Trnavská univerzita v Trnave

Pedagogická fakulta

Katedra matematiky a informatiky

Modelovanie databázových systémov

Ing. Milan Štrbo, PhD.

2020

Názov publikácie: **Modelovanie databázových systémov**

Autor: Ing. Milan Štrbo, PhD.

Recenzenti: Ing. Katarína Pribilová, PhD.

doc. Ing. Andrej Trnka, PhD.

Typografická úprava: Mgr. Ing. Roman Horváth, PhD.

Vydavateľské údaje:

Dokument neprešiel jazykovou úpravou.

ISBN 978-80-568-0340-0

9 788056 803400

Obsah

Zoznam použitých skratiek

ANSI/SPARC – American National Standards Institute – Standard Planning and Requirements Committee

AP – aplikačný program

BCNF – Boyce-Coddova normálová forma

CK – cudzí kľúč, zvykne sa označovať aj ako „foreign key“ so skratkou FK

DB – databáza

DBS – databázový systém

E-R diagram – entitno-relačný diagram

ES – externá schéma

FS – fyzická schéma

ID – identifikačné číslo (z angl. identification)

IS – informačný systém

KS – konceptuálna schéma

NF – normálová forma

PK – primárny kľúč

RDM – relačný údajový model (z angl. relation data model)

SD – schéma údajov (z angl. D – data)

SRBÚ – systém riadenia bázy údajov (prípadne SRBD – dát)

STD – stavový diagram (z angl. state transition diagram)

Zoznam obrázkov a tabuliek

Úvod

So slovom databáza (DB) sa v súčasnosti stretol už takmer každý človek pracujúci s výpočtovou technikou. Pod týmto slovom si obyčajne predstavujeme množinu informácií usporiadaných podľa určitých pravidiel, ktoré sú spracovávané na aplikačných programoch. Áno, databáza je naozaj množina štruktúrovaných údajov. Slúži na uloženie, zoradenie a triedenie údajov rôzneho typu. Vďaka presne určenej štruktúre umožňuje ľahké vyhľadávanie a triedenie aj pri veľkom množstve obsiahnutých údajov. Databázy môžu obsahovať informácie o používateľoch, zamestnancoch, produktoch, objednávkach, študentoch alebo o čomkoľvek inom. Sú zdrojom údajov aj pre aplikácie a umožňujú, aby tieto údaje mohli byť spracované a neskôr vyhľadané. V súčasnosti už prakticky každá organizácia používa svoj vlastný databázový systém (DBS), ktorý je presne vytvorený s ohľadom na ich požiadavky.

S aktuálnym nárastom používania databázových systémov vzniká potreba správneho a kvalitného návrhu. Pri väčšom objeme údajov môže reprezentácia relačnej databázy jedinou nesprávnou tabuľkou viesť k problémom. Z dôvodu aby sme mohli prácne získané údaje využívať, nestačí ich iba vlastniť, ale treba ich aj požadovaným spôsobom spravovať. Myslíme tým pridávať nové údaje, aktualizovať už existujúce či odstraňovať nepotrebné údaje. Na rýchlu orientáciu potrebujeme údaje triediť a zoraďovať podľa rôznych pravidiel (napríklad zoradiť zamestnancov podľa mesta, v ktorom bývajú) a vyhľadávať údaje z databázy podľa konkrétnych kritérií (vyhľadať telefónne číslo podľa priezviska alebo časti jeho priezviska). Práve na tento účel slúži databázový systém, ktorý spomínané údaje zahŕňa a súčasne umožňuje manipuláciu s nimi.

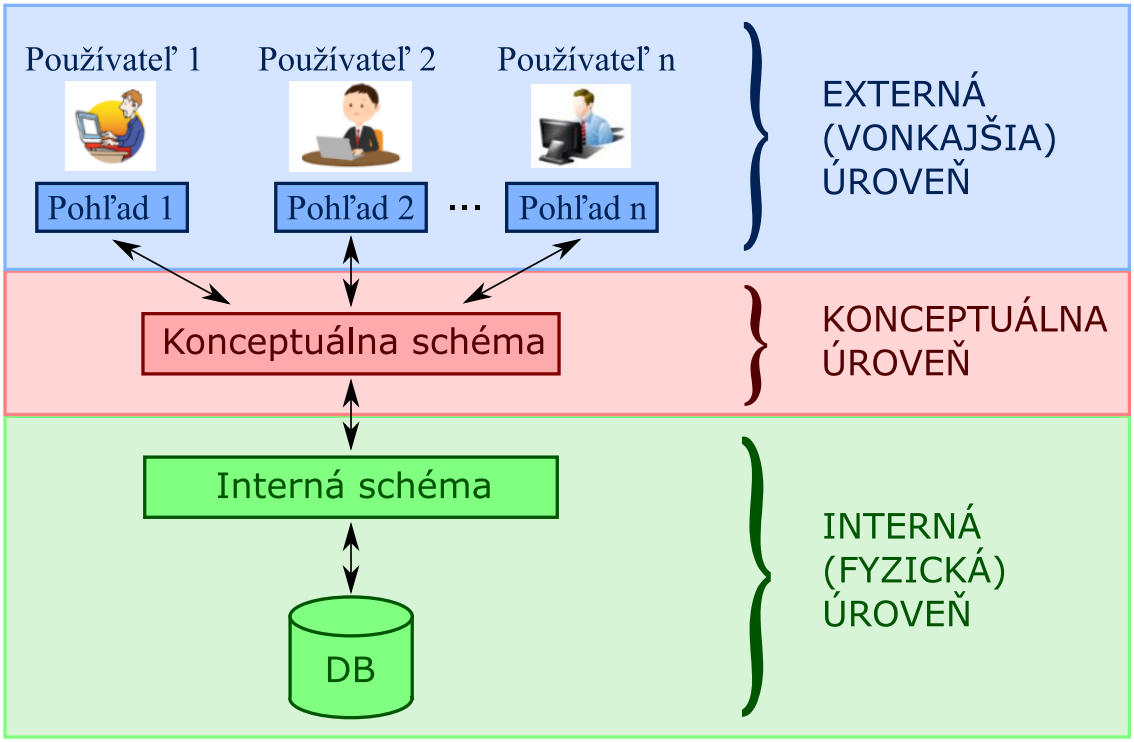
Základom požiadaviek pre budúcich používateľov databázového systému je kvalitný návrh. Cieľom učebnice je preto poskytnúť čitateľovi informácie práve z oblasti modelovania databázových systémov. Učebnicu sme rozdelili do siedmich kapitol, ktoré na seba navzájom nadväzujú. Začíname teoretickými východiskami z oblasti architektúry databázových systémov, následne prechádzame cez analýzu požiadaviek na databázový systém až ku vypracovaniu údajového modelu. Poslednou, siedmou, kapitolou je konceptuálny návrh jednoduchého školského databázového systému. Čitateľ po preštudovaní tejto učebnice bude schopný vypracovať konceptuálny návrh databázového systému pre ľubovoľnú organizáciu. Učebnica je vytvorená najmä pre študentov učiteľstva informatiky, ale rovnako pre ľudí zaujímajúcich sa o túto problematiku.

1 Architektúra databázových systémov

Môžeme povedať, že architektúra databázových systémov je špecializovaná disciplína informatiky, ktorá sa zaoberá navrhovaním štruktúry databázového systému a jeho zložiek, ktoré majú stanovené funkcie a vzájomné vzťahy. Samotná architektúra poskytuje spôsoby abstraktného opisu štruktúry databázových systémov a princípy a pravidlá, ktorými sa dynamicky riadi ich návrh a vývoj.

K najznámejším patrí trojúrovňová architektúra, ktorú poznáme pod názvom **ANSI/SPARC** (**A**merican **N**ational **S**tandards **I**nstitute – **S**tandard **P**lanning **a**nd **R**equirements **C**ommittee). Prvý raz bola predstavená v roku 1975 a odvtedy sa stala štandardom pri návrhu databázových systémov. Predstavuje všeobecný databázový koncept na vysvetlenie štruktúry databázového systému. Navrhnutá bola na dosiahnutie oddelenia programov od údajov, na podporu viacerých používateľských pohľadov a na použitie údajového slovníka (katalógu údajov) slúžiaceho na uschovanie opisu databázy (schémy).

ANSI/SPARC architektúra sa usiluje vytvoriť tri úrovne abstrakcie nad údajmi. Jej úsilím je oddeliť skutočné uloženie údajov v hardvéri od práce so samotnými údajmi (tým myslíme: pridávanie, úprava a mazanie údajov).



Obrázok č. 1.1 – ANSI/SPARC architektúra databázových systémov.

Jej hlavné črty sú zhrnuté v nasledujúcich bodoch [1]:

- **Vytváranie pohľadov** – pohľad slúži na sprístupnenie len tých údajov, ktoré sú pre určitú skupinu používateľov databázového systému podstatné a prípustné.
- **Schovať detaily fyzického uloženia údajov** – používateľ nepotrebuje vedieť, akým spôsobom sú údaje na fyzickom médiu uložené.

- **Zmena fyzického uloženia nemá vplyv na zobrazenie údajov používateľovi** – ak by administrátor zmenil spôsob fyzického uloženia údajov (pridal by napríklad index nad nejakou skupinou údajov kvôli zrýchleniu prístupu), nemalo by to ovplyvniť zobrazenie údajov používateľovi.
- **Zmena hardvéru nemá vplyv na štruktúru uloženia údajov** – príkladom môže byť výmena pevného disku alebo počítača, na ktorom je nainštalovaný databázový server.
- **Zmena logickej reprezentácie údajov nemá vplyv na zobrazenie údajov používateľom** – pridanie, zmena alebo odstránenie údajových štruktúr nesmie ovplyvniť zobrazenia údajov používateľom.

ANSI/SPARC architektúra sa skladá z troch úrovní, ktoré sú opísané prostredníctvom tzv. schém alebo modelov. Tie opisujú štruktúru alebo formát údajov na každej úrovni architektúry. Jednotlivé úrovne (zvykne sa používať aj termín vrstvy) tejto architektúry sú **externá, konceptuálna a interná.**

1.1 Externá (vonkajšia) úroveň

Externá schéma je implementačne nezávislá množina údajov opisujúca používateľské pohľady aplikácie. V odbornej terminológii sa často používa termín **pohľad** (view). Predstavuje teda čiastočný pohľad na množinu objektov používaných používateľmi alebo skupinou používateľov. To znamená, že zahŕňa množstvo externých schém alebo používateľských pohľadov pričom každá z týchto schém opisuje databázový pohľad jednej skupiny používateľov databázy. Z toho vyplýva, že každý používateľ má svoj vlastný pohľad na databázu, a ten zahŕňa výlučne iba tie údaje, ktoré sú pre konkrétneho používateľa dôležité, a s ktorými potom následne ďalej pracuje [2].

Externá úroveň súvisí s individuálnymi pohľadmi jednotlivých používateľov aplikácie, ktoré sú zjednotené v konceptuálnej úrovni do všeobecného používateľského pohľadu. Pri analýze požiadaviek súvisiacich s návrhom databázy aplikácie môže existovať veľké množstvo externých pohľadov, ktoré obsahujú viac alebo menej abstraktnú reprezentáciu časti databázy. Spresnenie samotnej databázy sa vlastne vytvára definíciou konceptuálneho pohľadu, ktorý je abstraktnou reprezentáciou celého modelu. Až konceptuálny pohľad je transformovaný do internej úrovne, ktorá je reprezentovaná interným pohľadom. Tieto pohľady (externý, konceptuálny, interný) sú vlastne zapísané v schémach, ktorých zápis je často umožnený v niektorých z programovacích jazykov.

Zaujímavé je aj použitie externých schém z hľadiska ochrany údajov, pretože sprístupnenie len časti údajového modelu predstavuje ochranu pred neoprávneným prístupom do inej časti databázy. Je to spôsobené tým, že každý používateľ má možnosť prístupu len do tej časti databázy, ktorá je opísaná v externej schéme [3].

To znamená, že existuje skupina používateľov s rôznymi prístupovými právami do databázy (používa sa tiež pojem **externé organizácie**). Tie poskytujú pohľad iba na tú časť množiny údajov, ktoré sú pre konkrétneho používateľa prístupné a poskytujú mu určitý komfort v rámci využívania funkcií [4].

Ako príklad môžeme uviesť klasický univerzitný informačný systém (IS), kde používateľ „Učiteľ“ bude mať možnosť pracovať s inými údajmi ako používateľ „Študent.“ To znamená, že používateľ „Študent“ sa môže prihlásiť na konkrétny termín skúšky (ktorý vypísal používateľ „Učiteľ“), ale nemôže tento termín zmeniť. Prípadne „Študent“ si môže pozrieť výsledok hodnotenia jeho skúšky, ale nemôže si pozrieť výsledok hodnotenia skúšky iného študenta, prípadne si nemôže sám sebe zapísať výslednú známku do informačného systému. Naopak, môže to uskutočniť používateľ „Učiteľ.“

Na druhej strane to však neznamená, že nie je vylúčená reprezentácia rovnakých údajov v rôznych pohľadoch a pre rôznych používateľov. Povedzme, že jeden používateľ si zobrazí dátum vo formáte DD/MM/RRRR, druhý si zobrazí rovnaký dátum vo formáte MM/DD/RRRR a tretí si vo svojom pohľade z toho istého dátumu s pomocou vhodnej funkcie vyberie iba konkrétny rok. Všetci traja používatelia v tomto prípade pracovali s rovnakým údajom (dátumom), avšak každý ich pohľad na ten istý údaj je rozdielny. Okrem toho používateľské pohľady môžu obsahovať aj dopočítané údaje, ktoré nie sú súčasťou databázy [6]. Napríklad tým myslíme, ak si učiteľ (používateľ systému) nechá zobraziť priemernú známku, ktorú dosiahli študenti zo skúšky alebo si účtovník spoločnosti nechá zobraziť priemernú mzdu zamestnancov na konkrétnom oddelení, konkrétnej pracovnej pozícii a podobne.

1.2 Konceptuálna úroveň (schéma)

Konceptuálna schéma (zvykneme označovať aj ako logická schéma) je implementačne nezávislá množina údajov opisujúca údajový model. Je určená na návrh štruktúry celej databázy a jej zobrazenie pre všetkých zainteresovaných používateľov databázového systému. Konceptuálnu úroveň môžeme chápať ako všeobecný pohľad na databázu. Táto úroveň opisuje, aké údaje sú v databáze uložené a aké sú vzťahy medzi nimi. Obsahuje logickú štruktúru celej databázy v takej podobe, ako ju vidí správca databázy. Je to vlastne komplexný pohľad na údaje organizácie bez nutnosti uvažovania o uložení údajov [2].

Konceptuálna úroveň:

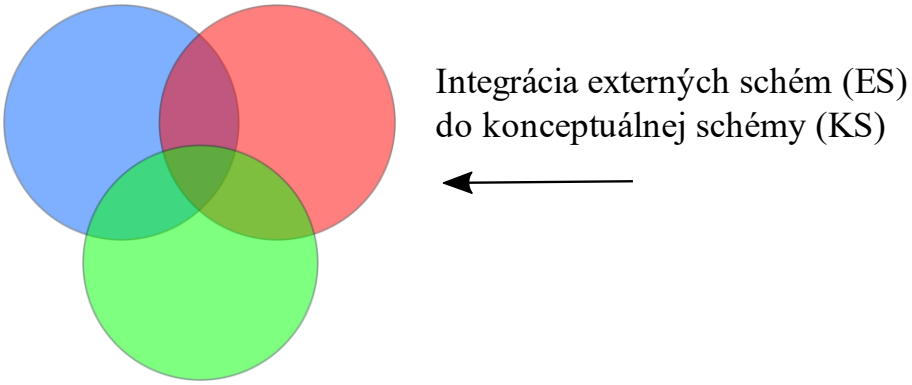
- Reprezentuje sémantickú úroveň (význam) údajov.
- Rieši bezpečnosť.
- Rieši integritu (celistvosť) – sémantickú korektnosť údajov, čiže to, že hodnoty údajov sú správne, konzistentné a aktuálne.
- Vyjadruje atribúty (vlastnosti) objektov, vzťahy, podmienky.

Nešpecifikuje, akým spôsobom budú údaje fyzicky uložené, ale zahŕňa definíciu, aký údajový typ bude priradený k údajom, aké vzťahy budú medzi týmito údajmi a tiež aké integritné obmedzenia sú kladené na jednotlivé údaje. Nad týmto modelom má administrátor databázy najväčšiu kontrolu. Na to slúži vysokoúrovňový údajový model alebo implementačný údajový model [1].

Konceptuálnu schému môžeme tvoriť:

- **Podnikovým prístupom** – je nezávislý od jednotlivých používateľských pohľadov a mal by čo najvernejšie odrážať konkrétnu realitu systému.
- **Integračným prístupom** – je zjednotením rôznych používateľských pohľadov na údaje.

Konceptuálna schéma nie je súčtom jednotlivých externých schém, ale vzniká ich integráciou. Databázu navrhujeme tak, aby sme vylúčili duplicitu (redundanciu) spoločných údajov.



Obrázok č. 1.2 – Integrácia externých schém do konceptuálne schémy [4].

Konceptuálna schéma opisuje aké údajové štruktúry v databáze existujú a ako sú prepojené (podstatné mená opisujú štruktúru, slovesá predstavujú funkcie) [4].

Konceptuálny pohľad reprezentuje celý informačný pohľad na databázu odvodený z externých pohľadov. Je potrebné si uvedomiť, že konceptuálny pohľad sa môže dosť výrazne líšiť od spôsobu používateľských definícií v externých pohľadoch. Na externej úrovni sú pohľady reprezentované tak, ako si používateľ želá, aby mohli byť. Naopak, na konceptuálnej úrovni sú definované tak, aká je skutočnosť. Konceptuálny pohľad teda pozostáva z rôznych výskytov konceptuálnych záznamov (napr. záznamy o katedrách, zamestnancoch, študentoch, predmetoch, a podobne). Samotný konceptuálny pohľad je definovaný v konceptuálnej schéme, ktorá zahŕňa definíciu každého konceptuálneho záznamu. Takisto každá konceptuálna schéma môže byť zapísaná prostredníctvom niektorého z programovacích jazykov [3].

Pri vytváraní konceptuálnej schémy sa v súčasnosti používa entitno-relačné (E-R) modelovanie. Pod pojmom E-R modelovanie sa rozumie proces, ktorý prebieha najmä na začiatku vytvárania informačných systémov a jeho výsledkom je opis informácií, ktoré sa majú v databáze ukladať. Grafickým výstupom je tzv. E-R diagram, v ktorom sú opísané jednotlivé entity, ich atribúty a vzťahy medzi jednotlivými entitami [1]. Entitno-relačnému modelovaniu sa budeme podrobnejšie venovať v tretej kapitole tejto učebnice.

1.3 Interná (fyzická) úroveň

Zvykneme ju označovať aj ako vnútorná či fyzická úroveň (schéma). Predstavuje informácie o spôsobe uloženia údajov na fyzickej úrovni. Je implementačne závislá množina údajov, ktorá opisuje údajové štruktúry a prístupové metódy uložených údajov v externej pamäti. Vnútorná schéma používa fyzický údajový model a opisuje detaily úschovy údajov a prístup k nim pre databázu [2].

Uloženie údajov je podstatou internej úrovne. Ide totiž o fyzickú reprezentáciu databázy v počítači, čím myslíme údajové štruktúry a organizáciu súborov uložených na pamäťových jednotkách. Jej význam spočíva v dosahovaní optimálneho využitia ukladacieho priestoru a tiež optimálneho výkonu databázy počas činnosti. Interná úroveň vytvára tiež rozhrania k prístupovým metódam operačných systémov, ktoré pod ňou na fyzickej úrovni spravujú napríklad radenie záznamov na pamäťových médiách a podobne [6].

Interná schéma teda predstavuje informácie o spôsobe uloženia údajov na fyzickej úrovni. To znamená, že schéma obsahuje opis údajových štruktúr, typov súborov (v prípade ak sa používajú), formáty (údajové typy) záznamov, definíciu polí, prístupové metódy, ďalej obsahuje informácie o použití a špecifikácií indexových súborov, index sekvenčných súborov, hashovacie mechanizmy a iné charakteristiky

potrebné na fyzickú implementáciu databázy. Interná schéma podobne ako ostatné schémy využíva niektorý z programovacích jazykov na svoj opis a zároveň umožní (hlavne systémovým programátorom) špeciálny prístup k údajom, ktorý sa využíva v určitých výnimočných situáciách [1].

Pre každý databázový systém je fyzická schéma len jedna a vytvárajú ju programátori databázových systémov. Medzi základné modely fyzickej úrovne ANSI – SPARC architektúry patria [4]:

- **„Flat file" model** – údaje sú ukladané do dvojrozmerného poľa (tabuľky). Predpokladá sa, že hodnoty v stĺpcoch tabuľky sú rovnakého typu a jednotlivé hodnoty v riadku majú medzi sebou nejaký vzťah.
- **Hierarchický model** – údaje sú organizované v stromovej štruktúre. Pri tejto štruktúre je možné modelovanie vzťahov 1 : 1 a 1 : N. Nevýhodou je redundancia jednotlivých hodnôt.
- **Sieťový model** – vychádza z hierarchického modelu, ktorý rozširuje možnosť, že každá položka nachádzajúca sa v stromovej štruktúre môže mať viacerých predkov a viacerých nasledovníkov, to znamená možnosť modelovania vzťahov aj M : N. Výsledkom je odstránenie redundancie z databázy.
- **Relačný model** – vychádza z teórie množín, pre ktorú sú základné termíny relácia, atribút a doména. Reprezentáciou relácie v implementácii databázového systému sú tabuľky, ktoré zastupujú určitú entitu v skutočnom svete. Jednotlivé stĺpce tabuľky sú v teórii množín atribútmi a sú vyjadrením určitej vlastnosti alebo stavu entity v reálnom svete. Množina hodnôt, ktorá sa v stĺpci môže vyskytovať sa nazýva doména.
- **Objektovo orientovaný model** – usiluje sa o to aby reprezentácia údajov v databáze bola čo najpríbuznejšia s reprezentáciou údajov v aplikačnom programe.

1.4 Zobrazenia (mapovanie)

V architektúre databáz poznáme dve úrovne zobrazenia:

- **Externo-konceptuálne zobrazenie** – definuje transformáciu medzi externým pohľadom a konceptuálnym pohľadom. Na tejto úrovni sa identifikujú údajové objekty, zjednotia sa údajové typy, určí sa veľkosť údajových polí, definícia mien objektov, atribútov, relácií… Zmeny na konceptuálnej úrovni by mali byť nezávislé od externých pohľadov a zabezpečujú vlastne logickú nezávislosť údajov.
- **Konceptuálno-interné zobrazenie** – vyjadruje transformáciu konceptuálneho pohľadu na interný pohľad a určuje konkrétnu reprezentáciu databázy na internej úrovni. Zmeny na internej úrovni by mali byť nezávislé od konceptuálneho modelu a zabezpečujú vlastne fyzickú nezávislosť údajov [3].

1.5 Postup návrhu údajovej štruktúry

Postup návrhu údajovej štruktúry podľa ANSI/SPARC architektúry vykonávame podľa nasledujúcich bodov:

1. Zistiť pohľady jednotlivých typov používateľov – externé schémy (ES).
2. Integrácia externých schém (ES) do konceptuálnej schémy (KS).

3. Transformácia konceptuálnej schémy (KS) do schémy údajov (SD), ktorú vykonávame s pomocou skriptov.
4. Schéma údajov (SD) sa preloží do fyzickej schémy (FS) [4].

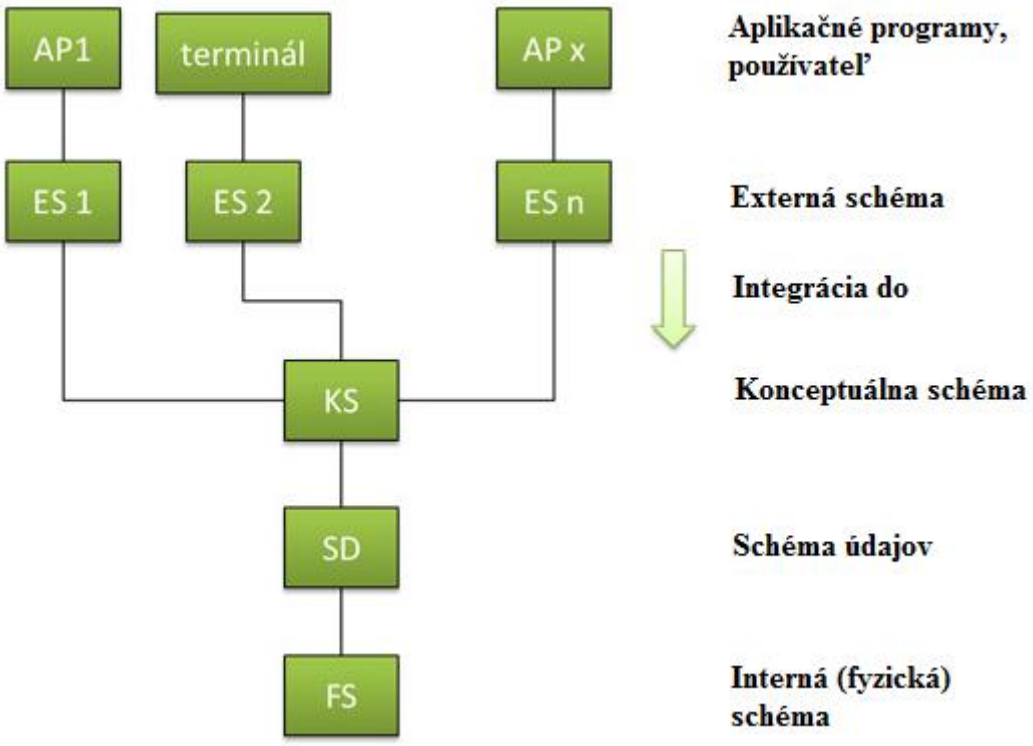
Konceptuálne modely sú prostriedky na návrh údajových schém. Použitím konceptuálneho modelu vznikne konceptuálna schéma (KS). Konceptuálna schéma nerieši funkčnú analýzu aplikácie. Štandardom v nástrojoch konceptuálneho modelovania je entitno-relačný model (E-R diagram) [8].

