

# UVOD U BAZE PODATAKA

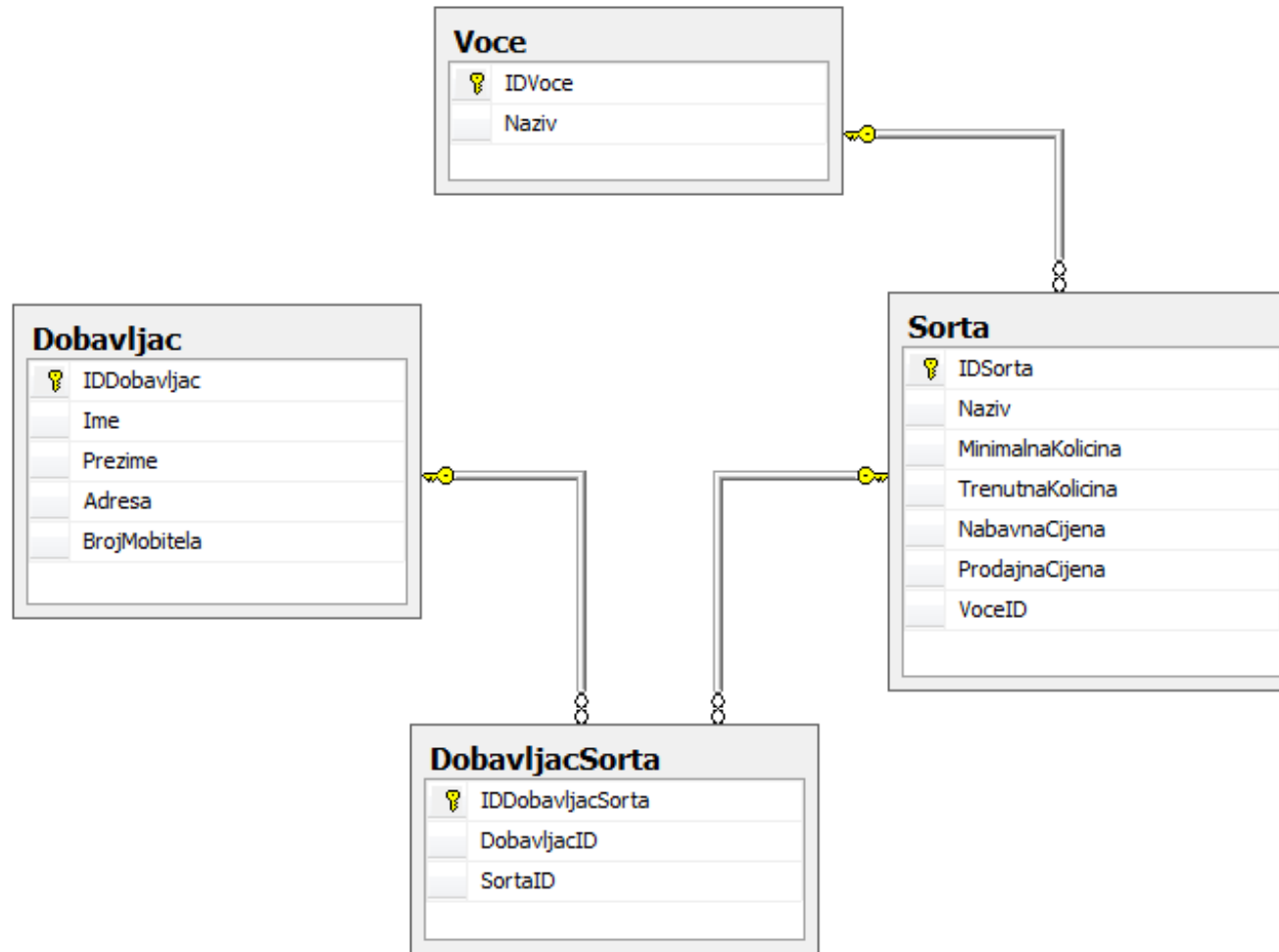
Predavanje 5

# Ponavljjanje

# Pretvaranje ER modela u relacijski model

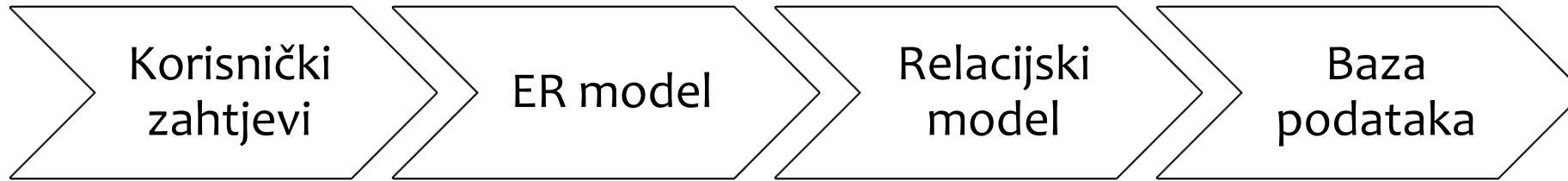
- Prvo pretvaramo entitete, a onda odnose
- Svaki **entitet** pretvaramo u **tablicu**
- Kod **odnosa** gledamo broj instanci i članstvo
  - Za **1:1** imamo opcije:
    - Ako su oba obavezna entiteta, svejedno je u koju tablicu dodajemo strani ključ
    - Ako postoji jedan obavezni entitet, u njegovu tablicu dodamo strani ključ
    - Ako ne postoji, radimo novu tablicu
  - Za **1:N** i **N:1** gledamo entitet uz N
    - Ako je obavezan, dodamo strani ključ u njegovu tablicu
    - Ako nije obavezan, radimo novu tablicu
  - Za **M:N** uvijek radimo novu tablicu

# Primjer relacijskog modela



# Uvod u SQL

# Gdje se nalazimo?



- Imamo relacijski model
- Želimo ga implementirati u RDBMS-u po izboru

# Nastanak SQL-a

- **SQL** (engl. *Structured Query Language*) je jezik baziran na skupovima
- Služi za rad s podacima u relacijskim bazama te za upravljanje tim bazama
- U praksi nerazdvojan od RDBMS-a
- Prva verzija SQL-a:
  - Razvijena početkom 70-ih u IBM-u na osnovi relacijskog modela
  - Razvijena za eksperimentalni RDBMS zvan **System R**
  - Nazvana **SEQUEL** (engl. *Structured English **Q**Uery Language*)
  - Zbog autorskih prava kasnije naziv skraćen na SQL

# Razvoj SQL-a

- Krajem 70-ih Oracle je predstavio svoj **Oracle v2**
- SQL usvojen kao standard:
