Database Design
kurs 2017 – database design and programming with sql
kombinacija teachers i students slajdova

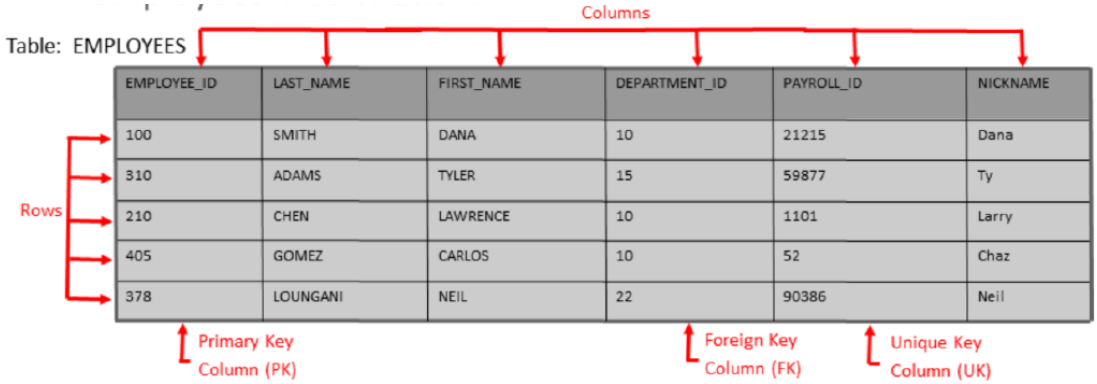
9-2 Osnovno mapiranje : Proces transformacije

Smisao

- Kada se dizajnira kuća, cilj je napraviti kuću. Iako ne radite baš izgradnju treba razumeti termine koji se koriste od strane građevinara da bi im pomogli sa konceptualnim dizajnom i napravili fizičku realnost. Inicijalni dbd se može koristiti za dalje diskusije između dizajnera, dba i razvoja aplikacija
- Kada kreiramo konceptualni model, fokusiramo se na pravila biznisa. Kada kreiramo dbd, fokus je na db teme smeštanja, brzine transakcija, sigurnost...Npr, u Data Werhouse, fizički model se često namerno denormalizuju za brže odvijanje
- Iako su ovo važne teme, one se ne ispituju pre biznis uslova. Data modeling ima fokus na biznis uslove, bez obzira na implementaciju. Možda imate najbrži i najsigurniji db u svetu, ali ako ne ispunjava vaše biznis uslove, neće biti od prevelike koristi.

Pogled na relacione tabele

- Tabela je obična struktura u kojima podatak se organizuje i smešta.
- U sledećem primeru, EMPLOYEES tabela se koristi za smeštanje info o zaposlenima

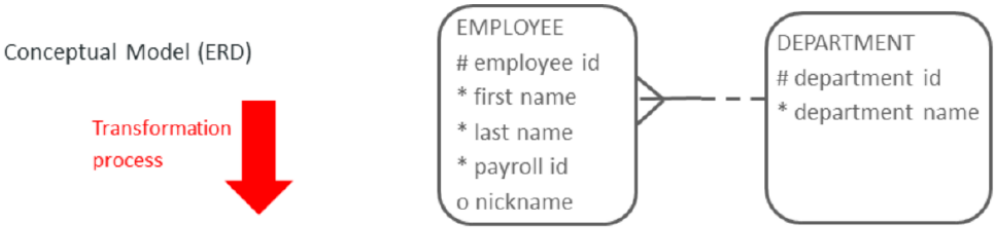


- Tabele imaju kolone i redove (vrste); u primeru svaki red opisuje pojave zaposlenih
- Svaka kolona se koristi za smeštanje posebnih tipova vrednosti, kao što su boj zaposlenog, prezime, ime
- Osnovni ključ je emmpoyee_id kolona
- Svaki zaposleni ima UID broj u ovoj tabeli; vrednost u koloni primarnog ključa se razlikuje po vrstama
- payroll_id je jedinstveni ključ; to znači da sistem ne dozvoljava dve vrste sa istom payroll_id
- FK kolona se odnosi na kolonu u drugoj tabeli; u ovom primeru, department_id se odnosi na kolonu u DEPARTMENTS tabeli
- znamo da Dana Smith radi u sektoru 10; ako želimo da znamo o Dana Smith sektoru, pogledaćemo u vrsti pod DEPARTMENTS tabeli koji ima department_id = 10

Transformacija konceptualnog u fizički

- Konceptualni model (ERD) je transformisan u fizički model
- Fizička implementacija će biti relaciona db

- Transformacija: promena elemenata ERD (entiteti, atributi, relacije) u db elemente (tabele, atributi, FK)
- EMPLOYEE entitet u ERD (konceptualni model) transformiše u dijagram od EMPLOYEES tabele, koji reprezentuje definiciju tabele u relacionom modelu (fizička implementacija)



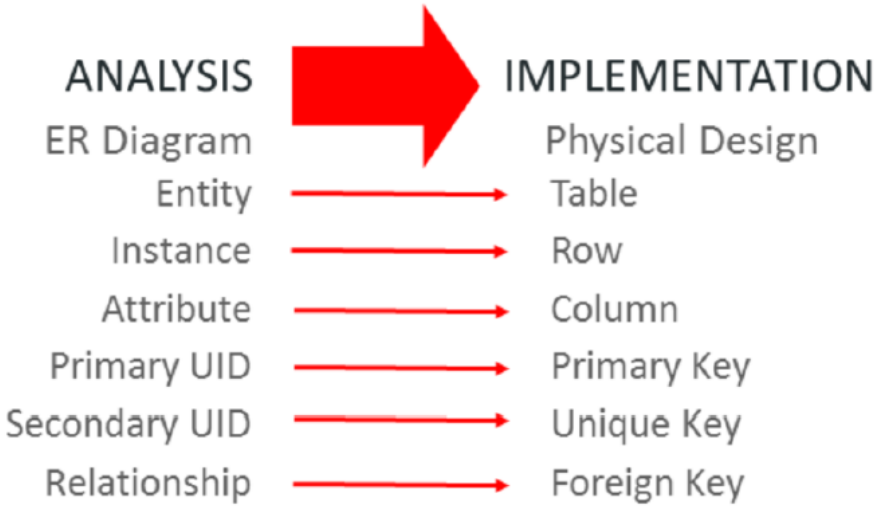
Physical Implementation: Relational Database