Schéma údajov je transformovaná KS, kedy vzniká **logický model.** Logický model závisí od použitého údajového modelu.

Interná (fyzická) schéma vzniká modifikáciou SD (logického modelu) pridaním špecifických charakteristík prostriedku, ktorým je realizovaný údajový model v počítači [4].

Postup pri navrhovaní databázy zahŕňa:

1. analýzu požiadaviek používateľov,
2. návrh konceptuálnej schémy databázy,
3. návrh logickej štruktúry databázy,
4. návrh fyzického interného modelu databázy.



Obrázok č. 1.3 – Postup návrhu údajovej štruktúry [4].

Zhrnutie 1. kapitoly

ANSI/SPARC architektúra je rozdelená do troch úrovní:

- **Externá úroveň** – predstavuje čiastočný pohľad jednotlivých používateľov alebo skupinou používateľov DBS na množinu objektov nachádzajúcich sa v databáze.
- **Konceptuálna úroveň** – opisuje, aké údaje sú v databáze uložené a aké sú vzťahy medzi nimi. Obsahuje logickú štruktúru celej databázy v takej podobe, ako ju vidí správca databázy.
- **Interná úroveň** – predstavuje informácie o spôsobe uloženia údajov na fyzickej úrovni. Je implementačne závislá množina údajov, ktorá opisuje údajové štruktúry a prístupové metódy uložených údajov v externej pamäti (na pamäťovom médiu).

Otázky na zopakovanie

1. Čím sa zaoberá architektúra databázových systémov?
2. Vymenujte z akých úrovní sa skladá ANSI/SPARC architektúra.
3. Opíšte jednotlivé úrovne ANSI/SPARC architektúry.
4. Aké dve úrovne zobrazenia rozoznávame v architektúre databáz?
5. Opíšte postup návrhu údajovej štruktúry databázy.

Doplnkové materiály na štúdium

MINÁRIK, R.: Generátor skúšobných testov, Žilinská univerzita v Žiline, Elektrotechnická fakulta, Žilina, 2006. [cit. 2020-05-01]. Dostupné na internete: ⟨http://diplom.utc.sk/wan/704.pdf⟩.

ŠIŠÁK, I.: Nástroj na vizualizaci databázového modelu existující databáze, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2010. [cit. 2020-05-01]. Dostupné na internete: ⟨http://hdl.handle.net/11012/55902⟩.

2 Analýza požiadaviek na databázový systém

Samotná analýza nastáva vo chvíli, kedy vznikne potreba nového spôsobu ukladania a spracovania údajov. V tomto prípade nezáleží na tom, či prechádzame z klasického systému a teda súborového spracovania údajov (napríklad kartotéky) na spôsob databázového spracovania alebo je dôvodom zastaraný pôvodný databázový systém – napríklad generuje chyby alebo nezvláda zaťaženie. Všeobecne ide o fázu, kedy dochádza ku skúmaniu existujúceho systému a požiadaviek používateľa.

V tomto kroku je potrebné vytvoriť zoznam otázok, na ktoré budeme potrebovať jednoznačné odpovede a musíme takisto vykonať nasledujúce kroky…

2.1 Analýza organizácie

Je potrebné vziať do úvahy širší obraz organizácie začínajúc hardvérom, existujúcej štruktúry databázy a takisto obraz situácie celej organizácie. Je rozdiel medzi veľkou nadnárodnou spoločnosťou a malou tuzemskou firmou. Nemôžeme implementovať riešenie, ktoré by bolo vytvorené pre malú firmu do návrhu databázy veľkej nadnárodnej spoločnosti. Účelom analyzovania je zistiť čo najviac informácií o organizácii, aj keď nie úplne súvisia s návrhom databázy. Hlavným cieľom analýzy organizácie je pochopiť čo organizácia vykonáva, ako funguje a pracuje. Pochopenie organizácie, pre ktorú je databáza vytváraná je teda zásadnou podmienkou na dobrý návrh databázy [7].

Otázky súvisiace s organizáciou:

- Aký je účel dotknutej organizácie? Ako by bol tento účel vysvetlený zákazníkovi?
- Čo je podľa vedenia organizácie účelom a hlavným zámerom organizácie?
- Aká je hlavná funkcia organizácie?
- Ako je možné opísať, čo táto organizácia robí?
- Aká je organizačná štruktúra v organizácii?
- Koľko má organizácia zamestnancov? A čo majú v popise práce? [7]

2.2 Definovanie problémov, možností a obmedzení

Tento krok nasleduje po úspešnom pochopení organizačnej štruktúry a je potreba sa používateľov existujúceho systému pýtať na problémy a obmedzenia vyskytujúce sa v súčasnom systéme a na následné potreby a ciele nového DBS. Nemôžeme zabudnúť zistiť, aké obmedzenia budú fungovať aj naďalej. Obmedzenia myslíme napríklad aj požiadavky na hardvér (staré riešenie databázového serveru), ľudí, prípadne limity rôznych hodnôt [7].

Je potrebné pýtať sa nielen jedného používateľa, ale širšieho spektra používateľov, ktorí budú mať k údajom a neskôr k informáciám v databáze prístup. Postup týchto rozhovorov je vhodný od vedúcich funkcií k bežným zamestnancom. Zamestnanci na vedúcich pozíciách sú schopní definovať základné problémy a požiadavky. Naopak, bežní zamestnanci, ktorí k databáze pristupujú každodenne sú schopní požiadavky a problémy následne konkretizovať z používateľského pohľadu [7].

Je takisto dôležité pýtať sa na budúce požiadavky kladené na databázu, súvisiace napríklad s rozširovaním spoločnosti a podobne. Zjednodušíme tým implementáciu a v budúcnosti ušetríme čas [7].

Otázky súvisiace s jestvujúcou databázou:

- Prečo je vôbec potrebná nová databáza? Aký je dôvod jej potreby?
- Aké údaje súvisia s procesom zberu – o aké údaje ide?
- Akým spôsobom organizácia zhromažďuje a prezentuje údaje?
- Ako v súčasnosti organizácia používa a spravuje svoje údaje?
- Aké informácie v súčasnosti organizácia získava z databázy a aké sú potreby informácií v organizácii? [7]

Otázky súvisiace s použitím databázy – vo vzťahu rozhovorov s používateľmi a manažmentom:

- Ako pracujú s databázou jednotlivé skupiny používateľov?
- Aké aplikácie pristupujú k databáze?
- Aké sú požiadavky jednotlivých skupín používateľov na informácie?
- Aké obmedzenia sa predpokladajú?
- Ako je definovaný popis práce jednotlivých používateľov? Akú prácu denne vykonávajú?
- S akým typom údajov pracujú?
- Aké veci sa sledujú a zaznamenávajú a aké typy správ a hlásení sú požadované?
- Existujú ešte nejaké ďalšie veci, ktoré by bolo dobré do databázy implementovať?
- Aká je predstava do budúcna ohľadne rozšírenia organizácie a tým aj rozšírenia databázy? [7]

2.3 Definovanie cieľov

Z odpovedí na predchádzajúce otázky je možné vytvoriť zoznam požiadaviek, ktoré po doladení definujú jednotlivé ciele. Ciele reprezentujú všeobecné úkony používateľa s údajmi, ktoré môžeme v databáze uskutočňovať. Tvoria ich základné požiadavky organizácie na informácie a zaisťujú východiskový/koncový bod na návrh novej databázy. Každý cieľ jasne definuje jednu všeobecnú úlohu a je ľahko pochopiteľný. Nič však nie je definitívne a v priebehu návrhu sa tieto ciele môžu čiastočne meniť, rozširovať a ďalej spresňovať [7].

2.4 Definovanie rozsahu

Ako návrhári DBS musíme jasne a zreteľne určiť rozsah a časový plán čiastkových úloh a to vrátane rozsahu samotnej implementácie [7].

2.5 Údajová analýza

Údajová analýza definuje hlavne údajové toky, ktoré súvisia s vybraným systémovým procesom. Analýza sa sústreďuje rovnako na vstupné a výstupné údaje a takisto na samotnú transformáciu vstupných údajov na výstupné. Výsledok analýzy poskytuje dostatočný prehľad o potrebných údajoch a väzbách medzi nimi tak, aby mohol riešiteľ projektu vytvoriť požadovaný databázový prípadne informačný systém [7].

Môžeme povedať, že údajová analýza pre koncového používateľa systému vyjadruje spôsob komunikácie s riešiteľmi konkrétneho systému. Údajovú analýzu preto vykonávame v rámci definovania problémov, možností a obmedzení, kedy na základe predložených údajových formulárov, tlačových zostáv a vhodne položených otázok sme schopní identifikovať prebiehajúce údajové toky v celom procese. Výsledkom údajovej analýzy je model na konceptuálnej úrovni, ktorý vyjadruje statický obraz opisu reality. Vyjadrujeme ho napríklad prostredníctvom E-R diagramu. Táto analýza často pomáha odstrániť bariéru medzi konečnými používateľmi na jednej strane a realizátormi produktu na strane druhej. Prax ukazuje, že používatelia obvykle opisujú svoje požiadavky na základe svojich skúseností, ale hlavne v začiatkoch analýzy nedokážu domyslieť využitie všetkých možností respektíve potenciálu, ktoré im poskytuje konečný systém. Naopak, realizátori podrobne poznajú softvérové a hardvérové možnosti. Na druhú stranu im však nie sú obvykle dobre známe potreby a požiadavky od používateľov. Na účely vzájomného pochopenia medzi oboma stranami boli vytvorené špeciálne vyjadrovacie prostriedky napríklad už spomínaný E-R diagram [7].

2.6 Funkčná analýza

Funkčná analýza sa spoločne s údajovou vykonáva na začiatku návrhu databázových systémov a sú spolu prepojené. S pomocou nej vykonávame návrh činnosti DBS a vyjadruje, ako má proces pracovať. Môžeme povedať, že je to zoznam a opis jednotlivých akcií uskutočňovaných nad konceptuálnym návrhom a externými návrhmi DBS. Ide o opis stavov údajových štruktúr a prechodov medzi nimi [7].

Pri funkčnej analýze sa vytvárajú modely na správne pochopenie systému a na komunikáciu medzi zadávateľom projektu, realizátorom a konečným používateľom. Tento model musí byť zrozumiteľný a čo najviac presný pre budúcu implementáciu [7].

Výsledkom funkčnej analýzy je:

- Zoznam funkčných požiadaviek.
- Zoznam udalostí a reakcií – model vonkajšieho správania sa systému.
- Požadované vstupy a výstupy [7].

Funkčná analýza definuje najmä obsah a rozsah výstupných zostáv ako elektronické súbory, zostavy na tlačenie, výstupy na monitor a potrebných vstupných údajov ako interné a externé údaje, číselníky. Súčasťou analýzy je aj opis transformácie vstupných údajov na výstupné, vrátane logických a formálnych kontrol týchto údajov. Funkčná analýza sa teda zaoberá definíciou základných funkcií DBS a môžeme ju vyjadriť viacerými rôznymi modelmi, ktoré opisujú funkcionalitu systému, ako napríklad [7]:

- **Diagram toku údajov** (data flow diagram) – zobrazuje procesy a toky údajov medzi nimi (ako sú údaje použité a transformované pri postupne systémom). Opisuje vnútornú funkcionalitu systému na základe analýzy toku údajov medzi internými funkciami systému navzájom a s okolím systému.

Pri zložitejších systémoch s viacerými vstupmi a výstupmi sa tento diagram kreslí v niekoľkých úrovniach. Ako prvá úroveň sa vytvára **kontextový diagram,** ktorý celý systém zobrazuje ako jediný proces a s ním všetky súvisiace externé entity. Kontextový diagram sa potom rozpracúva v jednotlivých úrovniach. Pri bežných systémov sa používa väčšinou diagram v troch maximálne štyroch úrovniach [7].

- **Kontextový diagram** – opisuje vonkajšie správanie sa systému respektíve vzťah systému k jeho okoliu. Diagram odpovedá tomu, čo potrebuje používateľ od systému. Získame ho analýzou udalostí, ktoré sa môžu vyskytnúť medzi systémom a subjektmi v jeho okolí. Medzi tieto subjekty patria napríklad iní používatelia nadväzujúcich prípadne spolupracujúcich systémov alebo nadradený informačný systém. Celý systém (celá databáza) je pri tomto type diagramu zobrazená ako celok. Subjekt mimo systému sa zvykne označovať pojmom **terminátor.** Každá udalosť medzi systémom a jeho okolím vyvolá konkrétny väčší alebo menší prenos informácií (niekedy aj prenos materiálov) [7].
- **Stavový diagram** (state transition diagram – STD) – definuje možné stavy, možné prechody medzi stavmi, udalosti, ktoré prechody iniciujú, podmienky prechodov a akcie, ktoré s prechodmi súvisia. Stavový diagram je možné použiť aj na opis dynamiky objektu (ak má rozpoznateľné stavy), na opis metódy (ak poznáme algoritmus) alebo na opis protokolu (vrátane protokolu o spojenie používateľa so systémom). Je veľmi potrebný z hľadiska pochopenia logiky aplikácie. Používateľ neoznámi algoritmus celého procesu, ale pozná stavy – napríklad stav objednávky, stav faktúry, stav palety a podobne. Dá sa z neho odvodiť funkcionalita systému a následne algoritmus spracovania údajov [7].

2.7 Katalóg požiadaviek na databázové systémy

Katalóg požiadaviek je základným analytickým materiálom na vytvorenie databázového systému. V tomto dokumente sa zvyčajne nepoužíva žiadna technická terminológia, ktorej by zadávateľ projektu (budúci používateľ systému) nemusel rozumieť. Autormi sú obvykle riešitelia projektu, ktorí jeho vypracovaním dávajú najavo, že správne pochopili zadanie. Dokument tvorí formálnu dohodu medzi zadávateľom a riešiteľom, takže musí obsahovať všetky špecifikácie a definovanie požiadaviek, ktoré zadávateľ má na konkrétny systém. Jeho koncová podoba preto vzniká až po viacnásobnom stretnutí alebo komunikáciou so zadávateľom. Môžeme teda povedať, že katalóg požiadaviek je súčasťou záväzného zadania na vypracovanie databázového systému.

Katalóg požiadaviek zvyčajne nerieši otázky implementácie systému a takisto špecifikácií technologických prostriedkov, na ktorých bude systém fungovať. Môže sa to stať v prípade, keby na tento aspekt kládol zadávateľ projektu nejaké zvláštne nároky. V takom prípade musia byť tieto požiadavky uvedené v dokumentácii. Katalóg požiadaviek môže obsahovať aj grafickú dokumentáciu na zvýšenie názornosti.

Zhrnutie 2. kapitoly

Analýza nastáva v počiatočnej fáze vývoja databázového systému. Tvorca systému musí podrobne spoznať činnosť organizácie, pre ktorú sa systém vytvára. Správne pochopenie činnosti organizácie, pre ktorú je databáza vytváraná je teda zásadnou podmienkou na dobrý návrh DBS. Dôležité je takisto zistiť problémy s používaním aktuálneho systému, rôzne obmedzenia (napr. hardvérové požiadavky), limity rôznych hodnôt a tým zamerať návrh na následné potreby, požiadavky a ciele nového systému.

V údajovej analýze sa sústredíme na vstupné a výstupné údaje, a rovnako na samotnú transformáciu vstupných údajov a údaje výstupné. Údajová analýza poskytuje prehľad o údajoch a takisto o jednotlivých väzbách (vzťahoch) medzi nimi.

Okrem údajovej analýzy vykonávame aj funkčnú. Funkčná analýza sa zaoberá definíciou základných funkcií DBS a vyjadrujeme ju prostredníctvom rôznych modelov. Definuje najmä obsah a rozsah výstupných zostáv ako elektronické súbory, zostavy na tlačenie, výstupy na monitor a potrebných vstupných údajov ako interné a externé údaje, číselníky.

Výsledkom analýzy organizácie je katalóg požiadaviek, ktorý je základným analytickým materiálom na vytvorenie databázového systému. Výsledkom katalógu je súhrn požiadaviek, ktoré zadávateľ má na konkrétny systém. Tvorca DBS ním dáva najavo, že správne pochopil fungovanie organizácie, pre ktorú DBS buduje.

Otázky na zopakovanie

1. Vlastnými slovami opíšte, čo sa vykonáva vo fáze analýzy organizácie pri tvorbe DBS.
2. Čo sa vykonáva v údajovej analýze?
3. Čo sa vykonáva vo funkčnej analýze?
4. Čo je obsahom katalógu požiadaviek pri tvorbe DBS?
5. Vymenujte modely, ktoré opisujú funkcionalitu systému.

3 Návrh

Je realizovaný rôznymi modelmi systému na základe už skôr definovaných požiadaviek. Sú tu vytvárané modely nie len na úrovni konceptuálnej a logickej schémy, ale tiež fyzický návrh ako príprava na implementáciu.

Cieľom návrhu je, že databáza musí uchovávať údaje potrebné na plnenie informačných požiadaviek známych v čase návrhu databázy, ale aj budúcich požiadaviek. Údaje musia poskytovať platné a presné informácie, ktoré majú význam na účel vyhotovenia. Takisto nemôžeme zabúdať, že databáza musí byť v budúcnosti do určitej miery rozšíriteľná [7].

Samotný návrh uskutočňujeme vtedy, keď sú známe koncové požiadavky, identifikované v analýze. Tie budú použité ako základ na vývoj nového systému. Všeobecne nastáva prevod rámcového pochopenia údajových štruktúr na technické chápanie [7].

Otázky typu:

- Aký/aká/aké? nahradíme otázkou typu → Ako?
- Aké údaje sú požadované? → Ako budú údaje organizované?
- Aké problémy sa budú riešiť? → Ako bude zaistený prístup k údajom? [7]

Tento krok zahŕňa údajovú a funkčnú analýzu, kedy sa v údajovej analýze vytvárajú modely na troch úrovniach štruktúry údajov, to znamená na troch úrovniach návrhu modelu DBS. Tým myslíme konceptuálnu, technologickú a implementačnú úroveň. Vytvorí sa konceptuálny model (E-R diagram), potom sa tento model prevedie na technologický model (relačná schéma) a v konečnej fáze sa zvolí SRBÚ v závislosti od požiadaviek a komplexnosti štruktúry údajov [7].

3.1 Entitno-relačný diagram (entity relationship diagram, E-R diagram)

Entitno-relačný diagram je grafický nástroj, ktorý sa používa pri návrhu databázových systémov na vyjadrenie modelu údajov, pričom charakterizuje pohľad na statickú časť systému. V rámci E-R diagramu sú definované jednotlivé údajové objekty sveta (entity) a vzájomné vzťahy medzi nimi (relácie). Entitno-relačné diagramy pozostávajú z nasledujúcich prvkov:

- **Entita** je špecifickým predmetom záujmu z oblasti reálneho sveta. Je to akýkoľvek údajový objekt, ktorý je predmetom záujmu, o ktorom chceme uchovávať údaje.
- **Atribút** je elementárny prvok, ktorý vyjadruje bližšiu charakteristiku, vlastnosť alebo informáciu o konkrétnej entite. Identifikátorom alebo kľúčovým atribútom budeme nazývať atribút alebo množinu atribútov, ktorých hodnoty umožňujú jednoznačne rozlíšiť jednotlivé inštancie (záznamy) entity navzájom medzi sebou. Takýto atribút budeme nazývať primárnym kľúčom.
- **Doména atribútu** je obor hodnôt, ktorý môže atribút nadobúdať.
- **Relácia** (vzťah) je určitá forma väzby, spojenia medzi jednotlivými entitami, ktorú evidujeme. Medzi entitami môže existovať viac rôznych typov vzťahov.

Postup pri tvorbe E-R diagramu

1. Definícia entít.
2. Pridelenie atribútov jednotlivým entitám.
3. Definícia relácií (vzťahov) medzi entitami.
4. Definícia kardinality pre relácie.
5. Reprezentácia navrhnutej údajovej štruktúry prostredníctvom grafických notácií.

3.2 Entita

Každá entita má jasný názov, ktorý ju čo najviac vystihuje. Tento názov zvykneme pomenúvať v jednotnom čísle podstatného mena. Samotné entity majú svoje atribúty, ktoré opisujú vlastnosti konkrétnej entity. Entita taktiež obsahuje atribút tzv. identifikátor, ktorý jednoznačne identifikuje jednotlivé inštancie entity. Tento atribút nazývame **primárny kľúč.** Inštancie sú konkrétne záznamy v každej entite.

Notácia entity

Entity budeme v E-R diagrame znázorňovať prostredníctvom obdĺžnikov, do ktorých vpíšeme názov entity.

Obrázok č. 3.1 – Notácia entít.

Príklad

Entita „Študent“ bude mať inštancie jednotlivých študentov (Peter Novák, Katarína Pekná). Každý študent je teda jedna inštancia (jeden záznam) entity „Študent,“ a odlišuje sa od inej inštancie v tej istej entite s pomocou hodnôt jednotlivých atribútov, minimálne však v hodnote, ktorú obsahuje **primárny kľúč.**

Predstavme si, že univerzita má niekoľko tisíc študentov. Je teda pravdepodobné, že aspoň dvaja študenti budú mať rovnaké meno a priezvisko, prípadne budú mať rovnaké hodnoty aj v ostatných atribútoch. Z tohto dôvodu sa musia odlišovať aspoň v hodnote primárneho kľúča. Primárny kľúč v tomto prípade môže byť napríklad: *identifikačné číslo (ID) študenta, rodné číslo* alebo *číslo občianskeho preukazu.* Identifikačné číslo, rodné číslo alebo číslo občianskeho preukazu sú čísla pridelené (vygenerované) systémom a nemalo by sa stať, aby mali dvaja ľudia mali toto číslo rovnaké.

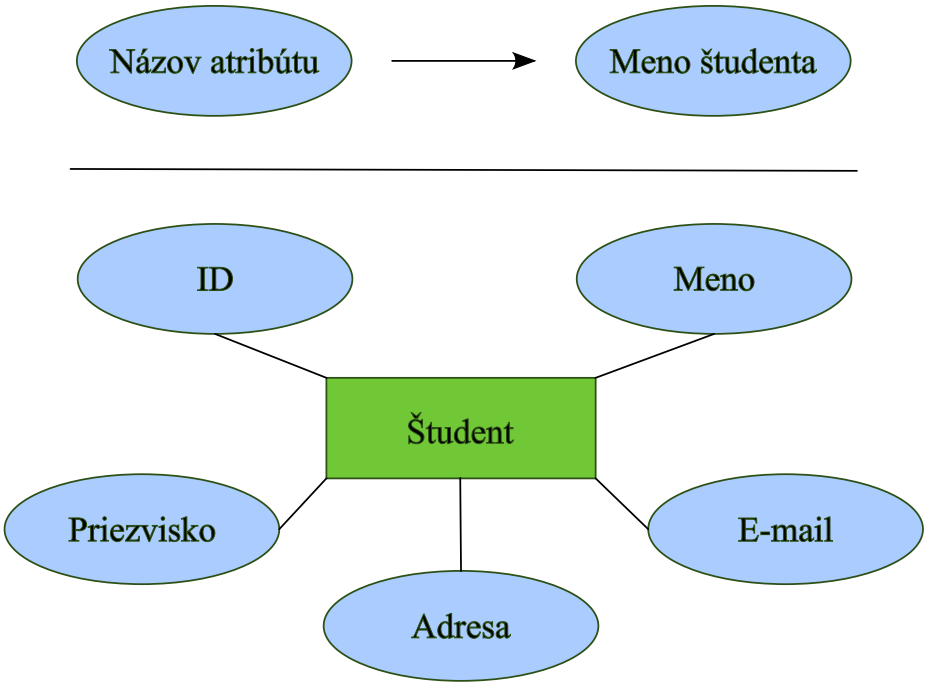
Obrázok č. 3.2 – Jedna inštancia entity „Zamestnanec."

3.3 Atribút entity

Atribúty sú vlastnosti entity respektíve môžeme aj povedať vlastnosti jednotlivých inštancií entity, ktoré ich dostatočným spôsobom opisujú a od ostatných inštancií odlišujú. Sú to údaje, ktoré v sebe uchovávajú informácie o jednotlivých záznamoch entity. Uvedieme na príklade entity „Študent," kde môžeme ako atribúty uviesť: „Identifikačné číslo (ID) študenta," „Meno," „Priezvisko," „Bydlisko," „Študijný odbor," „Dátum narodenia," „Email" a ďalšie.

Každému atribútu určujeme údajový typ podľa toho akú informáciu do neho budeme vkladať. Medzi najpoužívanejšie údajové typy pre databázové systémy patria:

- **Číselné (numerické) typy** – uchovávajú v sebe číselné údaje. Môžeme použiť celočíselný typ, čísla s desatinnou čiarkou (reálne čísla), takisto čísla so zápornou hodnotou.
- **Automatické číslovanie** – ide o jedinečnú hodnotu číselného údaju, ktorá je vygenerovaná systémom pre každý nový záznam, pričom nie je možné aby dva záznamy mali tú istú hodnotu.
- **Textové (znakové) reťazce** – vkladáme alfanumerické údaje. Sem patria okrem písmen a čísiel aj znaky ako čiarka, otáznik, výkričník a ďalšie. Môžeme určiť presný počet znakov, prípadne zadefinovať maximálny alebo minimálny počet použitých znakov.
- **Dátum a čas (date)** – uchováva údaje o čase a dátume.
- **Peňažné (menové) typy** – uchovávajú informácie o peňažných údajoch s presnosťou na niekoľko desatinných miest (napríklad v prípade meny Euro až na niekoľko miest centov).
- **Logické hodnoty (boolean)** – uchovávajú v sebe hodnoty 0 alebo 1, respektíve TRUE alebo FALSE (pravda alebo nepravda).
- **Audio, video- či obrázkové údajové typy** – niektoré databázové systémy umožňujú uchovávať aj informácie vo zvukových, či videoformátoch a takisto aj obrázkové formáty.



Obrázok č. 3.3 – Notácia atribútov v E-R diagrame.

