UNIVERZITET METROPOLITAN BEOGRAD

IT250 - BAZE PODATAKA

Transformacija konceptulanog u logički i fizički model baze podataka

Lekcija 06

PRIRUČNIK ZA STUDENTE

IT250 - BAZE PODATAKA

Lekcija 06

TRANSFORMACIJA KONCEPTULANOG U LOGIČKI I FIZIČKI MODEL BAZE PODATAKA

UVOD

Šta ćemo naučiti u ovoj lekciji?

Veoma često, u slučajevima kada je konceptualni model baze podataka predstavljen ER konceptualnim i logičkim modelom, logički model je potrebno transformisati u fizički model baze podataka na osnovu kojeg se mogu generisati skripte za generisanje tabela relacionog modela.

Prilikom transformacije ER modela u fizički model baze podataka se koriste jasno definisana pravila koja su ugrađena u CASE alate. U slučaju kada je ER model urađen u nekom CASE alatu, ta transformacija se može automatski obaviti.

Mogu se izdvojiti dve grupe pravila:

- za transformaciju entiteta i
- za transformaciju veza.

Na osnovu fizičkog modela mogu se kreirati .sql skripte za generisanje tabela u konkretnom RDBMS-u.

Poglavlje 1

Transformacija konceptulanog u logički i fizički

KAKO SE VRŠI TRANSFORMACIJA?

U daljem tekstu će biti opisani načini transformacije elemenata konceptualnog modela u logički i fizički model baze podataka.

Transformacija konceptualnog u logički i fizički model je ključan korak u procesu dizajniranja i implementacije baze podataka. Ovaj proces pomaže prelazak od apstraktnog opisa podataka na višem nivou do konkretne implementacije u odgovarajućem modelu baze podataka.

Konceptualni model predstavlja apstraktnu i nezavisnu reprezentaciju podataka, fokusiranu na entitete (objekte), njihove atribute i međusobne veze. Ova faza obuhvata identifikaciju ključnih entiteta, atributa koji opisuju te entitete i odnosa između njih. Popularni alati za kreiranje konceptualnog modela su Entity-Relationship dijagrami (ER dijagrami) i UML dijagrami.

Logički model usmeren je na transformaciju konceptualnog modela u način koji je prilagođen određenom DBMS. Ova faza uključuje pretvaranje entiteta, atributa i veza u odgovarajuće tabele, kolone i ključeve u bazi podataka. Cilj je obezbediti konzistentan i efikasan pristup podacima u skladu sa pravilima i mogućnostima DBMS-a.
Fizički model predstavlja konkretnu implementaciju baze podataka za odabrani DBMS. U ovoj fazi, logički model se pretvara u specifičan kod i postavlja se na hardverski sistem za upravljanje bazom podataka. Fizički model uključuje određivanje tipova podataka, indeksiranje, izbor ključeva, optimizaciju performansi i sve ostale tehničke detalje potrebne za stvarnu realizaciju baze podataka.

Važno je napomenuti da se tokom ovih transformacija mogu javiti promene, optimizacije i dorade kako bi se model prilagodio zahtevima sistema i efikasno upravljalo podacima. Ovaj postupak često zahteva saradnju između različitih timova i stručnjaka kako bi se osiguralo da konačni fizički model odgovara ciljevima i potrebama organizacije.

TRANSFORMACIJA TIPOVA ENTITETA

Svaki tip entiteta iz ER dijagrama se transformiše u jednu tabelu baze podataka.

Transformacija tipova entiteta u logički i fizički model baze podataka se vrši tako da svaki tip entiteta iz ER dijagramu postaje jedna relacija u relacionom, odnosno tabela u fizičkom modelu baze podataka.

Identifikator tipa entiteta postaje primarni ključ tabele a drugi atributi tipa entiteta postaju atributi ne primarnih ključeva relacije (tabele).

Postoje, međutim, izuzeci od ovog "jedna tabela (relacija) po entitetu" pravila:

1. Neke klase entiteta mogu biti isključene iz baze podataka
2. Klase entiteta za klasifikaciju (ako su uključene u konceptualni model) ne treba predstaviti kao tabele
3. Tabele se takođe kreiraju za implementaciju veza više na više i n-arnih relacija (one koje uključuju više od dve klase entiteta)
4. Supertipovi i njihovi podtipovi se ne moraju uvek implementirati kao tabele.

POSTUPAK DEFINISANJA PRIMARNIH KLJUČEVA

Najpre treba odrediti primarne ključeve za tabele koje odgovaraju nezavisnim klasama entiteta, a zatim ih dupliramo kao strane ključeve u tabelama klasa entiteta s kojima su u vezi

Prilikom transformacije treba voditi računa o sledećem:

Primarni ključ treba da ima sledeće svojstva:
Vrednost primarnog ključa jedinstveno identifikuje svaku torku u relaciji (red u tabeli). Ključ mora da bude neredundantan; nema atributa u ključu koji može biti izbrisan a da se ne naruši njegova jedinstvena identifikacija
Idealan primarni ključ treba da bude kratak, numerički i fiksan. U slučajevima kada se to ne može postići, treba razmotriti mogućnost korišćenja nekog kandidat ključa kao primarnog. Ako nema kandidat ključeva, ili nijedan od njih ne zadovoljava prethodne kriterijume, treba razmotriti mogućnost korišćenja surogat ključa.

Surogat ključ je identifikator kojeg generiše RDBMS, njegov redosled je jedinstven u okviru jedne tabele i nikada se ne menja . On se dodeljuje u trenutku kreiranja reda u tabeli a uništava se kada se red obriše.

Kandidat ključevi su alternativni identifikatori jedinstvenih redova u tabeli. Umesto termina kandidat ključ, često se koristi i termin alternativni ključ.

Na primer: U relaciji ZAPOSLENI (BrojZaposlenog, ImeZaposlenog, Telefon, Email, DatumZaposlenja, DatumProvere, KodZaposlenog)

BrojZaposlenog je primarni ključ, dok Email može biti kandidat ključ ili alternativni ključ.

U principu, primarni ključ tabele može uključivati strani ključ iz druge tabele. Međutim, u trenutku prevođenja konceptualnog u logički model, strani ključevi mogu još uvek bit nedefinisani - te to ne možemo uraditi sve dok ne definišemo primarne ključeve tabele na koje se referenciramo.

Na početku ovog koraka treba odrediti primarne ključeve samo za one tabele koje odgovaraju nezavisnim klasama entiteta, pošto, kao što smo videli, primarni ključevi takvih tabela neće uključivati strane ključeve. Zbog toga prvo izaberete odgovarajući primarni ključ za svaku od ovih tabela, ako je potrebno dodavanjem kolone surogat ključa kao ključa ili dopunjavanjem postojećih atributa. Nakon navođenja primarnog ključa za najmanje nekoliko tabela, u mogućnosti smo da ih dupliramo kao strane ključeve u tabelama koje odgovaraju klasama entiteta s kojima su u vezi.
Nakon toga možemo da odredimo primarne ključeve tabela koje predstavljaju klase entiteta koje su u vezi sa klasama entiteta za koje smo već identifikovali primarne ključeve (pošto sada imate kompletnu listu kolona za ove tabele, uključujući i strane ključeve).

TRANSFORMACIJA ATRIBUTA TIPOVA ENTITETA

Prilikom transformacije atributa tipova entiteta, treba voditi računa o null statusu, tipu podataka, default vrednosti za svaki atributa.

Kada se vrši transformacija atributa tipova entiteta u atribute relacije treba obraditi pažnju na sledeće:

- null status
- tip podataka
- default vrednosti
- ograničenja podataka

Null status atributa: Definiše da kolona ne mora imati vrednost. Dozvoljena null vrednost se definiše frazom NULL dok se nedozvoljena definiše sa NOT NULL. NULL ne znači da je kolona uvek NULL već znači da su dozvoljene NULL vrednosti.

Tipovi podataka atributa : Svaki DBMS ima neke svoje specifične tipove podataka koje koristi. Međutim ako se želi postići nezavisnost od konkretnog RDBMS, mogu se specificirati generički tipovi podataka gde spadaju:

- CHAR(n) – karakter string fiksne dužine
- VARCHAR(n) - karakter string promenljive dužine
- DATE
- TIME
- MONEY
- INTEGER
- DECIMAL

Default vrednost atributa:To je podrazumevajuća vrednost koju generiše RDBMS prilikom kreiranja redova tabele. Ta vrednost može biti konstanta ili rezultat neke funkcije kao što je sistemski datum ili vreme. Nekada se default vrednosti sračunavaju korišćenjem mnogo složenije logike, za šta se najčešće koriste trigeri.

OGRANIČENJA NA VREDNOSTIMA PODATAKA

Postoje nekoliko različitih ograničenja na vrednostima podataka.

Postoje nekoliko različitih ograničenja na vrednostima podataka koja su definisana domenom:

1. Kojim se vrednost kolona ograničava na određeni skup vrednosti. *Na primer. KodZaposlenog može biti ograničen vrednostima ('novozaposleni', 'stalno zaposlen', 'honorarno zaposlen' itd.)*
2. Opseg vrednosti – kojim se vrednosti ograničavaju na određeni interval. *Na primer: datum zaposlenja može biti u opsegu od 1. januara 2002. do 31. decembra 2022. godine.*
3. Intra relaciono ograničenje u odnosu na kolone iste tabele – ograničava vrednost kolone u odnosu na druge kolone iste tabele. *Na primer: datum provere može biti najmanje 3 meseca posle datuma zaposlenja.*
4. Inter relaciono ograničenje u odnosu na kolone druge tabele -ograničava vrednost kolone u odnosu na druge kolone drugih tabela. *Ograničenje referencijalnog integriteta je jedan tip inter relacionog ograničenja.*

Na kraju procesa transformacije tipova entiteta iz ER modela u tabele baze podatka treba proveriti da li su tabele normalizovane . Drugim rečima, treba dati odgovor na pitanje da li su tabele najmanje u trećoj normalnoj formi (3NF) i da li su otklonjene sve funkcionalne zavisnosti. Ukoliko nisu, relacije treba normalizovati. Kako je normalizacija u nekim slučajevima nepoželjna, treba proveriti da li normalizovane relacije treba denormalizovati.

Sledeći korak je transformisati veze između tipova entiteta.

Poglavlje 2

Transformacija veza

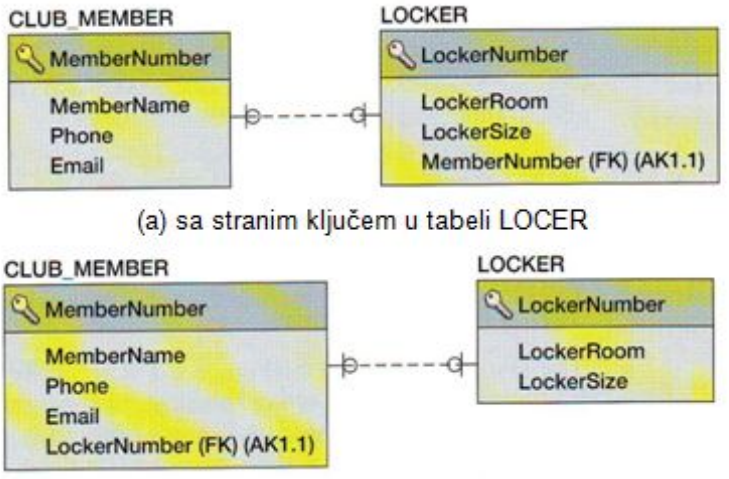
TRANSFORMACIJA VEZA MAKSIMALNE KARDINALNOSTI 1:1

Vrši se dodavanjem primarnog ključa relacije A kao stranog ključa u B ili obrnuto.

Transformacija veza maksimalne kardinalnosti 1:1 vrši se dodavanjem primarnog ključa relacije A kao stranog ključa u B ili obrnuto (slika 2.1). Međutim, ono što zaista treba podržati ovde više nije relacija jedan-prema-jedan, nego relacija jedan-prema-više.

Relacija jedan-prema-jedan može biti podržana u relacionoj bazi podataka i tako što se obe klase entiteta koriste kao tabele, a zatim se koristi isti primarni ključ za obe. Zapravo, ovo je način koji koristimo kada postoji veza (jedan-na-jedan) između supertipa i njegovih podtipova a kada obe klase entiteta treba da se implementiraju kao tabele.

Primer:



Slika 2.1 (a) sa stranim ključem u tabeli LOCER; (b) sa stranim ključem u tabeli CLIB_MEMBER [Izvor: NM IT350-2020/2021.]

TRANSFORMACIJA VEZA MAKSIMALNE KARDINALNOSTI 1:M

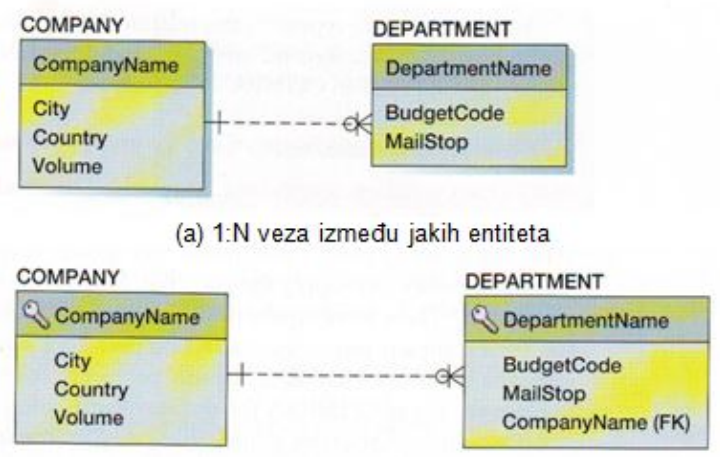
Vrši se spuštanjem atributa primarnog ključa relacije koji je na strani veze jedan, kao stranog ključa u relaciju koja je na strani veze više.

Veza sa maksimalnom kardinalnošću jedan-prema-više (1:N) se transformiše spuštanjem atributa primarnog ključa relacije koji je na strani veze jedan, kao stranog ključa u relaciju koja je na strani veze više.

Stoga, u logičkom (fizičkom) modelu podataka, u tabeli koja predstavlja klasu entiteta na kraju veze "više", kreiramo kopiju primarnog ključa klase entiteta na kraju veze "jedan". (Imajte na umu da primarni ključ može biti predstavljen sa više od jedne kolone i naravno da ćemo morati da kopiramo sve kolone kako bismo formirali strani ključ.) Svakoj koloni stranog ključa treba dati ime koje ima kolona primarnog ključa iz koje je izveden, moguće sa dodavanjem prefiksa. Prefiksi su neophodni u dve situacije:

1. Ako postoji više od jedne relacije između dve iste klase entiteta, u tom slučaju je neophodno koristiti prefiks za razlikovanje dva različita strana ključa, na primer Preparation Employee ID and Approval Employee ID.
2. Rekurzivna relacija će biti predstavljena stranim ključem koji sadrži istu kolonu kao primarni ključ iste tabele, tako da će biti potreban prefiks za imena kolona stranog ključa; tipični prefiksi su "Roditelj", "Vlasnik", "Menadžer".

Primer:



Slika 2.2 Spuštanje ključa roditelja u dete [Izvor: NM IT350-2020/2021.]

Podsetimo se da je termin roditelj korišćen za tabelu na strani jedan a termin dete za tabelu na strani više. Korišćenjem ove terminologije za dizajn 1:N veze se može reći da predstavlja „spuštanje ključa roditelja u dete" (slika 2.2).