EMPLOYEES (EPE)		
Key type	Optionality	Column name
pk	*	employee_id
uk	*	payroll_id
	*	last_name
	*	first_name
	o	nickname
fk	*	department_id

DEPARTMENTS (DPT)		
Key type	Optionality	Column name
pk	*	department_id
	*	department_name

Terminologija mapiranja

- Promena iz analize (konceptualni model) do implementacije (fizički model) takođe znači promenu terminologije:
    1. entitet postaje tabela
    2. instanca postaje vrsta
    3. atribut postaje kolona
    4. primarni UID postaje primerni ključ
    5. sekundarni UID postaje jedinstveni ključ
    6. relacija se transformiše u FK kolonu i FK constraint
- Mapiranje: pridružiti elemente ERD (entiteti, atributi, relacije) sa db elementima (tabele, atributi, FK)
- Analiza i dizajn su faze ciklusa razvoja sistema (system development life cycle). Pri dizajnu sistema, analiza je pre dizajna. Modelovanje podataka se radi u fazi analize. Kada smo zadovoljni sa postignućem obuhvatanja zahteva biznisa u data modelu, ide se dalje na dizajn fazu, gde ERD je mapiran u fizičku implementaciju



Beleške u dijagramu tabela (table diagram notations)

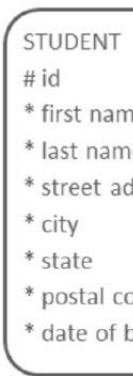
- Prvi red u dijagramu tabele sadrži ime tabele i njeno skraćeno ime
- Kolona Key Type treba da sadrži vrednosti pk za primarni ključ, uk za unique key, fk za foreign key kolonu
- U ovim jednostavnim primerima postoje jedan prema jedan mapiranje između konceptualne i fizičke terminologije (npr jedan entitet postaje jedna tabela) ali ovo nije uvek tako u kompleksnijim modelima

TABLE NAME (short name)		
Key Type (pk, uk, fk)	Optionality ("*", "o")	Column Name

- Biće prazno ako kolona nije deo nijednog ključa
- Kolona Optionality mora sadržati "*" ako je kolona obavezna, "o" ako je opcionalna. Ove je slično sa dijagramom entiteta. Treća kolona je za ime kolone

Konvencija imenovanja za tabele i kolone

- Ime tabele je u množini imena entiteta
- Primer: STUDENT postaje STUDENTS



STUDENTS		
Key Type	Optionality	Column Name
pk	*	id
	*	first_name
	*	last_name
	*	str_addr
	*	city
	*	state
	*	p_code
	*	dob

- Imena kolona su ista imenima atributa osim posebnih karaktera i praznih mesta koje se zamenjuju sa underscores (_)

- Za imena kolona se češće koriste abervacije nego imena atributa. Npr, ime postaje first_name ili fname

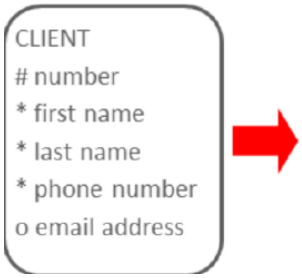
Kratka imena tabela (skraćenice)

- Jedinstvene skraćenice za svaku tabelu su korisne pri imenovanju FK kolona
- Jedan mogući način za pravljenje ovih skraćenica je bazirano na pravilima: Ako ime entiteta ima više od jedne reči, uzme se: prvi znak prve reči, prvi znak druge reči i zadnji znak poslednje reči; primer: JOB ASSIGNMENT dobija skraćeno JAT
- Skraćenice su opcione, ali korisne. Predložena pravila su neka od mogućih konvencija za određivanje skraćenica
- Ova pravila ne garantuju jedinstvenost, ali iskustvo je pokazalo da duplikacija imena je relativno retko. U slučaju istih skraćenica, samo dodati broj na jedno od njih koje se ređe koristi. Npr, CTR i CTR1

PRIVATE HOMES (PHE)

Key Type	Optionality	Column Name
pk	*	id
	*	address
	o	comments

- Za imena entiteta od jedne reči ali sa više od jednog sloga: prvi znak je sa prvog sloga, prvi znak drugog sloga i poslednji znak poslednjeg sloga; EMPLOYEE daje EPE, CLIENT daje CET



CLIENTS (CET)

Key Type	Optionality	Column Name
pk	*	client_num
	*	first_name
	*	last_name
	*	phone_num
	o	email_addr

- Za imena entiteta sa jednim slogom ali sa više od jednog znaka: prvi znak, drugi znak, poslednji znak: FLIGHT postaje FLT

THEMES (THE)

Key Type	Optionality	Column Name
pk	*	code
	*	description

Restrikcije u imenovanju od strane Oracle

- Imena tabela i kolona: moraju početi sa slovom, mogu imati do 30 alfanumerika, ne smeju imati prazna mesta ili specijalne karaktere kao “!”, ali smeju “$”, “#”, “_”
- Imena tabela moraju biti jedinstvena za jedan korisnički račun u Oracle db a imena kolona moraju biti jedinstvena unutar tabele
- Primetiti da Oracle imena tabela i kolone mogu sadržati _ . Npr SALES_ORDERS je validno ime tabele ali SALES-ORDERS nije

- Svi db sistemi prave preporuke za imenovanje objekata. Ako se ne koristi Oracle db, trebalo bi i dalje odlučiti o konvenciji imenovanja i biti siguran da je kompatabilna sa db sistemom koji je izabran
- Neke reči imaju specijalno značenje u Oracle db i u SQL programskom jeziku, oni se nazivaju rezervisane reči; treba izbegavati korišćenje njih kao imena za tabele i kolone

