Malé veľké databázy II. /4.časť - intermezzo

Namiesto predslovu

Už je to tu! Aj vy isto poznáte ten prípad zo školských lavíc. V našom obľúbenom predmete sme sa ledva prehrýzli povinnou úvodnou teóriou a prešli sme konečne k praxi, ktorá nás toľko zaujíma. A jedného dňa, namiesto praktickej interesantnej hodiny príde učiteľ a zasype nás množstvom daľšej nezáživnej teórie. Vtedy som si myslel, že nám to hádam robí naschvál. Ale on dobre vedel, že sme sa dostali do štádia, keď na ďaľšiu praktickú činnosť naše teoretické znalosti už nestačia a len on vedel, že bez následnej teórie by sme sa dopúšťali na pohľad nepatrných omylov, ale s obrovskými následkami. A verte či neverte, história mu vždy dala za pravdu. Robil to často, skoro s určitou pravidelnosťou, akoby medzi dvomi úrovňovými stupňami nášho odborného života.
A tento stav nastal teraz aj v našom seriáli. Z vašich mailov som zistil, že ste skutočne snaživí študenti a aj vaše projektíky naznačujú, že sa Slovensko báť nemusí o dostatok šikovných programátorov. Mnohí z vás sa púšťajú do rozsiahlejších (inak veľmi solídnych) projektov, a aby ste sa nedopustili tých nepatrných chýb s obrovskými následkami, nastal čas, aby sme si povedali niečo o koncepcii návrhu projektov. Už vás počujem - „Zase len teória!“ (to isté sme vravievali aj my!), ale až sa ňou prehryzieme, zistíme, že dokážeme robiť efektívnejšie projekty, odolnejšie voči systémovým chybám, stabilnejšie a flexibilnejšie voči požiadavkám v budúcnosti. Naozaj, nestraším vás, a skúste mi uveriť, tak ako sme aj my časom uverili slovám nášho učiteľa. A na okraj, tieto teórie budú obecne platné, teda neviažu sa ku konkrétnemu databázovému systému.

Relačný model

Vráťme sa naspäť k našim začiatkom. Hovorili sme si, že vzťahy medzi dátami sú postavené na relačnej algebre. Je to súbor základných matematických princípov odvodených z teórie množín a predikátovej logiky. Relačný model definuje spôsob, akým je možné dáta reprezentovať (štruktúru dát), spôsoby ich ochrany (integritu dát) a nakoniec operácie, ktoré môžeme nad dátami vykonávať (manipuláciu s dátami). Len pre úplnosť - toto dátové modelovanie zaviedol Dr.E.F.Codd a svoje výsledky zverejnil v roku 1970 (teda nič nové), ktorý sa stal neskôr výskumným pracovníkom firmy IBM.
Relačný model nepredstavuje jedinnú metódu spravovania dát. Existujú aj iné možnosti, ako hierarchický, sieťový alebo hviezdicový model. Každý má samozrejme svojich zastáncov aj odporcov a každý má pre určitú skupinu úloh rôzne prednosti. Relačný model je vďaka svojej vysokej efektivite a flexibilite veľmi obľúbený v realizácii databáz z bežného života.
Obecne sa dá povedať, že relačné databázové systémy vykazujú následné charakteristické vlastnosti:
- všetky dáta sa pomyselne dajú reprezentovať v pravidelne usporiadaných štruktúrach s riadkami a stĺpcami, ktorým hovoríme ***relácie***
- všetky hodnoty v databáze sú ***skalárne***. To znamená, že v každej konkrétnej pozícii riadku a stĺpca danej relácie sa nachádza jedna (presnejšie práve jedna) hodnota
- operácie v databáze sa vykonávajú vždy nad celou reláciou a ich výsledkom je opäť iná celá relácia; tomuto hovoríme ***uzáver***

Čo je to tá ***relácia***? Ak v nej poznávate niečo, čo je to čosi ako tabuľka, nie ste ďaleko od pravdy. Dr. Codd pri formulácii relačného modelu zvolil pojem ***relácia***, pretože ten bol voči iným pojmom relatívne „voľný“ a nemal iné, zavádzajúce významy ako napr. slovo ***tabuľka***. Reláciou môže byť čokoľvek, čo je usporiadané do štruktúry (formátu) riadkov a stĺpcov a čo obsahuje skalárne hodnoty. Existencia určitej relácie je teda úplne nezávislá na jej fyzickej reprezentácii (nezaujíma nás, ako a kde je fyzicky uložená na disku).