ŠTA JE STRANI KLJUČ?

Sredstvo za implementaciju relacija jedan prema više (i povremeno jedan-prema-jedan)

Strani ključevi su sredstvo za implementaciju relacija jedan prema više (i povremeno jedan-prema-jedan). Ova faza logičkog projektovanja zahteva da znamo primarni ključ klase entiteta na kraju veze "jedan", a kao što je poznato, definicija primarnih ključeva zavisi od definicije stranih ključeva. Dakle, najpre treba implementirati relacije koje ispunjavaju ovaj

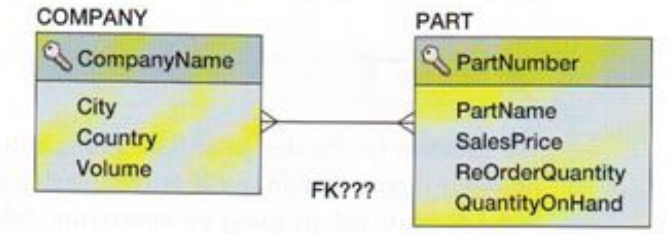
kriterijum, onda se vratiti kako bi se definisali ostali primarni ključevi. Ova sekcija počinje sa opisom osnovnih pravila za implementaciju veza jedan-prema-više. Ovim pravilom će biti obuhvaćena ogromna većina situacija. U ostatku sekcije se razmatraju različite situacije koje se mogu pojaviti kao neuobičajene. Korisno je biti upoznat i sa njima jer se pojavljuju s vremena na vreme i morate biti u mogućnosti da ih prepoznate i rešite.

TRANSFORMACIJA VEZA MAKSIMALNE KARDINALNOSTI N:M

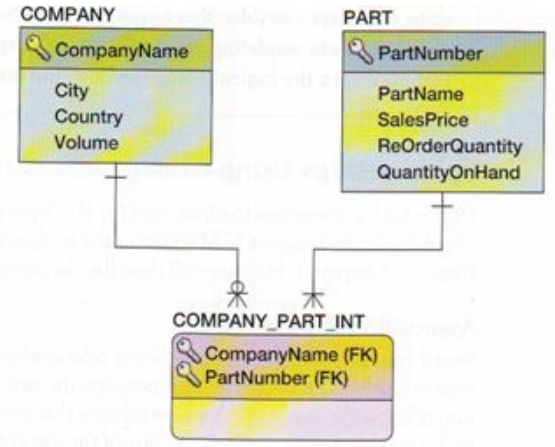
Za takvu vezu se kreira posebna tabela. Primarni ključ te tabele je složeni ključ koji se sastoji od primarnih ključeva svake od dve tabele koje učestvuju u vezi.

Transformacija veze više -prema-više veza se vrši tako što se kreira posebna relacija . Primarni ključ te relacije je složeni ključ koji se sastoji od primarnih ključeva svake od dve relacije dobijene transformacijom tipova entiteta koji učestvuju u M:N vezi.

Primer: prikazan na slikama 2.3 i 2.4. .



Slika 2.3 Primer maksimalne kardinalnosti veze više na prema više [Izvor: NM IT350-2020/2021.]



Slika 2.4 Transformacija veze maksimalne kardinalnosti više na prema više u fizički model [Izvor: NM IT350-2020/2021.]

Napomenimo da se svaka M:N veza može uvek korišćenjem dodatne tabele dekomponovati na dve 1:N veze. Neki CASE alati nemaju mogućnost da predstave M:N veze u modelu podataka, pa projektant mora da ovu transformaciju izvrši tokom faze modeliranja podataka.

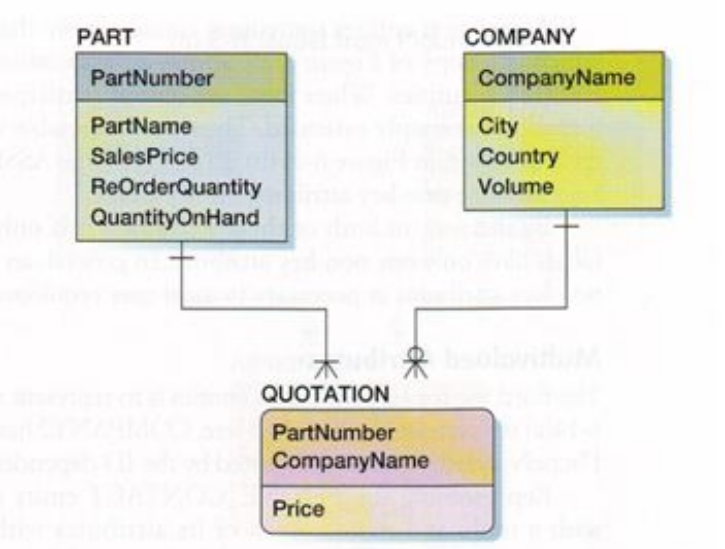
Međutim, ovakav zahtev se može smatrati neopravdanim jer u modeliranje podataka unosi složenost a logika modeliranja je upravo smanjenje složenosti.

TRANSFORMACIJA ASOCIJATIVNIH ENTITETA

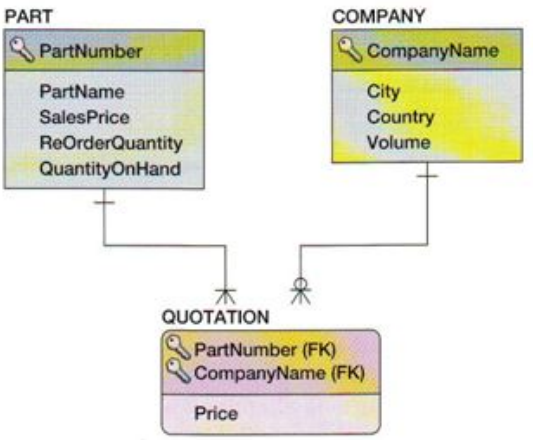
Vrši se na sličan način kao i kod N:M relacija, jedina razlika je u tome što asocijativne relacije imaju svoje atribute.

Transformacija asocijativnih entiteta (relacija koja imaju atribute) je veoma slična transformaciji N:M relacija. Jedina razlika je u tome što asocijativni entiteti imaju jedan ili više atributa koji se dodeljuju relaciji nastaloj transformacijom veze više-na-prema-više na relacijama nastalim transformacijom tipova entiteta koji učestvuju N:M relaciji.

Primer transformacije prikazan na slikama 2. 5 i 2. 6 .



Slika 2.5 Primer asocijativne veze [Izvor: NM IT350-2020/2021.]

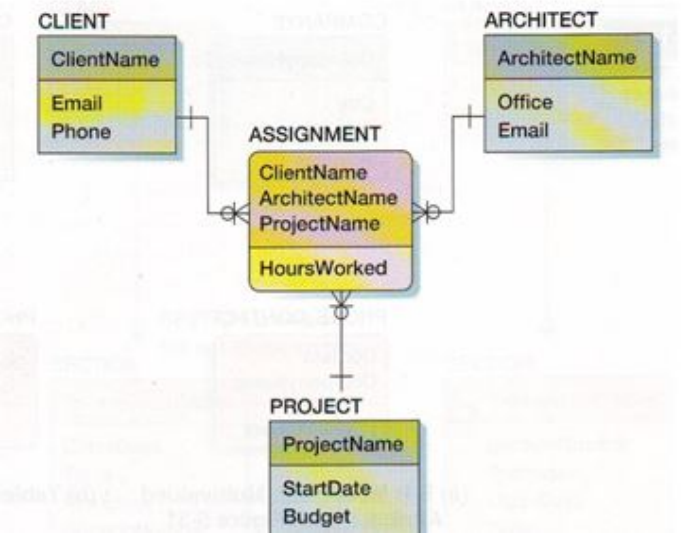


Slika 2.6 Primer transformacije asocijativne veze [Izvor: NM IT350-2020/2021.]

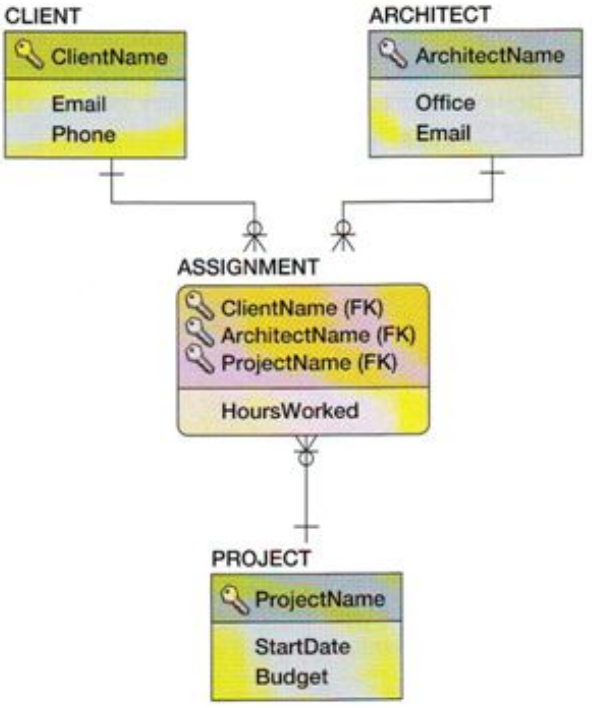
TRANSFORMACIJA ASOCIJATIVNIH ENTITETA KOJI PREDSTAVLJAJU TERNARNE ILI KVATERNARNE RELACIJE

U tom slučaju asocijativna relacija će sadržati ključeve svakog od svojih roditelja.

Transformacija asocijativnih entiteta koji predstavljaju ternarne ili kvaternarne relacije se transformišu tako što će relacija dobijena njihovim transformacijama sadržati ključeve svakog od svojih roditelja što je prikazano na primeru prikazanom na slikama 2. 7 i 2. 8 .



Slika 2.7 Primer ternarne relacije [Izvor: NM IT350-2020/2021.]

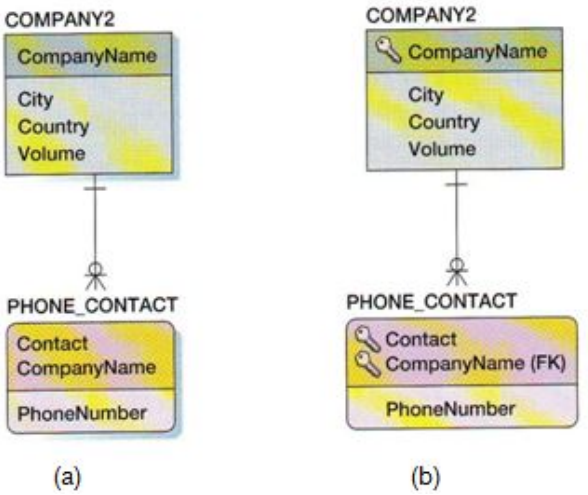


Slika 2.8 Transformacija ternarne relacije u relacioni model [Izvor: NM IT350-2020/2021.]

TRANSFORMACIJA ATRIBUTA SA VIŠE VREDNOSTI

Svaki atribut sa više vrednosti se može predstaviti ID zavisnim entitetom.

Transformacija atributa sa više vrednosti se vrši tako što se svaki atribut sa više vrednosti predstavlja ID zavisnim entitetom. Neka tip entiteta COMPANY ima atribute CompanyName (primarni ključ), City, Country, Volume itd. kao i atribut sa više vrednosti Contact. Taj se tip entitet može predstaviti sa dve relacije, što je predstavljeno na slici 2. 9 .



Slika 2.9 Atribut sa više vrednosti i njegova transformacija [Izvor: NM IT350-2020/2021.]

Na slici 2.9(a) je predstavljen ER model na kojem je predstavljen tip entiteta koji sadrži atribut sa više vrednosti.

Na slici 2.9(b) su predstavljene relacije kojima se mogu predstaviti tipovi entiteta koji imaju atribute sa više vrednosti.

Prilikom transformacije u relacije, tip entiteta PHONE_CONTACT treba transformisati u relaciju i svaki njen atribut predstaviti kolonom. U ovom slučaju atribut CompanyName je strani ključ koji je postao deo primarnog ključa relacije PHONE_CONTACT.

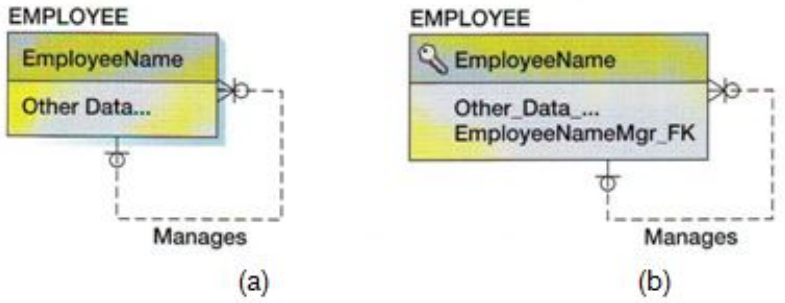
TRANSFORMACIJA REKURZIVNE RELACIJE

Zavisi od kardinalnosti rekurzivne relacije koja može biti jedan-prema-više ili više-prema-više.

Unarna (rekurzivna) relacija može imati kardinalnost jedan-prema-više ili više-prema-više.

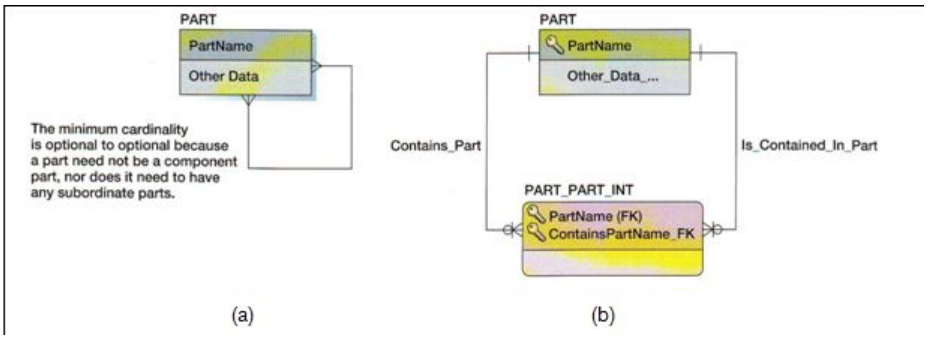
Razmotrimo unarnu 1:M relaciju prikazanu na slici 2. 10 pod (a).

Transformacija rekurzivne relacije kardinalnosti jedan prema više se vrši kao i u slučaju drugih tipova 1:M relacija, tako što se tip entiteta nad kojim postoji rekurzivna relacija modelira kao relacija. Njen primarni ključ je isti kao i identifikator tipa entiteta. Tada se relaciji dodaje rekurzivni strani ključ koji se referencira na vrednost primarnog ključa iste relacije. Relacija koja se dobija transformacijom rekurzivne veze Manages nad tipom entiteta EMPLOYEE je predstavljena na slikom 2. 10 .



Slika 2.10 Unarna relacija sa maksimalnom kardinalnišću veze 1:M nad tipom entiteta EMPLOYEE i njena transformacija u odgovarajuću relaciju. [Izvor: NM IT350-2020/2021.]