9-3 Mapiranje relacija

Smisao

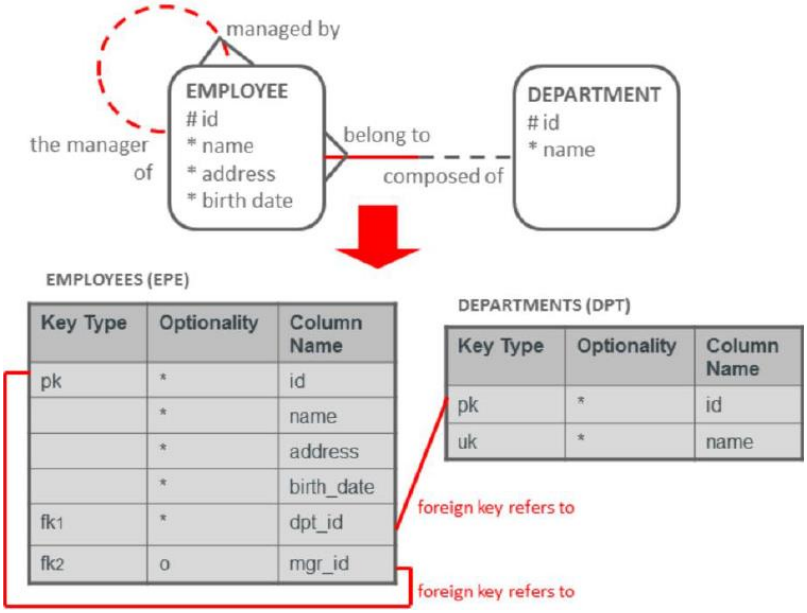
- Ako pravite kuću za nekoga, imate materijal (drvo, boju, vrata, prozore, eksere, šrafove...) i veštinu ali nemate dizajn; na početku ne znate koliko soba treba biti, gde treba staviti prozore, kako vrata treba da su orjentisana ili kojom bojom obijiti koju sobu; možete napraviti kuću na takav način i napraviti ove odluke ali ako nemate plan strukturnog dizajna i finalni produkt može biti da nije ono što mušterija hoće
- Relacije su mapirane između PK i FK da omoguće da jedna tabela referencira drugu
- Ako ne mapirate relaciju, imamo mnogo samostalnih tabela koje imaju info koje ne povezuju ništa drugo u db
- Mapiranje relacija između entiteta služi kao kritičan prvi korak koji određuje diskusiju između mušterije, dizajnera, razvoja i administracije db proizvoda

Pravila relacija

- Relacija kreira jedan ili više FK kolona u tabelama na više strani relacije
- Koristimo skraćenice tabela za imenovanje FK kolona
- Na sledećem primeru FK kolona u EMPLOYEES tabeli je dpt_id relacija sa DEPARTMENT a mgr_id je za rekurzivnu relaciju sa samom sobom
- FK kolona može biti ili obavezna ili opcionalna, u zavisnosti od potreba biznisa; na primeru dpt_id je obavezna a mgr_id je opcionalna

Pravila za ilustrovane relacije

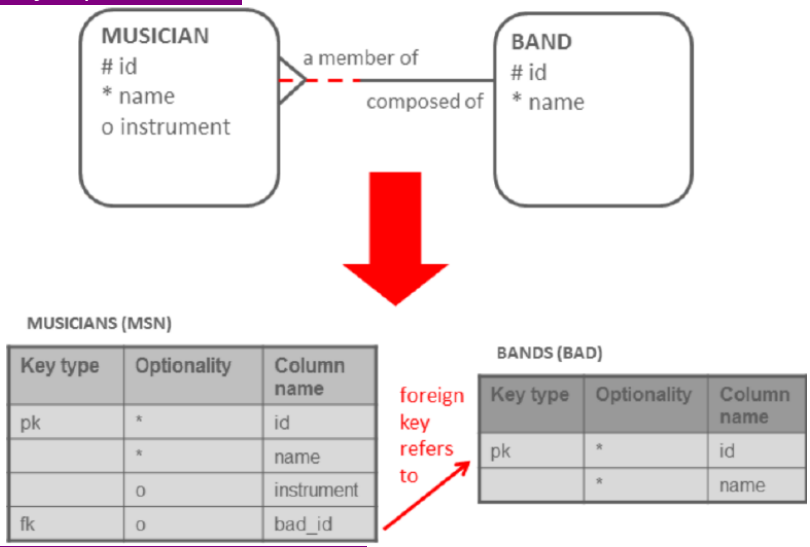
- mgr_id je opcionalna zbog „svaki EMPLOYEE može biti upravljan sa...“
- ERD odslikava relaciju između entiteta, preko biznis izraza. Kada je konceptualni model transformisan, relacije postaju FK kolone, ali ime relacije se ne prenosi. Dbd će biti osnova sistema, ali početak konceptualnog modela obezbeđuje da tabele, kolone i ograničenja kreirana u db su relevantna za biznis i ispunjavaju svoje zahteve.



Mapiranje obaveznih relacija na jednoj strani

- Relacije koje su obavezne na jednoj strani ili obavezne sa obe strane, su mapirane tačno na isti način kao i relacije koje su opcionalne na jednoj strani
- Konceptualni model je dovoljno bogat za dostizanje opcionalnosti na obe strane relacije
- Ipak, fizički model je limiotiran tako što FK ograničenja može uključiti obaveznu relaciju samo na više strani
- U sledećem primeru, fizički model ne može biti primenjen tako da BAND mora biti sačinjen od najmanje jednog MUSICIAN
- Opcionalnost na jednom kraju će morati biti implementirana preko dodatnog programiranja

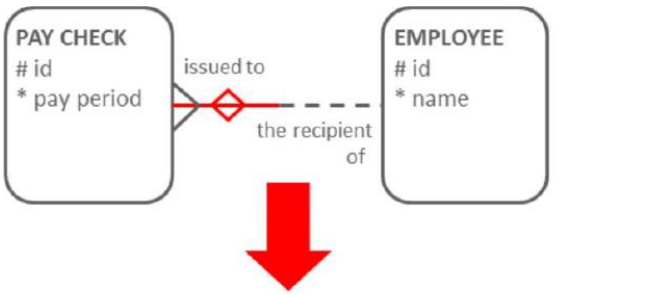
Uključivanje opcionalnosti



Mapiranje netransferabilnih relacija

- Netransferabilne relacije u konceptualnom modelu znači da FK kolona u db tabeli ne može da se updejtuje

- FK ograničenje samo po sebi ne može uključiti ovo u db; dodatno programiranje će biti neophodno da bi se osiguralo da db prati pravila biznisa
- Važno je dokumentovati ovakva pravila tako da članovi tima pišu odgovarajući kod i primenjuje pravila biznisa
- Netransferabilne relacije: informacije koje se ne mogu updejtovati
- Uključivanje (enforcing) nertansferabilnih relacija
- U primeru, paycheck ne bi mogao biti transferovan do drugog zaposlenog. Ovo znači da epe_id, što je FK kolona u tabeli PAYCHECKS, ne može biti updejtovano. Ovo će uključiti dodatno programiranje



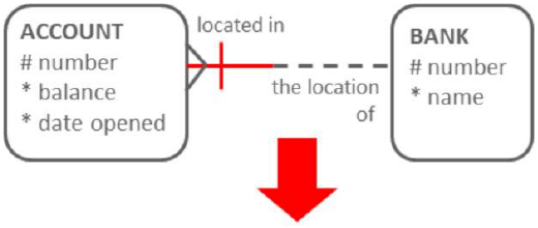
PAYCHECKS (PCK)

Key Type	Optionality	Column Name
pk	*	id
	*	pay_period
fk	*	epe_id

the value in this foreign-key column cannot be changed

Mapiranje zabranjenih relacija

- Zabranjene relacije su mapirane na FK kolonu na više strani, poput druge 1:M relacije
- U ovom slučaju, FK kolona ima duplu ulogu pošto je takođe deo primernog ključa
- U ovom slučaju, bak_number je FK kolona u ACCOUNTS koji referiše na osnovni ključ BANKS
- To je takođe deo PK od ACCOUNTS
- Act_nbr sam neće biti jedinstven unutar tabele, ali kombinacija act_nbr i bak_nbr će biti



ACCOUNTS (ACT)

Key Type	Optionality	Column Name
pk	*	act_nbr
	*	balance
	*	date_opened
pk,fk	*	bak_nbr

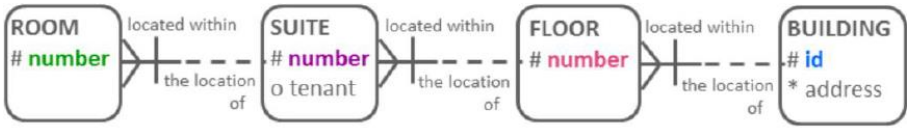
BANKS (BAK)

Key Type	Optionality	Column Name
pk	*	bank_number
	*	name

refers to

Kaskadne zabranjene relacije

- Hijerarhije mogu odvesti do kaskadne zabranjene relacije, gde UID entiteta na vrhu hijerarhije se prenosi sve do UID entiteta na dnu hijerarhije
- Npr,UID od ROOM se sastoji od ROOM broja, SUITE broja, FLOOR broja i BUILDING id
- Ovo je predstavljeno sa zabranjenim relacijama
- Kaskadna zabranjena relacija: serija relacija koje pokazuju da UID za svaki entitet u lancu se prenosi dole do entiteta na sledećem nivou
- Kada se ovo mapira na fizički model, rezultat može biti dugačko ime FK kolone pošto ona koristi skraćenice osnovne tabele kao prefiks
- Predložena konvencija je da se nikada ne koristi više od dva prefiksa tabele. U sledećem primeru, FK kolona u ROOMS koji dolaze sve od BUILDINGS je imenovan sue_bdg_id, umesto sue_flr_bdg_id.



ROOMS (ROM)

pk	*	rom_nbr
pk, fk	*	sue_nbr
pk, fk	*	sue_flr_nbr
pk, fk	*	sue_bdg_id

SUITES (SUE)

pk	*	sue_nbr
pk, fk	*	flr_nbr
pk, fk	*	flr_bdg_id
	o	tenant

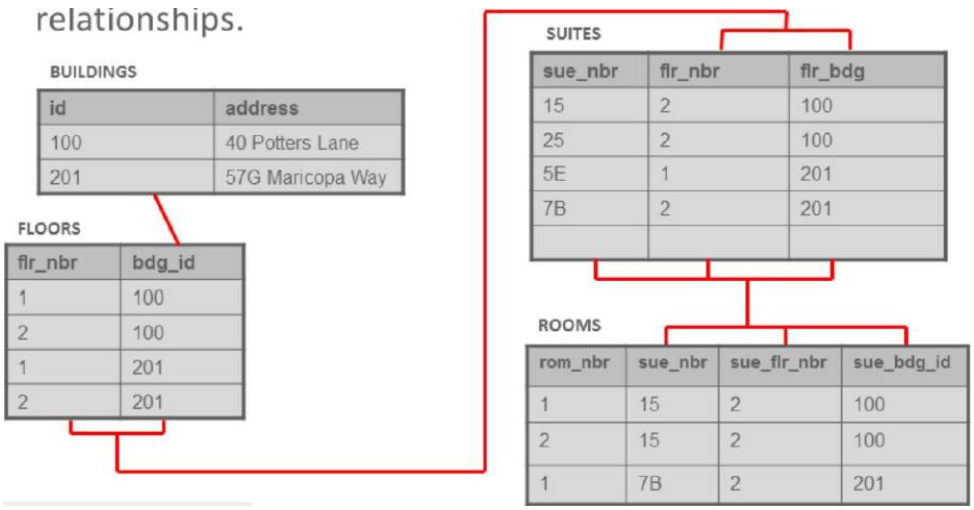
FLOORS (FLR)

pk	*	flr_nbr
pk, fk	*	bdg_id

BUILDINGS (BDG)

pk	*	id
	*	address

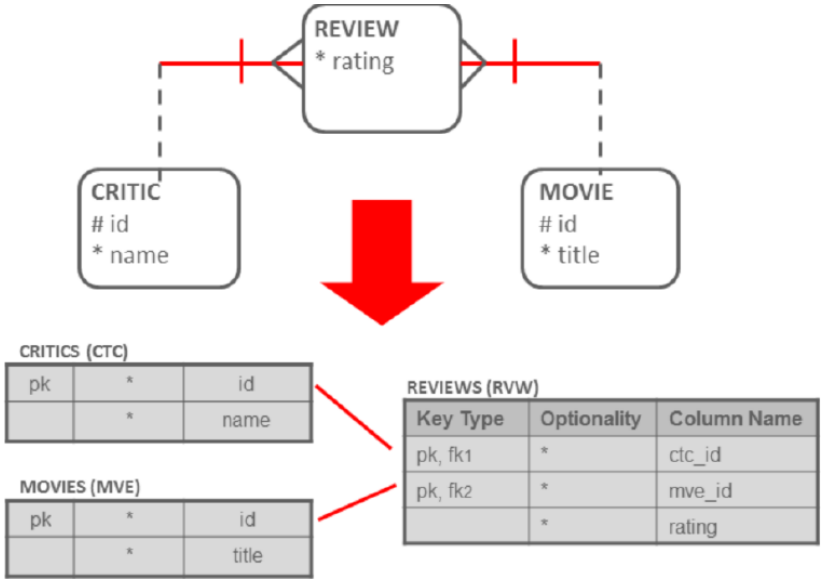
- UID i rezultujući PK su osvetljeni u drugačijim bojama da bi se pomoglo nalaženju traga UID kroz hijerarhiju
- PK od FLOORS sada postaje kompozitan od flr_nbr i bldg_id. PK od SUITES je kompozitan od sue_nbr, flr_nbr i flr_bldg_id. Videti da kompozitan je jedan PK (tabela može imati samo jedan PK).
- U ovom slučaju, takođe FK od svake tabele, čak ako je ključ kompozitan od višestrukih kolona. FK u SUITES je kombinacija flr_nbr i flr_bldg_id. FK u ROOMs je kombinacija sue_nbr, sue_flr_nbr i sue_bdg_id.
- Ilustracija kaskadnih zabranjenih relacija



- Šta je PK od ROOMS ?
- To je kombinacija rom_nbr, sue_nbr, sue_flr_nbr i sue_bdg_id. Verifikovati da kombinacija sva četiri je ono što je red jedinstvenim.

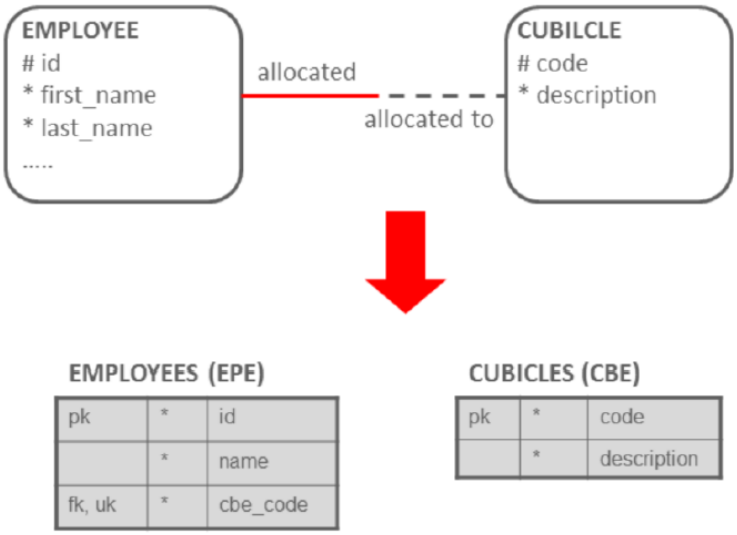
Mapiranje više prema više relacija

- M:M relacija se rešava preko intersekcionog entiteta, koji mapira na intersekcionu tabelu
- Ova intersekciona tabela će sadržati FK kolone koje se odnose na originalne tabele
- U primeru, REVIEWS sadrži sve kombinacije koje postoje između CRITIC i MOVIE



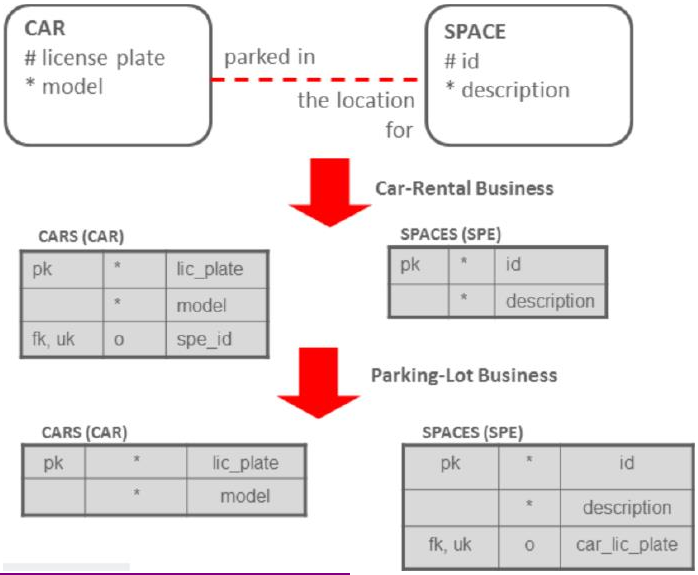
Mapiranje jedan prema jedan relacija

- Pri transformaciji 1:1 relacija, kreirate FK i jedinstveni ključ
- Sve kolone ovog FK su takođe deo jedinstvenog ključa
- Ako je relacija obavezna na jednoj strani, FK je kreirana u odgovarajućoj tabeli
- U primeru, cbe_code je FK kolona u EMPLOYEES koji se odnosi na PK od CUBICLES
- cbe_code će takođe biti jedinstveno unutar EMPLOYEES tabele
- Svaki EMPLOYEE mora biti alociran jedan i samo jedan CUBICLE; svaki CUBICLE može biti alocirana na jedan i samo jedan EMPLOYEE.



Opcionalna jedan prema jedan

- Ako je relacija opcionalna na obe strane, možeš izabrati koja tabela uzima FK
- Ne postoje apsolutna pravila, ali ovo su neka uputstva:
    1. implementiraj FK u tabeli sa što manje redova da bi se sačuvao prostor
    2. implementiraj FK gde se to čini razumnije za posao
- U primeru, car-rental agencija će biti više zainteresovana o kolima nego prostoru, tako da je razumno staviti FK u CARS
- Ipak, u biznisu parkirališta, glavni objekat je mesto za parkiranje
- Zato, ima smisla staviti FK u SPACES



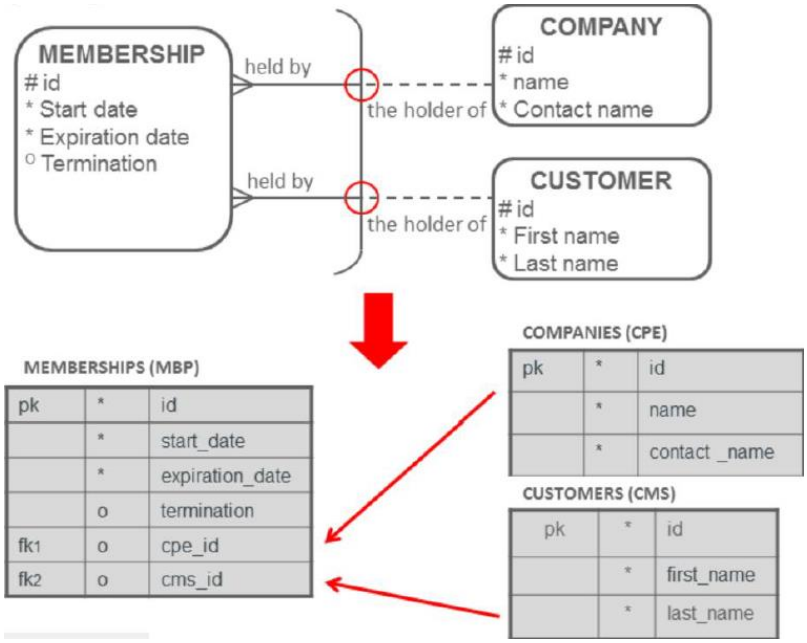
Uvođenje (enforcing) jedan prema više

- Ako relacija je obavezna na oba kraja, imate istu limitaciju u db kao 1:M relacija koja je obavezna na jednom kraju
- Zato, treba vam ispisati dodatni kod za njeno uvođenje
- Obavezna 1:1 relacija je retka. Najčešće ovo će biti modelovano kao jedna tabela, bez potrebe za 1:1 relacijom

Mapiranje Arcs

- Entitet koji ima arcs će mapirati u tabelu koja sadrži FK od tabele sa kraja jedan relacije

- Primetiti da čak ako relacija u arc su obavezne na više strani, rezultujući FK mora biti opcionalan, pošto jedan od njih će uvek biti prazan



- Pošto je luk predstvaljen ekskluzivnom relacijom, dodatni kod je potreban za uključenje da samo jedan FK ima vrednost za svaki red u tabeli
- Check constraint smešten u db može lako to da uradi
- U primeru, kod za proveru ograničenja će izgledati ovako: CHECK (pse_id nije null i phe_id je null), ili (pse_id je null i phe_id nije null)
- Ako je relacija potpuno opciona, dodaće se: ili (pse_id je null i phe_id je null)
- Provera ograničenja je programiranje koda koji može biti smešten u db. On može ugraditi jednostavna pravila koja se primenjuju na jednu vrstu u tabeli (kao što je upoređivanje vrednosti ili obezbeđujući da su null ili nisu null). Ovo je slučaj sa arc.
- U slučaju obaveznog jednog kraja od 1:M ili 1:1 relacije, moramo proveriti da li je uneti red u jednoj tabeli (master), red mora takođe biti unet u drugoj tabeli (tabela dete ili detalj). Provera ograničenja ne može span dve tabele ili različite vrste u istoj tabeli. Ne može sprečiti unos, update ili brisanje operacija. Ovo je zato što dodatno programiranje (umesto provere ograničenja) je neophodno.

9-4 Mapiranje podtipova

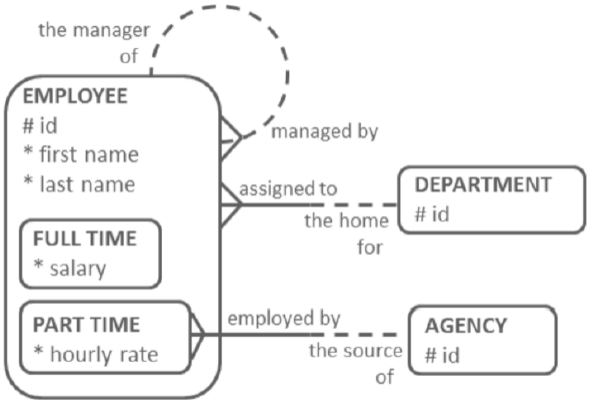
Smisao

- Drvodelja koji pravi kuću može znati da ćeš koristiti različite tipove sijalica po kući; ipak, ako ne omogućiš informacije gde određeni tipovi sijalica treba da se postave, možeš imati problema; mapiranje supertipova i subtipova osigurava da prava informacija se smešta sa svakim tipom

Supertip implementacija: jedna tabela

- Ovaj izbor proizvodi jednu tabelu za implementaciju supertip entiteta i njegovih subtipova; takođe se naziva “single table (ili one-table) implementation“
- Pravila:
    1. tabele – samo se jedna tabela pravi, bezobzira na broj subtipova;

2. kolone – jedna tabela dobija jednu kolonu za svaki atribut supertipa, zajedno sa originalnim opcionalnostima atributa
3. tabela takođe dobija kolonu za svaki atribut koji pripada subtipu, ali kolone sve postaju opcionalne
4. dodatno, obavezna kolona treb ada je kreirana da funkcioniše kao diskriminator kolona za razlikovanje između različitih subtipova entiteta
5. vrednost koju može imati iz seta svih subtipova skraćenica (FTE, PTE, OTR u primeru)
6. ova diskriminator kolona se obično naziva <table_short_name>_type, što bi bilo epe_type u primeru



DEPARTMENTS (DPT)		
pk	*	id

AGENCIES (AGY)		
pk	*	id

EMPLOYEES (EPE)		
Key Type	**Optionality**	**Column Name**
pk	*	id
	*	first_name
	*	last_name
	o	salary
	o	hourly_rate
fk1	*	dpt_id
fk2	o	agy_id
	*	epe_type
fk3	o	mgr_id

- Plata i hourly_rate su obe opcionalne, čak i ako ERD ima ih kao obavezne. Dpt_id može ostati obavezna, pošto dolazi iz relacije sa supertipa EMPLOYEE. Agy_id dolazi iz relacije sa subtipa i mora biti opcionalna
- Epe_type je diskriminator kolona. Može imati vrednosti FTE ili PTE.
- Primetiti da mgr_id prikazuje rekurzivnu relaciju za EMPLOYEE. To je FK kolona i normalno bi bila nazvana epe_id, po tabeli roditelja. Ipak, razumno je preimenovati je u ovoj fazi da bi se lakše razumela.
- Pravila:
    7. Identifikatori: UID se transformišu u osnovne i jedinstvene ključeve
    8. Relacije: relacije na nivou supertipa se transformišu kao i obično. Relacije na subtip nivou su implementirane kao opcione FK kolone
    9. Integitet ograničenja: provera ograničenja je potrebna za obezbeđivanje da za svaki određeni subtip, sve kolone koje dolaze iz obaveznih atributa nisu null
- U konceptualnom modelu, plata je obavezna za full-time zaposlenog a hourly rate je obavezna za part-time zaposlenog
- Kada EMPLOYEE supertip se implementira kao jedna tabela u fizičkom modelu, ovi atributi postaju opcioni
- Provera ograničenja je potrebna za ugradnju pravilla biznisa u ERD
- Crtica o OTHER subtipovima: OTHER subtipovi su savetovani u konceptualnom modelu da bi obezbedili da subtipovi su exhaustive. Ipak, dok ne počnemo fazu

dizajna, trebalo bi da smo uradili extensive analizu za određivanje ako je još neki subtip zaista potreban. Ako jeste, onda ovaj subtip mora biti imenovan, a njegovi atributi specificirani. Ako ne, onda OTHER subtip nije deo procesa transformacije