Základné pojmy

Najabstraktnejšou časťou celého návrhu databáze je ***datový model*** - to je totiž myšlienkový (pojmový) popis daného priestoru problému. Datové modely sa vyjadrujú pomocou ***entít***, ***atribútov***, ***domén*** a ***vzťahov***. Aby sme mohli pokračovať pri objasňovaní teórie a koncepcie databáz, vysvetlíme si základné pojmy, s ktorými budeme naďalej narábať:

Doména

Doména je množina hodnôt rovnakého významu. Doménou môže byť napríklad vek alebo priezvisko. Hodnoty v doméne sú rovnakého dátového typu - číslo, reťazec znakov, dátum a podobne. Je to ***obor hodnôt***, ktoré tvorí množina všetkých prípustných platných hodnôt, ktoré smie určitá položka obsahovať.
Aký je rozdiel medzi oborom hodnôt a dátovým typom?
Dátový typ je fyzický pojem, obor hodnôt je pojem logický: „číslo“ je dátový typ, zatiaľ čo „vek“ predstavuje už obor hodnôt, obsahujúci „čísla“.

A ešte jeden príklad: Položka „Priezvisko“ a položka „Adresa“ majú rovnaký dátový typ - je to reťazec znakov. Ale už z názvov je zrejmé, že budú mať iný obor hodnôt, takže náležia do rôznych domén.

Kartézsky súčin množín A, B

Majme množinu usporiadaných dvojíc *[x, y]* , pre ktoré platí, že *x* patrí do množiny **A** a *y* patrí do množiny **B**. Potom platí, že počet prvkov v kartézskom súčine je daný počtom prvkov v množine **A** *krát* počet prvkov v množine **B**.
Ak máme množinu *A = {1, 2, 3}* a množinu *B = {a, b},* potom je kartézsky súčin **M = A x B** daný takto:

M = {[1,a], [1,b], [2,a], [2,b], [3,a], [3,b]}

Príklad:
V našej knižnici máme dve množiny - množinu kníh, ktoré požičiavame a množinu čitateľov, ktorí si knihy požičiavaju. Kartézskym súčinom je spojenie týchto množín tak, že každá kniha by bola priradená jednému čitateľovi a zároveň by jeden čitateľ mal priradené všetky knihy. Ak máme 10 kníh a 5 čitateľov, kartézsky súčin by obsahoval 10 x 5 =50 záznamov. Spomeňme si, že sme takéto spojenie tabuliek už robili, a to v 8.časti seriálu.

Relácia

Už vieme, čo je to relácia. Matematické vyjadrenie je, že relácia je ľubovoľná podmnožina kartézskeho súčinu, napr: *R = {[1,a], [2,b], [3,a]},* čo je naozaj podmnožinou (čiastočkou) kartézskeho súčinu *M*. To, aké prvky bude relácia obsahovať, je definované práve konkrétnym SQL príkazom s určenými podmienkami.
Príklad:
Ak by SQL príkaz definoval vybratie tých čitateľov, čo splňujú určitú podmienku, napr. nemajú požičanú ani jednu knihu, vytvorila by sa podmnožina kartézskeho súčinu, ktorá vyhovuje tejto podmienke. Tak by vznikla relácia. Relácia môže byť trvalá (ako napr. tabuľka), odvodená (ako určitý pohľad na reláciu trvalú) alebo dočasná (len v pamäti počítača, napr. pri spojovaní tabuliek).