  - 1986. ANSI (engl. *American National Standards Institute*)
  - 1987. ISO (engl. *International Standards Organization*)
- Najnoviji standard je SQL:2016 iz 2016. godine



# Implementacije SQL standarda

- Na osnovu standarda nastaju konkretne **implementacije**:
  - Proširuju standard
  - Ne implementiraju ga u potpunosti
  - Ne poštuju dijelove standarda (primjerice, rad s datumima)
- Primjeri implementacija:
  - **T-SQL (Microsoft, Sybase) – Transact SQL**
  - SQL/PSM (MySQL) – SQL/Persistent Stored Modules
  - PL/SQL (Oracle) - Procedural Language for SQL
  - SQL PL (IBM) – SQL Procedural Language
- **Posljedica**: netrivialni upiti za jedan RDBMS se ne mogu koristiti na drugom RDBMS-u

# Dijelovi SQL-a

- Tradicionalno, SQL se dijeli na:
  - **DDL** (engl. *Data Definition Language*)
    - Sastoji se od naredbi za stvaranje, mijenjanje i uklanjanje objekata iz baze (primjerice, tablica)
  - **DML** (engl. *Data Manipulation Language*)
    - Sastoji od naredbi za rad s podacima u bazi
    - Glavne naredbe služe za dohvaćanje, umetanje, izmjenu i brisanje podataka
  - **TCL** (engl. *Transaction Control Language*)
    - Sastoji od naredbi za upravljanje transakcijama
  - **DCL** (engl. *Data Control Language*)
    - Sastoji od naredbi vezanih uz sigurnost

# T-SQL tipovi podataka (1/3)


- Svaki **stupac** (i varijabla i parametar) ima svoj **tip podataka**
- Tipovi podataka ovise o RDBMS-u, **SQL Server** osnovni tipovi podataka su:
  - Cijeli brojevi:

