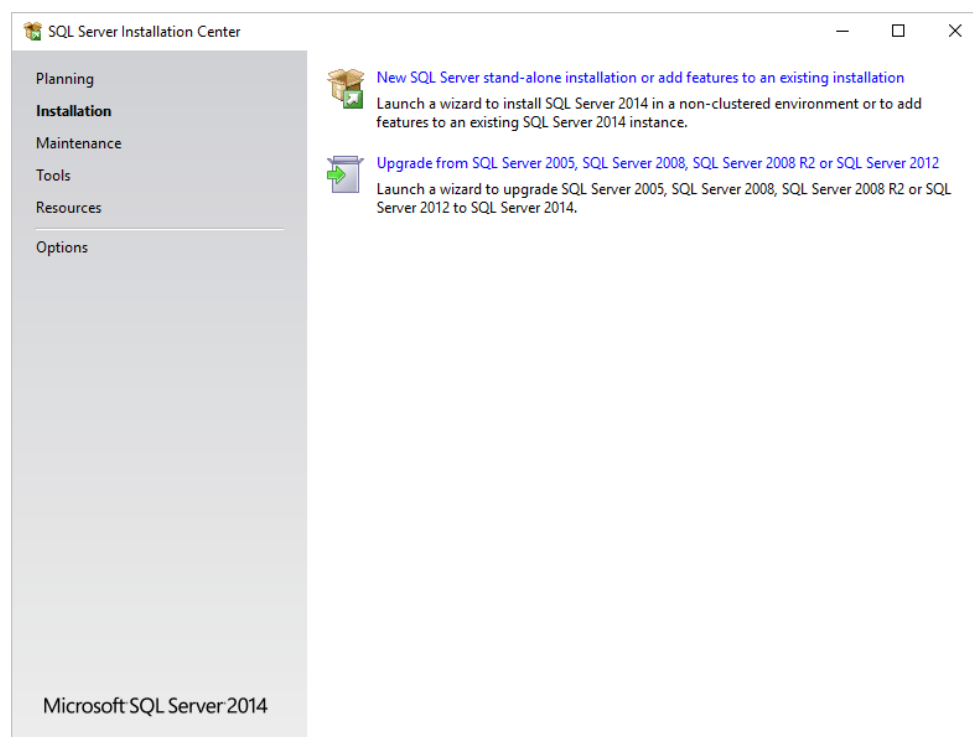
* **Please select the version of SQL Server 2014 Express you'd like to download**

- SQL Server 2014 LocalDB Express 32bit
- SQL Server 2014 LocalDB Express 64 Bit
- SQL Server 2014 Express 32 Bit
- SQL Server 2014 Express 64 Bit
- SQL Server 2014 Express with Tools 32 Bit
- SQL Server 2014 Express with Tools 64 Bit
- SQL Server 2014 Management Studio Express 32 Bit
- SQL Server 2014 Management Studio Express 64 Bit
- SQL Server 2014 Express with Advanced Services 32 Bit
- SQL Server 2014 Express with Advanced Services 64 Bit

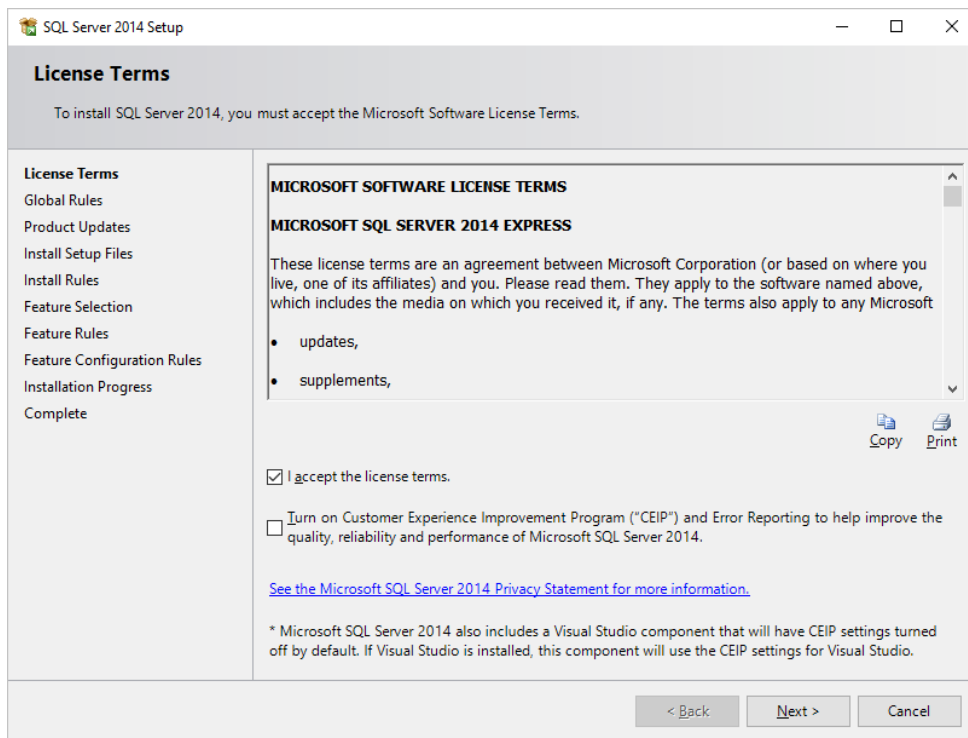
Nakon preuzimanja, pokrenite .exe datoteku. Pojavit će se sljedeći ekran, na kojem možete odabrati direktorij za spremanje instalacijskih datoteka. Nakon odabira direktorija kliknite na *OK*.