Entita

Entita je čokoľvek, o čom v systéme potrebujeme uchovávať potrebné informácie.
Pri zahájení prác na návrhu dátoveho modelu nie je zostavovanie prvotného zoznamu entít vôbec obtiažne. Ak si rozoberieme (sami, alebo s našim zákazníkom, pre ktorého projekt robíme) priestor problémov, používame podstatné mená ako vhodných kandidátov na entity. *„Autori píšu knihy“*. *„Vydavatelia vydávajú knihy“*. *„Dodávatelia dodávajú knihy“*. *„Čitatelia si požičiavajú knihy“*. Slová ako *autori*, *vydavatelia*, *dodávatelia*, *čitatelia* a *knihy* sú celkom isto entity.
Udalosti, ktoré v našom stručnom „rozhovore“ prezentovali slovesá *písať*, *vydávať*, *dodávať* a *požičiavať* sú **vzťahy** medzi entitami. Nie všetky však budeme v našom dátovom modeli potrebovať. To, že autori píšu, ba ani že vydavatelia vydávajú knihy nás netrápi, ale isto budeme chcieť evidovať vzťah medzi knihou a dodávateľom (*dodávať*), medzi knihou a čitateľom (*požičiavať*).
Entity môžu byť konkrétne - sú odrazom objektu vo fyzickom svete, napr. **kniha**, **čitateľ**, **autor** a pod. ale môžu byť aj abstraktné - modelujú myšlienky. Popisujú určitú vzájomnosť medzi rôznymi entitami. Príkladom môže byť napr. zodpovednosť konkrétneho pracovníka za istú oblasť činnosti firmy.
Veľmi zjednodušene môžeme povedať, že relácia je akási tabuľka o entite.

Atribúty

Vieme, že navrhovaný systém bude o každej entite zaznamenávať, sledovať a vyhodnocovať určité skutočnosti. Týmto skutočnostiam (údajom) sa hovorí ***atribúty danej entity***. Ak náš knižný systém obsahuje entitu **kniha**, chceme o nej viesť údaje o **názve**, **autorovi**, **vydavateľstve**, **dodávateľovi**, **žánri** a **cene**. Toto všetko sú atribúty. Nie vždy bývajú atribúty jednoznačné. Určovanie atribútov je sémantický proces. To znamená, že sa budeme rozhodovať podľa významu dát a podľa spôsobu ich využitia.
Zjednodušene povedané, atribúty sú v podstate stĺpce relácie (tabuľky).
Pozrime sa na jeden príklad - je ním adresa:
Stačí adresu modelovať ako entitu s jedným atribútom (*Adresa*), alebo je to vhodnejšie riešiť ako entitu s viacerými atribútmi (*Ulica, ČísloDomu, Mesto, Okres, PSČ*)? To záleží od požadovaného výsledku. Ak bude adresa slúžiť iba na korešpodenciu, stačí ju uvádzať ako jeden jedinný atribút, s ktorým sa iná činnosť okrem tlače nepočíta. Ak však budeme chcieť vyhodnocovať určitú štatistiku podľa okresov alebo dokonca ulíc, bude vhodnejšie evidovať adresu ako súbor atribútov *Ulica, ČísloDomu, Mesto, Okres, PSČ*.
Tu môžeme použiť dve stratégie:
Prvá stratégia znie takto: začnime požadovaným výsledkom a snažme sa neurobiť návrh zložitejší, než aký nutne musí byť. Dobrým spôsobom je klásť si otázky a na základe odpovede stanoviť požadovanú štruktúru. Stačí si položiť otázku: „Kam mám poslať čitateľovi poštu?“ a ak odpoveď vyhovuje, vyhovuje model s jedinným

atribútom. Ak však položíme otázku „V akom okrese daná osoba býva?“ , podľa odpovede vieme, že musíme nadefinovať inú štruktúru.
Nesmieme zabudnúť, že výsledný model musí byť natoľko flexibilný, aby dokázal zodpovedať nielen otázky, ktoré mu užívatelia budú klásť hneď teraz, ale tiež otázky, ktoré sa dajú do budúcna istým spôsobom predvídať. (A preto väčšina programátorov definuje adresu ako viacatribútovú entitu.). Ale pozor! Cenou za vysokú flexibilitu býva často zložitosť výsledného systému.

