Navrhování relačních databází

Školící materiály

ACI US, Inc. ©1995
20883 Stevens Creek Blvd.
Cupertino CA 93014
(408) 252-4444
e-mail: training@acius.com
url: http://www.aci-4d.com

Verze: 4D 6.0.5

Vytvořeno:

Jméno	E-Mail
Jeff Browning	browning@acius.com

Upraveno:

Jméno	E-Mail
Jaroslav Macháček	inforce@mbox.vol.cz

1. Úvod

1.1. Vítejte

Toto školení je zaměřeno na získání znalostí o úspěšném navrhování relačních databází. I když tento kurz učí na příkladech vytvářených ve 4th Dimension (4D), většina materiálu může byt aplikována na libovolnou relační databázi.

Obsah kurzu je na úrovni středoškolské výuky a není potřeba žádných velkých znalostí ze světa počítačů a počítačové hantýrky.

Školení jsou vedena pro středoškolské a vysokoškolské odborné pracovníky, školitele a začínající programátory. Jako nástroj byla zvolena 4th Dimension, protože její možnosti dovolují probírat kursy a příklady s minimálními znalostmi. Závěrem kurzu by jste měli být schopni vést rozhovory s klientem, který si přeje vyvinout relační databázový systém a s využitím získaných znalostí převést požadavky do funkčního návrhu databáze.

Měli by jste být schopni vytvořit řadu obrázků, s jejichž pomocí budete komunikovat s klientem (a jakýmkoliv jiným vyvojářem) o podstatě Vašeho návrhu.

1.2. Jak pracovat v kurzu

1.2.1. Čím se budete zabývat

Během kurzu prověříme detailně navrhování relační databáze. Během tohoto procesu definujeme množství technických termínů. Nenechte se tím odradit. Teorie relačních databází je dobře pochopitelná, protože vychází ze života. Všechno z tohoto konceptu je velice jednoduché, když to pochopíte.

Budeme kreslit obrázky návrhů databází v průběhu mnoha praktickych příkladů, které tento kurs obsahuje. Budeme používat jednoduchou technologii nazývanou papír a tužka. Existuje množství všelijakých nástrojů „CASE“ pro účely navrhování databází, pro náš účel není však vhodnější než tužka a papír. Editor struktury 4D je často používán jako jednoduchý „CASE“, pro nás má však zpočátku při vysvětlování a zkoušení některé nevýhody.

- Nezapomíná (soubory a pole nelze vymazat).
- Vedení čar v návrhu označujících relace nelze řídit.
- Některé použité termíny, jako primární klíč, nelze jednoznačně označit a odlišit.

Tužka a papír mají na druhou stranu tyto výhody :

- Jsou jednoduše přenosné.
- Umožňují všechny opravy (guma zde slouží univerzálně).
- Mají nízké náklady.
- Jsou uživatelsky přívětivé (nikdo z frekventantů se je asi nemusí učit).

Tento kurs není typickým kurzem s pomůckami z počítačového světa (v tomto chápání práce se softwarem). I když budeme pracovat s pomůckami tak, že celou dobu budeme konkrétně navrhovat databáze z reálného života.

Pamatujte při práci s příklady, že občas je možné více než jedno správné řešení a správná odpověď. Vše vždy záleží na konkrétních vstupních podmínkách. Navrhování relačních databází je rovněž druhem umění stejným dílem jako vědou. Obvykle Vás příručka upozorní, je-li možné více řešení a uvede příklad jednoho z řešení. Budete-li potřebovat potvrzení, že i Vaše odpověď je správná konzultujte svého školitele u kurzů vedených školitelem.

Ke konci kurzu začneme používat 4D, abychom již převedli návrh do skutečné struktury databáze. V závěru kurzu probereme optimalizační možnosti zahrnuté do navrhování databází ve 4D. Během této části budeme již používat 4D naplno.

1.2.2. Dělení kurzu

Normální trvání kurzu je 3 dny, včetně příkladů, t.j. cca 18 vyučovacích hodin.

První den:Nejdříve se zaměříme na teorii relačních databází a budeme definovat některé termíny. Odpoledne se zaměříme podrobně na pravidla návrhů databází s užitím koncepce nazývané normalizace. V průběhu této sekce si uvedeme několik dobrých a špatných příkladů návrhu databáze.

Druhý den: Naučíme se způsob reprezentovat databázi v obrázcích s použitím způsobu zvaného věcný relační diagram. Během této sekce vyzkoušíme již pokročilé databázové problémy.

Třetí den: Dopoledne, začneme zkoušet zahrnout obchodní pravidla a omezení, která musí být do návrhů databáze zakomponována. Budeme rovněž diskutovat způsob výběru názvů. Pak probereme bezpečnost jako součást návrhu databáze. Odpoledne se budeme zabývat otázkami chování databáze a uvedeme různé tipy a triky spojené s implementací 4D do relačních databází. V této sekci se naučíte kdy obejít nebo přizpůsobit pravidla k získání lepšího chování. Kurz zakončíme řadou příkladů, které prověří a procvičí vše co jste se v kurzu naučili.

Kurz je rozdělen tématicky do dvou částí. Jeden a půl dne se téměř vyhradně týká fáze navrhování při vývoji databází. Pouze poslední den je zaměřen na tvorbu databází s diskuzí optimalizace a možnostmi zahrnutí jednotlivých částí návrhu. Materiál důsledně odděluje tyto části vývoje. Rozbor problému, který navrhujeme (nikoliv programujeme) je záchytný bod, ke kterému se budete vracet při tvorbě databáze, kdy Vás nástroj (4D) povede k obejití, nebo porušení pravidel. Je to však lehčí (a bezpečnější) porušovat pravidla až po tom, kdy byl proveden kompletní a korektní návrh. Naše doporučení je vždy nejdřív dokončit návrh a pak teprve zkoumat oblasti, kde návrh potřebuje upravit, tak aby splnil požadavky na chování.

1.3. Konvence

Tento školící materiál užívá určité typografické konvence, zahrnující:

- Následující *písmo kurzíva* indikuje termín, který je právě definován a je používán poprvé.
- Normální písmo odpovídá databázovým objektům jako jsou tabulky a sloupce
- Znaky, které máte napsat jsou v tomto písmu.

  Příklad: Napište Johnson do pole Jméno.
- Speciální klávesy a znaky klávesnice jsou uvedeny: Enter a Return.
- Vyběr položky z nabídky je uveden následovně:

  Vyberte Soubor Â Otevřít

  Vysvětlení: Z nabídky Soubor, zvolte položku Otevřít.
- 4D programovací příkazy jsou uváděny stejně jako jsou zobrazovány ve 4D. Příklady zahrnují:

[Zákazníci]Země	Pole
ALERT	Příkaz
MyProcedure	Procedura

1.4. Fáze vývoje softwaru

Tento kurz je jedinečný mezi kurzy, které tvoří Univerzitu ACI. Ostatní kurzy se zabývají samotnou tvorbou databází pod 4D, a tak obsahují detaily o kódu, příkazech a syntaxi jazyka 4D a další jemnosti programovacího procesu.

Tento kurz se nezabývá detaily programování v žádné z kapitol. Zůstává pouze u témat z fáze návrhu vývoje softwaru.

Vývoj softwaru lze rozdělit na následujících šest částí :

- Určení systému
- Anylýza požadavků
- Návrh
- Tvorba (programování)
- Testování
- Údržba

Stručně definujme, co se rozumí pod jednotlivými termímy.

Určení systému odpovídá procesu rozboru, zda vůbec nový systém je potřeba a popisu problému a popis je proveden v zcela obecných termínech. Tato oblast je především obchodnická a není zahrnuta do žádného kurzu Univerzity ACI.

Analýza požadavků je určeno do detailu co se od software očekává za funkce a jak se bude postupovat dále. Opět tato fáze není pokryta žádným kurzem Univerzity ACI.

Návrh odpovídá procesu vytváření plánu jak bude software vytvářen, kde budou data uložena, jaký bude zdroj dat a jak data potečou systémem. To je tento kurz, kde se soustřeďujeme především na část dat, jak je dělit, ukládat a do jakých objektů.

Tvorba odpovídá procesu fyzické tvorby softwaru. Je to téma všech ostatních kurzů Univerzity ACI. Kromě toho, se budeme touto částí zabývat v posledních částech tohoto kurzu.

Testování je proces vyhodnocení a potvrzení, že software provádí a umožňuje to co se od něj očekává. Formální část testování není pokryta žádným kurzem ACI, i když způsoby a samotné testování vytvořených částí jsou samozřejmě zahrnuty.

Údržba je následný proces podpory software po nasazení a do úplného dokončení. Opět tato část není explicitně zahrnuta v žádném z kurzů Univerzity ACI.

Každá z těchto fází se v projektu až již ve více či méně formalizované podobě objeví. Například v případě malého ad hoc systému pro užití pouze jedním uživatelem (který bývá často i sám sobě programátorem), je většina fází přítomna pouze v hlavě programátora.

Na druhé straně, u velkých víceuživatelských systémů klient server pro velké společnosti, se tyto fáze pravděpodobně objeví v řádně formalizované/ papírové podobě. Ve skutečnosti mohou být některé fáze zkráceny, návrh například na návrh architektury a přímé vytvoření podrobné struktury již v samotném prostředku programování (4D) a jeho následným vytištěním, např.

1.5. Výhody formálního návrhu databáze

Základním principem počítačové vědy je, že všechny tyto fáze se vyskytují ve vývoji libovolného systému, bez ohledu na to, zda si je návrháři systému uvědomují, či nikoliv. Bylo rovněž prokázáno, že některé formální stránky v procesu návrhu mají obecně vliv na výrazné zlepšení výsledku, a rovněž snižují náklady na vývoj.

V případě malého ad hoc systému popisovaného výše je relativně malý následek i případných velkých chyb v návrhu. To je snadno pochopitelné, protože celý program může byt kompletně přepsán v relativně krátkém čase a s vlastními náklady.

Jak roste velikost projektu, následky základních chyb v návrhu rostou exponencielně.

Následující tabulka graficky ilustruje tento problém. čísla znamenají relativní náklady na korekci daného problému v závislosti na tom, do které fáze byl zaveden a v které fázi byl objeven.

Fáze zavedení

	Požadavky	Návrh	Tvorba
Fáze objevení			
Požadavky	1 x		
Návrh	2 x	1 x	
Tvorba	15 x	5x	1 x
Testování	25 x	10x	5x

Zdroj : Reference 1

Je zřejmé, že formalizovaná fáze návrhu, může ušetřit značné prostředky ve velkých vývojových projektech. Na druhou stranu neformalizovaný návrh (a tím obvykle nekompletní) obvykle odsoudí velké softwarové projekty k neúspěchu.

Použijeme-li tento princip na 4D, je zřejmé, že návrh struktury databáze je základní architektonické rozhodnutí. (Návrhem struktury zde samozřejmě rozumíme definici tabulek a polí v editoru struktury). Když je struktura navržena, je obvykle nesnadné ji výrazně měnit. Obzvlášť nesnadné jsou pak tyto změny, pokud jsou již vytvořeny formuláře, procedury a další programové objekty. (Doplňování a rozšiřování struktury je možné s daleko menším trestem, míníme zde především komplexní změny i ve struktuře tabulek)

Další výhody formalizace návrhu software jsou :

- Kontrola rozsahu. Protože klient obvykle očekává systém, který „umí vše co potřebujeme“, je riziko podstatného rozšiřování (přinejmenším v myšlení klienta) na něco co jste vůbec nezamýšleli vždy přítomno. Detailní dokumentace návrhu Vám dovolí získat formální souhlas klienta, že nic co není v dokumentaci nebude ani v celkovém projektu. Toto je často jediný způsob, jak se dostanete do fáze, kdy můžete říci jsem hotov.
- Systémová dokumentace. Ve skutečnosti každý software libovolné velikosti je udržován nakonec někým jiným, než kdo jej původně navrhoval a programoval. To znamená, že jiná osoba má problém rozebrat se v tom, co jste zamýšleli při návrhu systému. Zatímco dobře okomentovaný kód a další interní systémová dokumentace může v části podrobností velice pomoci, rozhodně neposkytne „makro“ pohled. Pouze detailní dokumentace k návrhu může dát programátoru údržby obrázek jak systém skutečně pracuje.

Protože 4D je dostupná varianta pro velké systémy školení k způsobům návrhu pro 4D software je potřeba. Tento kurz naplňuje tuto potřebu. Doufáme, že s pomocí tohoto kurzu budete schopni navrhnout skutečně profesionální databáze s pomocí 4D.

2. Pochopení modelu relačních databází

2.1. Co je databáze ?

Databáze je organizovaný nebo struktrovaný soubor dat. Obsahuje data nezbytná pro určitý účel, nebo účely. Data jsou uspořádána tak, aby dovolila určité, specifikované aktivity a tak aby byla dostupná a měnitelná účinným způsobem. Zdroj: Reference 2

Je zřejmé, že tato definice nepožaduje, aby databáze byla uložena na počítači. Některé praktické příklady zahrnují :

- Obchodní telefonní seznam
- Obchodní rejstřík
- Karty předpisů pacientů
- Seznam hodnocení žáků učitelem

Dalších příkladů je nespočet. člověk sbírá a třídí informace neustále, je to základ inteligentního života.

Můžete některé příklady databází z běžného života vymyslet a uvést?

Příklad : Rozpoznání databáze

Příklad 2.1

Často je dobré rozumět tomu co je a co není databáze a jak může být s drobnou změnou databáze vytvořena.

Vezmeme následující příklad: Univerzita chce nabrat nové studenty, takže umístí poutače do velkého počtu středích škol a žádá studenty o projevení zájmu dopisem. Obdrží velký počet odpovědí. Je zřejmé, že tyto dopisy budou obsahovat údaje jako jméno, adresu, město, PSČ. Některé rovněž budou obsahovat jiné informace, jako střední školu uchazeče, současné dokončené vzdělání, maturitní prospěch, průměrný prospěch za studium. Dopisy jsou po obdržení založeny do archivu pošty.

- Je to databáze? Jestliže ano, proč? Jestliže ne, proč?
- Jesliže administrativa univerzity chce nalézt určitý dopis studenta, může tak učinit?
- Jestliže administrativa univerzity chce třídit dopisy podle okresu a města, může to učinit snadno?

Po tom, kdy si uvědomili svou chybu přístupu, přidali k poutačům trhací bloky s formulářem pro studenty. Studenti nyní vyplní formulář s následujícími informacemi.

- Příjmení
- Jméno
- Ulice, číslo
- Město
- Okres
- PSČ
- Telefon
- Střední škola
- Dosažené vzdělání
- Maturitní průměr
- Studijní průměr

Opět univerzita obdrží velký počet odpovědí s vyplněnými formuláři a všechny je založí do šanonu seřazené dle příjmení.

- Je to databáze? Jestliže ano, proč? Jestliže ne, proč?
- Jesliže administrativa univerzity chce nalézt určitý dopis studenta, může tak učinit?
- Jestliže administrativa univerzity chce třídit dopisy podle okresu a města, může to učinit snadno?

2.2. Co je systém řízení databáze ?

Systém řízení databáze (DBMS) je software, který řídí data směrem k organizaci a strukturuje je. Zdroj: Reference 2

Tato definice vyplývá velice logicky z definice databáze uvedené vyše.

Běžné rysy nabízené většinou systémů řízení databází zahrnují :

- Způsob jak definovat šuplíky (typicky tabulky a sloupce) k uchování dat
- Příkazy a další rysy dovolující uživateli vyhledávat a třídit data
- Příkazy dovolující uživateli vkládat nová data, upravovat a mazat existující data
- Nástroj pro tvorbu uživatelského rozhraní (GUI) a další rysy dovolující vytvořit formuláře pro prohlížení
- Generátor zpráv dovolující tisknout data
- Podporu integrity dat a nástroj pro výpočty
- Současné ovládání dat pro více uživatelů

Zdroj: Reference 2

Z této definice je jasné, že 4D je systém řízení dat. Podporuje všechny výše zmíněné rysy.

Editor struktury splňuje první podmínku a dovoluje uživateli definovat tabulky a sloupce (pole).

Hledání a třídění záznamů podporují následující příkazy (mezi jinými):

QUERY
QUERY SELECTION
QUERY BY LAYOUT
QUERY BY FORMULA
ORDER BY SELECTION
ORDER BY FORMULA

Vkládání, úpravy a mazání záznamů je podporováno ve 4D následujícími příkazy (mezi jinými):

ADD RECORD
CREATE RECORD
SAVE RECORD

MODIFY RECORD

MODIFY SELECTION

DELETE RECORD

DELETE SELECTION

Tvorba rozhraní je umožněna editorem formulářů 4D (mimochodem původní graficky kreslící program).

Generování zpráv je umožněno editorem formulářů 4D a editory rychlých zpráv, štítků a diagramů.

Podpora relační integrity je umožněna řadou možností při vytváření relací v prostředí návrháře volbou v dialogu a kromě toho i následujícími příkazy:

RELATE ONE

RELATE MANY

AUTOMATIC RELATIONS

Řízení současného přístupu více uživatelů k datům je poskytováno pomocí architektury klient/server prostřednictvím 4D Server a 4D Client.

2.3. Co je relační databáze?

První pokusy vytvořit počítačové databáze byly úzce svázány s problémem fyzického ukládání dat na disk. Vyústilo to nakonec v řadu požadavků na vyvojáře systému a uživatele, aby byli vůbec schopni obdržet požadovaný výsledek. Například k vyhledání záznamů z disku bylo nezbytné napsat kód pro postupné načítání všech dat s použitím skutečných adres uložení záznamů na disku. T.j. v době programování tyto adresy musely být známy.

Model relačních databází byl navržen, aby vyřešil tyto problémy. Model relačních databází byl navržen k oddělení návrhu databáze a jejího užití od ukládání na disk. Tento koncept je znám jako nezávislost dat.

Relační databázový model vidí data ve formě tabulek. Zde je příklad :

Součásti

Číslo součásti	Popis
27	Nosič deštníků
32	Kobereček
48	Bič

V jazyce relačních databází je tento malý soubor dat znám jako tabulka. Také v modelu relační databáze jsou data reprezentována ve formě tabulky. Relační databáze je jednoduše databáze, v které uživatel vnímá všechna data obsažená v systému jako soubor tabulek. Povšimněme si, že způsob fyzického uložení záznamů zde není důležitý. Jediný podstatný pohled je pohled uživatele. Jak jsou data ve skutečnosti na disku uložena není pro naše účely podstatné.

Podíváme –li se na 4D tak zde zřejmě uživatel vnímá většinu dat jako tabulky. Je zde pouze jedna vyjímka a to jsou podtabulky. Podtabulky patří do hierarchické struktury a je to tedy nerelační rys 4D. Z praktických důvodů nebudeme podtabulky v tomto kurzu diskutovat a budeme 4D vnímat pouze jako relační databázový řídící systém (RDBMS).

2.4. Základní terminologie relačních databází

Předchozí diskuze obsahuje některé termíny, které v následující části definujeme formálně.

2.4.1. Tabulka

Tabulka je dvourozměrný soubor dat, to znamená, že obsahuje sloupce a řady.

Podívejme se opět na předchozí tabulku. Vypadá jako tabulka tabulkového procesoru, že? Ale je zde rozdíl, který odlišuje tabulku databáze od tabulky tabulkového procesoru.

Tabulka databáze nemá pořadí. Má to dva aspekty. První, znamená to, že pořadí ve kterých jsou sloupce nebo řady zobrazeny může být libolně měněno a nezáleží na něm. Všechno toto může být provedeno bez zásahu do tabulky databáze. (Zatímco v tabulkovém procesoru měníme tímto definici celé tabulky.)

Jako příklad se podívejme na následující tabulku :

Součásti

Popis	Číslo součásti
Kobereček	32
Nosič deštníků	27
Bič	48

Ačkoliv se změnilo pořadí řad i sloupců je to v chápání relačních databází stále tatáž tabulka. (Kobereček=32,.....)

V tomto kurzu budeme užívat konvence 4D k označení tabulek. Proto tabulku ukázanou nahoře budeme značit [Součásti].

2.4.2. Sloupec

Definice sloupce vyplývá z definice tabulky. Sloupec je jednoduše položka dat stejného druhu obsažená v každé řadě tabulky. Sloupce jsou někdy nazývány vlastnosti tabulky , protože obsahují něco z tabulky databáze co je zamýšleno uložit v každé řadě.

4D užívá termínu pole pro sloupec. V relační terminologii budeme však zde používat termín sloupec.

Jestliže mluvíme o sloupci v konkrétní tabulce budeme se na ně odvolávat jejich skutečným plným názvem. To znamená kombinací názvu tabulky a sloupce. Pokud se budeme odvolávat budeme používat konvenci 4D.

- Příklad: [Tabulka]Sloupec

Někdy se odvoláme jen na sloupec bez ohledu na jeho příslušnost k určité tabulce (buď je tabulka zřejmá, nebo např. sloupec, který je obsažen ve více tabulkách). V tomto případě se jednoduše odvoláme na sloupec pouze názvem sloupce.

- Příklad: Sloupec

Sloupce by měly obsahovat pouze jednu položku dat v každé řadě. Řečeno jinými slovy sloupec by neměl obsahovat více než jednu položku dat v jedné řadě. Technicky tato situace odpovídá skládání sloupců. Typický příklad takovéhoto skládání jsou účty a kódy pro zakázky

Kompletní číslo zakázky:

123-456-7890

kde

123 je kód společnosti

456 je kód střediska

7890 pořadové číslo zakázky

Vyhněte se takovýmto typům sloupců ve vaší databázi jak to jen půjde. Účetní se mohou usmívat, ale databáze s takovým údajem bude velice těžkopádně pracovat pokud musíte současně pracovat s jednotlivými částmi tohoto údaje. Buď musíte zdvojeně ukládat údaje o všech součástech společnost, středisko, pořadové číslo zakázky, nebo při práci s hledáním, tříděním.. budete muset napsat specielní procedury a nebo hledat přes typ hledání - údaj obsahuje. Obě tyto varianty jsou matoucí.

Jsou i jiné problémy, které vznikají, když se jedna ze součástí složeného údaje změní. Např. předpokládejme, že pořadové číslo zakázky putuje z jednoho střediska do druhého. To teoreticky předpokládá rovněž změnu v sloupci kompletní číslo zakázky. To nemusí být možné, pokud tento sloupec je určen k jednoznačné identifikaci údajů řady v tabulce (Viz část dále o primárních klíčích). Alternativa je ponechat složená data v sloupci nezměněna a upravit jen data v jednotlivých sloupcích z nichž byl údaj složen, jestliže to provedete jsou vaše data již nekonzistentní (neodpovídají původnímu záměru a není zřejmé zda jde o chybu...).

Lepší přístup je rozdělit složený sloupec do třech individuálních sloupců, jak je ukázáno v příkladu:

Kompletní číslo zakázky

Kód společnosti	Kód střediska	Poř.č. zakázky
123	456	7890

Každý z těchto tří sloupců nyní obsahuje jeden a pouze jeden údaj, položku dat. Takovýto sloupec kde sloupec obsahuje jednu a pouze jednu hodnotu dat pro každou řadu se nazývá atomární. Pokud navrhujete databázi vytvořte sloupce natolik atomární jak to jen půjde.

(Budeme záměrně vytvářet složené sloupce pro účely chování databáze v posledních částech kurzu. Nevynechejte je. Nyní navrhujeme, neprogramujeme. V té době kdy máte jasný návrh, je možné porušit pravidla, zvláště pokud jasně víte, že je porušujete. Více o tomto později.)

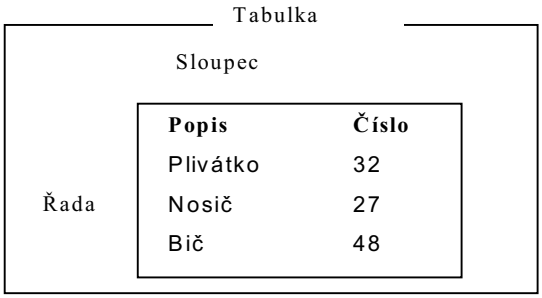
2.4.3. Řada

Podobně jako sloupce, definice řady vyplývá z definice tabulky uvedené výše.

Řada je jednoduše řečeno jeden výskyt čehokoli co je uloženo v tabulce. Každá řada obsahuje každý sloupec v tabulce dokonce i tehdy, jestliže hodnota v libovolném sloupci není definována (je prázdná).

Ve 4D je řada nazývána záznam. V následujícím budeme používat termínu z relačních databází tzn. řada. Pouze si pamatujte, že řada a záznam je totéž.

Následující diagram graficky shrnuje to co jsme probírali v tabulkách, sloupcích a řadách:



2.4.4. Primární klíč

Koncept primárního klíče je základem pro model relačních databází. Model relační databáze uvádí, že každá řada v tabulce musí být jedinečná. Proto v tabulce v každé řadě musí existovat buď jeden sloupec nebo kombinace, které jednoznačně identifikují danou řadu. Tento sloupec nebo kombinace sloupců je nazýván primárním klíčem.