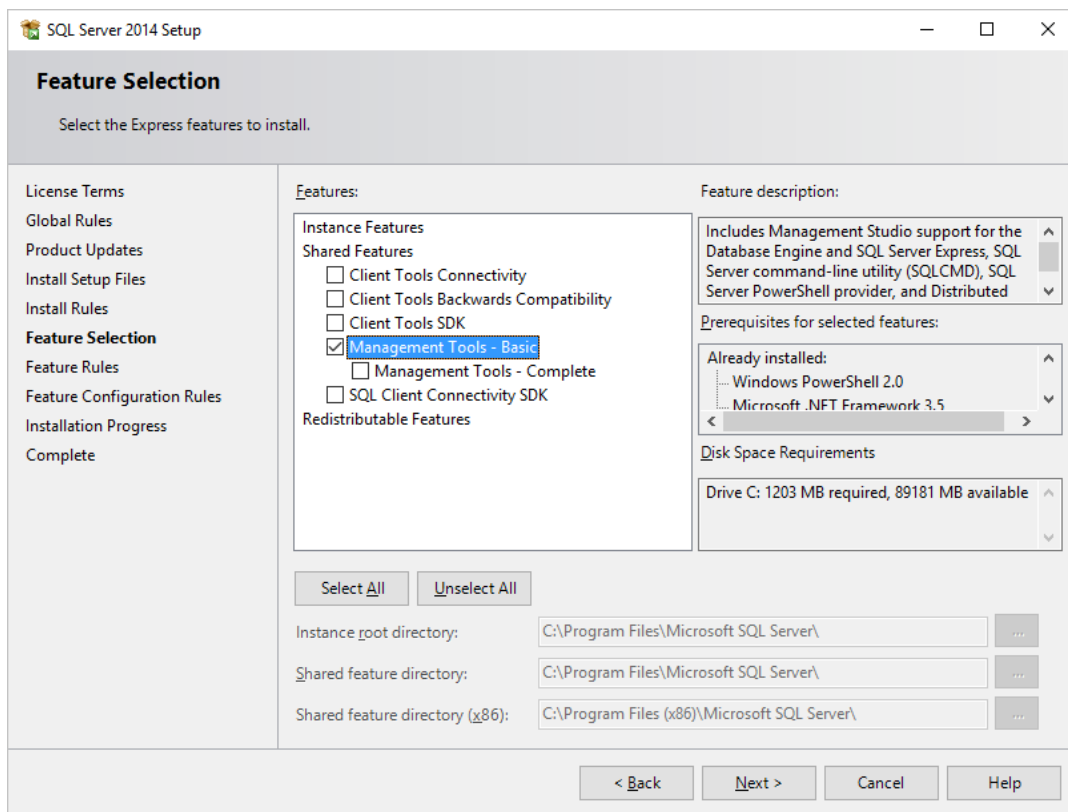
Na sljedećem ekranu kliknite na opciju *New SQL Server stand-alone installation or add features to an existing installation*.