Druhá stratégia znie: hľadajme výnimky (výnimočné situácie). Musíme vedieť identifikovať všetky výnimky a systém musíme navrhnúť tak, aby dokázal zvládnuť čo najviac výnimiek bez zbytočného obťažovania užívateľa. Čo to znamená, si ukážeme na príklade mena:
V našich krajoch býva zaužívané, že každý občan má spravidla jedno meno (krstné) a jedno priezvisko. Ale existujú výnimky. Možno existuje pán, čo sa volá Augustus Maroš Kačka. A čo je teraz krstné meno? Augustus? Maroš? A priezvisko? Je to Kačka alebo Maroš Kačka?
Alebo taký Pavol Országh - Hviezdoslav? A čo taký Abúl Quásim Mansúr Firdausí?
Koľko môže byť takých výnimiek? A práve na odpovedi záleží, či evidenčný formulár na napĺňanie databáze bude obsahovať položky ako PrvéMeno, DruhéMeno, PrvéPriezvisko, DruhéPriezvisko, ŠľachtickýTitul alebo len klasický našský - Meno, Priezvisko?
Správne sa rozhodnúť je skutočne veľkou zodpovednosťou. Často treba preštudovať určité národné zvyky, napr. v ruských oblastiach majú osoby zásadne tri mená - rodné, otcovské a priezvisko, napr. Vladimír Iljič Lenin, ale veľmi často sa oslovujú len prvými dvoma - Vladimír Iljič.

Vzťahy

Okrem atribútov jednotlivých entít musíme v dátovom modeli určiť vzťahy, definované medzi rôznymi entitami. Z tvrdenia *„Čitatelia si požičiavajú knihy“* tak vyplýva existencia určitého vzťahu medzi entitami **čitatel** a **kniha**. Entity, zapojené do určitého vzťahu, sa nazývajú **účastníkmi**. Počet účastníkov označujeme ako **stupeň vzťahu**.
Drtivá väčšina vzťahov je *binárnych*, teda s dvomi účastníkmi - ako náš vzťah *„Čitatelia si požičiavajú knihy“*, ale nutné to zďaleka nie je. Bežné sú aj *ternárne* vzťahy, teda vzťahy medzi tromi účastníkmi. Z dvoch binárnych vzťahov *„Autori píšu knihy“* a *„Čitatelia si požičiavajú knihy“* vyplýva ternárny vzťah, vyjadrený vetou *„Autori píšu knihy pre čitateľov“*. Na základe pôvodných dvoch binárnych vzťahov nie sme schopní zistiť, ktorý autor napísal ktorú knihu pre ktorého čitateľa. To dokáže ošetriť až ternárny vzťah.

Kardinalita vzťahu

Pod **kardinalitou vzťahu** rozumieme počet výskytu objektov oboch entít, ktoré sa vzťahu účastnia.
Vzťah medzi ľubovoľnými dvomi entitami môže byť typu **1:1**, **1:n** alebo **m:n**.

Vzťah **1:1** je vzťah, v ktorom na obidvoch stranách vystupuje iba jeden objekt danej entity. Tieto vzťahy sú v realite veľmi zriedkavé. Príkladom môže byť vzťah **manželia** medzi entitami **Muž** a **Žena**. V prípade monogamnej spoločnosti je zrejmé, že jedna žena má iba jedného muža a naopak, jeden muž má iba jednu ženu. Obdobným príkladom môže byť vzťah **triednictvo** medzi entitami **Učiteľ** a **Trieda**, kde učiteľ je triednym učiteľom iba v jednej triede a naopak, jedna trieda má iba jedného triedneho učiteľa.

Vzťah **1: n** (hovoríme mu aj *jedna k viacej*) býva v dátovom modeli najčastejší. Na jednej strane je jediný objekt entity, ktorý je vo vzťahu s jedným alebo viacerými objektami druhej entity. Ako príklad poslúži vzťah **čitateľ - kniha**, kde jeden čitateľ môže mať požičanú jednu alebo viac kníh, ale naopak, viacej kníh môže byť požičaných práve iba jedným čitateľom. Obdobne je to u vzťahu **trieda - žiak** (trieda ako inštitúcia, nie učebňa!), kde do jednej triedy (napr. 3.C) môže chodiť niekoľko žiakov, ale naopak, každý žiak z 3.C chodí iba do tejto jednej (jedinnej) triedy.