**bigint** – s predznakom

od  $-2^{63}$  (-9,223,372,036,854,775,808)

do  $2^{63-1}$  (9,223,372,036,854,775,807)

**int** – s predznakom

od  $-2^{31}$  (-2,147,483,648) 

do  $2^{31-1}$  (2,147,483,647)

**smallint** – s predznakom

od  $-2^{15}$  (-32,768)

do  $2^{15-1}$  (32,767)


**tinyint** – 0 to 255 – bez predznaka

# T-SQL tipovi podataka (2/3)

- Decimalni brojevi s gubitkom:

**float** – realni broj s pomičnom točkom

- Decimalni brojevi bez gubitka:

**decimal(a,b)** – realni broj 

- *a* definira ukupan broj znamenki (najviše 38)
- *b* definira broj decimalnih mjesta

**money** – sinonim za **decimal(19,4)** 

# T-SQL tipovi podataka (2/3)

- Datum i vrijeme:

**date** 0001-01-01 do 9999-12-31

**time** 00:00:00.0000000 do 23:59:59.9999999



**smalldatETIME** 1900-01-01 do 2079-06-06 / 00:00:00 do 23:59:59



**datetime** January 1, 1753 - December 31, 9999 / 00:00:00 - 23:59:59.997

**datetime2** 0001-01-01 do 9999-12-31 / 00:00:00 do 23:59:59.9999999

# T-SQL tipovi podataka (3/3)

- Stringovi:
  - `char(n)` – string ASCII znakova fiksne duljine,  $n \leq 8000$
  - `nchar(n)` – string Unicode znakova fiksne duljine,  $n \leq 4000$
  - `varchar(n)` – kao `char`, samo je duljina varijabilna
  - `nvarchar(n)` – kao `nchar`, samo je duljina varijabilna 
  - `varchar(MAX)` – kao `varchar`, najviše do  $2^{31}-1$  bajtova
  - `nvarchar(MAX)` – kao `nvarchar`, najviše do  $2^{30}-1$  bajtova
- Ostali tipovi:
  - `bit` – sadržava 0 (laž) ili 1 (istina) 

# Napomene

- Naredbe i tipovi podataka neosjetljivi na velika i mala slova
  - SELECT, select, Select, SeLeCt, ...
  - Naredbe ćemo pisati velikim, a tipove podataka malim slovima
- Nazivi objekata **mogu, ali ne moraju** biti neosjetljivi na velika i mala slova
  - ZAPOSLENIK, zaposlenik, Zaposlenik, ...
  - **Objekte ćemo uvijek pisati onako kako su kreirani**
- SQL upiti su neosjetljivi na razmake i prelaske u novi red



# Implementacija relacijskog modela



# Izrada baze podataka

- Prvi korak u implementaciji relacijskog modela je izrada baze podataka u odabranom RDBMS-u
  - Mi ćemo koristiti SQL Server
- Bazu podataka izrađujemo naredbom **CREATE DATABASE** i zadavanjem njenog naziva  
`CREATE DATABASE MojaPrvaBaza`
- Nova baza se smješta u podrazumijevanu mapu SQL Servera
  - Naredbom moguće definirati drukčiji smještaj
- Bazu podataka uklanjamo naredbom **DROP DATABASE**  
`DROP DATABASE MojaPrvaBaza`
  - **Ne pita "Are you sure?"**

# Izrada tablica

- Nakon izrade baze, izrađujemo u njoj tablice
- Struktura tablice se sastoji od stupaca i za svaki biramo:
  - Naziv
  - Tip podataka
  - Nula ili više **ograničenja** (engl. *constraints*) koja mogu biti:
    - Primarni ključ (engl. *primary key constraint*) 
    - Strani ključ (engl. *foreign key constraint*) 
    - Obaveznost unosa (engl. *not null constraint*)
    - Provjera (engl. *check constraint*)
    - Jedinственost (engl. *unique constraint*)
    - Podrazumijevana vrijednost (engl. *default constraint*)

# Izrada tablice bez ograničenja

- Za izradu svake tablice koristimo naredbu **CREATE TABLE**
  - Za svaki stupac navodimo **naziv** i **tip podataka**
  - Definicije stupaca odvajamo **zarezima**

- Primjer:

```
CREATE TABLE Student  
(  
  Ime nvarchar(50),  
  Prezime nvarchar(50),  
  Redovni bit,  
  DatumRodjenja datetime  
)
```