Transformacija rekurzivne relacije tipa više prema više se vrši kao drugi tipovi M:N veza: tip entiteta nad kojim postoji rekurzivna relacija se transformiše u relaciju a zatim se kreira posebna relacija da bismo predstavili M:N vezu. Primarni ključ te nove relacije je složeni ključ koji se sastoji od dva atributa (koji ne moraju da imaju isto ime) koji uzimaju vrednosti primarnih ključa relacije nastale transformacijom tipa entiteta. Najčešće se ovaj tip relacije predstavlja primerom prikazanim na slici 2.11(a), naziva struktura sastavnice. Ako je relaciji pridružen jedan ili više atributa (npr. atribut Quantity), on je u novu relaciju uključen kao atribut koji nije deo primarnog ključa. Rekurzivna relacija tipa sastavnice je prikazana na slici 2.11 (a) a rezultat transformacije je prikazan na slici 2. 11 (b):

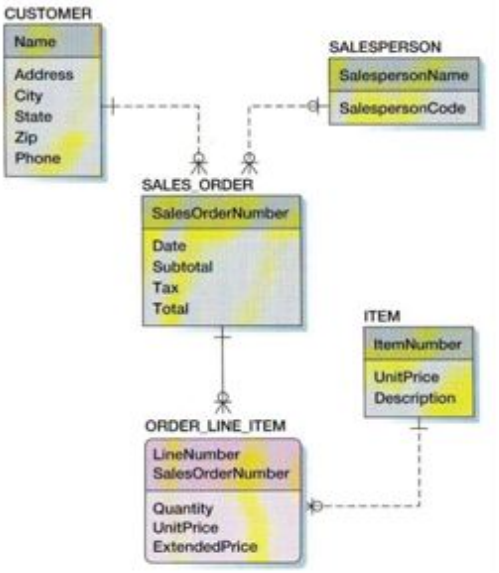


Slika 2.11 Transformacija rekurzivne veza kardinalnosti M:N [Izvor: NM IT350-2020/2021.]

TRANSFORMACIJA MEŠOVITIH VEZA

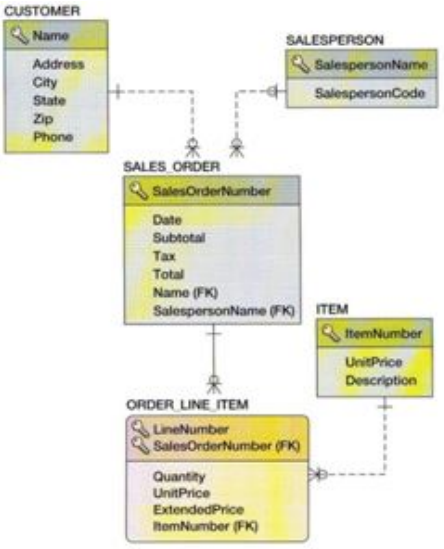
Transformacija mešovitih veza na primeru veze date na slici

Primer modela sa mešovitim vezama je predstavljen na slici 2. 12 .



Slika 2.12 Primer modela sa mešovitim vezama [Izvor: NM IT350-2020/2021.]

Tip entiteta SALES_ORDER na slici ima ID zavistan entitet ORDER_LINE_ITEM u kojem je broj narudžbenice (SalesOrderNumber) dobijen kao strani ključ, deo njegovog primarnog ključa. BrojStavke (ItemNumber) je strani ključ (slika 2.13).



Slika 2.13 Transformisani ER model sa mešovitim vezama [Izvor: NM IT350-2020/2021.]

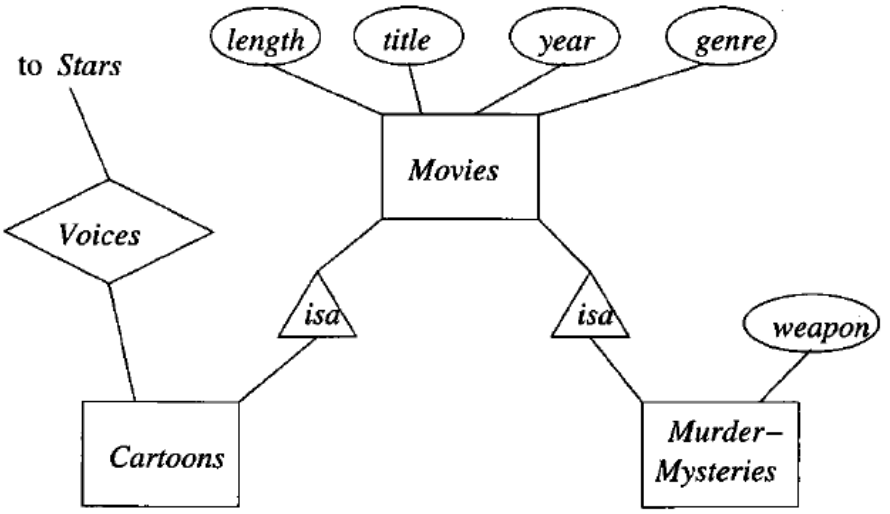
TRANSFORMACIJA PODKLASA I NJIHOVIH NADKLASA

Može se vršiti na tri načina: korišćenje E/R stila, tretiranjem tipova entiteta kao objekata koji pripadaju jednoj klasi i korišćenjem null vrednosti

Transformacija podklasa i njihovih nadklasa se može se vršiti na tri načina, korišćenjem:

1. ER stila transformacije podklasa i njihovih superklasa: u tom slučaju za svaki tip entiteta E u hijerarhiji se kreira relacija koja kao ključ uključuje atribute sa korena hijerarhije i atribute koji pripadaju entitetu E.
2. Objektno orijentisanog pristupa transformacije podklasa i njihovih superklasa: tipovi entiteti se tretiraju kao objekti koji pripadaju jednoj klasi: za svako moguće podstablo koje se nalazi ispod korena, kreira se jedna relacija čija šema uključuje sve atribute svih tipova entiteta u podstablu.
3. Null vrednosti za transformaciju podklasa i njihovih superklasa: kreira se jedna relacija sa svim atributima svih tipova entiteta u hijerarhiji. Svaki entitet je predstavljen jednom torkom i ta torka ima NULL vrednosti za bilo koji atribut koji entitet nema.

Primer dijagrama nad kojim će se primeniti različiti stilovi transformacija podklasa i njihovih superklasa je prikazan na slici 2.14.



Slika 2.14 Primer primene superklasa i podklasa u bazi filmova [Izvor: NM IT350-2020/2021.]

ER STIL TRANSFORMACIJE PODKLASA I NJIHOVIH SUPERKLASA

Relacija se kreira za svaki tip entiteta. Ukoliko tip entiteta E nije koren u hijerarhiji, tada će relacija za E uključivati atribut ključa za koren hijerarhije

ER stil transformacije se vrši tako što se kreira relacija za svaku podklasu i njihove superklase. Ukoliko klasa E nije koren u hijerarhiji, tada će relacija za E uključivati atribut ključa relacije kojom je predstavljen koren hijerarhije (superklasa) kako bi se identifikovali entiteti predstavljeni svakom torkom, plus svi atributi podklase E. Osim toga, ako E učestvuje u nekoj vezi, atribut ključa iz te veze se koristi kako bi se u entitetima iz E identifikovali entiteti koji odgovaraju vezi.
Napomenimo međutim, da mada mi o specijalizaciji *"isa"* (vidi sliku 14)govorimo kao o vezi, ona je drugačija od drugih veza u tome što ona povezuje entitete istog tipa entiteta, a ne različitih tipova entiteta. Zbog toga mi ne kreiramo relaciju za *"isa".*
Primer: Relacije koje su neophodne da bi se predstavile podklase i superklase u hijerarhiji sa slike 14 su :

1. MOVIES (Title ,Year, Length , Genre), prva dva atributa su ključ za sve filmove a zadnji je samostalni atribut koji odgovara odgovarajućem skupu entiteta.

2. MURDERMYSTERIES (Title, Year, Weapon)- Filmovi koji predstavljaju misteroizna ubustva ovde imaju torku kao i u relaciji MOVIES. Torke ove relacije imaju kao sekundarni ključ atribute Title, Year,

3. CARTOONS (Title, Year). Ova relacija predstavlja skup crtanih filmova. Ona nema druge atribute osim sekundarnog ključa spuštenog iz MOVIES, jer ekstra informacije o crtanim filmovima su sadržane u vezi Voices. Filmovi koji predstavljaju crtane filmove ovde imaju torku kao i u MOVIES. Napomenimo da četvrta vrsta filmova- oni koji predstavljaju

istovremeno i crtane filmove i misteriozna ubistva imaju torke u sve tri relacije. Pored ove tri relacije, potrebna nam je i relacija:

4. VOICES (Title, Year, StarName) između STARS i CARTOON. Zadnji atribut je sekundarni ključ iz STARS, a prva dva su sekundarni ključevi iz CARTOONS.
Na primer, film Roger Rabbit može imati torke u sve četiri relacije. Njegove osnovne informaciju bi bile u MOVIES, informacije o oružju bi se pojavile u MURDERMYSTERIES, a zvezde koje daju glas u filmu bi bile sačuvane u STARS. Napomenimo da relacija CARTOONS ima šemu koja predstavlja podskup šema za relaciju VOICE. U mnogim situacijama bi bili zadovoljni da eliminišemo relaciju kao što je CARTOONS, jer ona ne sadrži nijednu informaciju izvan onoga što se nalazi u VOICES. Međutim, u našoj bazi se mogu naći i nemi crtani filmovi. Ti crtani filmovi nemaju glas, pa bismo u tom slučaju izgubili informacije eliminišući relaciju CARTOONS.

OBJEKTNO ORIJENTISANI PRSTUP TRANSFORMACIJE PODKLASA I NJIHOVIH SUPERKLASA

Kreira se jedna relacija za klase u svakom podstablu. Šema te relacije ima sve atribute bilo kog tipa entiteta u podstablu

Alternativna strategija za konvertovanje ISA-hijerarhije u relacije jeste nabrojati sva moguća podstabla u hijerarhiji. Objektno orijentisani pristup transformacije se vrši tako što se kreira po jedna relacija za svaku podklasu i superklase koje se nalaze u podstablima hijerarhije. Svaka od tih relacija sadrži sve atribute superklase koja se nalazi u korenu podstabla plus atribute date podklase u podstablu. Ovaj pristup je motivisan pretpostavkom da su entiteti objekti koji pripadaju **jednoj i samo jednoj klasi.**

Primer: Razmotrimo hijerarhiju na slici 14. Postoje četiri moguća podstabla uključujući i koren (root).

1. *samo MOVIES*
2. *samo MOVIES i CATOONS*
3. *samo MOVIES i MURDERMYSTERIES*
4. *sve tri klase*

Moramo konstruisati relacije za sve klase:

- *MOVIIES (Title, Year, Length, Genre)*
- *MOVIESC (Title, Year, Length, Genre)*

- *MOVIESMM (Title, Year, Length, Genre, Weapon)*
- *MOVIESCMM (Title, Year, Length, Genre, Weapon)*

Ako bi klasa CARTOONS imala svoje jedinstvene atribute, tada bi sve četiri relacije morale imati različit skup atributa. Obzirom da to ovde nije slučaj, mi ćemo kombinovati MOVIIES sa MOVIESC (tj. kreirati jednu relaciju za non-murder-mysteries i kombinovati MOVIESMM sa MOVIESCMM (tj. kreirati jednu relaciju za sve murder mysteries), mada radeći tako gubimo neke informacije – npr. koji su filmovi crtani

Takođe je potrebno razmotriti relaciju VOICES od CARTOONS do STARS. Ako je VOICES relacija više na prema jedan, tada možemo dodati atribut voice u MOVIESC i MOVIESCMM, koji će predstavljati relaciju VOICES i uticaće na to da sve četiri relacije budu različite. Međutim, VOICES je relacija više prema više, pa je potrebno kreirati posebnu relaciju za tu vezu. Šema relacije VOICES bi bila
VOICES (Title, Year, StarName)
Može se razmotriti da li je neophodno kreirati takve dve relacije, jednu koja povezuje crtane filmove koji nisu misteriozna ubistva i drugu za crtane filmove koji to jesu. Međutim, ne čini se da bi to bilo dobro.

TRANSFORMACIJA PODKLASA I NJIHOVIH SUPERKLASA KORIŠĆENJEM NULL VREDNOSTI

Hijerarhiju klasa možemo predstaviti kao jednu jedinstvenu relaciju. Ta relacija ima sve atribute koji pripadaju bilo kojoj klasi u hijerarhiji

Postoji još jedan pristup za predstavljanje informacija o hijerarhiji klasa. Ukoliko nam je dozvoljeno da koristimo NULL vrednosti kao vrednost u torkama,transformacija korišćenjem NULL vrednosti bi omogućila da se hijerarhija klasa predstavi kao jedna jedinstvena relacija. Ta relacija ima sve atribute koji pripadaju bilo kojoj klasi u hijerarhiji. Entitet je tada predstavljen kao jedna torka. Ta torka ima NULL za svaki atribut koji nije definisan za tu klasu.
Primer: Ako ovaj pristup primenimo na dijagram sa slike 14, kreiraćemo samo jednu relaciju čija je šema:
MOVIES(Title, Year, Length, Genre, Weapon)

Oni filmovi koji se ne odnose na misteriozna ubistva će imati vrednost NULL za atribut weapon u torkama. Takođe je neophodno imati relaciju VOICES koja bi povezivala one filmove koji predstavljaju crtane filmove sa zvezdama koje daju svoju glas.

Poglavlje 3

Primer transformacije u logički i fizički model

TRANSFORMACIJA KONCEPTUALNOG U LOGIČKI U FIZIČKI MODEL

Na primeru umetničke galerije View Ridge

View Ridge je mala umetnička galerija koja prodaje umetničke predmete iz Evrope i Severne Amerike uključujući litografije, originalne slike i fotografije. Ona posluje skoro 30 godina i ima svog vlasnika i troje zaposlenih. Zahtevi za izradu aplikacije koja treba da podrži poslovanje ove galerije su:

1. Da se evidentiraju i prate svi kupci i njihovo interesovanje za pojedinim umetnicima
2. Da se evidentiraju sve porudžbine koje izvrši galerija
3. Da se evidentiraju sve porudžbine kupaca
4. Da se formira lista svih umetnika i njihovih radova
5. Da se omogući izveštavanje o tome koliko brzo su radovi pojedinih umetnika prodati
6. Da se prikaže tekuće stanje zaliha
7. Prvo, vlasnik i prodavci žele da beleže imena, prezimena, adrese, telefone, email-ove svojih kupaca. Oni takođe žele da znaju kojim se kupcima sviđaju koji umetnici.

Kada galerija kupi novi umetnički predmet, beleže se podaci o umetniku, prirodi umetničkog rada, datumu nabavke i ceni nabavke. Galerija može da od kupca koji je kupio neki umetnički rad, taj rad otkupi i ponovo ga proda, tako da se jedan rad u galeriji može pojaviti više puta. Kada se umetnički rad otkupi od kupca koji ga je kupio, podaci o umetniku i umetničkom radu se ne unose ponovo u bazu, ali se unosi novi datum i cena nabavke. Takođe, kada se umetničko delo proda, u bazi podataka se pamte datum i cena prodaje i identitet kupca.