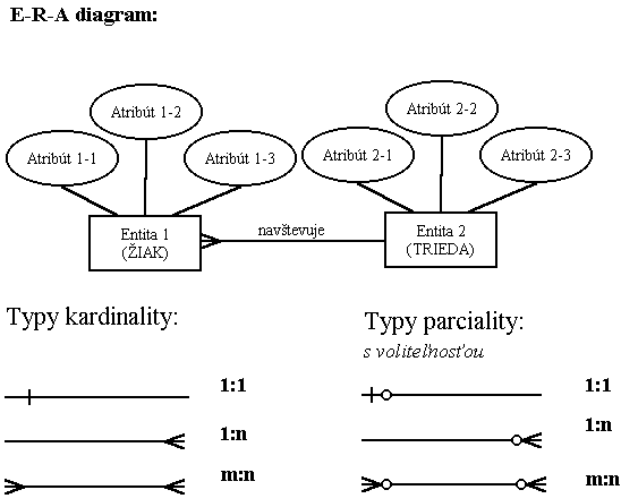
Vzťah **m:n** (nazývaný aj *viac ku viac*) je špecifickým vzťahom, v ktorom vystupuje viac objektov na obidvoch stranách. Napríklad: každý učiteľ vyučuje mnoho žiakov a každý žiak chodí na hodiny k mnohým učiteľom. Alebo vo vzťahu zamestnanec - úloha môže viacej zamestancov riešiť jednu úlohu a zároveň môže jeden zamestnanec riešiť viac úloh.

Parcialita vzťahu

Okrem *kardinality* vzťahu môžeme ešte rozlišovať *povinnosť* a *voliteľnosť* jeho existencie. Pozrime sa na to takto: Musí mať každá žena manžela a každý muž manželku? Musí byť každý učiteľ triednym učiteľom? Vidíme, že sa môžu vyskytovať typy vzťahov, ktoré nemusia existovať u všetkých objektov danej entity - existujú predsa slobodné ženy a slobodní muži a učitelia bez triedníctva.

E-R a E-R-A diagramy

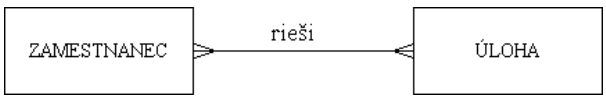
Model entít a vzťahov, ktorý popisuje dáta ako entity, atribúty a vzťahy medzi nimi, zaviedol poprvýkrát Peter Pin Shan Chen v roku 1976. Súčasne navrhol metódu jeho zobrazenia do diagramov, ktoré pomenoval ***diagramy entít a vzťahov*** (Entity - Relationship Diagram), ktoré poznáme pod skratkou ***E-R diagramy***. Ak v E-R diagramoch uvádzame aj atribúty entít, hovoríme o ***E-R-A diagramoch*** (Entity - Relationship - Attribute Diagram). V E-R diagramoch sa entity označujú pomocou obdĺžnikov, atribúty pomocou elipsy alebo oválov a vzťahy sa znázorňujú spojovaciou čiarou medzi entitami. Keďže ani najlepšie nakreslené diagramy nedokážu popísať všetky situácie z reálneho sveta, je nutné doplniť každý diagram slovným popisom. Na obrázku č. 14-1 je zoznam najzaužívanejších značiek v E-R-A diagramoch:



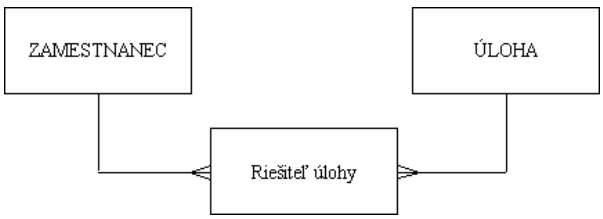
Všimnime si „vidličky“ na strane **žiaka** vo vzťahu k **triede**. Vyjadruje kardinalitu vzťahu žiaka a triedy. Takisto vidíme, že je možné znázorniť particialitu vzťahu - prázdny krúžok vyjadruje voliteľnosť na strane entity, ktorá nemusí existovať. Hovorím, že je to možné, ale nie povinné. Vzťah **1:n** akosi automaticky predpokladá výskyt *0* (nula) až *n* objektov.

Dekompozícia vzťahu m:n

Vzťah **m:n** je z hľadiska ďalšej práce veľmi komplexný a je nutné pokúsiť sa o jeho zjednodušenie. Nerobíme tak kvôli jeho implementácii na ďalšej, logickej úrovni návrhu., ale i pre prípad, že v tomto vzťahu je „ukrytá“ ďalšia entita, ktorá zatiaľ našej pozornosti unikla. Súčasťou tejto entity môžu byť aj atribúty, o ktorých sme tušili, že existujú, ale sme nevedeli, ku ktorej entite ich priradiť.
Dekompozíciou vzťahu m:n rozumieme vytvorenie novej, tzv. *väzbovej* entity, ktorá bude mať vzťahy **1:n** na obidve pôvodné entity vzťahu **m:n**. Pozrime sa na obrázok č. 14-2, čo je E-R diagram vzťahu *m:n* medzi entitami **Zamestanec** a **Úloha**:



Na obr. č. 14-3 je dekompozícia tohto vzťahu:



Novo vzniknutá entita **Riešiteľ úlohy** vyjadruje fakt, že zamestnanec môže byť naraz riešiteľom viacerých úloh a jedna úloha môže byť riešená naraz viacerými riešiteľmi.
Súčasťou tejto entity možu byť atribúty, ktoré nemožno priradiť k žiadnej z entít **Zamestnanec** a **Úloha**. Príkladom je atribút popisujúci hodnotenie zamestnanca za odvedenú prácu na danej úlohe. Pretože zamestnanec môže pracovať na viacerých úlohách a za každú môže byť hodnotený inak, nemôže byť tento atribút umiestnený do entity **Zamestnanec**. Zároveň nemôže byť umiestnený do entity **Úloha**, pretože na jednej úlohe môžu pracovať viacerí zamestnanci, každý s iným úspechom. Jediným správnym umiestnením atribútu **Hodnotenie** je do entity **Riešiteľ úlohy**, pretože sa vzťahuje vždy k dvojici *{zamestnanec, úloha}*.

Tri úrovne návrhu

Návrh projektu a jeho štruktúry by nemal byť, zvlášť u rozsiahlejších systémov, živelným procesom postupne reagujúcim na vzniknuté požiadavky. V súčasnosti existujú historicky overené postupy a pravidlá návrhu, ktoré umožnia a do istej miery aj zaručia vytvorenie kvalitnej dátovej základne, ktorá bude riadne zadokumentovaná, logicky konzistentná a bude umožňovať relatívne jednoduché zapracovanie skutočností, vzniknutých neskôr. Najdôležitejšia je počiatočná analýza oblasti, ktorú chceme v projekte zobraziť, ešte predtým, ako začneme vytvárať tabuľky a SQL príkazy. Čím neskôr totiž odhalíme chyby v návrhu dátovej základne, tým väčšiu námahu a tým pádom aj finančné prostriedky budeme musieť obetovať na ich nápravu.
Jedným z možných postupov je oddelenie popisu oblasti nášho záujmu od vlastnej implementácie na počítači. Spôsob implementácie môže výrazne ovplyvniť štruktúru dátovej základne, ale s oblasťou, ktorú dátová základňa popisuje, nemá nič spoločné. Princíp troch úrovní rozdeľuje postup návrhu datovej základne na tri kroky, ktoré si popíšeme.

Konceptuálna úroveň

Na tejto úrovni sa snažíme popísať predmetnú oblasť pomocou všetkých entít, ktoré sa v nej vyskytujú a všetkých vzťahov medzi týmito entitami. V žiadnom prípade v tejto fáze neberieme do úvahy neskorší spôsob implementácie a do istej miery ani neskoršie obmedzenia technologického charakteru. Týmto môžeme venovať všetku energiu na pochopenie vlastného problému. Nakoniec získame aj obecne platný popis danej oblasti, ktorý môžeme použiť pre implementáciu v odlišných databázových systémoch bez nutnosti opätovnej analýzy.
Tu sú ciele konceptuálneho modelu:
- vytvoriť obraz reality vo formalizovanej podobe, nezávislý na neskoršom spôsobe implementácie
- formalizovať požiadavky užívateľov a dať návrhárom ľahko pochopiteľný prostriedok pre komunikáciu s užívateľom, ktorému budú aj užívatelia rozumieť
- vytvoriť podklad pre návrh dátovej základne

Výsledkom tejto úrovne by mali byť E-R diagramy so slovným popisom zobrazovanej situácie. Predchádzajú im rozsiahle debaty so zadávateľom projektu.
Spokojne sa môže stať, že zadávateľom, projektantom, programátorom, užívateľom a správcom projektu je tá istá osoba - teda my. To však neznamená, že by sme nemohli všetky príslušné debaty a hádky vykonať sami so sebou. (a nie je to duševná porucha). Stačí sa vždy postaviť do jednej z uvedených rolí a nazerať na projekt z iného uhlu. Najkritickejšia je rola užívateľa - ten máva najviac nepríjemných otázok, a preto nebuďme ako programátori ješitní, aby sme sa v tejto "jednoroli" nezhnusili sami sebe!
Osvedčilo sa mi, že pri tvorbe projektu som chvíľu pracoval s budúcim užívateľom. Zistil som jeho zvyky, návyky a názory. Odmenou mi bolo, že sa tento užívateľ neskôr na projekt nesťažoval.

Ak nemáme o oblasti budúceho projektu ani "páru", požiadajme zodpovedného pracovníka, aby nám ukázal svoju činnosť "v tužke", teda ako pracuje doteraz s papierovou formou agendy. Čo píše, čo maže, ako to spočitáva, ktoré informácie sú pre neho vstupy a ktoré on produkuje ako svoje výstupy.

Prax hovorí, že tejto oblasti treba venovať asi 70 % celkových nákladov na projekt. (Myslím tým nielen peniaze, ale aj úsilie, čas a iné nefinančné prostriedky).

Ak toto všetko máme úspešne za sebou, pristúpime k logickej úrovni návrhu.

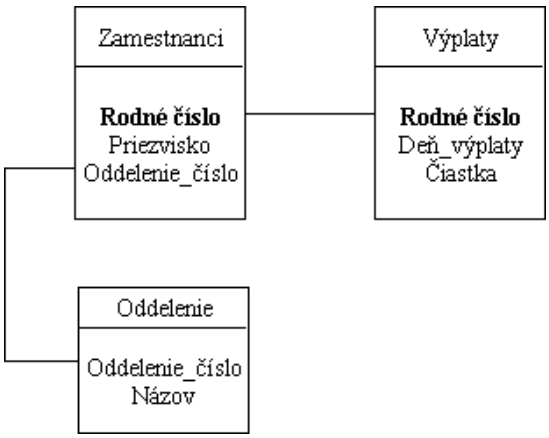
Logická úroveň

Pre popis dát na logickej úrovni sa v relačných databázach používa tzv. **relačná schéma**. Relačná schéma obsahuje tabuľky vrátane všetkých ich stĺpcov. V schéme sú vyznačené ***primárne kľúče*** v tabuľkách, ale aj ***cudzie kľúče*** ako odkaz na primárne kľúče v inej tabuľke. Tento odkaz je väčšinou vyznačený ako čiara spojujúca stĺpce v dvoch tabuľkách. Súčasťou relačnej schémy možu byť aj popisy integritných omedzení tabuliek a stĺpcov.