Primární klíč je sloupec nebo kombinace sloupců, které mohou být použity k jednoznačné identifikaci řady ze všech ostatních řad v tabulce.

K objasnění tohoto termínu rozšiřme náš známý příklad. Uvažujme nyní tři tabulky místo jedné:

Součásti

Číslo součásti	Popis součásti
27	Nosič deštníků
32	Kobereček
48	Bič

Obchody

Kód obchodu	Sídlo obchodu
A	Buffalo
B	Hong Kong
C	New York

Sklady

Číslo součásti	Kód obchodu	Skladové množství
27	A	10
27	B	15
27	C	7
32	B	25
48	A	8
48	B	12

Primární klíč tabulky [Součásti] je [Součásti]Číslo součásti. Tento sloupec jednoznačně identifikuje každou řadu mezi ostatními řadami tabulky. Podobně primární klíč tabulky [Obchody] je [Obchody]Kód obchodu.

Když se podíváme na tabulku [Sklady], na první pohled není zcela zřejmé co je zde primární klíč. V této tabulce neexistuje jediný sloupec, který by jednoznačně identifikoval jednu každou řadu této tabulky.

Odpověď je, že tabulka [Sklady] má primární klíč vícesloupcový. Primární klíč této tabulky se skládá ze dvou sloupců: [Sklady]Číslo součásti a [Sklady]Kód obchodu. Primární klíč, který obsahuje pouze jeden sloupec se nazývá jednoduchý klíč. Primární klíč, který obsahuje více než jeden sloupec, jako například předchozí kombinace [Sklady]Číslo součásti a [Sklady]Kód obchodu, je nazýván složený klíč.

(4D se v tomto místě liší od standartního modelu relační databáze,protože explicitně nepodporuje složené klíče. Neobávejme se však nyní navrhujeme a neprogramujeme. V části, kde se budeme zabývat programováním si ukážeme jakým způsobem ovládat ze 4D složené klíče. V dalším návrhu budeme se složenými klíči normálně zacházet.)

Dalším rysem primárního klíče je neměnost. Neměnost určuje, že hodnota v primárním klíči se nezmění nikdy v období existence řady.

Vezměme si jako příklad následující tabulku:

Osoba

Příjmení	Jméno	Rodné číslo
Vávra	Daniel	550613/0485
Pravdová	Karla	655512/0567
Vronský	Marek	640712/0123
Honzík	Petr	710321/1759

Podíváme-li se na tuto tabulku sloupců [Osoba]Příjmení a [Osoba]Jméno jednoznačně určuje jednu každou řadu v tabulce, máme zde ale problém s neměností těchto sloupců. Předpokládejme, že se Vávra Daniel ožení s Pravdovou Karlou a budou mít stejné příjmení. Najednou hodnoty uložené v tabulce budou vypadat následovně:

Osoba

Příjmení	Jméno	Rodné číslo
Vávra	Daniel	550613/0485
Vávrová	Karla	655512/0567
Vronský	Marek	640712/0123
Honzík	Petr	710321/1759

To se sice může zdát v pořádku, budeme-li zacházet pouze s touto tabulkou, ale co jestliže máme více tabulek a potřebujeme, aby tyto tabulky byly ve vzájemném vztahu (relaci)? (Viz dále část o relacích.) To by znamenalo v tuto chvíli změnit údaje ve všech ostatních tabulkách a pokud tyto ještě budou mít vztah dále, opět v nich a tak bychom mohli pokračovat donekonečna.

A to jsme ještě nevzali v úvahu jiný fakt. Ve složeném klíči musí zůstat všechny sloupce neměnné jakákoli změna do libovolného ze sloupců primárního klíče má za následek změnu tohoto primárního klíče a to není dovoleno.

Zřejmé řešení je, že musíme zajistit, aby se primárním klíčem stal sloupec, jehož hodnoty nebudeme potřebovat během existence řady měnit. Podíváme-li se do tabulky [Osoba] můžeme vybrat sloupec [Osoba]Rodné číslo jako primární klíč (je to časté řešení).

Zvolíme-li toto řešení můžeme narazit na jiný problém. Evidentně musí být primární klíč zadán v každé řadě tabulky (jinými slovy musí být povinný). Předpokládejme však běžnou situaci, kdy tuto tabulku budeme vytvářet záznam po záznamu např. při příjmu pacientů. Pacient v komatu bez dokladů nám asi své rodné číslo nesdělí a přesto je jeho záznam nutné pořídit. Náhle po příjmu takového pacienta se tabulka změní:

Osoba

Příjmení	Jméno	Rodné číslo
Vávra	Daniel	550613/0485
Vávrová	Karla	655512/0567
Vronský	Marek	640712/0123
Honzík	Petr	710321/1759
Krákora	Martin	

To je nepřijatelné, protože pro další návazné tabulky není možnost jak jednoznačně tuto osobu identifikovat (mimochodem rovněž jsme uvedli, že sloupec primárního klíče je povinný).

Z tohoto důvodu jsme se rozhodli vytvořit zvláštní sloupec s jedinečnými hodnotami, který bude sloužit jako primární klíč:

Osoba

Příjmení	Jméno	Rodné číslo	Pořadové číslo
Vávra	Daniel	550613/0485	1
Vávrová	Karla	655512/0567	2
Vronský	Marek	640712/0123	3
Honzík	Petr	710321/1759	4
Krákora	Martin		5

Primární klíč (jako [Osoba]Rodné číslo je-li to možné), který je přirozeným způsobem obsažen v tabulce je povinný neměnný a jednoznačně identifikuje jednu každou řadu je nazýván *přirozený klíč*. Primární klíč, který vyrábíme sami (jako [Osoba]Pořadové číslo) je nazýván *umělý klíč*.

Přirozené klíče jsou vzácné, ale existují. Jak uvidíme později, 4D obsahuje vestavěné rysy pro vytváření umělých klíčů. Ve většině tvorba vlastního umělého klíče je hledaným řešením.

Ještě jednu poznámku. Všechny sloupce v tabulce, které nejsou primárním klíčem jsou nazývány *neklíčové sloupce*. Např. ve výše uvedené tabulce [Osoba]Příjmení, [Osoba]Jméno a [Osoba]Rodné číslo jsou neklíčové sloupce.

2.4.5. Vztahy (relace)

Úzký vztah k primárním klíčům mají vztahy mezi tabulkami. Podívejme se opět na tabulku [Sklady]:

Sklady

Číslo součásti	Kód obchodu	Skladové množství
27	A	10
27	B	15
27	C	7
32	B	25
48	A	8
48	B	12

Jak bylo uvedeno výše kombinace sloupců [Sklady]Číslo součásti a [Sklady]Kód obchodu je primárním klíčem k této tabulce. Předpokládejme nyní, že si přejeme ke každé součásti vidět i popis, který je obsažen v tabulce [Součásti] a sloupci [Součásti]Popis. Vytvořením vztahu (relace) mezi tabulkami můžeme vytvořit následující formulář, který nám umožní bez zadávání popisu do tabulky [Sklady] vidět odpovídající popis součásti v každé řadě:

Formulář Sklady/Součásti

Číslo součásti	Popis z tabulky Součásti	Kód obchodu	Skladové množství
27	Nosič deštníků	A	10
27	Nosič deštníků	B	15
27	Nosič deštníků	C	7
32	Kobereček	B	25
48	Bič	A	8
48	Bič	B	12

Poznamenejme, že hodnoty pro Popis jsou taženy z tabulky [Součásti]. Jinými slovy, neukládáme popis několikanásobně v každé řadě tabulky [Sklady]. Můžeme využít toho, že hodnoty ve sloupcích [Sklady]Číslo součásti a [Součásti]Číslo součásti si odpovídají. Všimněte si, že např. hodnot pro Nosič deštníků (Číslo součásti 27) si odpovídá v tabulkách [Sklady] a [Součásti].

Tento příklad ukazuje několik věcí z modelu relačních databází:

- Model relačních databází určuje, že vazby mezi tabulkami mohou být provedeny pouze tehdy, když si odpovídají hodnoty uložené ve sloupcích různých tabulek. Ve výše uvedeném příkladu [Sklad]Číslo součásti je použito tak, aby odpovídalo [Součásti]Číslo součásti.
- Tento rys umožňuje dvěma tabulkám být ve vztahu. Vztah (relace) je jednoduše vazba mezi dvěma tabulkami, které obě obsahují ve svých sloupcích data, která si odpovídají (v tomto kurzu budeme užívat termíny vztah, vztažené a souvztažnost. Tyto termíny všechny vyjadřují vazbu mezi dvěma tabulkami jak je popsáno v této části.)
- Vztah je dvoustranný. Můžete získat řadu nebo řady tabulky jedné když jste v tabulce druhé a naopak. Např. získáte odpovídající řadu tabulky [Součásti] pro danou řadu tabulky [Sklady] a získáte odpovídající řady tabulky [Sklady] pro danou řadu tabulky [Součásti].
- Existují tři druhy vztahů (relací): skupina-skupina, jedinec-skupina nebo jedinec-jedinec.
- Vztah jedinec-skupina je znázorňován jako: 1:M (jeden k mnoha)
- Vztah skupina-skupina je znázorňován jako: M:M (mnoho k mnoha)
- Vztah jedinec-jedinec je znázorňován jako: 1:1 (jeden k jednomu)
- Tabulka na straně jedinců ve vztahu 1:M se nazývá tabulka jedinců. Tabulka na straně skupin ve vztahu 1:M se nazývá tabulka skupin.
- Vztahy M:M mohou být požadovány a navrženy ve fázi návrhu, ale musí být vytvářeny dvěma vztahy 1:M (M:1:1:M více na toto téma později).

Cizí klíč je sloupec nebo sada sloupců, který je užíván pro odpovídající hodnoty ve vztahu a není primárním klíčem této tabulky. Ve správně navržené databázi je cizí klíč vždycky v tabulce skupin ve vztahu 1:M. Rovněž odpovídající sloupec či sada sloupců v tabulce jedinců musí být primární klíč tabulky jedinců. Jestliže některá z těchto vět není pravdivá váš návrh databáze je špatný a nebude pracovat.

Podívejme se opět na náš příklad užívající tabulku [Sklady]. Vztah mezi tabulkou [Součásti] a tabulkou [Sklady] je 1:M. Znamená to, že součást může být ve skladu více než jednoho obchodního domu, ale řada v tabulce [Sklady] musí odpovídat jedné a pouze jedné řadě tabulky [Součásti] (Bič může být těžko

Inforce

současně Plivátkem). V tomto příkladě odpovídající sloupec tabulky skupin tj. [Sklady]Číslo součásti je cizím klíčem této tabulky a tohoto vztahu.

Povšimněme si, že odpovídající sloupec tabulky jedinců tj. [Součásti]Číslo součásti je primární klíč tabulky [Součásti]. Důvod, proč sloupec či odpovídající sada sloupců na straně vztahu 1:M je vždy primární klíč této tabulky je jasný: primární klíč je jediný způsob spolehlivě zjistit odpovídající řadu v tabulce jedinců. Podle definice primární klíč musí být neměnný, povinný a jedinečný. To znamená, že primární klíč, je vždy nejlepší cesta k rozlišení řady v tabulce.

Rovněž si povšimněme, že [Sklady]Číslo součásti není primárním klíčem tabulky [Sklady] (i když sloupec [Sklady]Číslo součásti se podílí na primárním klíči tabulky [Sklady] není sám o sobě primárním klíčem tabulky [Sklady]).

Jestliže odpovídající sloupec nebo sada sloupců v obou vztažených tabulkách jsou primárním klíčem svých tabulek musí být tento vztah 1:1. Viz následující příklad:

Osoba 1

Příjmení	Jméno	Rodné číslo	Pořadové číslo
Vávra	Daniel	550613/0485	1
Vávrová	Karla	655512/0567	2
Vronský	Marek	640712/0123	3
Honzík	Petr	710321/1759	4

Osoba 2

Adresa	Město	Okres	Pořadové číslo
Amforová 1896	Plzeň	Plzeň město	1
Jilemnická 42	Praha	Praha západ	2
Klášterská 1	Brno	Brno město	3
Anenská 41	Ostrava	Ostrava jih	4

Zde [Osoba 1] a [Osoba 2] jsou vztažené sloupci Pořadové číslo. Tyto sloupce jsou rovněž primárním klíčem obou těchto tabulek a protože hodnoty v obou těchto sloupcích musí být tedy jedinečné je výsledkem vztah 1:1 (jinými slovy na obou stranách vztahu může existovat pouze jedna řada).

Vztahy 1:1 není obvyklý, ale vyskytuje se. Až se dostaneme k skutečnému vytváření relační databáze uvidíte, že existují oprávněné důvody chování databáze pro vytváření vztahů 1:1. Kromě toho jsou jiné oprávněné důvody pro občasné vytváření vztahů 1:1. Více na toto téma později.

2.4.6. Shrnutí

Témata probraná v přechozích částech jsou zcela nezbytná k úplnému pochopení teorie relační databáze. Probereme si je proto ještě jednou a znovu se nad nimi zamyslíme.

- Model relační databáze je založen na uložených datech v tabulkách
- Tabulka je dvourozměrný soubor dat. V tabulkách není důležité pořadí. Sloupce a řádky mohou být přeorganizovány bez změny tabulky.
- Sloupec je jedna položka dat, která je obsažena ve všech řadách tabulky. Ideální je když, sloupec obsahuje jednu a pouze jednu položku dat o dané hodnotě (atomárně). Např. jestliže tabulka obsahuje data o osobách. Příjmení bude dobrý sloupec této tabulky. Sloupec má stejný koncept jako pole ve 4D
- Řada je jeden výskyt něčeho co je uloženo v tabulce. Např. jestliže tabulka uchovává data o osobách, pak řada odpovídá jedné osobě. Řada má ten samý koncept jako záznam ve 4D.
- Každá tabulka musí mít primární klíč. Primární klíč je sloupec nebo sada sloupců, které jednoznačně odlišují každou řadu v tabulce.
- Všechny sloupce primárního klíče musí být povinné. To znamená, že musí obsahovat hodnotu pro každou řádku tabulky.
- Kromě toho, všechny sloupce v primárním klíči musí být neměnné. To znamená, že se nemohou změnit během existence řady v tabulce.
- Sloupce v tabulce, které nejsou součástí primárního klíče se nazývají neklíčové sloupce.
- Vztahy (relace) jsou vazby mezi tabulkami založené na odpovídajících si hodnotách dat v sloupcích.
- Vztahy jsou založeny na sloupcích nebo sadách sloupců, které obsahují stejné hodnoty ve dvou vztažených tabulkách.
- Vztahy mohou být skupina k skupině (M:M) , jedinec k skupině (1:M), nebo jedinec k jedinci (1:1).
- Případný vztah M:M musí být řešen dvěma vztahy 1:M.
- Odpovídající sloupec či sada sloupců tabulky skupin vztahu 1:M je nazýván cizí klíč.
- Odpovídající klíč tabulky vztahu 1:M je vždy (ve správně navržené databázi) primární klíč této tabulky.
- Vztah v kterém oba odpovídající klíče tabulek jsou primární klíče svých tabulek je vždy vztah 1:1.

Příklad 2.2

Předpokládejme, že data jsou uložena na disku a jsou v následující struktuře (Poznámka: Je to způsob uložení dat a ne pouze tisková sestava)

Součásti

Součást číslo	Popis součásti	Kód obchodu	Skladové množství
27	Nosič deštníků		
		A	10
		B	15
		C	7
32	Kobereček		
		B	25
48	Bič		
		A	8
		B	12

Odpovídá tato tabulka relačnímu modelu? (Nápověda: Je tato struktura dvourozměrná?)

Jak by jste navrhli vyhledání součástí uložených v obchodu B? Bude to snadné?

Setříďte součásti podle skladového množství . Jde to snadno?

Je tato struktura úsporná co se týče místa na disku?

Inforce

Příklad 2.3

Uvažujme následující tabulky:

Položka

Položka číslo	Popis položky	Cena
123	Zouvák	100.00
124	Škrabka	5.00
125	Čistič oken	55.00
126	Drbátko	25.00
127	Louskáček	7.50
128	Bidýlko	87.50

Objednávka

Pořadové číslo	Položka číslo	Množství
1234	123	1
1235	126	12
1236	128	7
1237	124	65
1238	123	2
1239	128	3

Určete primární klíče obou tabulek.

Jsou zde přirozené klíče?

Jsou zde klíče umělé?

Jsou tabulky ve vztahu?

Jestliže ano, jaký je druh vztahu (1:M, 1:1, nebo M:M)?

Je zde tabulka jedinců a případně, která?

Je zde tabulka skupin a případně, která?

Jaké jsou si odpovídající sloupce, nebo sada sloupců?

Jaký sloupec, či sada sloupců je cizí klíč?

Inforce

Příklad 2.4

Nyní trochu přeorganizujme :

Položka

Položka číslo	Popis položky	Cena
123	Zouvák	100.00
124	Škrabka	5.00
125	Čistič oken	55.00
126	Drbátko	25.00
127	Louskáček	7.50
128	Bidýlko	87.50

Objednávka

Pořadové číslo	Zákazník číslo	Datum
1234	8015	12/8/98
1235	0478	4/1/98
1236	9678	5/6/98

Položky objednávky

Pořadové číslo	Položka číslo	Množství
1234	123	1
1234	126	12
1234	128	7
1235	124	65
1236	123	2
1236	128	3

Určete primární klíče všech tří tabulek.

Jsou zde přirozené klíče?

Jsou zde klíče umělé?

Jsou tabulky ve vztahu a jak?

Jestliže ano, jaký je druh vztahu (1:M, 1:1, nebo M:M)?

Je zde tabulka jedinců a případně, která?

Je zde tabulka skupin a případně, která?

Jaké jsou si odpovídající sloupce, nebo sada sloupců ve vztazích?

Která tabulka je skupin a která jedinců v jednotlivých vztazích?

Jaký sloupec, či sada sloupců je cizí klíč?

Příklad 2.5

Uvažujme následující tabulku:

Osoby

Jméno	Rodné číslo	Adresa	Kraj
Marek Král	611211/5478	Ovčí 4, Praha 8,180 00	Středočeský
Petra Nová	725210/5698	Krátká 1, Úvaly, 250 82	Středočeský
Tomáš Dlouhý	740112/4589	Jánská 7, Brno, 613 00	Jihomoravský
Roman Sýkora	531203/1287	Luční 1, Praha 3,130 00	Středočeský

Jaký je problém s touto tabulkou? (Nápověda: podívejte se na složené sloupce.)

Zkuste najít řadu osoby kde příjmení je od písmene K do Z. Můžete to vůbec provést ?

Zkuste setřídit podle příjmení. Můžete to vůbec provést ?

Zkuste vyhledat všechny osoby z Prahy. Je to možné?

Zkuste setřídit všechny řady podle PSČ

3. K čemu je dobrá relační databáze

Určitě nastane doba, kdy vám bude nabídnuta příležitost vytvořit systém a vy si nebudete jisti zda, 4D je správným nástrojem pro tuto práci. Účelem této části je pomoci vám zodpovědět tuto otázku.

Jak jsme viděli 4D je systém pro řízení relačních databází. Obecně systémy, které mohou být vytvořeny v systémech řízení relačních databází mohou být vytvořeny i ve 4D. (Zůstávají otázky týkající se chování a kapacity, které jsou mimo rámec tohoto kursu. Avšak s velkou kapacitou 4D většina dnes vytvářených relačních databází může být ve 4D napsána.) Takže se soustředíme na to, které typy systémů mohou (nebo by měly) být vytvořeny pomocí systému řízení relačních databází. Abychom tak mohli učinit uvedeme tři stará základní pravidla:

- Řady jsou levné
- Sloupce jsou drahé
- Tabulky jsou nezaplatitelné

Vysvětlení je snadné. Je velice jednoduché přidat do tabulky řady. Na druhé straně přidání sloupců do tabulky je poněkud komplikované. A přidání nových tabulek do již existující databáze můžeme přirovnat k operaci srdce.

Takže, struktura navrhovaného systému musí být popsána a vytvořena pečlivě a kompletně. (Jak popisovat a vytvářet je předmětem tohoto kursu.)

Kromě toho, musíte být schopni kompletně popsat strukturu dat v termínech relační databáze. Popsat strukturu znamená, že tato musí být stabilní (neměnná) v čase. Jestliže se libovolná část struktury dat pravděpodobně bude více či méně často měnit za běhu systému (tj. při jeho spuštění), tento systém pravděpodobně nemůže být vůbec pomocí relační databáze vytvořen. Kvalita libovolné části struktury dat, která se často mění je nazývaná nestálou. Samozřejmě mohou být části struktury dat, které se dají popsat a části, které ne. (Takže pak máme stálé části struktury dat a nestálé části struktury dat.)

Nemusíte se leknout takovéto práce. Řekli jsme, že takováto úloha nemůže být splněna pouze relační databází. Jestliže bude potřeba tabulkový procesor nebo textový editor pro uložení nestálých dat můžete použít 4D Calc nebo 4D Write.

Uveďme příklad systému, který nemůže být vytvořen pouze za pomoci relační databáze:

> Klient je velká ropná společnost, která si přeje vytvořit systém dlouhodobého finančního plánování projektů společnosti. Každý projekt je samostatná kapitola (t.j. vrtání řady ropných zdrojů,...) s konečnou dobou trvání. Chování projektu je modelováno ve finančních ukazatelích, které jsou zobrazovány v časových úsecích jednoho roku. Zodpovědné osoby z finančních ukazatelů musí vypočíst další stanovené finanční proměnné. Nakonec, jak bývá v reálném životě běžné, se předpokládá, že podmínky a stav projektu jsou vyjádřitelné různými finančními proměnnými.
>
> Klient Vám není schopen sdělit, jaké definční proměnné budou v jakémkoliv čase a projektu použity, protože svou finančně plánovací metodologii neustále mění a přidává různé finanční proměnné. Kromě toho, finanční proměnné a použité operátory se neustále mění, protože klient zkouší metodu různých pohledů na projekt, aby zjistil a modeloval možné změny.
>
> Dále není rovněž schopen říci, které periody, kromě roků budou voleny a jaká období, pro jednotlivé projekty, zvlášť protože životnost každého projektu je jiná.

Většina z toho co zde bylo popsáno, je samozřejmě typická úloha pro tabulkový procesor. Je zde však část systému, která bude stabilní a lze ji navrhnout jako relační databázi. Tabulka Projekty, například může obsahovat název projektu, region, odpovědné osoby za plánování projektu, datum začátku, atd.

Pokud se pustíme do finančních proměnných budeme mít obrovské potíže. Jedna z alternativ by byla vytvořit velmi obsáhlou tabulku s vysokým počtem sloupců rovným počtu let, které si klient myslí, že bude maximálně potřebovat, jak je ukázáno v tomto příkladu:

Finanční proměnné

Název	Rok 1	Rok 2	Rok N
Rok	1995	1996	NNNN
Investiční náklady	1 000 000	500 000	0
Provozní náklady	0	10 000	300 000
Náklady na provize	0	0	250 000

Tento způsob přístupu bude mít za následek nepřijatelné chování (čas, ergonomika) a mnoho nevyužitého místa. Kromě toho, musíte vyřešit, jak ukládat operátory a prováděné operace mezi proměnnými v jednotlivých letech, a stanovit výsledek. Protože klient může chtít provést libovolnou operaci mezi proměnnými, bude muset být váš program neúměrně složitý. A protože klient svou metodiku neustále mění, připravíte si noční můru na údržbu tohoto díla.

Ačkoliv se Vám toto zadání může zdát absurdní, bylo skutečně před nedávnem navrženo a zrealizováno u jedné z hlavních velkých ropných akciových společností.

Řešení bylo provedeno ve 4D, relační databáze byla použita pro ukládání, ale samozřejmě ne výše popsaným způsobem. Místo toho byly finanční proměnné uloženy v tabulkách 4D Calc,tabulkovém procesoru, a tyto tabulky (spreadsheets) byly uloženy v řádce v sloupci tabulky relační databáce, s použitím 4D vestavěného rysu BLOB (binární velký objekt).

Tím, že byly tabulky tabulkového procesoru přímo přiřazeny k různým projektům a bylo tak umožněno jejich okamžité vyvolání s projektem ve víceuživatelském prostředí relační databáze, dosáhl klient význačného pokroku v celém finančním oddělení a finančním plánování. Tak se nový systém stal úspěšný. Nemohl by být úspěšný, pokud by jediným nástrojem v projektu zůstal systém pro řízení relačních databází.