Naziv stupca

Tip podataka

# Ograničenja

- Ograničenje služi boljem definiranju domene stupca
  - Primjerice, kako ćemo osigurati da se u stupac Ocjena mogu upisati samo vrijednosti od 1 do 5? Ograničenjem provjere!
- Ograničenje se može odnositi na jedan ili na više stupaca
  - Na svaki stupac se može odnositi nula ili više ograničenja
- Postoji više načina kako možemo dodavati ograničenja na stupce
  - Mi ćemo za svako ograničenje naučiti jedan način
  - Na vježbama ćete upoznati i neke druge načine
  - Svejedno koji način ćete koristiti
- Neka nazivi ograničenja budu jedinstveni u bazi

# Dodavanje ograničenja primarnog ključa

- Primarni ključ definiramo na sljedeći način:
  - Definiciju smještamo iza naziva stupca i tipa podataka
  - Navodimo naredbu **CONSTRAINT** i dajemo naziv ograničenja
  - Na kraju pišemo naredbu **PRIMARY KEY** kojom označavamo o kakvom ograničenju se radi

- Primjer:

```
CREATE TABLE Student
(  
  IDStudent int CONSTRAINT PK_Student PRIMARY KEY,  
  Ime nvarchar(50),  
  Prezime nvarchar(50)  
)
```

# Surogatni primarni ključ

- RDBMS nudi mogućnost automatskog generiranja vrijednosti primarnog ključa
- Iza definicije primarnog ključa navodimo naredbu **IDENTITY**
- Primjer:

```
CREATE TABLE Student
(
  IDStudent int CONSTRAINT PK_Student PRIMARY KEY IDENTITY,
  Ime nvarchar(50),
  Prezime nvarchar(50)
)
```

- Ako je primarni ključ označen kao IDENTITY, korisnik ne može upisivati niti mijenjati vrijednosti tog stupca
  - Upisivanje vrijednosti radi RDBMS pri svakom umetanju retka
  - Promjena vrijednosti nikako nije moguća

# Dodavanje ograničenja stranog ključa

- Strani ključ definiramo na sličan način:
  - Definiciju smještamo iza naziva stupca i tipa podataka
  - Navodimo naredbu **CONSTRAINT** i naziv ograničenja
  - Pišemo naredbu **FOREIGN KEY REFERENCES** iza koje pišemo naziv ciljne tablice i u zagradi naziv ciljnog stupca

- Primjer:

```
CREATE TABLE Grad
(
  IDGrad int CONSTRAINT PK_Grad PRIMARY KEY,
  Naziv nvarchar(50),
  DrzavaID int CONSTRAINT FK_Grad_Drzava
    FOREIGN KEY REFERENCES Drzava(IDDrzava)
)
```

# Dodavanje ograničenja obaveznosti unosa

- **NULL vrijednošću** nazivamo nepostojeću ili nepoznatu vrijednost
  - Primjerice, ako za neku osobu ne znamo datum rođenja, za njega ćemo u stupac DatumRodjenja upisati NULL vrijednost
- Za svaki stupac možemo definirati smije li ili ne smije sadržavati NULL vrijednosti
  - Ako iza definicije dodamo **NULL**, smije
  - Ako iza definicije dodamo **NOT NULL**, ne smije
  - Izostavljanjem obje naredbe smije
- Primarni ključ podrazumijeva nemogućnost unosa NULL vrijednosti



# Primjer ograničenja obaveznosti unosa

```
CREATE TABLE Ispit
```

```
(
```

```
  IDIspit int CONSTRAINT PK_Ispit PRIMARY KEY,
```

```
  StudentID int CONSTRAINT FK_Ispit_Student
```

```
    FOREIGN KEY REFERENCES Student(IDStudent) NOT NULL,
```

```
  KolegijID int CONSTRAINT FK_Ispit_Kolegij
```

```
    FOREIGN KEY REFERENCES Kolegij(IDKolegij) NOT NULL,
```

```
  Ocjena int NOT NULL,
```

```
  StegovnaMjera nvarchar(500) NULL,
```

```
  Napomena nvarchar(200)
```

```
)
```