Notácia atribútov

Atribúty budeme v E-R diagrame znázorňovať prostredníctvom elíps, do ktorých budeme zapisovať ich názov – pozri obrázok č. 3.3.

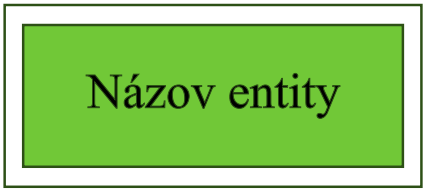
3.4 Primárny kľúč (PK)

Ako sme už spomínali, je veľmi dôležité navzájom rozlíšiť jednotlivé inštancie v entitách medzi sebou. Na to slúži atribút, ktorého hodnota je jedinečná a odlišná pre každú inštanciu v konkrétnej entite. Tento atribút budeme nazývať **primárny kľúč** (niekde sa zvykne označovať aj ako „superkľúč"). Je to jedinečný atribút, ktorý jednoznačne identifikuje každú inštanciu (záznam) v entite. Nemôže sa teda vyskytnúť prípad, kedy by dve inštancie (záznamy) v tabuľke mali rovnakú hodnotu primárneho kľúča. Atribút, ktorý by mohol byť primárny kľúč nazývame **kandidát na kľúč.** PK môže byť tvorený jedným alebo viacerými atribútmi, hovoríme potom o jednoduchom respektíve zloženom primárnom kľúči. Pri návrhoch databázových systémov sa zvykne primárny kľúč podčiarknuť. Je to hlavne z dôvodu zvýraznenia tohto jedinečného atribútu medzi ostatnými atribútmi v množine atribútov.

V niektorých entitách môže existovať viac kandidátov (adeptov) na primárny kľúč. Napríklad entita „Zamestnanec" môže mať primárny kľúč: Identifikačné číslo (vygenerované systémom), číslo občianskeho preukazu, rodné číslo, prípadne iné. V databázových systémoch kde uchovávame údaje o motorových vozidlách býva často primárnym kľúčom ŠPZ respektíve EČV (evidenčné číslo vozidla), pričom vieme, že každé motorové vozidlo má svoje evidenčné číslo jedinečné.

Entita, z ktorej atribútov nie je možné vybrať primárny kľúč označujeme pojmom **slabá entita.** V takom prípade môžeme napríklad v entite vytvoriť nový atribút, už spomínané **identifikačné číslo,** ktoré bude automaticky generované systémom a nemôže sa teda stať, že by dve inštancie z jednej entity mali rovnaké identifikačné číslo. Entitu, ktorá má určený primárny kľúč označujeme ako **silná entita.**

Slabú entitu znázorňujeme v E-R diagramu tak, že ju vložíme do ďalšieho väčšieho obdĺžnika.



Obrázok č. 3.4 – Notácia slabej entity v E-R diagrame.

3.5 Cudzí kľúč (CK)

V súvislosti s primárnym kľúčom existuje aj atribút s názvom **cudzí kľúč,** ktorý sa zvykne označovať aj ako „foreign key“ (FK). Cudzí kľúč je vlastne primárny kľúč z inej entity v rámci jednej relácie, s pomocou ktorého budeme odkazovať na práve inštancie z druhej tabuľky. Bližšie tento princíp vysvetlíme v kapitole Údajové modely, pretože na jeho dôkladné vysvetlenie je potrebné získať znalosti z oblasti relácií.

3.6 Doména atribútu

Hovoríme o množine dovolených hodnôt atribútu. Jej definovanie umožňuje kontrolu správnosti typov a hodnôt údajov pri zápise do systému.

Príklady domén:

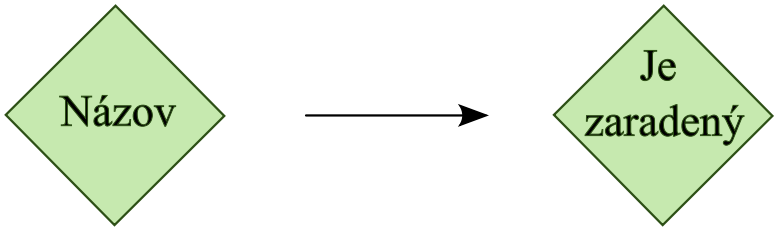
- Deň v mesiaci je celé číslo z intervalu: 1 – 31.
- Známka zo skúšky, kde sú dovolené hodnoty: A, B, C, D, E, FX.

3.7 Relácia

Relácia ilustruje spojenie (vzťah) medzi dvomi entitami. Názvy relácií uvádzame vo forme slovies. Pri relácii definujeme kardinalitu, ktorá sa zvykne označovať aj ako mohutnosť medzi entitami v danej relácii. Mohutnosť definuje maximálny počet, respektíve koľkokrát môže byť jedna inštancia z entity prepojená s inštanciami druhej súvisiacej entity v rámci relácie.

Notácia relácií

Relácie budeme v E-R modeloch znázorňovať prostredníctvom kosoštvorcov, do ktorých budeme zapisovať názov relácie.



Obrázok č. 3.5 – Notácia relácií v E-R diagrame.

3.8 Kardinalita relácií

Ako sme už spomínali, kardinalita vlastne vyjadruje maximálny počet inštancií jednej entity, ktoré môžu asociovať aspoň s jednou inštanciou inej entity v rámci jednej relácie. Všeobecne existujú tri typy kardinality a síce: 1 : 1, 1 : N a M : N. Určenie kardinality relácie zvyčajne záleží od reálneho sveta, ktorý práve modelujeme.

Kardinalita 1 : 1

Kardinalita relácie jedna k jednej (1 : 1) znamená vzťah keď je jedna inštancia z entity **A** vo vzťahu iba s jednou inštanciou z entity **B.** Kardinalita 1 : 1 sa vyskytuje len vo výnimočných prípadoch. Je to najmä preto, že tento typ reláciu nemusíme ani použiť. Vo väčšine prípadov stačí iba pridať do jednej z entít o jeden atribút naviac. Ak sa aj táto relácia niekde vyskytne, tak to je väčšinou z dôvodu sprehľadnenia rozsiahlych tabuliek.

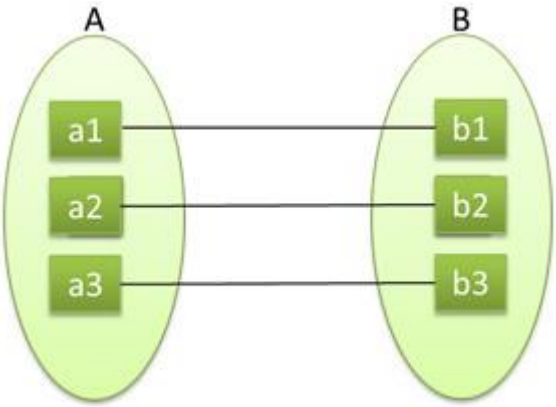
Notácia relácie 1 : 1

Reláciu 1 : 1 budeme v E-R diagramoch znázorňovať prostredníctvom prepájacích čiar (spojníc), ktorých konce pri entitách budú zakončené šípkami.

Obrázok č. 3.6 – Notácia relácie 1 : 1.

Príklady relácie 1 : 1

- Predstavme si kino s jednou kinosálou. V tejto jednej kinosále (entita **A**) sa podľa programu môže premietať iba jeden film (entita **B**). Nie je možné aby kino premietalo v jednom čase viacero filmov na jednom premietacom plátne.
- Dobrým príkladom môže byť tradičné európske manželstvo tvorené medzi mužom (entita **A**) a ženou (entita **B**), kedy jeden muž je v manželstve iba s jednou unikátnou ženou.
- Ako sme už spomínali, relácia 1 : 1 sa vyskytuje veľmi zriedkavo, pretože vo veľa prípadov stačí ak do jednej z entít takéhoto vzťahu pridáme o jeden atribút viac. Napríklad entita „Človek“ a entita „Občiansky preukaz“ (prípadne iný doklad ako vodičský preukaz, rodný list, identifikačné číslo zákazníckej karty…). Jednej inštancii z entity „Človek“ prináleží iba jedna inštancia z entity „Občiansky preukaz“ (ak predpokladáme, že databáza nie je archívom policajného oddelenia). Túto reláciu však môžeme veľmi jednoducho z modelu odstrániť a to tým, že jednoducho do entity „Človek“ priradíme atribút s názvom „Číslo OP.“ Do tohto atribútu potom priradíme každej inštancii z entity „Človek“ jeho číslo občianskeho preukazu (prípadne iného dokladu).



Obrázok č. 3.7 – Kardinalita 1 : 1 [4].

Kardinalita 1 : N

Vzťah jedna k viacerým (1 : N) reprezentuje prípad, kedy jedna inštancia z entity **A** môže byť vo vzťahu s viacerými inštanciami entity **B.** To znamená, že pre jednu inštanciu z entity **A** je priradených nula, jedna alebo viac inštancií z entity **B,** ale naopak pre inštanciu z entity **B** je priradená iba jedna inštancia z entity **A.** Vzťah jedna k mnohým patrí medzi najčastejšie využívané kardinality pri modelovaní databázových systémov.

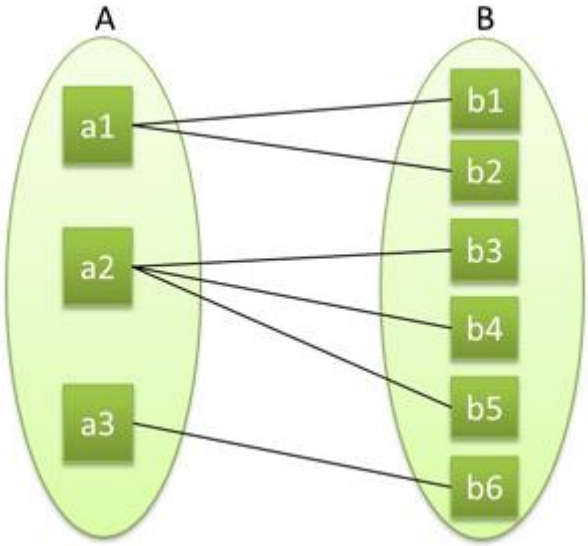
Notácia relácie 1 : N

Reláciu 1 : N budeme v E-R diagramoch znázorňovať prostredníctvom prepájacích čiar, pričom koniec čiary pri entite, kde je vo vzťahu 1 inštancia bude zakončený šípkami a koniec pri entite, kde je vo vzťahu N inštancií bude zakončený bez použitia šípky.

Obrázok č. 3.8 – Notácia relácie 1 : N.

Príklady relácie 1 : N

- Uvažujeme na príklade školského informačného systému, kedy je jeden študent (entita **A**) priradený do jednej študijnej skupiny (entita **B**), ale naopak do jednej študijnej skupiny môže byť priradených viac (N) študentov.
- Jeden vysokoškolský učiteľ (entita **A**) je pridelený na jednu katedru (entita **B**), ale jedna katedra môže mať pridelených viac (N) učiteľov.
- Cestujúci a dopravný prostriedok (autobus, vlak…). Uvažujeme o situácii kedy v jednom časovom momente môže byť cestujúci (entita **A**) iba v jednom autobuse (entita **B**). V tomto jednom autobuse sa ale v tejto jednej chvíli môže nachádzať viac (N) cestujúcich z entity **A.**



Obrázok č. 3.9 – Kardinalita 1 : N [4].

Kardinalita M : N

Tento vzťah umožňuje každej inštancii z entity **A** priradiť ľubovoľný počet inštancií z entity **B,** pričom takisto platí, že každej inštancii z entity **B** môžeme priradiť ľubovoľný počet inštancií z entity **A.** Ľubovoľný v tomto prípade znamená: nula, jedna, dva… až N. V praxi sa často stretávame s tým, že táto relácia býva často realizovaná prostredníctvom kombinácie dvoch relácií 1 : N, pričom vzniká nová pomocná tabuľka (nazývame aj väzobná tabuľka), ktorá je zložená z primárnych kľúčov z obidvoch relácií. Táto pomocná tabuľka môže obsahovať aj svoje ďalšie atribúty.

Notácia relácie M : N

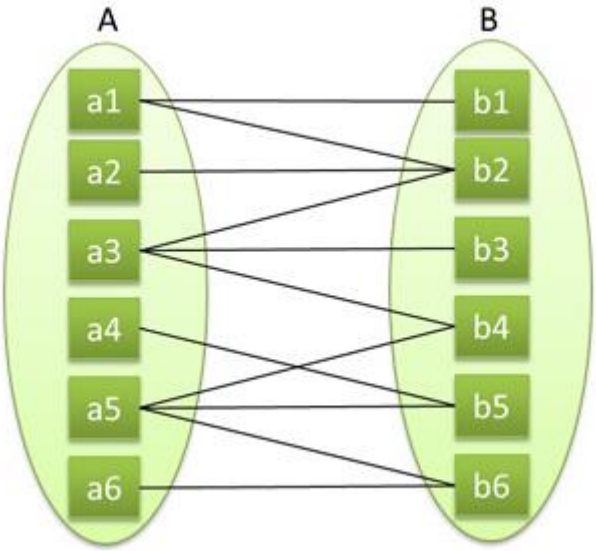
Reláciu M : N budeme v E-R diagrame znázorňovať prostredníctvom prepájacích čiar, ktorých konce pri entitách budú zakončené bez použitia šípok.

Obrázok č. 3.10 – Notácia relácie M : N.

Príklady relácie M : N

- Uvažujme o vzťahu medzi entitami „Film“ a „Herec.“ V jednom filme (entita **A**) môže hrať viacero (M) hercov (entita **B**) a naopak – jeden herec môže hrať vo viacerých (N) filmov.
- Na príklade školského informačného systému, kedy jeden študent (entita **A**) má zapísaných viac (M) predmetov (entita **B**) a naopak – na jeden predmet je zapísaných viacero (N) študentov.

- Príklad z databázového systému internetového obchodu, kedy jeden zákazník (entita **A**) si môže kúpiť viac (M) druhov tovaru (entita **B**) a naopak – jeden druh tovaru si môže zakúpiť viacero (N) zákazníkov.

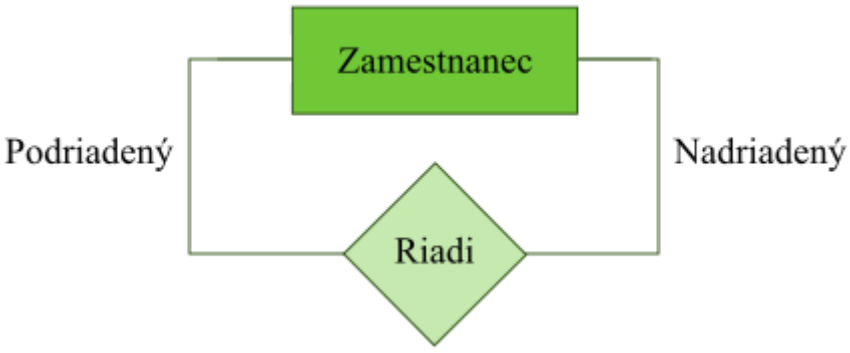


Obrázok č. 3.11 – Kardinalita M : N [4].

3.9 Stupeň relácií

Relácia je okrem kardinality klasifikovaná aj podľa stupňa. V závislosti od toho, koľko entít sa nachádza vo vzájomnom vzťahu a teda vo vzájomnej jednej relácii poznáme nasledujúce:

- **Unárna (rekurzívna) relácia:** je to typ vzťahu kedy je entita spojená sama so sebou. Ako jednoduchý príklad môžeme uviesť vzťah zamestnanec a jeho nadriadený respektíve podriadený. Vtedy uvažujeme, že nadriadený/podriadený je vlastne tiež zamestnancom a tento záznam patrí takisto do entity zamestnanec.

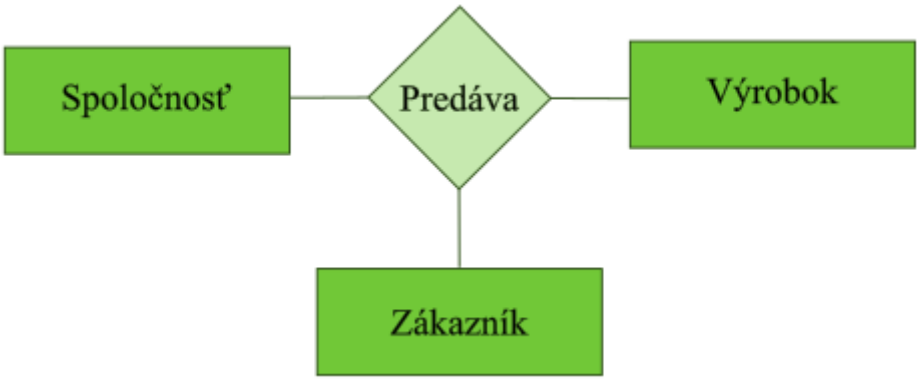


Obrázok č. 3.12 – Unárna relácia.

- **Binárna relácia:** vo vzťahu sa vyskytujú dve entity. Binárny stupeň relácie je najčastejšie sa vyskytujúci. Ako príklad uvádzame reláciu zo školského informačného systému medzi entitami „Študent“ a „Predmet.“

Obrázok č. 3.13 – Binárna relácia.

- **Ternárna relácia:** vo vzťahu sa vyskytujú tri entity. Ternárny stupeň relácie sa v praxi vyskytuje len veľmi zriedka, dôvodom je náročné modelovanie týchto vzťahov. Uvádzame na príklade medzi entitami „Spoločnosť," „Tovar," „Zákazník," pričom uvažujeme, že jedna spoločnosť predáva N výrobkov M zákazníkom. Rovnako jeden výrobok predáva P spoločností M zákazníkom. Pre zákazníka platí, že jeden zákazník si môže kúpiť N výrobkov od P spoločností. V tomto prípade hovoríme o kardinalite M : N : P.



Obrázok č. 3.14 – Ternárna relácia.

- **N-árna relácia:** hovoríme o vzťahu medzi N-entitami súčasne. N-árny stupeň relácie sa v praxi nepoužíva, spomíname ho len z teoretického hľadiska. Z tohto dôvodu nepovažujeme za potrebné uvádzať konkrétny príklad typu relácie.

3.10Členstvo vo vzťahu

Entity, ktoré sú navzájom prepojené prostredníctvom relácie hovoríme, že sú členmi vzťahu. V súvislosti s tým potom rozlišujeme na **povinné** a **nepovinné** členstvo v tomto vzťahu.

Príklad: Uvažujeme o členstve vo vzťahu medzi entitami „Študent" a „Študijná skupina."

Obrázok č. 3.15 – Členstvo vo vzťahu.

Študijná skupina za určitých okolností môže existovať aj bez študentov. V prípade ak sme pred začiatkom školského roka vytvorili novú študijnú skupinu, ale ešte sme do nej nepriradili žiadnych študentov. Naopak, každý záznam v tabuľke „Študent“ musí byť zaradený do určitej študijnej skupiny.

- Členstvo entity „Študent“ v tejto relácii je **povinné,** pretože všetky inštancie musia byť zaradené do študijnej skupiny.
- Entita „Študijná skupina“ má naopak členstvo vo vzťahu **nepovinné.**

3.11Zovšeobecnenie (generalizácia) entít

Zovšeobecniť môžeme entity, ktoré vo všeobecnosti spadajú do entity vyššej úrovne. Používame to vtedy, ak chceme zvýrazniť podobnosť, spoločné črty, charakteristiky medzi entitami a naopak, keď chceme ignorovať rozdiely medzi nimi. Zovšeobecnenie využívame pri navrhovaní relačných schém.

Základné body zovšeobecnenia

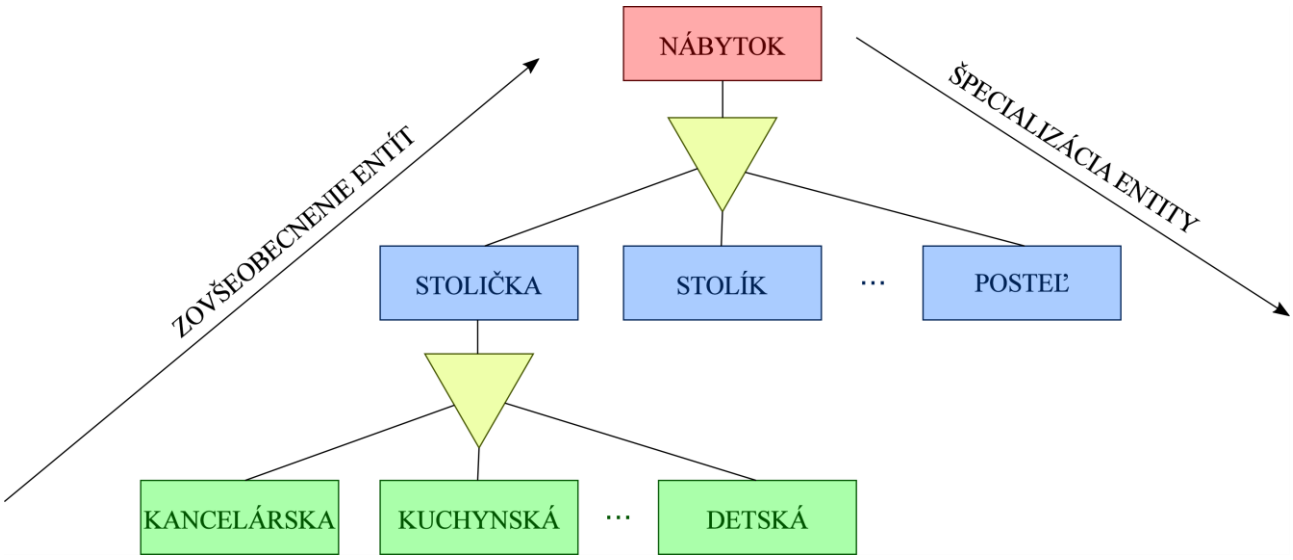
- Postupujeme smerom z nižšej úrovne entít do vyššej (zdola nahor).
- Vyberáme (extrahujeme) spoločné vlastnosti viacerých entít na vytvorenie novej entity.
- Vedie k vytvoreniu jednej entity kombináciou množiny viacerých entít z nižšej úrovne.
- Entita z vyššej úrovne (rodičovskej – nadradenej) súčasne patrí do jednej z entít nižšej (detskej – podradenej) úrovne.
- Zovšeobecnenie sa uplatňuje vždy na skupine viacerých entít.
- Zovšeobecnením znižujeme veľkosť konceptuálnej schémy.

3.12Špecializácia entity

Špecializáciou rozčleňujeme entitu na viaceré nové entity na nižšej úrovni, ktoré budú dediť určitú časť vlastností z entity, ktorú špecializujeme. Môže sa stať prípad, kedy entitu z vyššej úrovne už nevieme ďalej rozdeliť. Preto táto entita nemusí mať žiadne entity na nižšej úrovni.

Základné body špecializácie

- Postupujeme smerom z vyššej úrovne entít do nižšej (zhora nadol).
- Rozdeľujeme entitu na vytvorenie viacerých nových entít, ktoré zdedia určité charakteristiky, črty z entity, ktorú štiepime.
- Vedie k vytvoreniu viacerých nových entít z jednej entity.
- Entita z vyššej úrovne nemusí patriť do entity, ktorá je na nižšej úrovni.
- Špecializácia sa uplatňuje vždy iba na jednu entitu.
- Špecializáciou zväčšujeme konceptuálnu schému.



Obrázok č. 3.16 – Zovšeobecnenie vs. špecializácia entít.

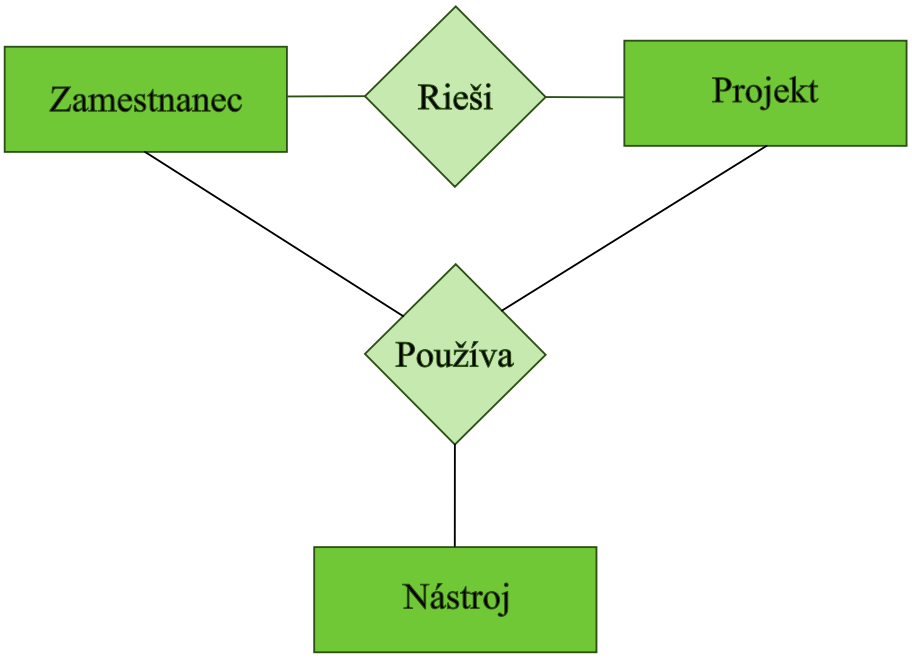
Zovšeobecnenie (generalizácia) a špecializácia patria medzi návrhové postupy a obe sú rovnako dôležité na návrh konceptuálnej schémy. Grafickou notáciou je trojuholník, ako je znázornené na obrázku číslo 3.16.

3.13Agregácia (súhrn)

V E-R modeli nie je možné znázorniť relácie (vzťahy) medzi reláciami. To považujeme za určité obmedzenie. Tento nedostatok dokážeme vyriešiť prostredníctvom agregácie. Agregácia umožňuje zoskupiť vzťah medzi dvomi entitami do jedného celku. Tento vzťah je aj s príslušnými entitami spojený do entity vyššej úrovne.