- U ovom primeru kod za proveru ograničenja će izgledati ovako: CHECK (epe_type = 'FTE' i salary nije null i hourly_rate je null i agy_id je null); ili (epe_type = 'PTE' i salary je null i hourly_rate nije null i agy_id nije null)
- kod proverava ako je full-time employee (epe_type = 'FTE') onda vrednost mora postojati u koloni plate i hourly_rate i agy_id kolona mora biti prazna
- Ako je to part-time zaposleni (epe_type = 'PTE'), onda vrednost mora postojati u hourly_rate i agy_id ali salary mora biti prazno

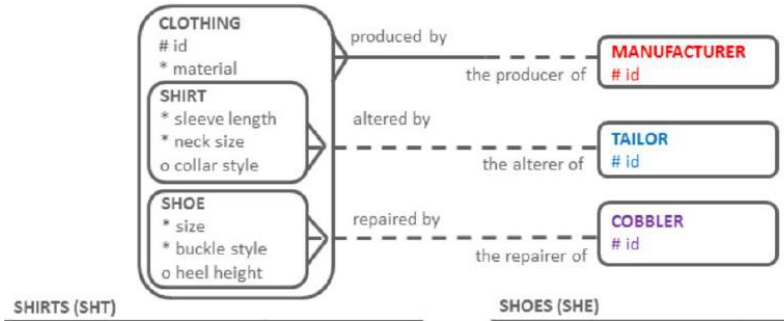
Sample Data for EMPLOYEES

id	first_name	last_name	salary	hourly_rate	dpt_id	agy_id	epe_type	epe_id
2000	Joan	Merrick	50000		10		FTE	111
111	Sylvia	Patakis	90000		10		FTE	
2101	Marcus	Rivera		65.00	10	17	PTE	111
2102	Hector	Chen		75.00	25	17	PTE	45
45	Rajesh	Vishwan	90000		25		FTE	

Zašto bi izabrao jedna tabela/supertip implementaciju

- Ovo je fleksibilna i uobičajena implementacija.
- Ona je najčešće prva koja se istražuje i posebno je dobra gde:
  - najviše atributa su supertip nivoa
  - najviše relacija su supertip nivoa
  - biznis pravila su globalno ista za subtipove
- Obično se kreira pogled (view) za svaki podtip, pokazujući samo kolone koje pripadaju nekom određenom subtipu. Tačne vrste su izabrane korišćenjem uslova baziranim na diskriminator koloni
- Subtip implementacija: dve tabele
- Ovo se takođe zove i "implementacija dve tabele"; kreira se tabela za svaki subtip; pa u realnosti možeš imati više od dve tabele ako imaš više od dva subtipa
- Subtip: nešto na šta se deli entitet bazirano na uobičajenim atributima i/ili relacijama
- Pravila:
  1. tabele – jedna tabela po subtipu prvog nivoa
  2. kolone – svaka tabela dobija jednu kolonu za svaki atribut supertipa zajedno sa njenim originalnim opcionalnostima
  3. svaka tabela takođe dobija jednu kolonu od svakog atributa koji pripada subtipu zajedno sa njegovim originalnim opcionalnostima
  4. identifikatori – primarni UID na nivou supertipa kreira PK za svaku tabelu. Sekundarni UIDs supertipa postaju jedinstveni ključevi u svakoj tabeli
  5. relacije – sve tabele dobijaju FK za relacije na nivou supertipa, sa originalnim opcionalnostima; za relacije na subtip nivou, FK je implementiran u tabeli na koju je mapiran; originalna opcionalnost se zadržava

- Id, materijal i mnr_id dolaze iz atributa u OR relacijama na supertip. Zato se pojavljuju u obe tabele



SHIRTS (SHT)

Key Type	Optionality	Column Name
pk	*	id
	*	material
	*	sleeve_length
	*	neck_size
	o	collar_style
fk1	o	tlr_id
fk2	*	mnr_id

refers to manufacturers refers to tailors

SHOES (SHE)

Key Type	Optionality	Column Name
pk	*	id
	*	material
	*	size
	*	buckle_style
	o	heel_height
fk1	o	clr_id
fk2	*	mnr_id

refers to manufacturers refers to cobblers

- U ovom primeru, odvojene tabele će biti kreirane za SHIRTS i SHOES

Sample Data for SHIRTS

id	material	sleeve_length	neck_size	collar_style	mnr_id	tlr_id
10	linen	33	16	button down	65	14
11	wool	32	15.5	nehru	65	22
14	cotton	33	15.5		60	22

Sample Data for SHOES

id	material	size	buckle_style	heel_height	mnr_id	clr_id
3	leather	7.5	monkstrap	1.5	75	44
7	canvas	8	velcro	1	70	44

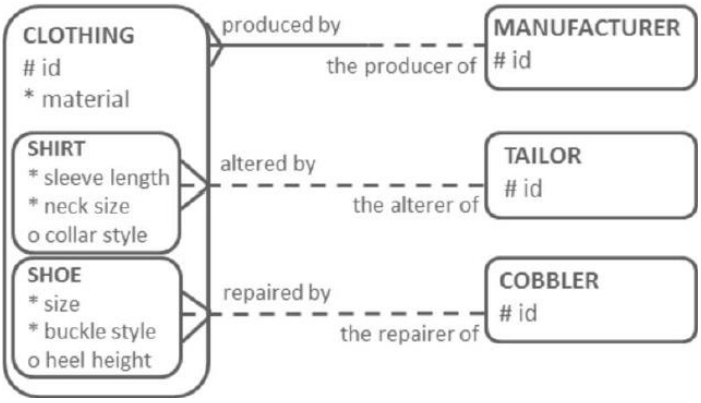
- Obično, kreiraćete dodatni pogled koji predstavlja supertip, pokazujući sve kolone supertipa i različite subtipove