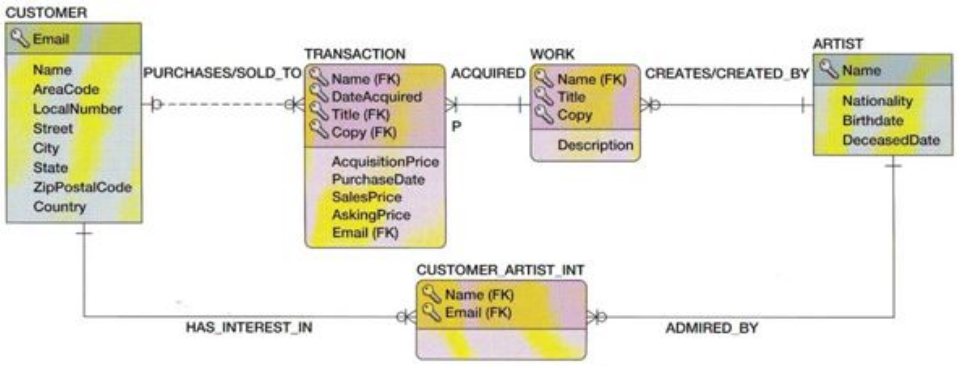
Treba omogućiti da se analiziraju datumi kupovine kako bi se za najaktivnije kupce moglo izdvojiti više vremena. Takođe podaci o kupovini se često koriste kako bi se pronašle lokacije umetničkih radova na kojima su oni zadnji put prodati.

Za marketinške svrhe, galerija želi da iz baze podataka dobije listu umetnika i radova koji su se pojavili u galeriji. Vlasnik takođe želi da odredi koliko se brzo prodaje jedno umetničko delo i koje su margine prodaje.

KONCEPTUALNI (LOGIČKI) MODEL PODATAKA ZA BAZU GALERIJE

Model ima dva jaka tipa entiteta, ID zavisne tipove entiteta koji su povezani različitim tipovima veza

Na slici 3.1 je prikazan konceptualni model podataka za bazu podataka View Ridge galerije koji ujedno zadovoljava kriterijume logičkog modela. Ovaj model ima dva jaka tipa entiteta: CUSTOMER (kupac) i ARTIST (umetnik). Pored toga, tip entiteta WORK (umetnički rad) je ID zavistan od ARTIST a entitet TRANSACION (transakcija) je ID zavistan od WORK.
Umetnik (ARTIST) se može uneti u bazu podataka čak i ako se nijedan njegov rad ne pojavi u galeriji. Tako umetnik može imati 0 ili 1 umetničkih radova. Identifikator entiteta umetnički rad (WORK) je složen (Title i Copy) jer u slučaju litografija i slika može postojati više kopija za dati naslov.



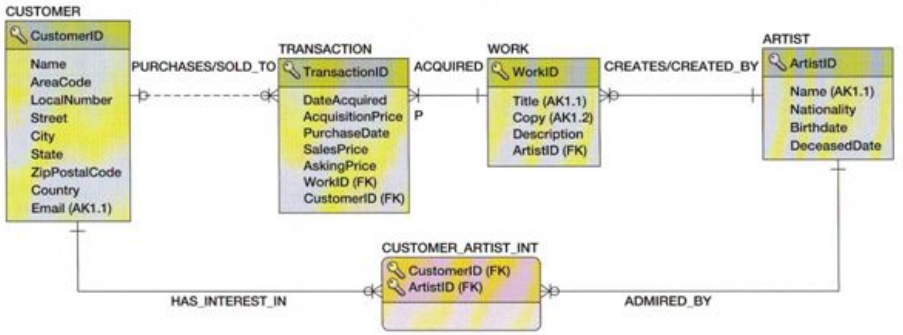
Slika 3.1 Konceptualni (logički) model podataka za bazu podataka View Ridge galerije [Izvor: NM IT350-2020/2021.]

Takođe, jedan umetnički rad se u galeriji može pojaviti više puta, pa postoji potreba da se za svaki umetnički rad pamti više transakcija (TRANSACTION). Svaki put kada se umetnički rad pojavi u galeriji, moraju se zabeležiti datum i cena nabavke, pa za svaki umetnički rad mora da postoji bar jedan red u entitetu transakcija. Kupac može da kupi više radova što se beleži kao relacija 1:M između CUSTOMER i TRANSACION. Ova relacija je opciona u oba smera. Na kraju, između CUSTOMER I ARTIST postoji relacija N:M.

FIZIČKI MODEL BAZE PODATAKA

U fizičkom modelu su prirodni ključevi zbog svoje složenosti zamenjeni surogat ključevima.

Fizički model baze podataka za prethodno prikazan konceptualni odnosno logički model podataka je prikazan na slici 3.2. Iz dizajna se može videti da su svi primarni ključevi osim ARTIST.name problematični. Ključevi za WORK i TRANSACION su suviše veliki a ključ za CUSTOMER može biti sumnjiv jer neki kupci ne moraju imati email. Zbog ovih problema su kreirani surogat ključevi. Dizajn baze podataka sa surogat ključevima je prikazan na slici 3.2.



Slika 3.2 Fizički model baze podataka sa surogat ključevima [Izvor: NM IT350-2020/2021.]

Poglavlje 4

Transformacija minimalne kardinalnosti veza

ANALIZA MOGUĆIH MINIMALNIH KARDINALNOSTI SA STRANE RODITELJA I DECE

Analiza je obavljena u dva pravca: za slučaj kada su obavezni roditelji (relacije M-O i M-M) i za slučaj kada su obavezna deca (O-M i M-M).

Ova analiza je obavljena samo za slučaj maksimalne kardinalnosti veze jedan-prema-više koja se često naziva relacijom RODITELJ - DETE gde se roditeljem smatra tip entiteta sa strane relacije 1, a detetom tip entiteta sa strane relacije više. U tom slučaju, relacije mogu imati četiri minimalne kardinalnosti:

1. Opcione i roditelje i decu (O-O),
2. Obavezne roditelje, a opcionu decu (M-O),
3. Opcione roditelje, a obaveznu decu (O-M) i
4. Obavezne roditelje i obaveznu decu (M-M).

Analiza je obavljena u dva pravca:

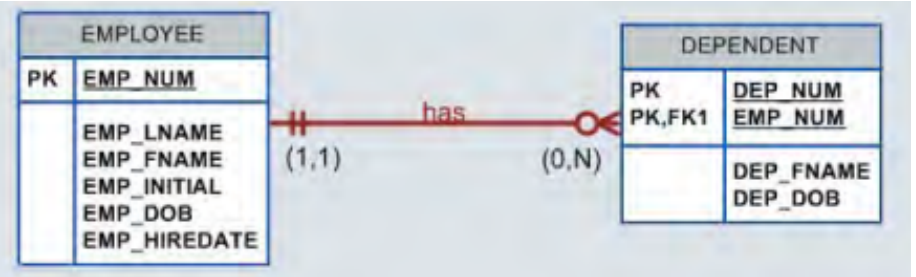
1. za slučaj kada su obavezni roditelji (relacije M-O i M-M) i
2. za slučaj kada su obavezna deca (O-M i M-M).

u slučaju relacije O-O ne treba preduzimati nikakve akcije.

AKCIJE NAD REDOVIMA RODITELJA U SLUČAJU OBAVEZNOG RODITELJA

Potrebno je obezbediti da svaki red tabele deteta ima validan, NOT NULL strani ključ.

Transformacija minimalne kardinalnosti u slučaj kada su obavezni roditelji se vrši tako što svaki red tabele deteta ima validan, NOT NULL strani ključ. Da bi se to postiglo, potrebno je ograničiti akcije ažuriranja i brisanja primarnog ključa roditelja i akcije kreiranja i modifikovanja stranog ključa kod dece.



Slika 4.1 Primer relacije u kojoj je obavezan roditelj Izvor: [1]

Akcije nad redovima roditelja u slučaju obaveznog roditelja:

1. U slučaju kreiranja novog roditelja ne treba raditi ništa, jer nijedan red deteta još uvek ne zavisi od tog novog reda.
2. U slučaju promene vrednosti postojećeg primarnog ključa roditelja, morale bi se promeniti sve postojeće vrednosti stranih ključeva dece (ako ih ima) tako da se podudaraju sa novim vrednostima primarnih ključeva roditelja, ili se modifikacija primarnih ključeva roditelja mora zabraniti. Politika menjanja stranih ključeva dece saglasno primarnim ključevima roditelja se naziva kaskadno ažuriranje.
3. U slučaju da se izbriše red tabele roditelja, moraju se brisati i svi postojeći odgovarajući redovi tabele dece (ako ih ima), ili se brisanje mora zabraniti. Brisanje dece saglasno brisanju roditelja se zove kaskadno brisanje.

AKCIJE NAD REDOVIMA DECE U SLUČAJU OBAVEZNIH RODITELJA

Zavise da li treba kreirati novi red deteta, promeniti strani ključ deteta ili izbrisati red deteta.

Akcije nad redovima dece u slučaju obaveznih roditelja:

1. U slučaju kreiranja novog reda dece, važno je obezbediti da taj red ima validan strani ključ, ako se on zahteva. U suprotnom, kreiranje reda nije dozvoljeno. Često, postoji neka podrazumevajuća (default) vrednost koja se dodeljuje stranom ključu i odgovara vrednosti primarnog ključa roditelja.
2. U slučaju da se strani ključ deteta menja, nova vrednost mora da se podudara sa vrednošću primarnog ključa roditelja. U suprotnom, menjanje je zabranjeno.
3. U slučaju brisanja redova dece, nema nikakvih restrikcija. Redovi dece se mogu izbrisati bez ikakvih konsekvenci na redove roditelja.

Akcije koje treba preduzeti u slučaju obaveznog roditelja se sumarno mogu prikazati tabelom na slici 4. 2 :

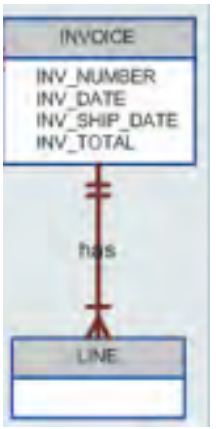
Obavezan roditelj	Akcije nad roditeljem	Akcije nad decom
INSERT	Ništa	Nađi roditelja
Modifikovanje ključa	• Promeniti strani ključ dece tako da se slaže sa novom vrednošću roditelja (kaskadno ažuriranje) • Zabrani	OK, ako se vrednost stranog ključa podudara sa postojećim roditeljima
DELETE	• Izbrisati dete (kaskadno brisanje) • Zabrani	Nema

Slika 4.2 Akcije u slučaju obaveznoh roditelja nad redovima roditelja i dece [Izvor: NM IT350-2020/2021.]

AKCIJE NAD REDOVIMA RODITELJA U SLUČAJU OBAVEZNE DECA

Znači da treba obezbediti bar jedan red u tabeli dece za svaki red tabele roditelja.

Ako su obavezna deca, to znači da treba obezbediti bar jedan red u tabeli dece za svaki red tabele roditelja. Poštovanje ovog ograničenja je mnogo teže od prethodnih kada je obavezan roditelj. Obaveznost roditelja se može obezbediti usaglašavanjem vrednosti stranih i primarnih ključeva, dok se za proveru obaveznosti dece mora uvek sračunavati broj dece koji je pridružen svakom roditelju, za šta treba napisati odgovarajući kod.



Slika 4.3 Primer relacije kada su obavezna deca Izvor: [1]

Akcije nad redovima roditelja kada su obavezna deca:

1. Kreiranje novog reda roditelja podrazumeva i kreiranje reda dece. To znači da moramo ili naći neki postojeći red u tabeli dece i promeniti njegov strani ključ tako da se podudara sa primarnim ključem roditelja, ili moramo kreirati novi red dece u istom trenutku kada kreiramo i red roditelja. Ako se ne preduzmu obe akcije, kreiranje je zabranjeno.
2. U slučaju da se primarni ključ roditelja ažurira, mora se ažurirati i strani ključ bar jednog reda deteta, ili se ažuriranje ne sme dozvoliti. Ovo ograničenje se nikad ne odnosi na roditelje sa surogat ključem, jer se njegova vrednost nikada ne menja.

3. Ako se red roditelja izbriše, ne treba preduzimati nikakve akcije, jer obavezno je postojanje deteta a ne roditelja.

AKCIJE NAD REDOVIMA DECE KADA SU OBAVEZNA DECA

Zavise od toga da li treba kreirati novi red deteta, promeniti strani ključ deteta ili izbrisati red deteta.

Akcije nad redovima dece kada su obavezna deca:

1. Prilikom ubacivanja novog reda dece ne treba preduzimati nikakve akcije.
2. Prilikom ažuriranja stranog ključa dece, ne postoje nikakve restrikcije osim u specijalnom slučaju kada je dete zadnje za datog roditelja. U tom slučaju se ažuriranje ne sme izvršiti. Međutim, da bi se odredio broj dece za datog roditelja, potrebno je napisati proceduru
3. Prilikom brisanja redova dece postoje slične restrikcije kao i prilikom ažuriranja.

Akcije koje treba preduzeti u slučaju obavezne dece se sumarno mogu prikazati sledećom tabelom:

Obavezno dete	Akcije nad roditeljem	Akcije nad decom
INSERT	• Nađi dete • Zabrani	Nema
Modifikovanje ključa	• Ažuriraj strani ključ (bar jednog) deteta • Zabrani	• Ako nije zadnje dete, OK • Ako je zadnje, zabrani ili zameni
DELETE	Nema	• Ako nije zadnje dete, OK Ako je zadnje, zabrani ili zameni

Slika 4.4 Akcije nad redovima roditelja i dece kada su obavezna deca [Izvor: NM IT350-2020/2021.]

SUMIRANE AKCIJE KOJE PODRŽAVAJU MINIMALNU KARDINALNOSTI

Prikazane tabelarno.

U sledećoj tabeli su sumirane akcije koje treba preduzeti u aplikaciji da bi se podržao svaki od tipova minimalne kardinalnosti.

Realizacija akcija koje podržavaju M-O kardinalnost: (potrebno je da svako dete ima svog roditelja). Ove akcije obezbeđuje sam DBMS korišćenjem dva ograničenja:
1. **Definisanjem ograničenja referencijalnog integriteta** koji obezbeđuje da svaka vrednost stranog ključa ima odgovarajuću vrednost u tabeli roditelja.
Na primer: ImeOdeljenja u ZAPOSLENI mora da postoji u ImeOdeljnja u ODELJENJE

Minimalna kardinalnost relacije	Akcija koju treba primeniti	Napomena
O-O	Ništa	
M-O	Preduzeti akcije nad roditeljem	Podržava sam DBMS; definisati ograničenje referencijalnog integriteta i za strani ključ staviti NOT NULL vrednost
O-M	Preduzeti akcije nad decom	Teško se može podržati; Zahteva korišćenje trigera ili drugog aplikativnog koda;
M-M	Preduzeti akcije i nad roditeljem i nad decom	Teško se može podržati; Zahteva kombinaciju složenih trigera. Trigeri mogu zaključati jedan drugog što stvara mnogo problema.

Slika 4.5 Akcije koje treba podržati za svaki od tipova minimalne kardinalnosti [Izvor: NM IT350-2020/ 2021.]

2. **Stavljanjem karakteristike NOT NULL za kolonu stranog ključa**. Postoji i opcija kojom se definiše da li ažuriranje i brisanje kaskadno ili je zabranjeno.

Implementiranje akcija koje podržavaju O-M kardinalnost: DBMS ne obezbeđuje ove akcije, već se one mogu implementirati korišćenjem trigera. Sa strane roditelja, treba napisati trigere na INSERT i UPDATE redova u tabeli roditelja kojima treba ili kreirati odgovarajući red u tabeli deteta ili “ukrasti” postojeći red deteta od nekog drugog roditelja. Ako se ove akcije ne mogu izvršiti, INSERT i UPDATE se moraju zabraniti.