Prevod konceptuálneho modelu dát na relačnú schému je možné skoro automatizovať pomocou nasledujúcich pravidiel:

1) Každá entita v konceptuálnom modeli sa stáva samostatnou tabuľkou.
2) Identifikátor entity sa stáva primárnym kľúčom
3) Ak je súčasťou vzťahu jeden alebo viac atribútov, je nutné vytvoriť novú (väzbovú) entitu podobne ako u vzťahu *m:n* a z nej vytvoriť tabuľku.
4) U každého vzťahu zvolíme tabuľku, ktorá bude obsahovať cudzí kľuč ako odkaz do inej tabuľky. Pre voľbu tabuľky použijeme tieto pravidlá:
   a) Ak je vzťah typu *1:n*, bude cudzí kľúč pridaný do tabuľky na strane *n*.
   b) Ak sa jedná o parciálny vzťah *1:1*, bude cudzí kľúč pridaný do tabuľky, ktorá sa vyskytuje vo vzťahu nepovinne.
   c) Ak sa jedná o neparciálny vzťah *1:1*, volíme tabuľku, ktorá je vecne "podriadená" (napr. vo vzťahu *zamestnanec - učiteľ* to bude učiteľ, lebo je špecializovanou formou (podmnožinou) zamestnanca).
   d) Vzťahy typu *m:n* najprv dekomponujeme na dva vzťahy *1:n* a potom postupujeme podľa pravidiel pre tento typ vzťahov.
5) Ak sa vyskytne v konceptuálnom modeli špecializácia, máme niektorú z týchto možností:
   a) Vytvoríme jednu tabuľku (napr. Zamestnanec), ktorá bude obsahovať stĺpce pre všetky atribúty, vrátane tých, čo sa vyskytujú len u špecializovaných entít (napr. učiteľov). Na každom riadku zostanú niektoré stĺpce nevyplnené (budú obsahovať hodnotu *NULL*). Tento spôsob je najmenej náročný z hľadiska tvorby SQL dopytov, ale je nehospodárny miestom, ktoré budú dáta zaberať, a spracovávanie dopytov nad takto vytvorenou tabuľkou bude trvať dlhšie.
   b) Vytvoríme zvláštnu tabuľku pre každú špecializovanú entitu. Budeme mať teda tabuľky **Učitelia**, **Sekretárky**, **Študijní_Referenti** atď. Nebudeme sice plýtvať miestom, ale bude nám činiť potiaže práca so všetkými zamestnancami naraz - púhe vypísanie menného zoznamu, zvýšenia platu a podobne. Veľmi ľahko sa môže stať, že budeme nútení pridať ďalšiu tabuľku (napr. **Externisti**) a zabudneme opraviť niektorý z už vytvorených dopytov, pracujúci so všetkými zamestnancami.
   c) Vytvoríme tabuľku **Zamestnanci**, ktorá bude obsahovať stĺpce spoločné pre všetky typy zamestnancov. Pre každú špecializáciu (entitu) potom vytvoríme tabuľku ďalšiu, ktorá bude obsahovať stĺpce špecifické pre túto entitu. Tento spôsob spojuje výhody obidvoch predchodzích. Musíme však zaistiť referenčnú integritu dát medzi špecializovanými tabuľkami a tabuľkou hlavnou.

Na obr. č. 14-4 je ukážka jednoduchej relačnej schémy, popisujúcej tri tabuľky - **Zamestnanci**, **Výplaty** a **Oddelenia** a ich vzájomné vzťahy. Tá by mala byť výsledkom tejto úrovne návrhu:



Na záver definujeme aj príkazy na získavanie a manipuláciu s dátami v jazyku SQL a pristúpime k implementačnej úrovni.

Implementačná úroveň

Na implementačnej úrovni vyberáme konkrétny databázový systém, v ktorom vytvoríme dátovu základňu. Po jeho výbere môžeme začať využívať aj rôzne neštandardné funkcie zvoleného prostredia. Ich použitie by sme však mali dôsledne zvážiť, zvlášt kvôli možnému neskoršiemu prechodu na iný databázový systém. Z hľadiska jazyka SQL je nutné na tejto úrovni vziať do úvahy aj možné dielčie odlišnosti v príkazoch, zvlášť v skupine príkazov pre definíciu dát.

Nabudúce sa budeme venovať normalizácii relácií. Že ste to ešte nepočuli? Tak to je na databázach asi tá najdôležitejšia časť teórie.

Miroslav Oravec
www.mior.host.sk
mior@host.sk
moravec@pcrevue.sk