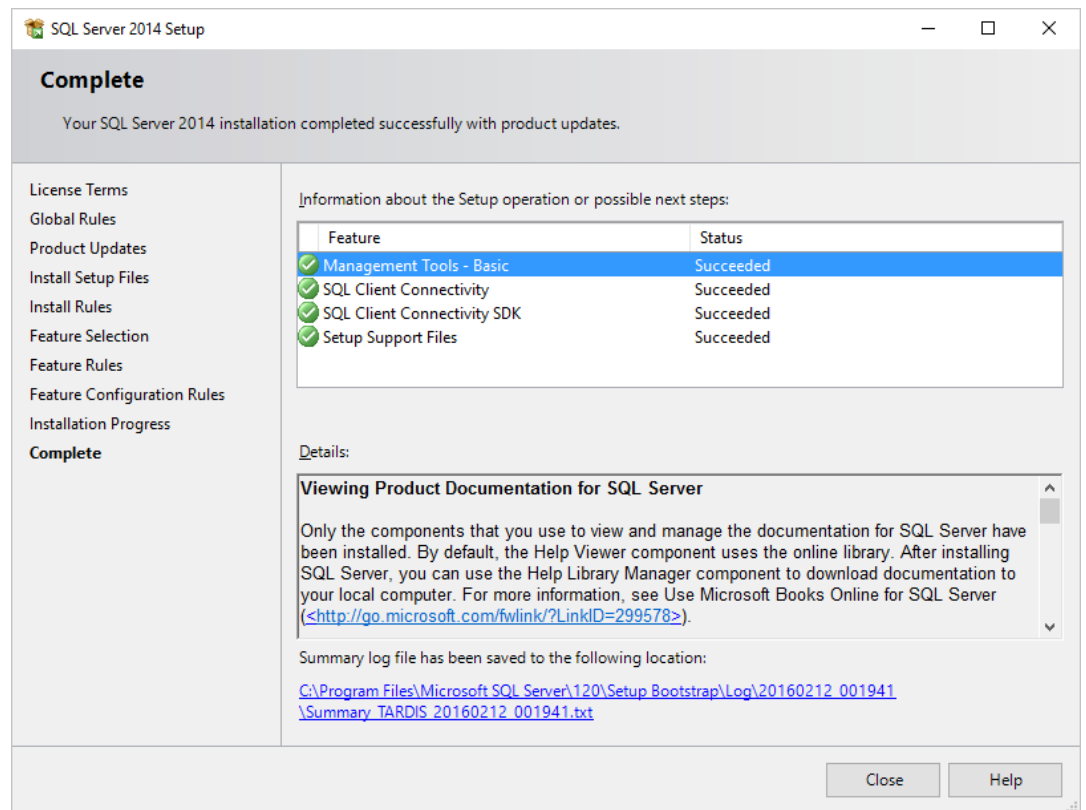
Na sljedećem ekranu odaberite *I accept the license terms* i kliknite na *Next*.



Na sljedećem ekranu ostavite označenu samo opciju Management Tools – Basic i kliknite na Next.



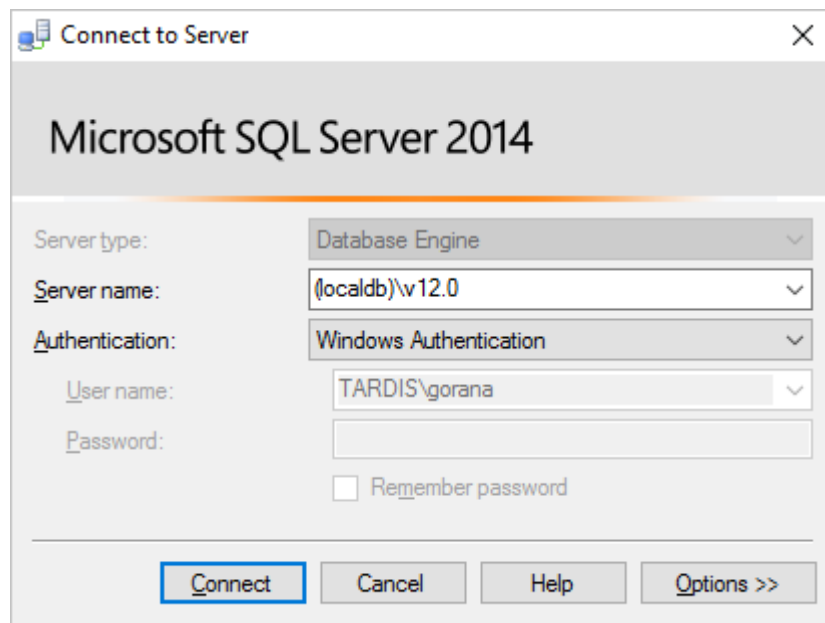
Na završnom ekranu kliknite na *Close*.



Drugi prozor iz ranijeg koraka koji je ostao otvoren možete sada zatvoriti.

Aplikacija SQL Server Management Studio sada je instalirana te je možete pokrenuti.

U dijalogu za spajanje na poslužitelj pod *Server Name* upišite naziv LocalDb instance, što je `(localdb)\v12.0`. Nakon toga kliknite na *Connect*.



Bilješke: