Fakulta elektrotechniky a informatiky
Slovenská technická univerzita v Bratislave

Databázové systémy

Vladislav Novák

pracovná verzia (aktualizácia 25. 3. 2021)

Obsah

0 Úvod

Databáza je súbor súvisiacich údajov. Počítačový softvér, ktorý umožňuje vytváranie, používanie a údržbu databázy sa nazýva **systém pre správu databázy** (angl.: database management system). Budeme preň používať skratku odvodenú z anglického názvu **DBMS**.

1 Dátový model

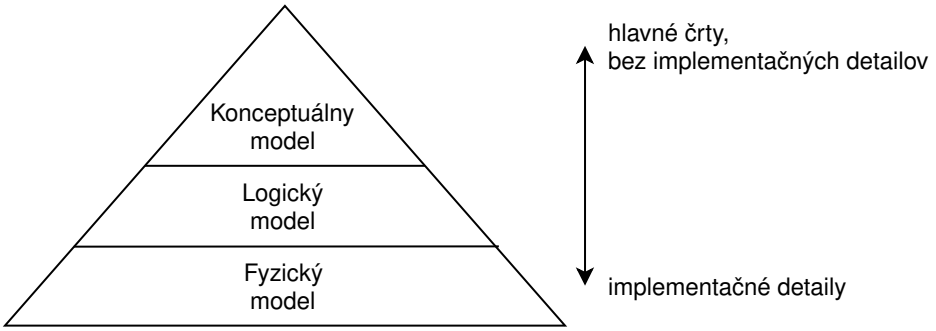
Dátový model reprezentuje:
štruktúru údajov, podmienky/pravidlá (angl.: constraints) pre hodnoty a ich vzájomné vzťahy, operácie nad údajmi.

Dátový model definuje schému databázy (jej štruktúru a podmienky ktoré musia byť splnené). Stav databázy reprezentuje údaje v nej uložené v určitom časovom okamihu. TODO

1.1 Hierarchia modelov podľa úrovne abstrakcie

Pri návrhu databázy, najprv v jej modely zachytávame hlavné črty a postupne do modelu pridávame rôzne detaily. Modelovanie môžeme rozdeliť do troch hlavných úrovní:

- konceptuálny model,
- logický model,
- fyzický model.



Obrázok 1.1: Hierarchia modelov

Najabstraktnejším je konceptuálny model. Zachytáva hlavné črty a je nezávislý na spôsobe implementácie. Po jeho vytvorení máme prehľad o navrhovanom systéme, na základe ktorého môžeme vybrať vhodný spôsob implementácie. Konceptuálny model dovoľuje zachytiť požiadavky na databázu bez toho, aby bol analytik alebo návrhár obmedzovaný konkrétnym typom implementácie. To v prvých fázach umožní lepšie sústredenie sa na ciele projektu a nebyť usmerňovaný (obmedzovaný) spôsobom práce určitého typu databázy. Ak by sme od začiatku modelovali databázový systém napríklad ako relačnú databázu, tak by sme boli nútený do spôsobu riešenia, ktorý je daný spôsobom práce relačnej databázy.

Logický model už zohľadňuje typ databázy, v ktorom budú údaje uložené. Typ databázy určuje spôsob uchovávania a práce s údajmi. Napríklad, či sú údaje reprezentované ako súbor tabuliek, súbor objektov, graf, atď. Typy databázy sú napríklad:

- relačná databáza,

- objektovo orientovaná databáza,
- objektovo relačná databáza,
- grafová databáza,
- dokumentová databáza,
- stĺpcová databáza.

Fyzický model zahŕňa implementačné detaily, ktoré sú dané konkrétnou implementáciou DBMS. Pre relačné databázy existujú viaceré implementácie (napr. PostgreSQL, MySQL, SQLite). Ich spoločnou vlastnosťou je napríklad ukladanie údajov ako súbor tabuliek. Líšia sa ale rôznymi detailami, ako napríklad poskytovanými dátovými typmi pre údaje, atď. Podľa fyzického modelu môže programátor implementovať navrhnutú databázu.

Aby bol model prehľadný, používa sa grafický zápis. Tento zápis ale neumožňuje zachytiť všetky potrebné informácie, preto je ho potrebné doplniť textovým popisom.

2 Konceptuálny model

2.1 Entitno-relačný model

Na definíciu konceptuálneho modelu sa najčastejšie používa **entitno-relačný model** (anglicky: Entity-Relationship Model) skrátene: ER model alebo ERM. Tento typ modelu sa graficky reprezentuje pomocou **entitno-relačný diagramu** (anglicky: Entity-Relationship Diagram), skrátene: ER diagram alebo ERD). Zápis ER diagramu nie je štandardizovaný. Pre tento diagram existuje množstvo spôsobov zápisu, ktoré sa líšia nie len v grafickej reprezentácii, ale niekedy aj v sémantike (násobnosť v n-árnom vzťahu (časť 2.1.7)). V tomto učebnom texte použijeme grafickú reprezentáciu ER diagramu, ktorá má veľa spoločných čŕt s Chenovym spôsobom grafickej reprezentácie. V inej literatúre sa môžete stretnúť s inými spôsobmi grafickej reprezentácie. Prehľad rôznych reprezentácii môžete nájsť napríklad v [1].

ER diagram je graf reprezentujúci entity, vzťahy a ich atribúty. V tomto učebnom texte budú všetky tieto prvky tvoriť uzly grafu. Použijeme len ich neformálne definície.

Ďalšími alternatívami pre zápis konceptuálneho modelu je UML (Unified Modeling Language)[1], alebo ORM (Object-Role Modeling).

2.1.1 Entita

Prvým krokom pri návrhu ER modelu je identifikácia entít a vzťahov medzi nimi. **Entita** (angl.: Entity) reprezentuje objekt, udalosť alebo pojem, o ktorom potrebujeme evidovať informácie. Entita môže reprezentovať fyzický objekt (napr. zamestnanca, budovu), alebo nefyzický objekt (napr. pracovná pozícia, schôdza).

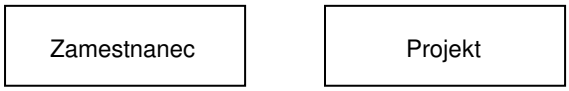
V rámci používania informačného systému, informácie ktoré o entite evidujeme, musia v reálnom svete jednoznačne identifikovať objekt, ktorý entita reprezentuje.

Entity môžeme zoskupiť podľa typov objektov z reálneho sveta, ktorý reprezentujú. Napríklad niektoré entity reprezentujú zamestnancov, iné reprezentujú miestnosti v budove, atď. **Typ entity** (angl.: entity type) definuje množinu všetkých entít, reprezentujúcich rovnaký typ objektov z reálneho sveta. Z toho vyplýva, že údaje, ktoré o entitách toho istého typu evidujeme, majú rovnakú štruktúru. Čiže typ entity určuje aj štruktúru údajov, ktoré o entitách daného typu evidujeme. Preto typ entity definuje atribúty, ktoré sú rovnaké pre všetky entity daného typu (časť 2.1.4), ale líšia sa ich hodnotami. Každá entita je zaradená práve do jedného typu.

Názov typu entity musí byť jedinečný. Zvyčajne sa volí podstatné meno v jednotnom čísle.

V ER diagrame reprezentujeme typ entity obdĺžnikom, obsahujúcim názov typu entity. Ak v databáze informačného systému firmy chceme evidovať informácie o zamestnancoch a projektoch, tak ER diagram bude obsahovať dva typy entít s názvami `Zamestnanec` a `Projekt` (obr. č. 2.1).

1 UML bol pôvodne vytvorený pre popis objektovo-orientovaného návrhu, ale môže byť použitý aj pre popis iného typu návrhu.



Obrázok 2.1: Entity reprezentujúce zamestnancov a projekty

Entita reprezentuje jeden objekt z reálneho sveta, ale pre stručnosť komunikácie sa niekedy tento pojem používa pre označenie celej množiny entít rovnakého typu. Tento nepresný spôsob vyjadrovania bude občas použitý aj v tomto texte, na miestach kde by nemal spôsobiť nesprávne porozumenie. Preto pre zvýraznenie zámeru označenia jednej entity môžeme použiť (teoreticky nadbytočný) pojem **inštancia** typu entity (angl.: entity instance), ktorý má rovnaký význam ako pojem „entita“. Napríklad `Zamestnanec` je názov typu entity, konkrétni zamestnanci Jano a Peter sú entity typu `Zamestnanec`, resp. inštancie typu `Zamestnanec`.

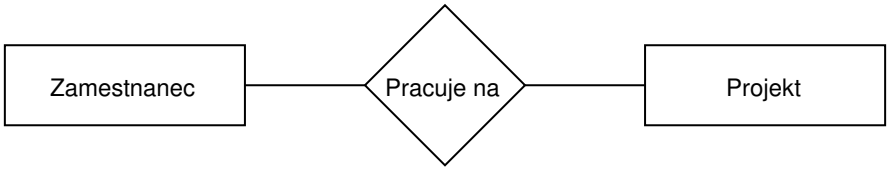
2.1.2 Vzťah

Vzťah (anglicky: relationship) reprezentuje asociáciu medzi entitami. Matematicky ju môžeme definivať ako n-ticu entít, medzi ktorými je asociácia.

Typ vzťahu (angl.: relationship type) definuje spoločné charakteristiky pre skupinu vzťahov. Spoločnou charakteristikou je napríklad skupina typov entít, medzi ktorými môžu vznikať vzťahy daného typu. Ďalšou spoločnou charakteristikou sú typy údajov, ktoré o vzťahoch daného typu zaznamenávame. Každý vzťah je zaradený do práve jedného typu. Názov typu vzťahu je zvyčajne tvorený slovesom (ak existuje vhodné sloveso).

Podobne ako pri entitách, sa v komunikácii často namiesto pojmu „typ vzťahu“, pre stručnosť nesprávne použije iba slovo „vzťah“. Analogicky, pre zvýraznenie, že ide o jeden vzťah medzi inštanciami entít, môžeme použiť pojem **inštancia** typu vzťahu, ktorý má rovnaký význam ako pojem „vzťah“ (napríklad inštancia typu `Pracuje na` (obr. č. 2.2)).

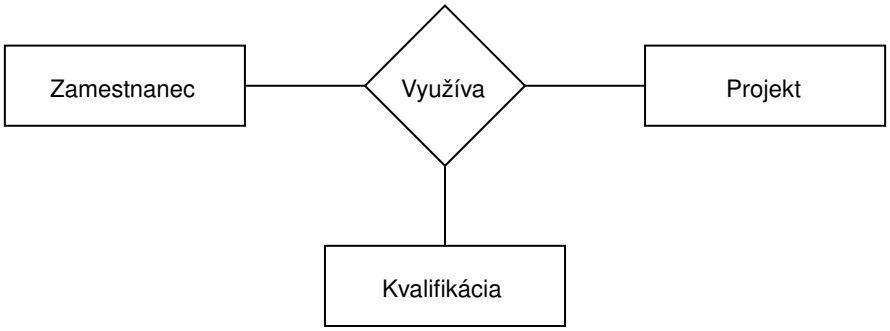
V ER diagrame označujeme typ vzťahu kosoštvorcom (prípadne kosodĺžnikom roztiahnutým do šírky). Kosoštvorec obsahuje názov typu vzťahu. Je spojený čiarami s typmi entít, ktorých inštancie môžu byť vo vzťahu. Na obrázku č. 2.2 je zobrazený typ vzťahu reprezentujúci prácu zamestnanca na projekte. Každá inštancia typu `Pracuje na` je asociáciou jedného zamestnanca a jedného projektu. Ak napríklad jeden zamestnanec pracuje na dvoch projektoch, tak databáza obsahuje dve inštancie vzťahu, v ktorých je tento zamestnanec, ale dva rôzne projekty. Ak zamestnanec nepracuje na žiadnom projekte, tak v databáze neexistuje inštancia tohto typu vzťahu, v ktorej by bol tento zamestnanec. Nie je určené, či každý zamestnanec musí pracovať aspoň na jednom projekte, ani či jeden zamestnanec môže pracovať na viacerých projektoch, takže aj tieto prípady sú možné. Neskôr si ukážeme aj možnosti určenia takýchto pravidiel.



Obrázok 2.2: Vzťah určujúci priradenia zamestnanca na projekt

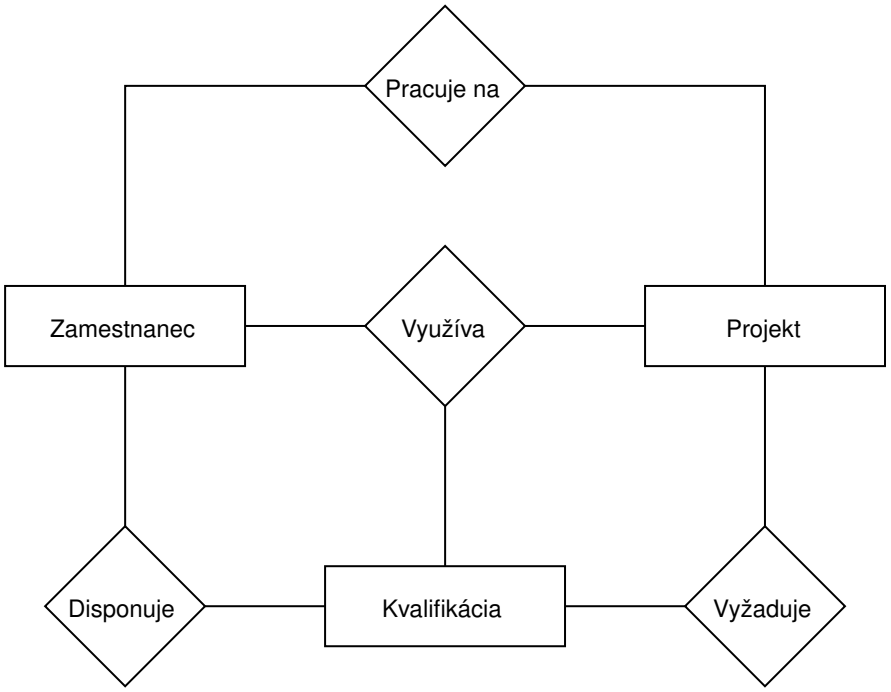
Názov typu vzťahu je vhodné voliť tak, aby sa podľa diagramu dala ľahko zostaviť veta čítaním zľava doprava alebo zhora nadol. Pri čítaní diagramu na obr. č. 2.12 zľava doprava, môžeme vytvoriť vetu „zamestnanec je členom tímu“, pri čítaní sprava doľava vetu „tím je zložený zo zamestnancov“. Čiže typ vzťahu môžeme napríklad nazvať `Je členom` alebo `Je zložený`. Z týchto dvoch možností sme vybrali tú, ktorá je vhodnejšia pri čítaní zľava doprava.

Najčastejšie sa vyskytujú vzťahy medzi dvoma entitami rôznych typov. Vtedy príslušný typ vzťahu spája dva typy entít. Ale typ vzťahu môže definovať asociáciu medzi entitami jedného, dvoch, troch, alebo viacerých typov. Príklad typu vzťahu medzi dvoma typmi entít je na obrázku č. 2.2. Na obrázku č. 2.3 je znázornený typ vzťahu medzi troma typmi entít. Každá inštancia vzťahu nesie údaje o tom, ktorý zamestnanec využíva ktoré kvalifikácie na ktorom projekte. Typ vzťahu nereprezentuje zaradenie zamestnanca na projekt, ani zoznam kvalifikácii zamestnancov. Tieto informácie musia byť reprezentované ďalšími typmi vzťahov. Diagram reprezentujúci aj tieto typy vzťahov je na obrázku č. 2.4. Vo všeobecnosti platí, že vzťahy 3. a vyššieho stupňa, nie je možné nahradiť vzťahmi druhého stupňa[2]. Ak napríklad zamestnanec disponuje určitou kvalifikáciou a projekt túto kvalifikáciu vyžaduje, neznamená to automaticky, že zamestnanec túto kvalifikáciu na projekte využíva (ani ak na projekte pracuje).



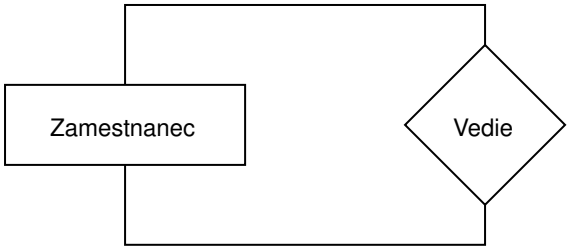
Obrázok 2.3: Zamestnanec na projekte využíva kvalifikáciu

2 Je to možné len v špeciálnych prípadoch.



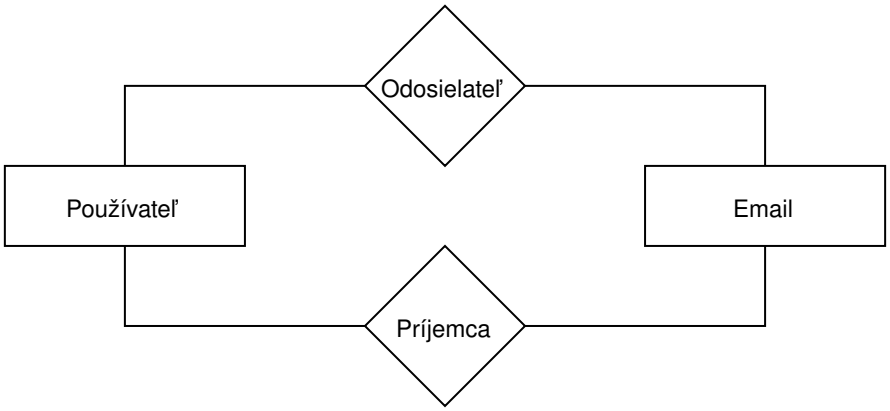
Obrázok 2.4: Typy vzťahov reprezentujúce zaradenie zamestnancov na projekty, zoznam kvalifikácii zamestnancov a skutočné využitie kvalifikácii zamestnancov na projektoch

Entity jedného typu sa môžu vzťahu zúčastniť viac krát. Na obrázku č. 2.5 je typ vzťahu reprezentujúci hierarchiu medzi zamestnancami. Každá inštancia vzťahu asociuje dvoch zamestnancov (v roliach nadriadeného a podriadeného).



Obrázok 2.5: Typ vzťahu všeobecne reprezentujúci hierarchiu zamestnancov

Medzi dvoma typmi entít môže existovať viacero typov vzťahov. Napríklad v databáze emailového servera, môže medzi typmi entity `Používateľ` a `Email` existovať typ vzťahu `Odosielateľ` aj `Príjemca` (obr. č. 2.6).



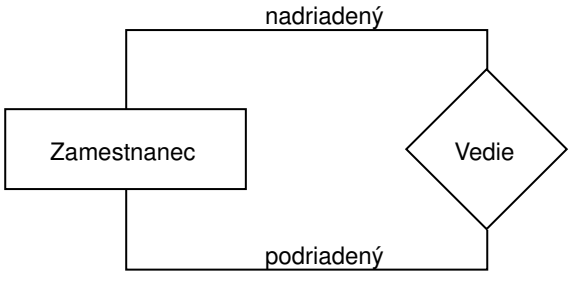
Obrázok 2.6: Dva typy vzťahov medzi tými istými typmi entít

Stupeň typu vzťahu (angl.: degree of a relationship type) je počet typov entít, zúčastňujúcich sa vo vzťahu. Typ vzťahu `Vedie` (obr. č. 2.5) má stupeň 1 (**unárny** typ vzťahu), typ vzťahu `Pracuje na` (obr. č. 2.2) má stupeň 2 (**binárny** typ vzťahu), typ vzťahu `Využíva` (obr. č. 2.4) má stupeň 3 (**ternárny** typ vzťahu). V ER modely môžu existovať aj vzťahy vyššieho stupňa.

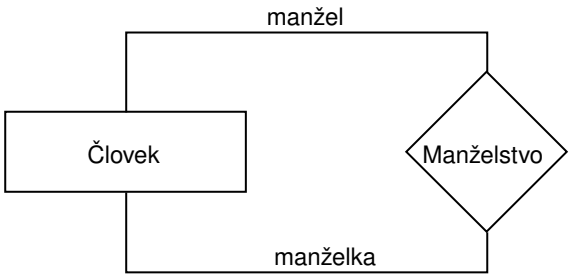
Vzťah `Vedie` na obr.č. 2.5 sa nazýva aj **rekurzívny typ vzťahu**, pretože jeden typ entity sa zúčastňuje v type vzťahu viac krát, v rôznych roliach (časť 2.1.3).

2.1.3 Rola vo vzťahu

Rola (angl.: role) pomáha presnejšie špecifikovať význam entity vo vzťahu. V ER diagrame zapisujeme jej názov ku čiare medzi typom vzťahu a typom entity. Určenie role nie je vždy potrebné. Ak je rola entity vo vzťahu jasná, nie je potrebné rolu explicitne definovať a ako názov role môže byť často použitý názov typu entity. Určenie rolí je najviac potrebné, ak sa jeden typ entity zúčastňuje vzťahu viac krát (vzťahu sa zúčastňuje viacero entít toho istého typu v rôznych roliach). Na obrázku č. 2.7 je znázornenie rolí zamestnancov vo vzťahu `Vedie`, ktorý reprezentuje hierarchiu medzi zamestnancami. Názvy rolí ozrejmujú význam účasti typu entity `Zamestnanec`, ktorého inštancie vystupujú v roliach nadriadeného a podriadeného. Na obrázku č. 2.8 je diagram reprezentujúci dvoch ľudí vo vzťahu `Manželstvo`. Jeden človek sa zúčastňuje manželstva v roli manžela, druhý v roli manželky.

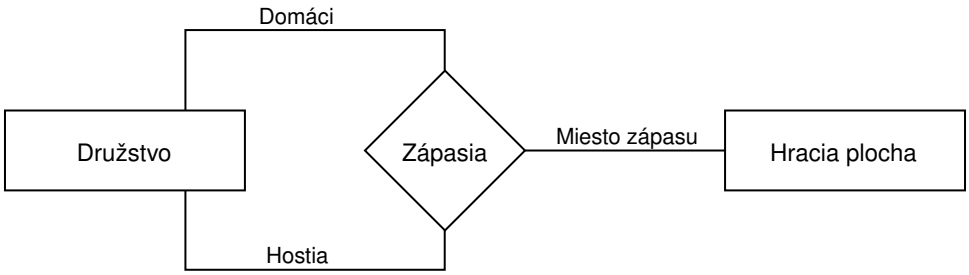


Obrázok 2.7: Role nadriadeného a podriadeného v hierarchii zamestnancov



Obrázok 2.8: Role manžela a manželky v manželstve

Na obrázku č. 2.9 je príklad rolí v type vzťahu, ktorý reprezentuje hranie zápasu medzi domácim a hosťujúcim družstvom. Predpokladáme, že športový klub (nie je v diagrame) do ktorého družstvo patrí, môže vlastniť viaceré hracie plochy. Preto vzťah zahŕňa aj hraciu plochu. Ak by športový klub domáceho družstva vlastnil vždy iba jednu hraciu plochu, potom by entita `Hracia plocha` nemusela byť vo vzťahu, pretože miesto zápasu by bolo určené domácim družstvom, ktoré by v ER modely malo vzťah so športovým klubom, ktorý by mal vzťah s hracou plochou, ktorú vlastní. To že jedno družstvo je vždy domáce, je umelý predpoklad, ktorý sme si dali z dôvodu uvedenia príkladu explicitného definovania rolí vo vzťahu, kde vystupujú tri entity, dvoch typov.



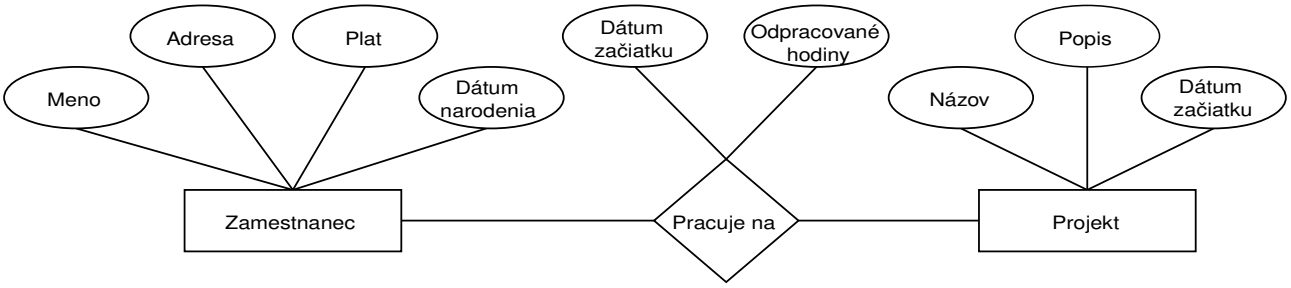
Obrázok 2.9: Role vo vzťahu reprezentujúcom hranie zápasu družstiev. Predpokladáme, že športový klub do ktorého družstvo patrí, môže vlastniť viacero hracích plôch.

2.1.4 Atribút

Atribút (anglicky: attribute) reprezentuje vlastnosť entity alebo vzťahu. Je to funkcia, ktorá každej entite alebo vzťahu priradí hodnotu určitej vlastnosti (napr. názov, dĺžku, alebo hmotnosť). Entita alebo vzťah môže obsahovať viacero atribútov. Entity patriace rovnakému typu majú rovnakú množinu atribútov. Podobne vzťahy toho istého typu majú rovnakú množinu atribútov. Každá inštancia entity alebo vzťahu má ale vlastné hodnoty atribútov.

V ER diagrame atribút reprezentujeme oválom obsahujúcim jeho názov. Ovál je spojený čiarou s typom entity alebo typom vzťahu ktorému patrí.

Príklad ER modelu obsahujúceho atribúty je zobrazený na obrázku č. 2.10. Typ entity `Zamestnanec` má atribúty: `Meno`, `Adresa`, `Plat` a `Dátum narodenia`. Entita typu `Projekt` má atribúty: `Názov`, `Popis` a `Dátum začiatku` (kedy sa projekt začal). Vzťah typu `Pracuje na` má atribúty: `Dátum začiatku` (od kedy zamestnanec pracuje na projekte) a `Odpracované hodiny`. Každá inštancia typu `Zamestnanec`, `Pracuje na` a `Projekt` má vlastné hodnoty atribútov. Tieto hodnoty sú uložené v databáze.



Obrázok 2.10: Atribúty typu entity a typu vzťahu

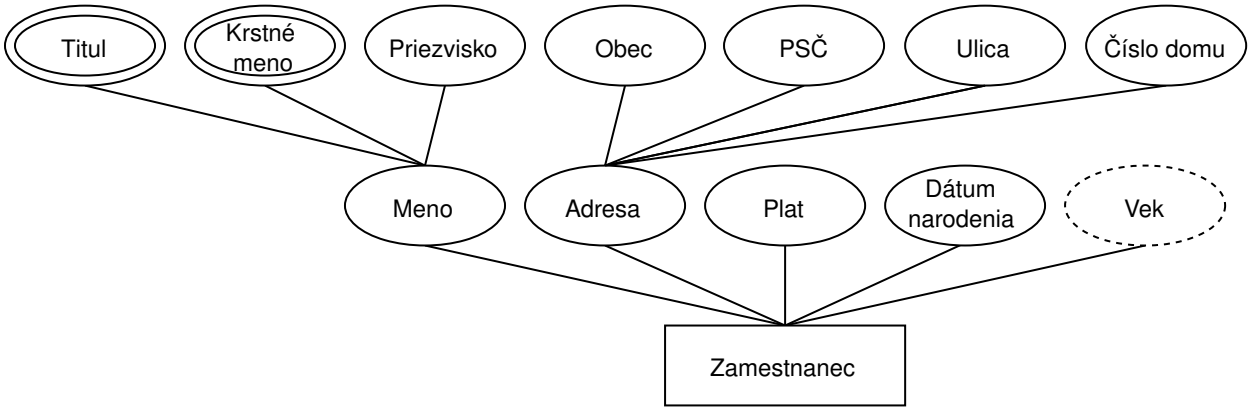
Definičný obor atribútu (angl.: domain of an attribute) je množina všetkých hodnôt, ktoré môže nadobúdať atribút inštancie entity alebo vzťahu. V ER diagrame doménu nezobrazujeme.

Podľa komplexnosti rozlišujeme viaceré typy atribútov. **Zložený atribút** (angl.: composite attribute) sa skladá z viacerých nezávislých častí (z viacerých podatribútov). Podatribút zloženého atribútu môže byť tiež zloženým atribútom.

Takýmto atribútom môže byť napríklad meno osoby, skladajúce sa z krstných mien, priezviska, titulov. Na obr. č. 2.11 sú zloženými atribútmi `Meno` a `Adresa`. Modely atribútov `Meno` a `Adresa` sú zjednodušené. Atribút `Meno` napríklad neobsahuje informáciu o umiestnení titulov v celom mene. Atribút `Adresa` by mohol ešte obsahovať krajinu, čísla vchodu, čísla bytu.

Jednoduchý (atomický) atribút (angl.: simple (atomic) attribute) je atribút, ktorý nie je možné zmysluplne deliť. Na obr. č. 2.11 sú atomickými atribútmi napríklad `Priezvisko` a `Plat`.

Zložené atribúty sú užitočné ak sa v niektorých prípadoch používajú ako celok, ale v niektorých prípadoch sa používa iba ich časť. Ak vždy používame len celé meno osoby, tak na reprezentáciu mena stačí jeden jednoduchý atribút. Ak okrem celého mena, potrebujeme zvlášť pracovať aj s jeho časťami (napr. usporadúvať zamestnancov podľa priezviska, alebo potrebujeme vytvárať rôzne formáty výpisu mena), tak je potrebné meno reprezentovať ako zložený atribút.



Obrázok 2.11: Zložene, viachodnotové a odvodené atribúty

Viachodnotový atribút (angl. multivalued attribute) je atribút, ktorý môže obsahovať v jednom časovom okamihu viac hodnôt naraz. Napríklad človek môže mať viacero titulov, alebo krstných mien, predmet môže mať viac farieb. V ER diagrame takýto atribút znázorňujeme dvojitým oválom (obr. č. 2.11).

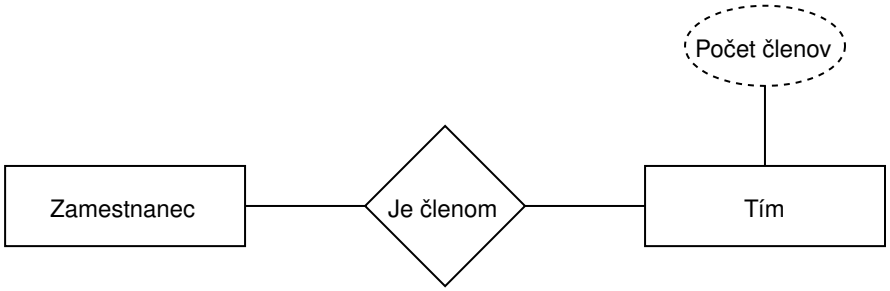
Atribút, ktorý v jednom časovom okamihu môže obsahovať najviac jednu hodnotu, nazývame **jednohodnotový atribút** (angl.: single valued). Napríklad priezvisko sa môže meniť, ale v jednom časovom okamihu má človek jedno priezvisko. V ER diagrame takéto atribúty nemajú špeciálne označenie.

Niektoré atribúty nemusia obsahovať žiadnou hodnotu, pretože je neznáma, prípadne ju ani nemá zmysel určovať. Príklad neznámej hodnoty je hmotnosť tovaru. Tovar má svoju hmotnosť, ale nie je zaznamenaná, prípadne ani zmeraná. Príklad hodnoty, ktorú v niektorých prípadoch nemá zmysel určovať je titul (človek nemusí mať titul), alebo ulica v adrese (ak adresa smeruje do obce, ktorá nemá ulice). Ak hodnota atribútu nie je určená, tak sa pre hodnotu atribútu používa označenie **NULL**. V spôsobe kreslenia ER diagramu ktorý používame, nie je možnosť naznačiť povinnosť alebo voliteľnosť hodnoty atribútu. Niektoré iné spôsoby kreslenia ER diagramov túto možnosť ale obsahujú.

Hodnoty niektorých atribútov môžeme odvodiť z iných informácii, ktoré sú v databáze zaznamenané, alebo sú pre informačný systém dostupné. Napríklad z hodnôt iných atribútov zo súvisiacich entít, alebo z informácii o aktuálnom čase. Napríklad vek zamestnanca možno odvodiť z dátumu jeho narodenia a aktuálneho času (obr. č. 2.11). Počet členov pracovného tímu môžno zistiť sčítaním počtu jeho členov (obr. č. 2.12). Takéto atribúty sa nazývajú **odvodené atribúty** (angl.: derived attributes). V ER diagrame ich označujeme čiarkovaným oválom.

Hodnoty odvodených atribútov nie je potrebné v databáze ukladať. Tieto hodnoty je často výhodnejšie vypočítavať za behu, pretože ich uloženie v databáze by mohlo spôsobiť nekonzistenciu údajov. Napríklad pri pridaní člena do tímu, môže programátor zabudnúť aktualizovať počet členov tímu. Udržiavanie záznamu obsahujúceho aktuálny vek zamestnanca by bolo tiež komplikované. Pri implementácii ale môže nastať rozhodnutie, ukladať hodnoty týchto atribútov v databáze z dôvodu optimalizácie. Na konceptuálnej úrovni o týchto implementačných detailoch ale nerozhodujeme. V ER modely iba zaznačíme, že atribút je odvodený.

Atribúty, ktoré je v databáze potrebné ukladať sa nazývajú **uložene atribúty** (angl.: stored attributes). Keďže tvoria prevažnú väčšinu atribútov, v ER diagrame nemajú špeciálne označenie.

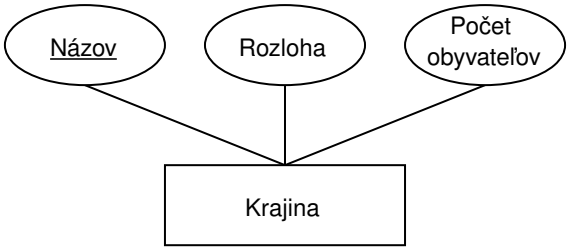


Obrázok 2.12: Hodnota atribútu odvodená z počtu súvisiacich entít

2.1.5 Kľúč

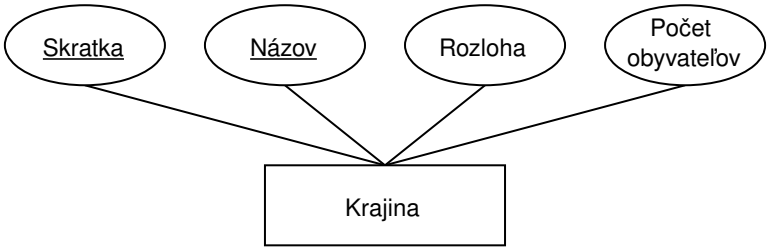
Kľúč (angl.: key, alebo key attribute) je jeden alebo viac atribútov typu entity, ktorého hodnoty alebo kombinácie hodnôt sú odlišné pre každú inštanciu daného typu entity. Hodnota kľúča môže byť preto použitá na jednoznačnú identifikáciu entity.

Kľúč je identifikátor, ktorý navzájom odlišuje entity toho istého typu. Entity rôznych typov môžu mať hodnotu kľúča rovnakú.



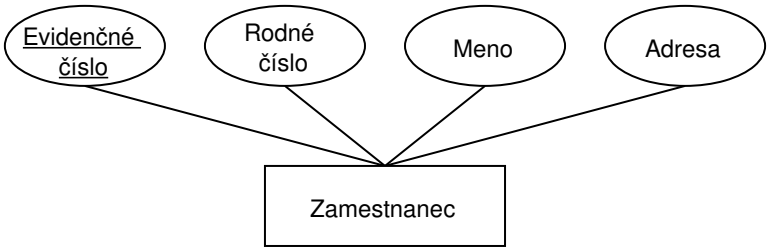
Obrázok 2.13: Kľúč (identifikátor entity)

Krajina na obr. č. 2.13 môže byť jednoznačne identifikovaná názvom. Preto názov krajiny tvorí kľúč. V ER diagrame označujeme kľúč podčiarknutím jeho názvu.



Obrázok 2.14: Entita má dva kľúče

Typ entity môže obsahovať viacero atribútov, ktorých hodnoty (samostatne, nie ich kombinácie) môžu jednoznačne identifikovať inštanciu daného typu. Ak sú v databáze evidované aj skratky názvov krajín, tak typ entity `Krajina` obsahuje dva kľúče (krajinu môžeme jednoznačne identifikovať podľa názvu alebo skratky). Na obr. č. 2.14 sú zobrazené dva kľúče (dva jednoznačné identifikátory). V ER diagrame podčiarkujeme všetky kľúče.



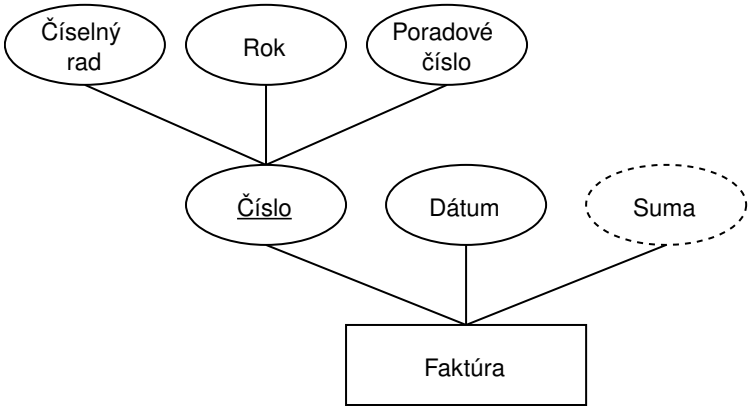
Obrázok 2.15: Umelý kľúč

Pri návrhu ER modelu často nastáva prípad, kedy typ entity neobsahuje **prirodzený kľúč** (angl.: natural key). Preto musíme doplniť **umelý kľúč** (ang.: surrogate key). Pri evidencii zamestnancov (obr. č. 2.15) by sme teoreticky mohli ako kľúč zvoliť rodné číslo zamestnanca[3]. Z hľadiska návrhu by to bol prirodzený kľúč. V praxi sa ale zistilo, že niektorý ľudia majú to isté rodné číslo. Preto musíme doplniť umelý kľúč, ktorý môžeme nazvať `Evidenčné číslo`.

3 Podľa zákona č. 301/1995 Z. z. je rodné číslo trvalý identifikačný osobný údaj fyzickej osoby, ktorý zabezpečuje jej jednoznačnosť v informačných systémoch.

Voľba prirodzeného alebo umelého kľúča má rôzne výhody a nevýhody. Niektoré sa prejavia už na konceptuálnej úrovní, iné až na úrovniach bližších ku implementácii. Vlastnosti prirodzených a umelých kľúčov:

- Podľa hodnoty prirodzeného kľúča môžeme odvodiť ďalšie informácie o entite. Jedinou informáciou, ktorú nesie umelý kľúč je identifikácia entity (v závislosti od implementácie to tak ale nemusí byť). Napríklad z názvu krajiny vieme aj bez použitia databázy zistiť jej umiestnenie a rozlohu. Z umelo vytvoreného evidenčného čísla nemusíme vedieť zistiť nič (aj keď napríklad pri použití algoritmu vytvárania evidenčných čísiel, takým spôsobom, že každé nové číslo má hodnotu vyššiu ako predchádzajúce, môžeme odvodiť niektoré informácie).
- Použitím prirodzeného kľúča nepridávame ďalšie informácie.
- Prirodzený kľúč súvisí s oblasťou, o ktorej databáza nesie informácie. Preto ak sa zmenia pravidlá v danej oblasti, tak pri používaní prirodzených kľúčov, môže byť zmena databázy náročnejšia, ako pri používaní umelých kľúčov, pretože umelý kľúč priamo nesúvisí s pravidlami v danej oblasti.
  Zoberme si ako príklad evidenciu miestností. Ak ako kľúč použijeme označenie, ktoré je umiestnené na štítku pri vstupe do miestnosti (prirodzený kľúč), tak pri zmene označení miestností bude úprava databázy náročnejšia, ako keby sme používali umelo vytvorené evidenčné číslo miestnosti (umelý kľúč).
  Príkladom, kde by podobný problém nemal nastať, je použitie čísla faktúry ako prirodzeného kľúča. Podľa zákona sa číslo faktúry nemôže zmeniť, takže aj v prípade zmeny súvisiacich zákonov, by nemala nastať potreba meniť hodnoty kľúčov.
  Pozor na to, že ak by koncový používatelia aplikácie začali používať umelý kľúč, tak sa umelý kľúč stane súčasťou oblasti o ktorej sa v databáze evidujú údaje a v prípade zmeny podmienok, môže byť zmena databázy náročnejšia.
- Umelý kľúč môže byť ťažšie čitateľný pre človeka, alebo je ho ťažšie použiť pri komunikácii ľudí.
- Zostavenie príkazov vyhľadávacieho algoritmu môže byť jednoduchšie pri používaní prirodzených kľúčov.
- Umelý kľúč väčšinou zaberá menej pamäte ako prirodzený kľúč. Porovnávanie hodnôt umelých kľúčov je na implementačnej úrovni väčšinou efektívnejšie.



Obrázok 2.16: Zložený kľúč

Ak je kľúč tvorený viacerými atribútmi (entita je identifikovaná kombináciou hodnôt viacerých atribútov), tak treba vytvoriť **zložený kľúč** (angl.: composite key). Zložený kľúč musí byť

minimálny (nesmú v ňom byť podatribúty, ktoré nie sú súčasťou kľúča). Zložený kľúč sa v ER diagrame označuje podčiarknutím zloženého atribútu, ktorý je kľúčom. Na obr. č. 2.16 je faktúra identifikovaná jej „číslom“ („číslo“ v skutočnosti môže obsahovať aj iné znaky)[4], ktoré sa skladá z číselného radu[5], roku a poradového čísla.

Prípady viacerých kľúčov a zloženého kľúča sú popísané podľa [2]. Keďže ER model nie je štandardizovaný, iná literatúra môže popisovať definovanie kľúčov inak. Príkladom je [3].

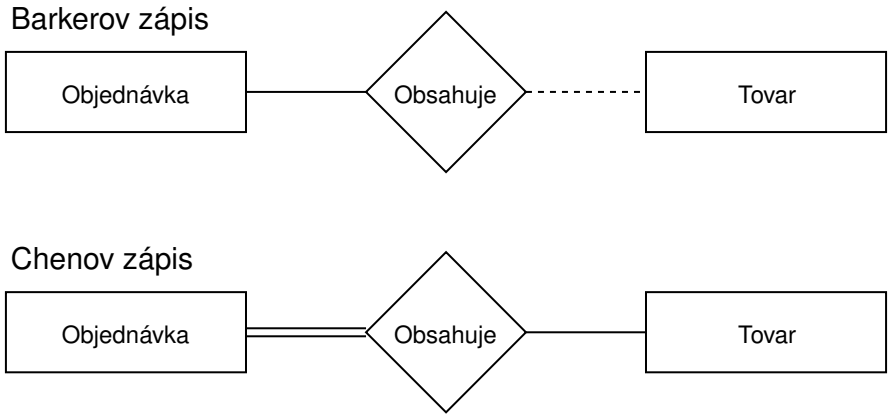
Typ entity nemusí mať kľúč. V tom prípade ide o typ slabej entity (časť 2.1.8).

2.1.6 Násobnosť a voliteľnosť binárnych vzťahov

V tejto časti sa budeme venovať násobnosti a voliteľnosti len binárnych vzťahov. Násobnosť a voliteľnosť vzťahov vyššieho stupňa budeme preberať neskôr, pretože táto problematika je výrazne náročnejšia.

2.1.6.1 Voliteľnosť členstva vo vzťahu

V ER modely môžeme určiť požiadavku na povinnosť alebo nepovinnosť účasti entity vo vzťahu. **Povinná účasť** (angl.: mandatory / total participation) typu entity v type vzťahu znamená, že entita daného typu musí byť v danom type vzťahu. **Nepovinná (voliteľná) účasť** (angl.: optional / partial participation) typu entity v type vzťahu znamená, že entita daného typu môže, ale nemusí byť v danom type vzťahu.



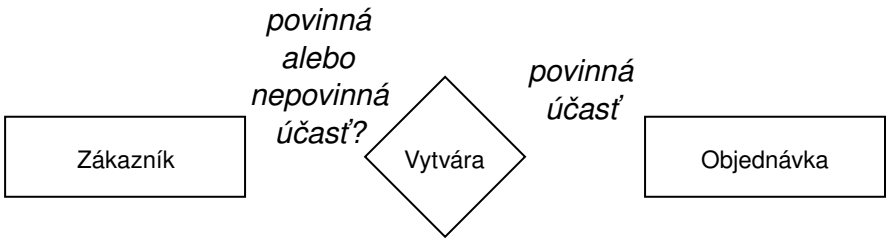
Obrázok 2.17: Voliteľnosť účasti vo vzťahu. Objednávka má povinnú účasť vo vzťahu, ale tovar má nepovinnú účasť.

Napríklad v databáze patriacej obchodu môžeme evidovať údaje o druhoch tovaru a objednávkach na tieto tovary (obr. č. 2.17). Každá objednávka musí byť vystavená aspoň na jeden tovar (povinná účasť objednávky vo vzťahu s tovarom), ale na tovar nemusí byť vystavená žiadna objednávka (nepovinná účasť tovaru vo vzťahu s objednávkou). Na obrázku sú znázornené dva spôsoby zápisu voliteľnosti vo vzťahu pomocou rôznych typov čiar: Barkerov a Chenov. V tomto texte ale tieto zápisy nebudeme používať. Sú tu uvedené z dôvodu, že sa čitateľ môže stretnúť s týmito druhmi zápisu pri štúdiu inej literatúry. Zápis ktorý budeme používať, bude uvedený až po prebratí témy o

4 Zákon nepredpisuje spôsob označovania faktúr, preto v praxi môžete použiť aj iný spôsob.

5 Číselným radom môžeme označiť typ faktúry, podľa nami zvoleného rozdelenia. Napríklad na riadne (ostré) faktúry, zálohové (preddavkové) faktúry, zahraničný faktúry, atď.

násobnosti vzťahov, pretože v nami používanom spôsobe zápisu nie je možnosť určiť voliteľnosť bez určenia násobnosti vzťahu. Voliteľnosť budeme určovať pomocou minimálnej násobnosti vzťahu (časť 2.1.6.3).

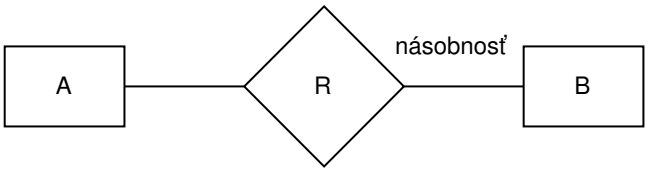


Obrázok 2.18: Voľba účasti zákazníka vo vzťahu reprezentujúcom vytvorenie objednávky

Určenie voliteľnosti nemusí byť vždy hneď jasné. Napríklad databáza pre obchod môže obsahovať typ vzťahu reprezentujúci vytvorenie objednávky zákazníkom (obr. č. 2.18). Ak je niekto zákazník, znamená to, že vytvoril aspoň jednu objednávku. Táto úvaha by viedla k tomu, že účasť zákazníka vo vzťahu reprezentujúcom vytvorenie objednávky je povinná. Toto určenie voliteľnosti ale nemusí byť vhodné, ak sa z marketingových dôvodov rozhodneme aj pre možnosť registrovania potenciálnych zákazníkov, ktorí ešte nevytvorili žiadnu objednávku. V takomto prípade je potrebné, aby účasť zákazníka vo vzťahu bola nepovinná.

2.1.6.2 Násobnosť vzťahu

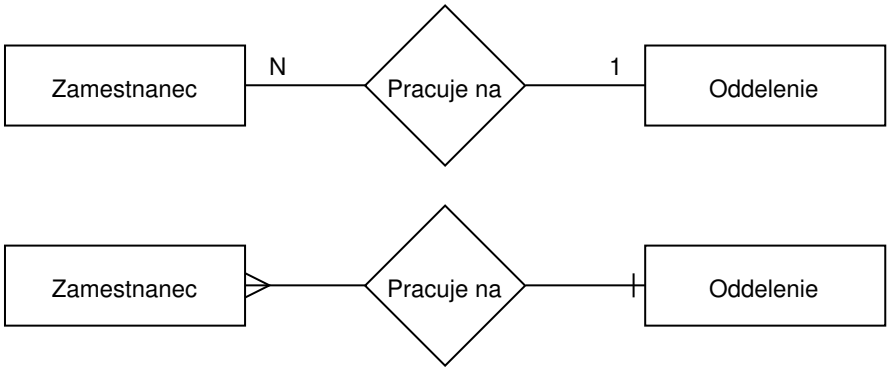
Násobnosť (mohutnosť) (angl.: cardinality) určuje pre každú inštanciu typu entity, maximálny počet inštancii typu entity, ktorý je na druhej strane daného typu vzťahu. Na obr. č. 2.19 je znázornený zápis násobnosti určujúci pre každú inštanciu `A`, maximálny počet inštancii `B`, s ktorými je vo vzťahu typu `R`. Násobnosti ale určujeme na obidvoch stranách typu vzťahu.



Obrázok 2.19: Násobnosť

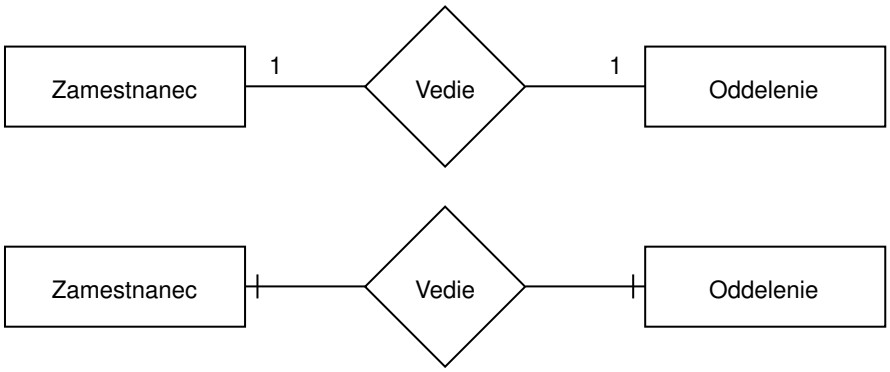
Diagram na obr. č. 2.20 znázorňuje, že každý zamestnanec môže pracovať maximálne na jednom oddelení a na každom oddelení môže pracovať neobmedzený počet zamestnancov. N znamená neobmedzený maximálny počet. **Pomer násobnosti (vzájomnú násobnosť)** (angl.: cardinality ratio) tohto vzťahu skrátene označujeme 1:N[6] (výslovnosť: 1 ku N). Diagram znázorňuje dva často používané spôsoby znázornenia násobnosti entít vo vzťahu: alfanumerickým znakom alebo grafickou značkou na konci čiary. Význam grafických značiek je na obr. č. 2.23. V prípade potreby môžeme okrem 1 a N použiť ľubovoľné čísla. Zápis ER diagramu nie je štandardizovaný a preto sa pri štúdiu inej literatúry môže vyskytnúť prípad, že značenia násobností budú na opačných stranách.

6 Môžeme použiť aj označenie N:1, ale väčšinou sa používa poradie 1:N.



Obrázok 2.20: Vzťah 1:N (Chenov zápis)

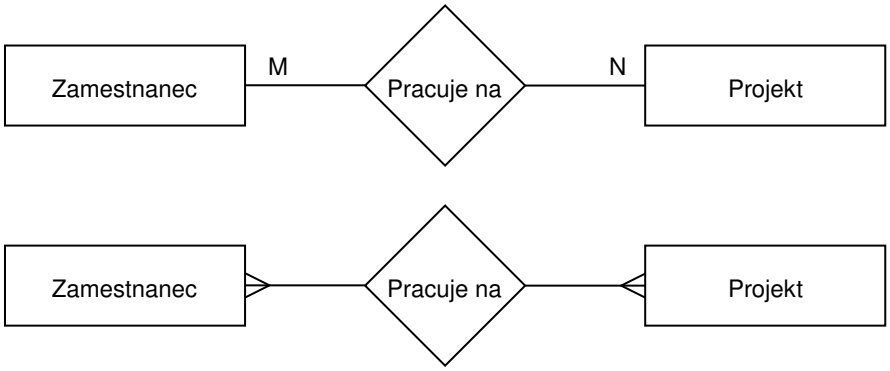
Násobnosti na obr. č. 2.21 znázorňujú, že každý zamestnanec môže viesť najviac jedno oddelenie a každé oddelenie môže byť vedené najviac jedným zamestnancom. Tento pomer násobnosti vzťahu označujeme 1:1 (výslovnosť: 1 ku 1).



Obrázok 2.21: Vzťah 1:1

Násobnosti na obr. č. 2.22 znázorňujú, že každý zamestnanec môže pracovať na neobmedzenom počte projektov a na každom projekte môže pracovať neobmedzený počet zamestnancov. M a N označujú neobmedzený počet[7]. Tento pomer násobnosti vzťahu označujeme M:N (výslovnosť: M ku N).

7 N je najčastejšie písmeno, ktoré sa zvykne používať pre označenie neobmedzeného počtu, ale ak sa v jednom type vzťahu vyskytujú viaceré neobmedzené počty, tak sa použije ďalšie písmeno, pretože počet entít na obidvoch stranách vzťahu môže byť rôzny. V ER diagrame s viacerými vzťahmi budeme ale pre jednoduchosť označovať neobmedzený počet v každom type vzťahu tým istým písmenom (N prípadne ďalším). Pre označenie neobmedzeného počtu sa namiesto písmena používa aj hviezdička.

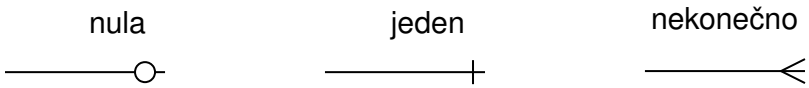


Obrázok 2.22: Vzťah M:N

2.1.6.3 Označenie násobnosti a voliteľnosti vzťahu rozsahom (intervalom)

V predchádzajúcich príkladoch sme určovali násobnosť a voliteľnosť zvlášť. Pri určovaní obidvoch charakteristík, budeme používať rozsah hodnôt. Minimálna hodnota rozsahu bude väčšinou 0 alebo 1, maximálna hodnota bude väčšinou 1 alebo N. V týchto prípadoch minimálna hodnota označuje voliteľnosť (0) alebo povinnosť vzťahu (1), maximálna hodnota označuje násobnosť vzťahu (1...N).

Namiesto alfanumerických znakov, môžeme použiť aj grafické značky. Grafické značky pre hodnoty 0, 1 a N (nekonečno) sú znázornené na obr. č. 2.23.



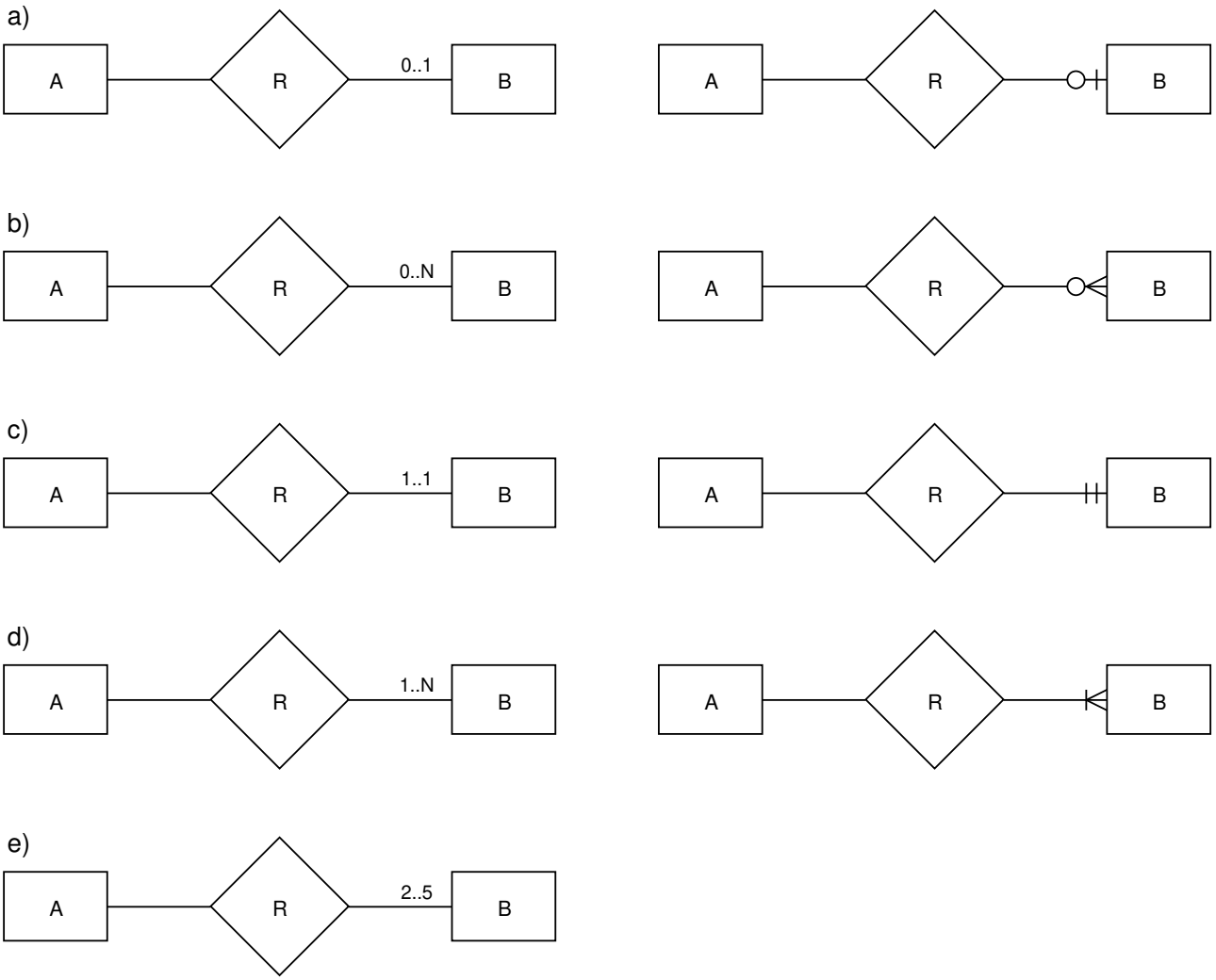
Obrázok 2.23: Grafické značky pre označenie najčastejších prípadov násobnosti a voliteľnosti vzťahu

V prípade potreby môžeme použiť aj iné hodnoty rozsahu, napríklad rozsah 2 až 5. Pre tieto málo vyskytujúce sa prípady, ale nie sú definované grafické značky. Pojem voliteľnosti/povinnosti vo vzťahu zodpovedá minimálnym hodnotám rozsahu 0 alebo 1.

Pri určení voliteľnosti aj násobnosti spolu, sa často používa len pojem násobnosť. Dôvodom je stručnejšie vyjadrovanie sa a možnosť použitia aj iných hodnôt ako 0 a 1 pre minimum rozsahu násobnosti.

Na obr. č. 2.24 sú znázornené príklady násobností a voliteľností (*Chenov zápis*). Na ľavej strane obrázku je rozsah znázornený alfanumericky, na pravej strane je ten istý rozsah znázornený grafickými značkami. Pre zjednodušenie je násobnosť a voliteľnosť určená len na jednej strane typu vzťahu, v reálnom modely by mala byť určená na obidvoch stranách. Popis jednotlivých prípadov:

a) každá inštancia typu A môže byť vo vzťahu typu R s nula alebo jednou inštanciou typu B
b) každá inštancia typu A môže byť vo vzťahu typu R s ľubovolným počtom inštancii typu B,
c) každá inštancia typu A musí byť vo vzťahu typu R s jednou inštanciou typu B,
d) každá inštancia typu A musí byť vo vzťahu typu R s minimálne jednou inštanciou typu B,
e) každá inštancia typu A musí byť vo vzťahu typu R s minimálne 2 a maximálne 5 inštanciami typu B. Grafické značky pre tento prípad nie sú definované.

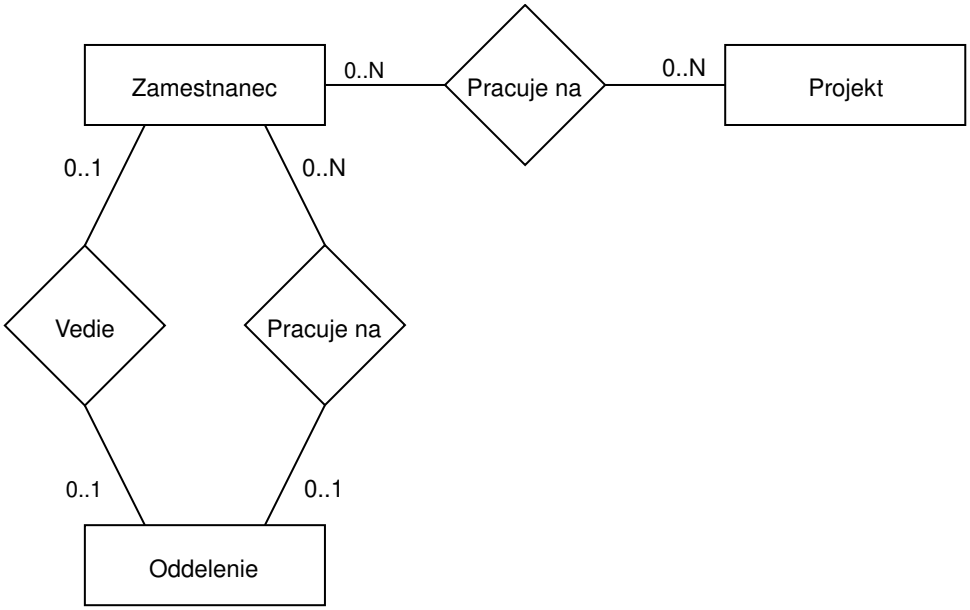


Obrázok 2.24: Určenie násobnosti aj voliteľnosti vzťahu rozsahom hodnôt (Chenov zápis)

Na obr. č. 2.25 je príklad určenia násobnosti vzťahov entity typu `Zamestnanec` s entitami typu `Projekt` a `Oddelenie`. Popis násobnosti vzťahov:

- Zamestnanec môže pracovať na ľubovolnom počte projektov.
- Na projekte môže pracovať ľubovoľný počet zamestnancov. Ak vo firme existuje projekt, niekto ho musí inicializovať a teda na ňom pracuje. Minimum rozsahu je ale 0, pretože napríklad v malej firme môže všetky projekty vybavovať jediný majiteľ a neskôr ich prideliť zamestnancom. Čiže majiteľ pracuje na každom projekte. Z dôvodu prehľadnosti ale môže požadovať, aby informačný systém nevyžadoval minimálne jedného pracovníka na projekte.
- Zamestnanec môže pracovať najviac na jednom oddelení. Či zamestnanec musí byť vždy zaradený na oddelenie treba prediskutovať pri tvorbe špecifikácie systému. Napríklad pri prijatí zamestnanca môže byť tento evidovaný v databáze, ale nemusí byť ihneď zaradený na oddelenie.
- Na oddelení môže pracovať ľubovoľný počet zamestnancov. Pri špecifikácii požiadaviek treba zvážiť, či môže existovať oddelenie bez zamestnancov. Napríklad či pri vytvorení a zaevidovaní nového oddelenia do systému musia byť oddeleniu priradení zamestnanci.
- Zamestnanec môže viesť najviac jedno oddelenie.

- Oddelenie je pod vedením najviac jedného zamestnanca. Na krátky (alebo dlhší) čas sa môže stať, že oddelenie nebude mať svojho vedúceho.



Obrázok 2.25: Určenie násobností a voliteľností vo firme (Chenov zápis)

Tento spôsob zápisu rozsahu násobnosti sa nazýva **Chenov zápis**. Tento zápis budeme používať. V literatúre sa ale môžete stretnúť aj s ďalším typom zápisu, ktorý sa nazýva **Merisov zápis**. Merisov zápis má inú definíciu, ale v binárnych vzťahoch je jej význam ten istý, odlišuje sa len zápisom (rozsahu) násobnosti na druhej strane typu vzťahu. Rozdiel vo význame sa prejaví až vo vzťahoch vyššieho stupňa.

V Merisovom zápise by násobnosť na obr. č. 2.19 znamenala maximálny počet účastí inštancie `B` vo vzťahu typu `R` (v koľkých vzťahoch typu `R` môže byť jedna inštancia `B`).