Príklad agregácie

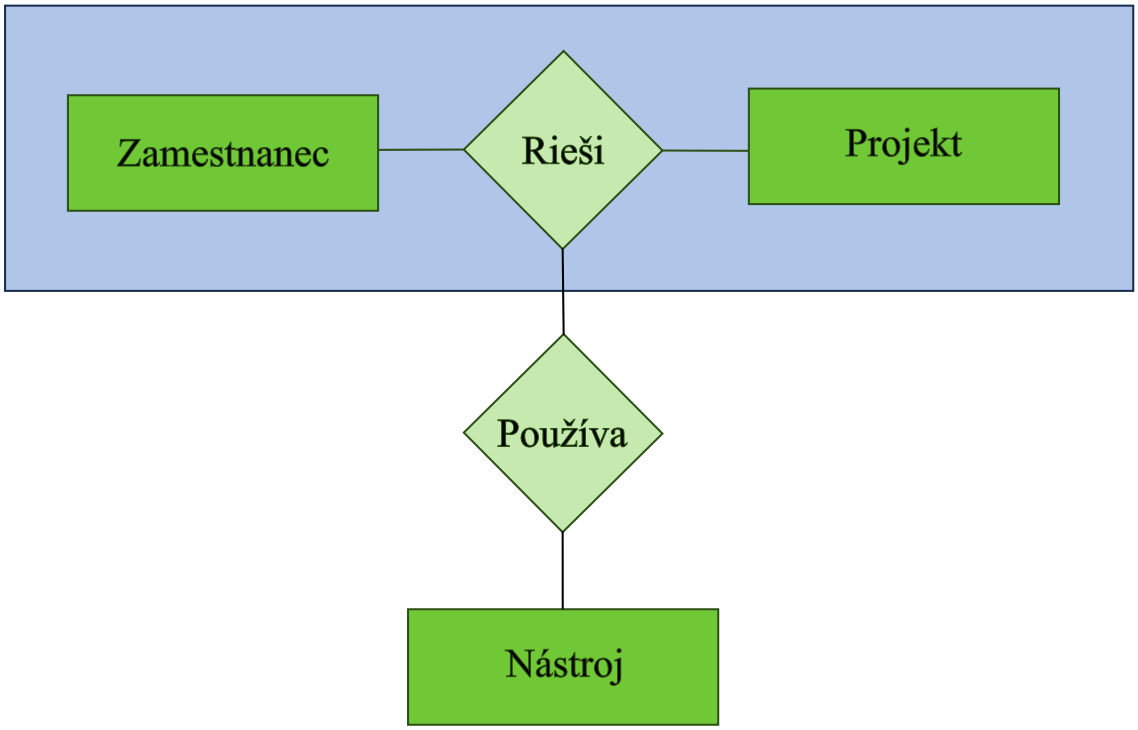
Uvažujme o databázovom systéme, v ktorom máme entity: „Zamestnanec," „Projekt" a „Nástroj." Pričom platí, že M zamestnancov rieši N projektov, pri ktorých používajú P rôznych nástrojov.



Obrázok č. 3.17 – Príklad agregácie [4].

V tomto príklade potrebujeme vyjadriť vzťah medzi reláciami „Rieši“ a „Používa.“ Nemôžeme tu použiť ternárnu reláciu M : N : P, pretože na jednom projekte pracuje viacero (M) zamestnancov, pričom každý z nich používa viacero (P) rôznych nástrojov a tie isté nástroje môžu používať aj iní zamestnanci na riešení viacerých (N) projektov [4].

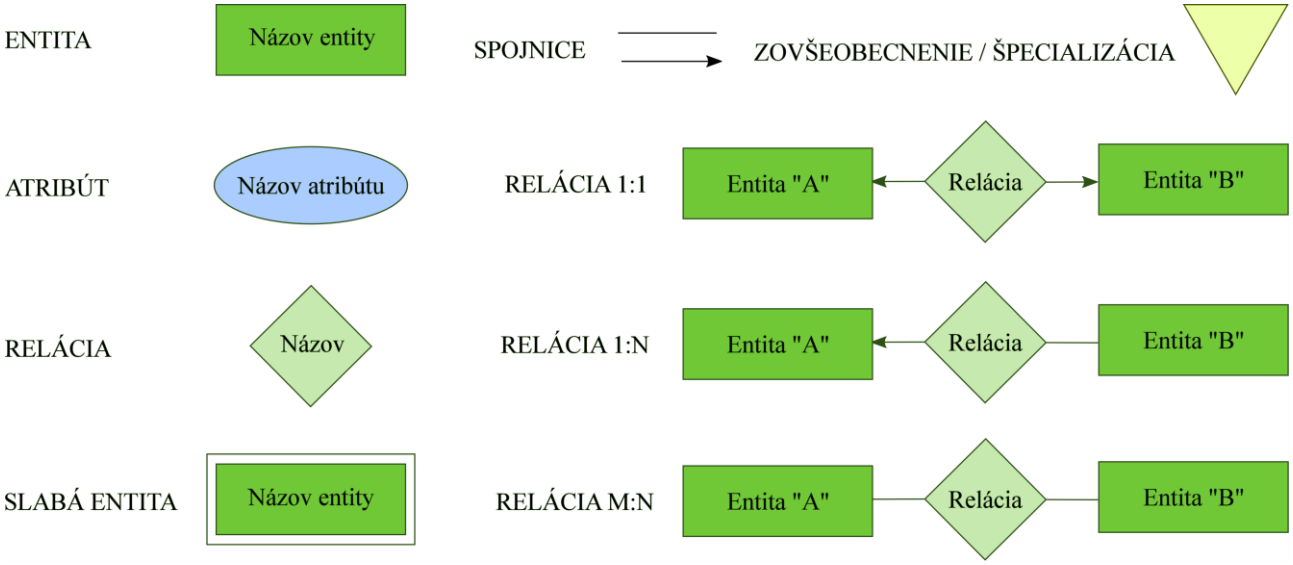
Riešením je agregácia, s pomocou ktorej sa na relácie môžeme pozerať ako na entity vyššej úrovne.



Obrázok č. 3.18 – Výsledok agregácie.

3.14Notácia (symboly) E-R diagramu

E-R diagram je grafický nástroj, preto sú definované pre jeho jednotlivé prvky príslušné symboly – notácie [5].



Obrázok č. 3.19 – Notácia E-R diagramu.

Zhrnutie 3. kapitoly

E-R diagram je grafický nástroj, ktorý sa používa pri návrhu databázových systémov na vyjadrenie modelu údajov. V rámci neho sú definované jednotlivé údajové objekty sveta (entity) a vzájomné vzťahy medzi nimi.

Entita je údajový objekt, o ktorom chceme uchovávať údaje.

Atribúty entity vyjadrujú jej bližšiu charakteristiku/vlastnosti.

Doména atribútu je obor hodnôt, ktorý môže atribút nadobúdať.

Relácia definuje vzťah, spojenie medzi jednotlivými entitami.

Primárny kľúč jednoznačne identifikuje každú inštanciu (každý záznam) v entite.

Cudzí kľúč je vlastne primárny kľúč z inej entity, prostredníctvom ktorého budeme odkazovať na inštancie z druhej entity.

Kardinalita relácie vyjadruje maximálny počet inštancií jednej entity, ktoré môžu asociovať aspoň s jednou inštanciou inej entity v rámci relácie. Poznáme tri typy kardinality: 1 : 1, 1 : N a M : N.

Relácia je okrem kardinality klasifikovaná aj podľa **stupňa relácie.** Poznáme unárnu, binárnu, ternárnu a N-árnu.

Členstvo vo vzťahu (v relácii) medzi entitami rozlišujeme na **povinné** a **nepovinné.**

Zovšeobecnenie entít používame, ak chceme zvýrazniť podobnosť medzi entitami.

Špecializácia je opak zovšeobecnenia. Ak rozčleňujeme entitu na nové entity, ktoré budú dediť určitú časť vlastností z entity, ktorú špecializujeme.

Agregácia – v E-R diagrame nie je možné znázorniť relácie (vzťahy) medzi reláciami. To považujeme za určité obmedzenie. Agregácia umožňuje zoskupiť vzťah medzi dvomi entitami do jedného celku, čím vieme tento problém odstrániť.

Otázky na zopakovanie

1. Na čo slúži E-R diagram?
2. Vlastnými slovami charakterizujte pojmy: entita, atribút, relácia, doména atribútu a primárny kľúč.
3. Uveďte príklady na všetky tri typy kardinality relácie.
4. Na príklade vysvetlite pojem povinné a nepovinné členstvo medzi entitami v relácii.
5. Uveďte príklady na zovšeobecnenie a špecializáciu entít.
6. S pomocou akého nástroja dokážeme vytvoriť vzťah medzi dvomi reláciami?

Doplnkové materiály na štúdium

PRIBILOVÁ, K.: Databázové systémy 1. Pedagogická fakulta Trnavskej univerzity v Trnave, Trnava, 2013. ISBN 978-80-8082-680-2. [cit. 2020-05-01]. Dostupné na internete: ⟨http://pdf.truni.sk/e-ucebnice/databazove-systemy1/⟩.

KASALOVÁ, Z.: Analýza informačného systému jazykové školy, Masarykova univerzita v Brně, Fakulta informatiky, Brno, 2006. [cit. 2020-05-01]. Dostupné na internete: ⟨https://is.muni.cz/th/b55si/bakalarska_praca.pdf⟩.

OTTE, L.: Databázové systémy – Životní cyklus databáze. Ostrava : Vysoká škola Baňská, Technická univerzita Ostrava, Fakulta strojní, 2013.

FORNUSEK, L.: Databázové technológie v podnikových informačných systémoch, Moravská vysoká škola Olomouc, 2009. [cit. 2020-05-01]. Dostupné na internete: ⟨https://theses.cz/id/5x5pam/Fornusek_PIS_2009_-_BP.pdf⟩.

Entity Relationship Diagram (ERD) Tutorial – Part 1. YouTube, Lucidchart, 6. 3. 2017. [cit. 2020-05-01]. Dostupné na internete: ⟨https://www.youtube.com/watch?v=QpdhBUYk7Kk⟩.

Entity Relationship Diagram (ERD) Tutorial – Part 2. YouTube, Lucidchart, 14. 7. 2017. [cit. 2020-05-01]. Dostupné na internete: ⟨https://www.youtube.com/watch?v=-CuY5ADwn24⟩.

4 Údajový model

Po tom ako sme vytvorili E-R diagram, je potrebné určiť spôsob reprezentácie tohto diagramu v báze údajov. Organizáciu údajov navrhujeme prostredníctvom údajového modelu.

Údajový model

- Je množina pravidiel, podľa ktorých sú organizované logické vzťahy medzi údajmi.
- Zobrazuje štruktúru údajov na úrovni ich typov.
- Je prostriedok, prostredníctvom ktorého sú údaje organizované na logickej úrovni.
- Pozostáva z pomenovaných logických jednotiek údajov a vyjadruje vzťahy medzi údajmi.
- Poznáme niekoľko údajových modelov. Rozdiel medzi nimi spočíva v spôsobe reprezentácie vzťahov medzi údajmi.

Poznáme dva druhy vzťahov

- Vzťah atribútov.
- Vzťah asociácií (medzi entitami).

Typy údajových modelov

Poznáme tri základné údajové modely: *hierarchický údajový model, sieťový údajový model* a *relačný údajový model.*

4.1 Hierarchický údajový model

Je špeciálnym prípadom sieťového modelu, na ktorom boli postavené prvé databázové systémy. Tento model je založený na hierarchickej (stromovej) štruktúre údajov vychádzajúcej z koreňa, pri ktorej sa využíva vzťah nadradenosti a podradenosti (rodič a potomok) na opis vzájomných vzťahov. Údajovým štruktúram na jednotlivých úrovniach sa hovorí uzly. Ak z uzla nevychádza ďalšia vetva, nazývame ho list. Aj keď tento model často vhodne vystihuje hierarchiu vzťahov v realite (napríklad štát → kraj → okres → obec) a v minulosti si našiel v praxi široké uplatnenie, v súčasnosti sa v databázových systémoch využíva len zriedkavo.

Hierarchický model sa priamo neopiera o matematickú teóriu, avšak čiastočne používa terminológiu teórie grafov (graf, uzol, hrana). Napríklad záznam zodpovedá uzlu v grafe [23].

Pre hierarchickú štruktúru platí, že:

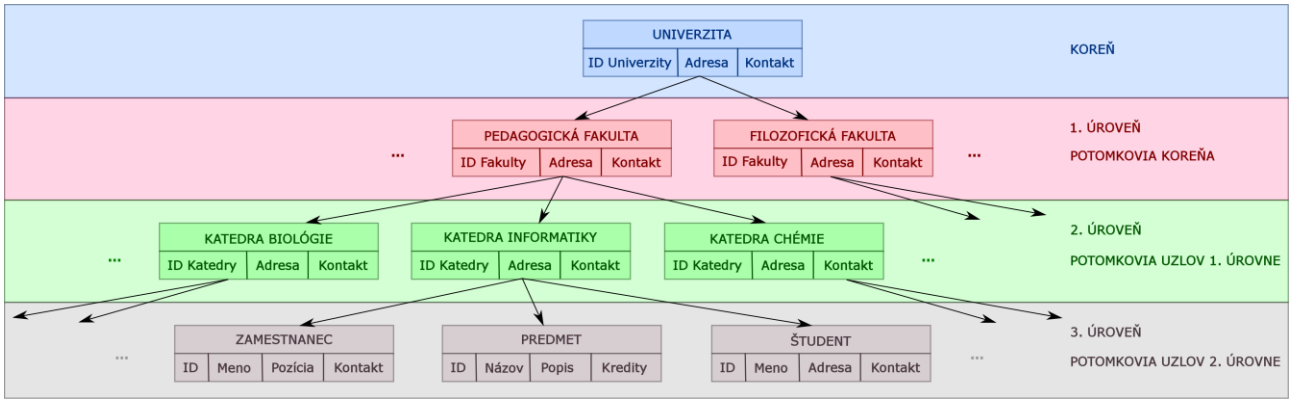
- Každý potomok má len jedného rodiča.
- Existuje jediný rodič, ktorý nie je potomok (označujeme ho aj ako koreň stromovej štruktúry).
- Potomok v jednom vzťahu môže byť rodičom v inom [11].

Hierarchické modelovanie zodpovedá vzťahom typu 1 : N a 1 : 1. Základnou výhodou hierarchického systému je rýchle vyhľadávanie entít, ktoré sú jej podriadené. Naopak, vyhľadávanie entity, ktorá je

nadriadená znamená sekvenčné prehľadávanie celého stromu (v súčasnosti neefektívne). Pri vyhľadávaní údajov sa postupuje vždy od koreňového uzla (záznamu) cez stromovú štruktúru k požadovanému uzlu, pričom ku každému záznamu existuje len jediná cesta.

Nevýhody hierarchického modelu:

- Neumožňuje modelovanie väzieb M : N.
- Zložité vkladanie nových uzlov (záznamov) a takisto zložité mazanie existujúcich uzlov. To znamená, že vzťahy medzi údajmi sú zabudované priamo v štruktúre databázy a je veľmi problematické ich v prípade potreby neskôr meniť.
- Neprirodzená organizácia údajov.
- Tvorba dopytov je závislá od logickej štruktúry databázy.



Obrázok č. 4.1 – Hierarchický údajový model.

4.2 Sieťový údajový model

Sieťový model vznikol približne v rovnakom časovom období ako hierarchický model. Využíva sieťovú štruktúru údajov, v ktorej (na rozdiel od hierarchického modelu) môže byť každý prvok v sieti zviazaný s ktorýmkoľvek iným prvkom (každý prvok môže mať aj viac rodičov) [12].

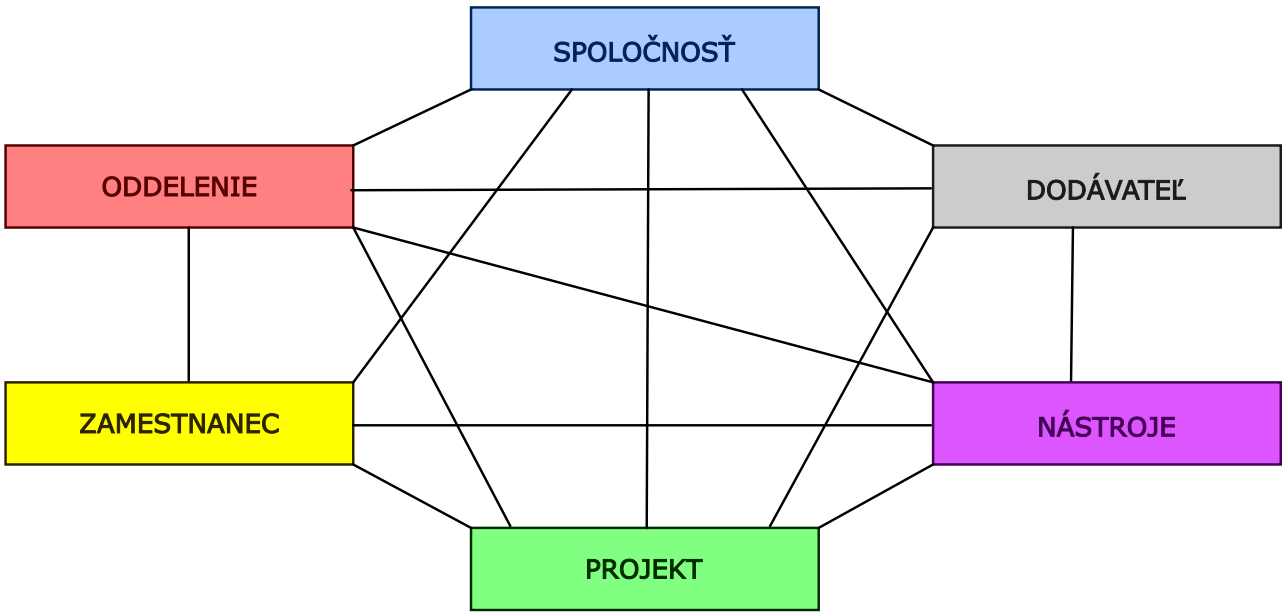
Tvoríme ho prostredníctvom orientovaného grafu, v ktorom sú entity zobrazené s použitím uzlov a vzťahy s použitím odkazov (prepojovacích čiar). Tento model teda údaje predstavuje ako množinu záznamov (entít) a párových vzťahov medzi nimi, ktoré môžu mať vzťahy typu 1 : 1, 1 : N alebo M : N [4].

Tým rozširuje možnosti hierarchického modelu a umožňuje vernejšiu reprezentáciu vzťahov, avšak na úkor jednoduchosti modelu.

Atribúty záznamov v sieťových modeloch môžu byť jednoduché, opakujúce sa, zložené alebo zložené opakujúce sa. Na rozdiel od predchádzajúceho hierarchického údajového modelu, sieťový model môže obsahovať aj cykly a slučky [4].

Sieťový graf sa usiluje odstrániť hlavné nevýhody hierarchického modelu. Model zaručuje minimálnu redundanciu (výskyt rovnakých údajov), pretože každá entita sa nachádza v báze údajov iba jeden raz. Ak sa nachádza táto entita vo viacerých vzťahoch, vedie k nej viac odkazov v rôznych vzťahových súboroch. Z toho vyplýva aj základná nevýhoda sieťových modelov, ktorou je veľmi náročná modifikovateľnosť. Napríklad so zmenou počtu entít (najmä pri odstraňovaní entít) je potrebné prechádzať všetky

vzťahové súbory a opravovať (rušiť) odkazy. Sieťový model bol z dôvodu svojej implementačnej náročnosti a zložitosti používaný pomerne málo [13].



Obrázok č. 4.2 – Sieťový údajový model.

4.3 Relačný údajový model (RDM z angl. relation data model)

Relačný údajový model vychádza z teórie množín. Vznikol v roku 1970, kedy ho prvý raz publikoval Dr. E. F. Codd. Väčšina dnešných databázových systémov je založená práve na relačnom údajovom modeli. Ide o logické vyjadrenie väzieb medzi jednotlivými údajmi. Manipulácia s údajmi je založená na operáciách relačnej algebry. Tieto operácie zahŕňajú množinové operácie ako **zjednotenie, prienik, rozdiel** a **karteziánsky súčin.** Medzi špeciálne **relačné operácie** patria **selekcia, spojenie, delenie** a **projekcia** [10].

Tabuľka č. 4.1 – Analógia pojmov: relačný údajový model – systémy súborov [10].

Relačný pojem	**Reprezentácia**	**Súborová analógia**
Relácia	Tabuľka	Súbor
N-tica	Riadok	Záznam
Atribút	Stĺpec	Položka
Hodnota atribútu	Hodnota bunky	Hodnota položky

Medzi základné pojmy relačného údajového modelu patrí:

- **Relácie** – sú základnými prvkami relačného údajového modelu. V implementácii databázového systému sú relácie reprezentované ako dvojrozmerné **tabuľky.** Každá tabuľka potom zastupuje konkrétnu entitu reálneho sveta. Relácia sa skladá z **atribútov** a z **usporiadaných n-tíc.**

- **Atribúty** – sú reprezentované jednotlivými stĺpcami tabuliek a vyjadrujú konkrétne vlastnosti o entitách. Hodnota atribútu je vyjadrená konkrétnym údajom v bunke tabuľky.
- **Primárny kľúč** – jednoznačný identifikátor v tabuľke, ktorý je reprezentovaný jedným alebo viacerými stĺpcami tabuľky.
- **N-tica** – zodpovedá jednému riadku tabuľky.
- **Doména** – vyjadruje množinu dovolených hodnôt, ktoré sa v stĺpci môžu vyskytovať. Ide o skalárne hodnoty rovnakého údajového typu. Skalárna hodnota je hodnota, ktorá reprezentuje najmenšiu sémantickú jednotku údajov. Napríklad pre atribút „Meno študenta" je údaj „Peter" skalárna hodnota. **Domény** zohrávajú v relačnom modeli veľmi dôležitú úlohu, pretože **sú základnými nositeľmi informácie.**

Základné vlastnosti každej relácie (tabuľky) [10]

- **Neobsahuje duplicitné n-tice** – táto vlastnosť vychádza z faktu, že telo relácie je matematická množina (množina n-tíc), a podľa definície neobsahuje duplicitné prvky. Základným predpokladom na splnenie tohto bodu je nutná existencia **primárneho kľúča** v každej relácii.
- **N-tice sú neusporiadané (zhora nadol)** – n-tice tvoria matematickú množinu (nie je usporiadaná) hoci z praktického hľadiska je často vhodné udržiavať reláciu ako usporiadanú množinu. Na druhej strane neusporiadanosť umožňuje zjednodušiť algoritmy na prácu nad reláciou.
- **Atribúty sú neusporiadané (zľava doprava)** – atribúty takisto tvoria matematickú množinu (nie je usporiadaná). Táto vlastnosť hovorí o tom, že k jednotlivým stĺpcom alebo k hodnotám atribútu pristupuje prostredníctvom identifikátora (mena atribútu) a nie podľa pozície v riadku. To umožní tvorbu programov nezávislých od údajov.
- **Hodnoty atribútov sú atomické** – to znamená, že každému atribútu je priradená vždy len jedna hodnota a nie množina hodnôt. Je potrebné, aby všetky atribúty boli atomické, čím dosiahneme to, že v relácii sa nemôžu vyskytovať tzv. opakujúce sa skupiny atribútov, ktoré sa vyskytujú v nenormalizovaných záznamoch. **Relácii bez opakujúcich sa skupín hovoríme, že je normalizovaná.**

Študent

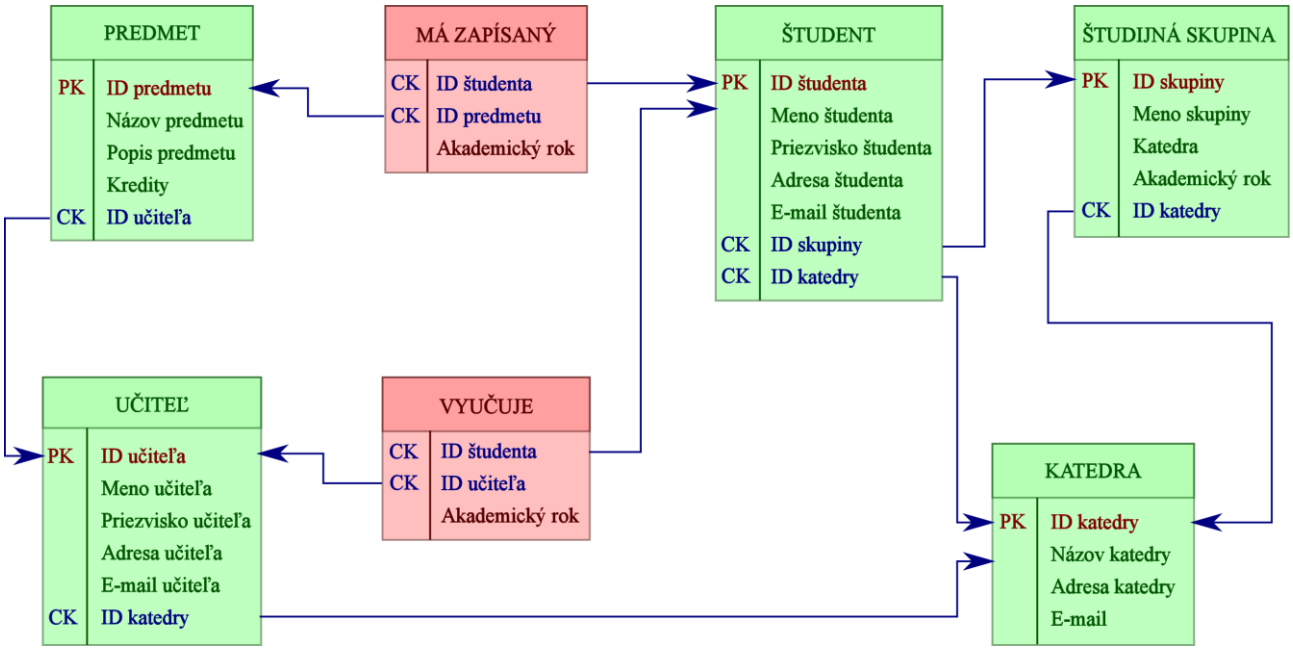
ID študenta	Meno študenta	Priezvisko študenta	Adresa študenta	E-mail študenta	Študijná skupina
Š_0001	Peter	Novák	Jarná 37, 94905 Nitra	peter.novak@univerzita.sk	Sk_0001
Š_0002	Zuzana	Pekná	Veterná 15, 91105 Trenčín	zuzana.pekna@univerzita.sk	Sk_0002
Š_0003	Tomáš	Novotný	Hlboká 17, 91701 Trnava	tomas.novotny@univerzita.sk	Sk_0003
Š_0004	Monika	Kráľová	Čerešňová 22, 91701 Trnava	monika.kralova@univerzita.sk	Sk_0001
Š_0005	Rudolf	Veľký	J. Kráľa 45, 94905 Nitra	rudolf.velky@univerzita.sk	Sk_0001
Š_0006	Ema	Veselá	Slnečná 36, 91105 Trenčín	ema.vesela@univerzita.sk	Sk_0002

Obrázok č. 4.3 – Príklad relácie (tabuľky s M riadkami a N stĺpcami).

Relačný údajový model podľa pôvodnej definície vychádzal z nasledujúcich požiadaviek:

- Zabezpečiť vysoký stupeň údajovej nezávislosti.

- Zabezpečiť minimálnu redundanciu údajov spolu s konzistenciou údajov s podporou sémantiky jazyka.
- Sprístupniť databázu prostredníctvom množinovo orientovaného neprocedurálneho jazyka.
- Umožniť jednoduchým spôsobom reštrukturalizáciu a rast údajového modelu [9].



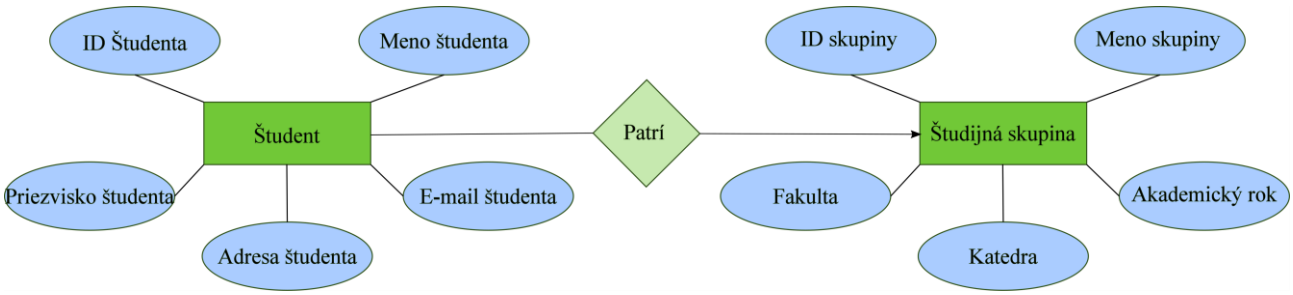
Obrázok č. 4.4 – Relačný údajový model.

4.4 Cudzí kľúč (CK)

Cudzí kľúč (CK) je atribút, ktorý je vlastne primárnym kľúčom (PK) inej entity (tabuľky) zo spoločnej relácie (vzťahu) medzi dvomi prípadne viacerými entitami. Budeme ho využívať pri transformácii E-R modelu do relačného modelu. Prostredníctvom CK vieme vytvoriť reláciu, vzťah, určitý súvis medzi údajmi z dvoch entít (platí vo väčšine relácií) prípadne viacerých entít. Zabezpečuje vytvorenie odkazu na inštanciu z inej tabuľky, čím sa odstraňuje výskyt rovnakých (duplicitných) údajov v databáze (odstráni sa nežiadúca redundancia údajov). Samotná entita (tabuľka) môže obsahovať aj viac cudzích kľúčov, záleží od počtu a typu vzťahov (relácií) s inými tabuľkami. CK teda zabezpečuje zistenie všetkých informácií o určitej inštancii z inej („pridruženej“) tabuľky.

Príklad použitia CK v relácii 1 : N

Uvažujme o relácii 1 : N medzi entitami „Študijná skupina“ a „Študent.“ Vieme, že jeden konkrétny študent môže patriť iba do jednej študijnej skupiny, pričom do jednej (1) študijnej skupiny môže patriť viacero (N) študentov. Medzi týmito entitami teda existuje relácia (vzťah), ktorý nazveme „Patrí.“

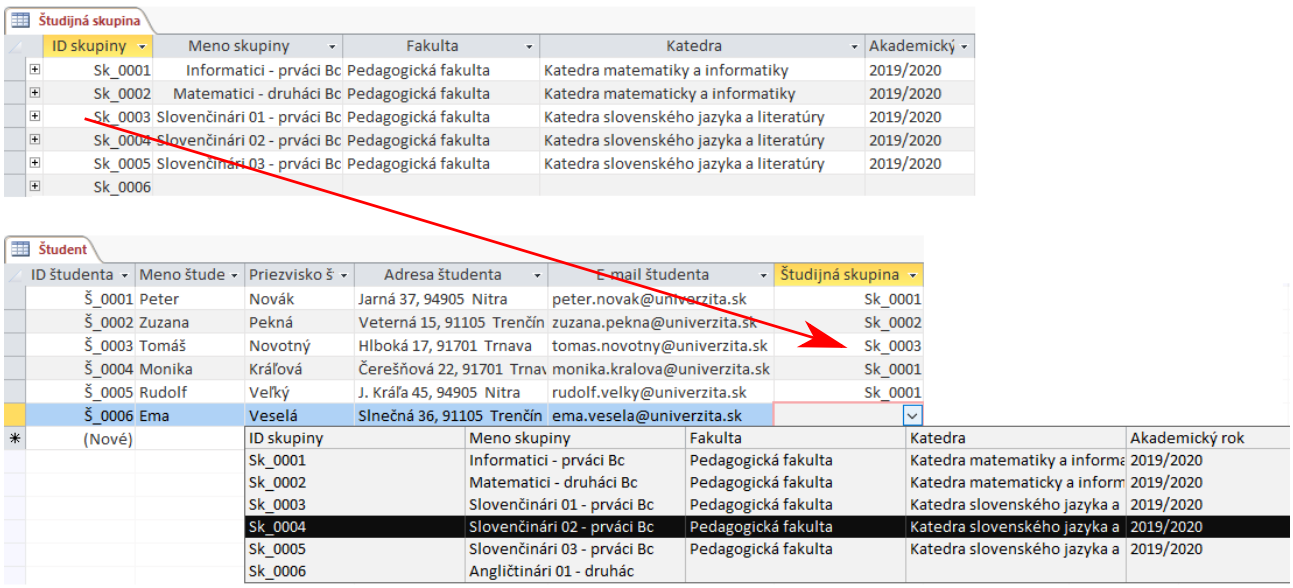


Obrázok č. 4.5 – Relácia 1 : N.

Tabuľka „Študijná skupina“ má atribúty: „ID skupiny“ (PK), „Meno skupiny,“ „Fakulta,“ „Katedra,“ „Akademický rok.“

Tabuľka „Študent“ má atribúty: „ID študenta“ (PK), „Meno študenta,“ „Priezvisko študenta,“ „Adresa študenta,“ „Email študenta.“

Fakt, že medzi tabuľkami „Študijná skupina“ a „Študent“ existuje relácia (vzťah s názvom „Patrí“) znamená, že musíme prepojiť tieto dve tabuľky, aby bolo jasné, ktorý študent patrí do ktorej konkrétnej študijnej skupiny. Bolo by nepraktické, ak by sme do tabuľky „Študent“ dopracovali atribúty „Meno študijnej skupiny“ prípadne ďalšie atribúty z tabuľky „Študijná skupina.“ Po prvé by samotné pridávanie nových záznamov do systému bolo prácnejšie a po druhé by sa v systéme vyskytla duplicita údajov, čím by sme neodstránili problém s redundanciou a to nechceme. Na takéto prípady slúži prepojenie medzi tabuľkami prostredníctvom primárneho kľúča z tabuľky „Študijná skupina,“ ktorý vložíme do tabuľky „Študent“ ako cudzí kľúč. Tento spôsob prepojenia tabuliek sa používa na relácie typu 1 : N, pričom cudzí kľúč vkladáme do tabuľky na strane N, čiže v tomto prípade do tabuľky „Študent.“



Obrázok č. 4.6 – Cudzí kľúč v relácii s kardinalitou 1 : N.

Tabuľka „Študent“ bude potom vyzerať takto:

> **„Študent“** („ID študenta“ (PK), „Meno študenta,“ „Priezvisko študenta,“ „Adresa študenta,“ „Email študenta,“ „ID skupiny“ (CK)).

Príklad použitia CK v relácii 1 : 1

Ako sme už spomínali, výskyt relácie 1 : 1 je skôr výnimočným prípadom a väčšinou nie je potrebné ju ani použiť (dôvody sme uviedli v kapitole o reláciách). Relácia 1 : 1 vyjadruje vzťah kedy záznam z jednej tabuľky zodpovedá iba jednému záznamu z druhej tabuľky tejto relácie. Tieto typy relácie obvykle riešime ako špeciálny prípad relácie 1 : N s použitím cudzieho kľúča.

Príklad použitia CK v relácii M : N

Pri reláciách M : N vznikajú určité komplikácie, ktoré teraz vysvetlíme na konkrétnom príklade. Uvažujme znova o školskom informačnom systéme, konkrétne o vzťahu medzi entitami „Študent“ a „Predmet.“ Vieme, že jeden študent môže mať zapísaných viac (M) predmetov a takisto jeden predmet môže mať zapísaných viacero (N) študentov. Tu vzniká komplikácia, pri ktorej vzťah M : N nevieme pretransformovať do relačného údajového modelu.

Tabuľka „Študent“ má atribúty: „ID študenta“ (PK), „Meno študenta,“ „Priezvisko študenta,“ „Adresa študenta,“ „Email študenta.“

Tabuľka „Predmet“ má atribúty: „ID predmetu“ (PK), „Názov predmetu,“ „Opis predmetu,“ „Kredity.“

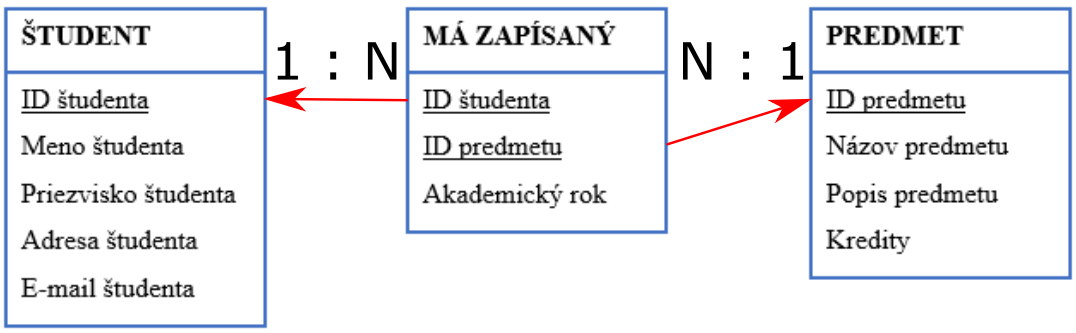
Riešenie tohto problému spočíva v rozdelení relácie M : N na dve relácie 1 : N s použitím novej asociatívnej tabuľky (join table). Túto novovytvorenú tabuľku môžeme nazvať napríklad rovnako ako túto reláciu čiže „Má zapísaný.“

Novovzniknutá tabuľka „Má zapísaný“ má potom atribúty: „ID študenta“ (CK), „ID predmetu“ (CK), „Akademický rok.“

V prípade, že jeden študent bude mať zapísaných viac predmetov, tak bude tabuľka „Má zapísaný“ obsahovať viac záznamov s atribútom „ID študenta,“ ku ktorému bude priradený vždy iné „ID predmetu.“ To platí aj v opačnom prípade, kedy každý záznam „ID predmetu“ v novovzniknutej tabuľke bude mať priradený vždy inú hodnotu „ID študenta.“ Novovzniknutá tabuľka „Má zapísaný“ obsahuje teda cudzie kľúče z tabuliek „Študent“ a „Predmet.“ Do tejto tabuľky sme ešte pridali atribút „Akademický rok,“ aby sme vedeli kedy mal študent zapísaný konkrétny predmet. Primárny kľúč tabuľky „Má zapísaný“ je vytvorený s využitím obidvoch cudzích kľúčov a teda kombináciou atribútov „ID študenta“ a „ID predmetu.“

Medzi tabuľkami „Študent“ a „Má zapísaný“ je relácia 1 : N. Takisto medzi tabuľkami „Predmet“ a „Má zapísaný“ je relácia 1 : N. Takto jednoducho sme sa zbavili počiatočného vzťahu M : N medzi tabuľkami „Študent“ a „Predmet.“

Na tomto príklade môžeme vidieť, kedy môže byť primárny kľúč vytvorený aj s použitím viacerých atribútov dohromady. Pre každý záznam (inštanciu) z tabuliek „Študent“ a „Predmet“ zostáva v platnosti, že jeho primárny kľúč je stále jedinečný. Každý záznam v tabuľke „Má zapísaný“ teda musí obsahovať jedinečnú kombináciu hodnôt cudzích kľúčov, ktoré sú oba súčasťou primárneho kľúča tejto tabuľky. Pridaný atribút „Akademický rok“ má skôr informatívny charakter a tabuľka môže existovať aj bez neho.



Obrázok č. 4.7 – Cudzie kľúče v relácii s kardinalitou M : N.

Zhrnutie 4. kapitoly

Údajový model – je množina pravidiel, podľa ktorých sú organizované logické vzťahy medzi údajmi v databáze.

Typy údajových modelov:

- **Hierarchický** – je založený na hierarchickej (stromovej) štruktúre údajov. Jeho modelovanie zodpovedá vzťahom typu 1 : N a 1 : 1.
- **Sieťový** – je orientovaný graf, v ktorom sú entity zobrazené s použitím uzlov a vzťahy prostredníctvom prepojovacích čiar. Môžeme modelovať vzťahy typu 1 : 1, 1 : N, M : N. Z dôvodu svojej implementačnej náročnosti a zložitosti bol používaný pomerne málo.
- **Relačný** – relácie sú reprezentované ako dvojrozmerné tabuľky. Každá tabuľka potom zastupuje konkrétnu entitu reálneho sveta. Modelujeme vzťahy typu 1 : 1, 1 : N, M : N.

Definícia relácie v relačnom údajovom modeli

- Je reprezentovaná ako tabuľka (matica zostavená z M riadkov a N stĺpcov).
- Jeden riadok reprezentuje jeden záznam.
- Nemôžu byť dva rovnaké riadky, odlišujú sa minimálne v hodnote primárneho kľúča.
- Poradie jednotlivých riadkov je preto nepodstatné.
- Jeden stĺpec reprezentuje jeden atribút.
- Každý atribút (stĺpec) má jedinečný názov odlišný od ostatných atribútov.
- Poradie jednotlivých stĺpcov je preto nepodstatné.
- Každý atribút má svoju doménu, ktorá vyjadruje množinu dovolených hodnôt.

Cudzí kľúč – je atribút, ktorý je vlastne primárnym kľúčom inej entity zo spoločnej relácie. Prostredníctvom CK vieme vytvoriť reláciu medzi údajmi z dvoch tabuliek.

Otázky na zopakovanie

1. Charakterizujte údajový model.

2. Aké typy údajových modelov poznáme?
3. Opíšte jednotlivé údajové modely.
4. Vymenujte a opíšte základné zložky relačného údajového modelu.
5. Charakterizujte cudzí kľúč.
6. Uveďte príklad na použitie cudzieho kľúča.

Doplnkové materiály na štúdium

Microsoft SQL Server 2008: A Beginner's Guide, Relational Database Systems – An Introduction, [cit. 2020-05-01]. Dostupné na internete: ⟨http://www.mhprofessional.com/downloads/products/0071546383/0071546383_ch01.pdf⟩.

SZABÓ, P.: Databázové a informačné systémy – Databázová terminológia – relačný dátový model, TUKE, 2005. [cit. 2020-05-01]. Dostupné na internete: ⟨https://spseke.sk/tutor/prednasky/dbs/rdm.html⟩.

GORBÁR, P.: Editor ER diagramu s podporou transformace do relačního modelu a SQL, Univerzita Karlova v Praze, 2007. [cit. 2020-05-01]. Dostupné na internete: ⟨https://www.ksi.mff.cuni.cz/~holubova/bp/Gorbar.pdf⟩.

DURAČIOVÁ, R.: Databázové systémy v GIS, Slovenská technická univerzita v Bratislave, 2014, ISBN 978-80-227-4292-4. [cit. 2020-05-01]. Dostupné na internete: ⟨https://www.researchgate.net/profile/Renata_Duraciova/publication/317102680_Databazove_systemy_v_GIS/links/5926a0f5458515e3d457fa87/Databazove-systemy-v-GIS.pdf⟩.

5 Transformácia E-R modelu do relačného údajového modelu

Po konceptuálnom návrhu databázy (vytvorenie E-R modelu) nasleduje logický návrh databázy. Pri logickom návrhu budeme transformovať E-R model do relačného modelu, ktorý sme predstavili v predchádzajúcej kapitole.

Transformácia E-R modelu do relačného modelu je úsilím o implementáciu E-R konceptu v relačnej databáze. Aplikáciou tejto transformácie vznikne zoznam tabuliek, ktoré obsahujú (v ideálnom prípade) všetky integritné obmedzenia špecifikované v E-R diagrame [14].

Vzhľadom na to, ako sú si tieto modely blízke, je aj transformácia relatívne jednoduchá a priamočiara. Často sú prvky takmer rovnaké, mení sa len terminológia.

- Entitné množiny a ich atribúty sú mapované na relácie/tabuľky a ich atribúty.
- Vzťahové množiny s atribútmi sú mapované na relácie.
- Vzťahové množiny so vzťahom M : N sú taktiež mapované na relácie. Vytvára sa **asociatívna tabuľka,** v praxi často nazývaná aj „join table.“ Tá obsahuje N cudzích kľúčov, ktorých kombinácia je unikátna, často fungujú ako kompozitný primárny kľúč a odkazujú na primárne kľúče ostatných entít vo vzťahu. Môže obsahovať svoje atribúty a môže obsahovať aj iný, vlastný primárny kľúč.
- Jednoduché vzťahové množiny so vzťahom 1 : N sú mapované ako cudzí kľúč vložený do relácie reprezentujúcej násobnú entitnú množinu vo vzťahu. Často sa nazýva taktiež agregačný vzťah.
- Jednoduché vzťahové množiny so vzťahom 1 : 1 sú mapované ako cudzí kľúč bez preferencie relácie.
- Slabé entitné množiny sú špeciálnym prípadom 1 : N vzťahu, kde je násobná entitná množina podradená a nemôže existovať bez nadradenej. Cudzí kľúč k nadradenej relácii sa stáva primárnym kľúčom. Často sa nazýva taktiež kompozitný vzťah.
- Dedičnosť alebo generalizácia, je špeciálna konštrukcia, kde viacero entitných množín zdieľa spoločné atribúty [15].

Postup pri transformácii E-R modelu do relačného modelu

Výsledkom tejto transformácie je vlastne tvorba tabuliek, tak ako budú vyzerať v databázovom systéme. Štruktúru jednotlivých tabuliek vytvoríme z informácií uvedených v E-R modeli.

5.1 Reprezentácia entít

Pre každú množinu entít vytvoríme jednu tabuľku, v ktorej počet stĺpcov zodpovedá počtu atribútov entity. Názov tabuľky odpovedá typu entity a názov stĺpca odpovedá názvu atribútu entity.

5.1.1 Silná entita

Primárny kľúč v relačnej schéme zodpovedá primárnemu kľúču entity. Tabuľka môže obsahovať viac stĺpcov ako je atribútov v tejto entite. Pri reláciách, ktorých kardinalita je typu 1 : N, môže obsahovať aj cudzie kľúče.

5.1.2 Slabá entita

Pre každý atribút slabej entity musí tabuľka obsahovať stĺpec alebo stĺpce pre atribúty, ktoré tvoria primárny kľúč silnej entity, na ktorej je slabá entita existenčne závislá [4].

5.2 Reprezentácia vzťahov

Technika transformácie relačných vzťahov závisí od viacerých aspektov, ktoré si teraz ukážeme na jednotlivých príkladoch.

5.2.1 Povinné členstvo vo vzťahu

Uvažujeme o relácii typu 1 : N medzi entitami „Katedra“ a „Učiteľ.“ Pričom platí, že jeden (1) učiteľ môže pôsobiť iba na jednej (1) katedre, ale na jednej (1) katedre môže pôsobiť viac (N) učiteľov. Členstvo entity „Učiteľ“ je v tomto vzťahu **povinné,** pretože každý učiteľ musí byť zaradený do konkrétnej katedry. Naopak, entita „Katedra“ má v tomto vzťahu nepovinné členstvo, pretože za určitých okolností môže existovať aj bez učiteľov.

Obrázok č. 5.1 – Transformácia relačných vzťahov, povinné členstvo entity vo relácii 1 : N.

Riešenie

Pre každú entitu vytvoríme tabuľku. V tabuľke „Učiteľ“ vzniká nový atribút, a síce pridáme do nej primárny kľúč z entity „Katedra“ ako cudzí kľúč.

Vzniknuté relačné tabuľky

„Katedra“ („ID katedry“ (PK), „Názov katedry,“ „Adresa katedry,“ „Kontakt“).

„Učiteľ“ („ID učiteľa“ (PK), „Meno učiteľa,“ „Priezvisko učiteľa,“ „Adresa učiteľa,“ „Email učiteľa,“ „ID katedry“ (CK)).

5.2.2 Nepovinné členstvo vo vzťahu

Uvažujeme o relácii typu 1 : N medzi entitami „Čitateľ“ a „Kniha.“ Pričom platí, že jednu (1) knihu si môže požičať iba jeden (1) čitateľ, ale jeden (1) čitateľ si môže požičať aj viac (N) rôznych kníh. Členstvo entity „Kniha“ je v tomto vzťahu **nepovinné,** pretože konkrétna kniha nemusí byť požičaná žiadnym čitateľom. To znamená, že musíme zaviesť nulovú hodnotu a teda do tabuľky nemôžeme vložiť nový atribút ako cudzí kľúč.

Obrázok č. 5.2 – Transformácia relačných vzťahov, nepovinné členstvo v relácii 1 : N [4].

Riešenie

Riešením je v tomto vzťahu vznik novej tabuľky, do ktorej budeme vkladať údaje o tom, ak si nejaký čitateľ požičia nejakú knihu. Výsledkom tohto vzťahu sú tri tabuľky: „Čitateľ," „Kniha" a „Pôžička" [4].

Vzniknuté relačné tabuľky

„Čitateľ" („ID čitateľa" (PK), „Meno čitateľa," „Priezvisko čitateľa," „Adresa čitateľa").

„Kniha" („ID knihy" (PK), „Názov knihy," „Autor knihy," „Žáner").

„Pôžička" („ID čitateľa" (CK), „ID knihy" (CK), „Dátum pôžičky") [4].

5.2.3 Reprezentácia binárnych vzťahov s kardinalitou M : N

Uvažujme o vzťahu medzi entitami „Študent" a „Predmet." Vieme, že jeden študent môže mať zapísaných viac (M) predmetov a takisto jeden predmet môže mať zapísaných viacero (N) študentov.

Obrázok č. 5.3 – Transformácia relačných vzťahov, binárny vzťah M : N.

Riešenie

Riešení je vznik novej samostatnej asociatívnej tabuľky, ktorá obsahuje primárny kľúč z entity „Študent," takisto primárny kľúč z entity „Predmet," plus môže obsahovať vlastné atribúty.

Vzniknuté relačné tabuľky

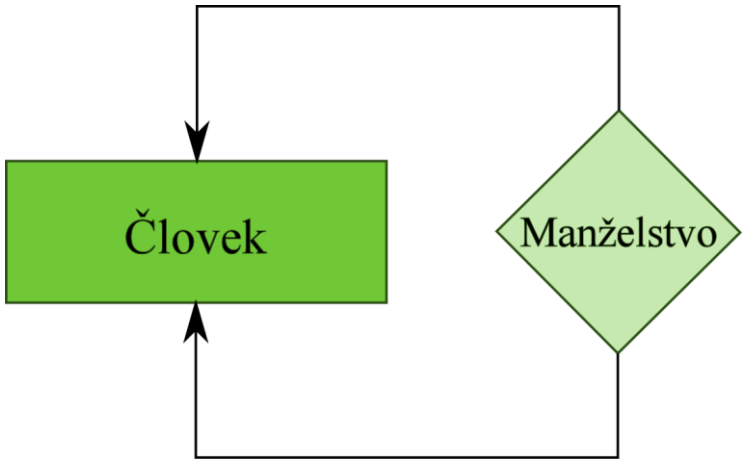
„Študent" („ID študenta" (PK), „Meno študenta," „Priezvisko študenta," „Adresa študenta," „Email študenta").

„Predmet" („ID predmetu" (PK), „Názov predmetu," „Opis predmetu," „Kredity").

„Má zapísaný" („ID študenta" (CK), „ID predmetu" (CK), „Akademický rok").

5.2.4 Reprezentácia relačného vzťahu 1 : 1 medzi inštanciami jednej entity

Uvažujme o relácii „Manželstvo," pri ktorom platí, že jeden (1) človek (muž) môže byť v manželskom zväzku iba s jedným (1) človekom (žena).



Obrázok č. 5.4 – Reprezentácia relačného vzťahu 1 : 1 medzi inštanciami jednej entity.

Riešenie 1 – nepovinné členstvo

Vychádzame z toho, že nie každý človek musí byť v manželskom zväzku. Riešením je vznik novej samostatnej tabuľky, v ktorej budeme uchovávať údaje iba o tých ľuďoch, ktorý sú v manželskom zväzku.

Vzniknuté tabuľky

„Človek" („Rodné číslo" (PK), „Meno človeka," „Priezvisko človeka," „Adresa človeka," „Pohlavie").

„Manželstvo" („Rodné číslo muž" (CK), „Rodné číslo žena" (CK), „Dátum svadby").

Ako vidíme, cudzie kľúče v asociatívnej tabuľke nemusia mať rovnaké názvy ako ich názvy v tabuľke kde sú primárnym kľúčom. Vznikne tým prehľadnejšia tabuľka.

Riešenie 2 – povinné členstvo

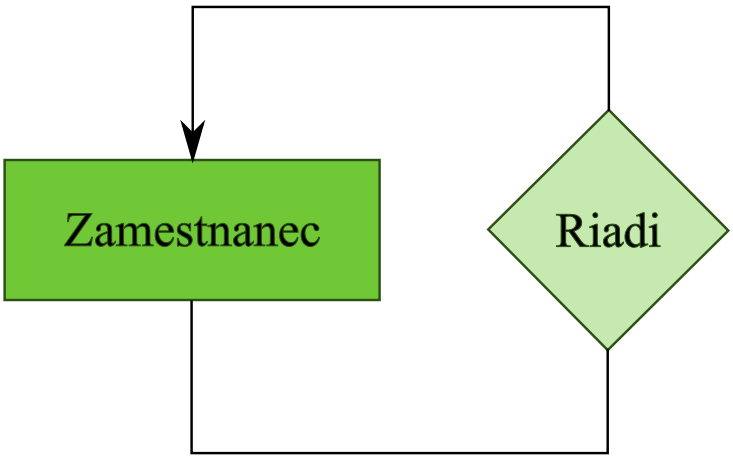
Vychádzame z predpokladu, že by sme každému človeku museli povinne priradiť partnera. Stačí ak do vzniknutej tabuľky vložíme nový atribút, v tomto prípade „Rodné číslo partnera" ako cudzí kľúč.

Vzniknutá tabuľka

„Človek" („Rodné číslo" (PK), „Meno človeka," „Priezvisko človeka," „Adresa človeka," „Pohlavie," „Rodné číslo partnera" (CK)).

5.2.5 Reprezentácia relačného vzťahu 1 : N medzi inštanciami jednej entity

Uvažujme o relácii „Riadi," pri ktorom platí, že jeden (1) zamestnanec môže byť nadriadený viacerým (N) iným zamestnancom, ale naopak platí, že jeden (1) zamestnanec môže mať iba jedného (1) nadriadeného.



Obrázok č. 5.5 – Reprezentácia relačného vzťahu 1 : N medzi inštanciami jednej entity.

Riešenie 1 – nepovinné členstvo

Vychádzame z predpokladu, že nie každý zamestnanec musí mať priradeného svojho nadriadeného. Riešením je vznik novej samostatnej asociatívnej tabuľky, v ktorej budeme uchovávať údaje iba o tých zamestnancoch, ktorý majú nadriadeného respektíve podriadeného zamestnanca.

Vzniknuté tabuľky

„Zamestnanec“ („ID zamestnanca“ (PK), „Meno zamestnanca,“ „Priezvisko zamestnanca,“ „Adresa zamestnanca,“ „Telefón zamestnanca,“ „Email zamestnanca“).

„Riadi“ („ID zamestnanca vedúci“ (CK), „ID zamestnanca podriadený“ (CK)).

Riešenie 2 – povinné členstvo

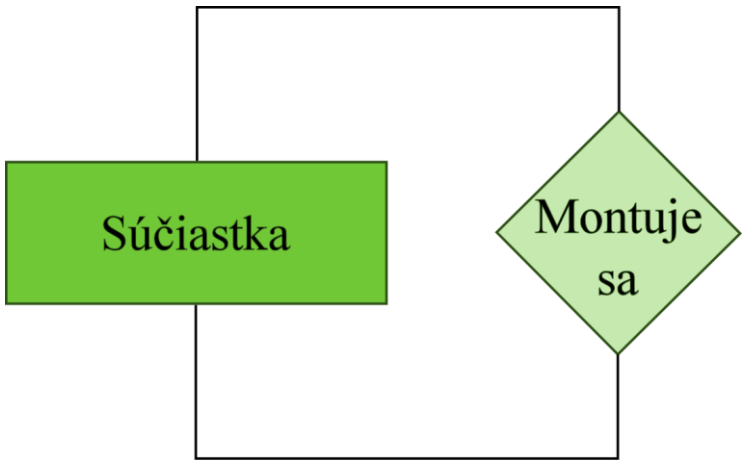
V prípade ak ide o povinné členstvo, postupujeme ako v predchádzajúcich prípadoch a síce pridáme do tabuľky zamestnanec nový atribút „ID zamestnanca vedúci“ ako cudzí kľúč.

Vzniknutá tabuľka

„Zamestnanec“ („ID zamestnanca“ (PK), „Meno zamestnanca,“ „Priezvisko zamestnanca,“ „Adresa zamestnanca,“ „Telefón zamestnanca,“ „Email zamestnanca,“ „ID zamestnanca vedúci“).

5.2.6 Reprezentácia relačného vzťahu M : N medzi inštanciami jednej entity

Uvažujeme o relácii „Montuje sa,“ pri ktorej platí, že (M) rôznych súčiastok môže byť vo vzťahu s (N) ďalšími súčiastkami, pričom tvoria jeden celok (výrobok, zloženú súčiastku…).



Obrázok č. 5.6 – Reprezentácia relačného vzťahu M : N medzi inštanciami jednej entity.

Riešenie

Riešení je v tomto príklade vznik novej tabuľky „Montuje sa," do ktorej budeme vkladať údaje o súčiastkach, ktoré tvoria celok s inými súčiastkami. PK v tabuľke budú spoločne tvoriť atribúty: „ID zostavy" a „ID súčiastky." Nemôže teda nastať situácia, v ktorej by sme mali dva rovnaké PK.

Ako príklad si predstavme motor, ktorý sa skladá z viacerých súčiastok a takisto sám motor je ako celok súčiastka, ktorú využívame potom ďalej.

Vzniknuté tabuľky

„Súčiastka" („ID súčiastky" (PK), „Názov súčiastky").

„Montuje sa" („ID zostavy" (PK), „Názov zostavy," „ID súčiastky" (PK), „Počet súčiastok").

Súčiastka

ID súčiastky	Názov súčiastky
S001	Vačkový hriadeľ
S002	Výfukové potrubie
S003	Pružina ventilu
S004	Ventil
S005	Hlava valcon
S006	Sedlo ventilu
S007	Blok valcon
S008	Piest
S009	Ojnica
S010	Piestne krúžky
S011	Kľukový hriadeľ
S012	Turbodúchadlo
S013	Časticový filter
S014	Plastový kryt
S015	Piezo vstrekovať
S016	Výva poháňaná vačkovým hriadelom
S017	Plastové sacie potrubie
S018	EGR ventil

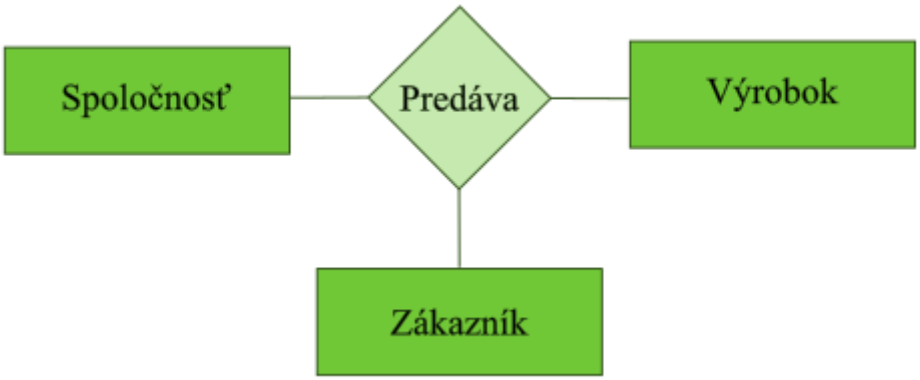
Montuje sa

ID zostavy	Názov zostavy	ID súčiastky	Počet súčiastok
1	Motor Kia R - 2.0 CRDi	S012	1
1	Motor Kia R - 2.0 CRDi	S013	1
1	Motor Kia R - 2.0 CRDi	S014	3
1	Motor Kia R - 2.0 CRDi	S015	4
1	Motor Kia R - 2.0 CRDi	S018	1
1	Motor Kia R - 2.0 CRDi	S017	1
1	Motor Kia R - 2.0 CRDi	S016	1
1	Motor Kia R - 2.0 CRDi	S001	2
2	Piestový motor	S003	8
2	Piestový motor	S005	6

Obrázok č. 5.7 – Príklad relácie „Montuje sa."

5.2.7 Reprezentácia ternárnych relačných vzťahov

Uvažujeme o vzťahu, v ktorom sa vyskytujú tri entity. Uvádzame na príklade medzi entitami „Spoločnosť," „Tovar," „Zákazník," pričom uvažujeme, že jedna spoločnosť predáva viac (N) výrobkov, ktoré kupuje viacero (M) zákazníkov. Rovnako jeden (1) výrobok predáva viacero (P) spoločností viacerým (M) zákazníkom. A pre zákazníka platí, že jeden (1) zákazník si môže kúpiť viac (N) výrobkov od viacerých (P) spoločností. V tomto prípade hovoríme o kardinalite M : N : P.



Obrázok č. 5.8 – Reprezentácia ternárneho relačného vzťahu M : N : P.

Riešenie

Riešením je vznik novej samostatnej asociatívnej tabuľky „Predáva," ktorá bude obsahovať všetky tri primárne kľúče zo všetkých entít („Spoločnosť," „Výrobok," „Zákazník") ako cudzie kľúče.

Vzniknuté tabuľky

„Spoločnosť" („ID spoločnosti" (PK), „Meno spoločnosti," „Adresa spoločnosti," „Email spoločnosti," „Telefón spoločnosti").

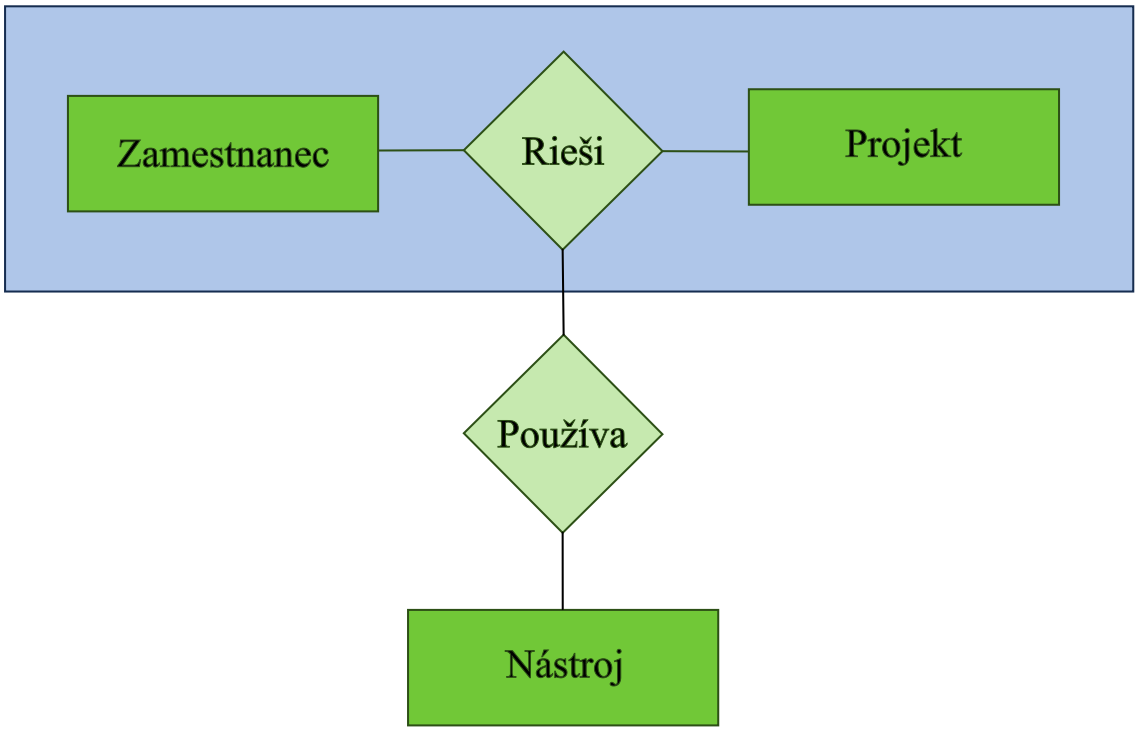
„Výrobok" („ID výrobku" (PK), „Názov výrobku," „Opis výrobku," „Cena výrobku").

„Zákazník" („ID zákazníka" (PK), „Meno zákazníka," „Priezvisko zákazníka," „Adresa zákazníka," „Email zákazníka," „Telefón zákazníka").

„Predáva" („ID spoločnosti" (CK), „ID výrobku" (CK), „ID zákazníka" (CK), „Množstvo výrobkov," „Celková cena").

5.2.8 Reprezentácia agregácie

Uvažujme o databázovom systéme, v ktorom máme entity: „Zamestnanec," „Projekt" a „Nástroj." Pričom platí, že M zamestnancov rieši N projektov, pri ktorých používajú P rôznych nástrojov.



Obrázok č. 5.9 – Reprezentácia agregácie.

Riešenie

Riešením je vznik dvoch asociatívnych tabuliek.

Vzniknuté tabuľky

„Zamestnanec“ („ID zamestnanca“ (PK), „Meno zamestnanca,“ „Priezvisko zamestnanca,“ „Adresa zamestnanca,“ „Telefón zamestnanca,“ „Email zamestnanca“).

„Projekt“ („ID projektu“ (PK), „Meno projektu“).

„Nástroj“ („ID nástroja,“ „Názov nástroja“).

„Rieši“ („ID zamestnanca“ (CK), „ID projektu“ (CK)).

„Používa“ („ID zamestnanca“ (CK), „ID projektu“ (CK), „ID nástroja“ (CK)).

Zhrnutie 4. kapitoly

Transformácia E-R modelu do relačného modelu je úsilím o implementáciu E-R konceptu v relačnej databáze. Aplikáciou tejto transformácie vznikne zoznam tabuliek, ktoré obsahujú (v ideálnom prípade) všetky integritné obmedzenia špecifikované v E-R diagrame [14].

Výsledkom tejto transformácie je vlastne tvorba tabuliek, tak ako budú vyzerať v databázovom systéme. Štruktúru jednotlivých tabuliek vytvoríme z informácií uvedených v E-R diagrame.

Otázky na zopakovanie

1. Vlastnými slovami vysvetlite, čo vykonávame transformáciou E-R modelu do relačného údajového modelu.
2. Čo reprezentuje entity a relácie v relačnom údajovom modeli?
3. Uveďte príklad na reláciu s kardinalitou 1 : N a transformujte ju do relačného modelu.
4. Uveďte príklad na reláciu s kardinalitou M : N a transformujte ju do relačného modelu.
5. Uveďte príklady na povinné a nepovinné členstvo entít v relácii a transformujte tento vzťah do relačného modelu.
6. Uveďte príklad na ternárnu reláciu s kardinalitou M : N : P a transformujte ju do relačného modelu.

Doplnkové materiály na štúdium

Informatika štúdium, 2020. [cit. 2020-05-01]. Dostupné na internete: ⟨https://sk-informatika.studentske.cz/2008/09/sprvca-databzy-administrtor.html⟩.

6 Normalizácia

Normalizácia relačnej databázovej schémy je postupný proces, v rámci ktorého sa pôvodné relácie nahrádzajú novými s jednoduchšou štruktúrou. Normalizácia databázovej schémy predstavuje úpravu vzájomných vzťahov v databáze. Na začiatku návrhu je štruktúra relácií nenormalizovaná, to znamená taká, ako si ju predstavuje používateľ. Externé predstavy používateľa o obsahu databázy je nutné upraviť do stavu použiteľného v relačnom systéme, čo je práve súčasť normalizácie. Proces normalizácie znamená postupné rozloženie pôvodných „nenormalizovaných" tabuliek do sústavy viacerých menších tabuliek tak, aby nedošlo ku strate obsiahnutých údajov v pôvodných tabuľkách [17].

Výsledkom normalizácie je rozumné rozloženie atribútov do tabuliek. Žiadanou vlastnosťou databázy, ktorú možno dostať s pomocou normalizácie je, že všetky atribúty medzi ktorými je funkčná závislosť, sú v jednej relačnej tabuľke. Normalizáciu možno charakterizovať ako proces, ktorý je možné použiť v ľubovoľnom okamihu pri navrhovaní databázy. Výhodou databázy ktorá obsahuje normalizované relačné tabuľky, sú hlavne jednoduchší prístup používateľov k údajom v databáze, lepšie udržiavanie údajov a nižšie pamäťové nároky [15].

Príklad nežiadanej duplicity údajov

Predstavme si tabuľku „Spoločnosť" s atribútmi: „Názov spoločnosti," „Adresa spoločnosti," „Kontakt," „Výrobok." Táto spoločnosť by dodávala 1 000 rôznych výrobkov. Z toho vyplýva, že tabuľka by obsahovala 1 000-krát (1 000 inštancií) rovnaký údaj o názve spoločnosti, rovnakú adresu, rovnaký kontakt na spoločnosť, ale každá z týchto 1 000 inštancií (každý riadok tabuľky) by obsahoval inú hodnotu v atribúte „Výrobok." Toto samozrejme nechceme. Okrem zvýšenej náročnosti na úložný priestor pamäte databázy vznikajú aj problémy s anomáliami.

Úprava databázy do príslušnej normálnej formy umožňuje eliminovať jej anomálie. Odstránenie anomálií v databáze zabezpečí jej konzistentnosť, efektívnosť vyhľadávania, a tiež minimalizáciu redundancie údajov [17].

Medzi základné anomálie databázy patria:

- **Anomália vkladania nových údajov** – situácia, keď do (logickej) relácie nie je možné vložiť nový záznam z dôvodu umelej závislosti od inej relácie.

  Uvedieme na príklade povinného členstva, kedy entita „Študent" má povinné členstvo v relácii s entitou „Študijná skupina." To znamená, že každý študent musí byť povinne zaradený do konkrétnej študijnej skupiny. Ak potrebujeme vložiť do databázy údaje o novom študentovi, nemôžeme to vykonať až do chvíle, kým nie je vytvorená študijná skupina, do ktorej ho priradíme.

- **Anomália aktualizácie údajov** – ak pri aktualizácii jednej údajovej hodnoty je nutné súčasne aktualizovať niekoľko údajových záznamov.

  Uvažujeme o situácii, kedy by v školskom systéme neexistovala entita „Predmet," v ktorej uchovávame informácie o všetkých predmetoch. Jednotlivé predmety by sme vkladali do tabuľky „Študent," kde by sme vytvorili atribúty: „Predmet 1," „Predmet 2"… „Predmet N."

  Teraz si predstavme situáciu, kedy by sme chceli aktualizovať názov predmetu *databázové systémy 1* napríklad na názov: *tvorba databázových systémov.* V tomto príklade by sme museli každému študentovi v systéme prepísať tento nový názov predmetu, čo je samozrejme veľmi nepraktické, zdĺhavé a najmä vedúce k vzniku chybných údajov.

V situácii kedy máme vytvorené entity „Študent“ aj „Predmet,“ tento nový názov predmetu aktualizujeme iba v entite (tabuľke) „Predmet,“ v ktorej sa informácie o predmete *databázové systémy 1* nachádzajú iba raz. Tento spôsob uľahčí prácu a nevedie k tvorbe chybných údajov v databáze.

- **Anomália odstraňovania údajov** – ak odstránenie údajov spôsobí nežiadúcu a nenávratnú stratu aj iných dôležitých údajov.

  Uvažujeme o príklade kedy máme tabuľku „Dodávateľ,“ ktorá má atribúty: „ID dodávateľa,“ „Názov dodávateľa,“ „Adresa dodávateľa,“ „Kontakt dodávateľa,“ „Výrobok.“ V databáze by sa nachádzal konkrétny dodávateľ, ktorý by do našej spoločnosti dodával nejaké výrobky. Naša spoločnosť by sa rozhodla, že výrobky od tohto dodávateľa ďalej nebude potrebovať. Preto by sme z databázy vymazali všetky výrobky, ktoré tento dodávateľ dodáva. S týmito výrobkami by sme ale vymazali aj údaje o názve tohto dodávateľa, adresu a kontakt na tohto dodávateľa. V našej databáze by sa viac tento dodávateľ nenachádzal. Predstavme si situáciu, že by sme sa po určitom období rozhodli znova osloviť tohto dodávateľa o spoluprácu. V databáze už ale nemáme žiadny údaj o tejto spoločnosti, pretože o tieto údaje sme prišli pri mazaní výrobkov, ktoré tento dodávateľ dodával.

Riešením je dekompozícia tejto tabuľky do dvoch tabuliek:

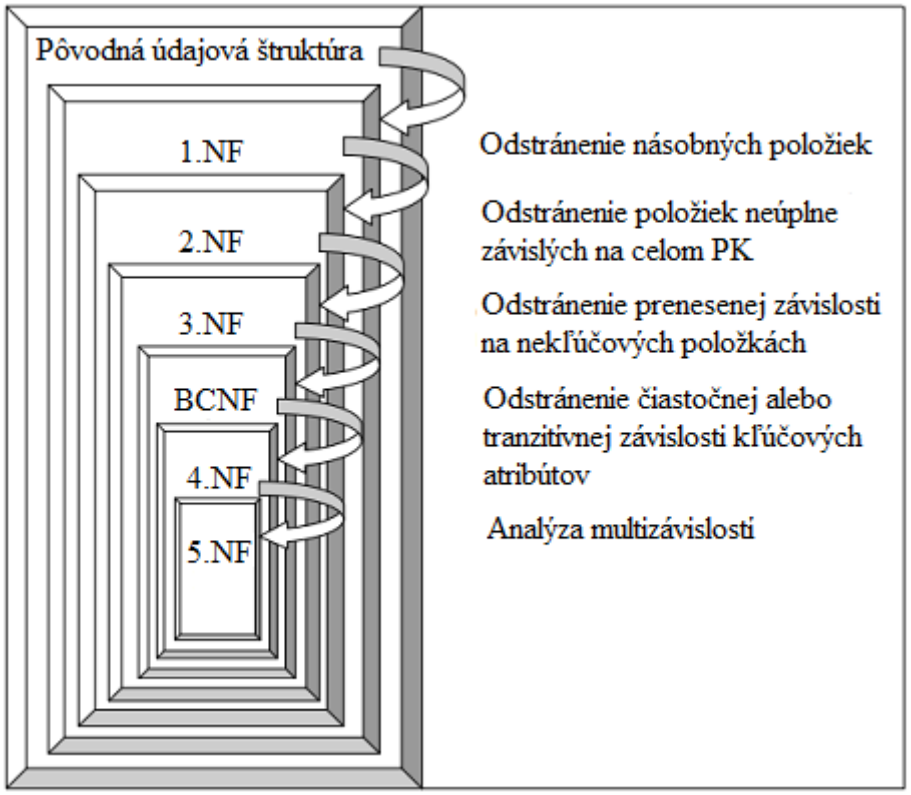
- Dodávateľ – kde by sa nachádzali údaje o spoločnostiach, ktoré dodávajú výrobky.
- Výrobok – kde by sa nachádzali údaje o jednotlivých výrobkoch.

Existuje niekoľko spôsobov, ktoré vedú k lepšiemu návrhu:

1. Nepotrebné údaje by mali byť v oddelených tabuľkách.
2. Údaje, ktoré je možné vypočítať, nie je nutné ukladať (napr. máme čísla A a B, ktoré sú uložené a zaujíma nás ich súčin A × B, potom tento súčin nie je potrebné ukladať).
3. Návrh musí odpovedať všetkým podmienkam, ktoré boli definované pri analýze. Je jednoduché pri návrhu si nevšimnúť nejakú podmienku. Používateľ ľahko objaví pri konzultácii v E-R diagrame chybnú podmienku, ale nie vždy objaví nejakú chýbajúcu.
4. Pri voľbe atribútov je dôležité dbať na ich jasné názvy. Je vhodné sa vyhýbať aj rovnakým názvom pre odlišné pole, napríklad pole primárneho kľúča nazvané ID je vhodné doplniť o konkrétny predmet tabuľky („ID zamestnanca,“ „ID zákazníka“…).
5. Na jednej strane nie je vhodné vytvárať príliš veľa komplikovaných vzťahov – napríklad študent patrí do nejakej študijnej skupiny, tá patrí k nejakej katedre a tá je priradená na určitú fakultu. Na druhej strane, je potrebné pokryť všetky vzťahy tak, aby bolo možné dohľadať všetky spojitosti.
6. Je nutné rozdeliť vzťahy kardinality M : N na dva vzťahy 1 : N a N : 1 prostredníctvom fiktívnej novovytvorenej tabuľky práve na tento vzťah.
7. Je vhodné zamyslieť sa nad obmedzením a teda nad tvorbou limitov, ktoré vyplývajú z použitia aplikácie, napríklad keď vek študenta na strednej škole nemôže presiahnuť 25 rokov, že pohlavie je len mužské alebo ženské, alebo že e-mailová adresa musí obsahovať zavináč. Zabránime tým neskorším problémom pri implementácii.
8. Nie je vhodné ukladať príliš veľa údajov, ktoré nesúvisia s cieľom a účelom aplikácie, pretože to môže v konečnom dôsledku viesť k obťažovaniu koncového používateľa aj osoby, ktorá údaje poskytuje. Napríklad zhromažďovaním informácií o osobách s cieľom registrácie na odber elektronických správ, potom nie je dôležité poznať a ukladať farbu očí alebo obľúbené jedlo (ak nejde

o informačný portál zameraný na varenie). Je vhodné zvážiť aj náročnosť a čas potrebný na spracovanie všetkých takých údajov a to aj v súvislosti s následnou rýchlosťou databázy.

9. Je potrebné dodržať prvé tri normálové formy tak, že v každom poli (atribútu) sú uložené iba atomické hodnoty, z ktorých tie nekľúčové budú vždy úplne závislé od celého primárneho kľúča a v najlepšom prípade nebudú vznikať tranzitívne závislosti.
10. Na tvorbu primárneho kľúča je vhodné vytvoriť nové pole, ako ho vytvárať z kombinácie už existujúcich polí. Nie vždy bude platiť, že táto kombinácia musí byť jedinečná počas celej životnosti databázy.
11. Pri zaisťovaní referenčnej integrity je potrebné dbať na to, aby cudzí kľúč nemal vzťah k primárnemu kľúču z inej tabuľky, ktorá už neexistuje.
12. Je vhodné mať na pamäti dostatočné zabezpečenie databázy a obmedzenie prístupových práv pre jednotlivých používateľov. Nie je možné sprístupniť prehliadanie údajov nepovolaným používateľom aj v prípadoch, kedy ich nemôžu zmeniť [7].



Obrázok č. 6.1 – Postup normalizácie [18].

6.1 Prvá normálová forma (1. NF)

Relácia sa nachádza v prvej normálovej forme (1. NF) ak všetky jej atribúty sú atomické, čiže ďalej nedeliteľné. To znamená, že nesmú byť:

- zložené,
- viachodnotové,
- ani kombinácia predchádzajúcich dvoch uvedených prípadov.

Tabuľka č. 6.1 – Príklad „Hodnotenie," zložený atribút „Známka."

„ID študenta"	„ID predmetu"	Známka		
		Písomný test	**Ústna skúška**	**Výsledná známka**
Š_02510	P_01051	A	C	B
Š_10221	P_00956	D	B	C

Riešenie spočíva v dekompozícii tabuľky „Hodnotenie" a vzniku novej tabuľky „Známka," ktorá bude mať atribúty: „ID známky," „Písomný test," „Ústna skúška," „Výsledná známka."

Príklad viachodnotového atribútu

Uvažujme o tabuľke „Budova školy," ktorá obsahuje atribúty:

„Budova školy" („ID budovy" (PK), „Názov budovy," „Adresa," „Učebňa").

Atribút „Učebňa" obsahuje viacnásobné hodnoty (všetky miestnosti v danej budove), preto táto tabuľka nie je v 1. NF.

Riešením je vytvorenie novej tabuľky „Učebňa," ktorá bude vyzerať takto:

„Učebňa" („ID učebne" (PK), „Výučbové zameranie," „Poschodie," „Kapacita").

Medzi tabuľkami „Budova školy" a „Učebňa" vznikne relácia 1 : N, kedy sa v jednej školskej budove môže nachádzať viac učební. Teraz sú obidve tabuľky v 1. NF a všetky atribúty obsahujú iba jednu hodnotu vo svojom políčku v tabuľke.

6.2 Druhá normálová forma (2. NF)

Relácie je v 2. NF, ak spĺňa podmienky 1. NF a každý jej atribút, ktorý nie je primárnym kľúčom je úplne závislý od každého PK (t. j. sú úplne funkčne závislé od celého zloženého kľúča). Už vieme, že primárny kľúč môže byť tvorený z viacerých atribútov. Táto podmienka súvisí iba s tými tabuľkami, ktoré majú viac primárnych kľúčov. Pri tabuľkách s jedným PK je splnená automaticky. Riešenie problému spočíva v dekompozícii tabuľky.

Príklad

Uvažujme o tabuľke „Tovar," ktorá má nasledujúce atribúty:

„Tovar" („ID tovaru" (PK), „Názov tovaru," „Cena," „ID dodávateľa" (PK), „Názov dodávateľa," „Adresa dodávateľa," „Telefón dodávateľa").

Tabuľka nie je v 2. NF.

Napríklad atribút „Meno dodávateľa" nie je závislý od primárneho kľúča „ID tovaru," ale je závislý iba od primárneho kľúča „ID dodávateľa."

Tu je problém pri modifikovaní údajov. Predstavme si, že sa zmení adresa dodávateľa. V tej chvíli musíme nájsť všetky výskyty konkrétnej adresy pri každom druhu tovaru, ktorý dotknutý dodávateľ

dodáva a všetky prepísať na novú adresu. Pri obrovských databázach, pri ktorých jeden dodávateľ dodáva stovky, tisícky tovarov je to problém.

Riešenie spočíva v dekompozícii tabuľky „Tovar“ na dve tabuľky, ktoré budú vyzerať takto:

„Tovar“ („ID tovaru“ (PK), Názov tovaru, Cena).

„Dodávateľ“ („ID dodávateľ“ (PK), Názov dodávateľa, Adresa dodávateľa, Telefón dodávateľa).

Teraz sú všetky nekľúčové atribúty v tabuľkách „Tovar“ a „Dodávateľ“ úplne závislé od PK a teda sú v 2. NF.

6.3 Tretia normálová forma (3. NF)

Relácia je v 3. NF ak je v 2. NF a každý nekľúčový atribút je netranzitívne závislý od primárneho kľúča. To znamená, že každý atribút je buď kľúčovým atribútom, alebo je svojou hodnotou jednoznačne závislý od celého primárneho kľúča alebo všetkých primárnych kľúčov v dotknutej relácii. 3. NF takisto hovorí, že všetky nekľúčové atribúty musia byť vzájomne nezávislé.

Príklad

Uvažujme o tabuľke „Predmet,“ ktorá má nasledujúce atribúty:

„Predmet“ („ID predmetu“ (PK), „Semester,“ „ID učiteľa“ (PK), „Meno učiteľa“).

Učiteľ sa v tabuľke jednoznačne identifikuje zloženým kľúčom „ID predmetu a Semester,“ kedy vieme, že ten predmet, učil v určitom semestri konkrétny učiteľ. V tomto prípade atribút „Meno učiteľa“ závisí od „ID učiteľa,“ ale nezávisí priamo od zloženého kľúča „ID predmetu a Semester.“

Takáto tranzitívna závislosť sa vyrieši rozdelením tabuľky:

„Predmet“ („ID predmetu“ (PK), „Semester,“ „ID učiteľa“ (CK)).

„Učiteľ“ („ID učiteľa“ (PK), „Meno učiteľa“).

Keby by sme vzali do úvahy, že v tabuľke „Predmet,“ kde by sme nechali atribút „Meno učiteľa,“ by vznikol preklep v mene učiteľa, napríklad meno „Miroslav“ by bolo zaznamenané ako „Miloslav.“ V databáze by vznikla nejednoznačnosť kedy by nikto nevedel určiť ako má meno správne znieť. Časť informácií by bola správne a časť nesprávne. Po odstránení netranzitívnej závislosti sa situácia zmení v tom zmysle, že meno je evidované len jediný raz. Ak by aj vznikol preklep, jeho zmena sa dá uskutočniť v jedinej tabuľke „Učiteľ“ pre všetky ostatné súvislosti v databáze s týmto menom.

6.4 Boyce-Coddova normálová forma (BCNF)

Pôvodný návrh E. F. Codda obsahoval iba tri normálne formy: 1. NF, 2. NF a 3. NF. Neskoršie bola navrhnutá prísnejšia definícia 3. NF, nazývaná Boyce-Coddova normálna forma (BCNF).

Táto normálová forma pôvodne vznikla ako nová a vylepšená 3. NF. Problémom bolo, že bola striktne silnejšia ako pôvodná. Preto bola prijatá ako nová normálová forma. Keďže však v čase vyslovenia tejto definície už existovala 4. NF, nechali jej meno po autoroch (BCNF sa niekde zvykne uvádzať aj ako **3,5. NF**).

Hovorí o tom, že relácie je v BCNF, ak všetky determinanty v tabuľke sú kandidátom na kľúč. Determinant je atribút alebo skupina atribútov, na ktorých sú ostatné atribúty plne funkčne závislé. Determinant, ktorý nie je kandidátom na kľúč a ktorý tiež nie je časťou primárneho kľúča, musí byť oddelený. To znamená, že sa vytvorí nová relácia, ktorej kľúč bude tento determinant.

Každá tabuľka, ktorá je v BCNF je zároveň aj v 3. NF. To však neplatí naopak. Relácia, ktorá je v 3. NF nemusí byť okamžite aj v BCNF, keď platí:

- V relácii existuje viac kandidátov na kľúč.
- Všetky kandidátne kľúče sú zložené z dvoch alebo viacerých atribútov.
- Existuje taký atribút, ktorý je spoločný pre všetky kandidátne kľúče.

Príklad

BCNF uvádzame na príklade tabuľky „Adresa.“ Táto tabuľka obsahuje nasledujúce atribúty:

„Adresa“ („ID adresa“ (PK), „Mesto,“ „Ulica,“ „Orientačné číslo,“ „PSČ“).

Tabuľka spĺňa 1. NF, pretože jej atribúty obsahujú iba atomické hodnoty. Tabuľka takisto spĺňa 2. NF, pretože každý atribút, ktorý nie je PK, je úplne závislý od PK.

Problém vzniká v 3. NF, pretože z atribútu „Ulica“ sa nedá jasne určiť, v akom meste sa nachádza. Keby sme pri atribúte „Ulica“ zaznamenávali aj GPS súradnice, tak by sme aj pre rovnaké názvy ulíc z rôznych miest dokázali jednoznačne určiť, do akého mesta patria. Zostaňme v tomto prípade pri tom, že názvy ulíc neobsahujú GPS súradnice a nie sú teda jedinečné. To znamená, že z ich názvu nevieme jasne určiť, do akého mesta patria. Z tohto hľadiska 3. NF zatiaľ neporušujeme [22].

Uvažujme však o atribúte „PSČ.“ Ak si vezmeme atribút („Mesto,“ + „Ulica“) vieme následne určiť ich PSČ. Vzniká tu teda závislosť: („Mesto,“ „Ulica“) → „PSČ.“

Tu sa môže zdať, že porušujeme 3. NF, pretože „PSČ“ nie je súčasťou PK a je závislý od atribútov, ktoré nie sú PK. Lenže 3. NF hovorí, že všetky nekľúčové atribúty musia byť vzájomne nezávislé. „PSČ“ ale v tomto prípade je kľúčový atribút, je totiž časťou kandidátneho kľúča. **Kandidátny kľúč** je atribút alebo skupina atribútov, ktoré by mohli slúžiť ako PK [22].

Tabuľka „Adresa“ má tieto kandidátne kľúče:

- **(„Mesto,“ „Ulica,“ „Orientačné číslo“)** → („PSČ“).
- **(„PSČ,“ „Ulica,“ „Orientačné číslo“)** → („Mesto“).

Z týchto kandidátnych kľúčov potom ľubovoľne jeden vyberieme, ktorý bude slúžiť ako PK. Tieto kandidátne kľúče budú v tabuľke „Adresa“ vždy jedinečné (v tomto prípade zabudnime na atribút „ID adresa,“ ten akoby sa v tabuľke nenachádzal). Všetky atribúty, ktoré sú súčasťou niektorého z kandidátnych kľúčov sú kľúčové atribúty a teda nie je možné, aby porušili podmienky pre 3. NF [22].

Tu nastáva fáza BCNF, ktorá je prísnejšou verziou 3. NF. Rozdiel medzi BCNF a 3. NF je natoľko formálny, že ľudia si často myslia, že tabuľka porušuje 3. NF a pritom v skutočnosti porušuje BCNF [22].

Determinanty funkčnej závislosti v tabuľke „Adresa“ sú:

- **(„Mesto,“ „Ulica,“ „Orientačné číslo“)** → („PSČ“).
- **(„PSČ,“ „Ulica,“ „Orientačné číslo“)** → („Mesto“).

- **(„Mesto,“ „Ulica“)** → („PSČ“).
- **(„PSČ“)** → („Mesto“).

To, čo sa nachádza na ľavej strane jednoznačne určuje hodnotu atribútu na pravej strane. Napríklad ak si zoberieme prvý riadok, podľa názvu mesta, ulice a orientačného čísla vieme jednoznačne odvodiť aj hodnotu atribútu „PSČ.“

Ak si zoberieme príklad posledných dvoch závislostí (posledné dva príklady) porušujú BCNF, pretože determinanty nie sú kandidátnym kľúčom. Tieto dvojice nie sú pre tabuľku „Adresa“ jednoznačné. Nejde však pritom o porušenie 3. NF, pretože „PSČ“ a „Mesto“ nie sú nekľúčové atribúty [22].

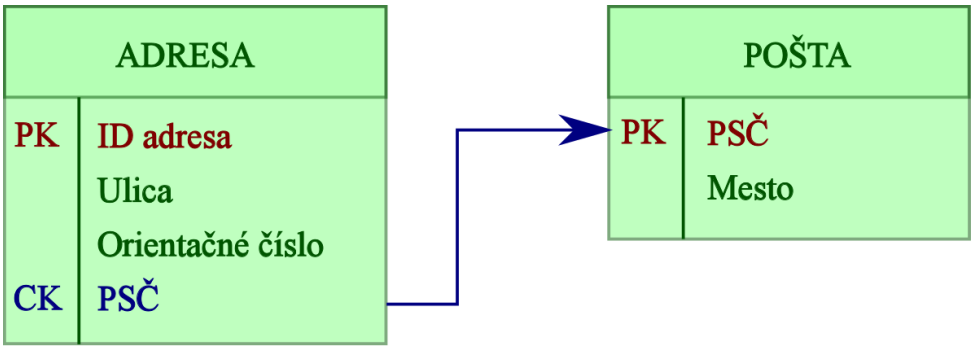
Riešení je vznik novej tabuľky, ktorú si nazveme napríklad „Pošta.“ Tabuľka „Pošta“ bude obsahovať závislosť (PSČ) → (mesto), v tabuľke „Adresa“ zostávajú ostatné atribúty.

Dekompozíciou vzniknú nasledujúce tabuľky:

„Adresa“ („ID adresa“ (PK), „Ulica,“ „Orientačné číslo,“ „PSČ“).

„Pošta“ („PSČ“ (PK), „Mesto“).

Medzi tabuľkami vzniká nová relácia s kardinalitou 1 : N. Kedy jedna inštancia z tabuľky „Pošta“ je vo vzťahu s viacerými rôznymi adresami, ale jedna adresa patrí iba jednej inštancii z tabuľky „Pošta.“ Je teda zrejmé, že hodnoty: „Ulica,“ „Orientačné číslo“ a „PSČ“ môžu patriť iba jednému mestu [22].

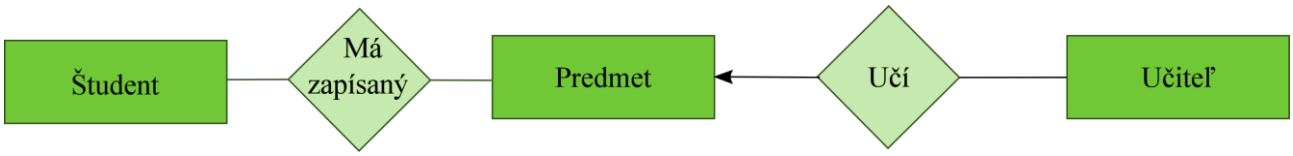


Obrázok č. 6.2 – Príklad dekompozície tabuľky na príklade BCNF.

Teraz sa už nemôže stať anomália, kedy by sme mali rovnaké „PSČ“ pre rôzne mestá.

Príklad

Uvažujeme o situácii zo školského prostredia kedy jeden predmet môže učiť viacero učiteľov, ale každý učiteľ môže učiť iba jeden predmet. Každý študent má zapísaných viac predmetov, ale na každý predmet má prideleného iba jedného učiteľa.



Obrázok č. 6.3 – Príklad na BCNF.

Majme reláciu „Má zapísaný,“ ktorej tabuľka vyzerá takto:

„Má zapísaný“ („ID študenta“ (PK), „Meno učiteľa,“ „Názov predmetu“).

Táto tabuľka nespĺňa ani podmienky pre 2. NF, pretože atribút „Názov predmetu“ nie je závislý od primárneho kľúča, ale je závislý od atribútu „Meno učiteľa.“

„Má zapísaný“ („ID študenta“ (PK), „Názov predmetu,“ „Meno učiteľa“).

Tabuľka je v 3. NF, ale nie je v BCNF. „Názov predmetu“ je závislý od atribútu „Meno učiteľa“ a ten nie je kandidát na kľúč. Takto nemôžeme vložiť fakt, že niektorý učiteľ učí určitý predmet, keď si ho nezapíše aspoň jeden študent.

Riešenie spočíva vo vytvorení novej relácie s názvom „Trieda“ a dekompozíciou relácie „Má zapísaný.“

Tabuľka bude potom vyzerať:

„Trieda“ („ID študenta“ (PK), „Meno učiteľa“).

Tabuľka „Trieda“ sa nachádza v BCNF.

Ďalšie dve normálové formy sa často v praxi nepoužívajú preto je len uvedená ich stručná charakteristika.

6.5 Štvrtá normálová forma (4. NF)

Tabuľka je v štvrtej normálovej forme, ak je v BCNF a stĺpce v nej opisujú len jeden fakt alebo jednu súvislosť [19].

6.6 Piata normálová forma (5. NF)

Posledná, piata, normálová forma hovorí, že tabuľka je v 5. NF, keď je vo 4. NF a nie je možné pridať ďalší atribút do tabuľky tak, aby sa tabuľka nerozpadla na ďalšie tabuľky z dôvodu skrytých závislostí [19].

Zhrnutie 6. kapitoly

Normalizácia je proces, s pomocou ktorého rozkladáme relácie s cieľom jednoduchšej manipulácie s údajmi, zabránenia redundancie údajov a lepšej konzistencie. Celý proces prebieha aplikovaním jednotlivých pravidiel, ktoré nazývame normálové formy [19].

1. NF

Význam prvej normálovej formy je, že každý atribút obsahuje len **atomické hodnoty,** ktoré sa nedajú ďalej deliť [19].

2. NF

Aby relačná databáza bola v druhej normálovej forme, musí byť v 1. NF a každý nekľúčový atribút musí byť závislý od celého primárneho kľúča a nielen od jeho podmnožiny [19].

3. NF

Aj pre tretiu normálovú formu platí, že musí byť v druhej normálovej forme a zároveň musí platiť, že žiadny z jej atribútov nie je tranzitívne závislý od kľúča [19].

BCNF

Hovorí, že relácia je v BCNF, keď pre každú netriviálnu funkčnú závislosť X → Y platí, že X je nadmnožinou nejakého kľúča [20].

Otázky na zopakovanie

1. Vysvetlite, čo je úlohou normalizácie.
2. Aké anomálie dokážeme správnou normalizáciou odstrániť? Uveďte na jednotlivé anomálie príklad.
3. Aké poznáme normálové formy? Vysvetlite, podľa čoho upravujeme tabuľky podľa jednotlivých NF. Uveďte príklady.

Doplnkové materiály na štúdium

Brehovský, M.: Podpora navrhovania relačnej schémy pre databázu, Univerzita Karlova v Praze, 2006. [cit. 2020-05-01]. Dostupné na internete: ⟨https://is.cuni.cz/webapps/zzp/download/130015827⟩.

Vlčák, M.: Normalizácia relačných databáz, Univerzita Komenského, Bratislava, Fakulta matematiky, fyziky a informatiky, Katedra informatiky, Bratislava, 2009. [cit. 2020-05-01]. Dostupné na internete: ⟨http://www.dcs.fmph.uniba.sk/bakalarky/obhajene/getfile.php/main.pdf?id=124&fid=235&type=application%2Fpdf⟩.

Koščák, J.: Preklad štruktúry relačných databáz do štruktúry grafových databáz, Masarykova univerzita Fakulta informatiky, [cit. 2020-04-02]. Dostupné na internete: ⟨https://is.muni.cz/th/xfe94/diplomka.pdf⟩.

Normalization – 1NF, 2NF, 3NF, and 4NF. YouTube, channel5567, 14. 8. 2015. [cit. 2020-05-01]. Dostupné na internete: ⟨https://www.youtube.com/watch?v=UrYLYV7WSHM⟩.

4. lekce: Normalizace relačních databází. YouTube, Distančně.cz, 6. 10. 2016. [cit. 2020-05-01]. Dostupné na internete: ⟨https://www.youtube.com/watch?v=AaHzlgHLS6g⟩.

7 Návrh školského databázového systému

Poslednou kapitolou v tejto učebnici je konkrétny príklad, ako by mal vyzerať návrh databázového systému. Dnes už každá škola využíva svoj vlastný informačný systém. Preto sme sa rozhodli vypracovať návrh jednoduchého školského databázového systéme, pretože zrejme každý čitateľ sa už určite s takýmto systémom stretol. Samozrejme náš príklad demonštruje iba časť databázového systému.

7.1 Analýza požiadaviek na systém

V databázovom systéme chceme evidovať informácie o študentoch, predmetoch, učiteľoch, študijných skupinách a katedrách. Vieme, že každý študent bude mať zapísaných viac predmetov a jeden predmet bude študovať viacero študentov. Každý študent bude priradený do jednej študijnej skupiny, pričom do tejto skupiny môže byť priradených viacero študentov. Každá študijná skupina bude patriť jednej katedre, pričom jedna katedra môže obsahovať aj viac študijných skupín. Každý učiteľ bude zamestnaný iba na jednej katedre, a na jednej katedre môže byť zamestnaných aj viacero učiteľov. Jeden učiteľ môže vyučovať viac predmetov, a jeden predmet môže vyučovať viacero učiteľov.

Pri jednotlivých objektoch databázy chceme uchovávať tieto údaje:

- **Študent:** meno, priezvisko, adresa, e-mail.
- **Predmet:** názov, opis, kredity.
- **Učiteľ:** meno, priezvisko, adresa, e-mail.
- **Študijná skupina:** meno, akademický rok.
- **Katedra:** názov, adresa, e-mail.

7.2 Údajová analýza

V tejto časti projektu si vykonáme analýzu údajov, ktoré budeme v systéme uchovávať.

7.2.1 Identifikácia entít a ich atribútov

„Študent“ je entita, v ktorej budeme uchovávať informácie o študentoch. Táto entita bude mať nasledujúce atribúty:

- **„ID študenta“** – kľúčový atribút entity (PK). Bude slúžiť na rozlíšenie jednotlivých inštancií v určenej entite. Formát atribútu bude automatické číslo.
- **„Meno študenta“** – údaj o mene študenta. Formát: textový reťazec s dĺžkou maximálne 20 znakov.
- **„Priezvisko študenta“** – údaj o priezvisku študenta. Formát: textový reťazec s dĺžkou maximálne 25 znakov.
- **„Adresa študenta“** – údaj o trvalom bydlisku študenta. Formát: textový reťazec s dĺžkou maximálne 100 znakov.

- **„Email študenta“** – údaj o elektronickej adrese študenta. Formát: textový reťazec s dĺžkou maximálne 30 znakov.

„Predmet“ je entita, v ktorej budeme uchovávať informácie o študijných predmetoch. Táto entita bude mať nasledujúce atribúty:

- **„ID predmetu“** – kľúčový atribút entity (PK). Bude slúžiť na rozlíšenie jednotlivých inštancií v tejto entite. Formát atribútu bude automatické číslo.
- **„Názov predmetu“** – údaj o názve predmetu. Formát: textový reťazec s dĺžkou 50 znakov.
- **„Opis predmetu“** – informácie o predmete, jeho charakteristika, cieľ. Formát: textový reťazec s dĺžkou 1500 znakov.
- **„Kredity“** – údaj o počte kreditov za konkrétny predmet. Formát: celočíselný typ.

„Učiteľ“ je entita, v ktorej budeme uchovávať informácie o učiteľoch. Táto entita bude mať nasledujúce atribúty:

- **„ID učiteľa“** – kľúčový atribút entity (PK). Bude slúžiť na rozlíšenie jednotlivých inštancií v tejto entite. Formát atribútu bude automatické číslo.
- **„Meno učiteľa“** – údaj o mene učiteľa. Formát: textový reťazec s dĺžkou maximálne 20 znakov.
- **„Priezvisko učiteľa“** – údaj o priezvisku učiteľa. Formát: textový reťazec s dĺžkou maximálne 25 znakov.
- **„Adresa učiteľa“** – údaj o bydlisku učiteľa. Formát: textový reťazec s dĺžkou maximálne 100 znakov.
- **„Email učiteľa“** – údaj o elektronickej adrese učiteľa. Formát: textový reťazec s dĺžkou maximálne 30 znakov.

„Študijná skupina“ je entita, v ktorej budeme uchovávať informácie o jednotlivých študijných skupinách na škole. Táto entita bude mať nasledujúce atribúty:

- **„ID skupiny“** – kľúčový atribút entity (PK). Bude slúžiť na rozlíšenie jednotlivých inštancií v tejto entite. Formát atribútu bude automatické číslo.
- **„Meno skupiny“** – názov študijnej skupiny. Formát: textový reťazec s dĺžkou maximálne 25 znakov.
- **„Akademický rok“** – údaj o akademickom roku. Formát: znakový reťazec, ktoré dĺžka bude presne 9 znakov (RRRR/RRRR).

„Katedra“ je entita, v ktorej budeme uchovávať údaje o katedrách na škole. Táto entita bude mať nasledujúce atribúty:

- **„ID katedry“** – kľúčový atribút entity (PK). Bude slúžiť na rozlíšenie jednotlivých inštancií v tejto entite. Formát atribútu bude automatické číslo.
- **„Názov katedry“** – údaj o názve katedry. Formát: textový reťazec s dĺžkou 70 znakov.
- **„Adresa katedry“** – údaj o adrese kde sídli katedra. Formát: textový reťazec s dĺžkou maximálne 100 znakov.

- **„Email katedry“** – údaj o elektronickej adrese na sekretariát katedry. Formát: textový reťazec s dĺžkou maximálne 30 znakov.

7.2.2 Definícia relácií (vzťahov) medzi entitami

„Má zapísaný“ relácia medzi entitami **„Študent“** a **„Predmet.“**

- Priraďuje predmet k študentovi. Platí, že jeden predmet môže mať zapísaných viacero študentov a jeden študent môže mať zapísaných viac predmetov.
- Relačný vzťah má kardinalitu M : N.
- Entita „Predmet“ je nepovinným členom v tejto relácii.
- Relácia nemá vlastný atribút (môžeme dopracovať atribút „Akademický rok“).

„Vyučuje“ relácia medzi entitami **„Učiteľ“** a **„Predmet.“**

- Priraďuje učiteľa k predmetu, pričom jeden učiteľ môže vyučovať viacero predmetov, a jeden predmet môže vyučovať viacero učiteľov.
- Relačný vzťah má kardinalitu M : N.
- Entita „Predmet“ je nepovinným členom v tejto relácii.
- Relácia nemá vlastný atribút.

„Priradený“ relácia medzi entitami **„Študent“** a **„Študijná skupina.“**

- Relácia priraďuje študenta do študijnej skupiny, pričom jeden študent môže patriť iba do jednej študijnej skupiny, ale do jednej študijnej skupiny môže patriť viacero študentov.
- Relačný vzťah má kardinalitu 1 : N.
- Entita „Študent“ je povinným členom vzťahu „Priradený.“

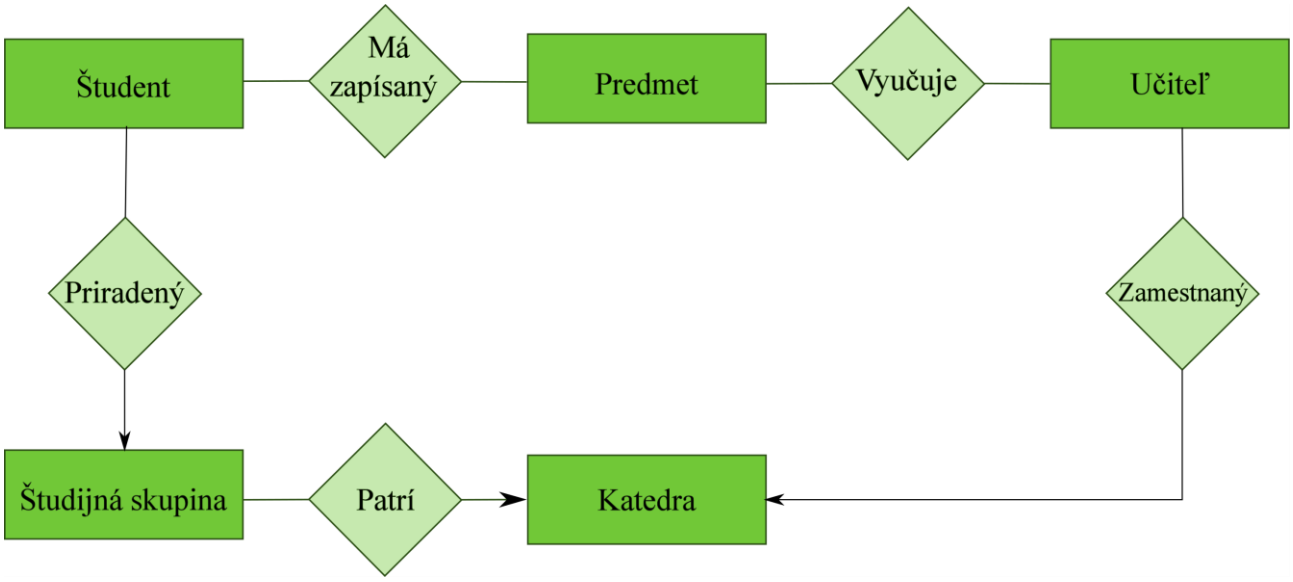
„Zamestnaný“ relácia medzi entitami **„Učiteľ“** a **„Katedra.“**

- Relácia priraďuje učiteľa ku konkrétnej katedre, pričom platí, že jeden učiteľ môže byť zamestnaný iba na jednej katedre, ale na jednej katedre môže byť zamestnaných viacero učiteľov.
- Relačný vzťah má kardinalitu 1 : N.
- Entita „Učiteľ“ je povinným členom vzťahu „Priradený.“

„Patrí“ relácia medzi entitami **„Študijná skupina“** a **„Katedra.“**

- Relácia priraďuje študijnú skupinu ku konkrétnej katedre, pričom platí, že jedna študijná skupina patrí iba do jednej katedry, ale na jednej katedre môže existovať viacero študijných skupín.
- Relačný vzťah má kardinalitu 1 : N.
- Entita „Študijná skupina“ je povinným členom v relácii **„Patrí.“**

7.2.3 Entitno-relačný model



Obrázok č. 7.1 – E-R diagram školského DBS.

7.3 Transformácia E-R diagramu do relačných schém

Transformuje jednotlivé entity a relácie z nášho modelu do relačných schém (tabuliek) do formátu, ako budú vyzerať v reálnom systéme. Postup je vysvetlený v kapitole číslo 5 tejto učebnice.

7.3.1 Transformácia entít

„Študent“: („ID študenta“ (PK), „Meno študenta,“ „Priezvisko študenta,“ „Adresa študenta,“ „Email študenta,“ „ID skupiny“ (CK)).

„Predmet“: („ID predmetu“ (PK), „Názov predmetu,“ „Opis predmetu,“ „Kredity“).

„Učiteľ“: („ID učiteľa“ (PK), „Meno učiteľa,“ „Priezvisko učiteľa,“ „Adresa učiteľa,“ „Email učiteľa,“ „ID katedry“ (CK)).

„Študijná skupina“: („ID skupiny“ (PK), „Meno skupiny,“ „Akademický rok,“ „ID katedry“ (CK)).

„Katedra“: („ID katedry“ (PK), „Názov katedry,“ „Adresa katedry,“ „Email katedry“).

7.3.2 Transformácia relácií

„Má zapísaný“: („ID študenta“ (CK), „ID predmetu“ (CK)).

„Vyučuje“: („ID učiteľa“ (CK), „ID predmetu“ (CK)).

„Priradený“: relácia je realizovaná prostredníctvom cudzieho kľúča „ID skupiny“ do relačnej schémy (tabuľky) **„Študent.“**

„Zamestnaný“: relácia je realizovaná prostredníctvom cudzieho kľúča „ID katedry“ do relačnej schémy **„Učiteľ.“**

„Patrí“: relácia je realizovaná prostredníctvom cudzieho kľúča „ID katedry“ do relačnej schémy **„Študijná skupina.“**

7.4 Normalizácia relačných schém

Všetky relačné schémy (tabuľky), ktoré sa nachádzajú v predchádzajúcej kapitole číslo 7.3 sa nachádzajú minimálne v 3. NF, takže vyhovujú požadovaným kritériám.

Ako môžeme vidieť, každý atribút obsahuje len **atomické hodnoty,** preto spĺňa podmienku pre 1. NF. Samozrejme to neplatí o atribútoch „Adresa“ v entitách „Študent,“ „Učiteľ“ a „Katedra.“ Tu by sa žiadalo rozdeliť tento atribút na viaceré atribúty: „Mesto,“ „Ulica,“ „Orientačné číslo,“ „PSČ.“ Tu ale nastáva ďalší problém medzi závislosťou atribútov „Mesto“ a „PSČ.“ Tento problém je bližšie vysvetlený v kapitole 6.4. Vznikla by potreba vytvorenia novej entity, čím by sa systém rozšíril. Z tohto dôvodu sme tento atribút ponechali v pôvodnom stave.

Každý nekľúčový atribút v jednotlivých relačných schémach **je závislý od celého primárneho kľúča** a preto tabuľky spĺňajú podmienku pre 2. NF.

Relačné schémy takisto spĺňajú podmienku pre 3. NF a BCNF pričom platí, že žiadny z ich atribútov nie je tranzitívne závislý od primárneho kľúča.

Pri jednoduchých databázových systémoch je jednoduché navrhnúť relačné schémy tak, aby spĺňali jednotlivé normalizačné podmienky. Omnoho ťažšie je to pri zložitých, obrovských informačných systémoch. Postup normalizácie je vysvetlený v kapitole číslo 6 tejto učebnice.

7.5 Návrh logickej schémy

V tejto kapitole zobrazíme všetky tabuľky, ktoré sa budú v databázovom systéme nachádzať.

„Študent“: („ID študenta“ (PK), „Meno študenta,“ „Priezvisko študenta,“ „Adresa študenta,“ „Email študenta,“ „ID skupiny“ (CK)).

„Predmet“: („ID predmetu“ (PK), „Názov predmetu,“ „Opis predmetu,“ „Kredity“).

„Učiteľ“: („ID učiteľa“ (PK), „Meno učiteľa,“ „Priezvisko učiteľa,“ „Adresa učiteľa,“ „Email učiteľa,“ „ID katedry“ (CK)).

„Študijná skupina“: („ID skupiny“ (PK), „Meno skupiny,“ „Akademický rok,“ „ID katedry“ (CK)).

„Katedra“: („ID katedry“ (PK), „Názov katedry,“ „Adresa katedry,“ „Email katedry“).

„Má zapísaný“: („ID študenta“ (CK), „ID predmetu“ (CK)).

„Vyučuje“: („ID učiteľa“ (CK), „ID predmetu“ (CK)).

Ako môžeme vidieť, v logickej schéme chýbajú relácie: **„Priradený,“ „Zamestnaný“ a „Patrí.“** Tie nebudú vyjadrené prostredníctvom tabuliek, ale prostredníctvom transformácie cudzieho kľúča.

Úlohy na rozšírenie školského databázového systému

1. Doplňte do systému novú entitu, v ktorej budeme uchovávať údaje o jednotlivých učebniach nachádzajúcich sa v budove školy. Požiadavkou je, aby táto entita v sebe obsahovala informácie o type

učebne (jazyková, počítačová, fyzikálna, laboratórna a podobne), ďalej o tom kde sa učebňa nachádza (pavilón, poschodie prípadne iný spôsob), a ešte aby obsahovala informácie o maximálnej kapacite týchto miestností. Táto entita bude vo vzťahu s entitou „Predmet," pričom platí podmienka, že v jednej učebne môže prebiehať výučba viacerých predmetov a takisto jeden predmet sa môže vyučovať vo viacerých učebniach. Vymyslite vhodný názov pre entitu, reláciu a takisto pre jednotlivé atribúty a dopracujte to do predchádzajúceho návrhu.

2. Vymyslite aj iné entity s reláciami a atribútmi, ktoré by sa hodili do predchádzajúceho návrhu pre školský databázový systém.

3. Upravte návrh školského DBS tak, aby spĺňal požiadavku pre 1. NF, podľa kapitoly 7.4.

Záver

V súčasnosti sa databázové systémy využívajú takmer všade, či už je to na úradoch, vo finančných domoch ako v bankách a poisťovníctve, ďalej v podnikovej sfére ale aj na školách, zdravotníctve a takto by sme mohli pokračovať. Neustále sa zvyšuje objem údajov, ktoré je potrebné požadovaným spôsobom spracovávať, uchovávať, manipulovať s nimi a v neposlednom rade ich chrániť pred odcudzením a znehodnotením od nepovolaných vplyvov. Databázové systémy sa ako vyučovací predmet vyučujú na stredných a vysokých školách. Počas života každý z nás pracuje s nejakou databázou, či už je to len práca s telefónnym zoznamom v našom mobilnom zariadení. Aj tieto už spomínané dôvody boli práve tie, prečo sme sa rozhodli vypracovať tento učebný materiál.

Obsahom predloženej učebnice je konceptuálne modelovanie databázových systémov. Toto modelovanie sa skladá z piatich fáz. Okrem týchto fáz, sme v úvode učebnice vypracovali kapitolu o architektúre databázových systémov. Samotné konceptuálne modelovanie začína analýzou požiadaviek od budúceho používateľa na systém. V tejto fáze je dôležité, aby riešiteľ návrhu budúceho systému správne pochopil to, čo od neho zadávateľ požaduje. Analýza končí vypracovaním katalógu požiadaviek. Tento dokument je akýmsi dôkazom toho, že riešiteľ správne pochopil ako by mal budúci systém fungovať. Po analýze nastáva fáza samotného návrhu. Ten realizujeme prostredníctvom entitno-relačného diagramu. Po vytvorení E-R modelu sme pokračovali teoretickými východiskami o údajovom modeli. Ten sme realizovali prostredníctvom relačného údajového modelu. Piatou kapitolou bola transformácia E-R modelu do relačného údajového modelu. Aplikáciou tejto transformácie vznikne zoznam tabuliek, ktoré obsahujú všetky integritné obmedzenia špecifikované v E-R schéme. Postup transformácie sme ukázali na jednotlivých príkladoch. Poslednou kapitolou modelovania bola normalizácia, s pomocou ktorej pôvodne navrhnuté tabuľky nahrádzame novými s jednoduchšou štruktúrou. Cieľom je najmä odstrániť viacnásobný výskyt rovnakých údajov v databáze. Na záver učebnice sme zaradili siedmu kapitolu, ktorej obsahom je vzorový príklad konkrétneho konceptuálneho návrhu školského databázového systému.

Zoznam použitej a odporúčanej literatúry

[1] Šišák, I.: Nástroj na vizualizaci databázového modelu existující databáze, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2010. [cit. 2020-04-02]. Dostupné na internete: ⟨http://hdl.handle.net/11012/55902⟩.

[2] Encyclopaedia Beliana: Architektúra databázových systémov. ISBN 978-80-89524-30-3. [cit. 2020-04-02]. Dostupné na internete: ⟨https://beliana.sav.sk/heslo/architektura-databazovych--systemov⟩.

[3] Minárik, R.: Generátor skúšobných testov, Žilinská univerzita v Žiline, Elektrotechnická fakulta, Žilina, 2006. [cit. 2020-04-02]. Dostupné na internete: ⟨http://diplom.utc.sk/wan/704.pdf⟩.

[4] Pribilová, K.: Databázové systémy 1. Pedagogická fakulta Trnavskej univerzity v Trnave, Trnava, 2013. ISBN 978-80-8082-680-2. [cit. 2020-04-02]. Dostupné na internete: ⟨http://pdf.truni.sk/e--ucebnice/databazove-systemy1/⟩.

[5] Kasalová, Z.: Analýza informačného systému jazykové školy, Masarykova univerzita v Brně, Fakulta informatiky, Brno, 2006. [cit. 2020-04-02]. Dostupné na internete: ⟨https://is.muni.cz/th/b55si/bakalarska_praca.pdf⟩.

[6] Oppel, A.: Databáze bez předchozích znalostí. Brno : Computer Press, a. s., 2006. ISBN 80-251-1199--7. S. 51.

[7] Otte, L.: Databázové systémy – Životní cyklus databáze. Vysoká škola Baňská, Technická univerzita Ostrava, Fakulta strojní, Ostrava, 2013. [cit. 2020-04-02]. Dostupné na internete: ⟨http://projekty.fs.vsb.cz/463/edubase/VY_01_044/Datab%C3%A1zov%C3%A9%20syst%C3%A9my.pdf⟩.

[8] Chen, P.: The entity-relationship model – toward a unified view of data, Massachusetts Institute of Technology, 1976. [cit. 2020-04-02]. Dostupné na internete: ⟨http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.526.369&rep=rep1&type=pdf⟩.

[9] Relational Database Systems – An Introduction, Chapter 1, Microsoft SQL Server 2008 : A Beginner's Guide, [cit. 2020-04-02]. Dostupné na internete: ⟨http://www.mhprofessional.com/downloads/products/0071546383/0071546383_ch01.pdf⟩.

[10] SZABÓ, P.: Databázové a informačné systémy – Databázová terminológia – relačný dátový model, informatika pre študentov štvrtého – piateho ročníka Leteckej fakulty, TUKE, 2005. [cit. 2020-04-02]. Dostupné na internete: ⟨https://spseke.sk/tutor/prednasky/dbs/rdm.html⟩.

[11] Duračiová, R.: Databázové systémy v GIS, Slovenská technická univerzita v Bratislave, 2014, ISBN 978-80-227-4292-4, [cit. 2020-04-02]. Dostupné na internete: ⟨https://www.researchgate.net/profile/Renata_Duraciova/publication/317102680_Databazove_systemy_v_GIS/links/5926a0f5458515e3d457fa87/Databazove-systemy-v-GIS.pdf⟩.

[12] Fornusek, L.: Databázové technológie v podnikových informačných systémoch, Moravská vysoká škola Olomouc, Ústav informatiky, Olomouc 2009, [cit. 2020-04-02]. Dostupné na internete: ⟨https://theses.cz/id/5x5pam/Fornusek_PIS_2009_-_BP.pdf⟩.

[13] Gorbár, P.: Editor ER diagramů s podporou transformace do relačního modelu a SQL, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, 2007, [cit. 2020-04-02]. Dostupné na internete: ⟨https://www.ksi.mff.cuni.cz/~holubova/bp/Gorbar.pdf⟩.

[14] Informatika štúdium, 2020. [cit. 2020-04-08]. Dostupné na internete: ⟨https://sk-informatika.studentske.cz/2008/09/sprvca-databzy-administrtor.html⟩.

[15] Vlčák, M.: Normalizácia relačných databáz, Univerzita Komenského, Bratislava, Fakulta matematiky, fyziky a informatiky, Katedra informatiky, Bratislava, 2009. [cit. 2020-04-02]. Dostupné na internete: ⟨http://www.dcs.fmph.uniba.sk/bakalarky/obhajene/getfile.php/main.pdf?id=124&fid=235&type=application%2Fpdf⟩.

[16] ManagementMania. [cit. 2020-04-08]. Dostupné na internete: ⟨https://managementmania.com/sk/server⟩.

[17] Rusko, M. – Halász, J.: Environmentálne orientované informačné systémy. Žilina : Strix, Edícia EV-64, Prvé vydanie, 2011. ISBN 978-80-89281-76-3. [cit. 2020-04-02]. Dostupné na internete: ⟨http://www.sszp.eu/wp-content/uploads/Uchovavanie-dat.pdf⟩.

[18] Koščák, J.: Preklad štruktúry relačných databáz do štruktúry grafových databáz, Masarykova univerzita Fakulta informatiky, [cit. 2020-04-02]. Dostupné na internete: ⟨https://is.muni.cz/th/xfe94/diplomka.pdf⟩.

[19] Date, C. J.: Database Design and Relational Theory : Normal Forms and All That Jazz (Theory in Practice) 1st Edition, 2012.

[20] Algoritmizácia. SPŠ elektrotechnická, Komenského 44, Košice. [cit. 2020-04-08]. Dostupné na internete: ⟨https://spseke.sk/tutor/projekt/algoritmy.htm⟩.

[21] Someber, A.: Databázové systémy. 1998. [cit. 2020-04-06]. Dostupné na internete: ⟨http://www.skripta.unas.cz/stahuj/databaza.pdf⟩.

[22] Bílek, Petr: Normalizace. 12. 9. 2015. [cit. 2020-04-06]. Dostupné na internete: ⟨https://www.sallyx.org/sally/psql/normalizace.php⟩.

[23] Dobešová, Z.: Databázové systémy v GIS. Olomouc : Vydavatelství UP, 2004. 76 s. ISBN 80-244--0891-0.

Terminologický slovník

Abstrakcia – je to, čo je odtrhnuté od skutočnosti alebo nenázorné (myšlienkové, pojmové). Abstraktné je niečo odmyslené od jeho nositeľa, napríklad múdrosť od múdreho človeka. Opakom abstraktného je konkrétne.

Administrátor (správca) databázy – osoba, ktorá sa stará o plynulú činnosť celého systému, ale aj skupina programov, ktoré zabezpečujú základné činnosti v správe údajov. Administrátor môže vytvárať, pridávať, meniť a vymazávať objekty v každej databáze a má prístupové práva do všetkých databáz, ktoré sú v rámci systému vytvorené [14].

Agregácia – pojem agregácia znamená zoskupenie, zhlukovanie alebo stmeľovanie.

Algoritmus – je konečná postupnosť dobre definovaných dostatočne elementárnych inštrukcií (krokov) vedúcich k splneniu určitej úlohy v konečnom čase.

Anomália – v informatike anomália znamená stav, keď databáza spôsobuje nekonzistenciu údajov. Poznáme anomáliu mazania údajov, anomáliu modifikácie údajov a anomáliu vkladania údajov do databázy.

Aplikácia – alebo zvykneme označovať aj ako používateľský aplikačný program je počítačový program, ktorý pomáha používateľovi pri vykonávaní činností rôzneho typu, napríklad manipulácia s textom, obrázkov, tabuľkami a podobne.

Asociatívna tabuľka (join table) – tabuľka, ktorá sa spája s inou tabuľkou.

Asociovať – združovať, spájať.

Atomické hodnoty atribútov – hodnoty, ktoré sú ďalej nedeliteľné.

Doména atribútu – množina povolených hodnôt nachádzajúcich sa v konkrétnom atribúte.

Explicitne – vyjadrené priamo, jasne, zreteľne.

Formát údajov – určitý štandardný spôsob, ktorým sú údaje kódované. Formát určuje význam jednotlivých bitov alebo širších pamäťových jednotiek na to, aby mohli byť údaje jednoznačne zakódované a dekódované.

Funkcionalita systému – je súbor funkcií systému, ktoré sú zamerané na určitú oblasť respektíve činnosť.

Hashovanie – je proces, pri ktorom sa ľubovoľne dlhá informácia pretransformuje na informáciu rovnakej dĺžky znakov. K hashovaniu neexistuje opačný proces. To znamená, že pôvodný text nie je možné zrekonštruovať do pôvodného stavu.

Implementácia – uskutočnenie, realizácia, naplnenie; dodržanie záväzku.

Index – je akoby register alebo kľúč na vyhľadávanie. Spravidla to býva abecedne usporiadaný zoznam výrazov vedného odboru. Číslo pri každom z nich je vlastne odkazom na hľadanú stranu, podľa ktorého sa potom ľahko dokážeme orientovať.

Indexový súbor – súbor, kde sú uložené indexy stĺpca (z tabuľky databázy).

Inštancia – príklad, prvok, záznam (v tabuľke).

Integrácia – je zjednotenie nejakých častí do jedného celku.

Integrita (celistvosť) – stav, do ktorého nemožno zasiahnuť, ktorý nemožno porušiť, zmeniť.

Integrita entít – zaručuje, že každá entita bude v RDM jednoznačne identifikovateľná. To je dané tým, že žiadna z veličín tvoriacich primárny kľúč nebude mať hodnotu NULL.

Integrita obmedzenia – sú logické výroky, ktoré majú platiť o údajoch v databáze. Mali by byť odvodené z konceptuálnych modelov, ktoré opisujú objekty reálneho sveta [17].

Integrita údajov – Pod integritou bázy údajov rozumieme jej pohotovú fyzickú disponibilitu, pričom báza údajov je v konzistentnom stave [17].

Kardinalita – určuje spojitosť údajov medzi tabuľkami v relačných databázach. Údaje sú v relačných databázach usporiadané vo viacerých tabuľkách a tie majú medzi sebou rôzne vzťahy (prepojenie). Túto spojitosť určuje kardinalita. Poznáme tri typy: 1 : 1, 1 : N, M : N.

Kartotéka – je organizovaný súbor záznamov na lístkoch rovnakej veľkosti, zoradených podľa abecedy alebo podľa iného systému. Môže to byť aj zariadenie, v ktorom je táto zbierka záznamov umiestnená [21].

Kompozitný vzťah – zložený.

Konzistencia databázy – znamená, že uložené údaje odpovedajú skutočnosti.

Konzistentné údaje – údaje, ktoré sú ucelené (celistvé), v súlade, v harmónii. Predstavme si, že by sme mali rovnaké údaje v dvoch tabuľkách. V jednom momente by sme ich zmenili, každý na inú hodnotu. Ktorá z hodnôt by bola korektná? Konzistentnosť zaručuje, že v každom okamihu sú údaje rovnaké a korektné.

Modifikácia – prispôsobenie, úprava (napríklad údajov v databáze).

Notácia – sústava znakov a symbolov (metajazyka), zápis, zapisovanie, spôsob označenia. (V informatike môže byť aj reťazec charakterizujúci určitý údaj alebo zobrazenie, označenie alebo spôsob zápisu.)

Preferencia – výhoda, prednosť, zvýhodnenie.

Procedurálny jazyk – je jazyk primárne podriadený a zameraný na používanie paradigmy procedurálneho programovania. Procedurálna programovacia paradigma – je to paradigma zakladajúca činnosť programov na spúšťaní podprogramov, funkcií alebo procedúr.

Redundancia – nadbytočnosť, prebytočnosť, presahovanie potrebnej miery. Je často sa vyskytujúci jav, pri ktorom vzniká nadbytočnosť alebo duplicita údajov. V tomto prípade údaje zaberajú zbytočne veľa miesta na pamäťovom úložisku a vtedy môže nastať problém s údržbou databázového systému, ktorého prevádzka sa zbytočne kvôli tomu predraží.

Referenčná integrita – nástroj databázového stroja, ktorý pomáha udržiavať vzťahy v relačne prepojených databázových tabuľkách.

Referenčná integrita – zaručuje, že sústava tabuliek bude navzájom prepojená s možnosťou spájať vzájomné hodnoty z navzájom súvisiacich tabuliek RDM.

Sémantická korektnosť údajov – hodnoty údajov sú správne, konzistentné a aktuálne.

Server – je termín označujúci počítačový program (softvér) a niekedy tiež počítač (hardvér), ktorý poskytuje svoje služby a výkon ostatným počítačom alebo serverom v počítačovej sieti [16].

Skalár – hodnota, ktorá predstavuje najmenšiu sémantickú hodnotu údajov.

Skalárna hodnota – napríklad doména atribútu je množina skalárnych hodnôt rovnakého typu.

SRBÚ (angl. DBMS – database management system) **systém riadenia bázy údajov** – je to systém programov, ktoré sa priamo podieľajú na práci s údajmi v databázovom systéme.

Súbor – je logické zoskupenie diskrétnych údajov.

Štruktúra – systém, zloženie, zostava.

Tranzitívna závislosť – závislosť atribútu od iného, nekľúčového atribútu.

Ing. Milan Štrbo, PhD. (nar. 1986) ukončil v roku 2011 2. stupeň vysokoškolského štúdia na Materiálovotechnologickej fakulte v Trnave Slovenskej technickej univerzity v Bratislave v študijnom odbore *informatizácia a automatizácia v priemysle.* Na rovnakej fakulte pokračoval v doktorandskom štúdiu, ktoré ukončil v roku 2014 a získaním titulu PhD. Po ukončení doktorandského štúdia začal pracovať ako vysokoškolský pedagóg na Pedagogickej fakulte Trnavskej univerzity v Trnave, kde pôsobí na Katedre matematiky a informatiky až doteraz.