Je staré tvrzení, které říká: „Jestliže jediný nástroj, který vlastníte je kladivo, každý problém, který řešíte se brzy začíná podobat hřebíku.“ Je mnoho věcí pro které se relační databáze nehodí. Pořiďte si tedy odpovídající nástroj pro svou práci. S kompletním prostředím 4D, si pořídíte dodatečné schopnosti, které tradiční systémy řízení relačních databází nevyřeší. Takže 4D se zavděčí i tam, kde je nutno ukládat nestálá data, jakmile si zapamatujete tuto kapitolu

4. Formální reprezentace tabulek

V kurzu se chystáme dívat se na tabulky různými způsoby. První způsob reprezentace tabulek, který se musíme naučit je nazýván věcný seznam vlastností. Nebojte se podivných termínů. Je to jednoduše seznam všech sloupců v tabulce, který ukazuje jejich typ a jejich účast na primárních a cizích klíčích. Zde je příklad takovéhoto věcného seznamu vlastností tabulky s kterou jsme již pracovali :

Sklady

Název sloupce	Typ dat	Klíče
Součást číslo	Longint	PK1 CK1
Kód obchodu	Řetězec (2)	PK2 CK2
Množství	Real	

Názvy sloupců a datových typů jsou dostatečně pochopitelné.

Longint - celočíselné dlouhé

Řetětec (2) - písmena maximálně 2

Real - reálné číslo

Klíče vyžadují ještě objasnění :

- PK určuje, že sloupec je součástí primárního klíče. Jestliže je více než jeden sloupec zahrnut v primárním klíči (jak je tomu zde), jsou sloupce číslovány. Jinak je uvedeno pouze PK.
- CK určuje cizí klíč. Cizí klíč je vždy číslován, jak je vidět z tabulky, dokonce je-li pouze jeden (pak samozřejmě CK1). Jestliže více než jeden sloupec skládá cizí klíč jsou sloupce číslovány následovně (pro klíč 1) CK1.1, CK1.2, atd.

Příklady Nákres věcného seznamu vlastností

Příklad 4.1

Nakreslete věcný seznam vlastností pro následující tabulky :

Položka

Položka číslo	Popis položky	Cena
123	Zouvák	100.00
124	Škrabka	5.00
125	Čistič oken	55.00
126	Drbátko	25.00
127	Louskáček	7.50
128	Bidýlko	87.50

Objednávka

Pořadové číslo	Zákazník číslo	Datum
1234	8015	12/8/98
1235	0478	4/1/98
1236	9678	5/6/98

Položky objednávky

Pořadové číslo	Položka číslo	Množství
1234	123	1
1234	126	12
1234	128	7
1235	124	65
1236	123	2
1236	128	3

Jsou to tytéž tabulky jako v předchozím příkladu

Jak vám věcný seznam vlastností pomůže porozumět tabulkám ?

Jestliže tabulky byly složité, pomohl vám porozumět více?

5. Normalizace

Nyní změníme téma. Normalizace je důležitá část návrhu relační databáze. Proto je nutno dobře porozumět normalizaci. Naučení se definic a porozumění jsou občas velmi rozdílné věci. Z tohoto důvodu uvedeme množství příkladů na toto téma.

5.1. Co je normalizace ?

Normalizace je proces rozhodování, jaký sloupec bude v jaké tabulce. Jeden ze základních principů normalizace je minimalizace přebytečných nebo zdvojených dat. Jiný základní princip je, že chceme vytvořit skupiny sloupců podle vztahů a sloučit je do tabulky podle logických vztahů mezi nimi. Zdroj: Reference 2

Výsledkem je logický návrh databáze. Normalizace má vliv na návrh databáze, která pak jasně, přehledně a soudržně, konzistentně zachycuje původní data.

Výsledek normalizace se nazývá normalizovaná forma. Normalizace je proces odehrávající se krok po kroku a je dosahováno postupně různých normalizovaných forem. První, druhé atd. Každá další normalizovaná forma je logicky lepší, než předchozí. Každá další normalizovaná forma obsahuje i předchozí, takže databáze, která je v druhé normalizované fromě je současně i v první atd.

Pravidla normalizace jsou založena na běžné logice. Je velmi pravděpodobné, že je používáte přirozeně. Výhoda pravidel je v tom, že budete mít jistotu, že tomu tak skutečně je, a že jste nic nezanedbali.

Někdo se může přít, že databáze v normalizované formě není nejoptimalizovanější. Řekneme to ještě jednou. Mějte na paměti, že nyní navrhujeme a nevytváříme, neprogramujeme. Až budete mít správný, normalizovaný návrh, pak je možno přistoupit k otázkám chování budoucí databáze a upravit návrh s tímto vědomím. Tyto otázky budeme řešit v závěru kurzu.

5.2. Proč normalizovat?

Je mnoho důvodů proč normalizovat. Zde je několik z nich :

- Normalizovaný návrh databáze organizuje data podle jejich významu.
- Normalizovaný návrh databáze bude masivní a nebude vnímavý k řadám problémů. Také bude lehčeji upravitelný pokud to bude potřeba, a to proto, že normalizace redukuje přebytečná a zdvojeně pořizovaná data.
- Proces normalizace nutí návrháře plně porozumět každé položce dat. Poslední výhoda je možná nejdůležitější.

Inforce

5.3. Funkční závislost

Klíčem k porozumění normalizace je porozumění konceptu funkční závislosti. Je to jemný koncept, proto se soustřeďte.

Říkáme, že Sloupec A je funkčně závislý na Sloupci B tehdy a jen tehdy, když existuje právě jedna hodnota ve sloupci B pro každou hodnotu ve sloupci A. Zdroj: Reference 2

Praktický příklad nám pomůže. Předpokládejme databázi, která sleduje akciový trh. Tato databáze obsahuje tabulku nazvanou [Obchodované akcie]. V tabulce jsou sloupce Název a cena :

Obchodované akcie

Název	Cena
Ford	25 3/8
GM	44 1/4
Chrysler	64 5/8
IBM	100 3/4
Apple	54 7/8
Microsoft	44 1/4

Nyní můžeme říci, že cena je funkčně závislá na Názvu, protože zde musí být jedna a právě jedna hodnota pro cenu pro libovolný vybraný Název. Na druhou stranu, Název není funkčně závislý na Ceně, protože nemusí být jedna hodnota Názvu pro libovolně vybranou cenu.

Funkční závislost se vyjadřuje :

Název -> Cena

To říká, že Cena je funkčně závislá na Názvu.

Povšimněme si, že je správná možnost může být, že sloupec je funkčně závislý na sadě sloupců. Podíváme se opět na jeden z předchozích příkladů :

Sklady

Číslo součásti	Kód obchodu	Skladové množství
27	A	10
27	B	15
27	C	7
32	B	25
48	A	8
48	B	12

Sloupec množství je funkčně závislý na kombinaci sloupců Číslo součásti a Kód obchodu.

Funkční závislost je zapsána jako :

Číslo součásti + Kód obchodu -> Množství

Poznamenejme, že tyto dva sloupce jsou rovněž primárním klíčem této tabulky, to bude dobrá nápověda pro příklad, který uvedeme.

Příklad 5.1

Uvažujme následující tabulky z chovné stanice :

Sklad

Skladové číslo	Druh	Odrůda	Získána
101	Pes	Boxer	1/30/95
102	Pes	Pit Bull	2/4/95
103	Pes	Kokr	2/14/95
104	Kočka	Perská	2/17/95
105	Kočka	Manx	3/2/95
106	Kočka	Perská	3/14/95
107	Pes	Kokr	3/19/95
108	Kočka	Barmská	5/2/95

Plán krmení

Odrůda	Druh	Krmivo	Množství
Boxer	Pes	Purina Dog Chow	16 Oz.
Pit Bull	Pes	Purina Dog Chow	24 Oz.
Kokr	Pes	Purina Cat Chow	18 Oz.
Barmská	Kočka	Purina Cat Chow	10 Oz.
Perská	Kočka	Purina Dog Chow	14 Oz.
Manx	Kočka	Purina Cat Chow	10 Oz.

Určete všechny funkční závislosti.

Které sloupce jsou závislé na jednom sloupci?

Které sloupce jsou závislé na více než jednom sloupci?

5.4. První normalizovaná forma

První normalizovaná forma je snadná. Je také trochu odlišná od ostatních forem. Požaduje, aby jakákoliv tabulka, kterou jsme navrhli se podřídila správné definici tabulky.

Předpokládejme, že chceme sledovat zaměstnance a projekty, na kterých právě pracují, založíme tedy tabulku, která vypadá následovně:

Zaměstnanci

Název sloupce	Typ dat	Klíč
Číslo zaměstnance	Longint	PK
Příjmení	Řetězec (45)	
Jméno	Řetězec (45)	
Název projektu 1	Řetězec (20)	
Datum zahájení 1	Datum	
Datum ukončení 1	Datum	
Název projektu 2	String (20)	
Datum zahájení 2	Datum	
Datum ukončení 2	Datum	
Název projektu 3	Řetězec (20)	
Datum zahájení 3	Datum	
Datum ukončení 3	Datum	

Tento způsob struktury je běžný u tzv. kartičkových databází. To je také důvod proč relační databáze pracují mnohem lépe než tyto kartičkové databáze. Technický termín pro skupinu sloupců reprezentujících zde Název projektu, datum zahájení a datum ukončení je opakovaná skupina. Kartičkové databáze používají mnoho opakovaných skupin, z jednoduchého důvodu, mohou pracovat pouze s jednou tabulkou v jednom čase. Proto ani nemáte jinou možnost, pokud volíte tento typ systému řízení databáze.

Podívejme se na efekt opakované skupiny, jak ji vidíme zde. Nejdříve předpokládejme, že chcete provést vyhledání všech zaměstnanců, kteří pracují na projektu GIS. Jediný způsob jak to provést je zeptat se následujícím způsobem :

[Zaměstnanci]Název projektu 1 = "GIS" Nebo

[Zaměstnanci]Název projektu 2 = "GIS" Nebo

[Zaměstnanci]Název projektu 3 = "GIS"

To by ještě nebylo tak zlé, ale nyní předpokládejme, že existuje schopný pracovník, který pracuje na čtyřech projektech najednou. Jediný způsob je pak nutnost přidat ještě další tři sloupce pro tento projekt do struktury tabulky. Pak musí být změněny formuláře a všechny programové příkazy, kterými podporujete tento typ hledání, atd.

Opakované skupiny jsou už ze své podstaty nestálé. To je jeden z důvodů, proč je toto řešení špatné. Opakované skupiny jsou noční můrou pro údržbu databáze, protože jich pravděpodobně nebude nikdy dost.

Jiný problém. Předpokládejme, že chcete vytvořit tiskovou zprávu ukazující seznam pracovníků, pracujících na každém z projektů. Chcete je tisnout v seznamu projektů.

Můžete to provést ? Asi ano, ale se značným programátorským úsilím. A opět, jestliže přidáte další prvek do opakované skupiny, můžete začít s programováním znovu.

Takže z toho nám vyplyne první normovaná forma následovně :

Tabulka je v první normalizované formě, jestliže nemá opakované skupiny. Zdroj: Reference 2

(To není legrace, to je skutečně formální definice první normalizované formy. Takže, to nebylo tak zlé, což?)

První normalizovaná froma se zapisuje jako: 1NF.

Příklad První normalizovaná forma

Příklad 5.2

Váš klient vás požádal o návrh databáze, která bude používána pro sledování záznamů o pacientech u lékaře. Dodal vám typický zdravotnický formulář, který chce jednoduše převést na elektronické ukládání. Tento záznam pacienta má následující papírový formulář (přepsali jsme jej pro vás do sloupců a formalizovali).

Záznam o osobní návštěvě

Název sloupce	Typ dat	Klíč
ID pacienta	Longint	PK
Příjmení pacienta	Řetězec (45)	
Jméno pacienta	Řetězec(45)	
Dohodnutá návštěva 1	Datum	
Dohodnutá návštěva 1	Čas	
Dohodnutá návštěva 2	Datum	
Dohodnutá návštěva 2	Čas	
Dohodnutá návštěva 3	Datum	
Dohodnutá návštěva 3	Čas	
Dohodnutá návštěva 4	Datum	
Dohodnutá návštěva 4	Čas	
Poslední návštěva	Datum	
Příznaky 1	Řetězec(80)	
Příznaky 2	Řetězec(80)	
Příznaky 3	Řetězec(80)	
Příznaky 4	Řetězec(80)	
Příznaky 5	Řetězec(80)	
Diagnóza 1	Řetězec(80)	
Diagnóza 2	Řetězec(80)	
Diagnóza 3	Řetězec(80)	

Inforce

Diagnóza 4	Řetězec(80)
Diagnóza 5	Řetězec(80)
Spalničky	Logické
Diagnostikováno dne	Datum
Plané neštovice	Logické
Diagnostikováno dne	Datum
DPT Shot	Logické
Diagnostikováno dne	Datum
Vakcinace vztekliny	Logické
Provedeno dne	Datum
Protichřipkové očkování	Logické
Provedeno dne	Datum
Příušnice	Logické
Diagnostikováno dne	Datum
Astma	Logické
Diagnostikováno dne	Datum
Pylová alergie	Logické
Diagnostikováno dne	Datum

Určete všechny opakované skupiny.

Navrhněte tabulku/ tabulky tak aby byli v 1NF.

Nyní vysvětlete vaší klientce, proč je její úmysl automatizovat záznamy pacientů v této formě nevhodný a proč Váš normalizovaný návrh databáze je lepší.

Příklad 5.3

Váš vedoucí plánovacího oddělení by chtěl být schopen vyjadřovat chování společnosti v grafech v periodách měsíčně, čtvrtletně a ročně. Je důležité, aby si mohl ukládat celkové prodejní ukazatele během těchto period a roční průběžný průměrný prodej (pouze měsíčně a čtvrtletně). Je to ekonom a je obeznámen dobře s tabulkovými editory. Proto navrhl následující strukturu.

Měsíční data

Název sloupce	Typ dat	Klíče
Rok	Integer	PK
Leden celkem	Real	
Roční průměr k lednu	Real	
Únor celkem	Real	
Roční průměr k únoru	Real	
Březen	Real	
Roční průměr k březnu	Real	
Duben	Real	
Roční průměr k dubnu	Real	
Květen	Real	
Roční průměr k květnu	Real	
Červen	Real	
Roční průměr k červnu	Real	
Červenec	Real	
Roční průměr k červenci	Real	
Srpen	Real	
Roční průměr k srpnu	Real	
Září	Real	
Roční průměr k září	Real	
Říjen	Real	
Roční průměr k říjnu	Real	
Listopad	Real	
Roční průměr k listopadu	Real	
Prosinec	Real	
Roční průměr k prosinci	Real	

Čtvrtletní data

Název sloupce	Typ dat	Klíče
Rok	Integer	PK
Čtvrtletí 1 celkem	Real	
Průměr k čtvrtletí 1	Real	
Čtvrtletí 2 celkem	Real	
Průměr k čtvrtletí 2	Real	
Čtvrtletí 3 celkem	Real	
Průměr k čtvrtletí 3	Real	
Čtvrtletí 4 celkem	Real	
Průměr k čtvrtletí 4	Real	

Roční data

Název sloupce	Typ dat	Klíče
Rok	Integer	PK
Roční celkový prodej	Real	

Myslí si samozřejmě, že je skutečně skvělá struktura dat a bude též skvěle pracovat. (Samozřejmě, vždyť neabsolvoval tento kurz!)

Vytvořte strukturu, která bude ve formě 1NF, a prodejte mu tuto myšlenku.

5.5. Druhá normalizovaná forma

Druhá normalizovaná forma obsahuje pojem zda jsou či nejsou všechny sloupce v tabulce funkčně závislé na primárním klíči tabulky. Podívejme se na příklad dat:

Č. pracovníka	Jméno	ID projektu	Název projektu	Datum zahájení	Datum ukončení
123	Jiří	XYZ	Vesmírný hrnec	7/7/91	8/8/91
123	Jiří	ZYX	Motorový vlak	8/9/91	12/24/91
231	Marie	XYZ	Vesmírný hrnec	7/12/91	12/24/91
231	Marie	XYX	Parní traktor	12/26/91	3/18/92
312	Láďa	ZYX	Motorový vlak	7/27/91	3/3/92

Nyní předpokládejme, že jsme rozeznali následující funkční závislosti :

Č. pracovníka -> Jméno

ID projektu -> Název projektu

Č. pracovníka + ID projektu -> Datum zahájení

Č. pracovníka + ID projektu -> Datum ukončení

Podívejme se na data. Uložili jsme zde údaje o tom, že pracovník 123 je Jiří dvakrát. Rovněž údaj, že projekt XYZ je Vesmírný hrnec je zde pořízen dvakrát. Jestliže budeme chtít tento údaj změnit musíme změnit všechny jejich výskyty. Data nebudou souhlasná jestli tak neučiníme.

Podobně tak může být pokud vznikne nový projekt a nikdo na něm ještě nepracuje pak musíme ponechat sloupce pro pracovníka prázdná a přijmeme-li nového pracovníka a ten ještě nepracuje na žádném projektu musí zůstat údaje o projektu prázdné.

Pokud se rozhodneme vymazat data o projektu můžeme vymazat i poslední výskyt záznamů o pracovníkovi a naopak při mazání údajů o projektu můžeme vymazat i data o pracovníkovi.

Jak můžeme řešit tento problém ? Nejdříve si povšimněme, že žádný ze sloupců není ve skutečnosti primární klíč. Musíme být schopni dle primárního klíče jednoznačně identifikovat řadu v tabulce.

Podíváme-li se na funkční závislosti, můžeme rovněž vidět, že zacházíme s třemi různými typy vztahů:

- Jméno je funkčně závislé na Č.pracovníka
  (Č.pracovníka -> Jméno).
- Název projektu je funkčně závislý na ID projektu
  (ID projektu -> Název projektu).
- Datum zahájení a Datum ukončení jsou funkčně závislé na kombinaci Č.pracovníka a ID projektu
  (Č.pracovníka + ID projektu -> Datum zahájení a
  Č.pracovníka + ID projektu -> Datum ukončení).

To znamená, že jsme zkombinovali několik rozdílných věcí či vztahů do jedné tabulky. Tyto problémy můžeme vyřešit pokud splníme pravidla pro druhou normalizovanou formu :

> Tabulka je v druhé normalizované formě, jestliže je již v první normalizované formě 1NF a všechny neklíčové sloupce jsou funkčně závislé na primárním klíči tabulky. Zdroj: Reference 2

Pokud použijeme tuto poučku na naše data dostaneme následující strukturu tabulek :

Pracovníci

Název sloupce	Typ dat	Klíče
Č.pracovníka	Longint	PK
Jméno	Řetězec (45)	

Projekty

Název sloupce	Typ dat	Klíče
ID projektu	Longint	PK
Název projektu	Řetězec(45)	

Přiřazení

Název sloupce	Typ dat	Klíče
Č.pracovníka	Longint	PK1 CK1
ID projektu	Longint	PK2 CK2
Datum zahájení	Datum	
Datum ukončení	Datum	

S následujícími daty příkladu:

Pracovníci

Č.pracovníka	Název
123	Jiří
231	Marie
312	Láďa

Projekt

ID projektu	Název projektu
XYZ	Vesmírný hrnec
ZYX	Motorový vlak
XYX	Parní traktor

Přiřazení

Č. pracovníka	ID projektu	Datum zahájení	Datum ukončení
123	XYZ	7/7/91	8/8/91
123	ZYX	8/9/91	12/24/91
231	XYZ	7/12/91	12/24/91
231	XYX	12/26/91	3/18/92
312	ZYX	7/27/91	3/3/92

Na první pohled to vypadá, jakoby jste zavedli zdvojenost pořizování do dat, opak je pravdou. Každý druh dat je pořizován a uložen pouze jednou. Souhlasné hodnoty z různých tabulek je způsob provázání těchto tabulek vztahem. Problémy, keré jsme zmínili v části výše jsme plně odstranili.

Druhá normalizovaná forma je zapisována jako : 2NF

Příklady Druhá normalizovaná forma

Příklad 5.4

Pojišťovací společnost má databázový systém, který chce přehodnotit. Hlavní tabulka systému se nazývá [Politika, a má následující strukturu.

Politika

Název sloupce	Typ dat	Klíče
ID Dopravce	Longint	PK1 CK1
Název dopravce	Řetězec (45)	
Politika označení	Řetězec(10)	PK2
ID zákazníka	Longint	CK2
Příjmení zákazníka	Řetězec(45)	
Jméno zákazníka	Řetězec(45)	

Identifikujte všechny funkční závislosti v této tabulce.

Vytvořte sadu tabulek, které vyhoví druhé normalizované formě 2NF. Napište věcné seznamy vlastností, a identifikujte všechny primární a cizí klíče.

5.6. Třetí normalizovaná forma

Třetí normalizovaná forma je velmi podobná 2NF. Ve skutečnosti jste pravděpodobně normalizovali příklad výše do třetí normalizované formy aniž by jste to věděli. Pravidlo tvrdí:

> Tabulka je v třetí normalizované formě jestliže je již v 2NF a žádný neklíčový sloupec není funkčně závislý na jiném neklíčovém sloupci.

Toto pravidlo využijeme v situacích podobných následující:

Zaměstnanci

Č.zaměstnance	Středisko	Budova
26622	Prodejní	Západní
41156	Účetnní	Centrální
33897	Výzkum	Severní laboratoř
88644	Prodejní	Západní
91214	Marketing	Centrální

Tato tabulka je v 1NF. Nemá žádné složené klíče, protože ID zaměstnance jednoznačně identifikuje každou řadu v tabulce. Proto je automaticky v 2NF. (Pamatujeme si, že 2NF musí zacházet se sloupci funkčně závislými na primárním klíči. Zde je primární klíč pouze jeden sloupec)

Pravidlo pro třetí normalizovanou formu vás žádá vyhledat druhořadé funkční závislosti. To je situace, kdy jeden sloupec je funkčně závislý na jiném, který je fukčně rovněž závislý na dalším sloupci. Tento druh závislosti se nazývá přenosná závislost.

Ptáme se tedy jestli je zde funkční závislost mezi neklíčovými sloupci Středisko a Budova. Ano je zde závislost, protože vnitřní politika firmy umisťuje pracovníky jednoho střediska firmy do jedné budovy. To znamená, že známe-li středisko, automaticky známe i budovu.

V tomto případě závislost mezi Budovou a Č.zaměstnance je druhořadá. Tato závislost existuje, protože známe-li číslo zaměstnance určíme ze střediska i budovu a je nepřímá a přenosná zapisujeme ji:

Č.zaměstnance -> Středisko -> Budova

Jestliže existuje přenosná závislost, tabulka není v třetí normalizované formě a měla by být rozdělena do dvou tabulek:

Zaměstnanci

Název sloupce	Typ dat	Klíče
Č.zaměstnance	Longint	PK
Středisko	Řetězec (20)	CK1

Střediska

Název sloupce	Typ dat	Klíče
Středisko	Řetězec (20)	PK
Budova	Řetězec (20)	

Jestliže však na druhou stranu zaměstanci jsou umístěni kdekoliv, kde je právě místo, není zde závislost mezi střediskem a budovou, ale budovy jsou závislé pouze na zaměstnancích a původní databáze je již v třetí normalizované formě.

Mimochodem je ještě jiná záležitost týkající se odvozování předpokladů z dat. V tomoto případě, pracovníci všech středisek pracují v jedné budově. Jestliže není nikde v politice firmy prohlášeno, že pracovníci nemohou pracovat v jiné budově, je lepší, když zůstanete u návrhu s jednou tabulkou, jak vypadala původně. Buďte opatrní, v odvozování nějakých předpokladů z dat. To co odvozujete z dat může být pravda, ale musíte prověřovat předpoklady správně cílenými otázkami na klienta, aby jste si tyto předpoklady ověřili.

Podívejme se na jiný příklad, aby jsme si utvrdili co jsme se již naučili.

Uvažujme následující strukturu tabulky pro muzeum umění:

Díla

Název sloupce	Typ dat	Klíče
ID díla	Longint	PK
Název	Řetězec (45)	
Příjmení autora	Řetězec (45)	
Jméno autora	Řetězec (45)	
Datum narození autora	Datum	

S následujícími příkladovými daty:

Díla

ID díla	Název	Jméno autora	Příjmení autora	Př.datum nar.
001	Mona Lisa	Leonardo	DaVinci	5.12.1697
002	Madona s dítětem	Raphael		9.2.1725
003	Portrét žebráka	James	Whiton	4.12.1892
004	Zlatý věk	Frederick	Remington	31.5.1974
005	Počátky	James	Whiton	4.12.1892

Nejsou zde opakované skupiny, tabulka je tedy v 1NF.

ID díla je jediný sloupec primárního klíče, tabulka je tedy v 2NF.

Podíváme-li se na neklíčové sloupce, povšimneme si, že datum narození autora je funkčně závislé na kombinaci sloupců Příjmení autora a Jméno autora. Tak jsme objevili přenosnou závislost, kerou můžeme zapsat jako:

ID díla -> Příjmení autora + Jméno autora -> Datum narození autora

Tabulka proto není v třetí normalizované formě, a je třeba ji rozdělit do dvou tabulek. (V struktuře, která následuje , jsme rovněž řešili, že v datech tabulky není žádný přirozený klíč pro tuto tabulku)

Díla

Název sloupce	Typ dat	Klíče
ID díla	Longint	PK
Název	Řetězec (45)	
ID autora	Longint	CK1

Autoři

Název sloupce	Typ dat	Klíče
ID autora	Longint	PK
Jméno autora	Řetězec(45)	
Příjmení autora	Řetězec(45)	
Datum narození	Datum	

A zde jsou příkladová data:

Díla

ID Díla	Název	ID autora
001	Mona Lisa	1001
002	Madona s dítětem	1002
003	Portrét žebráka	1003
004	Zlatý věk	1004
005	Počátky	1003

Autoři

ID autora	Jméno autora	Příjmení autora	Datum nar.
1001	Leonardo	DaVinci	5.12.1697
1002	Raphael		9.2.1725
1003	James	Whiton	4.12.1892
1004	Frederick	Remington	31.5.1974

Tyto tabulky jsou nyní v třetí normalizované formě.

Třetí normalizovanou formu zapisujeme jako: 3NF.

Příklad Třetí normalizovaná forma

Příklad 5.5

Uvažujte následující strukturu:

Zaměstnanci

Název sloupce	Typ dat	Klíče
ID zaměstnance	Longint	PK
Jméno zaměstnance	String (45)	
ID Střediska	String (3)	
Název střediska	String (20)	

Je tabulka v 1NF?

Je tabulka v 2NF?

Je tabulka v 3NF? (Nápověda: identifikujte funkční závislosti. Jsou zde neklíčové sloupce funkčně závislé na jiných neklíčových sloupcích ?)

Jak můžete vyřešit libovolný zde existující problém ? Vytvořte nezbytné tabulky a napište seznam vlastností pro každou z nich.

5.7. Další normalizace

Existuje pět až šest normalizovaných forem, podle toho, kterou knihu budete číst Normalizované formy nad 3NF se zaobírají něčím co se trochu podobá otázce, Kolik andělů se vejde na špičku jehly.

Pravděpodobně se vůbec nedostanete do situace, kdy budete nuceni normalizovat nad 3NF. Jestliže se stanete profesionály v normalizaci do 3NF, budete odvádět dobrou práci.

5.8. Shrnutí

Projděme vše co jsme se naučili v této kapitole :

- Normalizace je proces, kde se rozhoduje, který sloupec patří do které tabulky.
- Výsledkem normalizace je logický návrh databáze, který jasně a souhlasně zachytí stav původních/přirozených dat.
- Normalizace je založena na konceptu funkčních závislostí, což znamená zda může být hodnota ve sloupci určena na základě hodnoty v jiném sloupci a zda s ní přímo souvisí.
- Funkční závislost se zapisuje:
  Název akcie -> Cena akcie.
  Zápis znamená, že cena akcie je funkčně závislá na názvu akcie (nemá bez ní smysl).
- Tabulka je v první normální formě, jestliže neobsahuje opakované skupiny hodnot. Zapisuje se jako 1NF.
- Tabulka je v druhé normalizované formě, jestiže je již v 1NF, a každý neklíčový sloupec je funkčně závislý na klíčovém sloupci. Zapisuje se jako 2NF.
- Tabulka je v třetí normalizované formě, jestliže je již v 2NF a žádný neklíčový sloupec není funkčně závislý na jiném neklíčovém sloupci. Zapisuje se jako 3NF.
- Jiná definice 3NF je, že v 3NF nejsou dovoleny přenosné závislosti. Přenosná závislost je nepřímá závislost, což znamená, že sloupec je funkčně závislý na druhém sloupci, který je sám závislý na třetím sloupci.

Příklady Normalizace

Příklad 5.6

Předpokládejme následující funkční závislost :

ID včely -> Jméno včely

ID včely ->ID úlu (včela sídlí v nějakém úlu)

ID úlu -> Kapacita úlu

ID úlu -> Telefon do úlu

ID květiny -> Typ květiny

ID květiny -> Barva

ID osoby -> Jméno osoby

ID včely + ID osoby -> Počet žihadel

ID včely + Datum opylení + Čas opylení -> ID květiny

Ignorujte všechny složené slouce v tomto příkladu. Problém se zjednoduší. Předpokládejme zde, že včela může bodnout více než jednou a libovolný počet lidí, a že včela může opylit libovolný počet květin.

Inforce

Podobně, že libovolná osoba může být bodnuta od libovolné včely a květina může být opylena více včelami.

Je následující tabulka normalizována ?

Včela

Název sloupce	Typ dat	Klíče
ID včely	Longint	PK
Jméno včely	Řetězec (20)	
ID úlu	Longint	CK1
Kapacita úlu	Longint	

Jesliže ne, v které formě je tabulka a co bylo zanedbáno. Proveďte a okomentujte správnou normalizaci.

Je následující tabulka normalizována ?

Žihadla

Název sloupce	Typ dat	Klíče
ID včely	Longint	PK1
ID osoby	Longint	PK2
Počet žihadel	Integer	
Jméno osoby	Řetězec(45)	

Jestliže ne, která je porušena ? Proveďte a vysvětlete správnou normalizaci.

Vytvořte úplnou sadu normalizovaných tabulek pro scénář zmíněný výše. Podrobně vysvětlete, proč tabulky a vztahy jsou normalizované.

Příklad 5.7

Identifikujte všechny funkční závislosti a vytvořte úplnou sadu normalizovaných tabulek pro následující scénář. Nakreslete věcný seznam vlastností pro každou tabulku. Podrobně vysvětlete proč je váš návrh normalizovaný.

Vepři jsou pěstováni ve vepřínech a starají se o ně ošetřovatelé. Každý ošetřovatel má své jedinečné ID (ID ošetřovatele), jméno a velikost košile (Velikost košile). Ošetřovatel může ošetřovat ve více vepřínech. Každý vepřín je ošetřován pouze jedním ošetřovatelem. Každý vepřín je identifikován jedinečným ID (ID vepřína). Každý vepřín má své ocenění (Hodnota vepřína) a maximální kapacitu (Max.kapacita vepřína).

Daný vepř žije pouze v jednom vepříně. Každému vepři je přiřazeno jednoznačné ID (ID vepře), váhu (Váha vepře) a ocenění (Ocenění vepře)

Příklad 5.8

Identifikujte všechny funkční závislosti a vytvořte úplnou sadu normalizovaných tabulek pro následující scénář. Nakreslete věcný seznam vlastností pro každou tabulku. Podrobně vysvětlete proč je váš návrh normalizovaný.

Iluzionisti jsou odlišeni svým ID (ID iluzionisty), mají své jméno (Jméno iluzionisty) a denní sazbu, kterou kasírují za své představení (Sazba iluzionisty).

Iluzionisti jsou sdruženi do lóží. Iluzionista může patřit do více než jedné lóže a lože se skládá z více než jednoho iluzionisty.

Lóže je identifikována svým jednoznačným názvem (Název lóže). Buhužel však čas od času dochází k změnám názvu lóží. Každá lóže má svou poštovní adresu (Adresa lóže) a telefonní číslo (Telefon lóže).

Kouzla, která může iluzionista předvést jsou popsána v katalogu kouzel. Každé kouzlo je označeno svým jednoznačným ID (ID kouzla) a popisným názvem (Popis kouzla), stupeň ohromení publika, který kouzlo vyvolává je známkován a je zde uveden (Stupeň ohromení).

Většina iluzionistů je schopna provést více než jedno katalogové kouzlo. Několik iluzionistů je schopno provést totéž kouzlo. Každý iluzionista je proto hodnocen podle zkušenosti na jaké úrovni je schopen provést dané prováděné kouzlo (Úroveň provedení)

Příklad 5.9

Identifikujte všechny funkční závislosti a vytvořte úplnou sadu normalizovaných tabulek pro následující scénář. Nakreslete věcný seznam vlastností pro každou tabulku. Podrobně vysvětlete proč je váš návrh normalizovaný.

Koně jsou cvičeni trenéry a žijí ve stájích.

Každý trenér má svou identifikaci (ID trenéra), své jméno (Jméno trenéra) a velikost bot (Velikost trenéra).

Daný kůň žije pouze v jedné stáji. Každý kůň je odlišen svým identifikačním číslem (ID koně) a má svou váhu (Váha koně) a ocenění (Ocenění koně).

Trenér může trénovat více stájí. O každou stáj se stará pouze jeden trenér. Každá stáj je identifikována jedinečným identifikačním číslem (ID stáje). Každá stáj má svou kapacitu (Kapacita stáje) a své ocenění úrovně stáda (Ocenění stáje). Daný trenér se stará o všechny stáje, které vlastní. Takže můžeme říci který trenér trénuje koně, víme-li, kde je kůň ustájen.

Příklad 5.10

Identifikujte všechny funkční závislosti a vytvořte úplnou sadu normalizovaných tabulek pro následující scénář. Nakreslete věcný seznam vlastností pro každou tabulku. Podrobně vysvětlete proč je váš návrh normalizovaný.

Řemeslníci jsou odlišeni svým identifikačním číslem (ID řemeslníka). Mají svá jména (Jméno řemeslníka) a denní sazbu ze svou práci (Sazba řemeslníka).

Řemeslníci jsou sdruženi do party. Daný řemeslník může příslušet do více než jedné party. Parta je složena z více než jednoho řemeslníka. Každá parta je identifikována svým jednoznačným názvem, bohužel party čas od času mění svá jména se změnou vedoucího. Každá parta má svou adresu (Adresa party) a telefon (Telefon party).

Řemesla, která řemeslníci vykonávají jsou popsána v katalogu. Každé řemeslo má své jednoznačné číslo (ID řemesla) a popisný název (Popis řemesla) a ocenění obtížnosti řemesla (Ocenění řemesla). Většina řemeslníků je schopna vykonávat více než jedno řemeslo. Některá řemesla může vykonávat více řemeslníků. Každý řemeslník je v daném řemesle oceněn známkou jeho zručnosti (Řemeslná zručnost).

6. Modelování databáze

Nyní se budeme zabývat otázkou modelování dat. V této kapitole se chystáme vytvářet jednu z nejdůležitějších částí návrhu databáze: věcný relační diagram (VRD).

6.1. Co je věcný relační diagram?

Jednoduše řečeno věcný relační diagram je obrázek, kerý znázorňuje tabulky, které jsou zahrnuty v systému a vztahy mezi nimi. Zdroj:Reference 2

V terminologii relačních databází je to soubor subjektů do kterých se chystáme ukládat data. Pro účely tohoto kurzu považujme subjekt za tabulku. (Navzdory červeně psaným varováním v některých učebnicích jak může končit práce zaměňujeme-li je (subjekt(věc)/tabulku).)

Podobně vztah (relace) je spojení nebo funkční přiřazení mezi subjekty.

A nakonec vlastnost je charakteristika zájmu o subjekt. Pro naše účely stačí, že vlastnost a sloupec jsou jedno a totéž.

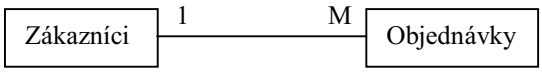
6.2. Proč diagram?

Máme přinejmenším tři dobré důvody se jím zabývat.

- Není možné úspěšně navrhnout systém bez kompletního chápání zahrnutých subjektů
- Zatímco normalizace je velmi užitečná technika, není vždy jednoduché identifikovat zahrnuté funkční závislosti. Proč, ptáme se, někdo rozhodl, že barva zboží je funkčně závislá na pořadovém čísle zboží. Je tak proto, že jsme je myšlenkově přiřadili obě jakémusi subjektu Zboží ? Proč to nepřipustit a nezabývat se touto myšlenkou formálně. Až tak učiníme uvidíme, že se funkční závislosti stanou zřetelnější.
- Pochopení, uchopení fyzického návrhu databáze za pomoci VRD diagramu je lehčí, než prohlížení si tabulek v seznamu vlastností, jak se i přesvědčíme.

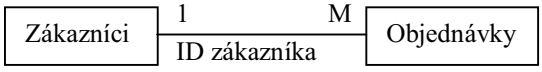
6.3. Technika

Bylo navrženo mnoho způsobů vytváření a reprezentace diagramů. Je možné, že jeden je lepší, jiný horší. Náš způsob je znázorněn na následujícím diagramu:



Vztah je zobrazen jako čára. 1 určuje, který konec čáry je připojen k subjektu jedinců a M určuje, který konec čáry je připojen k subjektu skupin.

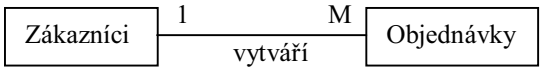
Zdokumentovat, která vlastnost nebo vlastnosti jsou užívány pro vztah, znamená použít jejich názvy, jak je vidět níže:



Je to však skutečně nezbytné. Jak uvidíme v kapitole 4 nemáme žádný jiný rozumný výběr pro vytvoření relací než tento. Na straně jedinců primární klíč a na straně skupin cizí klíč (který se skládá z toho samého sloupce nebo kombinace sloupců jako primární klíč na straně jedinců). Takže v diagramu

napíšeme pouze název či názvy vlastností a to jen v tom případě, kdy může vzniknout omyl (existuje dvojsmyslnost).

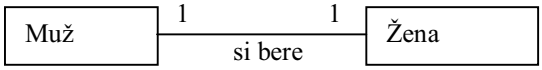
Existuje lepší způsob jak dokumentovat vztah. Protože vztah reprezentuje spojení mezi dvěma subjekty, není asi lepší vyjádření než sloveso. V naší situaci [Zákazníci] a [Objednávky] můžeme označit vztah následovně:



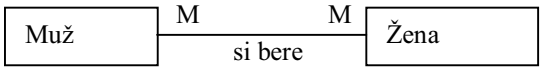
Tento diagram říká „Zákazník vytváří objednávky“ a toto prohlášení je natolik jednoduché a jasné jak jen dokumentace může být.

6.4. Proč subjekty a vztahy?

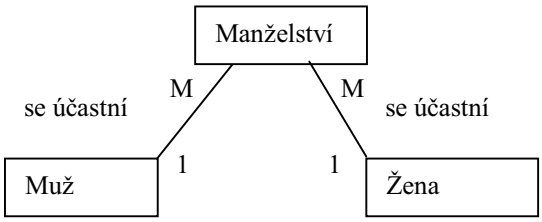
Pro ilustraci této techniky se podívejme na některé skutečné problémy. Uvažujme následující diagram ilustrující vztah mezi mužem a ženou v manželství:



V Evropské kultuře si jeden muž bere současně pouze jednu ženu (není tomu tak ve všech kulturách). Také se předpokládá že se berou pouze jedinci opačného pohlaví. Předpokládejme, že modelujeme vztah pouze dvou subjektů [Muž] a [Žena]. Zjistíme však, že i naše kultura umožňuje mnohoženství a mnohomužství (ovšem následně v čase). Takže předchozí diagram 1:1 není rozhodně platný a musíme vytvořit jiný následující typ vztahu a diagramu:



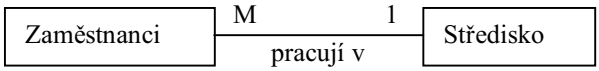
Z pohledu fyzického návrhu víme, že je možné vytvořit databázi s relací M:M nyní musíme proto vyhovět reálným systémům řízení databáze a rozdělit tento vztah do dvou 1:M vztahům. V následujícím diagramu je to provedeno vazbou subjektů na třetí vazební subjekt:



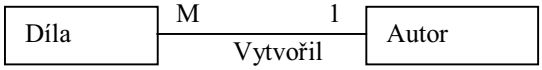
Prvním krokem při návrhu věcného relačního diagramu je rozeznat „věci“, které budou v naší databázi ukládány. Název subjektu bude téměř vždy podstatné jméno. Každý subjekt bude mít určitý typ identifikátoru (primární klíč) plus další vlastnosti (tj. sloupce). Určitá hodnota identifikátoru nás musí dovézt k jednomu a pouze jednomu výskytu subjektu.

6.5. Příklady

Znázorněme si v diagramu některé příklady použité dříve v tomto kurzu. V následujícím je uveden diagram pro první strukturu diskutovanou v sekci 5.6. ilustrující použití 3NF:

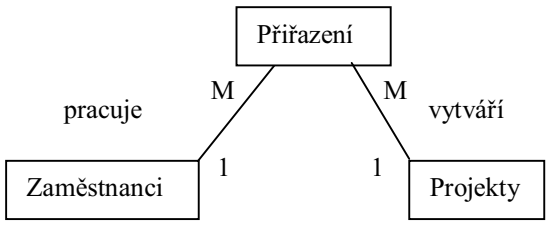


A zde je diagram pro druhou strukturu diskutovanou v sekci 5.6. rovněž ilustrující použití 3NF:



Tyto diagramy jsou přirozené a zřejmé.

Toto je diagram pro příklad užitý v sekci 5.5. ilustrující použití 2NF:



Tento diagram může vyvolat drobný odpor. [Zaměstnanci] a [Projekty], jsou určitě objekty z reálného světa, které určitě chceme sledovat. Ne zcela jasný bude asi subjekt [Přiřazení]. Vztahy mezi těmito subjekty jsou jasné. Je to rovněž dobrý příklad vztahu skupiny k skupině. Tabulka přiřazení zde slouží pro vytvoření vazby mezi tabulkami [Zaměstnanci] a [Projekty].

6.6. Shrnutí

Před tím než budeme pokračovat si shrňme koncepci věcného relačního diagramu.

- Věc (subjekt) je skutečná osoba, místo, věc nebo koncept, který systém zamýšlí sledovat. Pro naše účely považujeme subjekt a tabulku za totéž.
- Vlastnost je něco o subjektu co si přejeme uložit (sledovat). Opět považujeme vlastnost a sloupec za dvě strany téže mince.
- Vztah je přiřazení mezi subjekty nebo do subjektů. Vztah může být většinou označen slovesem.
- Věcný relační diagram je obrázek věcí z reálného světa, které náš systém zamýšlí sledovat a vztahů mezi nimi.

Příklady Vytváření věcných relačních diagramů (VRD)

Příklad 6.1

Vraťte se k příkladu 5.1. a vytvořte VRD představující strukturu použitou v příkladu.

Příklad 6.2

Vraťte se k příkladu 5.2 a vytvořte VRD představující normalizovanou strukturu, vytvořenou v příkladu.

Příklad 6.3

Vraťte se k příkladu 5.4. a vytvořte VRD představující normalizovanou strukturu vytvořenou v příkladu.

Příklad 6.5

Vraťte se k příkladu 5.6. a vytvořte VRD představující normalizovanou strukturu vytvořenou v příkladu.

Příklad 6.6

Vraťte se k příkladu 5.7. a vytvořte VRD představující normalizovanou strukturu vytvořenou v příkladu.

Příklad 6.7

Vraťte se k příkladu 5.8. a vytvořte VRD představující normalizovanou strukturu vytvořenou v příkladu.

Příklad 6.8

Vraťte se k příkladu 5.9. a vytvořte VRD představující normalizovanou strukturu vytvořenou v příkladu.

Příklad 6.9

Vraťte se k příkladu 5.10. a vytvořte VRD představující normalizovanou strukturu vytvořenou v příkladu.

Příklad 6.10

Uvažujte následující scénář.

Tato databáze bude užívána Centrální kartotékou pirátů umístěnou v Bermudském trojúhelníku (nepochybujte, že použitá 4D bude neregistrovaná pirátská kopie!).

Piráti se identifikují jedinečnými přezdívkami vydávanými Centrální kartotékou pirátů.

Pirátům není povoleno nikdy změnit svou přezdívku. Navíce každý pirát má své křestní jméno, stupeň hrůzostrašnosti a datum prvního lupu.

Pirátské lodi se rovněž rozlišují jedinečnými názvy, které jsou rovněž vydávány Centrální kartotékou pirátů. Důležité informace o pirátské lodi zahrnují počet stožárů, počet kanónů a datum spuštění.

Obchodní lodi jsou identifikovány jedinečným názvem obchodní lodi, vydávaným Úřadem pro potencionální oběti v Londýně, Anglie. Důležité charakteristiky obchodní lodi zahrnují váhu a počet stěžnů. Centrální kartotéka pirátů sleduje obchodní lodě, které byly skutečně vyloupeny nebo potopeny

Daný pirát je členem posádky vždy jedné a téže pirátské lodi. Pirát může být rovněž nezaměstnaný. Jeden a pouze právě jeden pirát je kapitán jedné a pouze jedné pirátské lodi (předpokládáme, že historie minulých zaměstnání není důležitá)

Centrální kartotéka pirátů sleduje každé přepadení. Záznam obsahuje pirátskou loď, obchodní loď a datum přepadení, zda obchodní loď byla či nebyla potopena a kolik nevinných námořníků bylo hozeno přes palubu. Jedna pirátská loď může přepadnout tutéž obchodní loď více než jednou.

Určete všechny funkční závislosti.

Navrhněte normalizovanou strukturu databáze.

Vytvořte úplnný věcný seznam vlastností pro všechny tabulky struktury.

Vytvořte věcný relační diagram pro strukturu.

7. Zjištění subjektů, vlastností a vztahů

Tato kapitola obsahuje jednoduchou kuchařku pro určení subjektů, vlastností a vztahů obsažených v systému.

7.1. Proces zpovídání zákazníka

Zjištění subjektů předpokládá rozhovor se zákazníkem a otázky typu „ Jaké druhy „věcí má systém sledovat?“. Odpověd na tuto otázku bude jistě obsahovat hodně podstatných a přídavných jmen.

Obecně podstatná jména budou subjekty a vlastnosti a přídavná jména budou (nebo naznačí) vlastnosti.

Podíváme-li se na náš přechozí příklad, položíme-li tuto otázku našemu zákazníkovi odpoví nám: „ Potřebujeme sledovat muže a ženy a jejich manželské svazky.“ (Toto je vyjímečně jasná odpověď a normálně to nebývá tak jednoduché.). V této větě máme tři podstatná jména, takže můžeme předpokládat tři kandidáty na subjekty [Muži], [Ženy] a [Manželské svazky].

Někdy je těžké říci zda něco bude subjektem nebo vlastností, můžeme tak rozhodnout pouze budeme-li uvažovat ještě o účelu návrhu. Považujeme něco za subjekt jestliže je to něčím kam chceme ukládat data. Také jestliže něco má vlastnosti je to subjekt. Vlastnosti nemohou mít vlastnosti.

Jakmile jsme přestali nahrávat a analyzovat odpovědi klienta, položíme následující otázku: „Co na tomto subjektu chcete sledovat?“. Opět dostanete odpověď se spoustou podstatných a přídavných jmen. Téměř všechny budou vlastnostmi, ale můžete objevit další subjekt nebo dva, které potom musíte přidat do svého seznamu subjektů a poté pokračovat.

Vrátíme-li se k našemu příkladu předpokládejme otázku:

„Co chcete sledovat o subjektu [Muži]? (Co chcete sledovat o mužích?)

a klient odpoví:

„Potřebujeme znát, příjmení, jméno, titul, adresu, město, stát, PSČ, věk, národnost a vzdělání.“

Opět nerealisticky jasná odpověď. Na základě této odpovědi můžeme sestavit následující věcný seznam vlastností pro [Muži].

Muži

Název sloupce	Typ dat	Klíče
ID Muže	Longint	PK
Příjmení	Řětězec (45)	
Jméno	Řětězec (45)	
Titul	Řětězec (45)	
Adresa	Řětězec (80)	
Město	Řětězec (30)	
Stát	Řětězec (2)	
PSČ	Řětězec (9)	
Národnost	Řětězec (20)	
Vzdělání	Řětězec (20)	

Inforce

Jesliže se nemůžete rozhodnout zda podstatné jméno je vlastnost nebo subjekt, považujte jej za subjekt. Jestliže jste se zmýlili, objevíte to v dalších fázích procesu návrhu.

V době zjišťování vztahů, se zeptejte klienta na každý vztah mezi každou dvojicí subjektů ve vašem seznamu:

> Je nějaký vztah mezi subjektem A a subjektem B?

Pro dvojici u které klient odpoví ano, položte dvě následující otázky:

> Může existovat více subjektů A pro každý subjekt B?

a

> Může existovat více subjektů B pro každý subjekt A?

Jestliže odpověď bude ano, pouze na jednu otázku, obdržíte vztah 1:M a budete podle toho postupovat. Jestliže odpověď bude ano na obě otázky, obdržíte vztah M:M a již víte že to znamená určitý uschovaný subjekt, který budete muset vytvořit a pak potřebujete o tomto vztahu zjistit více.

Vraťme se k našemu příkladu a předpokládejme, že se zeptáme:

> „Může mít žena více mužů?“

a klient odpoví:

> „Evropská kulturní norma nedovoluje mnohomužství. Ale protože dovoluje více sňatků za život musí odpovědět ano, žena může mít více mužů.“

Pak se zeptáte:

> „Může mít muž více žen?“

a klient odpoví:

> „Opět, musím říci, ano, muž může mít více žen.“

Tímto jsme objevili vztah M:M mezi muži a ženami. Potom se zeptáte:

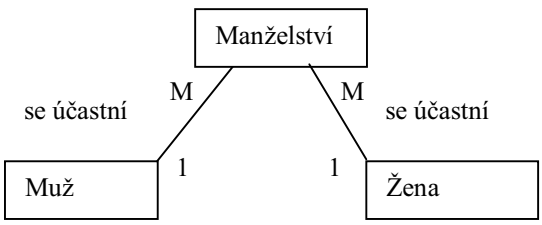
> „Viditelně je zde něco dalšího co je třeba sledovat. Protože existuje více mužů k jedné ženě a více žen k jednomu muži, musí existovat něco co je spojuje dohromady. Můžete mi říct co to může být?

A klient odpoví:

> „Samozřejmě je spojuje manželství.“

Náhle si vzpomenete, že vám tato skutečnost byla sdělena již na začátku. Připadáte si poněkud přihlouple, ale zaznamenáte vztah 1:M mezi [Muži] a [Manželství] a mezi [Ženy] a [Manželství].

Proces vede k VRD ukázanému níže (viz také předchozí kapitola)



Tato metoda na tomto našem příkladě se může zdát nesmyslně mechanická. Ale jestliže budete pracovat s klientem, který působí ve výrobním podniku na jakási hejblátka. To znamená že se nemůžete řídit svými současnými znalostmi nebo instinktem a pak shledáte tento způsob velice užitečný. Výhodou je, že nemusíte znát nic o zákazníkově činnosti, aby jste objevili podstatu systému, který si on nebo ona přeje. A ještě navíc se v průběhu „zpovědi dozvíte něco o hejblátkách.

7.2. Srovnání uživatelských pohledů

Do této doby jsme se soustředili na přístup ze shora dolů, který zjišťuje subjekty a vztahy podstaty problémů metodou zpovědi zákazníka. Tento způsob se dá použít často ne však vždy. A to z několika důvodů:

- Jednotliví uživatelé často popisují jejich osobní pohled na systém
- Situace může být natolik složitá, že subjekty a vztahy mezi nimi nejsou jednoznačně zřejmé. Čím větší projekt tím pravděpodobněji se dostanete do této situace.
- Potřebujete si být skutečně jisti, že jste zachytili všechny podstatné detaily.

Alternativním přístupem je srovnání a zdokumentování pohledů všech uživatelů. Což nejčastěji můžete zjistit z tiskových zpráv, formulářů nebo obrazovek již existujícího systému. Tento způsob se často nazývá přístupem ze spoda-nahoru. Zde je postup:

- Zdokumentujte pohled každého uživatele za pomoci „zpovědi popsané výše.
- Vytvořte seznam a definujte všechny prvky dat (subjekty, vlastnosti a vztahy).
- Zjistěte všechny funkční závislosti pohledu každého uživatele.
- Vytvořte VRD pro pohled každého uživatele.
- Normalizujte obsah pohledu každého uživatele.
- Prověřte všechny tyto pohledy.
- Vyhledejte subjekty s tímtéž klíčem. Obvykle je budete schopni sloučit do jednoho subjektu.
- Vyhledejte všechny subjekty nebo vztahy s vlastnostmi, které jsou klíčem nebo částí klíče jiných vlastností nebo subjektů. Tyto budou obvykle cizími klíči a indikují další vztahy.
- Vše proveďte pečlivě, aby jste se vyhnuli zdvojení subjektů, vztahů a vlastností.

Následující část zdůrazňuje některé z těchto problémů.

7.3. Konflikt názvů

První oblast, na kterou se soustředíme je konflikt názvů. Tzn. že musíme nalézt a prověřit všechna synonyma a homonyma.

Dvě slova jsou synonymem pokud mají stejný nebo velice podobný význam. Zde užíváme tato slova v trochu speciálním významu. V databázi jsou dva názvy synonymy jestliže představují tentýž koncept.

Dvě slova jsou homonymem jestliže znění stejně, píší se stejně ale mají jiný význam. V databázi jsou dva názvy homonymem jestliže pro daný název užívají různý koncept (jsou zde ukládána data jiného významu).

Oběma těmto způsobům se ve vašem návrhu databáze musíte vyhnout. Bohužel se homonyma i synonyma v návrzích databází vyskytují velice často.

Předpokládejme, že máte klienta, u kterého se na návrhu databází podílí několik osob. První osoba p. Kovář je obchodníkem a pracovníkem marketingu, druhá osoba pí. Honzíková je účetní.

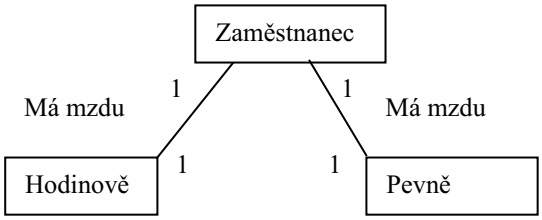
P. Kovář vám při rozhovoru sdělil, že systém musí sledovat požadavky zákazníků. Pí. Honzíková vám sdělí, že potřebuje, aby systém sledoval objednávky.

Samozřejmě se domníváte, že jde o dva různé subjekty, opak je však pravdou. Po prověření zjistíte, že požadavky zákazníků jsou pouze jiným vyjádřením pro objednávky (klient pouze vnitřně přezval tento subjekt pro potřeby marketingu).

Takže jste právě narazili na synonymum. Výsledkem je, že máte dva subjekty, které je potřeba sloučit.

Homonynum je podobné. Předpokládejme, že navrhujete aplikaci pro sledování personálních záležitostí. Jeden z našich subjektů je [Zaměstnanci]. Po prověření však objevíme, že existují dva typy zaměstanců. Zaměstanci s hodinovou a zaměstnanci s pevnou mzdou. Charakteristiky těchto dvou skupin zaměstnanců jsou však zcela rozdílné. Zaměstnanci s hodinovou mají jiné výhody, srážky atd.

Toto je příklad takového vztahu. Zaměstnanec s pevnou mzdou je prvkem třídy zaměstnanci. Stejně jako zaměstnanec s hodinovou mzdou je prvkem třídy zaměstnanci. Nicméně vlastnosti těchto subjektů jsou natolik odlišné, že je musím rozlišit formálně, což provedeme následující způsobem:



Toto je příklad vztahů 1:1 (zda se nám podaří včlenit tento vztah je jiná věc, mějte na paměti, že navrhujeme nikoli programujeme).

Homonymem je zde zaměstnanec. Považovali jsme jej za jeden subjekt. A zmýlili jsme se v základním aspektu systému. Homonymum (jeden název) nám ze dvou subjektů jeden neudělá.

Příklady Zpověď zákazníka

Příklad 7.1

Advokátní kancelář Dvořák, Chaloupka, Hruška, AK vás požádala o návrh systému pro sledování informací o jednotlivých případech a který jim současně pomůže automatizovat jejich praxi vymáhání náhrad za pracovní úrazy.

Vaším kontaktem je Alois Dvořák (známý svým přátelům jako Lojza), hlavní společník firmy. Lojza je agresivní, rozhodný mladý právník. Naneštěstí si myslí, že o navrhování databází toho ví tolik co vy.

Jeden z jeho hlavních cílů je sledovat účinost inzerce ve Zlatých stránkách, tisku, rádiu a televizi. Z tohoto důvodu chce evidovat klienty, kteří k němu přišli na základě té určité inzerce. Tak chce dosáhnout zvýšení účinnosti peněz vložených do reklamy.

Také dostává typy od doktorů, kteří s ním pracují na případech (on jim rovněž posílá pacienty, je to klasický příklad, přítelíčkování - já na bráchu brácha na mně). Tzn., že chce také sledovat, který doktor mu dohazuje nejvíce obchodů, aby mu mohl posílat též více pacientů.

Po svém příchodu zjistíte, že Lojza již za vás navrhl strukturu databáze. Po vás pouze chce, aby jste ji zavedli. A protože vám ušetřil tolik práce požaduje slevu. Zde je část jeho struktury:

Klient

Název sloupce	Typ dat	Klíče
Příjmení	Řetězec (45)	PK1
Jméno	Řetězec (45)	PK2
Titul	Řetězec (45)	PK3
Adresa	Text	
Rodné číslo	Řetězec (45)	
Telefon práce	Řetězec (10)	
Telefon domů	Řetězec (10)	
Přišel na inzerát	Logické	
Inzerováno kde	Řetězec (30)	CK1.1
Telefon inzert. kanceláře	Řetězec (10)	
Datum inzerátu	Datum	CK1.2
Od doktora	Logické	
Jméno doktora	Řetězec (80)	
Specializace doktora	Řetězec (20)	

Případ

Název sloupce	Typ dat	Klíče
Číslo případu	Longint	PK
Příjmení klienta	Řetězec (45)	CK1.1
Jméno klienta	Řetězec (45)	CK1.2
Telefon práce	Řetězec (10)	
Telefon domů	Řetězec (10)	
Jméno mandanta	Řetězec (45)	
Částka za případ	Real	
Adresa mandanta	Text	
RČ mandanta	Řetězec (45)	
Jméno pr.zást.dr.strany	Řetězec (80)	
Telefon pr. zást. dr. strany	Řetězec (10)	
Přišel na inzerát	Logické	
Inzerováno kde	Řetězec (30)	CK2.1
Telefon inzert. kanceláře	Řetězec (10)	
Datum inzerátu	Datum	CK2.2
Od doktora	Logické	
Jméno doktora	Řetězec (80)	
Specializace doktora	Řetězec (20)	

Inzerát

Název sloupce	Typ dat	Klíče
Inzerováno kde	Řetězec (30)	PK1 CK1
Datum inzerátu	Datum	PK2

Některé z těchto tabulek obsahují složené sloupce. Najděte je a nahraďte je atomickými sloupci.

Některé Lojzovy výběry primárních klíčů jsou pochybné. Opravte je. Pokud je to nutné vytvořte syntetické klíče.

Zjistěte všechny funkční závislosti.

Normalizujte tuto tabulku do 3NF.

V návrhu chybí některé tabulky. Přidejte je.

Vytvořte věcný seznam vlastností pro každou normalizovanou tabulku. Určete primární a cizí klíče.

Vytvořte VRD znázorňující normalizovanou strukturu.

Jste mladý hladový návrhář databází. Lojza se vám nijak zvlášť nelíbí, ale potřebujete peníze. Vysvětlete mu, proč je vaše struktura lepší než jeho a lépe splňuje účely automatizace jeho praxe.

Příklad 7.2

Vaším klientem je První tisková kancelář (PTK), používaná různými novinami a dalšími po celé zemi.

Jméno vašeho kontaktu je David. Je výkonný ředitel PTK. Provedl jste s ním následující rozhovor:

Vy: „Davide, můžete mi říci co by měl váš systém sledovat?

David: „Spolupracujeme s různými typy novin, časopisů a vydavateli dalších tiskovin. Tito vytvářejí reportáže. Reportáže jsou psány reportéry, kteří pracují pro jednotlivé vydavatele. Potřebujeme znát vydavatele reportáže a reportéra, který reportáž vytvořil. Je často důležité znát, který reportér pro koho pracuje.

„Jindy dostáváme požadavky na reportáže na určité téma. Témata reportáží charakterizujeme jedním slovem. Např. můžeme mít zákazníka, který požaduje reportáž slovy „Perskýě, „Zálivě, „Válkaě, „Poušťě, „Bouřeě. Musíme být taky schopni zjistit reportáže, které jsou označeny všemi těmito jednotlivými slovy a nebo nejsou charakterizována těmito slovy nebo jejich libovolnou kombinací. Takže to může být poněkud komplikované.“

„To je ve stručnosti vše.“

David byl odvolán na nenadálou schůzku a toto je vše co vám řekl a vy musíte pokračovat samostatně. Předtím než odešel vás požádal, aby jste sestavili první pracovní návrh systému, aby jej s vámi mohl později probrat.

Určete všechny subjekty zahrnuté v systému.

Podívejte se jestli můžete rovněž stanovit vlastnosti (zkuste je uhodnout, kde budete muset).

Určete funkční závislosti mezi vlastnostmi.

Vytvořte věcný seznam vlastností pro každý subjekt.

Pokuste se vytvořit VRD na základě svého rozboru.

Inforce

8. Pokročilejší způsoby modelování

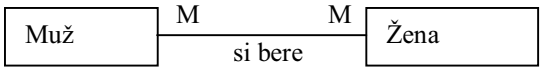
Tato kapitola pokračuje v diskusi na téma modelování vztahů subjektů. Budeme se zabývat určitými triky a představíme si některé pokročilejší metody, které vám pomohou rozhodnout se a problém vyřešit.

8.1. Ukládání historických dat

Již jsme se setkali s ukládáním historických dat. Typická data se s ohledem k času dělí na dvě skupiny:

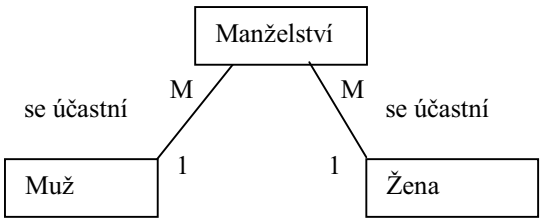
- Databáze ukládá jediný výskyt dat, označující současný stav dat a ignoruje historii.
- Databáze ukládá více výskytů dat odrážející jejich změny v čase. Typicky nejnovější výskyt určuje současný stav dat.

Pamatujete si náš příklad s manželstvím? Původně jsme modelovali databázi prvním způsobem:



Tento způsob ukládá pouze fakt, že muž je právě ženat s jednou ženou a naopak.

Pak, jsme se začali zajímat o ukládání změn manželského stavu v čase, který nás dovedl k následující struktuře:



Takže vidíme, že ukládání historie dat nás dovedlo k novým vlastnostem, které je potřeba ukládat a k novým subjektům a výsledkem je vztah 1:M starých subjektů k novému subjektu. Tento proces můžete použít kdykoliv, když budete potřebovat ukládat historii dat.

Obecně řečeno existují přinejmenším dva způsoby ke sledování historie dat. Jeden, který jsme si představili při sledování historie sňatků. Protože osoba může být v manželském stavu nebo svobodná v libovolné době, je nezbytné ukládat datum počátku a datum konce tohoto stavu jak je ukázáno na následujícím věcném seznamu vlastností:

Manželství

Název sloupce	Typ dat	Klíče
ID muže	Longint	PK1 CK1
ID ženy	Longint	PK2 CK2
Datum sňatku	Datum	PK3
Datum rozvodu	Datum	

Všimněte si, že je nezbytné zahrnuto do primárního klíče datum sňatku, protože dva stejní lidé mohou být oddání více než jednou.

Druhý způsob ukládání historických dat je nejlépe ilustrován na příkladu, který jsme dříve použili v příkladu se [Součásti]. Předpokládejme, že chceme sledovat změny v ceně libovolné součásti.

Samozřejmě součást má vždy cenu, takže není nezbytné ukládat datum ukončení ceny. Budeme jednoduše předpokládat, že datum konce platnosti jedné ceny je rovno datum začátku platnosti další ceny. Toto nás dovede k následující struktuře:

Cena součásti

Název sloupce	Typ dat	Klíče
ID součásti	Longint	PK1
Datum začátku platnosti	Datum	PK2
Cena	Real	

Opět musíme zahrnout Datum začátku platnosti do primárního klíče, protože součást může mít tutéž cenu více než jednou.

8.2. Vztah subjektu sám k sobě (účtování materiálu)

Jiným klasickým problémem relační databáze je problém účtování materiálu. Týká se situace, kde je součást zahrnuta v celku, který je sám součástí. Tento problém je znám jako rekurzivní vztah. Vztah je rekurzivní, protože vede zpět sám k sobě.

Takže jsme si řekli, že máme rekurzivní vztah jestliže subjekt má vztah sám k sobě. Příklad účtování materiálu je příkladem tohoto vztahu, protože součást může zahrnovat součást.

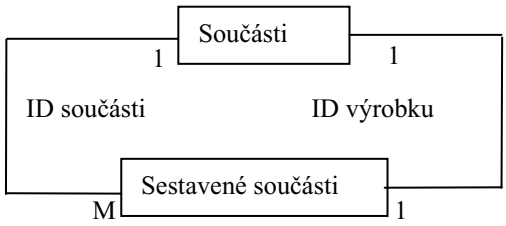
Struktura, která používá tento vztah vypadá typicky jako struktura následující:

Součásti

Název sloupce	Typ dat	Klíče
ID součásti	Longint	PK
Popis	Řetězec (80)	

Sestavené součásti

Název sloupce	Typ dat	Klíče
ID části	Longint	PK1
ID výrobku	Longint	PK2



Všimněte si, že vztah nás vede do kruhu. To je označení rekurzivního vztahu. Rovněž si povšimněte, že vztah je 1:M na jedné straně a 1:1 na druhé, takže součást může být prvkem mnoha celků (výrobků), ale celek je jenom jeden (jedna součást).

S tímto přístupem jsou vždy problémy, protože nemůže být řešen v návrhu relační databáze. Podívejme se na následující příkladová data:

Sestavené součásti

ID součásti	ID výrobku
123	456
123	567
123	678
456	777
456	778
567	887
567	888
678	222
678	223
678	224
222	123

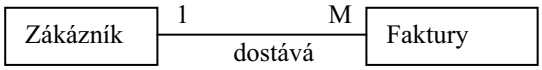
123 obsahuje 678, který obsahuje 222, který obsahuje 123 a který opět obsahuje 678 který opět....

Viditelně, je něco špatně. Někdo vložil nesprávná data. Výsledkem našeho pokusu je nekonečná smyčka. Program musí zkontrolovat každou řadu, jak je vložena, aby zjistil, který návrat po stavu dat je platný. Tento způsob musí být součástí zavedení databáze a je mimo rozsah tohoto kurzu. Nyní si pouze pamatujte, že takové situace a jejich řešení existují.

8.3. Ukládání odvozených dat

Tato situace často vzniká při návrhu databáze a týká se toho, zda ukládat data, která obsahují výpočty provedené na základě jiných uložených dat. Příklady jsou roční prodeje k datu podle střediska nebo průměr fakturované částky na zákazníka.

Jedna situace, kterou se musíme zabývat je normalizace s ohledem na odvozená data. Následující příklady se všechny zabývají touto datovou strukturou:



Uvažujme následující věcný seznam vlastností pro tabulku [Zákazníci]:

Zákazník

Název sloupce	Typ dat	Klíče
ID zákazníka	Longint	PK
Telefon	Řetězec (10)	
Adresa	Řetězec (80)	
Město	Řetězec (35)	
Stát	Řetězec (2)	
PSČ	Řetězec (9)	
Průměrná fakturovaná částka	Real	

Definovali jsme průměrnou fakturovanou částu, jako průměr fakturovaných částek toho jednoho zákazníka. Protože jestliže známe všechny fakturované částky zákazníka, můžeme vypočítat průměrnou fakturovanou částu. Můžeme se pak přít jestli průměrná fakturovaná částka je funkčně závislá na čísle faktury a ne ID zákazníka. Opak je pravdou. Vzpomeňme si na definici funkční závislosti z kapitoly 5:

> Řekli jsme, že sloupec A je funkčně závislý na sloupci B tehdy a jen tehdy, když existuje jedna hodnota ve sloupci B pro každou hodnotu ve sloupci A. Zdroj: Reference 2.

Takže pro každou hodnotu Čísla faktury neexistuje jedna hodnota Průměrné fakturované částky. Proto Průměrná fakturovaná částka není funkčně závislá na Čísle faktury. Dále můžeme vyvodit, že tento sloupec není porušením jakékoli normalizované formy, kterou jsme si uvedli.

Vezměme však následující příklad. Uvažujme tento věcný seznam vlastností:

Faktury

Název sloupce	Typ dat	Klíče
Číslo faktury	Longint	PK
ID zákazníka	Longint	CK1
Částka bez DPH	Real	
DPH	Real	
Částka celkem	Real	

Ukládáme zde Částku celkem, která se skládá z Částky bez DPH a z DPH. Protože Částka celkem je funkčně závislá na kombinaci sloupců Částka bez DPH a DPH porušujeme 3NF.

Existuje ještě třetí možnost. Podívejme se na následující variantu. Tentokrát budeme uvažovat obě tabulky [Zákazníci] a [Faktury]:

Zákazníci

Název sloupce	Typ dat	Klíče
ID zákazníka	Longint	PK
Telefon	Řetězec (10)	
Adresa	Řetězec (80)	
Město	Řetězec (35)	
Stát	Řetězec (2)	
PSČ	Řetězec (9)	
Koeficient slevy	Real	

Faktury

Název sloupce	Typ dat	Klíče
Číslo faktury	Longint	PK
ID zákazníka	Longint	CK1
Částka před slevou	Real	
Sleva celkem	Real	
Částka po slevě	Real	

Povšimněte si, že [Faktury]Sleva celkem je výpočtový sloupec, který se skládá z [Faktury]Částka před slevou násobené [Zákazníci]Koeficent slevy. Protože [Faktury]Sleva celkem je jednoznačně funkčně závislá na kombinaci sloupců [Faktury]Částka před slevou a [Zákazníci]Koeficient slevy, porušujeme evidentně 3NF.

Takže existují celkem tři typy odvozených dat. První typ jsou součtová odvozená data, která jsou počítána s ohledem na hodnoty v tabulce jedinců a počítají na základě hodnot z tabulky skupin. V našem prvním příkladě je Průměrná fakturovaná částka odvozená z tabulky [Faktury] ve vztahu 1:M.

Druhým typem jsou místní odvozená data, která jsou počítána ze sloupců uvnitř jedné tabulky a ukládána v této tabulce. V našem druhém příkladě je to Částka celkem.

Třetím typem jsou křížově odvozená data, která jsou počítána na základě sloupců dvou tabulek, ale ukládána na stranu skupin ve vztahu 1:M. V našem třetím příkladě je to [Faktury]Sleva celkem ([Faktury]Částka po slevě jsou místně odvozená data, protože jsou počítána ze sloupců uvnitř tabulky [Faktury]).

Pouze jeden typ z těchto dat je normalizován. Jsou to součtová odvozená data nebo data, která jsou počítána ze sloupců tabulky skupin a ukládána ve sloupci tabulky jedinců. Používání modelování odvozených dat jiných než součtových by jsme se měli vyvarovat. (Úplně se vyhnout ukládání těchto typů odvozených dat nelze, pamatujme že nyní navrhujeme a neprogramujeme.)

Dokonce uvažujeme-li součtová odvozená data narazíme na problémy zdvojování, protože odvozená data jsou počítána z jiných dat, jsou tato data již sama o sobě zdvojená. Můžeme je počítat kdykoliv chceme a potřebujeme a pak je nutně musíme uložit. Avšak otázka ukládání zde není hlavní, protože data jsou ukládána na straně jedinců ve vztahu 1:M tak již z definice vyplývá, že tato data zaberou mnohem méně místa než ostatní typy odvozených dat.

Kromě toho musíme vždy přepočítat data kdekoliv a kdykoliv se změní základní data, což vyvolává problém soudržnosti dat. Není to jeden z problémů, kterému se snažíme vyhnout normalizací? Je to samozřejmě vážný problém, ale existuje cosi, co zmírní tento problém? Metoda se nazývá Metoda potvrzení (uveřejnění) a vysvětlíme si ji dále.

Nakonec ještě jeden velký problém týkající se chování. Vraťme se k našmu původnímu příkladu:

Zákazníci

Název sloupce	Typ dat	Klíče
ID zákazníka	Longint	PK
Telefon	Řetězec (10)	
Adresa	Řetězec (80)	
Město	Řetězec (35)	
Stát	Řetězec (2)	
PSČ	Řetězec (9)	
Průměrná fakturovaná částka	Real	

Předpokládejme nyní, že pro daného zákazníka přidáváme fakturu. Výpočet průměrné fakturované částky nás povede následujícími kroky:

- Vložení faktury jak by jsme provedli i normálně.
- Výběr všech ostatních již vytvořených faktur tohoto zákazníka včetně nové faktury.
- Součet všech částek faktury [Faktury]Částka celkem ze všech faktur získaných v předchozím kroku.

- Podíl výsledného součtu počtem faktur získaných v předchozím kroku
- Uložení výsledku do sloupce [Zákazníci]Průměrná fakturovaná částka.

To je určitě mnoho problémů a kroků při vkládání pouze jedné faktury. Proces tím určitě velmi zpomalíme. A co hůře pro naše nejlepší zákazníky, kteří mají nejvíce faktur a které budeme nejvíce používat bude tento proces pomalejší a pomalejší, čím více faktur bude systém obsahovat.

Abychom vyloučili problémy se soudržností relačních dat a chováním systému, musíme použít úsudek při řešení tohoto problému. Abychom porozuměli jak použít úsudek musíme ještě provést další kategorizaci odvozených dat.

Uvažujme následující alternativní návrh tabulky [Zákazníci].

Zákazníci

Název sloupce	Typ dat	Klíče
ID zákazníka	Longint	PK
Telefon	Řetězec (10)	
Adresa	Řetězec (80)	
Město	Řetězec (35)	
Stát	Řetězec (2)	
PSČ	Řetězec (9)	
Celkové prodeje	Real	
Celkový počet faktur	Longint	

Zde jsme rozdělili sloupec Průměrná fakturovaná částka do dvou sloupců, a to Celkové prodeje a Celkový počet faktur. Povšimněme si, že to jsou kompomenty ze kterých se Průměrná fakturovaná částka počítá. Nechystáme se ukládat samotnou Průměrnou fakturovanou částku, protože víme, že by jsme tak porušili 3NF. (Obdrželi, by jsme potom místně odvozená data.) Nyní jsme schopni vypočítat Průměrnou fakturovanou částku, kdykoliv ji budeme potřebovat.

Když se nyní podíváme na proces vkládání nové faktury, obdržíme následující postup:

- Vložit fakturu.
- Zvětšit hodnotu v sloupci [Zákazníci]Celkové prodeje o částku uloženou v sloupci [Faktury]Částka celkem. (Poznamenejme, že v tomto okamžiku již máme řadu pro daného zákazníka zavedenu v paměti, takže tento krok nebude mít vliv na chování databáze.)
- Zvětšit sloupec [Zákazníci]Celkový počet faktur o jedna. (Opět žádný vliv na databázi.)

Neuvažovali jsme problém uložení těchto odvozených dat, který je triviální. Všechny ostatní problémy jsme však eliminovali a samotný údaj Průměrná fakturovaná částka můžeme vypočítat kdykoliv to potřebujeme.

Takže můžeme učinit závěr, že rozdělením sloupce [Zákazníci]Průměrná fakturovaná částka do dvou jednodušších částí jsme schopni vytvořit mnohem lepší návrh. První typ odvozených dat reprezentovaný sloupcem Průměrná fakturovaná částka se nazývá odvozená data druhého řádu.

Ukazatel, kterým rozeznáme odvozená data druhého řádu je to, že tato data jsou odvozena z dalších odvozených dat.

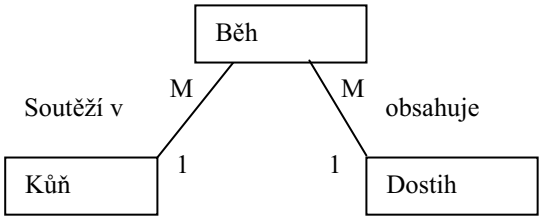
Druhý typ dat reprezentovaný sloupci [Zákazníci]Celkové prodeje a [Zákazníci]Celkový počet faktur se nazývá odvozená data prvního řádu. Ukazatel, kterým rozeznáme tato odvozená data je, že jsou

odvozena přímo z databáze tj. neodvozených dat. Na daném příkladě vidíme, že oba sloupce jsou vypočteny přímo ze základních dat tabulky [Faktury].

Velice důležité na odvozených datech prvního řádu je, že mohou být vypočítána pomocí metody potvrzení. Proces potvrzení zahrnuje přičtení nebo odečtení hodnot uložených v sloupci tabulky jedinců s pomocí hodnot v jednom nebo více sloupců v pouze jedné řadě tabulky skupin. Jestliže potvrzování je prováděno při každém vkládání, úpravě nebo mazání řady v tabulce skupin, jsou ostatní mimořádné problémy triviální. Pokud máme správně provedený systém potvrzování udržuje se sám.

Podívejme se na jeden další příklad, abychom si výše zmíněné ještě ujasnili.

Předpokládejme, že máme systém pro sledování koňských dostihů. Každý závodní kůň se účastní mnoha dostihů. Jednu účast koně v soutěži nazveme běh. Přejeme si sledovat průměrný čas každého koně ze všech dostihů. Následující je struktura této databáze.



Zde máme věcný seznam vlastností:

Kůň

Název sloupce	Typ dat	Klíče
ID koně	Longint	PK
Jméno koně	Řetězec (80)	
Trenér	Řetězec (80)	
Plemeno	Řetězec (35)	
Pohlaví	Logické	
Celková doba běhu	Real	
Počet účastí	Integer	

Běh

Název sloupce	Typ dat	Klíče
ID koně	Longint	PK1 CK1
ID dostihu	Longint	PK2 CK2
Doba běhu	Real	

Dostih

Název sloupce	Typ dat	Klíče
ID dostihu	Longint	PK1
Dráha	Řetězec (80)	PK2 CK1
Datum	Datum	

Zdá se trochu podivné ukládat Celkovou dobu běhu pro každého koně. Skutečně, celková doba, kterou kůň běží na dráze není to podstatné co nás zajímá. Totéž platí i o počtu účastí. To o co se doopravdy zajímáme je průměrná doba běhu tak, že budeme moci porovnávat koně a známkovat je.

Odvodili jsme tuto strukturu rozdělením průměrné doby běhu v jednom dostihu a obdrželi jsme odvozená data prvního řádu. Nyní již můžeme jednoduše použít metodu potvrzování a skládat do sloupce [Koně]Celková doba běhu a [Koně]Počet účastí jednotlivé údaje postupně po pořízení každého záznamu o jednom dostihu. Toto řešení nám poskytne rychlý a jednoduchý způsob k vypočítání průměrné doby běhu libovolného koně nebo k třídění tabulky [Koně] podle výsledků výpočtu.

Příklady Pokročilejší způsoby modelování

Příklad 8.1

Vraťme se zpět k příkladu 6.10. Nyní si přejem sledovat historii s ohledem k pirátské kariéře, to jest funkce pirátů na lodi a jejich postup od plavčíka ke kapitánovi.

Nalezněte všechny změny ve funkčních závislostech.

Proveďte všechny úpravy v věcném seznamu vlastností.

Vytvořted nový VRD zobrazující změny.

Příklad 8.2

Přejeme si vytvořit databázi na sledování rodokmenu jednotlivých osob. Do databáze musí být zahrnuty narození, sňatek, rozvod a úmrtí osob sledovaných v databázi.

Vytvořte návrh databáze včetně věcného seznamu vlastností pro každý subjekt a VRD zobrazující všechny subjekty.

Příklad 8.3

American Baseball League vás najala jako konzultanta v oboru databází k navržení databáze pro sledování výkonnosti jednotlivých týmů a hráčů. Protože je baseball hra založená na statistice, chtějí různé druhy statistik o hráčích, včetně:

- průměr na pálce (za sezónu)
- průměr na pálce (za život)
- účast (za sezónu)
- účast (za život)
- průměrný počet chyb na zápas (za sezónu)
- průměrný počet chyb na zápas (za život)
- oběhy (za sezónu)
- oběhy (za život)
- oběhy domů (za sezónu)
- oběhy domů (za život)

Také potřebuje vypočítat následující statistiky pro tým:

- Celkem oběhů (za sezónu)
- Průměr oběhů na hru (za sezónu)
- Výhrané zápasy (za sezónu)
- Prohrané zápasy (za sezónu)
- Remízy (za sezónu)

Rovněž je užitečné moci si vytvořit statistiky pro týmy v minulých sezónách.

Pro ty kdo neznají baseball, jeden hrací rok je sezóna. Průměr na pálce je určován podělením úspěšných úderů a počtu, kolikrát hráč stál na místě pálkaře. Během hry může hráč udělat jednu, nebo i více chyb. Účast je tehdy, když je hráč účasten hry. Oběh událost, kdy hráč přeběhne metu. Oběh domů je jednoduše událost, kdy hráč přeběhne všechny mety až k domácí metě. Hra je vyhrána týmem s největším počtem oběhů.

Navrhněte vhodnou databázi pro tyto účely včetně věcného seznamu vlastností pro každý subjekt a VRD zobrazující všechny subjekty.

9. Dokončení specifikace návrhu databáze

Tato kapitola zakončuje naší diskuzi o fázi návrhu databáze při tvorbě softwaru. V této kapitole dokončíme dokument nazvaný specifikace návrhu databáze, který vytvoří základ dokumentace návrhu.

9.1. Definice

Potřebujete dokončit definice všech subjektů, vlastností a vztahů v návrhu. Váš model není kompletní bez definic, které zajišťují, že další programátor porozumí vašemu návrhu do té úrovně jako vy. Samotný VRD není dostačující.

9.1.1. Definice subjektů

Definice subjektů musí popisovat dostatečně jasně subjekt, tak aby bylo možné určit zda určitá věc, záležitost, koncept je popsána touto definicí. Definice musí také jasně říkat proč je tento subjekt předmětem zájmu a sledování.

9.1.2. Definice vlastností

Definice vlastností musí popisovat danou věc, záležitost, koncept natolik jasně, aby bylo možno určit, který aspekt či charakterisatika subjektu je popsán. Definice musí také říci proč tato vlastnost je předmětem zájmu a sledování. Typ dat vlastnosti by měl být rovněž uveden.

9.1.3. Definice vztahů

Je velmi užitečné definovat podstatu vztahů mezi subjekty. Jak bylo již uvedeno vztah je obvykle vyjádřen slovesem. Opět, definice nám musí dostatečně jasně říci zda daná událost je skutečně popsaným vztahem. Měly by zde být uvedeny vlastnosti použité k vytvoření vztahu

Konečný výsledek tohoto procesu je slovník dat. Následuje příklad slovníku dat pro databázi prvně popsanou v oddíle 2.4:

Slovník dat:

Subjekty:

[Součást]

> [Součást] je typem položky, která je prodávána společností v normálním prodejním procesu. Takže tato definice se nevztahuje k jedné konkrétní položce, ale libovolné existující položce. Příklady jsou nosič deštníků, Kobereček, bič.....

[Obchod]

> [Obchod] je místo, kde jsou položky prodávány a uloženy těsně před prodejem. Příkladem je Obchod A Buffalo, B Hong Kong....

[Sklad]

> [Sklad] je jedna sada položek uložená v jednom obchodě těsně před prodejem. Všechny položky ve skladě jsou stejného druhu a to [Součásti]. Všechny položky stejného typu v jednom místě [Obchod] jsou brány dohromady. Příklad je 20 nosičů děštníků v obchodě A atd.

Vlastnosti:

Součást číslo

> Číslo, jež je přiřazeno pouze jedné součásti a jednoznačně identifikuje [Součást]. Je ukládána jako dlouhé celé číslo Longint.

Popis součásti

Textový popis jedné položky [Součást]. Je ukládána jako textový řetězec délky 80 znaků Řetězec(80). Každý popis musí být jedinečný v [Součást]

Kód obchodu

Jednoznačně identifikuje jeden [Obchod]. Uloženo jako textový Řetězec(2)

Město obchodu

Umístění jednoho [Obchodu]. Uloženo jako textový Řetězec(30)

Vztahy:

Být

[Součást] je položka jednoho [Skladu]. Evidované podle čísla.

Umístit

Jeden [Obchod] umisťuje [Sklady], označené kódem obchodu.

Jak můžete vidět slovník databáze je obzvlášť důležitý a to jak během vytváření databáze tak i v pozdější době.

9.2. Pravidla pro vytváření názvů

Názvy a jejich výběr jsou často velmi rozporuplné. Následující pravidla pro vytváření názvů jsou založena na použití zdravého rozumu a mohou pro vás být velice užitečná. Jestliže ne, rozhodně si vytvořte svá vlastní pravidla:

- K popisu dat užívejte běžná podstatná jména a slovesa. Vyhněte se speciální počítačové terminologii. Potřebujete terminologii, která bude něco říkat jak uživateli, tak programátorovi.
- Mezery mohou být použity během fáze návrhu, tak jak jsme to prováděli i my. Ve fázi programování mohou být vypuštěny.
- Ve fázi návrhu používejte názvy tak dlouhé jak potřebujete. Později je lze zkrátit tak, aby vyhovovaly prostředku programování tzn. 4D. Je velice užitečné vytvořit si seznam odkazů, obsahující názvy použité v návrhu a názvy použité v programu. Samozřejmě tyto názvy se nemohou nijak výrazně lišit.
- Všechny názvy obsažené v konečném dokumentu návrhu by měly být jedinečné. Nepoužívejte tytéž názvy vícekrát. Takže místo Příjmení použijte Příjmení zákazníka, neboť můžete mít v databázi ještě další sloupec, který bude ve významu Příjmení zaměstnance. Tento zvyk vám usnadní práci a učiní vaši dokumentaci čtivější a významově jasnou. Při vytváření databáze můžete tuto konvenci změnit, protože se již budete odkazovat na kompletní jméno sloupce vyjádřené názvem tabulky a názvem sloupce.

9.3. Zjištění potřebného místa k ukládání a požadavků na úkony

Musíte určit kolik dat bude uloženo v každé tabulce a kolik nových dat, změn dat a mazání se vyskytne pro jednotlivé řádky za určité období (normálně jeden rok). Potřebujete rovněž znát jestli během tohoto období existuje nějaký časový úsek s většími požadavky na zatížení.

Tato část rozhovoru se zákazníkem se objeví po vytvoření počátečního návrhu. Jednoduše se zeptáme zákazníka na následující otázky a stejnou otázku položíme pro každý subjekt v databázi.

„Kolik řad máte uloženo v tomto subjeku?“

„O kolik řad se rozroste vaše databáze v tomto subjektu během následujících dvanácti měsíců?“

„Kolik řad vymažete z tohoto subjektu během následujících dvanácti měsíců?“

„Kolik řad tohoto subjektu budete upravovat během následujících dvanácti měsíců?“

„Je během jakéhokoliv období používání databáze období, kdy dochází ke kumulaci práce a podstatně většímu zatížení systému?“

Vezměme příklad, který jsme použili v oddíle 5.6. týkající databáze Muzea. Ptáme se sl. Simonové, která je kurátorkou:

Vy: „Sl. Simonová identifikovali jsme dvě tabulky do kterých se budou data ukládat [Díla] a [Autoři]. Kolik uměleckých děl máte dnes v muzeu?“

Simonová: „Muzeum vlastní přibližně 30 000 různých objektů ve svých expozicích, depozitářích a některá jsou zapůjčena.“

Vy: „ Máte představu kolik autorů vytvořilo tato umělecká díla?“

Simonová: „Přibližně 5000.“

Vy: „Kolik nových uměleckých děl získáte za rok?“

Simonová: „Tento rok jsme získali 4500 nových prací. Minulý rok to bylo o něco více kolem 5500.“

Vy: „Je poměr uměleckých děl k autorům přibližně pořád tentýž 6:1?“

Simonová: „Odhaduji že tomu je právě tak

Vy: „Když prodáte umělecké dílo, předpokládám, že jej z databáze vymažete?“

Simonová: „Samozřejmě.“

Vy: „Existuje nějaký případ, kdy vymazáváte z databáze i autora?“

Simonová: „Ano, jestliže nemáme již žádné umělecké dílo od daného autora, již nás více nezajímá.“

Vy: „Kolik uměleckých děl prodáte za rok?“

Simonová: „Snažíme se bilancovat naše depozitáře během roku, takže počet nákupů a prodejů je přibližně stejný.“

Vy: „Existuje nějaká událost, která způsobí, že musíte upravit data o uměleckém díle?“

Simonová: „Ano, jestliže dílo zapůjčíme, umisťujeme je do depozitáře nebo jej z depozitáře vystavujeme, měníme stav tohoto díla.“

Vy: „Kolika uměleckých děl se to týká za rok?“

Simonová: „Odhaduji, že asi deseti procent za rok.“

Vy: „Existuje nějaká událost, která způsobí, že musíte upravit data autora?“

Simonová: „To je velice zřídka. Protože sbíráme pouze práce z období pozdního devatenáctého století, žádný z našich autorů již nežije. Takže nemusíme upravovat data, samozřejmě pokud se nestane chyba při pořizování.“

Vy: „Existuje nějaké období v roce, kdy obchody muzea nebo jiné aktivity jsou mnohem větší než v jiných obdobích?“

Simonová: „Ano, existuje. Téměř všechny naše obchodní aktivity jsou soustředěny na každoroční aukce v Sotheby´s a Christie´s. Tyto aukce se konají v červnu.“

Všimněte si, že jsme zcela nedodrželi formu otázek uvedenou výše, ale získali jsme všechny potřebné informace. Je rozhodně dobré pokládat je zákazníkovi v jazyce, kterému rozumí.

Tato konverzace nás dovedla k následujícímu seznamu zatížení:

Díla

Současný stav	30 000
Přírůstek	5 000
Mazání	5 000
Změna stavu	0
Úpravy	3 000
Nejvyšší počet úkonů za měsíc	10 000

Autoři

Současný stav	5 000
Přírůstek	833
Mazání	833
Změna stavu	0
Úpravy	0
Nejvyšší počet úkonů za měsíc	1 666

Praktické použití tohoto procesu je zřejmé. Známe počet řádek v každé tabulce. Srovnáním se strukturou dat můžeme určit celkové místo na uložení databáze. Také známe požadavky na budoucí růst databáze. Velice důležité je, že jsme objevili, že databáze musí umožnit více než 11000 úkonů za měsíc.

9.4. Zjištění obchodních pravidel a dalších omezení

Pro každou vlastnost musíme určit jaká jsou její obchodní pravidla a omezení. Takže jestliže vlastnost Odměna musí být mezi15 000 a 150 000 musíme to zdokumentovat. Rovněž by jsme měli zdokumentovat jakékoliv výchozí hodnoty. Podobně by jsme měli zdokumentovat libovolné výběrové seznamy, které budou používány pro výběr vlastností.

Výsledkem je seznam omezení jehož příklad je uveden níže:

Cena součásti

Název sloupce	Minimum	Maximum	Výchozí	Výběrový seznam
ID součásti				
Datum počátku platnosti ceny			Dnešní datum	
Cena součásti	0	1,500		
Typ změny ceny				Seznam typů změny ceny

Příkladová data pro výběrový seznam mohou být dokumentována následovně:

Seznam typů změny ceny

Dočasná sleva
Trvale
Nový dodavatel
Vynulování

9.5. Často hledané a tříděné parametry

Musíte určit sloupce, které bude klient často používat pro hledání a třídění řádek v databázi. Tento požadavek je kritický nejméně ze dvou důvodů:

- Tento proces vám pomůže objevit sloupce, které budou muset být indexovány.
- Požadavky na hledání přes tabulky vyžadují změnu strategie zlepšení chování systému. Poznání těchto požadavků vám pomůže k lepšímu plánování a případně změnám v návrhu, takže sníží pozdější požadavky na úpravu databáze, které by vyplynuly jako výsledek fáze testování.

Aby jste určili často hledané a tříděné parametry, zeptejte se jednoduše zákazníka na konci procesu návrhu, které sloupce nebo sady sloupců zamýšlejí vyhledávat či třídit často a to ze všech tabulek. Rovněž musíte do těchto otázek zahrnout sloupce primárních klíčů. Věnujte zvláštní pozornost tomu, když po zákaznících chce hledat nebo třídit některou tabulku na základě sloupců jiné tabulky. Toto se nazývá hledání či třídění přes tabulky a je již ze své podstaty velice pomalé (to není jenom vlastnost 4D, ale všech systémů řízení databází). Rovněž věnujte pozornost hledání a třídění na základě více sloupců, toto je jiný požadavek na chování. Je rovněž nezbytné určit jak často zákazník zamýšlí hledat či třídit podle určitých sloupců či sady sloupců. Strategie optimalizace se bude dost významně lišit v závislosti na frekvenci každé operace. Frekvence jsou následující:

- Denně+ označuje, že zákazní zamýšlí provádět tyto operace téměř neustále..
- Denně označuje, že klient zamýšlí provádět tyto operace nejméně jednou za den.
- Týdenně označuje, že klient zamýšlí provádět tyto operace nejméně jednou za týden.
- Měsíčně označuje, že klient zamýšlí provádět tyto operace nejméně jednou za měsíc.
- Zřídkakdy označuje, že klient zamýšlí provádět tyto operace s periodou větší než jeden měsíc. Pokud není tabulka příliš velká nebo operace velice časově náročná nebudeme se požadavky s touto frekvencí vůbec zabývat.

Až stanovíte sloupce častého hledání a třídění, zdokumentujte je v seznamu hledání a třídění následujícím způsobem:

Součásti

Sloupce častého hledání	Frekvence
ID součásti	Denně +
Popis součásti	Denně +
Sloupce častého třídění	**Frekvence**
ID součásti	Denně +
Popis součásti	Denně +

Obchod

Sloupce častého hledání	Frekvence
Kód obchodu	Denně +
Město	Denně +
Sloupce častého třídění	**Frekvence**
Kód obchodu	Denně +
Město	Denně +

Sklady

Sloupce častého hledání	Frekvence
ID součásti + Kód obchodu	Denně +
ID součásti	Denně
Kód obchodu	Denně
Sloupce častého třídění	**Frekvence**
ID součásti	Týdně
Kód obchodu + ID součásti	Týdně
ID součásti + Množství na skladě	Týdně
Kód obchodu + Množství na skladě	Měsíčně

9.6. Bezpečnost

Zcela samostatně a určitě by jste se měli zabývat otázkami bezpečnosti, již během fáze návrhu. Je to obvykle politický horký brambor a vy musíte přinutit vrcholový management vašeho klienta, aby tuto otázku pro vás vyřešil. Odložení této otázky do fáze zavádění systému je velice špatné. Posaďte se s nimi nad navržené tabulky co nejdříve.

Pro každý subjekt musíte klientovi položit následující otázky:

- Komu bude dovoleno číst data?
- Komu bude dovoleno přídávat data?
- Komu bude dovoleno upravovat data?
- Komu bude dovoleno mazat data?

Pro každé zjištěné nastavení bezpečnosti musí být sestavena odlišná skupina. Skupina je soubor uživatelů, kteří mají tatáž práva přístupu.

Skupiny mohou být hierarchické. To znamená, že jednotlivé skupiny skládají dohromady další skupinu. Nejvyšší skupina je obvykle skupina Návrhářů, která obsahuje pouze návrháře databáze (zatím předpokládáme pouze jednoho návrháře). Skupina Návrháře je obvykle včleněna do všech skupin tak, aby umožnila návrháři přístup ke všem funkcím programu.

Další skupiny mohou zahrnovat Účetní, Obchod, Exekutivu, Marketing atd. Je běžné, že skupiny jsou zakládány podle funkčního zařazení. Velice často jsou definice skupin uváděny ve slovníku dat.

Po zodpovězení vašich otázek můžete navrhnout sadu skupin, které řeší požadavky na bezpečnost. Typicky váš klient pro vás vytvoří skupiny a dá vám je takže, vám ušetří práci v této oblasti. (samozřejmě

až potom, kdy ho k tomu přinutíte a objasníte mu svůj návrh). Tyto skupiny a jejich práva mohou být pak zdokumentovány v Seznamu přistupových práv a to následovně:

Díla

Přístup	Skupina
Číst	Všichni
Přidávat	Prodejci
Měnit	Prodejci
Mazat	Kurátor

Autoři

Přístup	Skupina
Číst	Všichni
Přidávat	Prodejci
Měnit	Kurátor
Mazat	Kurátor

9.7. Dokončení Výkazu návrhu databáze

Výkaz návrhu databáze obsahuje všechny různé položky, které jsme vytvářeli během tohoto kurzu. Zahrnuje:

- Věcný relační diagram
- Slovník dat
- Věcný seznam vlastností
- Seznam zatížení tabulek
- Seznam omezení tabulek
- Seznam hledání a třídění
- Seznam přístupových práv

Všimněte si, že jsme do seznamů nezahrnuli funkční závislosti. Seznam funkčních závislostí je velice užitečný během normalizace. Pokud však již máte normalizovanou strukturu jeho užitečnost se snižuje, protože funkční závislosti jsou obvykle zřejmé ze struktury. Samozřejmě musíte seznam funkčních závislostí vytvořit pro účely návrhu, ale pravděpodobně jej nezahrnete do konečné dokumentace.

Příklady Dokončení Výkazu návrhu databáze

Příklad 9.1

Lojzovo zmoudření. (Příklad 7.1.) Nyní již jste zkušený návrhář a vytváříte projekt.

V konečné fázi rozhovorů, jste objevili tato fakta:

- Lojza náhle souhlasil, když jste nanesli otázku, že může být více stran vztažených k případu a více právních zástupců k jedné straně. Ve skutečnosti pak již upozornil, že v případu může být rovněž více klientů. Kromě toho někteří žalobci nemusí být zákazníky firmy a maji své právní zástupce.