Podrazumijeva se  
NOT NULL

Podrazumijeva se  
NULL

# Dodavanje ograničenja provjere

- Služi ograničavanju vrijednosti koje možemo upisati u stupac:
  - Definiciju smještamo iza naziva stupca i tipa podataka
  - Navodimo naredbu **CONSTRAINT** i dajemo naziv ograničenja
  - Na kraju pišemo naredbu **CHECK** i u zagradi navodimo izraz
    - Ako je izraz TRUE, unos je dozvoljen; ako je FALSE, nije

- Primjer:

```
CREATE TABLE Osoba
(
  IDOsoba int CONSTRAINT PK_Osoba PRIMARY KEY,
  Ime nvarchar(50),
  BrojCipela int CONSTRAINT CH_Osoba_BrojCipela
    CHECK (BrojCipela > 30 AND BrojCipela < 60)
)
```

# Dodavanje ograničenja jedinstvenosti (1/2)

- Ograničenje jedinstvenosti osigurava stupac ne sadržava duple vrijednosti
  - Primjerice, tablica Osoba ima primarni ključ IDOsoba. Kako ćemo osigurati da nemamo dvije osobe s istim OIB-om?
- Ako stupac dopušta NULL vrijednosti, najviše jedan redak može sadržavati NULL vrijednost
- Primarni ključ služi istoj svrsi
  - Tablica može imati samo jedan primarni ključ
  - Tablica može imati nula ili više ograničenja jedinstvenosti
  - Ograničenje jedinstvenost se koristi za osiguranje jedinstvenosti stupaca koji nisu primarni ključ

# Dodavanje ograničenja jedinstvenosti (2/2)

- Definiramo ga na sljedeći način:
  - Definiciju smještamo iza naziva stupca i tipa podataka
  - Navodimo naredbu **CONSTRAINT** i dajemo naziv ograničenja
  - Na kraju pišemo naredbu **UNIQUE**

- Primjer:

```
CREATE TABLE Student
(
  IDStudent int CONSTRAINT PK_Student PRIMARY KEY,
  Ime nvarchar(50),
  Prezime nvarchar(50),
  JMBAG nvarchar(20) CONSTRAINT UQ_Student_JMBAG
    UNIQUE NOT NULL
)
```

# Dodavanje ograničenja podrazumijevane vrijednosti

- Definira koju vrijednost treba RDBMS upisati u stupac ako je korisnik ne zada pri umetanju
- Definiramo ga na sljedeći način:
  - Definiciju smještamo iza naziva stupca i tipa podataka
  - Navodimo naredbu **CONSTRAINT** i dajemo naziv ograničenja
  - Na kraju pišemo naredbu **DEFAULT** i u zagradi pišemo vrijednost

```
CREATE TABLE Zaposlenik
```

```
(
```

```
  IDZaposlenik int CONSTRAINT PK_Zaposl PRIMARY KEY,
```

