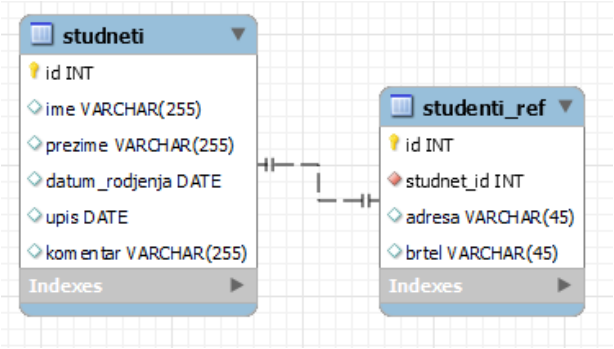
2-5 Veze između tabela

Tipovi veza

Prije detaljnog opisa spajanaj tabela, potrebno je opisati tipove veza koje mogu biti između tabela. Ti tipovi su:

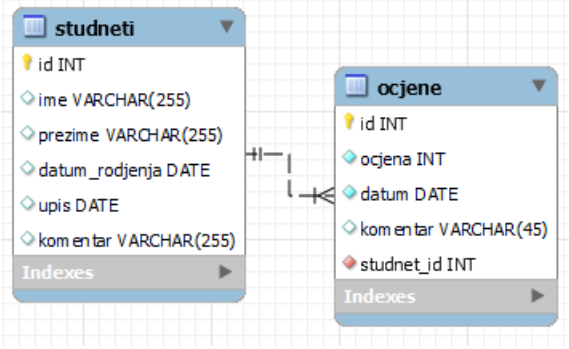
- Jedan na jedan
- Jedan na više
- Više na više

Recimo da imamo potrebu unijeti neke ostale referentne podatke za stuenta, poput, broj telefona, ulica, grad, poštanski broj, adresa, i još neke druge podatke koje ne želimo da pohranjujemo u tabelu studenti radi jednostavnosti. Ono što se može napraviti je tabela sa referentnim podacima, a koja se referira na tabelu studenti, tako označavajući kojem studentu referentni podaci pripadaju. Npr.:



Veza jedan-na-jedan

Ovo je vrsta veze jedan-na-jedan. Na slici je ERD (Entity Relationship Diagram) diagram koji predstavlja vezu jedan-na-jedan i ona je predstavljena sa dvije uspravne crte na oba kraja veze. U praksi, ovo znači da će studenti_ref imati strani ključ *student_id* koji predstavlja studenta za kojeg su ostale vrijednosti vezane. Veze jedan-na-jedan nam mogu pomoći da proširimo skup i odvojimo nužne od opcionalnih podataka.

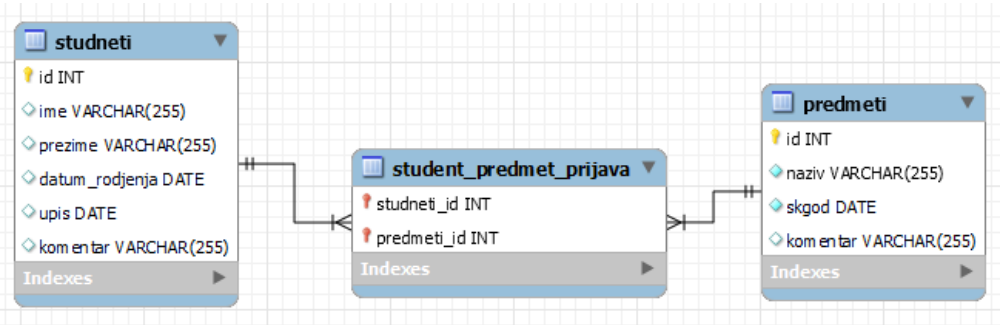


Veza jedan-na-više

Ukoliko je potrebno da jedan entitet ima više unosa za jedan element drugog entiteta, onda trebamo koristiti vezu jedan na više. Takva veza je prikazana na slici iznad i predstavlja potrebu za spašavanjem više ocjena za jednog studenta. Na ERD dijagramu ova veza se prikazuje sa crticama na jednoj strani i trokutom na drugoj strani i u tom smijeru čita se, jedan na više. Za suprotnu orijentaciju potrebno je zamijeniti trokut i crtice.

Ovo se tumači kao: jedan student može imati više ocijena. Potrebno je naglasiti da ova vrsta veze implementira koristeći strani ključ. Pa je tako u tabeli ocjena student predstavljen koristeći `student_id` i upravo to je strani ključ u tabeli `ocjene`. Vrijednost ove kolone se može ponavljati i upravo zato se ova veza zove jedan na više, jer jedan unos u tabeli studenti se može ponavljati više puta u tabeli `ocjene`.

Treća i zadnja veza koja postoji je više-na-više. Ove veze se prave kada je potrebno da se više unosa u jednoj tabeli spoji sa više unosa u drugoj tabeli. Recimo ako je potrebno modelirati prijavu studenata na nekom predmetu. U ovom slučaju, student može biti prijavljen na više predmeta ali istovremeno, jedan predmet može imati više studenata. Veza više-na-više se implementiraju koristeći dvije veze jedan-na-više koristeći međutabelu.