- Advokátní kanvelář Dvořák, Chaloupka, Hruška, AK (DCH) do této doby provedla 5000 případů pro 3500 zákazníků.
- DCH se zabývá přibližně 1500 případy za rok. Všechny i dokončené případy musí být v databázi uloženy po dobu minimálně 3 let a pak teprve mohou být archivovány a z databáze odstraněny. Případy jsou nabírány postupně a pravidelně. Archivování a očištění databáze se provádí jednou ročně na konci roku.
- Poměr nových případů a klientů je přibližně stále stejný t.j. 5:3.5.
- DCH nikdy nevymazává data o zákazníkovi a stále i minulým zákazníkům rozesílají marketingové dokumenty.
- Aktivní data případu jsou upravována velice často, Každá řada je upravována minimálně jednou za měsíc. Data klienta jsou modifikována, aby odrážela změny v adrese, telefonech atd. Za rok je upravováno asi 1000 řad klientů.
- DCH umisťuje inzerci u 35 vydavatelů. DCH zkouší i nové vydavatele. Minimálně 5 nových vydavatelů je kontaktováno za měsíc. DCH zřídka upravuje data o vydavatelích, ale děje se tak čas od času. Data o jedné provedené inzerci se téměř neupravují. Data o inzerci starší jednoho roku se archivují a odstraňují.
- DCH pracuje s 18 individuálními lékaři. Data o doktorech se upravují zřídka. Žádný lékař není z databáze vymazán. Do databáze je pořízen nejvíce jeden lékař za rok.
- Každý jednotlivý případ má minimálně jednu další stranu, než je klient kanceláře DCH. Někdy jsou strany účastny více než jednoho případu. Lojza uvedl, že strany jsou účastny více než jednoho případu velice zřídka. Strany jsou pořizovány do databáze v době kdy je pořizován případ. Jsou vymazávány pouze v době, kdy jsou případy archivovány. Ve většině případů jsou strany upravovány pouze jednou za rok. Protože případy jsou pořizovány neustále není doba, kdy by se nepořizovaly strany.
- Ačkoliv každá strana může mít minimálně jednoho právního zástupce, existuje pouze asi 100 advokátů zabývající se touto oblastí a tak si strany vybírají pouze mezi nimi. Jednou za čas se objeví nový advokát zaměřující se na tuto oblast, ale není to často. Advokáti často mění své kanceláře, takže každý advokát je upravován jednou za rok. DCH nikdy nevymazává svá data o právních zástupcích, protože se mohou účastnit na dalších případech později a je užitečné vědět na kterých případech, kdy pracovali. Je velmi neobvyklé, že strany mají více než jednoho právního zástupce ale stává se to. Protože právní zástupci jsou vkládáni a upravováni po celou dobu neexistuje čas, kdy se tak děje více nebo méně než v jiném období.
- DCH nikdy nevezme případ, kde by očekávaná částka byla menší než 100 000.
- Bude nezbytné vyhledávat klienty, podle příjmení a jména. Toto hledání bude prováděno často vícekrát než jednou za den. Třídění bude prováděno buď podle čísla klienta nebo podle kombinace příjmení a jména. Opět tento krok bude prováděn kdykoliv budou prověřována data o klientu tzn. více než jednou za den. K hodnocení marketingové úspěšnoti jednotlivých inzerátů bude nutné třídit podle PSČ. Toto se však bude dít pouze jednou za měsíc. Dále k prověření toho, kteří klienti přišli na jaký druh inzerátu je nutno hledat klienty, kteří přišli na základě inzerátu a třídit je podle inzerátů a vydavatelů. Opět tato tisková zpráva se provádi jednou na konci měsíce. Rovněž potřebujeme být schopni hledat klienty, kteří přišli na základě doporučení lékaře. Toto bude prováděno na základě jména a příjmení lékaře a bude se dít týdenně. Pro tiskovou zprávu musíme být schopni třídit zákazníka podle kombinace příjmení a jména lékaře. Tyto zprávy jsou prováděny pouze na konci měsíce
- DCH upřednostňuje případy s vyššími peněžními částkami. Tzn. že nejdůležitějším údajem o případu je peněžní částka. Potřebujeme však rovněž vyhledávat případy určitých klientů podle jejich příjmení nebo jména a nebo kombinace obojího a to vícekrát než jednou za den. Rovněž

musíme být schopni vytvářet zprávy, které třídí případy na základě kombinace jména a příjmení klienta a to více než jednou za týden.

- DCH potřebuje být schopna vyhledávat a třídit vydavatele podle názvu více než jednou za den. Také DCH potřebuje být schopna třídit zprávy ukazující inzeráty dle jednotlivých vydavatelů, které jsou tříděny podle názvu vydavatele a následně datumu inzerce. Tyto zprávy jsou prováděny na konci každého měsíce.
- DCH potřebuje být schopna třídit a hledat jednotlivé strany a jejich účast v případech podle jména, příjmení nebo kombinace jména a příjmení. Totéž platí pro právní zástupce. Toto bude prováděno více než jednou za den.
- DCH potřebuje třídit a hledat lékaře na základě jména, příjmení nebo kombinace jména a příjmení. Toto bude prováděno více než jednou za den.
- Lojza je hlavní společník, je jedinou osobou ve firmě, které je dovoleno přidávat nebo vymazávat klienty a účast klientů v případech, vydavatele a lékaře a rovněž vymazávat případy nebo právní zástupce. Každé další osobě je dovoleno přidávat případy a účast klientů v případech, strany a účast stran v případech, právní zástupce a inzeráty. Rovněž upravovat klienty, vydavatele, lékaře, případy a účast klientů v případech, strany a účast stran v případech nebo právní zástupce a účast právních zástupců v případech.

S uvážením toho co jste již vytvořili dříve dokončete výkaz návrhu databáze pro tuto úlohu. Proveďte odpovídající změny do vašeho předchozího návrhu s uvážením nových informací.

10. Optimalizace návrhu databáze

V této kapitole se budeme zabývat optimalizací. Otázky chování databáze, které vznikají při vytváření databází ve 4D jsou velmi podrobně popsány v dalších kurzech ACI. Tato kapitola se soustřeďuje speciálně na otázky, na které si musíte dát pozor již při navrhování struktury databáze.

Normalizace eliminuje zdvojené ukládání dat a uspořádává data do souborů podle logického významu. Tento proces nám pomáhá podrobně porozumět ukládaným datům. Úplná normalizace databáze je krok, který rozhodně nemůže být přeskočen. Avšak kompletně normalizovaná databáze nemusí být úplně optimalizovaná databáze. Existují význačné mimořádné problémy, které mohou být obsaženy ve způsobech vyhledávání záznamů, především s použitím více tabulek (hledání přes relace).

Seznam zatížení tabulek a Seznam hledání a třídění jsou nejdůležitější dokumenty, které budeme používat při optimalizaci struktury. Seznam hledání a třídění vám řekne, operace klient zamýšlí provádět nejčastěji a tabulky a sloupce zúčastněné v těchto operacích. Seznam zatížení tabulek určuje tabulky, v kterých bude uložen velký počet záznamů. Porovnáním těchto dvou seznamů tj. objevením, kde v kterém místě bude prováděno časté hledání a na jak velkých tabulkách, je přesně to, čím se musíme zabývat v řešení otázek chování databáze.

10.1. Dohoda v chování

Existují určité dohody v chování, které se projeví v porovnání zpracování jednoho záznamu (zavádění, vkládání, úpravy a mazání) a zpracování více záznamů (hledání a třídění). Chování při hledání a třídění nás vede k tomu, zlepšit jej pomocí vytváření indexovaných sloupců, denormalizací a ukládáním odvozených dat.

Např. čím větší je počet indexů, tím více indexů musí být obnoveno vždy, když je vkládána nová řada nebo tato řada upravována či mazána. Podobně jestliže je soubor nenormalizován, naroste množství dat uložených v každém záznamu, které způsobí, že záznam je pomaleji zaváděn do paměti a ukládán. Nakonec ukládání odvozených dat způsobí, že musí být prováděno více operací potvrzování. Zatímco operace potvrzení pro jeden záznam přinese málo dodatečných problémů, tyto operace pro více záznamů současně, mohou již způsobit závažný problém v chování.

V každém případě musíte mít na paměti, že všechny optimalizační techniky zvětší velikost datového souboru na disku.

Tento důvod nás většinou vede k tomu, že je struktura ponechána v normalizované formě a indexuje se minimální počet sloupců tak, aby jsme dosáhli uspokojivého chování. Problémy chování indikujeme při existenci relativně velkého počtu záznamů v tabulce a existenci často prováděných operací.

Samozřejmě vzniká otázka: „Co je velký počet záznamů?“. Odpověď však zní: „To záleží...“ V této oblasti musíte vždy použít rozum a provést kvalifikovaný odhad v závislosti na konkrétní situaci. Jesliže automatizujete databázi pro živé televizní vysílání, chování pomalejší než sekunda bude určitě absolutně kritické. V tomto případě, více než 100 záznamů může být mnoho. Jesliže vytváříte databázi pro sklad obchodu, několik sekund může být považováno za přijatelné chování a potom více tisíce záznamů může být málo.

Podstatná je: Na základě odpovědí vašeho klienta musíte určit jaká doba odezvy je přijatelná pro libovolnou operaci. Ujistěte se, že klient pochopil, že zlepšení chování a odezvy není zadarmo. A je to otázka dohody, která závisí na provedeném návrhu.

10.2. Otázky chování a odezvy

Tato část dále konkretizuje otázky vznikající při vytváření struktury databáze ve 4D.

10.2.1. Otázky ukládání

Každý index ve skutečnosti zdvojnásobuje požadavky na místo pro uložení jednoho sloupce (pole). Např. jestliže tabulka má 20 000 záznamů a sloupec v této tabulce je typu Alpha (10) ((Řetězec (10)), je místo

požadované pro uložení sloupce bez indexu maximálně 200 kb. (není to zcela přesný údaj ale je dostatečně blízko). Ovšem pokud je totéž pole indexováno je požadované místo pro uložení sloupce 400 kb.

Denormalizace, která zdvojeně ukládá některé hodnoty a ukládání odvozených dat, to vše zvyšuje počet sloupců v tabulce. Jak vzrůstá počet sloupců, vzrůstá samozřejmě i požadované místo pro uložení dat tabulky.

10.2.2. Zavádění záznamů

Při zavádění (načítání) záznamů do paměti 4D, ukládá v paměti dvě kopie jednoho záznamu. Do paměti je zaváděn celý upravovaný záznam a nikoliv pouze sloupce zobrazené na formuláři. Tzn. že zvýšením počtu sloupců v tabulce vzrůstá velikost paměti požadovaná systémem při úpravě jednoho záznamu. Samozřejmě tím vzrůstá i čas potřebný k zavedení záznamu do paměti.

Samotná velikost potřebné paměti pro jedno pole je samozřejmě malá (řekněme 20 bytes). Má-li však vaše tabulka stovky sloupců a mnoho z nich je indexováno, můžete vytvořit problém v chování. (V každém případě vaše tabulky, které obsahují takový počet záznamů musí být tak či onak prověřeny).

Další otázka při zavádění záznamů se týká velikosti dat uložených v polích. Jestliže existují velké datové položky jako jsou obrázky nebo textová pole obsahující velké množství textu. Můžete chtít rozdělit tabulku do dvou tabulek se vztahem 1:1. Tato technika je diskutovaná dále.

10.2.3. Vkládání, úpravy a mazání záznamů

Kdykoliv v tabulce vkládáme, upravujeme nebo vymazáváme záznam, musí být obnovovány indexy tabulky. To samozřejmě vyžaduje čas. Zvýšení počtu indexů, má proto efekt na chování systému při ukládání a mazání záznamů. Opět, množství času potřebného k obnovení každého indexu není velké, ale jestliže existuje velký počet indexů, jeho součet je velký a může vzniknout problém v chování.

10.2.4. Hledání a třídění (jedno pole)

Indexování dramaticky zlepší chování při třídění a hledání podle jednoho pole. Jestliže běžné operace zahrnují hledání a třídění pouze jednoho pole i u velkých databází je indexování nejrychlejší a nejjednodušší způsob zlepšení chování.

Mějte na paměti, pravidla uvedená v části Dohoda v chování: indexování zvětší místo potřebné pro uložení tabulky a existují určité mimořádné problémy s obnovováním indexů při vkládání, úpravách a mazání záznamů.

10.2.5. Hledání a třídění (více polí)

Když hledáme podle více polí (dotaz obsahuje více řádek na jednotlivá pole). 4D použije svou vnitřní optimalizaci a pokusí se nalézt nejefektivnější pro provedení dotazu. Např. jestliže jedno z hledaných polí je indexováno 4D, provede vyhledání tohoto pole nejdříve a pak dokončí hledání podle dalších neindexovaných polí. I když to pomůže, je potřeba si pamatovat, že hledání podle více polí je vždy pomalejší než hledání podle jednoho pole.

V případě třídění, neexistuje způsob, jak by 4D optimalizovala tuto operaci, proto libovolné třídění, které zahrnuje alespoň jedno neindexované pole je ze své vlastní podstaty neindexované třídění. Dokonce jestliže všechna pole jsou indexovaná je třídění podle více polí vždy pomalejší než třídění podle jednoho pole.

Jestliže běžně používáte operace, které zahrnují hledání a třídění podle více polí a tabulka je velká, budete určitě chtít vzít do úvahy data ukládaná zdvojeně v uměle vytvořeném sloupci vzniklém sloučením hodnot z ostatních sloupců a toto pole indexovat. Tato technika je diskutována dále.

10.2.6. Hledání a třídění (mezi tabulkami)

Hledání a třídění, které zahrnuje pole z více než jedné tabulky jsou vždy neindexovaná a jsou proto pomalá. V každém případě, jestliže musíte běžně provádět hledání a třídění mezi tabulkami, budete chtít

uvažovat denormalizaci dat. To znamená, že hodnoty z tabuky jedinců jsou zdvojeně ukládány v polích tabulky skupin. Pak hledání a třídění může být provedeno podle tohoto pole souboru skupin, což je rychlejší. Techniky denormalizace jsou diskutovány dále.

10.2.7. Hledání a třídění (odvozená data)

Nejpomalejší operací při hledání a třídění je operace s použitím výrazu (tzn. že program přinutíme vypočíst hodnotu z hodnot v uložených sloupcích - to jsou ve 4D příkazy: QUERY BY FORMULA, QUERY SELECTION BY FORMULA a ORDER SELECTION BY FORMULA). Jestliže běžně používáte tuto operaci ve velkých tabulkách, budete chtít uvažovat uložení odvozených dat v indexovaných sloupcích. To samozřejmě dramaticky vylepší chování této operace. Viz níže.

10.3. Výsledky indexování

Indexování výrazně zlepší schopnost vyhledávat záznamy. Index není nic jiného než další kopie dat tříděná podle hodnot, které jsou ukládány pouze v jedné kopii. K těmto hodnotám jsou pak přiřazena čísla záznamu a adresy záznamu, kde se tato hodnota vyskytuje. Indexy existovaly mnohem dříve, než počítače. Příkladem může být katalog kartiček knih podle oborů.

Protože indexy jsou přetříděny podle hodnot mohou být vyhledány, rychle a pak již je vyhledán patřičný záznam, který obsahuje informaci, hledanou podle indexu. Je možné vytvořit více než jeden index na tabulku, který nám poskytne další možnost rychlého vyhledání a třídění.

Není praktické indexovat každé pole v databázi. Indexy zabírají velké množství místa, pokud naše databáze obsahuje ve svém sloupci pouze jeden výskyt dané hodnoty, je index ve skutečnosti větší než originální data, protože index musí obsahovat samotnou hodnotu a umístění záznamu. Indexy musí být měněny vždy, když se mění samotná data, každý index musí být obnoven při každém vkládání nového záznamu nebo úpravách hodnoty v každém indexovaném poli. Ve velké tabulce s častými změnami a to dokonce i tehdy, když tabulka obsahuje pouze několik záznamů, tato skutečnost velice zpomaluje chování databáze, protože systém musí neustále trávit čas při obnově indexů.

Vícenásobné indexy proto nejsou vždy výhodné. 4D je schopna plně využít indexy při hledání a třídění podle jednoho pole, ačkoliv hledání ve 4D je optimalizováno, vyhledávání podle ostatních řádků dotazu je vždy prováděno sekvenčně a je tedy pomalé.

Existuje několik pravidel, kdy indexovat a kdy ne

- 4D automaticky indexuje, všechna pole, která se přímo účastní v nějakém vztahu
- Indexujte každé pole, které se často používá k hledání nebo přístupu.
- Vyvarujte se indexování libovolného pole, které je velice často měněno.
- Vyvarujte se indexování polí, která obsahují dlouhé textové řetězce.
- Buďte velice opatrní s indexováním v tabulkách, které jsou velice často upravovány přidáváním nebo mazáním záznamů. Můžete být velkorysí při indexování v tabulkách, které se zřídka obnovují.
- Indexy jsou rozhodně přepych pro tabulky, které obsahují pouze několik záznamů.

10.4. Denormalizace vztahu 1:M

Tato otázka je velice kontroverzní. Na jedné straně normalizace eliminuje zdvojená data. Na druhé straně, časté hledání přes tabulky je velice drahé co se týče chování a odezvy. Proto může být nezbytné denormalizovat za účelem zlepšení chování. Obzvláště jestliže hledání přes tabulky obsahuje veliké tabulky nebo je toto hledání prováděno přes několik tabulek.

Denormalizace je proces, kdy vezmeme data z pole souboru jedinců a uložíme je (zdvojeně) do pole souboru skupin. Např. uložení popisu zboží v položce faktury. Jestliže denormalizujete musíte si být jisti,

že je to skutečně výhodné a že neexistuje žádná jiná alternativa. Musíte si být jisti, že jiné další důležité transakce nejsou významně degradovány nebo dokonce nemožné. Např. změna popisu zboží a chceme-li tuto změnu promítnout do všech existujících položek faktury.

Denormalizovaná tabulka, téměř vždy vzniká uložením duplicitní informace ze souboru jedinců. Výsledná tabulka bude obsahovat mnohem více sloupců a dat a určitě i indexů. Pokaždé, když budeme přistupovat k záznamu této tabulky, budeme potřebovat ošetřit více dat. To samozřejmě zabere více času, paměti atd. Proto musíme být opatrní předtím, než takovou změnu provedeme.

Denormalizovaná tabulka je mnohem dražší na udržování. Data, která jsou často měněna nemohou být zdvojována bez velice dobrého důvodu. Uzamykání záznamů v databázích více uživatelů a více procesů, může znemožnit údržbu souhlasnosti těchto dat.

Jiná otázka, kterou musíme uvažovat je rychlost. Chování při hledání, může být významně vylepšeno pomocí denormalizace. Jestliže existují určité dotazy nebo obnova dat, která se děje často a my máme jasně určené požadavky na chování nemusíme mít pak jiný výběr než data denormalizovat. Pravděpodobně to bude pravda, pouze pro velice zatížené systémy, ve kterých data v databázi řídí reálné události.

10.5. Kdy, by jste měli vytvářet vztah 1:1?

10.5.1 "Objemná" tabulka

Dokonce, i když jste ve formuláři seznamu, užíváte příkazy DISPLAY SELECTION neb MODIFY SELECTION, pokaždé když použijete pole ve formuláři daného záznamu všechna ostatní pole tohoto záznamu jsou zavedena do paměti. V případě, 4D Server jsou pole dokonce přenesena přes síť do 4D Client.

Obrázky a velké textové bloky mohou potenciálně obsahovat velké množství dat.

Vezměme si příklad databáze studentů střední školy, která obsahuje všechny studenty, jejich adresy, telefony, jména rodičů, popisy mimoškolních aktivit a obrázek každého studenta. V úplně normalizovatné struktuře by obrázek měl být uložen v tabulce [Student]. To by však znamenalo, že pokaždé, když se někdo chce podívat na telefonní číslo studenta, vytisknout seznam studentů účastnících se nějaké akce, musí být na disku nalezený obrázek, zaveden do paměti a poslán na počítač klienta. Jestliže je potřeba pouze telefonní číslo studenta, zabírá to určitě zbytečně dlouhý čas.

Jiným příkladem může být databáze zaměstnanců, která obsahuje velké textové pole pro poznámky vedoucího o zaměstnanci. Poznámkové pole je zcela volná nestrukturovaná oblast, které vedoucí přidává nové poznámky a listuje si ve svých starých poznámkách. Jestliže bude toto pole uloženo v souboru [Zaměstnanci], jak to vyplývá z normalizace databáze, bude pokaždé při přístupu záznamu zaměstnance, zavedeno do paměti a přesouváno přes síť. Opět to může zabírat mnoho počítačového času a utrácet čas vedoucího pracovníka. Zatím neuvažujeme otázky bezpečnosti.

Řešení obou zmíněných situací je jednoduché. Vytvořte oddělený soubor [Objemný], který bude obsahovat tři sloupce na tabulku.

Tabulka [Objemný]

Název sloupce	Typ dat	Klíče
ID Objemný	Longint	PK
Obrázek	Obrázek	
Text	Text	

Kdykoliv, kdy v databázi potřebujete obrázek nebo větší textový blok, vytvoříte cizí klíč v tabulce, ve které budou tato data potřeba.

Studenti

Název sloupce	Typ dat	Klíče
ID studenta	Longint	PK
Jméno	Řetězec(20)	
Příjemní	Řetězec(25)	
Atd. atd.		
ID obrázku	Longint	CK1

Ve skutečnosti můžete mít více vazeb k téže objemné tabulce a proto v ní lze ukládat zcela rozdílná data pro každý odkaz.

Zaměstnanci

Název sloupce	Typ dat	Klíče
ID zaměstnance	Longint	PK
Jméno	Řetězec(20)	
Příjmení	Řetězec(25)	
Atd. atd.		
ID poznámka vedoucího	Longint	CK1
ID obrázku zaměstnance	Longint	CK2
ID os. hodn. zaměstnance	Longint	CK3

Vytvořte vztahy mezi hlavní tabukou a tabulkou [Objemný]a tyto vztahy budou neautomatické. Neautomatický vztah zajistí, že data se budou zavádět do paměti pouze budou-li potřeba. Kdykoliv budou data potřeba, můžete jednoduše použít příkaz RELATE MANY v souboru jedinců. Ačkoliv je tento vztah technicky vztahem 1:1 použijeme ve skutečnosti vztah 1:M. Skutečná data budou doopravdy reprezentovat vztah 1:1, protože pro každé ID objemný použité v hlavní tabulce bude existovat pouze jeden záznam v tabulce [Objemný].

10.5.2 Velké množství zřídkakdy užívaných dat

V některých databázích proces normalizace vede ke stovkám sloupců v jedné tabulce a přitom ve skutečnosti je mnoho z nich vyplňováno velice zřídka a tato data jsou zřídkakdy užívána. Tato situace často nastává, v případech databází o nějakém výzkumu. V případech, kdy se dostanete do této situace, je přijatelné umístit tato pole v odděleném souboru a vytvořit neautomatické vztahy mezi vzniklými tabulkami. To znamená, že hlavní soubor může být používán bez zavádění do paměti všech těchto zřídkakdy užívaných informací. Poznámka: jestliže máte pouze několik sloupců (menší než 40) nelze to pokládat za velký objem a nezískáte žádnou výhodu, rozdělením tabulky do více tabulek.

10.6. Ukládání odvozených dat (opakování)

V databázích existuje jedno klíčové pravidlo, které říká: „Nikdy neukládejte nic co můžete vypočítat!. Odvozená data zcela porušují toto pravidlo. Časté hledání nebo třídění na základě vypočítávaných dat může způsobit, že vaše databáze bude zcela nepoužitelná. Takže může být více než vhodné předem si tyto hodnoty připravit a uložit je do databáze.

V prvním příkladu se to týkalo faktury. Celková částka faktury není nic jiného než prostý součet násobku množství a jednotkové ceny. Avšak chcete-li často třídit faktury podle celkové částky bude databáze, která celkovou částku neuloží, nepoužitelná. Takže je často jediná možnost tyto částky vypočítat a uložit v databázi. V těchto zvláštních případech bude považováno za zcela normální, když databáze bude

obsahovat data, která jsou nenormalizovaná. V našem případě to bude dokonce několikrát. V souboru [Položky faktury], uložíme celkovou cenu položky (množství * jednotková cena) a v souboru [Faktury], uložíme celkovou cenu faktury.

Speciální pozornost musí být věnována i vašim databázím, které budou provozovány ve víceuživatelském prostředí pomocí 4D Server. Jestliže jakákoliv vaše zpráva nebo výpočty budou potřebovat libovolný příkaz „BY FORMULA“ může být výhodné jestliže uložíte odvozená data. Užití příkazů „BY FORMULA“ způsobuje, že všechna data jsou zaváděna do paměti a přesouvána přes síť (samozřejmě pokud to neošetříte jinak pomocí speciálních procedur spuštěných na serveru což je téma pokročilejších kurzů).

10.7. Ukládání skládaných polí

Opět pohledem na seznam hledání a třídění, můžeme rozeznat např. dvě či více polí z jedné či více tabulek, které jsou potřeba velice často (denně+). Protože hledání a třídění podle kombinace hodnot těchto polí, zabírá příliš mnoho času (viz výše diskusi na téma indexy), můžete uvážit uložení kombinace hodnot těchto polí do jednoho složeného pole. Pojďme zpět k prvnímu příkladu na složená pole:

123-456-7890

123 kód společnosti, 456 – kód střediska a 7890 – pořadové číslo (kód) zakázky

Když jsme vytvářeli návrh uvedli jsme, že takovýmto složeným polím, by jsme se měli vyhnout a to z důvodů, že např. hledání a třídění podle kódu střediska je téměř nemožné. Na druhé straně, zprávy a hledání jsou často prováděna podle kombinace těchto tří polí a to zabírá příliš času. Jedno z možných řešení je uložit obojí. V tomto příkladu máme čtyři sloupce (pole): Kód společnosti, Kód střediska, Kód zakázky a Celkový kód zakázky.