Sa strane deteta, red deteta se može insertovati bez ikakvih problema. Kada se dete veže za roditelja, ta se veza ne može ukinuti ako je to poslednje ili jedino dete. To znači da za redove dece treba napisati trigere na UPDATE i DELETE sa sledećom logikom: Ako je strani ključ null, red nema svog roditelja pa se UPDATE i DELETE mogu dozvoliti. Ako strani ključ mora da ima vrednost, proveriti da li je red deteta poslednji. Ako jeste, red roditelja se mora brisati, ili se UPDATE i DELETE moraju zabraniti.

Poglavlje 5

Reinženjering baza podataka

KADA SE RADI REINŽENJERING BAZA PODATAKA

Za to postoji nekoliko situacija

Postoji nekoliko situacija u kojima se može zahtevati reinženjering baze podataka:

1. Problemi sa performansama softvera uzrokovani ne efikasnom bazom podataka.
2. Promene u zahtevima za softverom koji zahtevaju ažuriranje strukture podataka.
3. Značajne razlike u najnovijoj verziji baze podataka u poređenju sa ranijim verzijama, gde ove razlike zahtevaju promene u bazi podataka.

Kada se na dnevnom redu nalazi reinženjering baza podataka, postoji niz mogućih pristupa. Oni su zasnovani na sledećim promenama u bazi podataka:

1. Redizajn: Ako je postojeća struktura podataka loša, njeno redizajniranje može rezultirati većim performansama.
2. Migracija na, ili dodavanje baze podataka istog tipa: Korišćenje druge baze podataka koja ima prihvatljive indikatore u poređenju sa postojećom, može poboljšati performanse softverskog sistema ili rešiti druga pitanja.
3. Migracija u drugu vrstu baze podataka: U nekim slučajevima, korišćenje ne-relacionih baza podataka umesto relacionih može rešiti probleme sa performansama.

REDIZAJNIRANJE BAZA PODATAKA

U slučaju da struktura podataka ne zadovoljava prvu i drugu normalnu formu ili ne odgovara promenjenim zahtevima korisnika

Ako se prilikom analize strukture baze podataka uoči da ona ne zadovoljava prvi i drugi normalni oblik, trebalo bi uraditi redizajniranje baze podataka. To znači da ćete možda morati promeniti tabele ili kolone, ili ukinuti odgovarajuće veze.

Prilikom redizajna baze podataka, mogu biti potrebne značajne nadogradnje i promene koda. Svaka promena ili redizajn strukture podataka će izazvati modifikaciju koda na strani servera.

Primer 1: *Šema baze podataka uključuje oko 450 tabela. Analiza podataka je pokazala da postoji problem sa nekom od ovih tabela. Prilikom reinženjeringa izvršene su sledeće promene:*

1. *dodate su ili uklonjene neke kolone;*

2. *ukinuti su svi neophodni elementi koji su bili nepropisno dizajnirani i umesto toga dodati novi.*

Kao rezultat, shema je uključivala više od 450 tabela. Performanse sistema su poboljšane pa je sistem brže radio zahvaljujući dodavanju novih bitnih elemenata (što uključuje kaskadne operacije, indekse, veze između tabela i primarne ključeve) i unapređenju postojećih (na primer, gde je to bilo moguće, uvedeni su jednostavniji tipovi podataka).

Primer 2: *U ovom slučaju, reinženjering baze podataka nije bio potreban zbog problema sa performansama, već promenama u poslovnoj logici, koja je zahtevala uvođenje novih podataka. Struktura baze podataka se tako promenila, a neke tabele su bile suvišne.*

Kao rezultat reinženjeringa, modifikovana baza podataka je postala relevantna za novu poslovnu logiku.

Dakle, razlozi za redizajniranje su: performanse, promena zahteva, povećanje obima podataka, redukcija redundancije, unapređenje integriteta podataka, prilagođavanje novim tehnologijama, održavanje, bezbednost i privatnost, integracija sistema.

MIGRACIJA ILI DODAVANJE BAZE PODATAKA ISTOG TIPA

Podrazumeva prevođenje relacione baze podataka iz RDBMS-a jednog proizvođača i RDBMS drugog proizvođača.

Migracija i dodavanje baze podataka istog tipa su dva različita pristupa koji se mogu primeniti u kontekstu proširenja ili unapređenja sistema sa bazom podataka.

Migracija baze podataka podrazumeva prenos podataka iz jednog sistema baze podataka u drugi sistem baze podataka. Ovo se obično radi kada postojeći sistem više ne zadovoljava zahteve, kada se prelazi na drugi sistem za upravljanje bazom podataka (DBMS), ili kada se vrši prelazak na novu verziju DBMS-a. Migracija zahteva pažljivo planiranje kako bi se osiguralo da se svi podaci prenesu tačno i bez gubitaka, kao i da se očuva referencijalni integritet. Ovo je posebno važno ako se menjaju tipovi podataka ili struktura baze podataka.

Dodavanje baze podataka istog tipa podrazumeva kreiranje nove baze podataka koja se koristi kao dodatak postojećoj bazi podataka. Ovo se često radi kada je postojeća baza podataka dostigla svoje kapacitete ili kada je potrebno odvojiti određene delove podataka u zasebnu bazu zbog organizacionih ili performacijskih razloga. Na primer, ako se sistem koristi za upravljanje prodajom i skladištem, može biti korisno odvojiti te dve funkcije u zasebne baze podataka radi bolje organizacije i efikasnosti.

Primer 3: *Razvijen je softverski proizvod za obradu podataka sačuvanih u Microsoft SQL Serveru. Novi klijent je imao sve podatke sačuvane u PostgreSQL-u i te podatke je koristio u sopstvenoj aplikaciji, što znači da se podaci nisu mogli migrirati u drugu bazu podataka. Softverski proizvod je trebalo da bude modifikovan tako da pored Microsoft SQL Server može koristiti i PostgreSQL. Svaka baza podataka ima svoje specifične karakteristike. Da bi se minimizirale promene kod jednog i drugog, bilo je potrebno ukinuti neke specifične funkcije i koristiti samo opšte. Pored toga, bilo je potrebno promeniti strukturu podataka u obe baze*

podataka kako bi one bile kompatibilne. Ipak, neophodno je bilo kreirati poseban kod za svaku bazu podataka.
Ključne razlike između migracije i dodavanja baze podataka istog tipa su:
Migracija uključuje prenos podataka iz jednog sistema baze podataka u drugi, dok dodavanje podrazumeva kreiranje nove baze podataka pored postojeće.

Migracija može zahtevati konverziju podataka i prilagođavanje strukture kako bi se podudarali sa ciljnim sistemom, dok dodavanje baze podataka koristi isti tip baze podataka kao postojeća.

Migracija može biti složenija i zahtevnija, jer podrazumeva prenos i sinhronizaciju podataka, dok dodavanje baze podataka može biti jednostavnije i manje rizično.

Poglavlje 6

Pokazna vežba

NAČIN ORGANIZACIJE POKAZNIH VEŽBI

Organizacija pokaznih vežbi

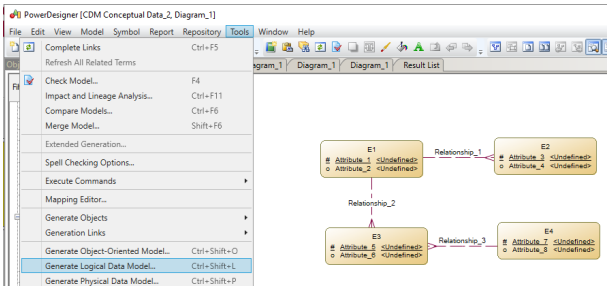
Vežba je organizovana kroz uvod deo i deo za samostalni rad studenata.
U uvodnom delu pokaznih vežbi se daje pokazni primer koji studentima treba da pomogne u samostalnom rešavanju zadataka.
Zadatke koji su zadati za samostalni rad student samostalno rešava uz pomoć asistenta.

6.1 Pokazni primer - 1. deo

GENERISANJE LOGIČKOG IZ KONCEPTUALNOG MODELA BAZE PODATAKA - (5 MIN.)

Način generisanje logičkog iz konceptualnog modela baze podataka u Power Designer-u

Na slici 7.1 je prikazan način za generisanje logičkog iz konceptualnog modela baze podataka u CASE alatu Power Designer koji je objašnjen u prethodnoj lekciji a ovde ponovljen radi lakšeg snalaženja.



Slika 6.1.1 Generisanje logičkog iz konceptualnog modela baze podataka [Izvor: NM IT350-2020/2021.]

Kao što se slike može videti, najpre treba izabrati opciju Tool a zatim opciju Generate Logical data model.

Izgled logičkog modela podataka za primer konceptualnog modela podataka sa slike 7.1 prikazan je na slici 7.2.

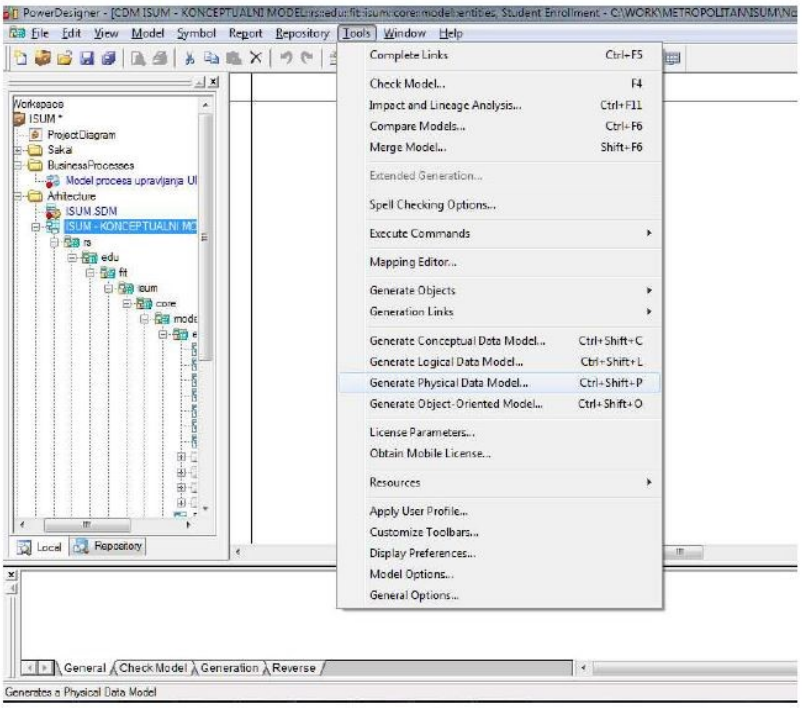
Slika 6.1.2 Logički model podataka za konceptualni model podataka sa slike 1. [Izvor: NM IT350-2020/2021.]

GENERISANJU FIZIČKOG IZ LOGIČKOG MODELA (1. KORAK) - (5 MIN.)

Odnosi se na kreiranje samog fizičkog modela

Korišćenjem PowerDesigner-a, generisanje fizičkog iz logičkog modela baze podataka se odvija u nekoliko koraka koji će ovde biti opisani:

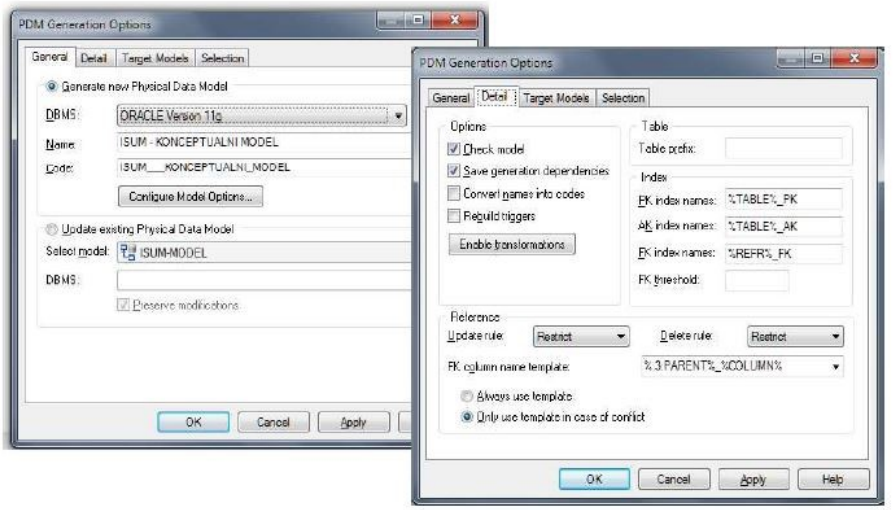
Prvi korak je kreiranje samog fizičkog modela. Opcija za kreiranje fizičkog modela sa nalazi u kartici Tools, kao na slici 7.3.



Slika 6.1.3 Kartica TOOLS [Izvor: NM IT350-2020/2021.]

Odabirom opcije *Generate Phisical Data Mod*el dobijamo prozor kao na slici 7.4.

Kao DBMS treba odabrati MySQL 5.0

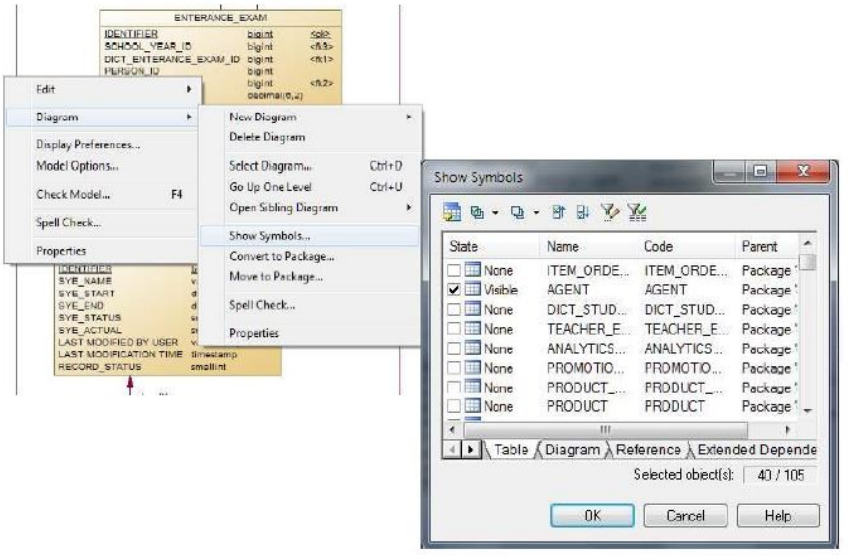


Slika 6.1.4 Ekran dobijen nakon izbora opcije Generate Phisical Data Model [Izvor: NM IT350-2020/2021.]

GENERISANJU FIZIČKOG IZ LOGIČKOG MODELA (2. KORAK) - (5 MIN.)