Kada se uzima u obzir subtip implementacija

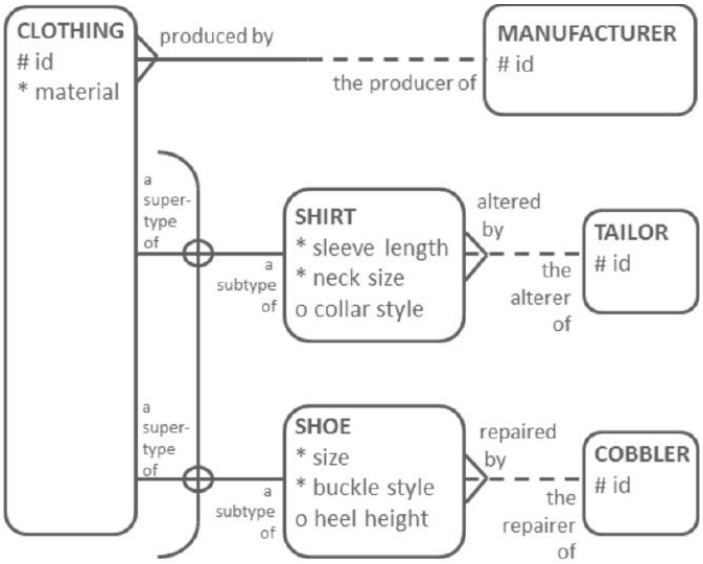
- Subtip implementacija se može koristiti kada: subtipovi imaju veoma malo zajedničkog (postoji nekoliko atributa na nivou supertipa i nekoliko na subtip nivou)
- Najviše relacija su na subtip nivou
- Pravuila buiznisa i funkcionalnosti su dosta različita između subtipova
- Kako se tabele koriste je različito – npr, jedna tabela je pod upitom dok se druga updejtuje

Modelovanje supertipa sa arc

- Supertip entitet i njegovi subtipovi mogu biti modelovani kao arc relacije
- Ovde ponovo je originalni ERD sa supertipom i subtipovima



- Ilustracija modela jednog arc
- Na ovom Erd imamo ponovo nacrtanu CLOTHING supertip i njegove subtipove SHIRT i SHOE kao zasebne entitete gde svaki zasebno ima 1:1 relacije sa supertipom, relacije su u arc

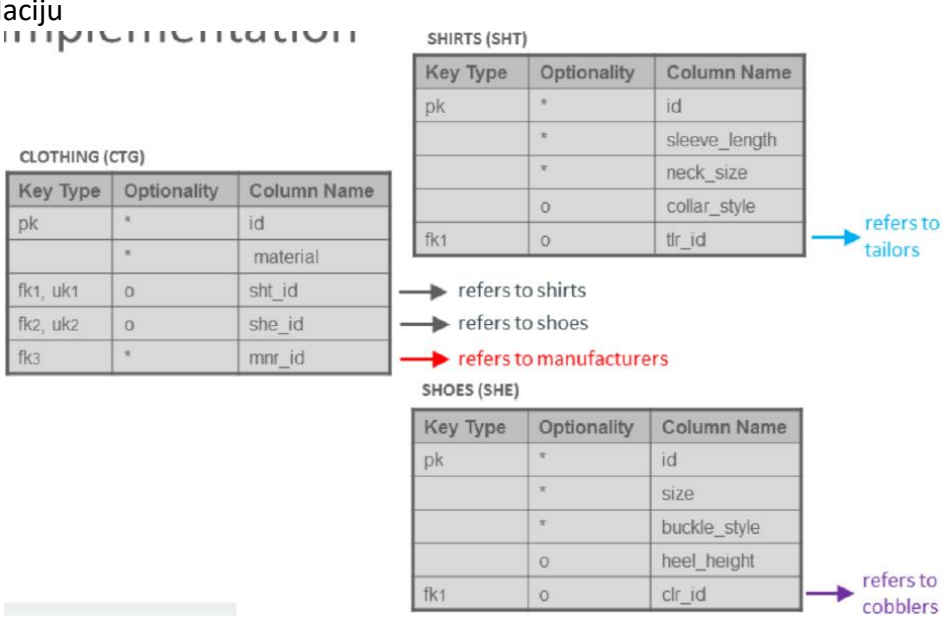


- Relacije 1:1 su obavezne i ekskluzivne: svaki SHIRT mora biti jedno parče CLOTHING, svaki SHOE mora biti jedno parče CLOTHING. To znači, svako parče CLOTHING mora biti ili SHOE ili SHIRT ali ne istovremeno

Supertip i subtip (arc) implementacija

- Ovaj izbor daje jednu tabelu za svaki entitet
- Supertip tabela ima FK za svaku supertip tabelu
- Ove FK predstavljaju ekskluzivne relacije
- Oni su opcionalne pošto samo jedna od njih može imati vrednost u svakom redu u tabeli
- Pravila:
    1. tabele – kreira se onoliko tabela koliko ima subtipova, takođe jedna za supertip
    2. kolone – svaka tabela ima kolonu za sve atribute entiteta na kojem je bazirana, sa originalnom opcionalnosti
- identifikatori: primarni UID supertip nivoa kreira PK za svaku od tabela; svi ostali UID postaju jedinstveni ključevi u njihovim odgovarajućim tabelama
- relacije: sve tabele dobijaju FK za relevantnu relaciju na nivou entiteta, sa originalnom opcionalnosti

- integitet ograničenje: dve dodatne kolone se kreiraju u tabeli baziranoj na supertipu
- To su FK kolone koje se odnose na tabele koje implementiraju subtipove
- Kolone su opcionalne pošto FK su u arc
- Dodatni check constraint je potreban za implementaciju arc
- FK kolona su takođže jedinstveni ključevi pošto oni implementiraju obaveznu 1:1 relaciju



- U ovom slučaju, nije verovatno da arc implementacija će biti izabrana, nego je prikazana samo kao demnstracija da se ova mogućnost može razmotriti
- Trbalo bi check constraint na CLOTHING za uključivanje da (sht_id nije null i she_id jeste null) ili (sht_id jeste null i she_id nije null)

Kada uzeti u obzir i supertip i subtip (arc) implementaciju

- Kada subtipovi imaju veoma malo zajedničkog i svaka tabela predstavlja informacije koje se mogu koristiti nezavisno
- npr, kada CLOTHING tabele daju sve globalne informacije, a obe SHOES i SHIRTS daju specifične informacije, a kmbinacija globalnih i specifičnih informacija retko se koristi
- biznis pravila i funkcionalnost su dosta različiti između svih tipova
- kako se tabele koriste je različito