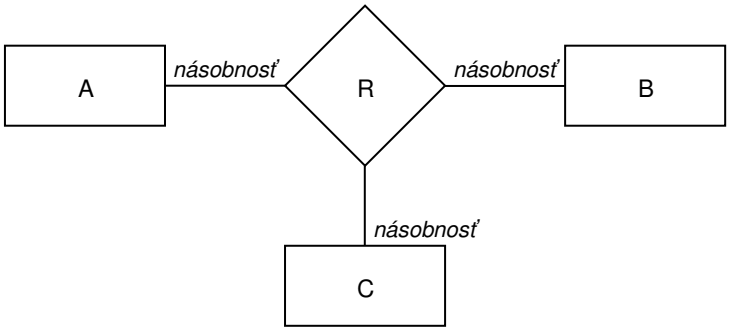
Chenov zápis môžeme nazvať **pozri na druhú stranu** (angl. look-across). Budeme preň používať skratu **LA**. Merisov zápis môžeme nazvať **pozri tu** (angl. look-here). Budeme preň používať skratku **LH**.

2.1.7 Násobnosť vzťahov vyššieho stupňa

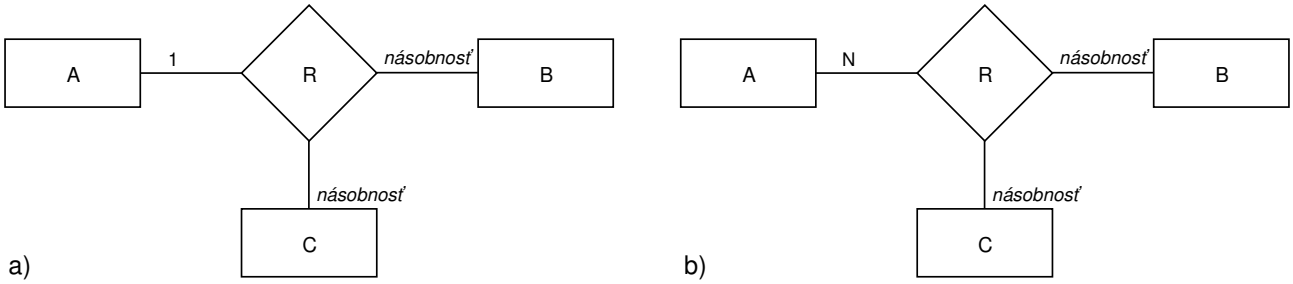
Pojmom **vzťah vyššieho stupňa** budeme označovať vzťahy stupňa vyššieho ako dva.

Problematika násobnosti vzťahov vyššieho stupňa je náročnejšia ako problematika násobnosti vzťahov druhého stupňa. Preto sa jej budeme venovať zvlášť. Príklad ternárneho vzťahu je na obr. č. 2.3. Pri vzťahoch vyššieho stupňa môžeme tiež určiť jeho násobnosť, ale najpopulárnejšie dva prístupy LA a LH neumožňujú špecifikovať všetky informácie o násobnosti (ani vzájomná kombinácia týchto dvoch prístupov). Ich vyjadrovacia schopnosť je nedostatočná. To znamená, že ER diagram nezachytáva všetky potrebné informácie o násobnosti vzťahov vyššieho stupňa a je ich potrebné doplniť slovným popisom.

2.1.7.1 Horná hranica násobnosti



Obrázok 2.26: Násobnosť ternárneho vzťahu



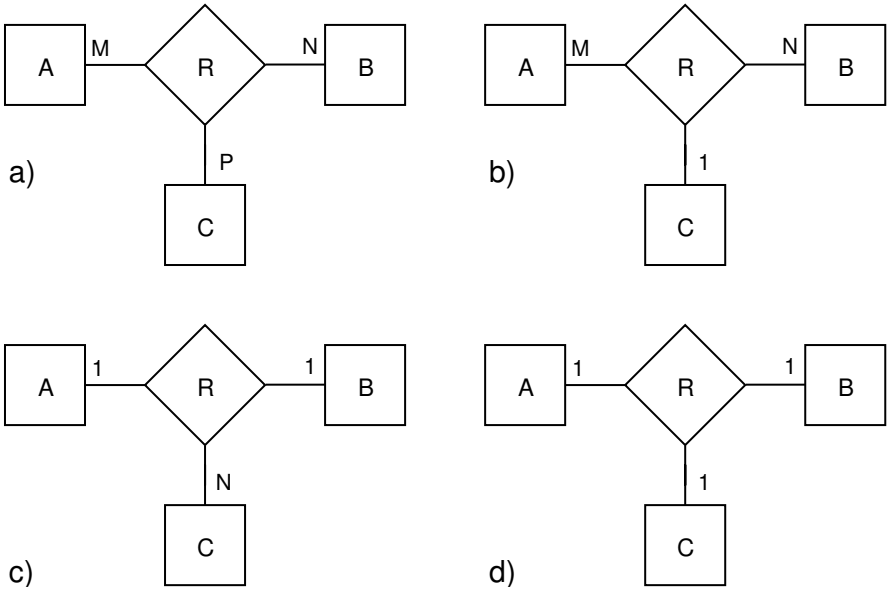
Obrázok 2.27: Násobnosť typu A v ternárnom vzťahu

V zápise LA, sa násobnosť typu entity v type vzťahu určuje tak, že pre ľubovoľnú kombináciu inštancii ostatných typov entít zúčastňujúcich sa vo vzťahu, určíme maximálny počet jej inštancii. Pri určovaní násobnosti typu A na obr. č. 2.26, vyberieme jednu inštanciu typu B a jednu typu C. Násobnosť typu A je maximálny počet jej inštancii, ktoré môžu byť vo vzťahu typu R, s vybranými inštanciami typu B a C. Príklad na obr. č. 2.27 a), v interpretácii podľa LA reprezentuje, že každá kombinácia inštancii typu B a C môže byť vo vzťahu typu R, maximálne s jednou inštanciou typu A. Príklad na obr. č. 2.27 b) reprezentuje, že každá kombinácia inštancii typu B a C môže byť vo vzťahu typu R, s neobmedzeným počtom inštancii typu A.

V zápise LH, násobnosť typu entity znamená maximálny počet inštancii typu vzťahu, v ktorých sa môže účastniť. Násobnosť typu A na obr. č. 2.26, je maximálny počet inštancii typu R, v ktorých sa môže ľubovolná inštancia typu A účastniť. Príklad na obr. č. 2.27 a), v interpretácii podľa LH reprezentuje, že každá inštancia typu A môže byť maximálne v jednom vzťahu typu R. Príklad na obr. č. 2.27 b) reprezentuje, že každá inštancia typu A môže mať neobmedzený počet vzťahov typu R.

Tu sa už prejavil rozdiel prístupov LA a LH v sémantike (význame) násobnosti uvedenej pri entite. V binárnych vzťahoch je rozdiel len v zápise ER diagramu (v umiestnení násobností na opačných stranách vzťahu).

Poznámka: inštancia vzťahu typu R na obr. č. 2.26 a 2.27 je trojica inštancii typu A, B a C.

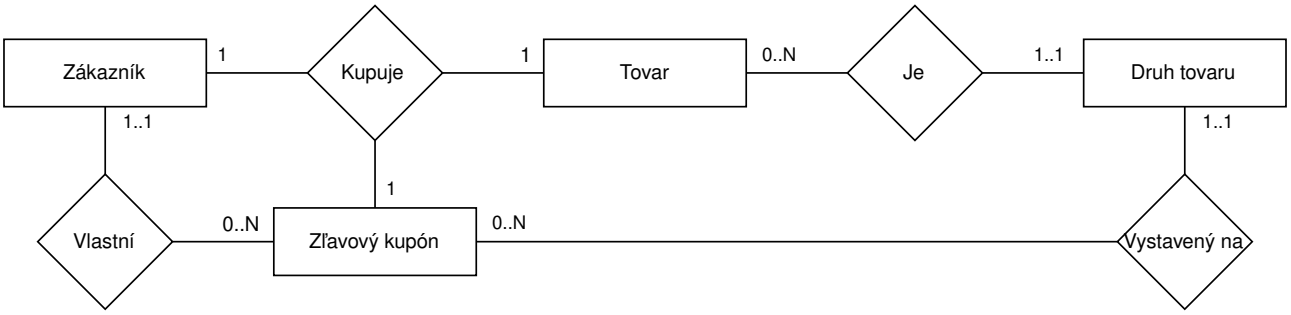


Obrázok 2.28: Možné násobnosti v ternárnom vzťahu

Na obr. č. 2.28 sú znázornené všetky prípady násobnosti ternárneho vzťahu. Nasledujúci text bude brať do úvahy iba interpretáciu podľa LA (interpretácia podľa LH by bola iná). V prípade na obr. č. 2.28 a) je násobnosť vzťahu M:N:P. Pre identifikáciu inštancie vzťahu typu R, potrebujeme poznať všetky inštancie zúčastnených typov entít. Na obr. č. 2.28 b) je vzťah 1:M:N. V tomto prípade pre identifikáciu inštancie vzťahu typu R, stačí poznať inštancie typov A a B, pretože každá dvojica inštancii typu A a B, môže byť vo vzťahu typu R, maximálne s jednou inštanciou typu C. Na obr. č. 2.28 c) je vzťah typu 1:1:N. Pre identifikáciu inštancie vzťahu typu R v tomto prípade postačuje poznať iba inštanciu typu C, pretože každá dvojica inštancii typu A a C môže byť vo vzťahu typu R maximálne s jednou inštanciou typu B a zároveň každá dvojica inštancii typu B a C, môže byť vo vzťahu typu R, maximálne s jednou inštanciou typu A. Na obr. č. 2.28 d) je vzťah typu 1:1:1. V tomto prípade pre identifikáciu inštancie vzťahu R stačí poznať ľubovolnú z inštancii typov A, B a C, pretože každá kombinácia dvoch inštancii účastniacich sa typov entít, môže byť vo vzťahu s maximálne jednou inštanciou zvyšného typu účastniacej sa entity.

Na nasledujúcich obrázkoch sú uvedené príklady všetkých typov násobnosti ternárneho vzťahu pri použití LA. Násobnosti na obr. č. 2.29 znamenajú, že

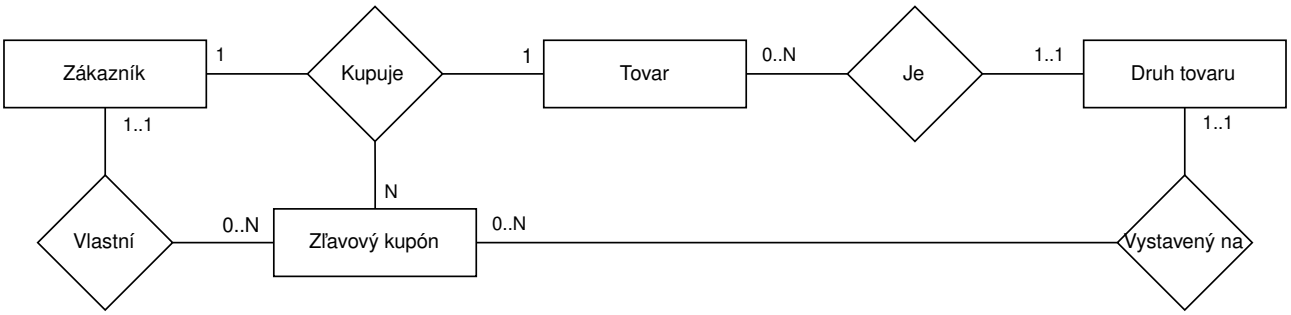
- jeden zákazník môže na jeden kus tovaru uplatniť najviac jeden zľavový kupón,
- jeden zákazník môže jeden zľavový kupón použiť najviac na jeden kus tovaru,
- jeden kus tovaru a jeden zľavový kupón súvisí v rámci nákupov najviac s jedným zákazníkom.



Obrázok 2.29: Horná hranica násobnosti ternárneho vzťahu 1:1:1 (evidencia kusov aj druhov tovaru)

Násobnosti na obr. č. 2.30 znamenajú, že

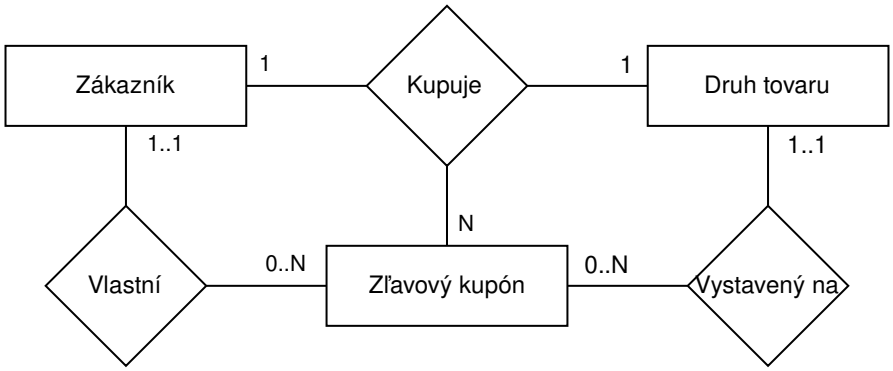
- jeden zákazník môže na jeden kus tovaru uplatniť neobmedzený počet zľavových kupónov,
- jeden zákazník môže jeden zľavový kupón použiť najviac na jeden kus tovar,
- jeden kus tovaru a jeden zľavový kupón súvisý v rámci nákupov najviac s jedným zákazníkom.



Obrázok 2.30: Horná hranica násobnosti ternárneho vzťahu 1:1:N (evidencia kusov aj druhov tovaru)

Násobnosti na obr. č. 2.31 znamenajú, že:

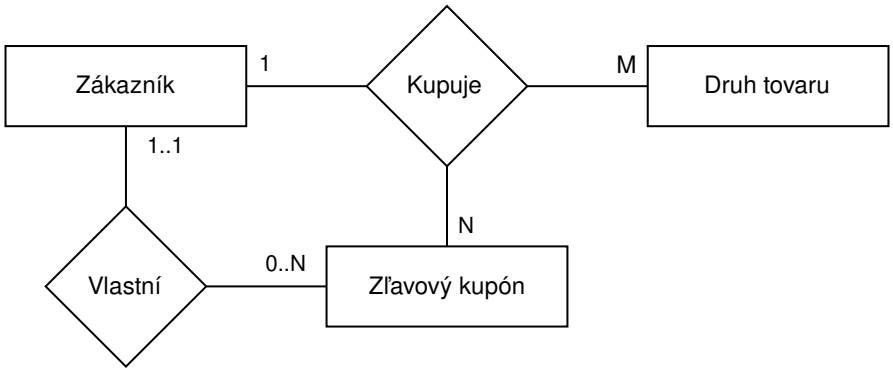
- jeden zákazník môže na jeden druh tovaru použiť neobmedzené množstvo zľavových kupónov,
- jeden zákazník môže uplatniť jeden zľavový kupón najviac na jeden druh tovaru,
- kombinácia jedeného druh tovaru a jedného zľavového kupónu súvisí v rámci nákupov najviac s jedným zákazníkom.



Obrázok 2.31: Horná hranica násobnosti ternárneho vzťahu 1:1:N (evidencia obsahuje len druhy tovaru)

Násobnosti na obr. č. 2.32 znamenajú, že

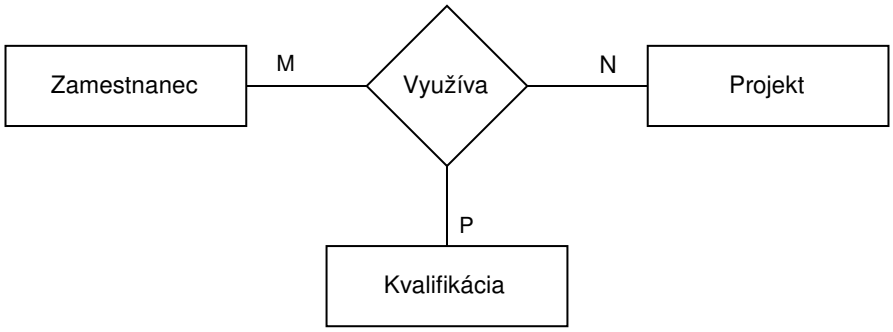
- jeden zákazník môže na jeden druh tovaru použiť neobmedzené množstvo zľavových kupónov (počas jedného alebo viacerých nákupov),
- jeden zákazník môže uplatniť jeden zľavový kupón na neobmedzený počet druhov tovaru,
- kombinácia jedeného druh tovaru a jedného zľavového kupónu súvisí v rámci nákupov najviac s jedným zákazníkom.



Obrázok 2.32: Horná hranica násobnosti ternárneho vzťahu 1:M:N (evidencia obsahuje len druhy tovaru)

Násobnosti na obr. č. 2.33 znamenajú, že

- jeden zamestnanec na jednom projekte môže využiť neobmedzené množstvo kvalifikácii,
- jeden zamestnanec môže jednu kvalifikáciu využiť na neobmedzenom množstve projektov,
- na jednom projekte môže jednu kvalifikáciu využívať neobmedzene množstvo zamestnancov.



Obrázok 2.33: Horná hranica násobnosti ternárneho vzťahu M:N:P

2.1.7.2 Dolná hranica násobnosti

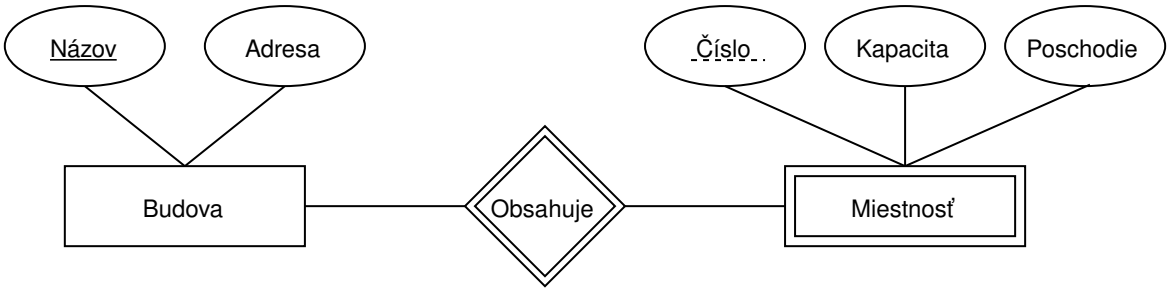
V zápise LA sa minimálna hranica násobnosti (voliteľnosť vo vzťahu) neurčuje, pretože by to nemalo zmysel. Minimálna hodnota by nevyjadrovala voliteľnosť. Podrobnejšie sa tomuto problému pre jeho zložitosť nebudeme venovať. Viac informácii o tejto problematike nájdete napríklad v [4].

2.1.8 Slabá entita

Typy entít, s ktorými sme sa doteraz zaoberali, musia mať definovaný kľúč. To neplatí o typoch slabých entít. Preto v prípade, že chceme povedať, že nejde o typ slabej entity, môžeme použiť výraz **typ regulárnej entity** alebo **typ silnej entity** (angl.: regular entity type alebo strong entity type).

Slabá entita (angl.: weak entity) je entita, ktorú nie je možné jednoznačne identifikovať len pomocou hodnôt jej atribútov. Pre jej identifikáciu je potrebné poznať nie len hodnotu (hodnoty) niektorého jej atribútu (niektorých jej atribútov), ale aj silnú entitu (jej kľúč) s ktorou je vo vzťahu. Vzťah s touto **identifikačnou entitou** (angl.: identifying entity)[8] sa nazýva **identifikačný vzťah** (angl.: identifying relationship). Slabá entita musí byť v identifikačnom vzťahu so silnou entitou, pretože bez neho nemôže byť identifikovaná a bez identifikačnej entity ani nemôže existovať. Prípadne môže byť slabá entita v identifikačnom vzťahu s inou slabou entitou, ktorá je v identifikačnom vzťahu so silnou entitou, atď.

Slabé entity toho istého typu sú reprezentované **typom slabej entity**[9] (angl.: weak entity type).



Obrázok 2.34: Slabá entita, čiastočný kľúč

8 Ďalšie používané názvy pre tento druh entity v angličtine sú owner entity type, parent entity type, alebo dominant entity type.

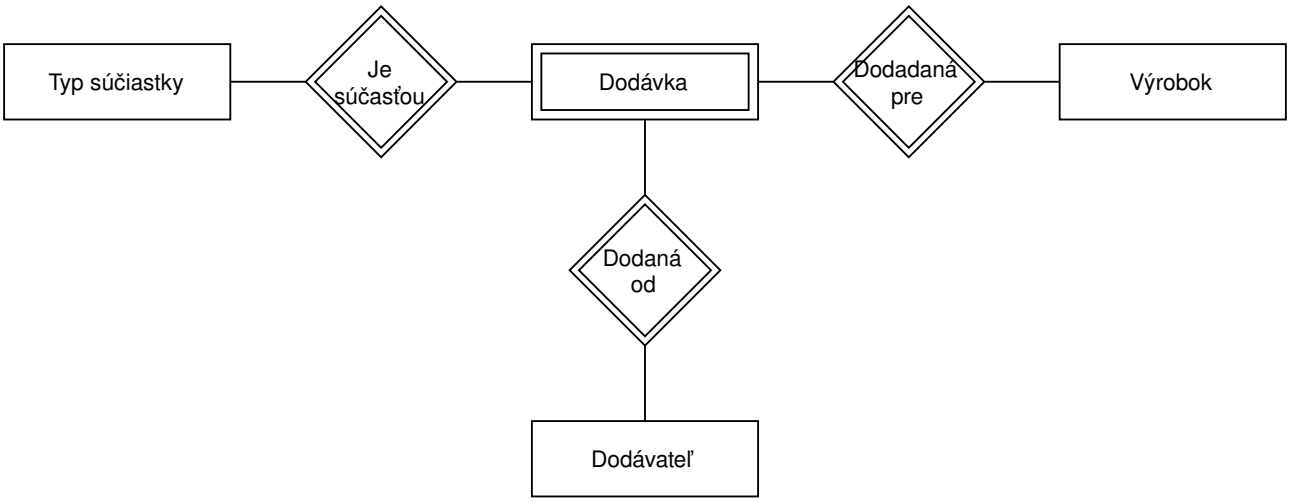
9 Ďalšie názvy používane v angličtine sú child entity type alebo subordinate entity type

Predpokladajme, že v databáze evidujeme budovy a miestnosti v nich (obr. č. 2.34). Každá budova má svoj názov. Miestnosti v budove rozlišujeme podľa ich číselného označenia. Miestnosti v rôznych budovách môžu mať to isté číslo. Preto pre jednoznačnú identifikáciu miestnosti potrebujeme poznať nie len číslo miestnosti, ale aj názov budovy v ktorej sa nachádza. Miestnosť je reprezentovaná typom slabej entity.

Spôsob návrhu databázy často určuje, či je entita slabá alebo silná. Ak by každá miestnosť v databáze mala svoje „globálne“ jedinečné číslo (ani miestnosť v inej budove by nemohla mať to isté číslo), tak by miestnosť bola reprezentovaná typom silnej entity.

Čiastočný kľúč (angl.: partial key)[10] je atribút, pomocou ktorého dokážeme navzájom rozlíšiť slabé entity, ktoré sú vo vzťahu s tou istou identifikačnou entitou. Čiastočný kľúč označujeme podčiarknutím jeho názvu prerušovanou čiarou. Na obr. č. 2.34 je čiastočným kľúčom číslo miestnosti.

Kľúč typu slabej entity teda pozostáva z čiastočného kľúča slabej entity a kľúča identifikačnej entity.



Obrázok 2.35: Typ slabej entity identifikovaný viacerými typmi identifikačných entít (súčasne)

Slabá entita môže byť identifikovaná aj viac ako jednou identifikačnou entitou. Napríklad dodávka súčiastky na výrobu výrobku, môže byť identifikovaná výrobkom (do ktorého bola použitá), typom súčiastky (ktorá bola dodaná) a dodávateľom (obr. č. 2.35).

2.2 Rozšírený entitno-relačný model

ER model poskytuje základný nástroj pre definovanie dátového modelu na konceptuálnej úrovni. Pri návrhu databáz je často potrebné zachytiť podmienky, ktoré v ER modely nie je možné špecifikovať. Preto k základnej verzii ER modelu bolo neskôr doplnených viacero rozšírení, ktoré umožňujú presnejšiu definíciu modelovaného systému. Tento model sa nazýva **rozšírený entitno-relačný model** (angl.: Enhanced Entity-Relationship Model)[11]. Budeme preň používať skratku

10 Ďalším používaným názvom v angličtine je discriminator.

11 Existujú aj označenia Extended Entity-Relationship Model (skratka: EER model) a Expanded Entity-Relationship Model (skratka: PERM).

odvodenú z anglického názvu: EER model, alebo EERM. Graficky sa tento typ modelu zapisuje pomocou diagramu, ktorý nazývaného **rozšírený entitno-relačný diagram** (angl.: *Enhanced Entity-Relationship Diagram*). Podobne používame skratku z anglického názvu: EER diagram, alebo EERD.

Medzi rozšírenia patria:

- Generalizácia a špecializácia. Rozšírenie je podobné nadtriedam a podtriedam v objektovo-orientovanom programovaní, ale je všeobecnejšie.
  - Príklad: nech `A` je nadtyp (nadtrieda), nech `B` a `C` sú jeho podtypy (podtriedy).
    - V objektovo-orientovanom programovaní nemôže byť jedna entita (objekt) typu `B` aj typu `C`. V EER modely môžeme určiť, či typ tej istej entity (objektu) môže byť len `B` alebo len `C`, alebo môže byť `B` aj `C`.
    - V EER modely môžeme určiť, či entita (objekt) typu A musí byť aj typu B alebo C, alebo nemusí byť.
  - Podtyp (podtrieda) môže mať viacero priamych nadtypov (nadtried).
  - S generalizáciou a špecializáciou súvisí aj podmienená definícia hodnôt atribútov v podtypoch (podtriedach).
- Union (kategória) definuje typ entity, ktorý je podtypom viacerých nesúvisiacich typov. Inštancia podtypu ale obsahuje atribúty len jedného nadtypu.
- Výlučné vzťahy definujú pre typ entity skupinu (skupiny) typov vzťahov a každá entita daného typu môže byť len v jednom type vzťahu z definovanej skupiny. Napríklad v databáze reprezentujúcej predaj lístkov môžeme výlučnými vzťahmi definovať obmedzenie, že lístok bol vydaný automatom, alebo bol predaný cez internet (evidujeme viacero internetových portálov), ale nemohol byť predaný aj automatom aj cez internet. Entita typu lístok môže mať vzťah reprezentujúci predajcu len s entitou reprezentujúcou automat, alebo len s entitou reprezentujúcou webový portál.
- Neprenosnosť vzťahu zabraňuje zmene vzťahu entity s inou entitou. Napríklad v databáze emailového servera, môžeme pomocou neprenosnosti vzťahu definovať obmedzenie, že email nemôže zmeniť svojho odosielateľa.

TODO

3 Relačné databázy

Teória relačných databáz je založená na relačnej algebre. Databáza je podľa tejto teórie reprezentovaná ako kolekcia matematických relácii. Pre jednoduchosť môžeme reláciu reprezentovať tabuľkou, a teda relačnú databázu ako kolekciu tabuliek.

Zamestnanci

meno: Andrej Bača
adresa: Bratislava, Ovčiarska 1
občiansky preukaz: BA000001
profesia: programátor
vodičské oprávnenie skupiny B: áno

meno: Juraj Cibuľka
adresa: Trnava, Bratislavská 2
občiansky preukaz: CI000002
profesia: správca počítačovej siete
vodičské oprávnenie skupiny B: nie

meno: Peter Hruška
adresa: Bratislava, Obrancov 3
občiansky preukaz: HR000003
profesia: manažér
vodičské oprávnenie skupiny B: áno
podriadený: Bača, Cibuľka, Husár

meno: Branislav Husár
adresa: Nitra, Jazdecká 4
občiansky preukaz: HU000004
profesia: analytik
vodičské oprávnenie skupiny B: nie

meno: Viliam Bosý
adresa: Bratislava, Obrancov 5
občiansky preukaz: BO000005
profesia: manažér
vodičské oprávnenie skupiny B: áno
priamy podriadený: Hruška, Langoš

meno: Ján Biľko
adresa: Nitra, Langova 6
občiansky preukaz: BI000006
profesia: analytik
vodičské oprávnenie skupiny B: nie

meno: Matej Langoš
adresa: Bratislava, Langova 7
občiansky preukaz: LA000007
profesia: manažér
vodičské oprávnenie skupiny B: áno
podriadený: Biľko, Medveď, Sokol, Vlk

meno: Pavol Medveď
adresa: Bratislava, Jelenia 8
občiansky preukaz: ME000008
profesia: architekt
vodičské oprávnenie skupiny B: nie

meno: Michal Sokol
adresa: Bratislava, Letecká 9
občiansky preukaz: SO000009
profesia: programátor
vodičské oprávnenie skupiny B: áno

meno: Levoslav Vlk
adresa: Trnava, Divoká 10
občiansky preukaz: VL000010
profesia: programátor
vodičské oprávnenie skupiny B: nie

Projekty

názov: PC shop
popis: web aplikácia na predaj PC
zamestnanci:
Hruška (vedúci 40hod/týž),
Bača (40h/týž),
Cibuľka (40h/týž),
Husár (40h/týž)

názov: Stellar
popis: spracovanie údajov z družíc
zamestnanci:
Bosý (vedúci 20h/týž),
Biľko (20h/týž),
Langoš (20h/týž),
Medveď (20h/týž),
Sokol (40h/týž)

názov: AIS
popis: akademický informačný systém
zamestnanci:
Bosý (vedúci 20h/týž),
Biľko (20h/týž),
Langoš (20h/týž),
Medveď (20h/týž),
Vlk (40h/týž)

Obrázok 3.1: Kartotéka s údajmi o firme

Zamestnanci

Evidenčné číslo	Meno	Priezvisko	Adresa	Občiansky preukaz	Nadriadený	Profesia	Vodičské oprávnenie
101	Andrej	Bača	Bratislava, Ovčiarska 1	BA000001	103	103	áno
102	Juraj	Cibuľka	Trnava, Bratislavska 2	CI000002	103	104	nie
103	Peter	Hruška	Bratislava, Obrahcov 3	HR000003	105	101	áno
104	Branislav	Husár	Nitra, Jazdecká 4	HU000004	103	107	nie
105	Viliam	Bosý	Bratislava, Obrancov 5	BO000005	NULL	101	áno
106	Ján	Biľko	Nitra, Langova 6	BI000006	107	107	nie
107	Matej	Langoš	Bratislava, Langova 7	LA000007	105	101	áno
108	Pavol	Medveď	Bratislava, Jelenia 8	ME000008	107	105	nie
109	Michal	Sokol	Bratislava, Letecká 9	SO000009	107	103	áno
110	Levoslav	Vlk	Trnava, Divoká 10	VL000010	107	103	nie

Projekty

Evidenčné číslo	Názov	Popis	Vedúci
101	PC shop	web aplikácia na predaj PC	103
102	Stellar	spracovanie údajov z družíc	105
103	AIS	akademický informačný systém	105

Zaradenie na projekt

Zamestnanec	Projekt	Hodiny za týždeň
101	101	40
102	101	40
103	101	40
104	101	40
105	102	20
106	102	20
107	102	20
108	102	10
109	102	40
105	103	20
106	103	20
107	103	20
108	103	30
110	103	40

Profesie

Evidenčné číslo	Názov
101	Manažér
102	Asistent
103	Programátor
104	Správca počítačovej siete
105	Architekt
106	DevOps
107	Analytik
108	Grafik

Obrázok 3.2: Obsah tabuliek relačnej databázy firmy

Na obr. č. 3.1 je znázornený obsah časti kartotéky firmy, s informáciami o zamestnancoch, ich hierarchii a projektoch na ktorých pracujú. Obsah relačnej databázy s týmito údajmi je na obr. č. 3.2

. Tabuľky obsahujú informácie o entitách a vzťahoch (z ER modelu). Každá tabuľka obsahuje informácie o všetkých entitách toho istého typu, alebo o všetkých vzťahoch toho istého typu. Môže obsahovať aj kombináciu informácii o entitách toho istého typu a niektorých vzťahoch viacerých vybraných typov (tabuľka `Zamestnanec` obsahuje aj vzťah zamestnanca s jeho profesiou a vzťah s jeho nadriadeným). Stĺpce tabuľky definujú, aké informácie sú v databáze o entitách alebo vzťahoch daného typu evidované. Každý riadok reprezentuje záznam o jednej entite, alebo o jednom vzťahu, alebo o kombinácii informácii o jednej entite a niektorých jej vzťahoch. Informácie v rôznych tabuľkách sú prepojené prostredníctvom hodnôt v stĺpcoch, obsahujúcich cudzie kľúče (časť 3.1.5). Napríklad v tabuľke zamestnancov, každý riadok obsahuje evidenčné číslo profesie príslušného zamestnanca. Informácie o profesii sú v inej tabuľke.

3.1 Definícia relačnej databázy

V tejto časti definujeme základné pojmy používané v relačných databázach. Niektoré s nich sme už používali v časti venujúcej sa ER modelu, tu ich zopakujeme v kontexte relačných databáz.

3.1.1 Integrita databázy

Integrita údajov (angl.: data integrity) v databáze je stav, kedy jednotlivé časti tvoria zmysluplný celok (databázu). Pri návrhu databázy špecifikujeme pravidlá, ktoré musia byť dodržané. Zabezpečujú správnosť obsahu databázy a kompatibilitu medzi jej jednotlivými časťami. Sú odvodené z pravidiel reálneho sveta, ktorého časť je mapovaná databázou. Tieto pravidla môžeme rozdeliť na pravidlá pre:

- stav obsahu databázy (angl.: static constraints alebo state constraints), ktoré môžeme rozdeliť na pravidlá pre:
  - integritu entity (angl.: entity integrity constraints)
  - referenčnú integritu (angl.: referential integrity constrainsts)
  - integritu definičného oboru hodnôt (ang.: domain integrity constraints)
  - sémantickú integritu (angl: semantic integrity constraints)
- zmenu stavu (obsahu) databázy (angl.: transition constraints alebo dynamic constraints)

Pravidlami pre integritu sa budeme postupne zaoberať v ďalších častiach.

3.1.2 Tabuľka

Pod pojmom **atomická hodnota** budeme rozumieť nedeliteľná v rámci dátového modelu, aj keď v reálnom svete môže byť hodnota deliteľná. Napríklad názov mesta „Banská Bystrica“ možno rozdeliť na slovná, slabiky alebo písmená. Ak ale názov mesta označíme ako atomickú hodnotu, tak to znamená, že v rámci štandardných operácii v relačnej databáze je nedeliteľná a s názvom mesta vždy pracujeme ako s celkom. Ďalším príkladom je celé meno osoby, skladajúce sa z krstného mena, priezviska a titulov. Ak celé meno osoby označíme ako atomickú hodnotu, tak s ním v databáze budeme vždy pracovať ako s celkom. Ak potrebujem zvlášť pracovať s časťami celého mena, tak ako atomické označíme jeho časti: krstné meno, priezvisko a titul, nie celé meno.

Usporiadaný zoznam, obsahujúci konečný počet prvkov *n* nazývame ***n*-tica** (výslovnosť „entica“) (angl.: n-tuple). Zoznam prvkov n-tice zapisujeme v okrúhlych zátvorkách, napr.: (1, 3, 2). V *n*-tici záleží na poradí prvkov. Napríklad (1, 2, 3) je iná *n*-tica (trojica) ako (3, 2, 1). V *n*-tici sa môžu

opakovať prvky s rovnakou hodnotou, napríklad nasledujúca pätica obsahuje 5 prvkov, ale niektoré sa opakujú (1, 2, 1, 3, 1).

Množina (angl.: set) je presne definovaný súbor rôznych prvkov. Množina sa zvykne označovať veľkými písmenami. Zoznam jej prvkov zapisujeme v zložených zátvorkách, napr.: {1, 2, 3}. V množine nezáleží na poradí zapísania jej prvkov, pretože poradie jej prvkov nie je definované. Napríklad {1, 2, 3} je tá istá množina ako {3, 2, 1}. V množine sa nenachádzajú prvky s rovnakými hodnotami.

Členstvo prvku v množine označujeme znakom $\in$. Ak prvok nepatrí do množiny, tak používame znak $\notin$. Príklady: $1 \in \{1, 2, 3\}$, $2 \in \{1, 2, 3\}$, $3 \in \{1, 2, 3\}$, $4 \notin \{1, 2, 3\}$, $5 \notin \{1, 2, 3\}$.

Množina *A* je **podmnožinou** (angl.: subset) množiny *B*, ak každý prvok množiny *A* patrí aj do množiny *B*. Vzťah podmnožiny zapisujeme znakom $\subseteq$. Napríklad $A \subseteq B$ znamená, že *A* je podmnožinou *B*.

Množina *A* je **vlastná podmnožina množiny** (angl.: proper subset) *B*, ak *A* je podmnožinou *B*, ale v množine *B* existuje aspoň jeden prvok, ktorý nepatrí do množiny *A*. Vzťah vlastnej podmnožiny zapisujeme znakom $\subset$. Napríklad $A \subset B$ znamená, že *A* je vlastnou podmnožinou množiny *B*.

Prienik množín *A* a *B* (angl.: set intersection) je množina, do ktorej patria prvky, ktoré sú v množine *A* aj množine *B*. Prienik množín označujeme $A \cap B$.

Napríklad ak $A = \{1, 2, 3, 4\}$, $B = \{3, 4, 5\}$, tak $A \cap B = \{3, 4\}$.

Zjednotenie množín *A* a *B* (angl.: union of sets) je množina prvkov, ktoré patria do *A*, alebo do *B*, alebo do *A* aj *B*. Zjednotenie množín označujeme $A \cup B$.

Napríklad ak $A = \{1, 2, 3, 4\}$, $B = \{3, 4, 5\}$, tak $A \cup B = \{1, 2, 3, 4, 5\}$.

Rozdiel množín *A* a *B* (angl.: set difference) je množina, obsahujúca tie prvky z *A*, ktoré nepatria aj do *B*. Rozdiel množín označujeme *A*-*B* alebo $A \setminus B$.

Napríklad ak $A = \{1, 2, 3, 4\}$, $B = \{3, 4, 5\}$, tak $A - B = \{1, 2\}$.

Nech M_1, M_2, …, M_n sú množiny. **Karteziánsky súčin** (angl.: cartesian product) týchto množín je množinou všetkých usporiadaných *n*-tíc $(v_1, v_2, \ldots. v_n)$, kde $v_i \in M_i$, pre $1 \le i \le n$. Karteziánsky súčet označujeme $M_1 \times M_2 \times ... \times M_n$.
Matematický zápis: $M_1 \times M_2 \times ... \times M_n = \{(v_1, v_2, \ldots. v_n) \mid v_1 \in M_1, v_2 \in M_2, \ldots v_n \in M_n\}$

Príklad: Nech $M_1 = \{101,102,103\}$, $M_2 = \{\text{Andrej, Juraj}\}$, $M_3 = \{\text{Bača, Cibuľka, Hruška, Husár}\}$, potom karteziánsky súčin $M_1 \times M_2 \times M_3$ je množina n-tíc:

{ (101, Andrej, Bača), (101, Andrej, Cibuľka), (101, Andrej, Hruška), (101, Andrej, Husár),
(101, Juraj, Bača), (101, Juraj, Cibuľka), (101, Juraj, Hruška), (101, Juraj, Husár),
(102, Andrej, Bača), (102, Andrej, Cibuľka), (102, Andrej, Hruška), (102, Andrej, Husár),
(102, Juraj, Bača), (102, Juraj, Cibuľka), (102, Juraj, Hruška), (102, Juraj, Husár) }

Ak by sme množinu M_1 definovali ako množinu prirodzených čísiel, do M_2 by patrili všetky existujúce krstné mená, do M_3 všetky existujúce priezviská, tak by n-tice v $M_1 \times M_2 \times M_3$ definovali všetky platné kombinácie, ktoré sa môžu vyskytnúť v riadkoch tabuľky evidujúcej evidenčné čísla, mená a priezviská zamestnancov. Reálna tabuľka by pravdepodobne obsahovala iba niektoré z týchto riadkov. Preto môžeme karteziánsky súčin využiť na formálnu definíciu možného obsahu tabuľky databázy a podmnožinu karteziánskeho súčinu na definíciu obsahu tabuľky v určitom časovom okamihu (keďže sa obsah tabuľky môže meniť).

Matematická **relácia** (angl.: relation) nad množinami M_1, M_2, …, M_n je podmnožina karteziánskeho súčinu $M_1 \times M_2 \times ... \times M_n$.

Tabuľku v databáze formálne definujeme pomocou schémy relácie. **Schéma relácie** (angl.: relation schema[12]) $R(A_1, A_2, \ldots, A_n)$ je tvorená názvom R a zoznamom atribútov $A_1, A_2, \ldots, A_n$. Názov R predstavuje názov tabuľky, atribúty $A_1, A_2, \ldots, A_n$ reprezentujú názvy a význam stĺpcov tabuľky. Každému atribútu musíme priradiť aj definičný obor hodnôt. **Definičný obor hodnôt** (angl.: domain) atribútu je množina atomických hodnôt, ktoré sa v príslušnom stĺpci môžu vyskytovať. Označujeme ho $dom(A_i)$ pre $1 \leq i \leq n$ (*dom* je označenie odvodené z anglického pojmu domain) Počet atribútov v schéme relácie n nazývame **stupeň schémy relácie**.

Definičným oborom hodnôt atribútu môže byť napríklad: množina krstných mien, priezvisk, názvov krajín, celých čísiel, celých čísiel v určitom rozsahu, textových reťazcov, možných známok na vysokej škole (A, B, C, D, E, FX, FN), atď. Z implementačných dôvodov, definičný obor hodnôt často špecifikujeme pomocou niektorého štandardného dátového typu poskytovaného implementáciou databázového systému. Napríklad definičný obor atribútu obsahujúceho priezviská, môžeme nahradiť množinou obsahujúcou všetky textové reťazce (ich veľkosť môže byť obmedzená implementáciou databázy). Hodnoty štandardných dátových typov môžeme obmedziť definíciou dodatočných podmienok. Napríklad emailová adresa je textový reťazec, pre ktorý definujeme ďalšie podmienky (obmedzenia) pre jeho formát. Teplota v kelvinoch je nezáporné číslo, teplota v stupňoch celzia je číslo s minimálnou hodnotou −273.15.

V jednej schéme relácie môže mať viacero atribútov ten istý definičný obor hodnôt. Názov atribútu ale napovedá, aká je interpretácia hodnôt. Napríklad v tabuľke s údajmi o projekte, môže jeden stĺpec obsahovať dátum začiatku projektu, ďalší stĺpec dátum konca projektu. Oba tieto stĺpce majú ten istý definičný obor hodnôt, ale líšia sa názvami a interpretáciou hodnôt v nich.

Relácia alebo **relácia stavu** alebo **inštancia relácie** r (angl.: relation alebo relation state[13] alebo relation instance) schémy relácie $R(A_1, A_2, \ldots, A_n)$ je podmnožina karteziánskeho súčinu $dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n)$. Označujeme ju $r(R)$. Reprezentuje obsah tabuľky v určitom časovom okamihu. Každá jej n-tica reprezentuje obsah jedného riadku tabuľky.

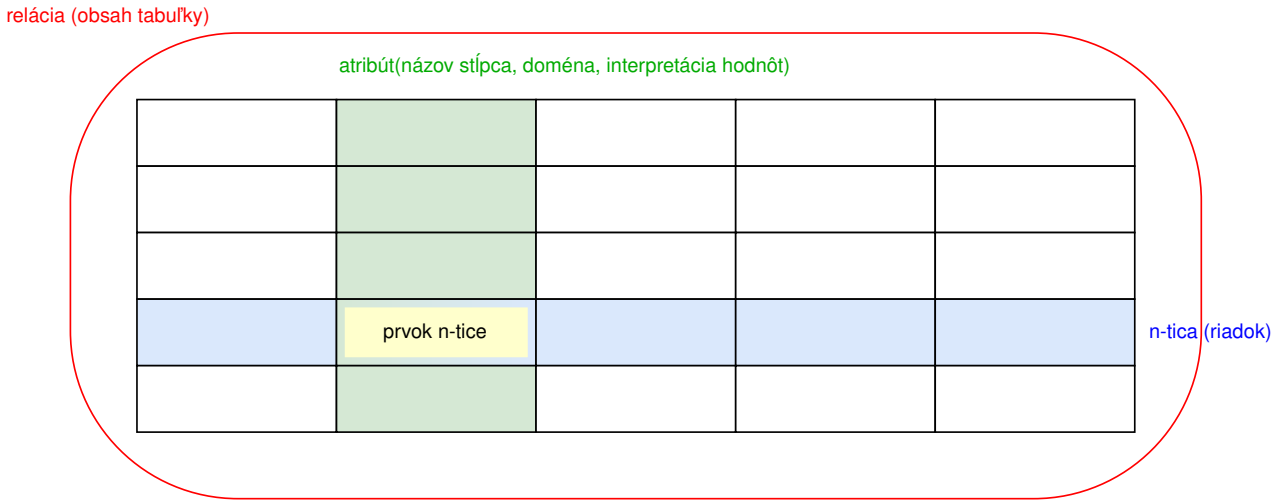
Nech n-tica $t = (h_1, h_2, \ldots, h_n)$ je prvkom relácie $r(R)$. Hodnotu h_i, ktorá zodpovedá atribútu A_i, označujeme $t[A_i]$ alebo $t.A_i$. Neformálne možeme povedať, že i-tu hodnotu v riadku t a v stĺpci definovanom atribútom A_i označujeme $t[A_i]$ alebo $t.A_i$. Podobne zápis $t[A_{j1}, A_{j2}, \ldots A_{jk}]$ alebo $t.(A_{j1}, A_{j2}, \ldots A_{jk})$, kde $\{A_{j1}, A_{j2}, \ldots A_{jk}\} \in \{A_1, A_2, \ldots, A_n\}$, označuje hodnoty $(h_{j1}, h_{j2}, \ldots, h_{jk})$ z riadku t.

12 Ďalší pojem používaný v angličtine je relation intension
13 Ďalší pojem používaný v angličtine je relation extension

Pravidlo určujúce, že $h_i \in dom(A_i)$ sa nazýva **pravidlo integrity definičného oboru** (angl.: domain integrity constraints).

Množina nemôže obsahovať rovnaké prvky. Preto ani tabuľka nemôže obsahovať riadky s rovnakou kombináciou hodnôt. Má to aj praktický význam, pretože každý riadok reprezentuje jednu entity alebo vzťah z reálneho sveta. Podľa hodnôt v riadku tabuľky musíme vedieť identifikovať entitu alebo vzťah v reálnom svete. Ak by dva riadky obsahovali rovnaké hodnoty, tak by sme nevedeli rozlíšiť medzi entitami alebo vzťahmi v reálnom svete. Príklad: Dvaja zamestnanci môžu mať rovnaké meno aj priezvisko, preto sme v príklade na obr. č. 3.2 do tabuľky zamestnancov pridali evidenčné číslo. Každý zamestnanec má iné evidenčné číslo, čo nám pomôže rozlíšiť zamestnancov aj v prípade, že majú rovnaké mená a priezviská.

V množine nie je definované poradie prvkov. Preto relácia nedefinuje poradie riadkov v tabuľke. Reprezentuje obsah tabuľky na abstraktnej úrovni. Preto ak zmeníme poradie riadkov v tabuľke, tak táto tabuľka bude reprezentovaná tou istou reláciou. Na implementačnej úrovni sa pravdaže poradie riadkov prejaví.



Obrázok 3.3: Teoretická definícia tabuľky v relačnej databáze

Na obr. č. 3.2 je databáza obsahujúca nasledujúce schémy relácií:

- `Zamestnanci`(`Evidenčné číslo`, `Meno`, `Priezvisko`, `Adresa`, `Občiansky preukaz`, `Nadriadený`, `Profesia`, `Vodičské oprávnenie`),
- `Projekty`(`Evidenčné číslo`, `Názov`, `Popis`, `Vedúci`),
- `Profesie`(`Evidenčné číslo`, `Názov`),
- `Zaradenie na projekt`(`Zamestnanec`, `Projekt`, `Hodiny za týždeň`).

Definičné obory hodnôt atribútov v schéme relácie `Zamestnanci` sú:

- *dom*(`Evidenčné číslo`) je množina celých kladných čísiel,
- *dom*(`Meno`) je množina textových reťazcov[14],
- *dom*(`Priezvisko`) je množina textových reťazcov,

14 V závislosti od implementácie databázy môže ale nemusí byť vhodné obmedziť maximálnu dĺžku textového reťazca.

- *dom*(`Adresa`) je množina textových reťazcov,
- *dom*(`Občiansky preukaz`) je množina textových reťazcov obsahujúcich 2 písmená a 6 číslic,
- *dom*(`Vedúci`) je množina celých kladných čísiel,
- *dom*(`Profesia`) je množina celých kladných čísiel,
- *dom*(`Vodičské oprávnenie`) je dvojprvková množina {áno, nie}[15].

Stupeň schémy relácie `Zamestnanci` je 8.

Reláciu `Projekty` môžeme zapísať množinou *n*-tíc
{ (101, „PC shop“, „web aplikácia na predaj PC“, 103),
(102, „Stellar“, „spracovanie údajov z družíc“, 105),
(103, „AIS“, „akademický informačný systém“, 105) }

Nech $t \in$ `Projekty` a *t* = (102, „Stellar“, „spracovanie údajov z družíc“, 105), potom
t[`Názov`] = („Stellar“),
t[`Evidenčné číslo`, `Názov`, `Vedúci`] = (102, „Stellar“, 105).

Každý riadok v tabuľke `Zamestnanci` je *n*-ticou relácie jej schémy, ktorá reprezentuje jedného zamestnanca z reálneho sveta. Prvý prvok *n*-tice obsahuje evidenčné číslo zamestnanca, druhý prvok krstné meno, atď. Podľa definície považujeme adresu zamestnanca za atomickú, to znamená, že vždy pracujeme s celou adresou. Stĺpec definovaný atribútom `Profesia` obsahuje evidenčné čísla profesii, ktoré majú jednotlivý zamestnanci. Informácie o profesiách sú v tabuľke `Profesie`. V tomto príklade sú v nej pre jednoduchosť len názvy profesii. Takmer každý zamestnanec, má v tabuľke `Zamestnanci`, uvedené evidenčné číslo jeho priameho nadriadeného (v stĺpci definovanom atribútom `Vedúci`). Jeden zo zamestnancov má ale v stĺpci `Vedúci` uvedenú špeciálnu hodnotu NULL, ktorá znamená, že tento zamestnanec nemá svojho nadriadeného (je najvyššie v hierarchii), alebo jeho nadriadený nie je určený.

Hodnota **NULL** sa používa v prípadoch, keď je hodnota neznáma, alebo keď hodnota neexistuje, alebo nemá zmysel. Hodnota NULL má špeciálne spracovanie v aritmetických operáciach a pri porovnávaní. Napríklad ak by dvaja zamestnanci mali adresu NULL, neznamenalo by to, že majú rovnakú adresu.

Pri definovaní pravidiel v databáze môžeme určiť, či stĺpec môže obsahovať hodnotu NULL.

Tabuľka `Zaradenie na projekt` slúži na priradenie zamestnancov na projekty. Každý riadok obsahuje priradenie jedného zamestnanca na jeden projekt prostredníctvom evidenčných čísiel zamestnancov a projektov. Ak je jeden zamestnanec priradený na viac projektov, tak sa jeho evidenčné číslo nachádza na viacerých riadkoch. Podobne je riešené aj to, keď na jednom projekte pracuje viacero zamestnancov. Každé priradenie obsahuje informáciu aj o počte hodín, ktoré má zamestnanec na projekte týždenne odpracovať.

15 Množina obsahuje dve hodnoty s významom pridelenia alebo nepridelenia vodičského oprávnenia (skupiny B). Tieto dve hodnoty nie sú textové reťazce, takáto ich implementácia by nebola efektívna.

3.1.3 Primárny kľúč

Množina atribútov schémy relácie, pre ktorú platí, že dve *n*-tice nemôžu mať rovnakú kombináciu hodnôt pre tieto atribúty sa nazýva **superkľúč** (angl.: superkey). Neformálne môžeme povedať, že supekľúčom je množina stĺpcov tabuľky, pre ktorú platí, že dva riadky v týchto stĺpcoch nemôžu mať rovnakú kombináciu hodnôt.

Podľa definície, nemôžu dva riadky v jednej tabuľke obsahovať rovnaké kombinácie hodnôt, preto množina všetkých stĺpcov jednej tabuľky tiež tvorí superkľúč. Zvyčajne sa dá nájsť podmnožina stĺpcov tabuľky, pre ktorú platí, že neexistujú dva riadky, ktoré by mohli mať rovnakú kombináciu hodnôt v týchto stĺpcov. Preto superkľúč môže tvoriť aj táto podmnožina stĺpcov.

Tabuľka môže mať viacero superkľúčov. Množstvo z nich má vlastnosť, že ak zo supekľúča odstránime niektorý atribút (stĺpec), tak tento zmenšený superkľúč bude stále superkľúč. Superkľúč, ktorý by odstránením atribútu prestal byť superkľúčom nazývame **kľúč** (angl.: key). Je to minimálny superkľúč.

Hodnota kľúča môže byť použitá pre identifikáciu riadku tabuľky. V tabuľke `Zamestnanci` na obr. č. 3.2 sú dva kľúče: `Evidenčné číslo` a `Občianský preukaz`. V tabuľke `Zaradenie na projekt` je jeden kľúč tvorený dvoma atribútmi: `Zamestnanec` a `Projekt`. Kombinácia týchto dvoch atribútov jednoznačne identifikuje riadok priradzujúci zamestnanca na projekt. Pri odstránení niektorého z atribútov, by zostávajúci atribút už netvoril kľúč.

Každý kľúč nazývame aj **kandidát na kľúč** (angl.: candidate key). Počas návrhu databázy vyberieme pre každú tabuľku (reláčnú schému) jeden z kandidátov na kľúč a označíme ho ako primárny kľúč. **Primárny kľúč** (angl.: primary key) je jeden z kandidátov na kľúč, ktorý sa zvykne používa pre identifikáciu riadku v tabuľke (*n*-tice v relácii). Primárny kľúč označujeme napríklad podčiarknutím, ale zaužívané sú aj iné spôsoby označenia, napríklad skratkou PK alebo hrubým písmom. Kandidát na kľúč, ktorý nebol vybraný za primárny kľúč, sa nazýva **sekundárny kľúč** (angl.: secondary key).

V tabuľke `Zamestnanci` na obr. č. 3.2 by sme `Evidenčné číslo` mohli teoreticky považovať za zbytočný stĺpec, pretože zamestnanca môžeme identifikovať aj podľa občianskeho preukazu. Číslo občianskeho preukazu sa ale môže meniť. Preto sú v tabuľke evidované evidenčné čísla zamestnancov a tieto sme zvolili za primárny kľúč. Ak by sme za primárny kľúč zvolili číslo občianskeho preukazu, tak by napríklad zamestnanci mali v stĺpci `Nadriadený` číslo občianskeho preukazu priameho nadriadeného. Takže ak by sa zmenilo číslo občianskeho preukazu zamestnanca, ktorý má podriadených, tak by sme museli zmeniť údaje o občianskom preukaze aj v riadkoch jeho priamych podriadených. Toto by bolo nie len implementačne menej efektívne riešenie, ale hlavne by sa zvýšilo riziko narušenia integrity údajov v prípade chyby programátora. Z rovnakých dôvodov (predpokladáme, že každá profesia má jedinečný názov) sme za primárny kľúč v tabuľke `Profesie` zvolili teoreticky nadbytočný stĺpec `Evidenčné číslo`. Zmena názvu profesie je menej riziková vzhľadom na narušenie integrity údajov a implementačne efektívnejšia. Naviac je porovnávanie čísiel pri hľadaní podľa kľúča implementačne efektívnejšie ako porovnávanie textových reťazcov.

Z definície kľúča vyplýva obmedzenie na obsah hodnoty NULL. Napríklad ak sa kľúč skladá len z jedného atribútu, tak žiadny riadok nesmie obsahovať v príslušnom stĺpci hodnotu NULL

(teoreticky maximálne jeden), pretože podľa hodnôt v tomto stĺpci by sme nevedeli identifikovať riadok.

Pravidlo pre integritu entity (angl.: entity integrity constraint) uvádza, že každá tabuľka musí mať primárny kľúč a tento kľúč nesmie obsahovať hodnotu NULL.

3.1.4 Databáza

Schéma relačnej databázy $D(S,I)$ (angl.: relational database schema) je množina schém relácii $S = \{R_1, R_2, \ldots R_m\}$ a množina pravidiel pre integritu údajov I. Predpisuje pravidlá konštrukcie databázy a manipulácie s údajmi.

Stav relačnej databázy alebo **inštancia relačnej databázy** d (angl.: relational database state alebo relational database instance) prislúchajúci schéme relačnej databázy $D(\{R_1, R_2, \ldots R_m\}, I)$ je množina relácii $d = \{r_1, r_2, \ldots, r_m\}$ takých, že r_i je relácia spĺňajúca schému R_i a relácie $r_1, r_2, \ldots, r_m$ spĺňajú množinu pravidiel pre integritu údajov I.

Obrázok č. 3.2 zobrazuje stav relačnej databázy. Množinu schém relácii v tejto databáze môžeme formálne zapísať `Firma` = {`Zamestnanci`, `Projekty`, `Profesie`, `Zaradenie na projekt`}.

Ak stav databázy nespĺňa pravidlá pre integritu údajov, tak hovoríme, že stav databázy je **neplatný** (angl.: invalid state). Ak stav databázy spĺňa pravidlá pre integritu údajov, tak hovoríme, že stav databázy je **platný** (angl.: valid state). Toto ale nie je presné vyjadrenie, pretože nie je v súlade s definíciou stavu relačnej databázy!

3.1.5 Referenčná integrita

Riadky v rôznych tabuľkách môžu súvisieť, tak ako to je aj na obr. č. 3.2, kde každý riadok tabuľky `Zamestnanec` obsahuje evidenčné číslo iného zamestnanca (evidovaného v tej istej tabuľke) a evidenčné číslo profesie (evidovanej v inej tabuľke).

Pravidlá referenčnej integrity (angl.: referential integrity constraint) zabezpečujú konzistenciu údajov medzi tabuľkami, alebo riadkami tej istej tabuľky. Údaje sú medzi dvomi tabuľkami, alebo dvomi riadkami v jednej tabuľke, prepojené pomocou cudzích kľúčov. Cudzí kľúč slúži na identifikáciu iného riadok v inej alebo tej istej tabuľke.

Nech R_1 a R_2 sú schémy relácii (môžu byť aj tie isté). Nech R_2 má primárny kľúč PK. Nech n-tice $t_1 \in r_1(R_1)$ a $t_2 \in r_2(R_2)$. **Cudzí kľúč** (angl.: foreign key) je množina atribútov FK relácie R_1, ktorej úlohou je identifikovať n-tice v relácii $r_2(R_2)$ a spĺňa nasledujúce podmienky:

- Atribúty FK majú rovnaké definičné obory hodnôt ako atribúty primárneho kľúča R_2.
- Pre každú n-ticu t_1 platí, jedna z podmienok:
  - cudzí kľúč $t_1[FK]$ má rovnakú hodnotu ako primárny kľúč v niektorej z n-tíc v $r_2(R_2)$ (pre n-ticu t_1 existuje n-tica t_2, pre ktorú platí $t_1[FK] = t_2[PK]$,
  - alebo cudzí kľúč má hodnotu NULL ($t_1[FK]$ = (NULL, NULL, … NULL)).

Cudzí kľúč v R_1 je referenciou (odkazuje) na R_2.

Hodnota cudzieho kľúča nemusí byť v referencujúcej relácii $r_1(R_1)$ jedinečná. Musí byť jedinečná len v referencovanej relácii $r_2(R_2)$, kde je primárnym kľúčom.

Ak hodnoty atribútov patriacich cudziemu kľúču spĺňajú podmienky definície, tak sú pravidlá referenčnej integrity z R_1 do R_2 splnené.

V schéme `Zamestnanci` na obr. č. 3.2 sú dva cudzie kľúče `Nadriadený` a `Profesia` (obidva obsahujú len jeden atribút). Hodnoty v zodpovedajúcich stĺpcoch obsahujú evidenčné čísla iného zamestnanca alebo profesie. Tieto sú primárnymi kľúčmi relácií `Zamestnanci` a `Profesie`.

3.1.6 Sémantická integrita

Pravidlá sémantickej integrity (angl.: semantic integrity constraints) sú všeobecnejšie pravidlá. Príkladom je pravidlo určujúce maximálny týždenný počet hodín odpracovaný zamestnancom na všetkých projektoch spolu na 40 hodín týždenne. Ďalším príkladom môže byť pravidlo určujúce, že podriadený pracovník nesmie mať vyššiu mzdu ako jeho nadriadený.

DBMS zvyčajne obsahuje nástroje pre kontrolu dodržiavania týchto pravidiel. Niekedy však ich dodržiavanie kontroluje aplikácia manipulujúca s databázou.

3.1.7 Transakčné pravidlá integrity

Doteraz sme sa zaoberali **pravidlami pre stav** alebo **statickými pravidlami** (angl.: static constraints alebo state constraints), ktoré určujú pravidlá platnosti stavu databázy (jej obsahu v ľubovolnom okamihu). Ďalším typom pravidiel sú **transakčné pravidlá** alebo **dynamické pravidlá** (angl.: transition constraints alebo dynamic constraints), ktoré určujú podmienky zmeny stavu databázy. Príkladom dynamického pravidla môže byť pravidlo určujúce, že mzda zamestnanca môže len stúpať.

3.2 Diagram schémy relačnej databázy

Model relačnej databázy často zobrazujeme graficky, pomocou diagramu. Diagram zobrazuje množinu schém relácii (tabuliek) a niektoré pravidlá pre integritu údajov.

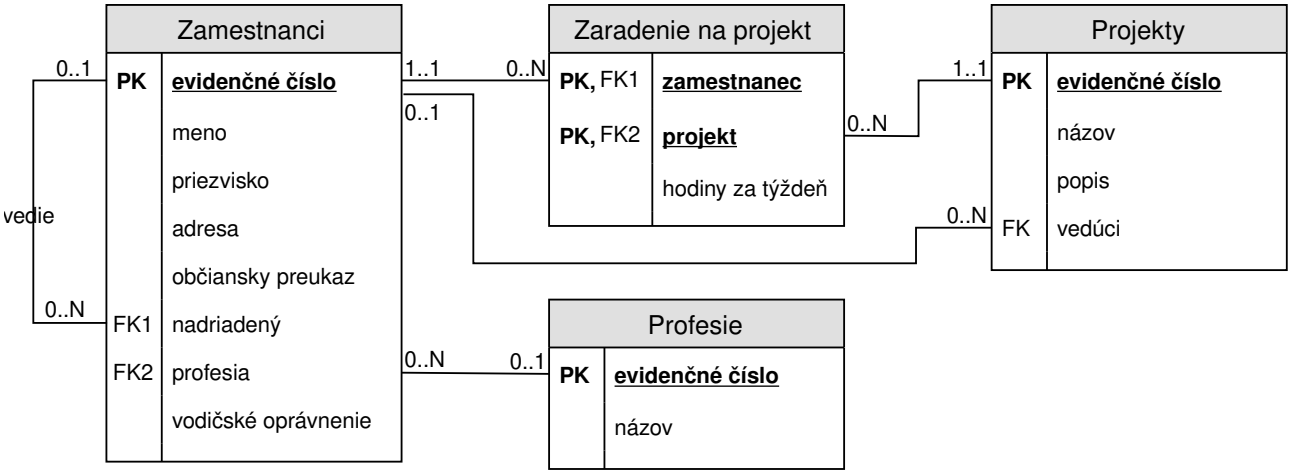
Pre databázu môžeme vytvoriť logický a fyzický model. Logický model je viac abstraktnejší, zahŕňa menej implementačných detailov ako fyzický model. Logický model relačnej databázy zobrazuje informácie, ktoré je možné implementovať v ľubovolnom systéme pre správu relačnej databázy (DBMS). DBMS pre relačnú databázu označujeme skratkou RDBMS (Relational DBMS). Fyzický model obsahuje implementačné detaily, ktoré závisia od konkrétneho použitého RDBMS. RDBMS sú napríklad PostgreSQL, MySQL, Apache Derby, SQLite, H2, Oracle DB, Microsoft SQL Server. Logický model relačnej databázy definuje spoločné črty, ktoré možno implementovať v ľubovoľnom z uvedených systémov, fyzický model je podrobnejší a je prispôsobený špecifickým detailom jedného vybraného systému.

3.2.1 Logický model

Diagram logického modelu databázy na obr. č. 3.2 je zobrazený na obr. č. 3.4. Každá tabuľka (schéma relácie) je zobrazená obdĺžnikom obsahujúcim názov a zoznam atribútov. Primárne kľúče sú označené značkou PK, podčiarknutím alebo hrubým písmom. Cudzie kľúče sú označené značkou

FK. Ak tabuľka obsahuje viacero cudzích kľúčov, tak sú očíslované pre prípad, že sa niektorý z cudzích kľúčov skladá z viacerých atribútov (aby sme vedeli, ktorý atribút je súčasťou ktorého cudzieho kľúča). V našom diagrame ale takýto prípad nie je.

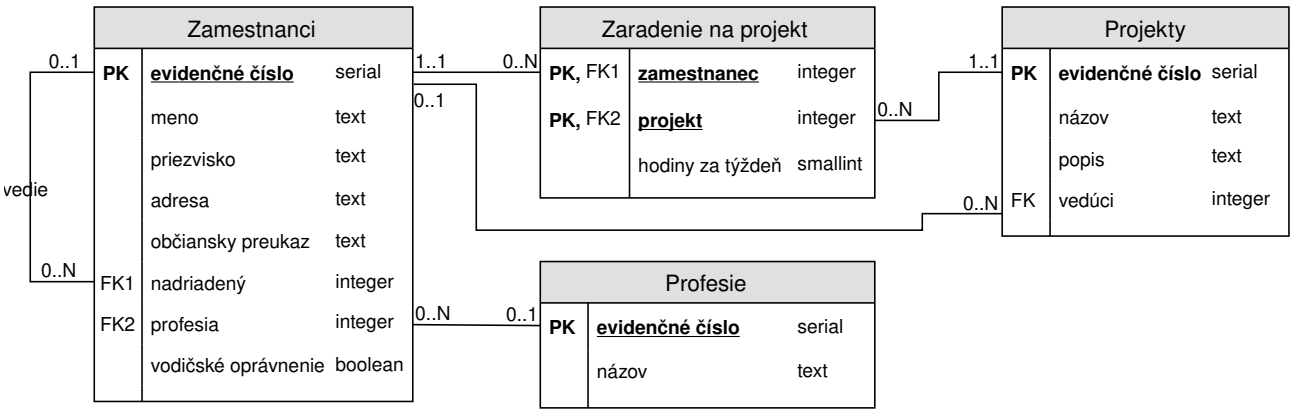
Čiary spájajúce obdĺžniky sa nazývajú vzťahy. Znázorňujú prepojenie tabuliek prostredníctvom cudzích kľúčov (súčasť definície referenčnej integrity). V relačnom modely sú len binárne vzťahy. Ale rovnako ako v konceptuálnom modely môžeme určiť voliteľnosť a násobnosť vzťahu. Ku vzťahu môžeme tiež doplniť jeho názov, alebo role spájaných tabuliek. Pre spôsob zápisu diagramu logického aj fyzického modelu existujú rôzne modifikácie.



Obrázok 3.4: Logický model relačnej databázy

3.2.2 Fyzický model

Fyzický model pridáva k informáciám obsiahnutým v logickom modely implementačné detaily, závisiace od konkrétnej implementácie RDBMS. Na obr. č. 3.5 sú znázornené dátové typy, (čiastočne) určujúce definičné obory hodnôt v stĺpcoch. Výber dátových typov závisí od vybraného systému pre manažment databázy. Na obrázku sú dátové typy, ktoré budú použité v PostgreSQL. Ak by sme použili MySQL, tak by sme namiesto `serial` použili `int`.



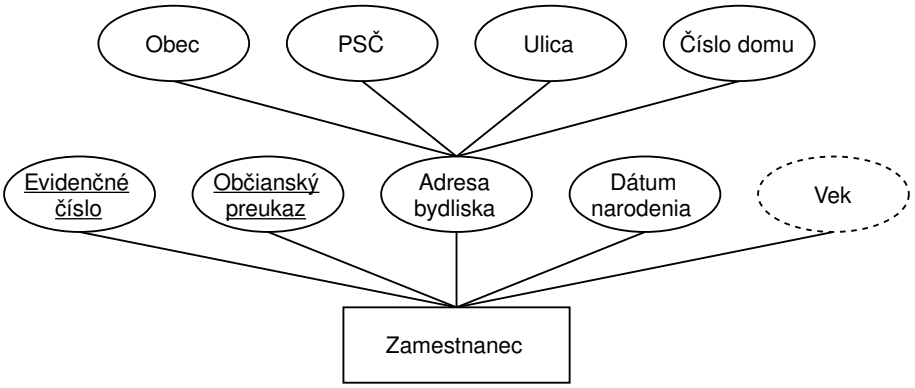
Obrázok 3.5: Fyzický model relačnej databázy

3.3 Transformácia konceptuálneho modelu na logický model relačnej databázy

Logický model relačnej databázy musí vychádzať z princípov v nej používaných. Je viac špecifický (bližší k implementácii) ako konceptuálny model. Z toho dôvodu model obsahuje iné typy prvkov ako konceptuálny model vo forme ER modelu (iný spôsob definovania štruktúry databázy a pravidiel v nej). Preto po zachytení hlavných čŕt databázy pomocou ER modelu je potrebná transformácia do logického modelu.

3.3.1 Transformácia (regulárnych/silných) entít

Pre každý typ entity v ER modely je potrebné v logickom modely vytvoriť schému relácie (tabuľku). Každý riadok tejto tabuľky nesie informácie o jednej entite (inštancii). Názov tabuľky môže byť rovnaký ako názov typu entity (niekedy sa používa názov v množnom čísle).



Zamestnanec	
PK	**Evidenčné číslo**
	Občianský preukaz
	Bydlisko - obec
	Bydlisko - PSČ
	Bydlisko - ulica
	Bydlisko - číslo domu
	Dátum narodenia

Obrázok 3.6: Transformácia typu entity

Príklad transformácie regulárneho typu entity je ilustrovaný na obr. č. 3.6. Poznámky ku príkladu: V modely je pre stručnosť vynechané napr. meno zamestnanca. Pre reprezentáciu adresy by bolo dobré zvážiť vytvorenie viacerých tabuliek, čo pre jednoduchosť v tomto príklade nie je spravené.

Jednoduché (atomické) atribúty entít transformovaného typu sú aj atribútmi schémy relácie (stĺpcami tabuľky). Môžeme pre ne použiť rovnaké názvy.

Logický model relačnej databázy nedokáže reprezentovať zložené atribúty. Preto sa zložené atribúty typu entity pretransformujú tak, že stĺpcami tabuľky sú len jednoduché zložky zložených atribútov.

Primárnym kľúčom tabuľky je kľúč typu entity. Ak typ entity obsahuje viacero kľúčov, tak sa za primárny kľúč zvolí jeden z nich. Stĺpce tabuľky zodpovedajúce ostatným kľúčom typu entity, sú sekundárnymi kľúčmi. Sekundárne kľúče môžeme vyznačiť v diagrame, alebo o nich napísať v textovej časti dokumentácie modelu. Ak je kľúč typu entity zloženým atribútom, tak je primárnym alebo sekundárnym kľúčom tabuľky množina stĺpcov, zodpovedajúcich jednoduchým (atomickým) atribútom, ktoré tvoria zložený kľúčový atribútu.

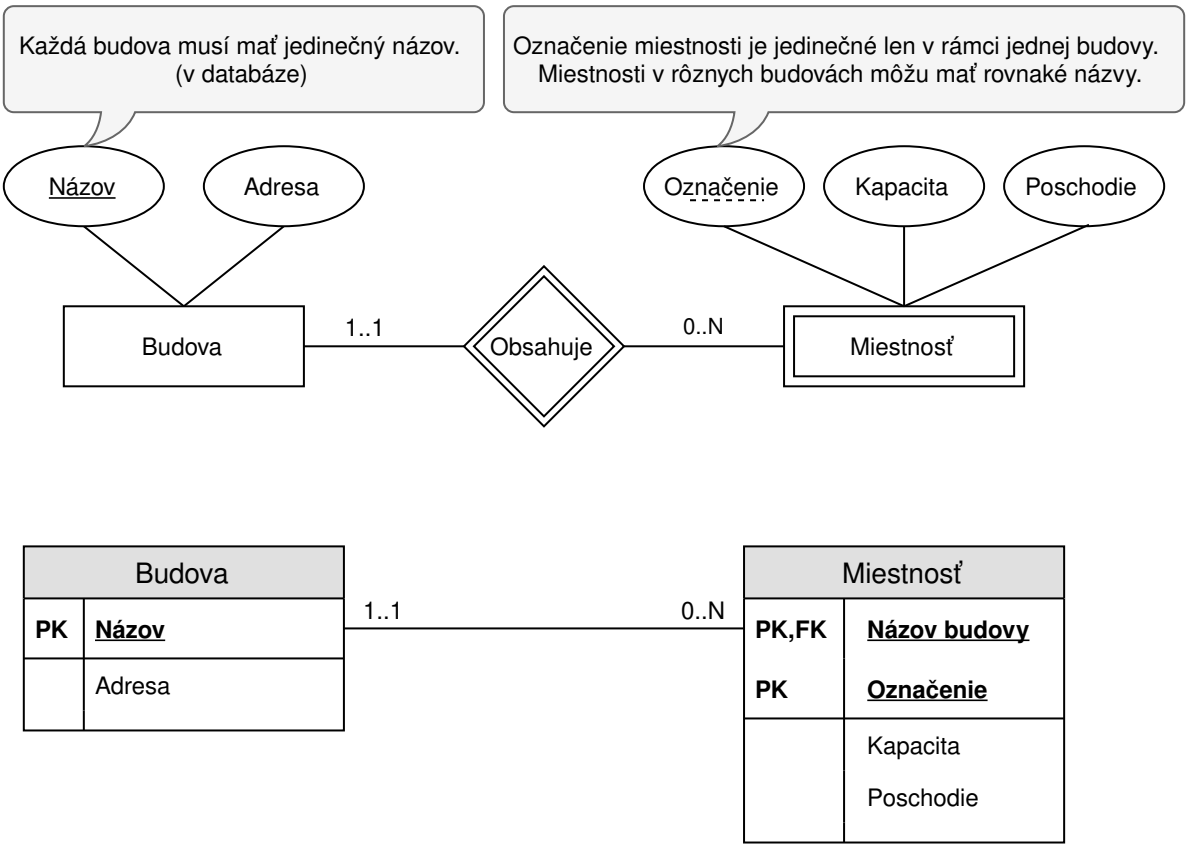
Transformovanie viachodnotových atribútov je popísané v časti 3.3.6.

Odvodené atribúty sa do logického modelu neprenášajú. V prípade implementačnej optimalizácie sa ale môžeme rozhodnúť inak (ak sa napríklad hodnota odvodeného atribútu mení málokedy, ale zisťovať jeho hodnotu potrebujeme často).

3.3.2 Transformácia typu slabej entity

Pri transformácii typu slabej entity je potrebné najprv pretransformovať identifikačný typ entity. Identifikačný typ môže byť silný alebo slabý, preto treba zvoliť vhodnú postupnosť transformácie typov entít.

Transformácia typu slabej entity sa od transformácie silného typu entity líši nasledovne. Tabuľka (schéma relácie) zodpovedajúca typu slabej entity, naviac obsahuje aj primárny kľúč identifikačnej entity. V tabuľke zodpovedajúcej typu slabej entity je cudzím kľúčom. Primárny kľúč tabuľky zodpovedajúcej typu slabej entity je tvorený čiastočným kľúčom a cudzím kľúčom, identifikujúcim identifikačnú entitu.



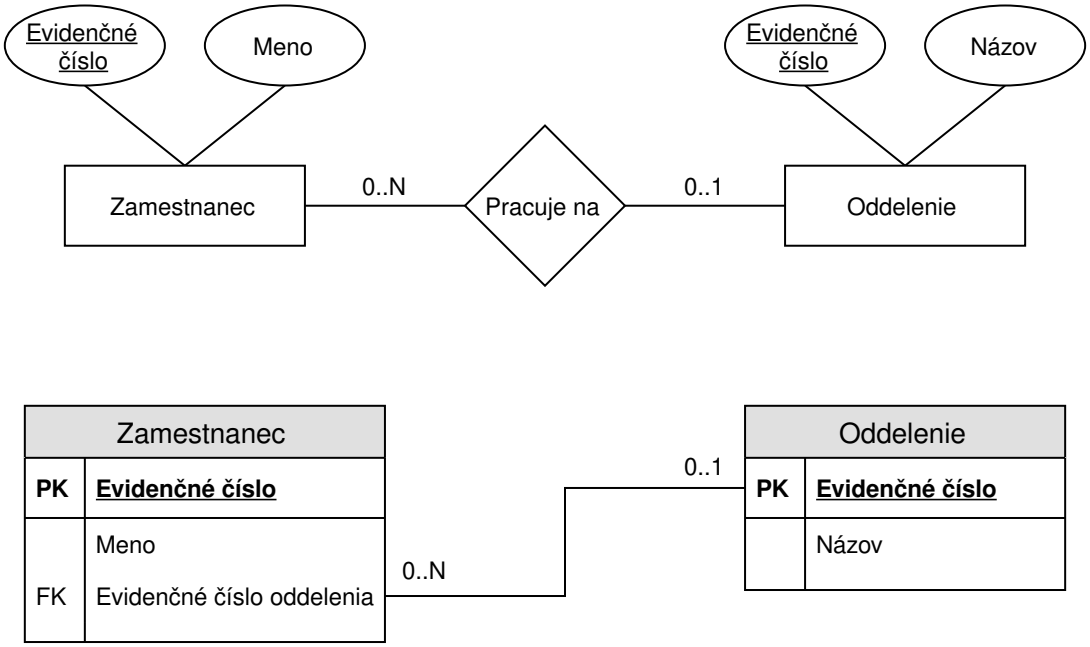
Obrázok 3.7: Transformácia typu slabej entity

Transformácia typu slabej entity je zobrazená na obr. č. 3.7. Pri tvorbe modelu predpokladáme, že každá budova má v databáze jedinečný názov, ale označenie miestnosti (napr. číslom) je jedinečné len v rámci jednej budovy. Preto je označenie miestnosti len čiastočným kľúčom a inštancie typu `Miestnosť` sú slabé entity. Inštancie typu `Budova` sú silné entity.

Primárny kľúč tabuľky `Miestnosť` sa skladá z atribútov (stĺpcov) `Názov budovy` a `Označenie`. Atribút (stĺpec) `Názov budovy` je zároveň cudzím kľúčom, identifikujúcim budovu, v ktorej je daná miestnosť. Čiže modeluje spôsob implementácie identifikačného vzťahu.

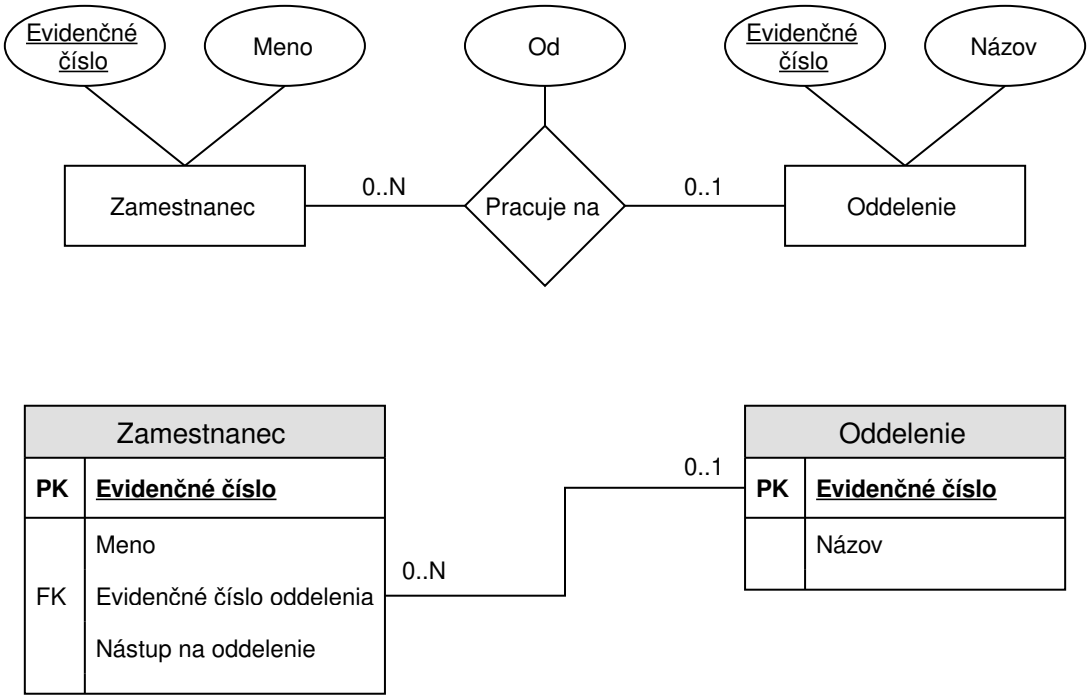
3.3.3 Transformácia binárneho vzťahu 1:N

Na realizáciu binárneho typu vzťahu s pomerom násobností 1:N sa v relačnej databáze používa cudzí kľúč. Príklad transformácie takéhoto vzťahu je na obr. č. 3.8. Keďže konceptuálny model definuje, že zamestnanec môže byť zaradený najviac na jedno oddelenie, tak do tabuľky `Zamestnanec` v logickom modely stačí pridať cudzí kľúč, ktorý obsahuje evidenčné číslo oddelenia, na ktorom zamestnanec pracuje. Tým je riadok v tabuľke `Zamestnanec` prepojený s riadkom v tabuľke `Oddelenie`. Podľa konceptuálneho modelu zamestnanec nemusí byť zaradený na žiadne oddelenie. Preto hodnota cudzieho kľúča môže byť NULL. Ak by konceptuálny model definoval, že každý zamestnanec musí byť povinne zaradený na niektoré oddelenie, tak by hodnota cudzieho kľúča nemohla byť NULL.



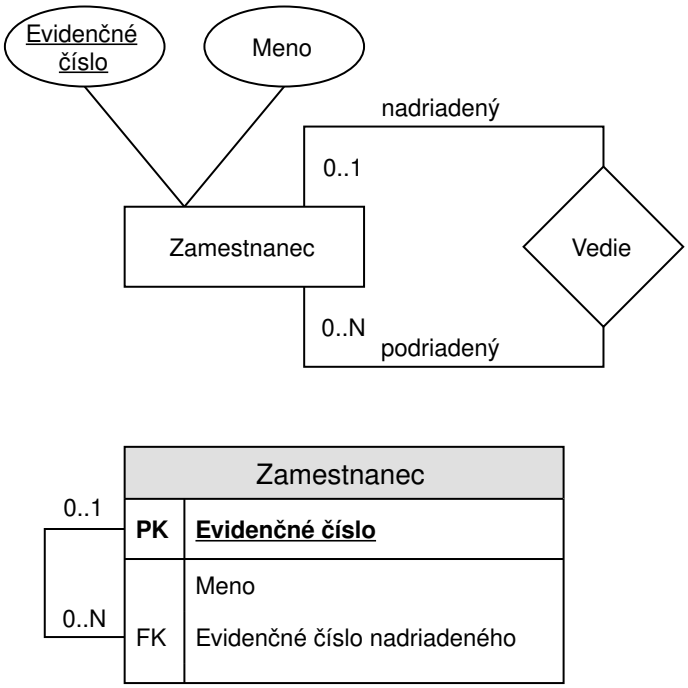
Obrázok 3.8: Transformácia vzťahu 1:N pomocou cudzieho

Ak má vzťah jednoduché (atomické) atribúty, tak tieto atribúty budú stĺpcami tabuľky (atribútmi schémy relácie), obsahujúcej príslušný cudzí kľúč (obr. č. 3.9). Podobne to platí aj pre jednoduché zložky zložených atribútov typu vzťahu.



Obrázok 3.9: Transformácia vzťahu 1:N obsahujúceho atribút

Na obr. č. 3.10 je príklad transformácie typu vzťahu 1:N medzi inštanciami rovnakého typu. Typ vzťahu `Vedie` reprezentuje vzťahy v hierarchii zamestnancov. Každý riadok tabuľky zamestnanec reprezentuje jedného zamestnanca. Je v ňom uvedené jeho evidenčné číslo, aj evidenčné číslo priameho nadriadeného. Evidenčné číslo priameho nadriadeného je cudzí kľúč.

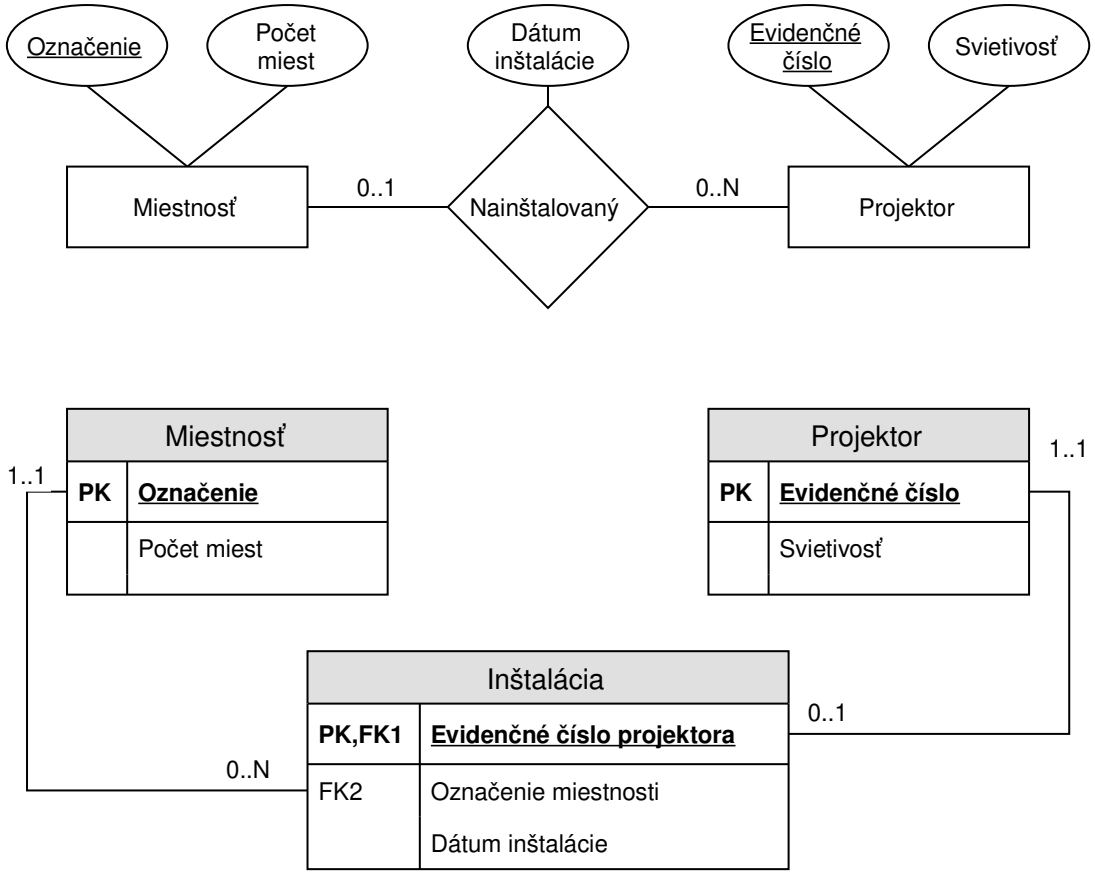


Obrázok 3.10: Transformácia unárneho vzťahu 1:N

Ak sú hodnoty cudzích kľúčov realizujúcich vzťah 1:N často NULL (čiže je relatívne málo inštancii daného typu vzťahu), potom je dobré zvážiť iný spôsob riešenia. Pre takýto prípad môžeme vytvoriť novú tabuľku, ktorá bude reprezentovať typ vzťahu z konceptuálneho modelu. Táto tabuľka bude obsahovať dva cudzie kľúče, ktoré budú nadobúdať hodnoty primárnych kľúčov tabuliek vo vzťahu. Tieto dva cudzie kľúče tvoria primárny kľúč novej tabuľky.

Na obr. č. 3.11 je evidencia inštalácie projektorov do miestností. Vo veľkých miestnostiach sú nainštalované viaceré projektory. Predpokladajme, že väčšina projektorov sú používané ako prenosné, čiže nie sú nainštalované v žiadnej miestnosti. Preto, ak by sme v logickom modely projektorom pridali cudzí kľúč obsahujúci označenie miestností, kde sú nainštalované, väčšina z nich by mala hodnotu NULL. Preto je v logickom modely vytvorená nová tabuľka `Inštalácia`, ktorej každý riadok nesie informáciu o inštalácii jedného projektora do miestnosti. Ak sú v miestnosti nainštalované napríklad dva projektory, tak tabuľka `Inštalácia` obsahuje dva riadky obsahujúce označenie tejto miestnosti spolu s evidenčnými číslami príslušných projektorov. Ak v miestnosti nie je nainštalovaný žiadny projektor, tak v tabuľke `Inštalácia` nie je žiadny riadok obsahujúci jej označenie. Podobne, ak projektor nie je nainštalovaný v žiadnej miestnosti, tak sa jeho evidenčné číslo v tabuľke `Inštalácia` nenachádza. Primárnym kľúčom tabuľky `Inštalácia` je evidenčné číslo projektora.

Takého riešenie môže znížiť pamäťové nároky. Niektoré typy vyhľadávania budú pomalšie, niektoré rýchlejšie.

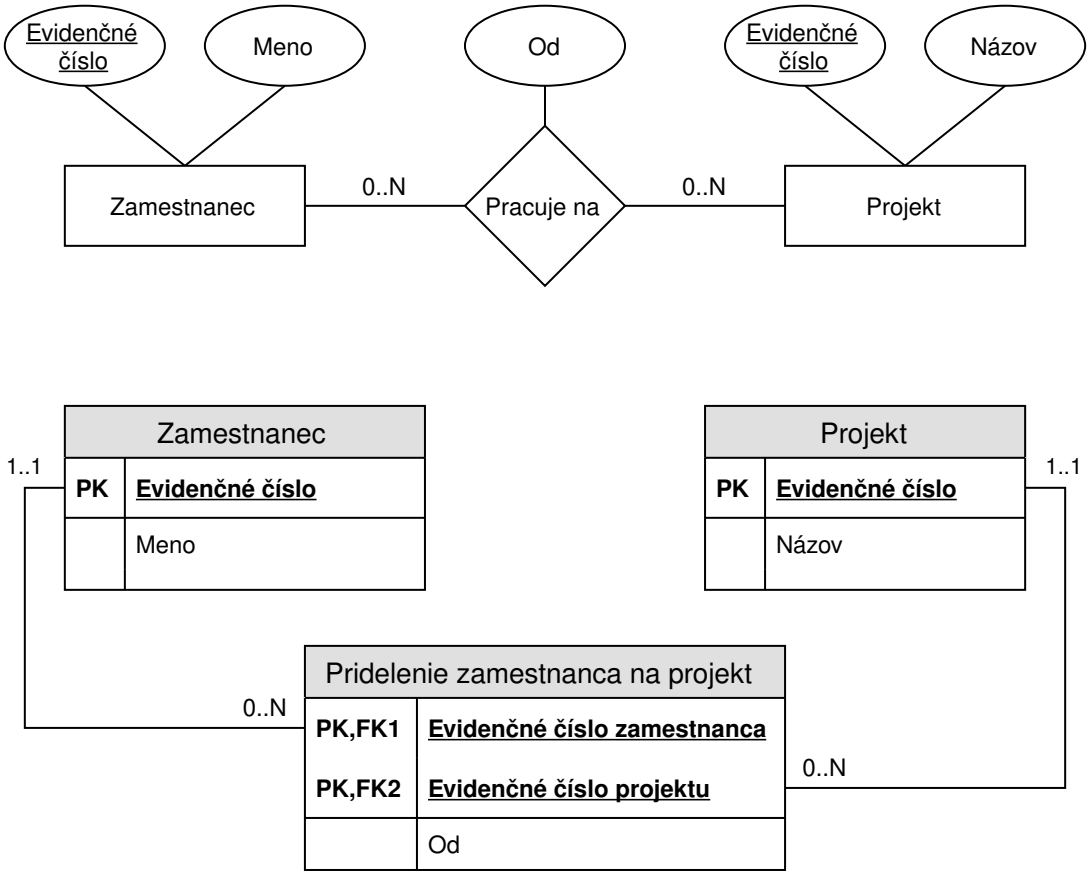


Obrázok 3.11: Transformácia binárneho vzťahu 1:N tabuľkou

3.3.4 Transformácia binárneho vzťahu M:N

Pre každý typ vzťahu s pomerom násobnosti M:N je potrebné vytvoriť (pridať) novú tabuľku. Primárne kľúče tabuliek (typov entít) vo vzťahu, budú cudzími kľúčmi v novej tabuľke. Kombinácia týchto cudzích kľúčov je primárnym kľúčom novej tabuľky. Ak typ vzťahu obsahuje jednoduché (atomické) atribúty, tak tieto sú v novej tabuľke. Podobne (ako pri transformácii typu entity) to platí aj pre jednoduché zložky zložených atribútov.

Podľa ER modelu na obr. č. 3.12 môže každý zamestnanec pracovať na viacerých projektoch a na každom projekte môže pracovať viacero zamestnancov. V logickom modely, každý riadok novej pridanej tabuľky `Pridelenie zamestnanca na projekt` reprezentuje pridelenie jedného zamestnanca na jeden projekt. Ak je jeden zamestnanec pridelený napr. na dva projekty, tak sa jeho evidenčné číslo nachádza v dvoch riadkoch tejto tabuľky, spolu s príslušnými evidenčnými číslami projektov.



Obrázok 3.12: Transformácia binárneho vzťahu M:N

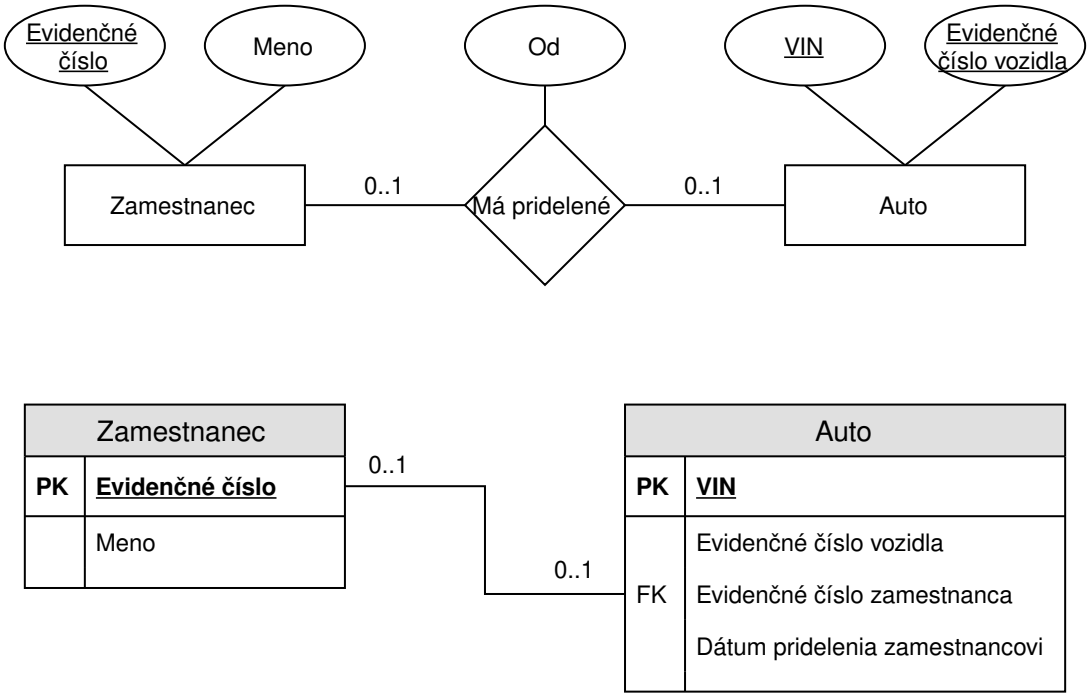
3.3.5 Transformácia binárneho vzťahu 1:1

Pri transformácii typu vzťahu s pomerom násobností 1:1 môžeme použiť viaceré možnosti.

Vo väčšine prípadov je vhodné použiť prepojenie prostredníctvom cudzích kľúčov. Toto prepojenie môže byť jednostranné (ako pri type vzťahu 1:N), alebo obojstranné (obidve tabuľky obsahujú cudzie kľúče do tabuľky na druhej strane vzťahu).

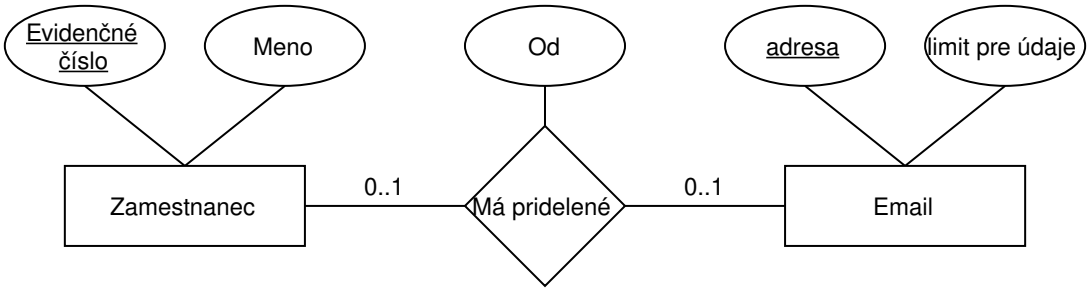
Na obr. č. 3.13 je príklad pridelenia služobného auta zamestnancovi. Predpokladáme, že menšina zamestnancov má pridelené auto, ale väčšina áut je pridelená niektorému zamestnancovi. Preto bude cudzí kľúč obsahovať tabuľka `Auto`.

Cudzí kľúč je lepšie umiestniť do tabuľky zodpovedajúcej typu entity s povinnou účasťou jej inštancii vo vzťahu.



Obrázok 3.13: Transformácia binárneho vzťahu 1:1 prostredníctvom cudzieho kľúča

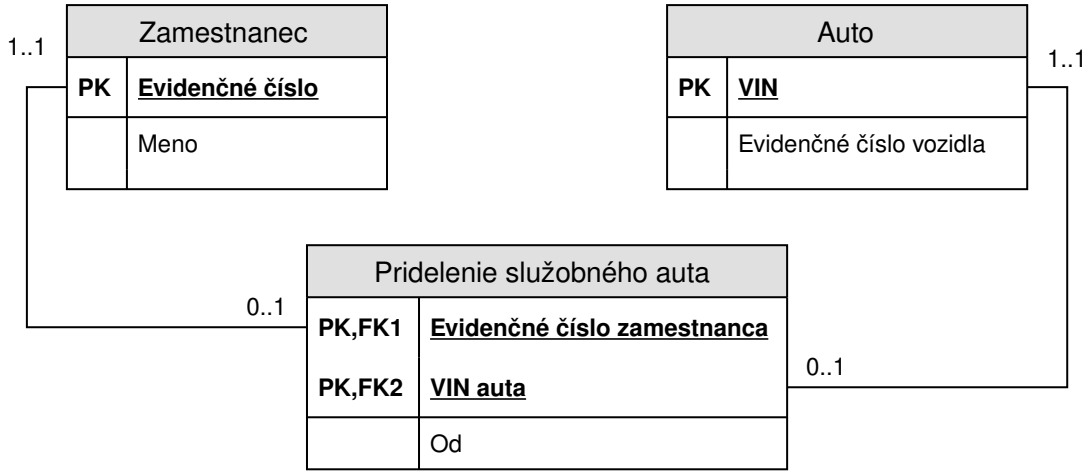
Ďalšou možnosťou je vytvorenie jednej spoločnej tabuľky pre obidva typy entít z ER modelu (obr. č. 3.14).



Zamestnanec	
PK	**Evidenčné číslo**
	Meno
	Emailová adresa
	Limit emailového účtu
	Dátum pridelenia emailu

Obrázok 3.14: Transformácia binárneho vzťahu 1:1 zlúčením do jednej tabuľky

Ďalšou možnosťou je vytvorenie novej tabuľky reprezentujúcej vzťah, podobne ako pri type vzťahu M:N (obr. č. 3.15).

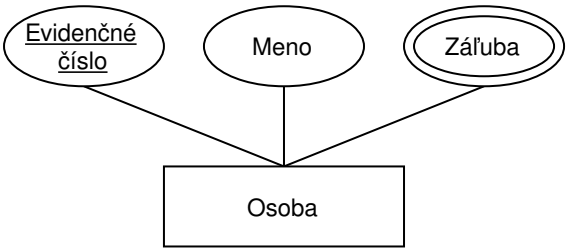


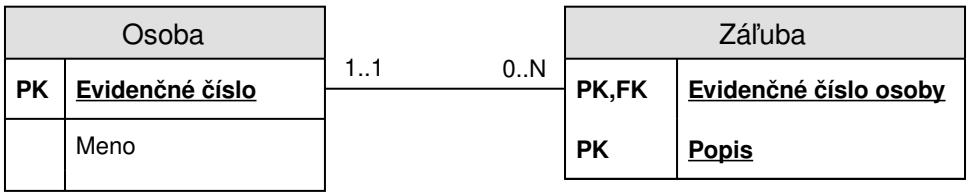
Obrázok 3.15: Transformácia binárneho vzťahu 1:1 pridaním tabuľky

3.3.6 Transformácia viachodnotových atribútov

Viachodnotový atribút typu entity alebo vzťahu pretransformujeme na novú tabuľku. Každá z hodnôt viachodnotového atribútu je evidovaná v samostatnom riadku novej tabuľky, pričom je s príslušnou entitou alebo vzťahom prepojená pomocou cudzieho kľúča. Cudzí kľúč obsahuje hodnotu primárneho kľúča tabuľky, reprezentujúcej príslušný typ entity alebo vzťahu. Primárny kľúč novej tabuľky je tvorený cudzím kľúčom a hodnotou atribútu.

Podľa modelu na obr. č. 3.16 môže mať osoba viacero záľub. Pri transformácii do logického modelu relačnej databázy je vytvorená tabuľka `Záľuba`, v ktorej je každá záľuba osoby evidovaná zvlášť. Táto tabuľka obsahuje `Evidenčné číslo osoby`, ktoré je cudzím kľúčom na osobu. Primárny kľúč tabuľky `Záľuba` je tvorený cudzím kľúčom identifikujúcim osobu a hodnotou (popisom) záľuby.



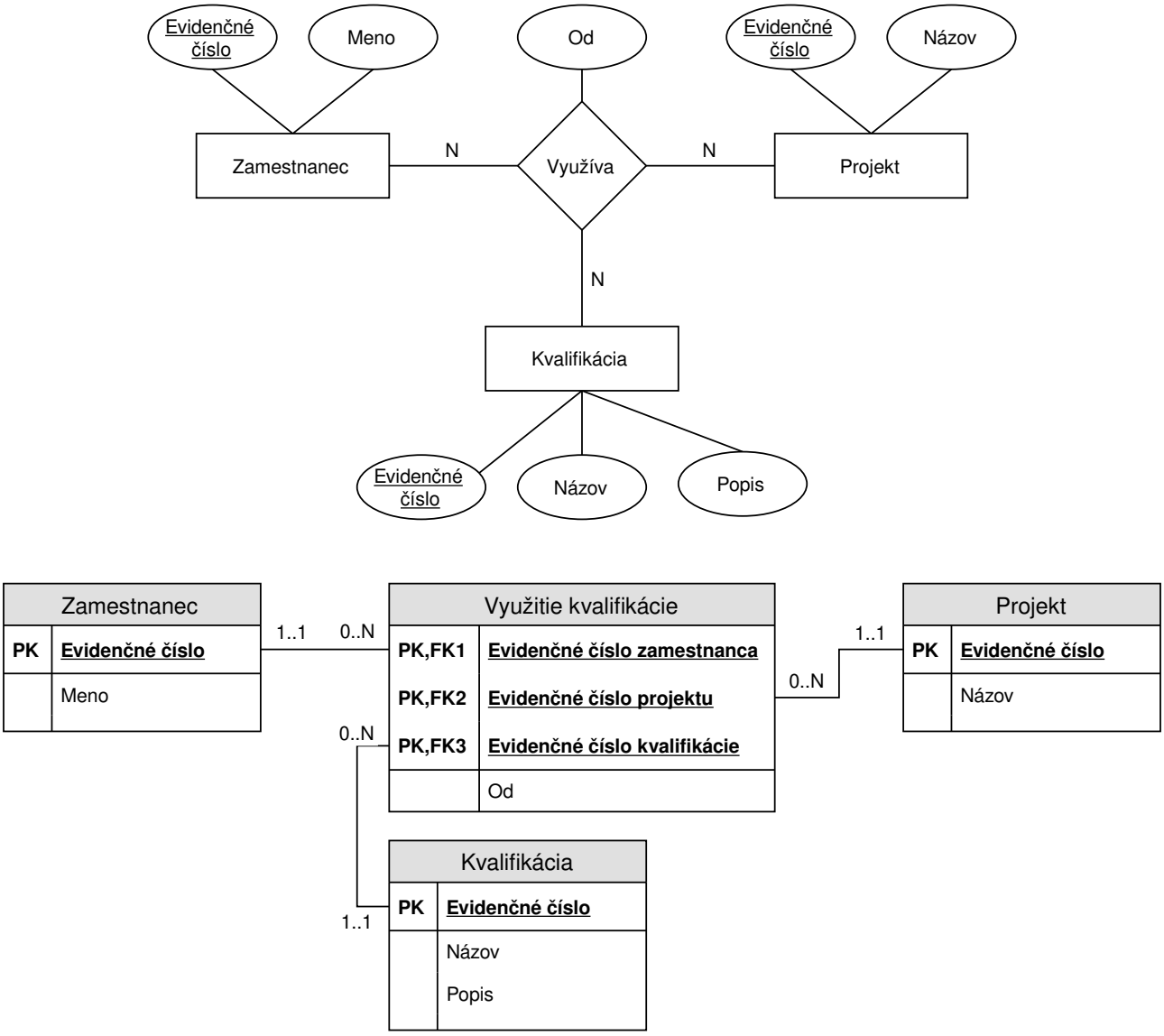


Obrázok 3.16: Transformácia viachodnotového atribútu

3.3.7 Transformácia vzťahov vyššieho stupňa

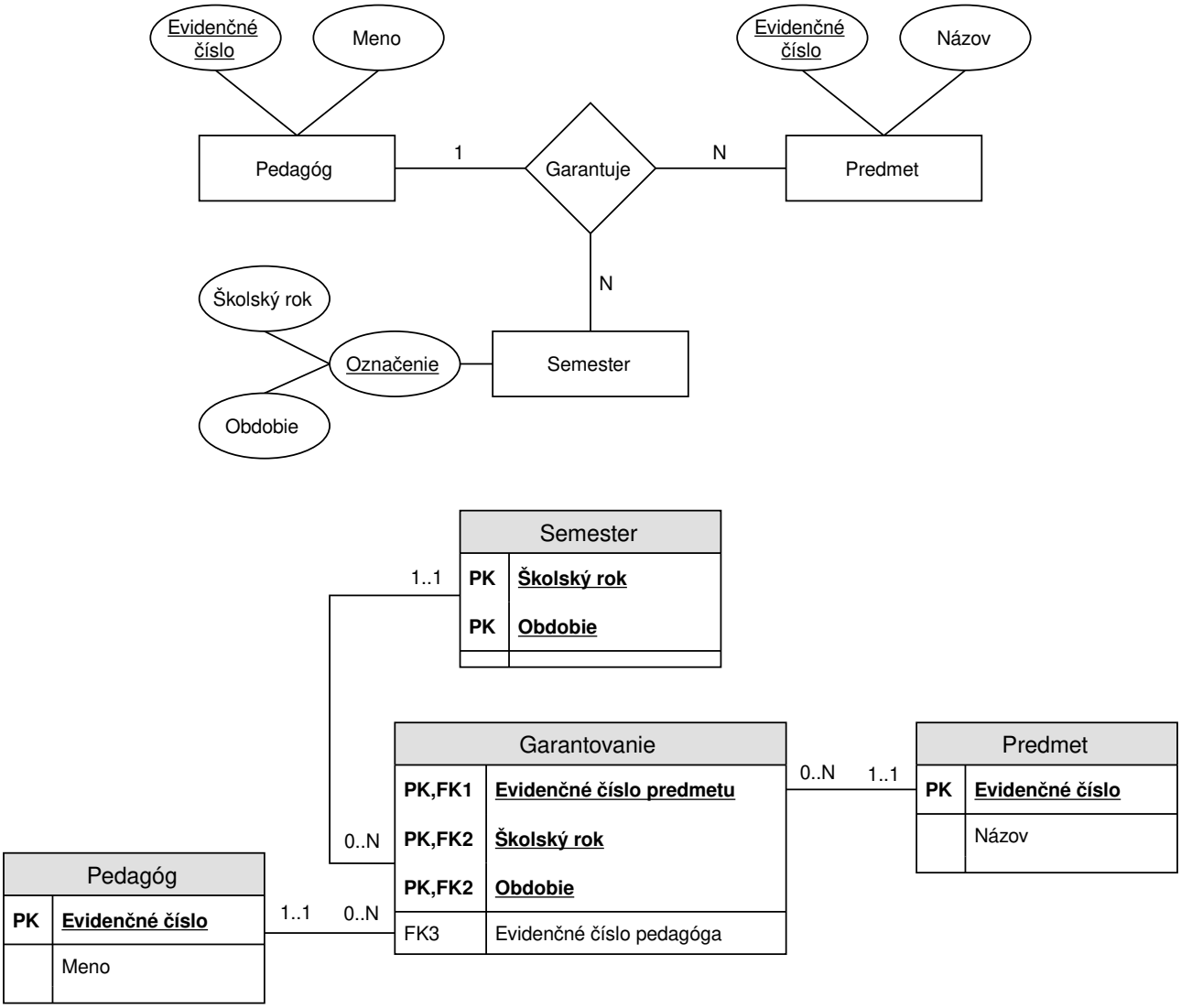
Typ vzťahu vyššieho stupňa (tretieho alebo vyššieho) sa transformuje do samostatnej tabuľky. Táto tabuľka obsahuje cudzie kľúče, ktorých hodnoty sú rovné hodnotám primárnych kľúčov tabuliek reprezentujúcich typy entít v danom type vzťahu. Táto tabuľka tiež obsahuje atribúty typu vzťahu (jednoduché atribúty a jednoduché zložky zložených atribútov). Primárny kľúč tabuľky je tvorený kombináciou všetkých cudzích kľúčov, ktoré identifikujú entity typov, ktorých násobnosť je viac ako 1.

Na obr. č. 3.17 je transformácia typu vzťahu reprezentujúceho využitie kvalifikácie zamestnancom na projekte. Typ vzťahu `Využíva` je v logickom modely reprezentovaný tabuľkou `Využitie kvalifikácie`, ktorá obsahuje cudzie kľúče identifikujúce riadky v tabuľkách vo vzťahu. Keďže je pomer násobnosti typu vzťahu `Využíva` M:N:P, tak je primárny kľúč tvorený všetkými troma cudzími kľúčmi. Tabuľka `Využitie kvalifikácie` obsahuje aj stĺpec, reprezentujúci atribút typu vzťahu `Využíva`.



Obrázok 3.17: Transformácie n-árneho typu vzťahu M:N:P

Na obr. č. 3.18 je príklad transformácie ternárneho typu vzťahu 1:M:N. Na univerzite musí byť každý vyučovaný predmet garantovaný pedagógom. Príklad modeluje garantovanie predmetu pedagógom na určitý semester. Primárny kľúč tabuľky `Garantovanie` sa skladá len z dvoch cudzích kľúčov. Evidenčné číslo pedagóga nie je potrebné pre identifikáciu riadku v tabuľke `Garantovanie`, pretože podľa konceptuálneho modelu môže určitý predmet na určitý semester garantovať najviac jeden pedagóg.

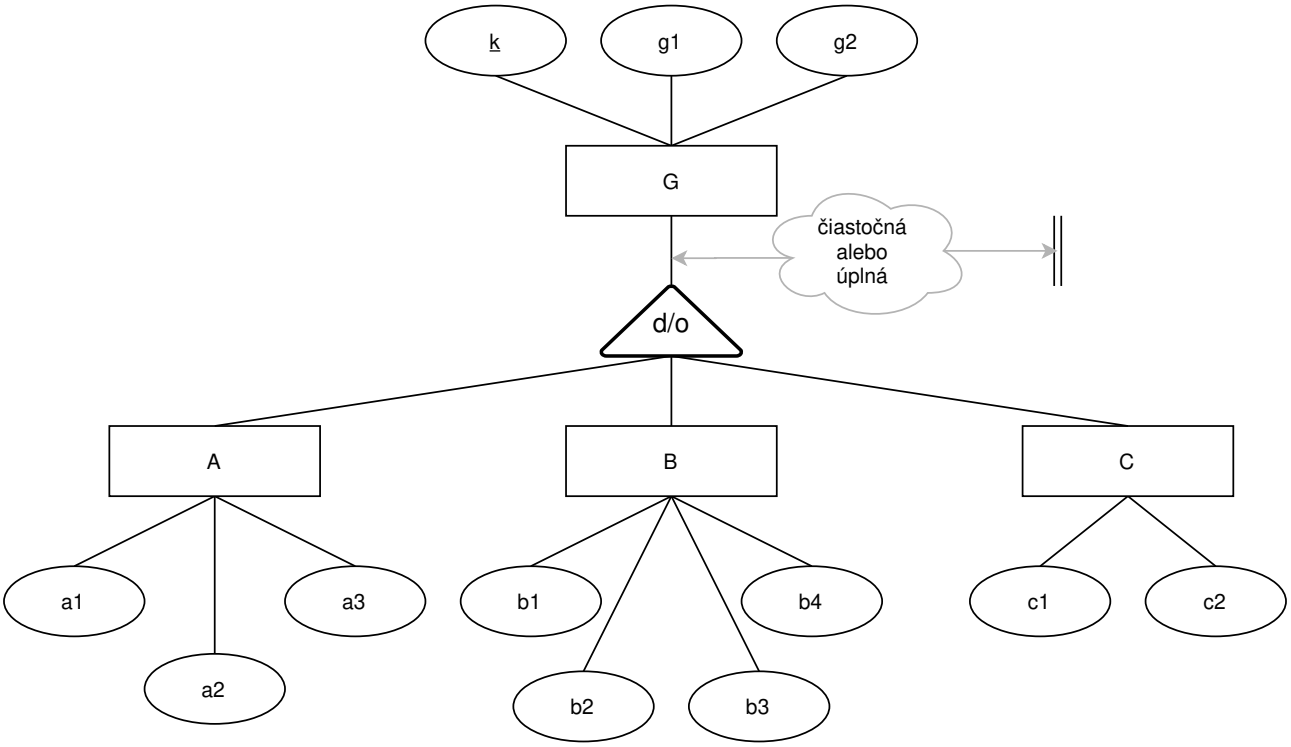


Obrázok 3.18: Transformácia n-árneho typu vzťahu 1:M:N

3.3.8 Transformácia generalizácie (špecializácie)

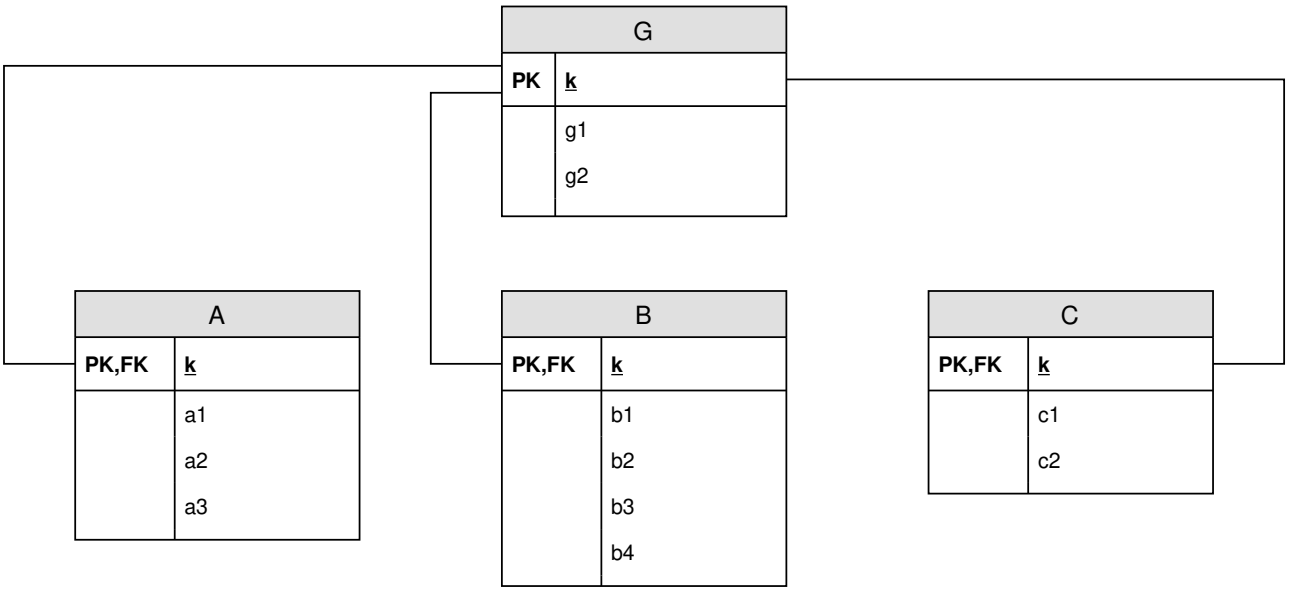
Pre transformáciu skupiny typov entít vo vzťahu generalizácie (špecializácie) existuje viacero možností. Vhodnú možnosť vyberáme napríklad aj podľa toho, či je generalizácia (špecializácia):

- čiastočná (partial) , alebo úplná (total),
- výlučná (disjoint), alebo prekrývajúca (overlapping).



Obrázok 3.19: ER model obsahujúci generalizáciu (špecializáciu)

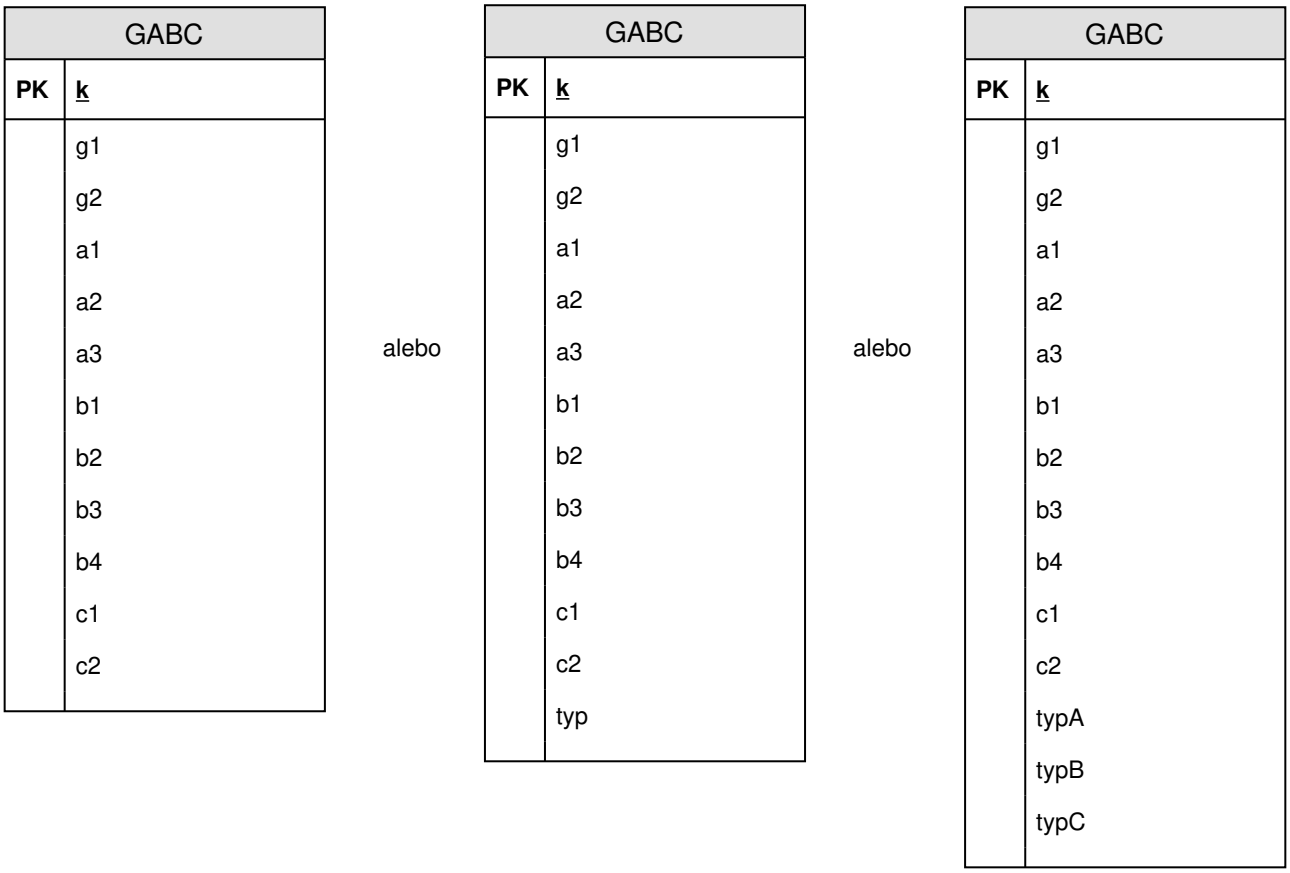
ER model na obr. č. 3.19 môžeme vo všeobecnosti pretransformovať tromi spôsobmi. Často najvhodnejším je transformácia na tabuľky, ktoré reprezentujú zvlášť všeobecnú a zvlášť každý špecializovaný typ entity (obr. č. 3.20). Tieto tabuľky sú prepojené prostredníctvom kľúča `k`. Táto možnosť je vhodná pre úplnú aj čiastočná, výlučnú aj prekrývajúcu sa generalizáciu (špecializáciu).



Obrázok 3.20: Transformácia generalizácie reprezentovaním každého typu entity tabuľkou

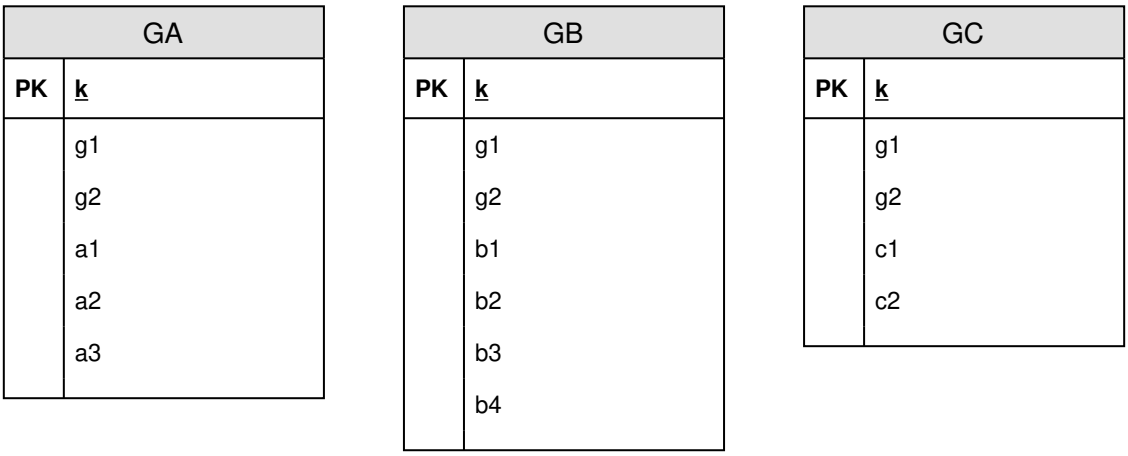
Ďalšou možnosťou je vytvorenie jedinej tabuľky obsahujúcej atribúty všetkých typov entít (obr. č. 3.21). Ak je špecializácia výlučná, tak v tejto tabuľke môže byť pridaný atribút (napr. `typ`)

obsahujúci informáciu, či je entita typu `A`, `B`, alebo `C`. Ak je špecializácia prekrývajúca, tak v tejto tabuľke môžu byť pridané boolovské atribúty (napr. `typA`, `typB`, `typC`), ktoré obsahujú informáciu, ktorých podtypov je entita. Nevýhodou tohto riešenia môže byť veľký počet hodnôt NULL.



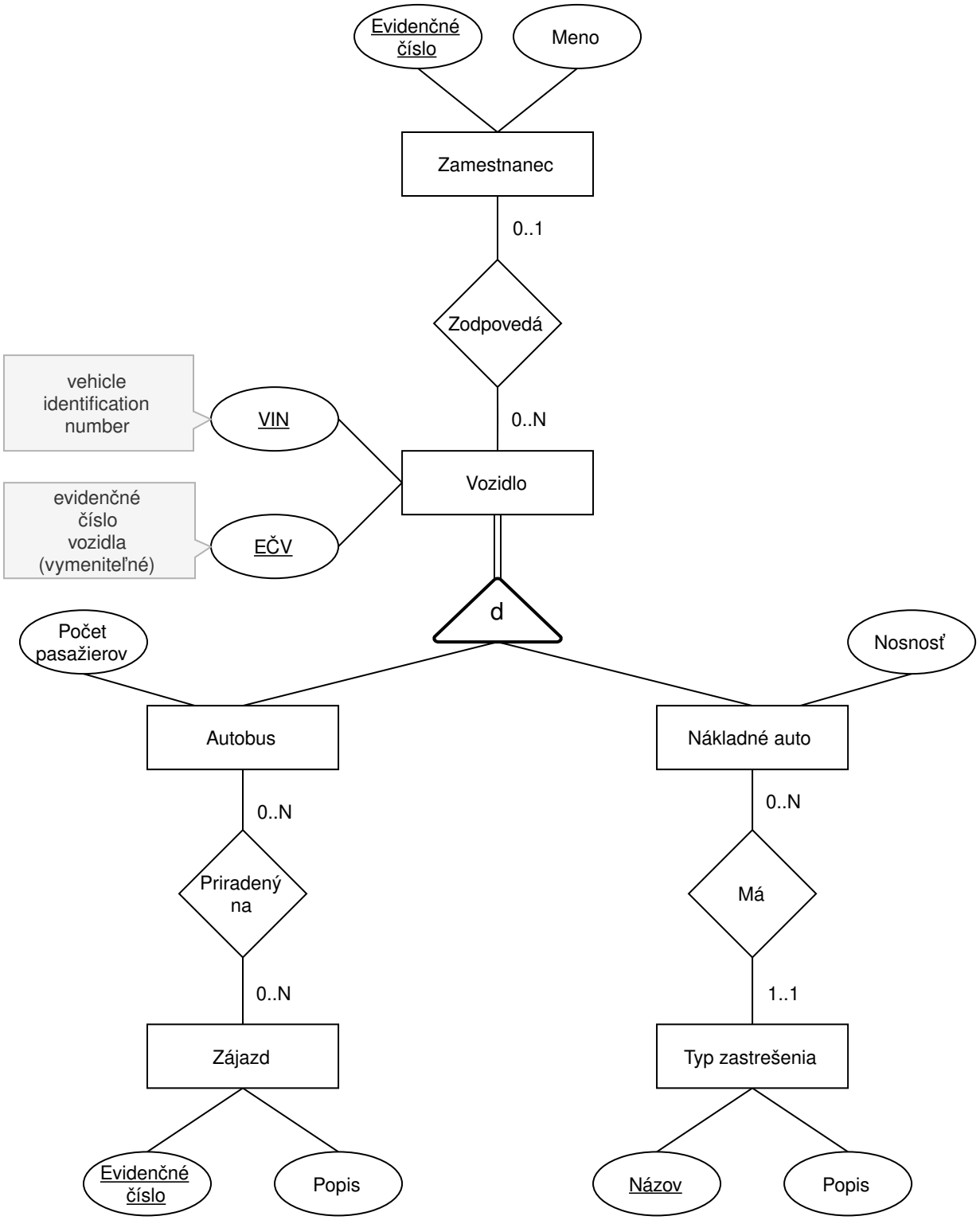
Obrázok 3.21: Transformácia generalizácie jednou spoločnou tabuľkou

Ďalšou možnosťou je vytvorenie tabuliek pre špecializovane podtypy entít, v ktorých ale každá obsahuje aj atribúty všeobecného nadtypu (obr č. 3.22). Táto možnosť je vhodná, ak je špecializácia úplná. Je viac výhodná pre výlučnú špecializáciu. V prípade, že je špecializácia prekrývajúca, nevýhodou je duplicita hodnôt definovaných v nadtype `G`.



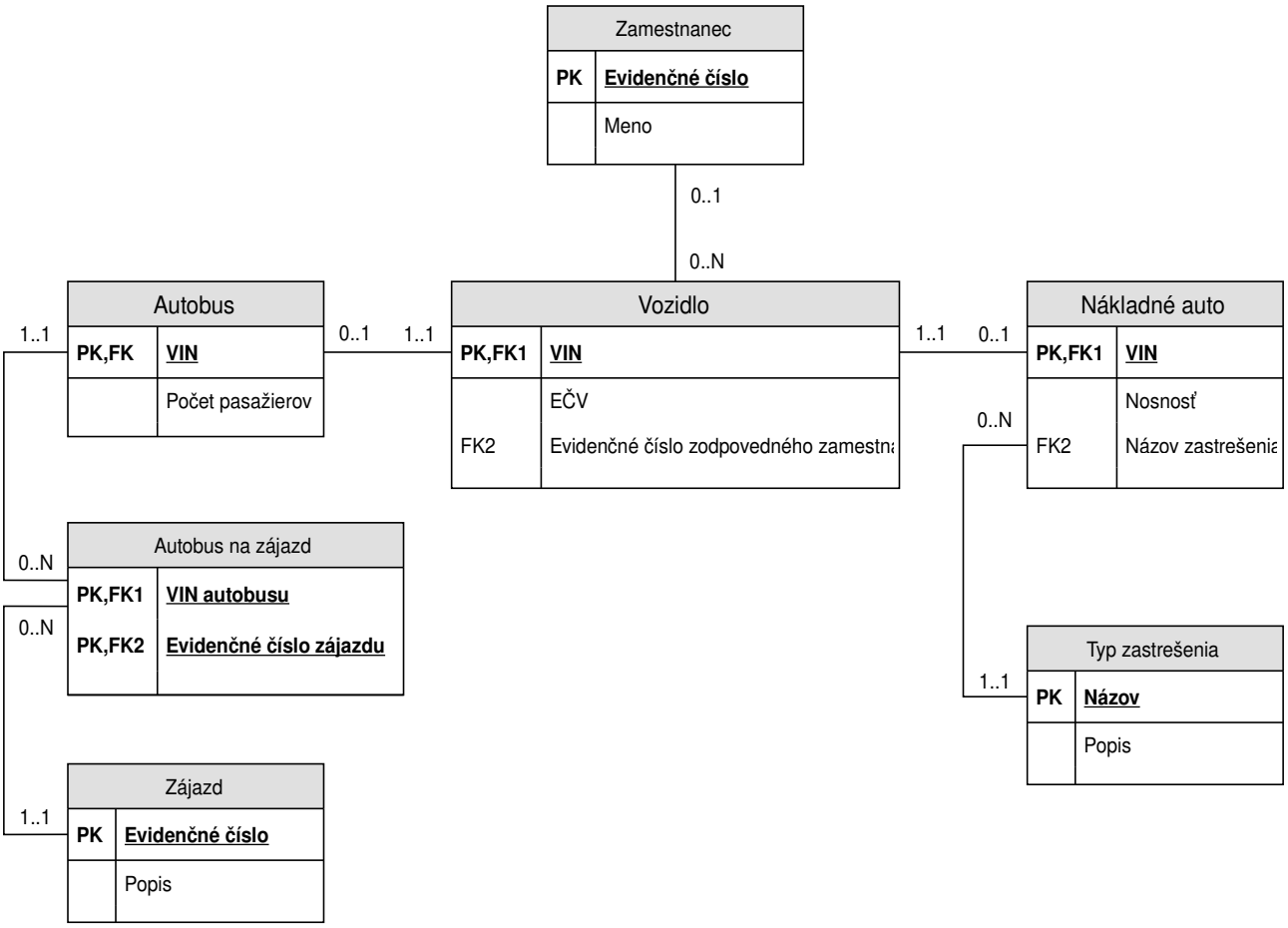
Obrázok 3.22: Transformácia generalizácie vytvorením tabuliek pre špecializované typy entít

Na obr. č. 3.23 je príklad modelovania databázy obsahujúcej dva typy vozidiel. Tieto dva typy majú nie len spoločné a špeciálne atribúty, ale aj spoločné a rozdielne vzťahy s inými typmi entít. Zamestnanec môže byť zodpovedný za autobus alebo nákladné auto. Na zájazd môže byť priradený len autobus. Typ zastrešenia je evidovaný len pre nákladné autá. Tieto rozdiely sa prejavia aj v logickom modely.



Obrázok 3.23: Evidencia vozidiel – ER model

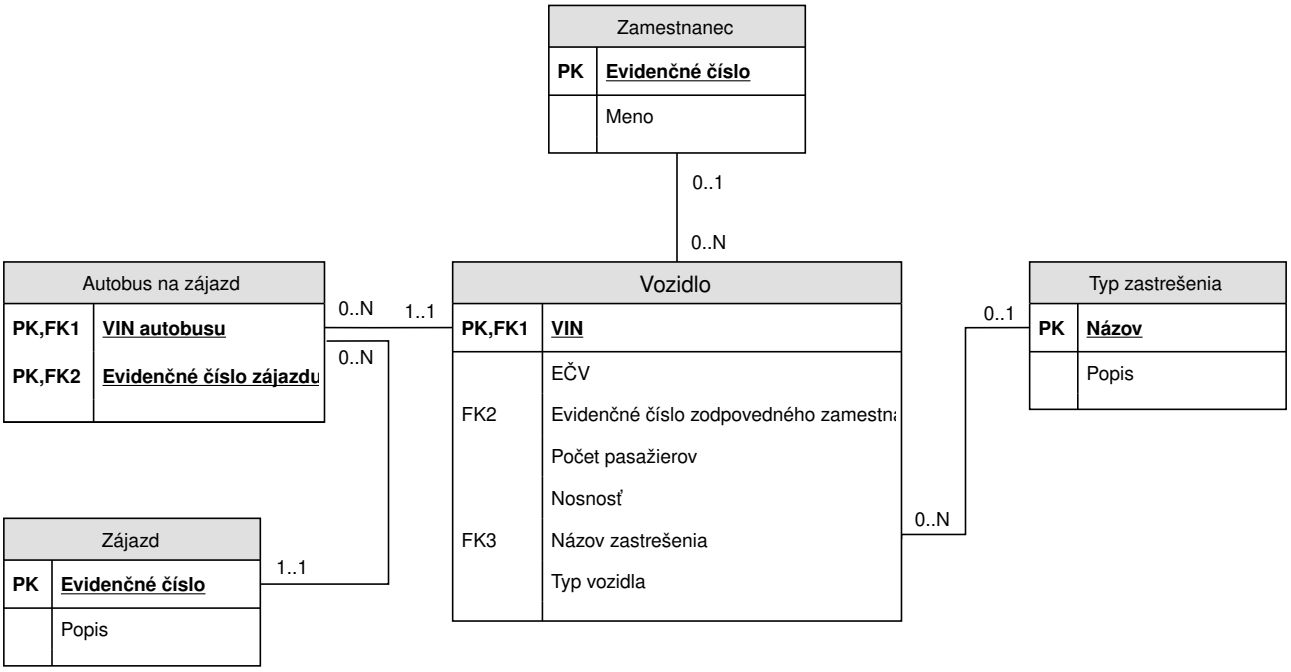
Na obr. č. 3.24 je logický model vytvorený tak, že spoločné vlastnosti vozidiel sú reprezentované spoločnou tabuľkou, špecifické vlastnosti ďalšími tabuľkami (podobne ako na obr. č. 3.20).



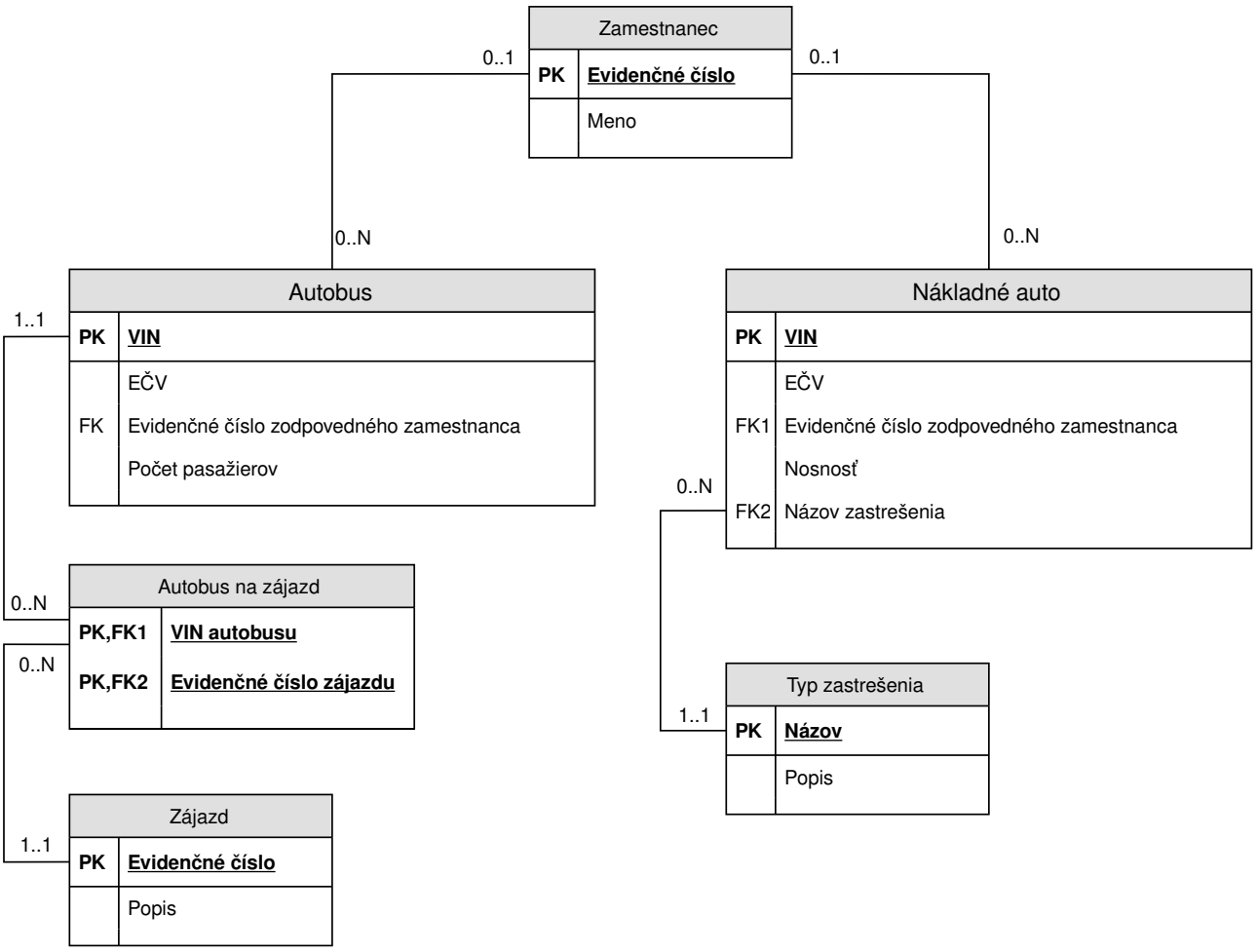
Obrázok 3.24: Evidencia vozidiel - logický model (verzia č. 1)

Logický model, v ktorom sú všetky vozidlá evidované v jednej tabuľke je na obr. č. 3.25 (podobne ako riešenie na obr. 3.21). Či je vozidlo autobus, alebo nákladné auto môžeme rozlišovať podľa hodnôt NULL, alebo podľa atribútu `Typ vozidla`. Nevýhodou riešenia sú väčšie pamäťové nároky. Veľa hodnôt v tabuľke `Vozidlo` bude NULL.

Logický model v ktorom sú len tabuľky reprezentujúce špecializované typy vozidiel je na obr. č. 3.26 (podobné riešenie ako na obr. č. 3.22). Keďže vozidlo nemôže byť zároveň autobusom aj nákladným vozidlom, databáza nebude obsahovať duplicitné údaje `VIN`, `EČV` a `Evidenčné číslo zodpovedného zamestnanca`.



Obrázok 3.25: Evidencia vozidiel - logický model (verzia č. 2)



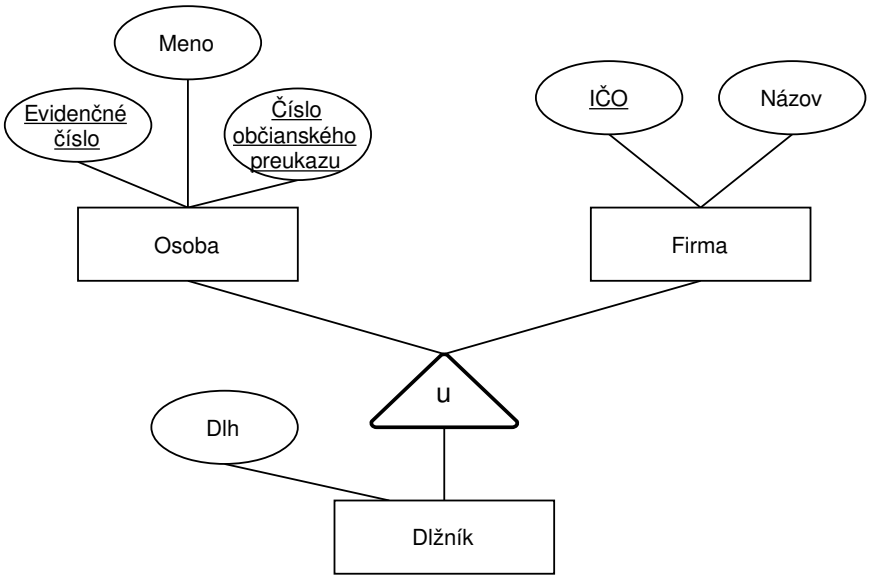
Obrázok 3.26: Evidencia vozidiel - logický model (verzia č. 3)

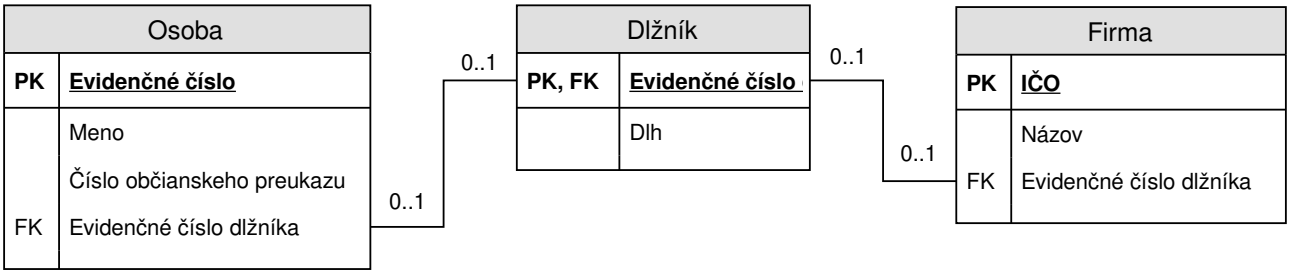
3.3.9 Transformácia zdielaných podtypov a unionu (kategórie)

Nadtypy zdielaného podtypu majú rovnaký kľúčový atribút. Nadtypy unionu (zjednotenia, kategórie) ale nemusia mať rovnaký kľúčový atribút, pretože vo všeobecnosti sú to navzájom nezávislé a teda rozdielne typy entít.

Ak nadtypy nemajú rovnaký kľúčový atribút, tak pri vytváraní logického modelu je v ich spoločnom podtype často definovaný **náhradný kľúč** (angl. surrogate key).

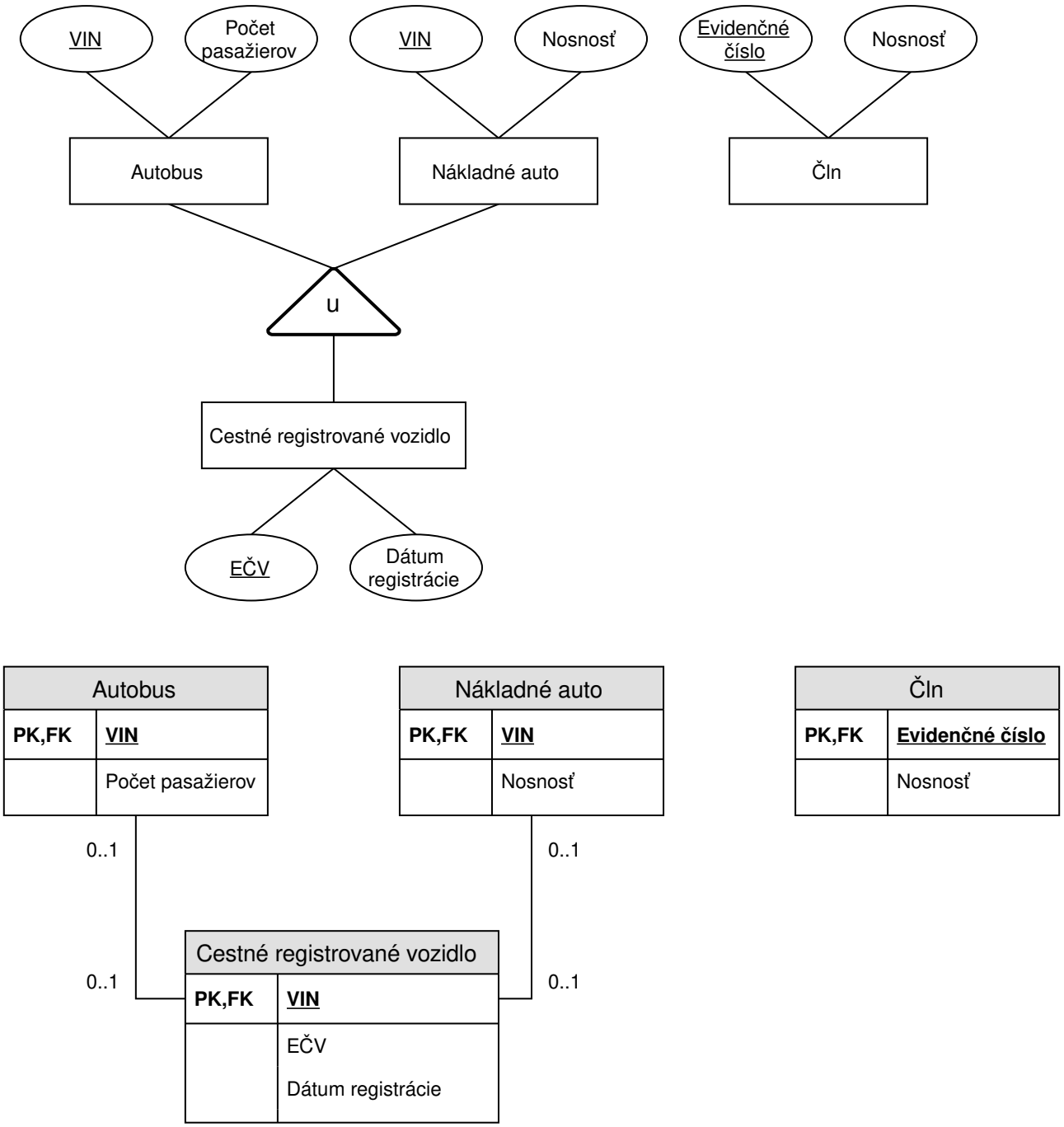
Na obr. č. 3.27 je union typu `Dlžník`. Dlžníkom môže byť niektorá osoba, alebo firma. Nie každá osoba alebo firma je dlžníkom. `Osoba` a `Firma` sú inak nezávislé typy entít. Majú iné kľúče, preto nemôžeme pre univerzálnu identifikáciu dlžníka použiť evidenčné číslo osoby, číslo občianskeho preukazu, ani identifikačné číslo organizácie (IČO). V logickom modely preto tabuľka reprezentujúca dlžníkov obsahuje náhradný kľúč, ktorým je evidenčné číslo dlžníka.





Obrázok 3.27: Transformácia unionu (zjednotenia, kategórie) s náhradným kľúčom

Na obr. č. 3.28 je union reprezentujúci cestné registrované vozidlo. Autobusy aj nákladné autá majú rovnaký kľúč (VIN - vehicle identification number). Preto nie je potrebné pridávať náhradný kľúč. Zaregistrované cestné vozidlá majú evidenčné číslo vozidla (EČV) a dátum registrácie.



Obrázok 3.28: Transformácia unionu (zjednotenia, kategórie) bez náhradného kľúča

3.4 Normalizácia

Cieľom dobrého návrhu je zachovanie všetkých konceptov, zachytených v ER modely (typy entít, vzťahov, atribútov, generalizácia atď.) a minimalizácia redundancie údajov (zamedzenie duplicitného ukladanie údajov a potreby aktualizovania viacerých kópii tých istých údajov, čo by mohlo byť príčinou vzniku chýb).

Doteraz bola kvalita návrhu závislá na intuícii návrhára. Vhodné je ale mať aj formálny spôsob vyhodnotenia kvality návrhu (aké schémy relácií potrebujeme a aké atribúty v nich majú byť). Formálne pravidlá pre návrh sú definované tzv. normálnymi formami. V praxi sa používajú nasledujúce normálne formy, alebo len prvé 3 z nich:

- 1. normálna forma,
- 2. normálna forma,
- 3. normálna forma
- a Boyce-Coddova normálna forma.

Týmito normálnymi formami sa budeme zaoberať. Existujú ale aj vyššie normálne formy. Predtým, než sa pustíme do normálnych foriem, musíme prebrať potrebný teoretický základ.

Ak schéma relačnej databázy spĺňa podmienky normálnych foriem, nie je to zárukou, že jej návrh je dobrý. Normálne formy majú napomôcť dobrému návrhu.

3.4.1 Funkcionálna závislosť

Pri návrhu pomocou normálnych foriem budeme určovať funkcionálnu závislosť medzi množinami atribútov (stĺpcov). Preto si ju v tejto časti definujeme.

Schéma univerzálnej relácie (angl.: universal relational schema) $R(A_1, A_2, \ldots, A_n)$ je schéma relácie, ktorá obsahuje všetky atribúty, nachádzajúce sa v databáze (implicitne predpokladáme, že všetky atribúty schémy univerzálnej relácie R majú odlišné názvy). Neformálne: všetky údaje databázy sú uložene v jednej tabuľke.

Univerzálna relácia r (angl.: universal relation) schémy univerzálnej relácie $R(A_1, A_2, \ldots, A_n)$ je podmnožina karteziánskeho súčinu $dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n)$. Označujeme ju $r(R)$.

Nech $R(A_1, A_2, \ldots, A_n)$ je schéma univerzálnej relácie. Nech X a Y sú množiny atribútov, pre ktoré platí: $X \subseteq \{A_1, A_2, \ldots, A_n\}$, $Y \subseteq \{A_1, A_2, \ldots, A_n\}$. Nech t_1 a t_2 sú n-tice patriace do univerzálnej relácie $r(R)$. **Funkcionálna závislosť** (angl.: functional dependency) označená $X \rightarrow Y$ je podmienka, že pre ľubovoľné t_1 a t_2 platí, že ak $t_1[X] = t_2[X]$, tak $t_1[Y] = t_2[Y]$. Skratka pre funkcionálnu závislosť je **FD** z anglického názvu. X nazývame **determinant**.

Hovoríme, že:

- existuje funkcionálna závislosť z X do Y,
- alebo Y je funkcionálne závislé od X,
- alebo X funkcionálne **determinuje** (určuje) Y,
- alebo Y je funkcionálne determinované (je určené) X.

Neformálne: Nech X a Y sú množiny stĺpcov z tabuľky obsahujúcej všetky stĺpce databázy. Nech t_1 a t_2 sú ľubovoľné riadky tejto tabuľky. Funkcionálna závislosť $X \rightarrow Y$ znamená, že ak sú hodnoty

riadkov t_1 a t_2 v stĺpcoch X rovnaké, tak aj hodnoty v stĺpcoch Y týchto riadkov sú rovnaké. Inými slovami, hodnoty v stĺpcoch X určujú hodnoty v stĺpcoch Y toho istého riadku.

Čiže stĺpce X sú superkľúčom pre časť tabuľky tvorenej stĺpcami X a Y.

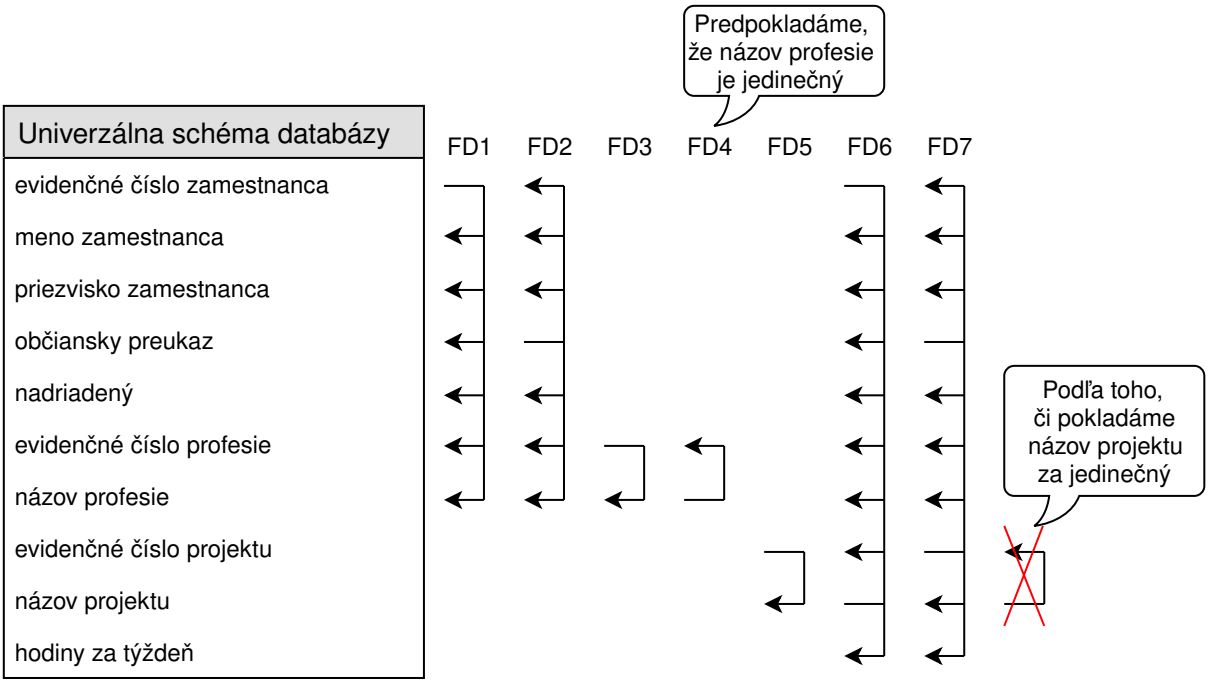
Platnosť $X \rightarrow Y$ neznamená, že platí aj $Y \rightarrow X$.

Atribúty v rôznych schémach relácii môžu byť nazvané rovnako. Preto pri vytváraní schémy univerzálnej relácie môžeme pred názov atribútu vložiť názov schémy relácie. Potom v databáze na obr. č. 3.4 sú napríklad tieto funkčné závislosti:

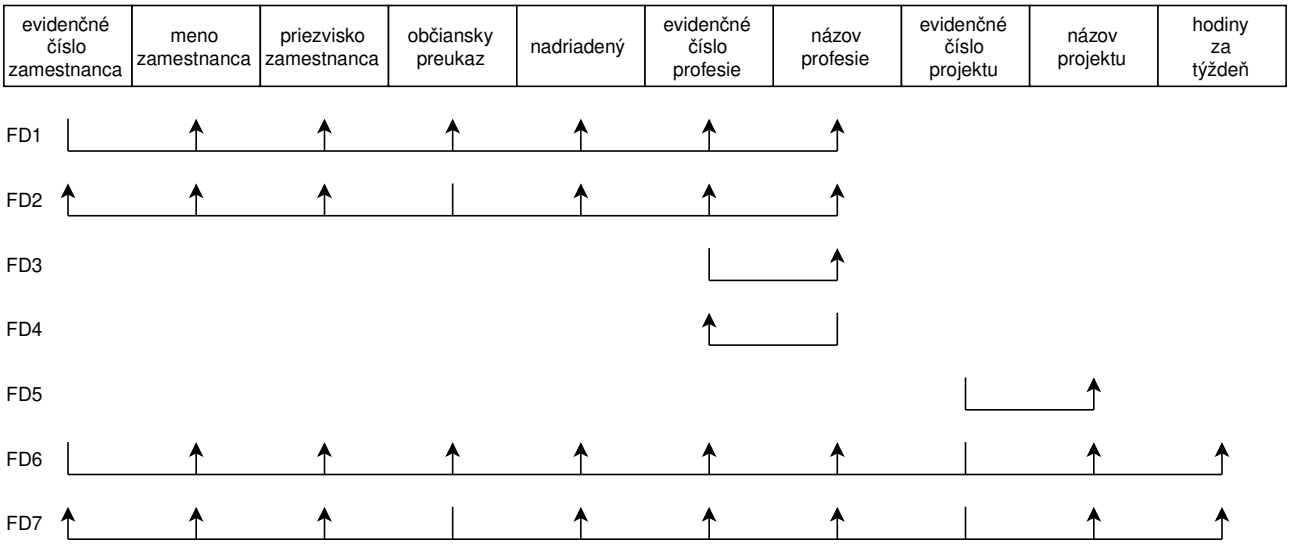
- {`Projekty.evidenčné číslo`} → {`Projekty.názov`, `Projekty.popis`, `Projekty.vedúci`},
- {`Zaradenie na projekt.zamestnanec`, `Zaradenie na projekt.projekt`} → {`Zaradenie na projekt.hodiny za týždeň`},
- {`Zamestnanci.evidenčné číslo`, `Projekty.evidenčné číslo`} → {`Zaradenie na projekt.hodiny za týždeň`},
- {`Zaradenie na projekt.projekt`} → {`Projekt.evidenčné číslo`, `Projekt.názov`},
- {`Zamestnanci.evidenčné číslo`} → {`Zamestnanci.meno`, `Zamestnanci.občiansky preukaz`},
- {`Zamestnanci.občiansky preukaz`} → {`Zamestnanci.evidenčné číslo`, `Zamestnanci.meno`}
- {`Zamestnanci.občiansky preukaz`} → {`Profesie.názov`}.

Funkcionálna závislosť je odvodená od významu atribútov. Je to vlastnosť schémy relácie. Preto funkcionálnu závislosť nemôžeme jednoducho určiť len z hodnôt v relácii.

Príklad grafického zobrazenia funkcionálnych závislostí je na obr. č. 3.29 a 3.30. Návrh databázy na týchto obrázkoch ale nie je dobrý a pre stručnosť neobsahuje všetky atribúty z databázy na obr. č. 3.4.



Obrázok 3.29: Grafické znázornenie funkcionálnych závislosti (v nevhodne navrhnutej databáze)



Obrázok 3.30: Grafické znázornenie funkcionálnych závislosti (v nevhodne navrhnutej databáze)

Triviálna funkcionálna závislosť $X \rightarrow Y$ (angl.: trivial functional dependency) je funkcionálna závislosť, pre ktorú platí $Y \subseteq X$.

Netriviálna funkcionálna závislosť $X \rightarrow Y$ (angl.: nontrivial functional dependency) je funkcionálna závislosť, pre ktorú platí $Y \not\subseteq X$.

Úplne netriviálna funkcionálna závislosť $X \rightarrow Y$ (angl.: completely nontrivial functional dependency) je funkcionálna závislosť, pre ktorú platí, že X a Y neobsahujú rovnaké atribúty.

Úplná funkcionálna závislosť (angl.: full functional dependency) $X \rightarrow Y$ je taká funkcionálna závislosť, pre ktorú platí, že ak odstránime ľubovolný atribút z X, tak prestane byť funkcionálnou závislosťou.

Čiastočná funkcionálna závislosť (angl.: partial functional dependency) $X \rightarrow Y$ je taká funkcionálna závislosť, pre ktorú platí, že môžeme odstrániť niektorý atribút z X tak, aby zostala funkcionálnou závislosťou.

V tabuľke `Zaradenie na projekt` na obr. č. 3.2 je funkcionálna závislosť
{`Zamestnanec`, `Projekt`} → {`Hodiny za týždeň`}
úplná, pretože odstránením atribútu `Zamestnanec` alebo `Projekt`, by prestala byť funkcionálnou závislosťou. V tabuľke `Zamestnanci` je funkcionálna závislosť
{`Evidenčné číslo`, `Meno`} → {`Profesia`}
čiastočná, pretože po odstránení atribútu `Meno` zostane funkcionálnou závislosťou.

Tranzitívna funkcionálna závislosť $X \rightarrow Y$ (angl.: transitive funkctional dependency) je funkcionálna závislosť pre ktorú platí, že existuje množina atribútov Z a platia funkcionálne závislosti $X \rightarrow Z$ a $Z \rightarrow Y$. Príklad tranzitívnej závislosti bude uvedený v časti 3.4.2.4.

3.4.2 Normálne formy

V tejto časti sa budeme zaoberať **procesom normalizácie** (angl.: normalization process), ktorý bude tvorený postupným aplikovaním 1., 2., 3. normálnej formy a Boyce-Coddovej normálnej formy. Proces by mohol pokračovať aj aplikovaním ďalších normálnych foriem, ale v praxi sa zvyčajne používa len 1., 2. a 3. normálna forma, prípadne aj Boyce-Coddova normálna forma.

Ak schéma relácie nespĺňa podmienky normálnej formy, tak sa v procese normalizácie rozloží na viac schém, ktoré podmienky spĺňajú.

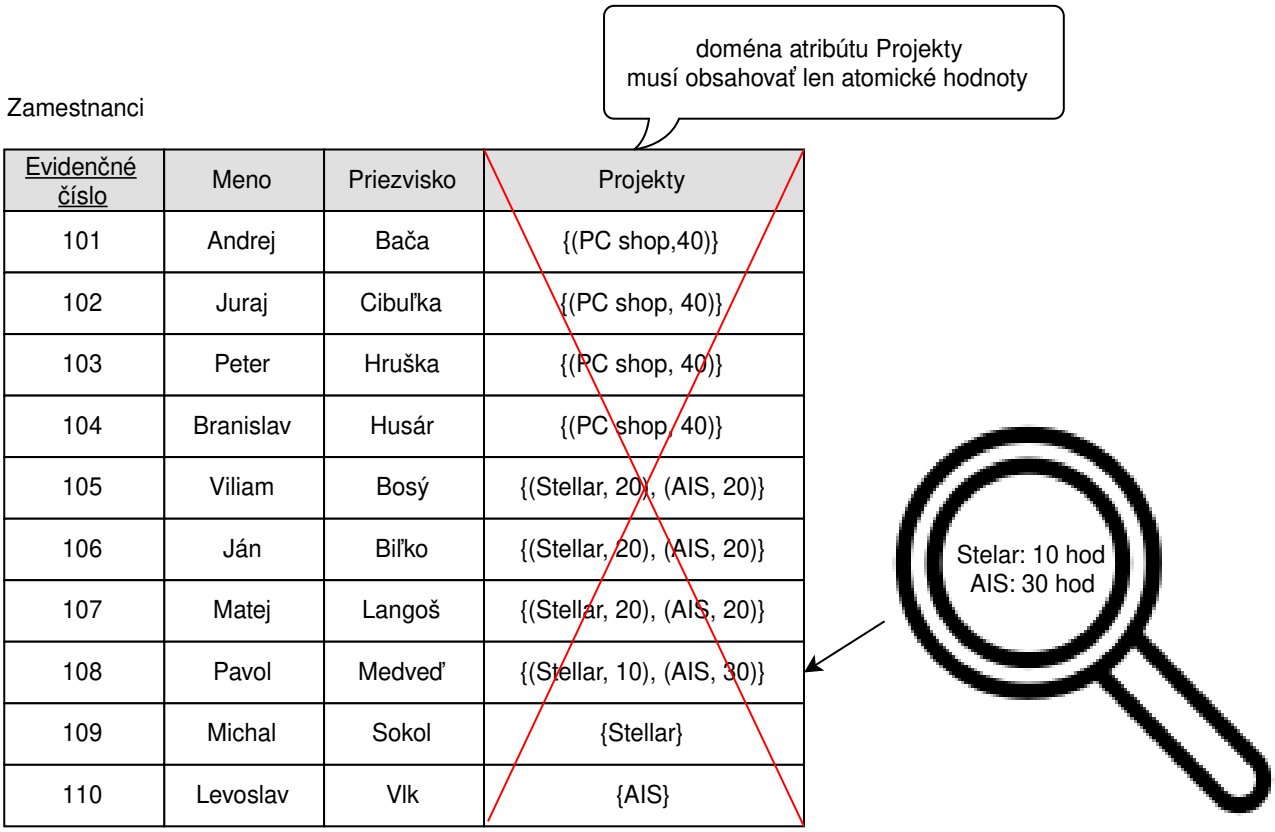
1.normálna forma definuje základnú formu relačnej databázy. 2., 3. a Boyce-Coddova normálna forma sú založené na funkcionálnej závislosti. Vyššie normálne formy sú ale založené na iných závislostiach.

Za procesom normalizácie, môže v niektorých prípadoch nasledovať aj proces denormalizácie. Denormalizácia nie je presným opakom normalizácie. Vykonáva sa vtedy, keď sa zistí, že je potrebné porušiť niektoré pravidlá uvedené v normálnych formách, aby sa zvýšila výkonnosť databázy, aj napriek tomu, že kvalita návrhu bude nižšia.

3.4.2.1 Prvá normálna forma

Schéma relácie je v **1. normálnej forme** (angl.: first normal form), ak definičné obory jej atribútov obsahujú iba atomické hodnoty (to je dané aj definíciou schémy relácie). Označujeme ju skratkou **1NF**. To znamená, že hodnoty v relácii (tabuľke) musia byť atomické.

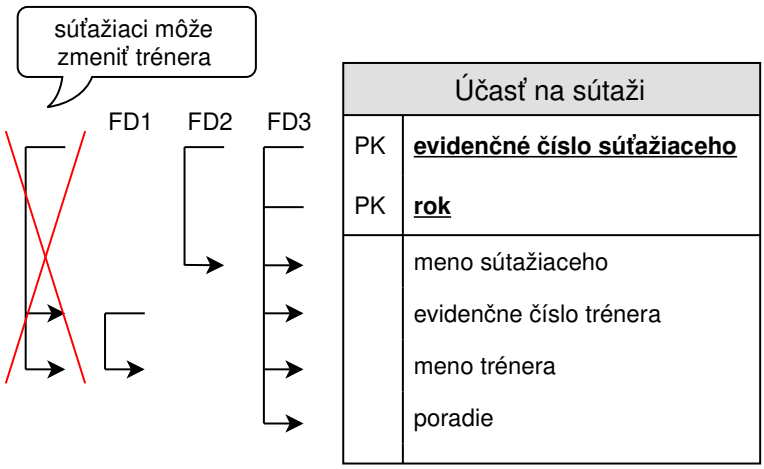
Na obr. č. 3.2 sú hodnoty v tabuľkách atomické. Na obr. č. 3.31 je príklad tabuľky obsahujúcej v stĺpci `Projekty` hodnoty, ktoré nie sú atomické. Konverzia tabuľky do 1NF môže zahŕňať vytvorenie ďalších tabuliek, prípadne ďalších stĺpcov.



Evidenčné číslo	Meno	Priezvisko	Projekty
101	Andrej	Bača	{(PC shop,40)}
102	Juraj	Cibuľka	{(PC shop, 40)}
103	Peter	Hruška	{(PC shop, 40)}
104	Branislav	Husár	{(PC shop, 40)}
105	Viliam	Bosý	{(Stellar, 20), (AIS, 20)}
106	Ján	Biľko	{(Stellar, 20), (AIS, 20)}
107	Matej	Langoš	{(Stellar, 20), (AIS, 20)}
108	Pavol	Medveď	{(Stellar, 10), (AIS, 30)}
109	Michal	Sokol	{Stellar}
110	Levoslav	Vlk	{AIS}

Obrázok 3.31: Tabuľka, ktorá nie je v 1. normálnej forme

Proces normalizácie založenej na primárnom kľúči si ukážeme na príklade databázy obsahujúcej informácie o športovej súťaži. Súťaž sa koná najviac raz do roka. Pre každú účasť na súťaži evidujeme súťažiacich (ich mená) a poradie umiestnenia v súťaži. Súťažiaci majú svojich trénerov. Súťažiaci môže medzi účasťami na súťaži vymeniť trénera. Logický model v 1NF aj s funkcionálnymi závislosťami je znázornený na obr. č. 3.32.



Obrázok 3.32: Logický model v 1NF

Návrh logického modelu nie je dobrý. V praxi by sme pravdepodobne dokázali hneď na začiatku vytvoriť lepší model. Tento model obsahuje len jednu tabuľku z dôvodu vysvetlenia postupu aplikácie normálnych foriem (alebo testovania splnenia podmienok definovaných v normálnych formách).

Po vytvorení modelu v 1NF potrebujeme vybrať primárny kľúč (ak existuje viacero kandidátov).

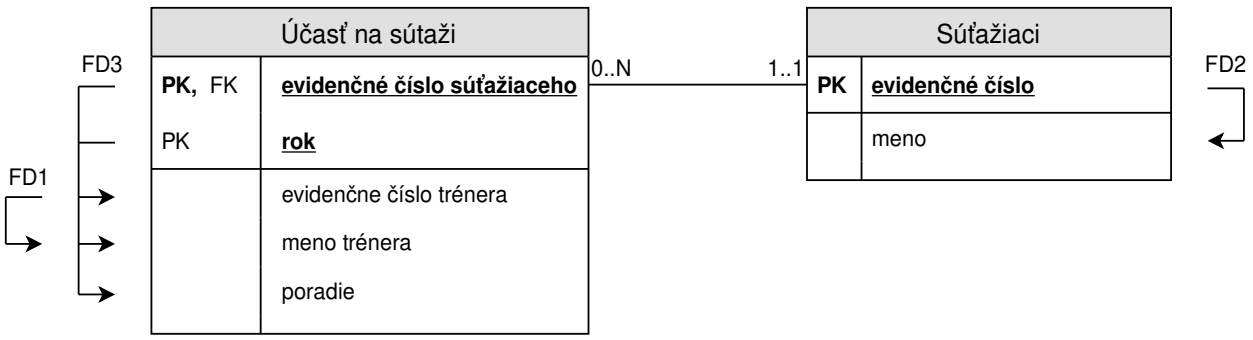
3.4.2.2 Druhá normálna forma založená na primárnom kľúči

2. normálna forma odstraňuje čiastočné funkcionálne závislosti od kľúča a ponecháva len úplné funkcionálne závislosti. Budeme ju označovať skratkou **2NF**.

Najprv si uvedieme definíciu 2NF založenú na primárnom kľúči. Schéma relácie *R* je v **2. normálnej forme** (angl.: second normal form) ak je v 1NF a každý jej atribút, ktorý nie je súčasťou primárneho kľúča, je úplne funkcionálne závislý od primárneho kľúča schémy *R*.

Ak primárny kľúč obsahuje len jeden atribút, tak schéma relácie spĺňa podmienky 2NF.

Model databázy na obr. č. 3.32 nie je v 2NF, pretože FD2 ju porušuje. Model normalizujeme do 2NF rozdelením tabuľky, resp. vytvorením novej tabuľky `Súťažiaci`, do ktorej prenesieme FD2. Potom určíme násobnosti vzťahu medzi tabuľkami `Účasť na súťaži` a `Súťažiaci`. Výsledný diagram po normalizácii je na obr. č. 3.33.



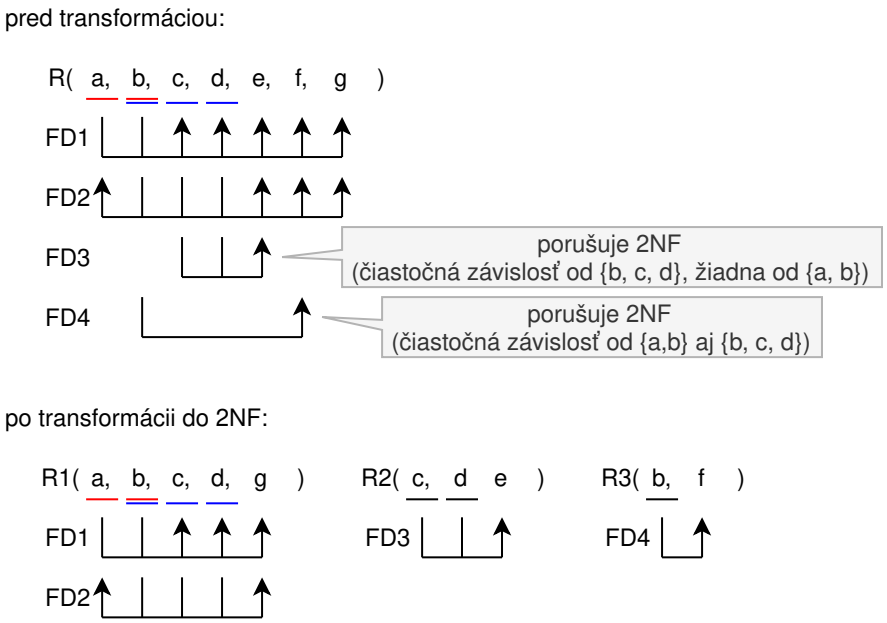
Obrázok 3.33: Logický model v 2NF

3.4.2.3 Všeobecná druhá normálna forma

Všeobecná definícia 2NF zahŕňa každý kandidát na kľúč, nie len primárny kľúč. Schéma relácie je v **2. normálnej forme** ak je v 1NF a každý jej atribút, ktorý nie je súčasťou žiadneho kandidáta na kľúč, je úplne funkcionálne závislý od každého kandidáta na kľúč.

Prevod do 2NF podľa všeobecnej definície si ukážeme na abstraktne zadanom príklade (názvy relácie a jej atribútov budú symbolické). Nech je zadaná schéma relácie $R(a, b, c, d, e, f, g)$, v ktorej platia funkcionálne závislosti (obr. č. 3.34):

- FD1: $\{a, b\} \rightarrow \{c, d, e, f, g\}$
- FD2: $\{b, c, d\} \rightarrow \{a, e, f, g\}$
- FD3: $\{c, d\} \rightarrow \{e\}$
- FD4: $\{b\} \rightarrow \{f\}$



Obrázok 3.34: Príklad transformácie do 2NF

Schéma relácie R má dvoch kandidátov na kľúč $\{a, b\}$ a $\{b, c, d\}$. Na obrázku sú farebne odlíšené. Za primány kľúč zvolíme $\{a, b\}$.

FD3 a FD4 porušujú podmienku všeobecne definovanej 2NF. Pri použití definície 2NF založenej na primárnom kľúči, by FD3 neporušovala podmienku, pretože by sme nebrali do úvahy kandidát na kľúč {b, c, d}. Transformáciu do 2NF vykonáme tak, že pre každú porušujúcu funkcionálnu závislosť vytvoríme novú schému relácie, do ktorej skopírujeme atribúty na jej ľavej strane a presunieme atribúty na jej pravej strane.

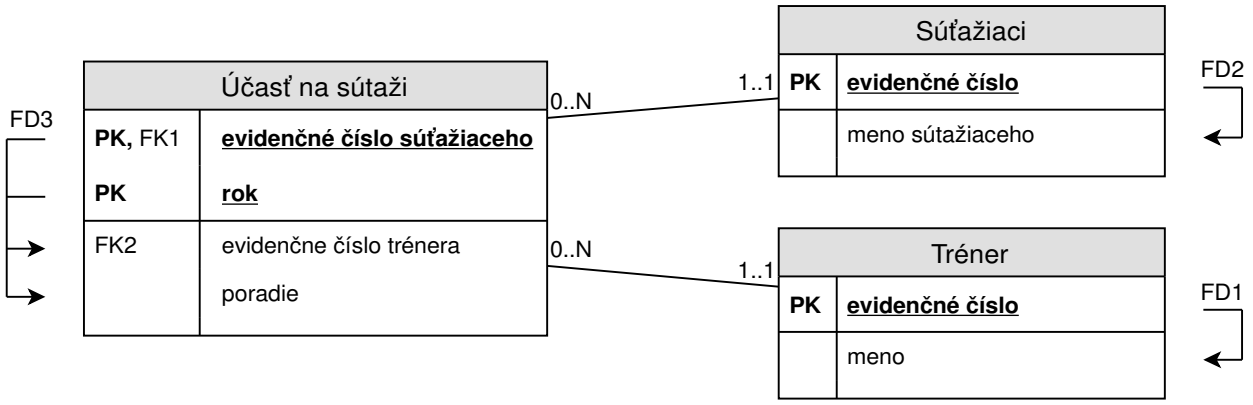
3.4.2.4 Tretia normálna forma založená na primárnom kľúči

3. normálna forma odstraňuje tranzitívne funkcionálne závislosti. Budeme ju označovať skratkou **3NF**.

Budeme pokračovať príkladom normalizácie databázy obsahujúcej informácie o športovej súťaži. Na obr. č. 3.33 je tranzitívna funkcionálna závislosť v tabuľke `Účasť na súťaži`
`{evidenčné číslo súťažiaceho, rok}`→`{meno trénera}`
pretože platia aj nasledujúce funkcionálne závislosti:
`{evidenčné číslo súťažiaceho, rok}`→`{evidenčné číslo trénera}`,
`{evidenčné číslo trénera}`→`{meno trénera}`.
Množina atribútov `{evidenčné číslo súťažiaceho, rok}` je X (v definícii),
množina atribútov `{evidenčné číslo trénera}` je Z (v definícii),
množina atribútov `{meno trénera}` je Y (v definícii).

Najprv si uvedieme definíciu 3NF založenej na primárnom kľúči. Schéma relácie je v **3. normálnej forme** (angl.: third normal form), ak je v 2NF a pre každý jej atribút, ktorý nie je súčasťou primárneho kľúča platí, že nie je tranzitívne závislý od primárneho kľúča.

Logický model na obr. č. 3.33 obsahuje tranzitívnu závislosť v tabuľke `Účasť na súťaži`. Do 3NF ho normalizujeme rozložením tabuľky. Medzi týmito tabuľkami vznikne nový vzťah, pre ktorý určíme násobnosti. Výsledný diagram je na obr. č. 3.35.



Obrázok 3.35: Logický model v 3NF

Pri použití definícii 2NF a 3NF založených na primárnom kľúči, by sme ich poradie v procese normalizácie mohli vymeniť. Historicky je ale dané toto poradie.

3.4.2.5 Všeobecná tretia normálna forma

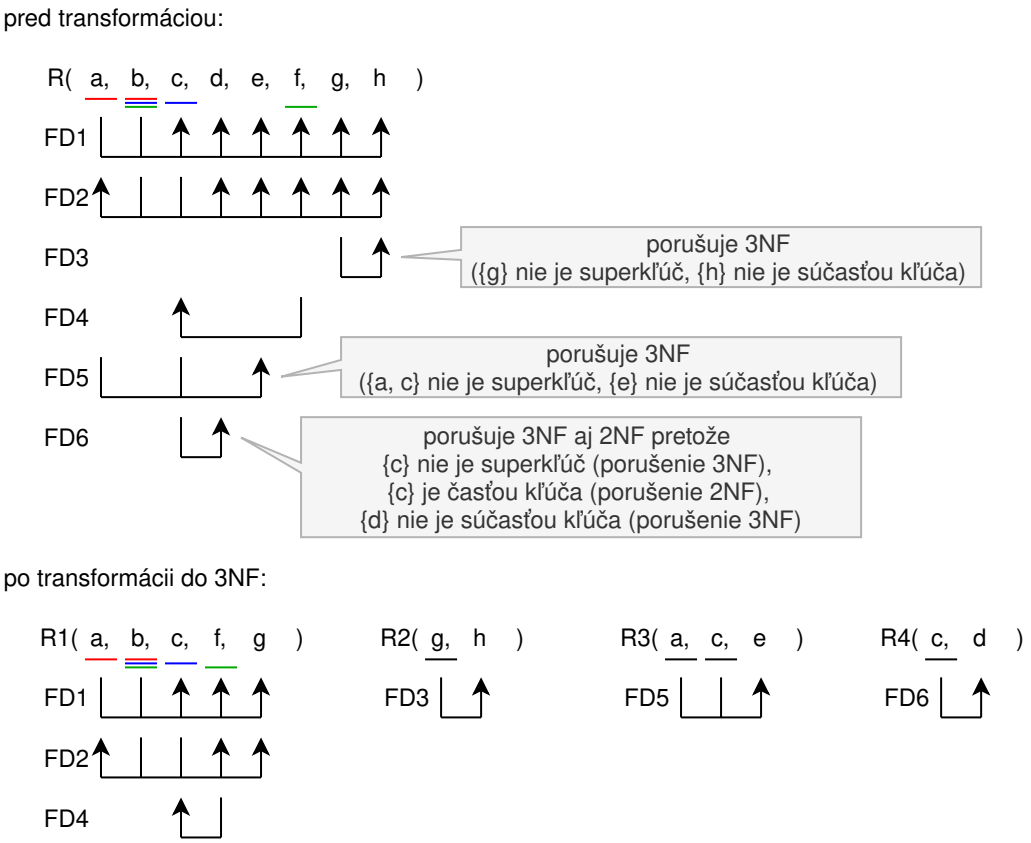
Všeobecná definícia 3NF je prísnejšia, pretože je potrebné testovať každý kandidát na kľúč, nie len primárny kľúč. Schéma relácie je v **3. normálne forme**, ak je v 2NF a pre všetky netriviálne funkcionálne závislosti $X \rightarrow Y$ platí jedna z podmienok:

a) X je superkľúč
b) Y je súčasťou niektorého kandidáta na kľúč.

Alternatívna definícia: Schéma relácie je v **3. normálnej forme**, ak je v 2NF a atribúty, ktoré nie sú súčasťou žiadneho kľúča, nie sú tranzitívne závisle na žiadnom kandidátovi na kľúč.

Prevod do 3NF podľa všeobecnej definície si znova ukážeme na abstraktne zadanom príklade. Nech je zadaná schéma relácie $R(a, b, c, d, e, f, g, h)$, v ktorej platia funkcionálne závislosti (obr. č. 3.36):

- FD1: $\{a, b\} \rightarrow \{c, d, e, f, g, h\}$
- FD2: $\{b, c\} \rightarrow \{a, d, e, f, g, h\}$
- FD3: $\{g\} \rightarrow \{h\}$
- FD4: $\{f\} \rightarrow \{c\}$
- FD5: $\{a, c\} \rightarrow \{e\}$
- FD6: $\{c\} \rightarrow \{d\}$



Obrázok 3.36: Príklad transformácie do 3NF

Schéma relácie R má troch kandidátov na kľúč $\{a, b\}$, $\{b, c\}$ a $\{b, f\}$. Na obrázku sú farebne odlíšené. Za primány kľúč zvolíme $\{a, b\}$.

FD3, FD5 a FD6 porušujú podmienky 3NF. Dôvody sú popísané na obrázku. Schému relácie pretransformujeme do 3NF tak, že pre každú porušujúcu funkcionálnu závislosť vytvoríme novú schému relácie, do ktorej prekopírujeme atribúty na jej ľavej strane a presunieme atribúty na jej pravej strane.

FD1 a FD2 spĺňajú podmienku 3NF, pretože $\{a, b\}$ a $\{b, c\}$ sú kandidáti na kľúč, FD4 spĺňa podmienku 3NF, pretože c je súčasťou kandidáta na kľúč.

Tranzitívna aj čiastočná závislosť porušujú 3NF. Preto môžeme všeobecnú definíciu 3NF aplikovať priamo na schému relácie, ktorá je v 1NF. To sme využili pri riešení príkladu na obr. č. 3.36, v ktorom FD6 porušovala nie len 3NF ale aj 2NF.

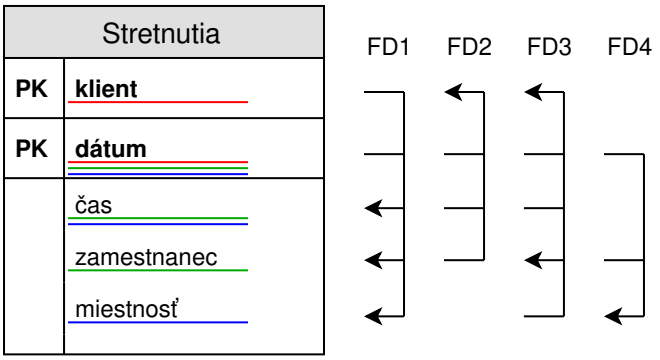
3.4.2.6 Boyce-Coddova normálna forma

Schéma relácie R je v **Boyce-Coddovej normálnej forme** (angl.: Boyce-Codd normal form), ak pre každú netriviálnu funkcionálnu závislosť $X \rightarrow Y$ platí, že X je superkľúč schémy R. Boyce-Coddovu normálnu formu označujeme skratkou **BCNF**.

Definícia BCNF sa líši od definície všeobecnej 3NF tým, že z definície 3NF vynecháva podmienku b). Preto sú podmienky BCNF prísnejšie ako podmienky 3NF.

Model na obr. č. 3.35 spĺňa podmienky BCNF, preto uvedieme iný príklad transformácie modelu z 3NF do BCNF. Budeme navrhovať databázu stretnutí s klientmi, pre personálnu agentúru [5]. V databáze budeme pre jednoduchosť evidovať (každý riadok tabuľky reprezentuje jedno stretnutie):

- evidenčné číslo klienta (atribút `klient`),
- dátum stretnutia (atribút `dátum`),
- čas stretnutia (atribút `čas`),
- evidenčné číslo zamestnanca agentúry (atribút `zamestnanec`),
- evidenčné číslo miestnosti (atribút `miestnosť`).

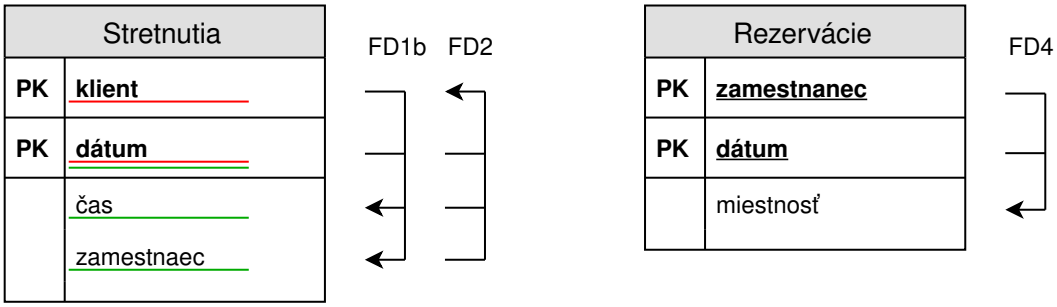


Obrázok 3.37: Model databázy stretnutí v 3NF

Budeme predpokladať, že každý klient príde na stretnutie najviac 1 krát za deň. Zamestnanec si na stretnutia môže rezervovať maximálne jednu miestnosť na deň, ale jedna miestnosť môže byť na jeden deň rezervovaná viacerými zamestnancami (na iné časy). Zamestnanec sa môže stretnúť s viacerými klientmi za deň. Diagram databázy je na obr. č. 3.37. Databáza je v 3NF. Na obrázku sú znázornené aj funkcionálne závislosti. Kandidátmi na kľúč sú:

- (`klient`, `dátum`),
- (`zamestnanec`, `dátum`, `čas`),
- (`miestnosť`, `dátum`, `čas`).

Za primárny kľúč je zvolený (`klient`, `dátum`). Funkcionálne závislosti FD1, FD2, FD3 spĺňajú podmienku BCNF, pretože ich determinanty sú kandidáti na kľúč. FD4 ale porušuje podmienku BCNF, pretože jej determinant nie je superkľúč. Nedostatkom návrhu je redundancia údajov, pretože ak má zamestnanec viac stretnutí počas dňa, tak je informácia o miestnosti zapísaná viac krát. Ak by zamestnanec na daný deň vymenil rezervovanú miestnosť, tak by bolo potrebné informáciu o miestnosti zmeniť v každom zázname o stretnutí.



Obrázok 3.38: Model databázy stretnutí v BCNF

Normalizáciou do BCNF vytvoríme novú tabuľku, do ktorej presunieme FD4. Výsledný model je zobrazený na obr. č. 3.38. Touto úpravou sme odstránili redundanciu, ale stratili funkcionálnu závislosť FD3 (`miestnosť`, `dátum`, `čas`) → (`klient`, `zamestnanec`). Ak je táto funkcionálna závislosť potrebná pri vyhľadávaní údajov, tak je lepšie ponechať model v 3NF. Preto pri návrhu treba zvážiť, či je transformácia do BCNF vhodná.

Transformáciu do BCNF môžeme vykonať pomocou nasledujúceho postupu.

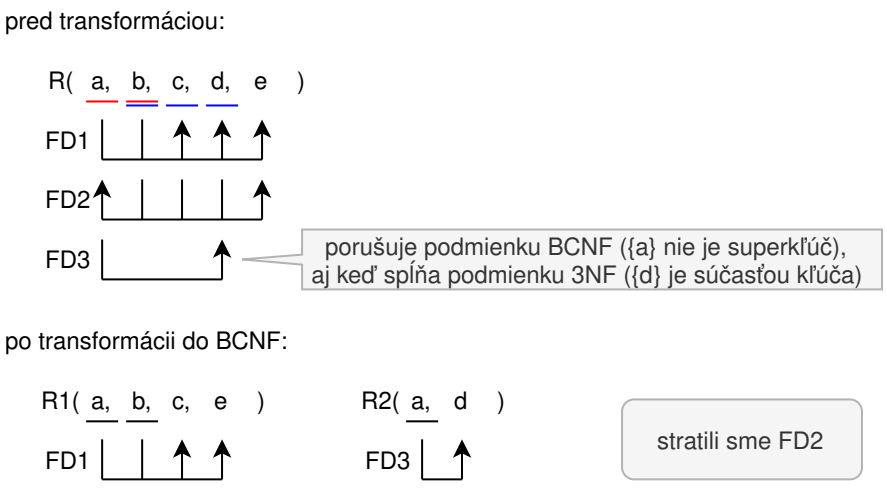
1. Najprv testujeme, či je schéma relácie *R* v BCNF. Pri testovaní zisťujeme, či je determinant každej funkcionálnej závislosti kandidátom na kľúč.
2. Pre každú funkcionálnu závislosť $X \rightarrow Y$, ktorá porušujú BCNF, rozložíme schému relácie *R* na dve schémy:
   R-Y (rozdiel - obsahuje také atribúty *R*, ktoré nepatria do *Y*),
   $X \cup Y$ (zjednotenie - obsahuje atribúty, ktoré patria do *X*, alebo do *Y*, alebo do *X* aj *Y*).
3. Ak niektorá z rozložených schém nespĺňa BCNF, tak na ňu znova aplikujeme ten istý postup.

Aj pre transformáciu do BCNF uvedieme abstraktne zadaný príklad. Nech je zadaná schéma relácie $R(a, b, c, d, e)$, v ktorej platia funkcionálne závislosti (obr. č. 3.39):

- FD1: $\{a, b\} \rightarrow \{c, d, e\}$
- FD2: $\{b, c, d\} \rightarrow \{a, e\}$
- FD3: $\{a\} \rightarrow \{d\}$

Schéma relácie *R* má dvoch kandidátov na kľúč $\{a, b\}$ a $\{b, c, d\}$. Na obrázku sú farebne odlíšené. Za primány kľúč zvolíme {a, b}.

FD3 nespĺňa podmienku BCNF, pretože $\{a\}$ nie je superkľúč. Schému relácie *R* pretransformujeme do BCNF vytvorením novej schémy relácie *R*2, do ktorej skopírujeme atribút *a* a presunieme atribút *d*. Touto transformáciou sme ale stratili FD2.



Obrázok 3.39: Príklad transformácie do BCNF

Zoznam použitej literatúry

1: Song, I.-Evans, M.-Park E. K. *A Comparative Analysis of Entity-Relationship Diagrams*. 1995
2: Elmasri, R.-Navathe, S. B. *Fundamentals of Database Systems*. 7. vyd. Pearson, 2016. 1242 s. ISBN 978-0-13-397077-7.
3: Garcia-Molina, H.-Ullman, J. D.-Widom, J. *Database Systems: The Complete Book*. 2. vyd. Pearson, 2009. 1248 s. ISBN 978-0-13-606701-6.
4: Camps, R. *Transforming N-ary relationships to database schemas: an old and forgotten problem*. 2002
5: Connolly, T. M.-Begg, C. E. *Database systems: A Practical Approach to Design, Implementation, and Management*. 6. vyd. Pearson Education, 2015. 1440 s. ISBN 978-1-292-06118-4.