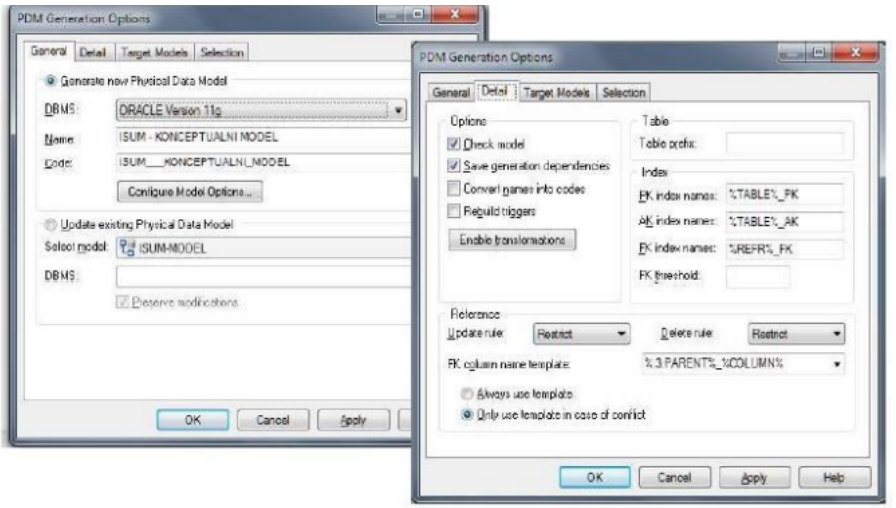
Imenovanje referenci.

Nakon kreiranja fizičkog modela, možemo izabrati tabele koje želimo da se vide u dijagramu koji prikazuje fizički model, kao na slici 7.5.



Slika 6.1.5 Opcija za izbor simbola [Izvor: NM IT350-2020/2021.]

Sledeći korak jeste imenovanje referenci kao na slici 7.6. Do pozora kao na slici dolazi se iz kartice *Model*, opcija R*eferences*

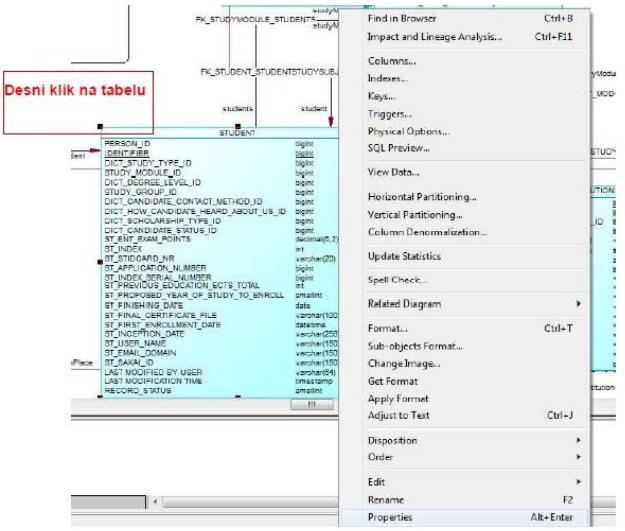


Slika 6.1.6 Primer imenovanja referenci [Izvor: NM IT350-2020/2021.]

GENERISANJU FIZIČKOG IZ LOGIČKOG MODELA (3. KORAK) - (5 MIN.)

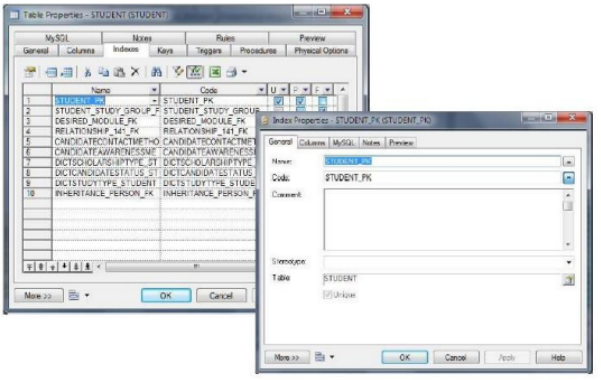
Pregled tabela, kolona, ključeva i indeksa.

Treći korak se odnosi na pregledavanje tabela, kolona, ključeva i indeksa koje se vrši tako što se na tabelu klikne desnim klikom i odabere opcija *Properties*, kao na slici 7.7.

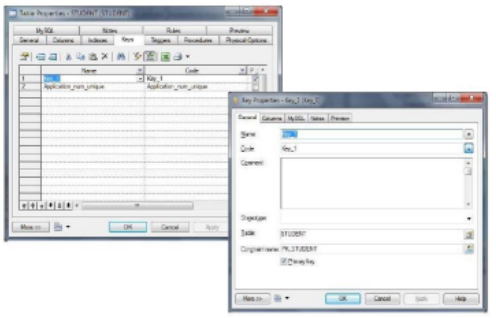


Slika 6.1.7 Pregled tabela, kolona, ključeva i indeksa [Izvor: NM IT350-2020/2021.]

Potom se dobijaju prozori kao na slikama 7.8 i 7.9 u nastavku.



Slika 6.1.8 Pregled tabela, kolona, ključeva i indeksa [Izvor: NM IT350-2020/2021.]

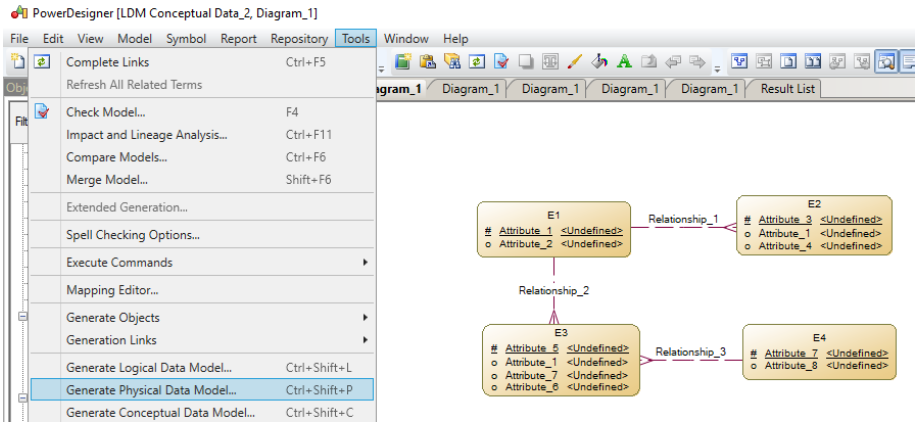


Slika 6.1.9 Pregled tabela, kolona, ključeva i indeksa [Izvor: NM IT350-2020/2021.]

POKAZNI PRIMER BR. 1 - (5 MIN.)

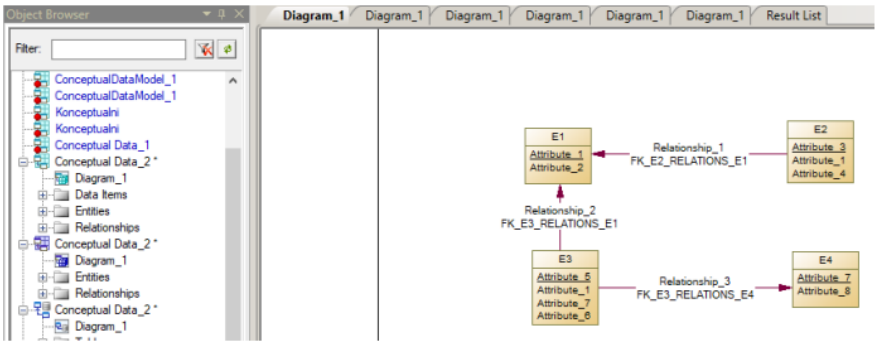
1. primer transformacije logičkog u fizički model

Na slici 7.10 je kao ilustrativan primer dat prikaz kreiranja fizičkog modela za logički model podataka koji je kreiran za dati konceptualni model podataka.



Slika 6.1.10 Generisanje fizičkog za logički model podataka [Izvor: NM IT350-2020/2021.]

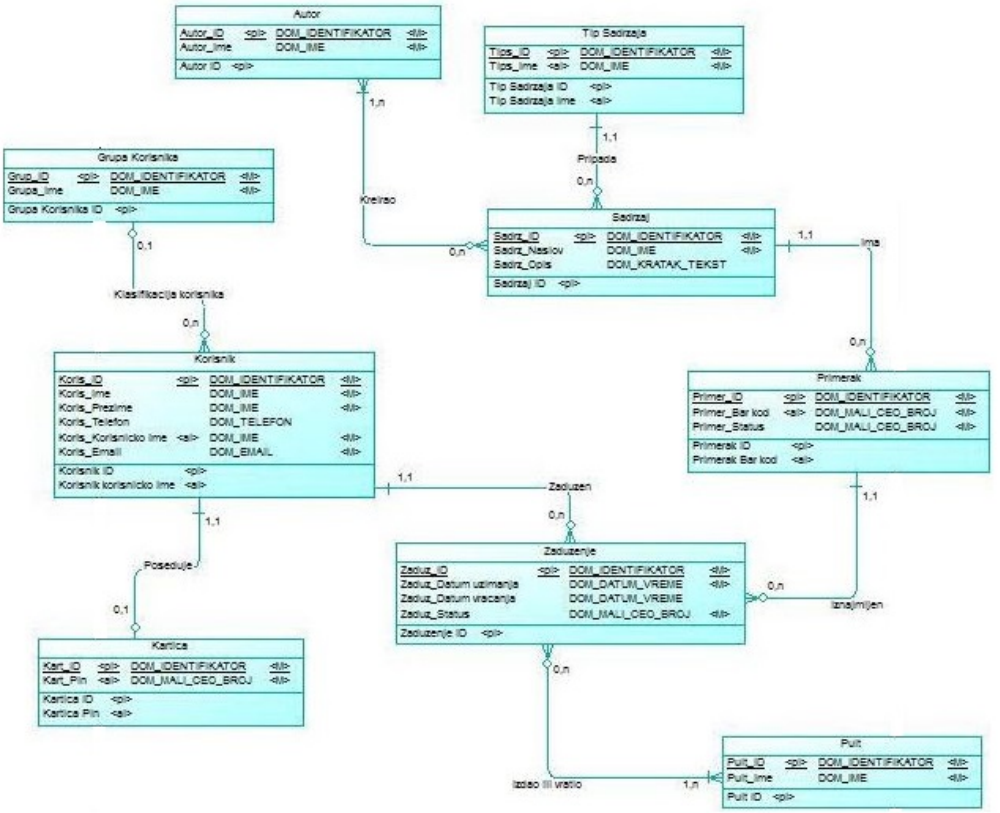
Slika 7.11 prikazuje izgled fizičkog modela podataka za dati logički model podataka.



Slika 6.1.11 Fizički model podataka za dati logički model podataka [Izvor: NM IT350-2020/2021.]

POKAZNI PRIMER BR. 2 - (5 MIN.)

Prikaz logičkog modela za koji treba kreirati fizički.



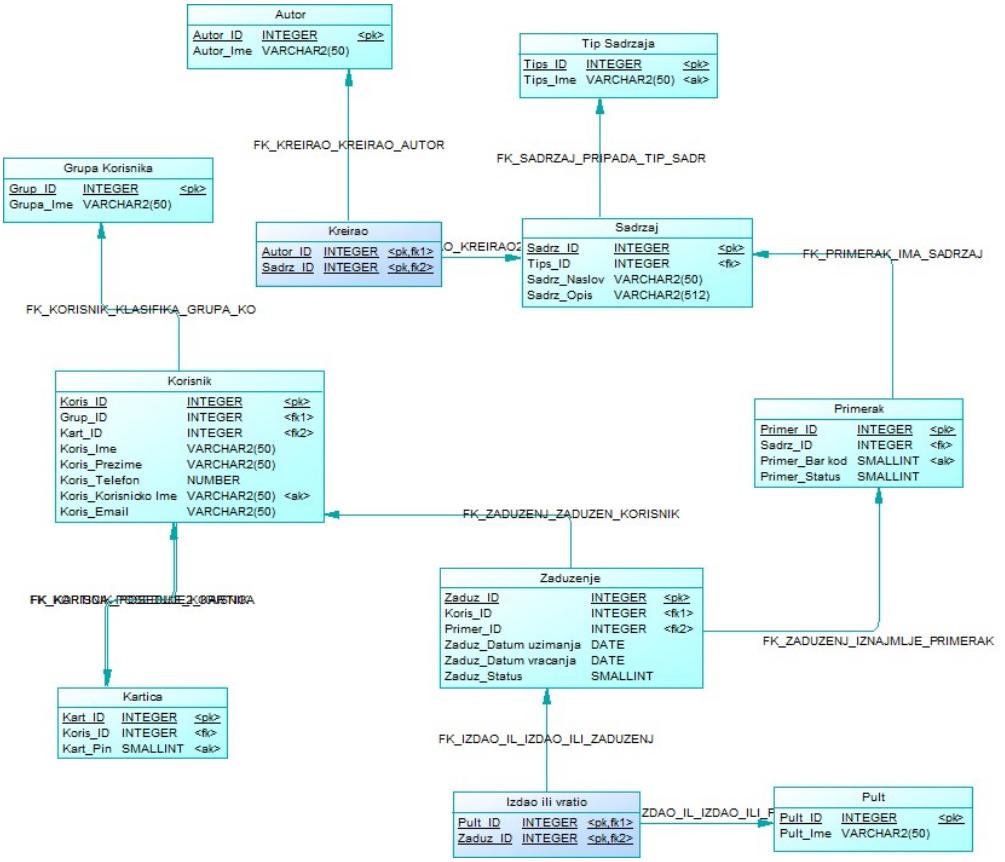
Slika 6.1.12 Logički model podataka savremene univerzitetske biblioteke [Izvor: NM IT350-2020/2021.]

U prethodnoj lekciji je urađen konceptualni model podataka jedne savremene univerzitetske biblioteke za koji je prikazan na slici 7.12 odgovarajući logički model dobijen transformacijom u Power Designer-u.

Za predstavljeni logički model treba napraviti fizički model podataka.

POKAZNI PRIMER BR. 2 - REŠENJE - (5 MIN.)

Prikaz dobijenog fizičkog modela opisanim postupkom.



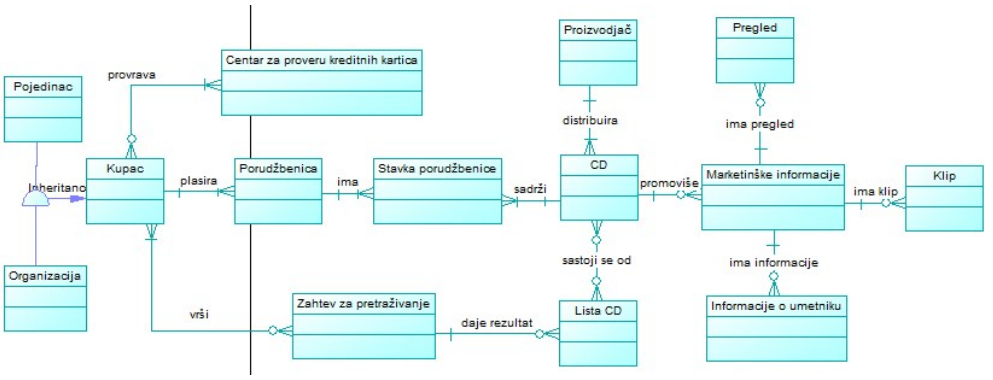
Slika 6.1.13 Fizički model podataka savremene univerzitetske biblioteke [Izvor: NM IT350-2020/2021.]

Fizički model koji je dobijen iz logičkog modela prikazan je na slici 7.13.

POKAZNI PRIMER BR. 3 - (5 MIN.)