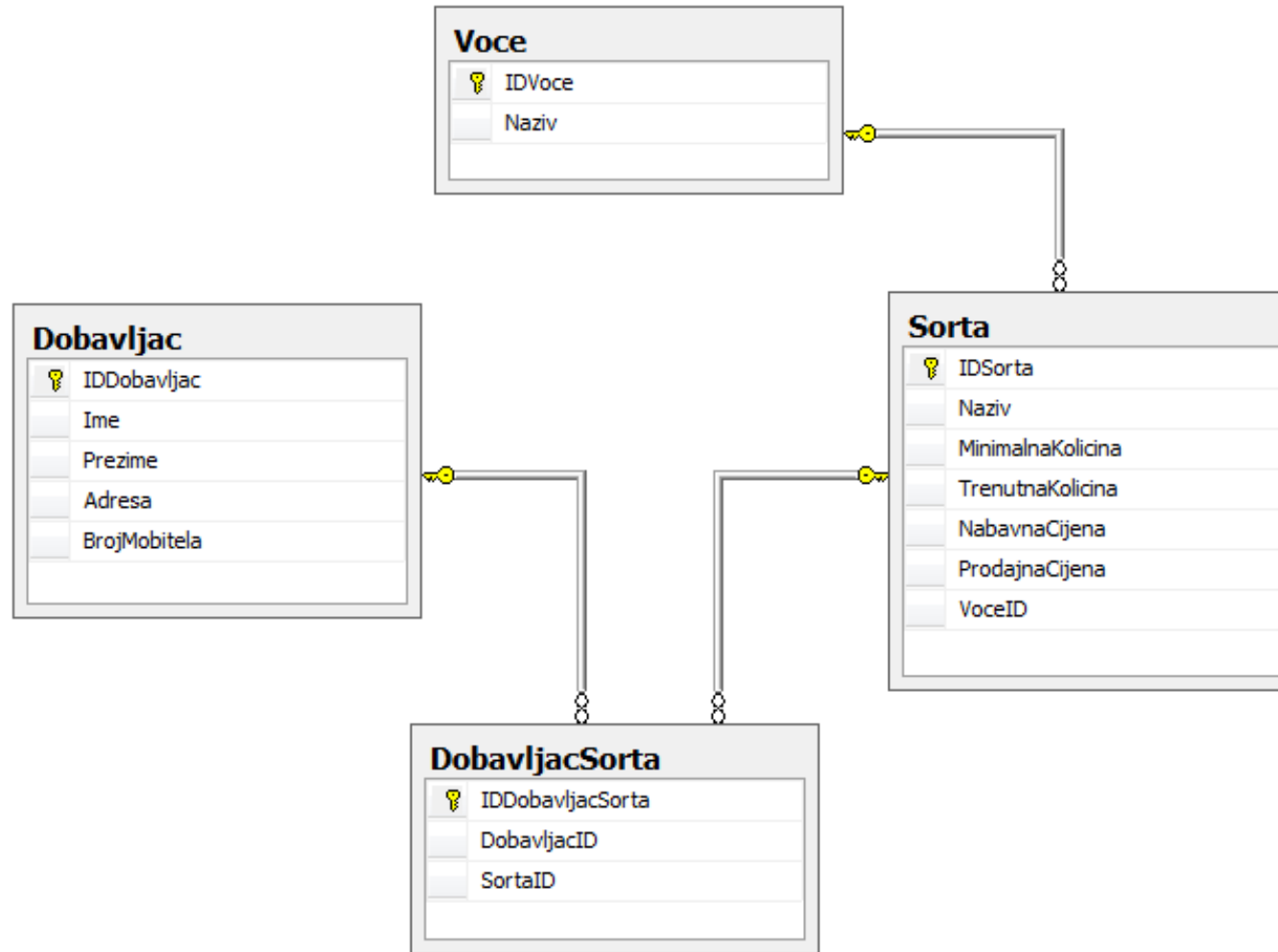
```
  Ime nvarchar(50),
```

```
  Prezime nvarchar(50),
```

```
  Pripravnik bit CONSTRAINT DF_Zaposlenik_Pripravnik DEFAULT (0)
```

```
)
```

# Primjer relacijskog modela



# Primjer skripte koja implementira bazu podataka

```
CREATE DATABASE StandNaPlacu
GO
USE StandNaPlacu
GO
CREATE TABLE Voce (
    IDVoce int CONSTRAINT PK_Voce PRIMARY KEY,
    Naziv nvarchar(200) CONSTRAINT UQ_Voce_Naziv UNIQUE NOT NULL
)
CREATE TABLE Sorta (
    IDSorta int CONSTRAINT PK_Sorta PRIMARY KEY,
    Naziv nvarchar(200) NOT NULL,
    MinimalnaKolicina int NULL,
    TrenutnaKolicina int NULL,
    NabavnaCijena money NULL,
    ProdajnaCijena money CONSTRAINT CH_Sorta_PC CHECK (ProdajnaCijena >= 0) NULL,
    VoceID int CONSTRAINT FK_Sorta_Voce FOREIGN KEY REFERENCES Voce(IDVoce)
)
CREATE TABLE Dobavljac (
    IDDobavljac int CONSTRAINT PK_Dobavljac PRIMARY KEY,
    Ime nvarchar(200) NOT NULL,
    Prezime nvarchar(200) NOT NULL,
    Adresa nvarchar(200) NULL,
    BrojMobitela nvarchar(200) NOT NULL
)
CREATE TABLE DobavljacSorta (
    IDDobavljacSorta int CONSTRAINT PK_DobavljacSorta PRIMARY KEY,
    DobavljacID int CONSTRAINT FK_DobavljacSorta_Dobavljac FOREIGN KEY REFERENCES Dobavljac(IDDobavljac),
    SortaID int CONSTRAINT FK_DobavljacSorta_Sorta FOREIGN KEY REFERENCES Sorta(IDSorta)
)
```