Veza više-na-više

Informaciju koji student je upisan na koji predmet se nalazi u tabeli `student_predmet_prijava`. Međutabela ne mora nužno imati samo strane ključeve. Moguće je dodati dodatna polja u ovu tabelu jer je ovo tabela kao i svaka druga.

Napomena oko tipova podataka

Spašavanje datoteka

Prilikom modeliranja baza podataka, potrebno je spasiti različite tipove podataka. Međutim oni tipovi koji postoje u bazama ne odgovaraju nužno potrebama u stvarnom svijetu.

Npr. ukoliko je potrebno spasiti neki dokument, sliku, video, kako bi to bilo najbolje uraditi? Poznajemo `BLOB` tip podataka za bilo koji niz nula i jedinica koji želimo spasiti. Međutim, detaljnijom analizom vidimo da `BLOB` može imati maksimalnu dužinu od 65535 bajti. Ovo znači da ne možemo spasiti ni veliku datoteku niti veliku sliku a pogotov ne video. Stoga, koji je najbolji način spasiti datoteku?

Najbolji način je zapravo da se datoteka ne spašava u bazu uopšte. Ono što se zapravo spasi je putanja do datoteke na disku, gdje se ona nalazi. Na ovaj način, baza podataka se ne opterećuje bespotrebnim podacima, a aplikacija zna gdje se datoteka nalazi, i korisnik može spasiti bilo koju veličinu dok god postoji dovoljno prostora na disku. Tako da se u bazi unosi samo `VARCHAR` ili `TEXT` polje koje označava putanju na kojoj se datoteka nalazi.

Razlika između `VARCHAR` i `TEXT` je što `VARCHAR` obično ima maksimalnu dužinu od 255 bajti, a `TEXT` od 65535 bajti ili karaktera. Još jedan tip podataka je `CHAR` koji isto tako ima maksimalnu dužinu od 255, međutim za razliku od `VARCHAR`-a, spašavanje `CHAR`-a zauzima tačno onoliko karaktera koliko je rezervisano dok `VARCHAR` može zauzimati i manje od onog broja koji je rezervisan.

Spašavanje šifri

Ukoliko želimo spasiti nečiju šifru u bazu podataka, prvi izbor bi bio vjerovatno `VARCHAR`. Taj izbor može biti u redu zavisno od onoga ŠTA se stavlja u taj `VARCHAR`. Ukoliko spasimo šifru, upravo onako kako ju je korisnik unio, tzv. cleartext, koji je problem sa kojim bi se tako dizajnirana baza podataka susrela?

Zapravo, problem nije tehničke prirode, već ljudske. Baze podataka bivaju hakovane te ovi podaci mogu doći u “ruke” nepovjerljivih osoba. Osim hakera, ove osobe mogu biti i administratori baza podataka, ili neko ko je hakovao njih. Postoji cijeli lanac nepovjerenja kao razlog zašto šifre ne bi trebale biti spašene u bazu podataka u izvornom obliku. Stoga se postavlja pitanje, kako onda spasiti šifru?

Za spašavanje šifri iskoristit će se jednosmjerna *hash* funkcija pod nazivom `SHA1`. Ova funkcija proizvodi *hash* na osnovu ulaznog stringa. Pretvaranje stringa u *hash* se zove *hash*-iranje. `SHA1` je samo jedna funkcija iz porodice *hash* funkcija. Karakteristike *hash* funkcija su da kreiraju *hash* iste dužine za ulazni tekst bilo koje dužine, tako da dva teksta različite dužine daju *hash* iste dužine. I druga karakteristika je da je šansa za kolizije veoma veoma mala, tj. ne postoje dva stringa bilo koje dužine koja mogu proizvesti isti *hash*. Treća karakteristika ovih funkcija je da nije moguće *hash* pretvoriti nazad u string, tj. ne postoji inverzna fukcija koja bi od *hash*-a vratila nazad orginalni string.

Posljedica ovih funkcija za šifre je to da zapravo u bazu možemo spasiti *hash* šifre, umjesto samu šifru. Zbog karakteristike jednosmjernosti, šifra se ne može povratiti nazad, a sama šifra bi se koristila samo tokom `INSERT` ili `UPDATE` komandi. Npr.:

```
UPDATE korisnici SET sifra = SHA1('sifra4321') WHERE id=5;
```

Na ovaj način smo zaštitili šifru od neželjenih pogleda. Ali kako će aplikacija provjeriti je li šifra tačna prilikom logovanja korisnika? Jednostavno će *hash*-irati šifru koji je korisnik unio i uporediti da li *hash* te šifre, odgovara spašenom *hash*-u u bazi. Npr.:

```
SELECT COUNT(*) FROM korisnici
WHERE email='test@email.com' AND sifra=SHA1('sifra');
```