Prikaz konceptualnog modela za koji treba kreirati fizički

Na slici 7.14 je prikazan konceptualni model podataka Sistema za Internet prodaju u CD Selections koji treba transformisati najpre u logički a zatim i u fizički model baze podataka korišćenjem PowerDesigner-a

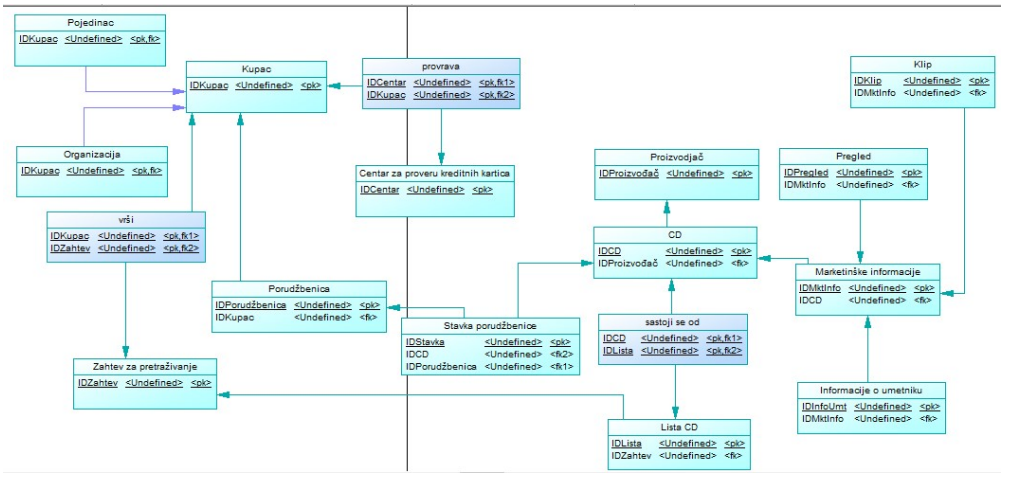


Slika 6.1.14 Logički model podataka Sistema za Internet prodaju u CD Selections koji treba transformisati u fizički model baze podataka [Izvor: NM IT350-2020/2021.]

POKAZNI PRIMER BR. 3 - REŠENJE - (5 MIN.)

(b) Prikaz dobijenog fizičkog modela opisanim postupkom.

Nakon izvršene transformacije najpre u logički a zatim i u fizički model prikazan na slici 7.15, treba izvršiti verifikaciju dobijenog fizičkog modela.



Slika 6.1.15 Fizički model za Internet prodaju u CD Selections [Izvor: NM IT350-2020/2021.]

6.2 Pokazni primer - 2. deo

KAKO KREIRATI .SQL SKRIPTOVE ZA KREIRANJE ILI IZMENU TABELA? - (3 MIN.)

Mogu se koristiti CASE alat u kojem je generisan fizički model.

Pošto su kreirani fizički modeli, oni se mogu transformisati u specifičnu prezentaciju baze podataka (vezanu za određeni sistem za upravljanje bazom podataka) u formi .sql skriptova

Za transformaciju fizičkog modela u odgovarajuće naredbe za kreiranje tabela se može koristiti CASE alat u kojem je fizički model generisan. Međutim, pre same transformacije, potrebno je postaviti odgovarajuće parametre za transformaciju, svojstva, kao i neke dodatne opcije transformacije.

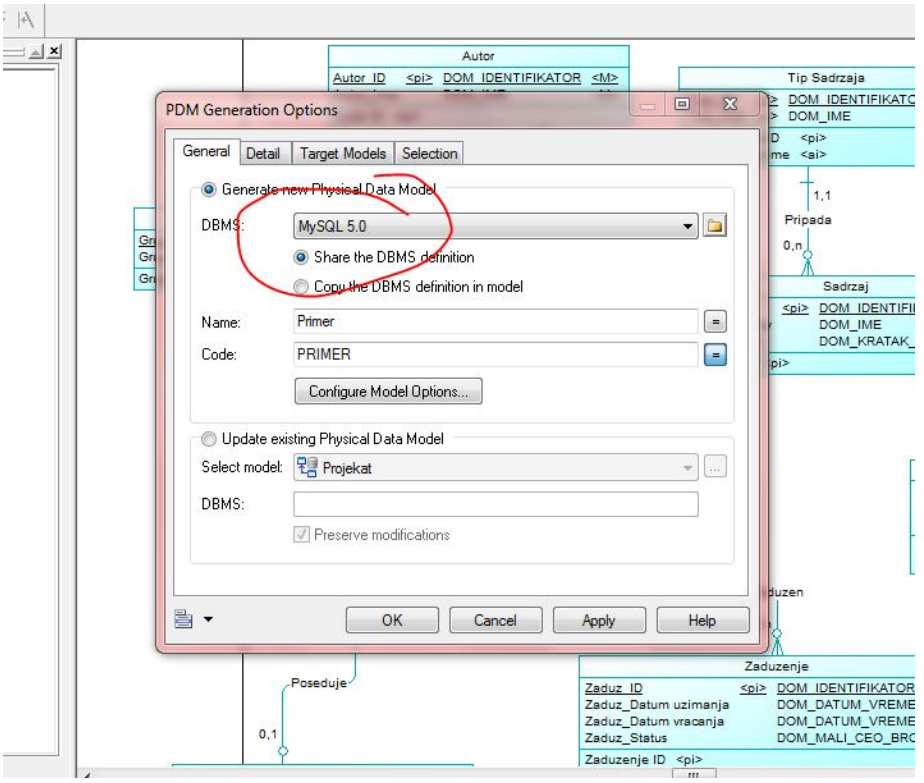
Na osnovu fizičkog modela baze podataka, se može kreirati skripta za generisanje novih tabela ili za izmenu postojećih tabela u bazi podataka.

Sam proces generisanja .sql skripti se vrši kroz nekoliko koraka što je prikazano u daljem tekstu.

1. KORAK U GENERISANJU .SQL SKRIPTE - (7 MIN.)

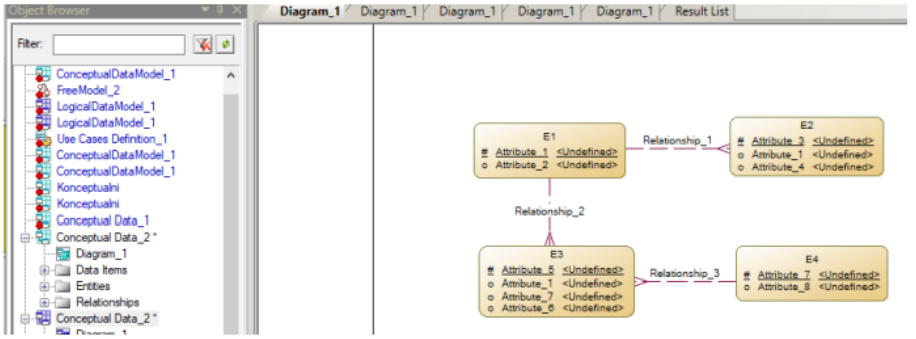
Prilikom kreiranja fizičkog modela se bira za koji RDBMS treba kreirati model.

Prilikom kreiranja skripti za generisanje baze podataka potrebno je izabrati za koji RDBMS se relacioni model kreira. (u ovom slučaju MySQL, verzija 5.0.) što je prikazano na slici 7.1.



Slika 6.2.1 Odabir MySQL-a prilikom generisanja frelacionog modela iz konceptualnog [Izvor: NM IT350-2020/2021.]

Nakon kreiranja relacionog model otvorio se meni *database*. (slika 7.2)

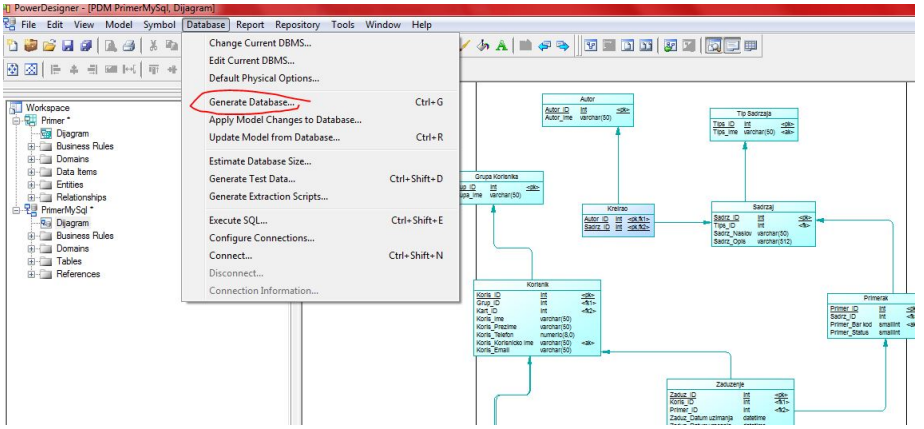


Slika 6.2.2 Meni Database [Izvor: NM IT350-2020/2021.]

2. KORAK U GENERISANJU .SQL SKRIPTE (7 MIN.)

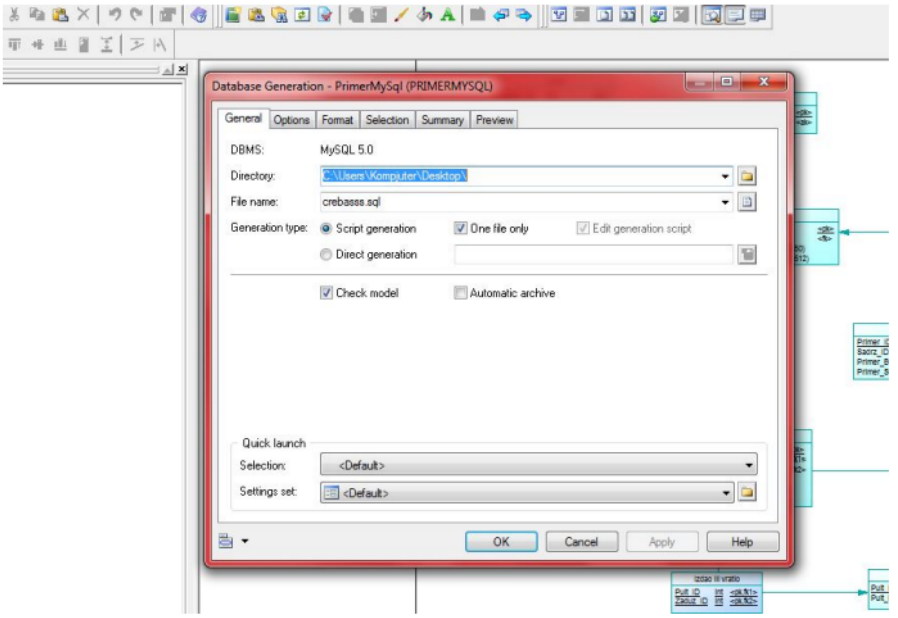
Prikaz opcije General.

Da bi generisali .sql skriptu, iz menija Database treba izabrati opciju *generate database* (slika 7.3) kada se otvara meni odakle treba izabrati opcije za generisanje .sql skripte.



Slika 6.2.3 Generate Database [Izvor: NM IT350-2020/2021.]

Opcije koje postoje prilikom generisanja skripti za kreiranje .sql file-a su prikazane na sledećim slikama. Izborom opcije *Generale* se može izabrati direktorijum u kome da se smešta .sql fajl u ovom slučaju pod nazivom crebasNajnoviji.sql (slika 7.4).

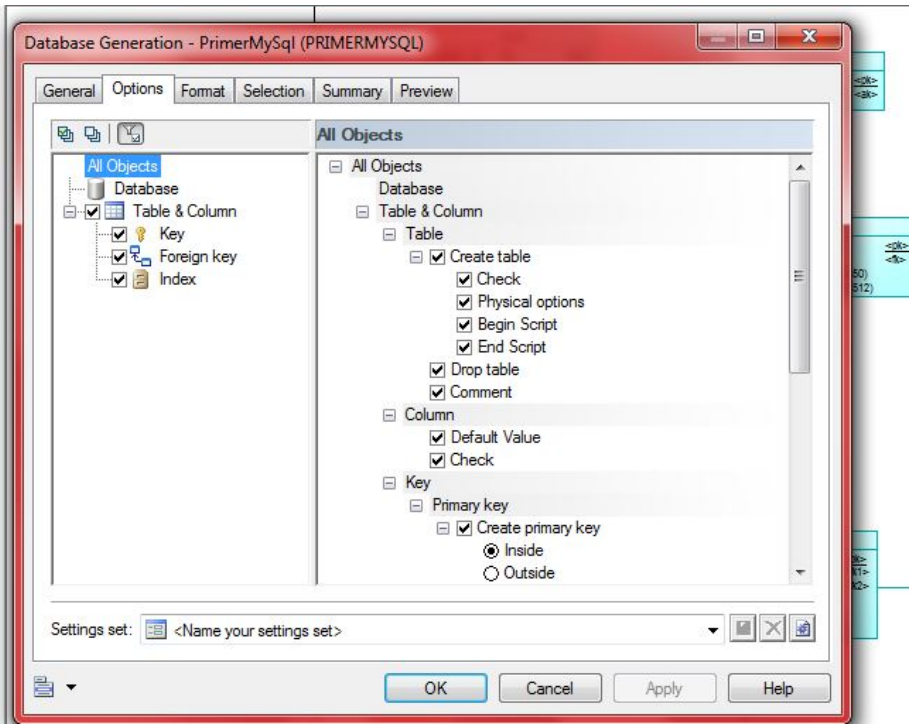


Slika 6.2.4 Opcija General [Izvor: NM IT350-2020/2021.]

3. KORAK U GENERISANJU .SQL SKRIPTE (7 MIN.)

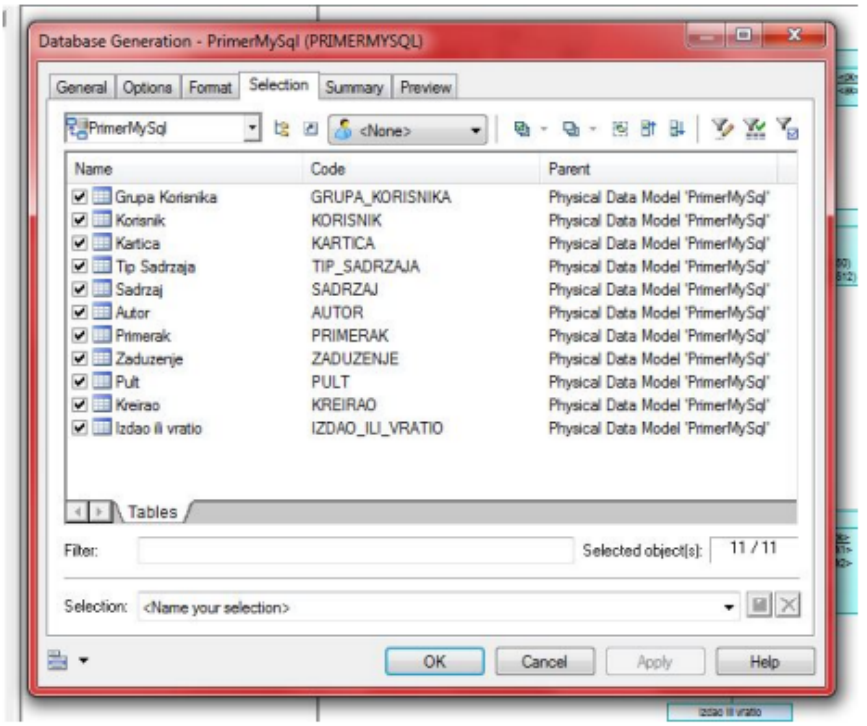
Prikaz Meni i Select opcija.

Izborom opcije *Options* određuje se šta želimo da generišemo i na koji način. (slika 7.5)



Slika 6.2.5 Opcija Options [Izvor: NM IT350-2020/2021.]

Izborom opcije *Select* se omogućava da se odredi koje tabele želimo da generišemo. (slika 7.6)

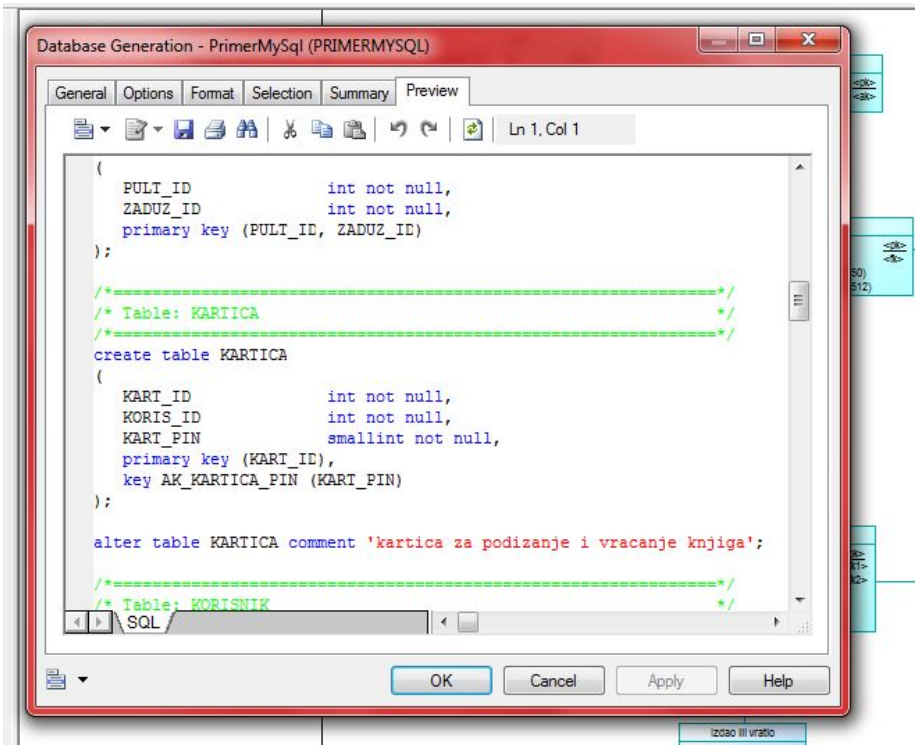


Slika 6.2.6 Opcija Select [Izvor: NM IT350-2020/2021.]

4. KORAK U GENERISANJU .SQL SKRIPTE (7 MIN.)

Opcija review

Opcija *Review* služi za pregledavanje .SQL skripte pre nego što se ona sačuva u .sql file na izabranom direktorijumu (slika 7.7).



Slika 6.2.7 Opcija review [Izvor: NM IT350-2020/2021.]

Nakon pritiska na OK dugme dolazi do generisana skripte koja se pamti. Izgled skripte je dat na slici 7.8.

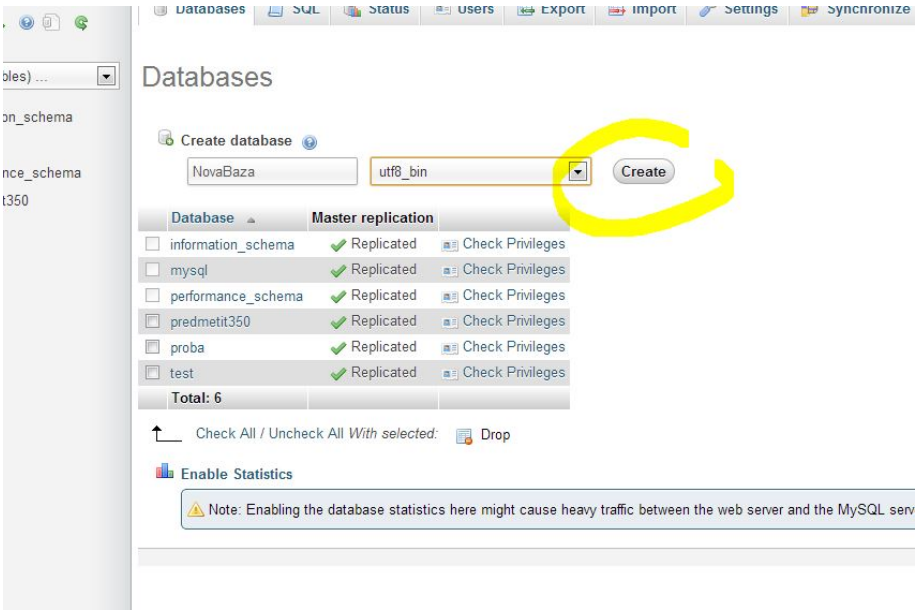
```
drop table if exists ZADUZENJE;

/*==========*/
/* Table: AUTOR */
/*==========*/
create table AUTOR
(
   AUTOR_ID2            int not null,
   AUTOR_IME            varchar(50) not null,
   primary key (AUTOR_ID2)
);

alter table AUTOR comment 'Autor sadrzaja';

/*==========*/
/* Table: GRUPA_KORISNIKA */
/*==========*/
create table GRUPA_KORISNIKA
(
   GRUP_ID              int not null,
   GRUPA_IME            varchar(50) not null,
   primary key (GRUP_ID)
);

alter table GRUPA_KORISNIKA comment 'grupisanje korisnika';

/*==========*/
/* Table: IZDAO_ILI_VRATIO */
/*==========*/
create table IZDAO_ILI_VRATIO
(
   PULT_ID              int not null,
   ZADUZ_ID             int not null,
   primary key (PULT_ID, ZADUZ_ID)
);

/*==========*/
/* Table: KARTICA */
/*==========*/
create table KARTICA
(
   KART_ID              int not null,
   KORIS_ID             int not null,
   KART_PIN             smallint not null,
   primary key (KART_ID),
   key AK_KARTICA_PIN (KART_PIN)
);

alter table KARTICA comment 'kartica za podizanje i vracanje knjiga';
```

Slika 6.2.8 Prikaz SQL generisane skripte [Izvor: NM IT350-2020/2021.]

5. KORAK U GENERISANJU .SQL SKRIPTE (7 MIN.)

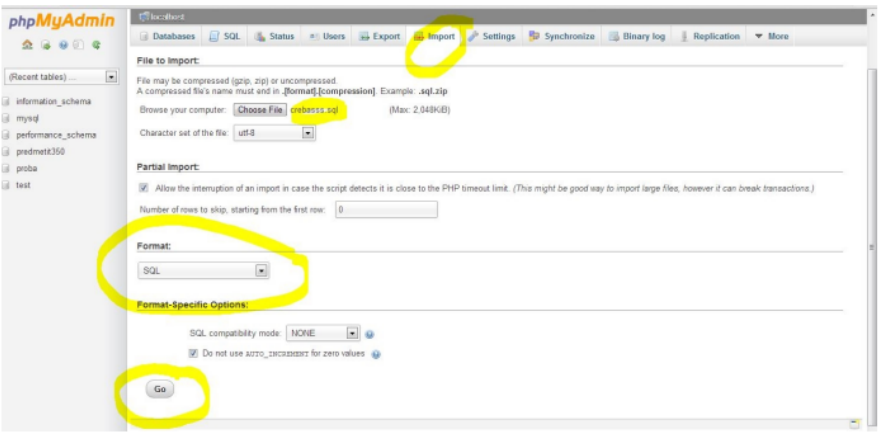
Importovanje baze podataka u phpMyAdmin.

Nakon kreiranja skripte potrebno je pokrenuti phpMyAdmin i napraviti novu bazu podataka (slika 7.9).



Slika 6.2.9 Kreiranje nove baze [Izvor: NM IT350-2020/2021.]

Nakon toga potrebno je ući na opciju import u phpMyAdmin i izabrati kreirani sql fajl. Za format je potrebno izabrati SQL i nakon toga kliknuti na opciju *go*. (slika 7.10)

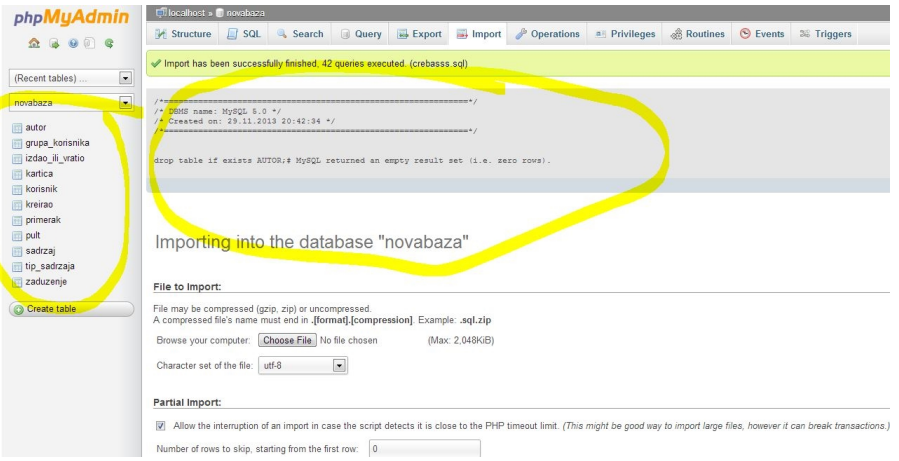


Slika 6.2.10 Prikaz importovanja baze [Izvor: NM IT350-2020/2021.]

6. KORAK U GENERISANJU .SQL SKRIPTE (7 MIN.)

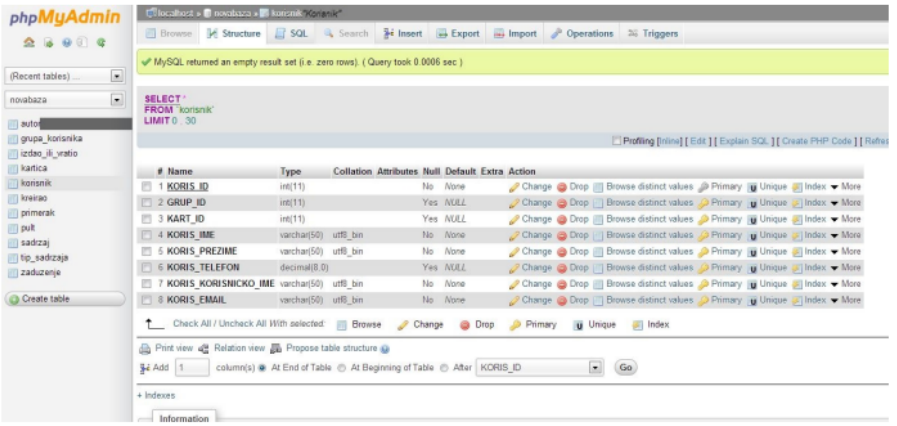
Prikaz importovane baze.

Nakon opisanog postupka tabele su kreirane što se vidi na slici 7.11.



Slika 6.2.11 Uspešno izvršeni uvoz baze [Izvor: NM IT350-2020/2021.]

Izborom jedne tabele može se videti njena detaljna struktura što se vidi na slici 7.12.



Slika 6.2.12 Prikaz jedne tabele [Izvor: NM IT350-2020/2021.]

6.3 Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD STUDENATA - POTREBNO VREME PO SVAKOM ZADATKU (1-3) JE 15 MINUTA.

Samostalno rešite sledeće zadatke:

Na osnovu logičkih šema baze podataka koji su dobijeni u okviru individualne vežbe iz lekcije 5 (zadaci 1-3):

1. Generisati odgovarajuće fizičke modele baze podataka
2. Na osnovu generisanih fizičkih modela, generisati .sql skripte za kreiranje tabela bate podataka

Potrebno vreme po svakom zadatku (1-3) je 15 minuta.

Poglavlje 7

Domaći zadatak

DOMAĆI ZADATAK 6 - VREME IZRADE 90 MIN.

Uraditi sledeće zadatke - vreme izrade svakog dela (1.-3.) je po 30 min.

Konceptualni model podataka predstavljen ER dijagramom u PowerDesigner-u iz domaćeg zadatka 4. treba:

1. Transformisati u logički model
2. Logički model transformisati u fizički model korišćenjem
3. Na osnovu fizičkog modela generisati skripte za kreiranje baze podataka

Prilikom transformacije iz logičkog u fizički model proveri da li su korektno primenjena pravila transformacije a ukoliko model sadrži veze generalizacija - specijalizacija, za transformaciju pod klasa i njihovih nad klasa koristi Object-oriented metod transformacije

Napomena: Modele navedene u zadatku poslati u odgovarajućem formatu (.cdm, .pdm, ...).

PRAVILA ZA SLANJE DOMAĆEG ZADATKA

Pravila

Prilikom slanja domaćih zadatka, neophodno je da ispunite sledeće:

- Subject mail-a mora biti IT250-DZbr (u slučaju kada šaljete domaći za ovu nedelju to je IT250-DZ06)
- U prilogu mail-a treba da se nalazi arhiviran projekat koji se ocenjuje imenovan na sledeći način: IT250-DZbr-BrojIndeksa-Ime Prezime. Na primer, IT250-DZ06-1234-VeljkoGrkovic
- Telo mail-a treba da ima pozdravnu poruku
- Arhivu sa zadatkom poslati na adresu predmetnog asistenta: milica.vlajkovic@metropolitan.ac.rs (studenti u Beogradu i online studenti) ili tamara.vukadinovic@metropolitan.ac.rs (studenti u Nišu).

Svi poslati mail-ovi koji ne ispunjavaju navedene uslove NEĆE biti pregledavani. Za sva pitanja ili nedoumice u vezi zadatka, možete se obratiti asistentu

ZAKLJUČAK

Šta smo naučili u ovoj lekciji?

U ovoj lekciji je opisan proces transformacije logičkog modela baze podataka u fizički model kroz više konkretnih primera.

Kroz ove primere je prikazan način transformacije tipova entiteta, atributa, relacija zavisno od njihove kardinalnosti u odgovarajuće elemente relacija (tabela) koje sačinjavaju relacioni (logički) model podataka.

Opisan je i način transformacije elemenata naprednijih ER modela kao što su slabi entiteti, asocijativne veze, rekurzivne relacije, generalizacija i specijalizacija. Naročita pažnja je posvećena transformaciji minimalne kardinalnosti veza i njen uticaj na način dobijanja sekundarnih ključeva.

Takođe je opisan način za kreiranje tabela u konkretnom RDBMS-u na osnovu relacionog modela.

LITERATURA:

Za pisanje ove lekcije korišćena je sledeća literatura:

1. http://corpgov.crew.ee/Materjalid/Database%20Systems%20-%20Design,%20Implementation,%20and%20Management%20(9th%20
2. Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, DATABASE SYSTEMS –The Department of Computer Science, Stanford University, 2009 by Pearson Education Inc.
3. David M. Kroenke, Database Processing – fundamentals, design and implementation, Prentice Hall, 2004.
4. C. J. Date, An introduction to Database Systems, Addison-Wesley Publishing Company, 1990
5. https://docs.oracle.com/cd/A84870_01/doc/server.816/a76994/physical.htm