10.8. Opakovaná pole

První normalizovaná forma říká: „Nesmíte mít opakovaná pole! To je výborné, ale co například s adresou o více řádcích. Máme dvě možná řešení. 1) Vytvořit pole jako textový blok. Nyní nemáme žádný problém, první řádek, druhý řádek, třetí řádek atd. - vše zapíšeme. Vzniknou úplně jiné problémy exportovat tato data pro jiné systémy se stává velice obtížné, textové pole totiž obsahují znaky nový řádek a většina jiných systémů požaduje znak nový řádek na konci záznamu, takže může být vhodné uvažovat další možnost. 2) Použití opakovaných polí. řádek Adresa 1, řádek Adresa 2, řádek Adresa 3.

10.9. První prázdná tabulka

4D má jednu další zvláštnost. Když je 4D nebo 4D Client spouštěn zavede do paměti tabulku adres první tabulky databáze. První tabulka je první založená tabulka při vytváření databáze. Tato tabulka nemůže být zaměněna za jinou tabulku. Jestliže tabulka adres této první tabulky je tabulkou adres největší tabulky v databázi, spouštění se prodlouží o dobu zavádění této tabulky. Jestliže na druhou stranu tato tabulka neobsahuje žádný záznam není tabulka adres vytvořena a do paměti se nezavede nic. Výsledkem je, že nezabereme žádný čas a paměť. Rozdíl mezi tabulkou, která neobsahuje žádný záznam a tabulkou, kde je uložen jeden záznam, ale byl vymazán je 32k.

Příklady *Optimalizace návrhu databáze*

Příklad 10.1

Velkoobchod ACI Video vás požádal o návrh databáze pro svou společnost. Chtějí sledovat své produkty, zákazníky a informace o fakturaci.

Produkty, jsou videokazety, které prodávají a jsou identifikovány jedinečným Číslem součásti. Toto Číslo součásti je přiřazováno společností ACI Video a neodpovídá libovolné identifikaci výrobce. Dále nás zajímá o každé videokazetě, název, prodejní cena, datum kdy byl film vydán na videokazetě a kategorie filmu.

Zákazníci jsou rozlišováni jedinečným názvem společnosti. ACI Video nás ujistilo, že žádný název společnosti se nemění Jestliže se tak stane, vytvářejí pro tohoto zákazníka nový záznam. Na zákazníkovi nás zajímá jméno kontaktních osob, informace o adrese a tel. číslo.

Faktury jsou rozlišovány jedinečným číslem faktury. Dále nás zajímá datum faktury, na koho byla vydána, zda byla zaplacena a jakým způsobem.

ACI Video chce být schopno hledat a třídit faktury dle největších faktur a dále určit všechny zákazníky kteří nakoupili přes 2000.

Ceny videokazet se čas od času mění v závislosti na nabídce a poptávce, době od vydání kazety a v závislosti na dalších faktorech jako je množství na skladě při pravidelných inspekcích vedení.

Navrhněte vhodnou databázi pro ACI Video včetně denormalizovaného věcného seznamu vlastností pro každý subjekt a věcného relačního diagramu ukazujícího všechny subjekty.

Příklad 10.2

Jste samostatný konzultant pro 4D. Čendovo železářství vás kontaktovalo, aby jste jim navrhl databázi. Čenda, nováček ve světě relačních databází od váš požaduje aby jste byli tvořiví při navrhování sloupců tabulek a rysů databáze.

Čendovo železářství chce sledovat následující informace:

Informace o zákaznících pro zasílání marketingových materiálů.

Účetní informace ve formě faktur.

Informace o zboží ve formě Čísla součásti, Popis, Současná prodejní cena, Stav skladu, Náklady atd.

Informace o dodavatelích ve formě Název, Adresa, Telefon atd.

Mnoho druhů zboží, které Čendovo železářství prodává je široce dostupných. Např. může nakoupit stovku hřebík od několika různých dodavatelů. Zákazníky nezájímá kdo je vyrobil. Nápověda: Čenda má pouze jedno číslo součásti pro stovku hřebík. Ale chtěl by sledovat co si který dodavatel účtuje.

Tato informace pomůže Čendovi při dalším objednávání. Rovněž chce sledovat informace o identifikačních číslech zboží jednotlivých dodavatelů, aby objednávkový proces byl co nejjednodušší. Rovněž si chce Čenda vést poznámky o kvalitě jednotlivého zboží od různých dodavatelů.

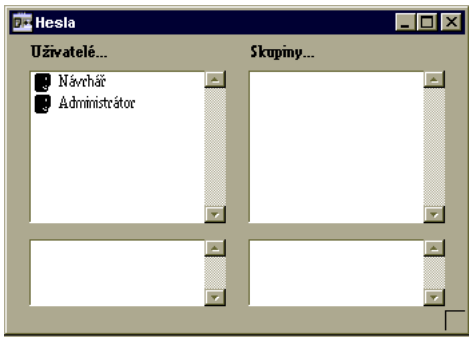
Navrhněte vhodnou databázi pro Čendovo železářství včetně denormalizovaného věcného seznamu vlastností, pro každý subjekt a věcného relačního diagramu ukazujícícho všechny subjekty.

11. Fyzické vytvoření struktury databáze

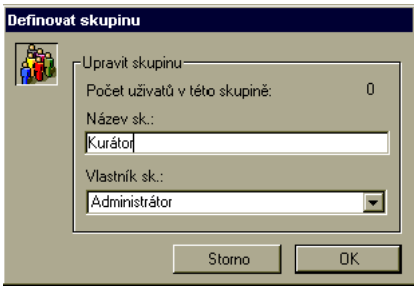
Nyní se budeme zabývat fází vytváření databáze. Až dokončíte Výkaz návrhu databáze, zjistíte že samotné vytvoření databáze je snadné.

11.1. Vytváření skupin

Prvním krokem při vytváření databáze je vytvoření skupin, které jste navrhli v Seznamu přístupových práv. Potřebujete jednu skupinu pro každou úroveň přístupových práv vašeho návrhu. Užijeme příklad z předchozí kapitoly tzn. že budeme potřebovat dvě skupiny: Kurátor a Prodejci. Vytvoření skupiny je jednoduché. Přepněte se do prostředí návrháře a zvolte Nástroje Hesla, otevře se Editor hesel, který je zobrazen níže:



Nyní zvolte Hesla - Nová skupina. Otevře se dialogové okno pro definici skupin.



Do tohoto dialogového okna napište název skupiny. To je vše co potřebujete v této fázi provést.

11.2. Konverze subjektů do tabulek

Dalším krokem je vytvoření tabulek. Je to velice jednoduché, zvolte Struktura Nová tabulka, kurzor se změní na následující obrázek:

Klepněte kamkoliv uvnitř okna struktury. V okně struktury se objeví nová tabulka v místě kam jste klepli:

Se stále vybranou tabulkou zvolte Struktura - Vlastnosti tabulky. Otevře se dialogové okno pro definici vlastností tabulky:

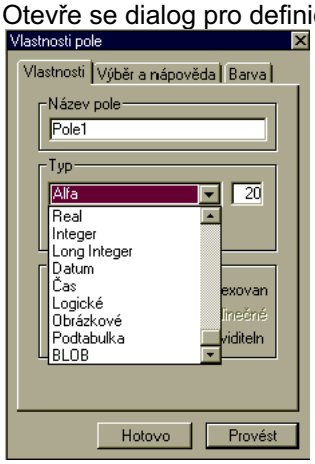
Pojmenujte tabulku a přiřaďte jí přístupová práva příslušných skupin. Při přiřazování přístupových práv vám pomůže následující tabulka:

- Načíst = Číst
- Uložit = Upravovat
- Přidat = Přidat
- Vymazat = Vymazat

Pokud nezamýšlíte programovat ve skupině více programátorů pomocí 4D Server, ignorujte Přístup k tabulce - Vlastník.

11.3. Přidání polí

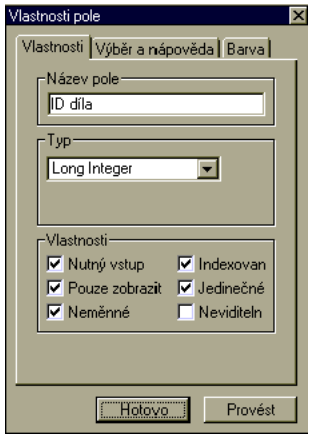
Přidání polí do tabulek je rovněž jednoduché. Zvolte Struktura - Nové pole nebo jednoduše poklepejte uvnitř grafického znázornění tabulky. Otevře se dialog pro definici vlastností pole jak je ukázáno níže:



Poz.: Textový řetězec pevné délky v materiálech znázorňovaný jako Řetězec(x) je ve 4D nazýván Alfa (Alfanumerický sloupec), ostatní typy jsou stejné jak jsme doposud používali.

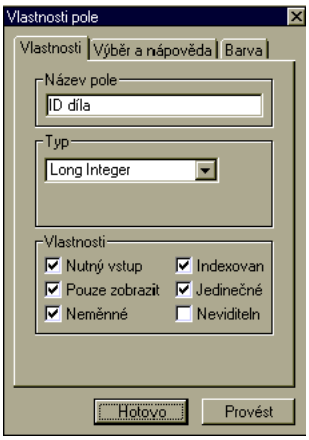
11.3.1. Vytvoření pole primárního klíče

Podle konvence je primární klíč vždy první pole tabulky. Jestliže primární klíče tabulky obsahuje pouze jedno pole, je tato operace snadná. Přidejte primární klíč jako první pole a přidělte mu vlastnosti indexované, povinné, jedinečné a neměnné. Jestliže pole je umělý klíč, který systém bude vytvářet, můžete mu přidělit vlastnost pouze zobrazit. (Poznámka: při volbě těchto vlastností nenarazíte na žádné dodatečné problémy, kromě toho, že pokud chcete, aby pole bylo povinné a jedinečné musí být rovněž indexované. Proto rovnou vyberme všechny výše zmíněné možnosti.) Následující obrázek ukazuje definovaný primární klíč:



Jeden neintuitivní aspekt zakládání polí v tabulkách 4D je případ složeného primárního klíče. 4D nepodporuje explicitně složené primární klíče. Aby jste umožnili jedinečnost budete potřebovat vytvořit dodatečné pole k zdvojenému uložení dat primárního klíče. Toto pole může být neviditelné, jestliže si přejete, takže uživatel toto pole normálně neuvidí.

Následující obrázek ukazuje příklad pole pro uložení složeného primárního klíče:



Povšimněte si, že vlastnost jedinečné byla pro toto pole zvolena. To je celý fígl. 4D musí obsahovat jedno pole s jedinečnou hodnotou. Pole, která jsou komponenty primárního klíče by měla mít vlasnost povinné, nelze upravit a indexované.

V tomto případě hodnota ukládaná v poli ID Souč Kód obch bude něco jako „123-A“, kde „123“ reprezentuje textový ekvivalent IDSoučásti a „A“ reprezentuje Kód obchodu.

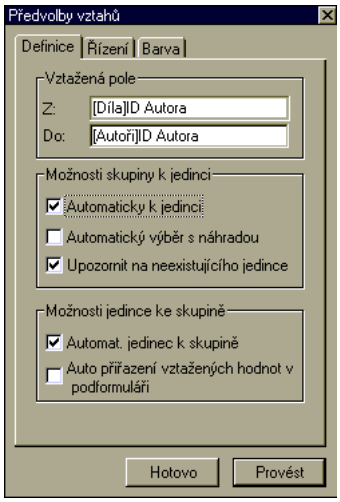
11.3.2. Vkládání neklíčových polí

Vložení dalších polí do struktury neobsahuje žádný další problém. Jednoduše procházejte váš návrh a vkládejte odpovídající pole pro každý sloupec. Definujte správně názvy polí, typ a délku. Na následujícím obrázku je struktura pro databázi Muzeum popsaná dříve:

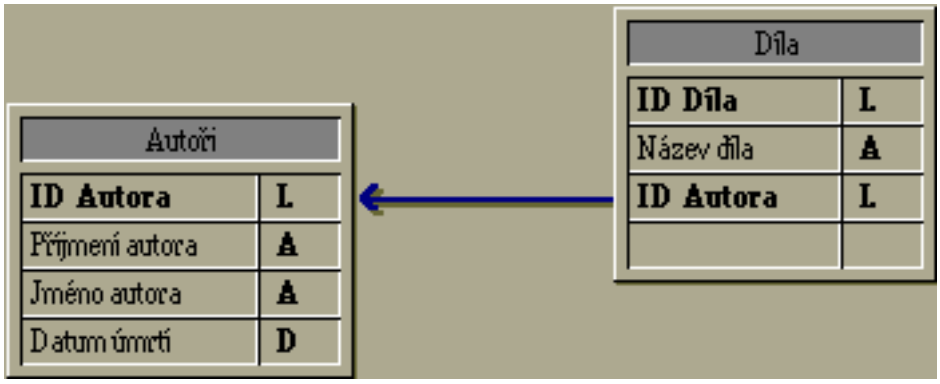
11.4. Vytvoření vztahů

4D obsahuje rys, který vám dovolí automaticky udržovat vztahy mezi soubory. Tento rys požaduje, aby vztahy byly vytvořeny v editoru struktury. Jejich vytvoření je snadné. Klepněte na pole cizí klíč v tabulce skupin a se stisknutým tlačítkem myši táhněte myší na pole primárního klíče v tabulce jedinců. Zde tlačítko myši uvolněte.

Tato operace otevře dialog pro definování vztahu:



Po ukončení definice vztahu, klepněte na OK a 4D zobrazí ve struktuře následující čáru, zobrazující vztah:



Tato čára indikuje, že relace bude automaticky udržována pomocí 4D, šipka ukazuje směr k souboru jedinců.

Příklady Fyzické vytváření struktury

Příklad 11.1

Vraťte se k příkladu 5.8 a 6.6. Vytvořte strukturu této databáze. Nezabývejte se problémem bezpečnosti.

Příklad 11.2

Vraťte se k příkladu 5.9 a 6.7. Vytvořte strukturu této databáze. Nezabývejte se problémem bezpečnosti.

Příklad 11.3

Vraťte se k příkladu 5.10 a 6.8. Vytvořte strukturu této databáze. Nezabývejte se problémem bezpečnosti.

Příklad 11.4

Vraťte se k příkladu 5.11 a 6.9. Vytvořte strukturu této databáze. Nezabývejte se problémem bezpečnosti.

12. Závěrečný příklad

12.1. Baškirská státní univerzita

Byli jste najmuti k vytvoření počítačového systému Baškirské státní univerzity pro registraci v semestrech a sledování dalších dat.

- Informace o studentech, kde jsou ubytováni, jejich akademické a vedlejší vzdělání, tituly, zapsané přednášky
- Informace o přednáškách více popisně, umístění třídy, požadované studijní materiály, potřebná zařízení, studijní vedoucí...
- Informace o katedrách, vedoucí katedry, příslušnost k fakultě a pravidělné studijní setkání katedry.
- Informace o fakultách pro rozesílání dopisů, sledování udělovaných titulů atd.

Následující je výtah z dopisu, kerý jste obdrželi s požadavkem na vaše služby

"Máme přibližně 50,000 a nabízíme více než 2,000 kurzů v 20 katedrách, které udělují 45 různých titulů. Velice potřebujeme systém úsporný v ukládání dat, protože počítače jsou stále velice drahá záležitost. Chtěli bychom sledovat i historii přednášek nabízenou katedrami až do doby pádu carismu.Od naší vlády máme slíbeno zakoupení 6 uživatelského 4d serveru a 7 počítačů PC Pentium II. Server bude mít pevný disk 2GB.

Studenti	Přednášky	Katedry	Fakulty
Jméno	Název	Název	Název
Adresa	Kredity	Vedoucí	Adresa
Telefon	Datumy	Fakulta	Katedry
Tituly	Čas	Datum schůzky	Udělované tituly
Kreditů	Katedra	Čas schůzky	Umístění
Katedra	Instruktor	Konání schůzky	Telefon
Akad.vzdělání	Knihy	Přednášky	Úřední hodiny
Ost .	Místo konání		Home Phone
Vzdělání	Prerequisites		Přednášky
Přednášky	(Fr,So,Jr,Sr)	Učební zařízení	Publikace
Knihy	Studijní materiály		
Materiály			

A Reference

Reference 1

Code Complete, a Practical Handbook of Software Construction, by Steve McConnell (Microsoft Press, 1993)

Tato kniha je jednou z nejkompletnějších knih o vytváření softwaru, kterou jsme našli. Je to povinná četba pro všechny programátory firmy Microsoft.

Obsahuje rovněž užitečné tipy pro návrhy SW, orientované především na to, aby programátor poznal, zda byl návrh vytvořen správně.

Reference 2

Relational Database Design, a PractitionerŐs Guide, by Charles J. Wertz (CRC Press 1993)

Tato kniha je dobrou učebnicí pro mírně pokročilé v navrhování relačních databází užívá konvenční strukturální analýzu. Je poměrně dost technická. Z této knihy jsme hodně čerpali, ale se zjednodušeními pomocí cvičení.

Reference 3

An Introduction to Database Systems, Volume I, by C.J. Date (Addison Wesley 1990)

Tato kniha je pravděpodobně nejrozšířenější učebnicí v teorii relačních databází. Je orientována především na lidi, kteří navrhují systémy pro řízení databází a ne na ty, kteří v nich navrhují své databáze. Obsahuje však rovněž část o samotném navrhování databází. Je napsána velice technickým jazykem na vysoké úrovni. Samozřejmě to není čtení do postele.

Reference 4

An Introduction to Database Systems, Volume II, by C.J. Date (Addison Wesley 1983)

Tato kniha dokončuje databázový kurs pana Date. Stejně jako kniha 1 je na vysoké úrovni.

Reference 5

Rapid System Development Using Structured Analysis and Relational Technology, by Chris Gane (Prentice Hall 1989)

Gane je původce myšlenky systémů RAD pro rychlý vývoj aplikací. Tato kniha obsahuje jeho metodiku, která vychází z JAD (Joint Application Development). Tato kniha obsahuje rovněž velice solidní úvod do navrhování databází. Pokud jste student, kerý se chce na slušné úrovni seznámit v relačním konceptem rychle a snadno, je to pro vás asi nejlepší literatura. V tomto kurzu jsme z tohoto materiálu rovněž hodně čerpali.

.

Reference 6

The Relational Model for Database Management, Version 2, by E.F. Codd (Addison Wesley 1990)

Codd je autorem relačního modelu řízení databází. Ve své práci v IBM během let 1970 byl odpovědný za návrh systému, který je znám jako DB2. Jeho příspěvek k tomuto oboru je neocenitelný. Bohužel jeho kniha je téměř nečitelná, pokud nemáte titul PhD z počítačových věd. Pokud nejste velmi technicky erudovaní vyberte si raději z jiných referencí zde uvedených.

Inforce

B Řešení příkladů

Kapitola 2

Příklad 2.1

Soubor dopisů univerzity není databáze, protože není strukturována, aby to byla databáze musí být jeden dopis organizován/Strukturován stejně.

Dopis nelze nalézt jednoduše, protože databáze není strukturována, je potřeba přečíst celý dopis, aby se zjistilo jméno studenta a jeho údaje.

Nelze je třídit bez pročtení každého dopisu.

Soubor trhacích kalendářů je datbáze, protože formulář má pevnou strukturu organizovanou do sloupců a řádek.

Jeden dopis studenta lze již najít snadněji prohlížením jediného přesně umístěného údaje ve formulářích (samozřejmě ne tak snadno, jako kdyby byly formuláře pořízeny do 4D)

Administrativa univerzity může nyní snadněji přetřídit formuláře podle libovolné kolonky ve formuláři.

Příklad 2.2

Struktura těchto dat není tabulka, protože má ve skutečnosti tři rozměry (číslo součásti, kód obchodu, součástí v obchodu) a tabulky mají pouze dva rozměry.

Hledání lze provést pouze sekvenčně projitím všech řádek (předpokládáme, že neexistuje žádný soubor indexů, odkazů, kde je co, podporující tento druh struktury)

Přetřídění tohoto druhu tabulky bude vyžadovat spoustu programátorského úsilí. Např, setřídění podle množství, odložení dat do jiné struktury, která pak bude tříděna podle součástí.

Povšimněte si rovněž spousty volného místa v tomto druhu ukládání na disk. Technicky je to matice.

Tento problém se řešil v hierarchických databázích, které předcházely relačním.

Příklad 2.3

Primární klíč [Položka] je [Položka]Položka číslo

Primární klíč [Objednávka] je [Objednávka]Položka číslo

Je zde položka, která by mohla být přirozený klíč [Položka]Popis. Přirozený primární klíč to bude tehdy, jestliže se nikdy nemění.

Oba primární klíče tabulek jsou umělé

Tabulky jsou ve vztahu podle sloupců Položka číslo. Vztah je 1:M a soubor jedinců je [Položka], protože může být více objednávek k jedné položce, ale pouze jedna konkrétní položka v řádku [Objednávka]. Drobný defekt, kerý je zde s objednávkami je korigován v dalším příkladu.

Cizí klíč je [Objednávka]Položka číslo

Příklad 2.4

Primární klíč [Položka] je [Položka]Položka číslo

Primární klíč [Objednávka] je [Objednávka]Pořadové číslo

Primární klíč [Položky objednávky] je [Položky objednávky]Položka číslo+[Položka objednávky]Pořadové číslo. Je to složený klíč z položky zboží a čísla objednávky.

Kandidát na přirozený klíč je opět [Položka]Popis, pokud se nemění a pak by mohl pracovat jako primární klíč. [Objednávka]Pořadové číslo je možno považovat za přirozený klíč, pokud vezmeme, že objednávky jsou a musí být číslovány svým pořadovým číslem.

Jsou zde dva vztahy 1:M, které tvoří dohromady jeden vztah M:M

Vztah 1:M je [Položka] [Položky objednávky], kde [Položka] je soubor jedinců a [Položky objednávky] soubor skupin ([Položky objednávky] mohou obsahovat vícekrát jednu položku zboží)

Vztah 1:M je [Objednávka] [Položky objednávky], kde [Objednávka] je soubor jedinců a [Položky objednávky] soubor skupin. (Pro jednu objednávku je více položek objednávky)

Vztah M:M je mezi tabulkami [Položka] a [Objednávka], libovolná položka /zboží/ může být v libovolné objednávce.

Jsou zde dva cizí klíče [Položky objednávky]Položka číslo a [Položka objednávky]Pořadové číslo

Příklad 2.5

Jsou zde dva složené sloupce [Osoba]Jméno a [Osoba]Adresa. Tyto sloupce je nutno rozdělit na sloupce atomární.

[Osoba]Jméno na [Osoba]Jméno (křestní) a [Osoba]Příjmení.

[Osoba]Adresa na [Osoba]Ulice, [Osoba]Město, [Osoba]PSČ

Nelze vyhledat osobu s konkrétním příjmením, protože nejste schopni určit kde začíná příjmení ve sloupci. Jiné řešení, než přeorganizovat strukturu není. Pokus o určení příjmení, že začíná vždy za mezerou není spolehlivý způsob.

Obdobně nelze spolehlivě nalézt osoby z Prahy, protože neurčíte přesně kde začíná město.

Nelze třídit podle PSČ z téhož důvodu.

.

Kapitola 4

Příklad 4.1

Položka

Název sloupce	Typ dat	Klíče
Položka číslo	Longint	PK
Název položky	Řetězec (80)	
Cena	Real	

Objednávka

Název sloupce	Typ dat	Klíče

Objednávka číslo	Longint	PK
Zákazník číslo	Longint	
Datum	Datum	

Položky objednávky

Název sloupce	Typ dat	Klíče
Objednávka číslo	Longint	PK1 CK1
Položka číslo	Longint	PK2 CK2
Množství	Real	

Jste schopni mnohem jasněji vidět vztahy mezi tabulkami a sloupci a primární klíče jednotlivých tabulek.

Až se tabulky stanou složitější (několik souborů a mnoho sloupců řekněme nad 10) je metoda příkladových dat již nepřehledná a již musíte použít zobrazení sloupců pod sebou ve věcném seznamu vlastností, aby jste mohli vidět sloupce tabulky jako celek.

Kapitola 5

Příklad 5.1

Zde jsou funkční závislosti:

[Sklad]Skladové číslo -> [Sklad]Druh

[Sklad] Skladové číslo -> [Sklad]Odrůda

[Sklad] Skladové číslo -> [Sklad]Acquired

[Sklad]Odrůda -> [Sklad] Druh

[Plán krmení]Odrůda -> [[Plán krmení]Druh

[Plán krmení]Odrůda -> [Plán krmení]Krmivo

[Plán krmení]Odrůda -> [Plán krmení]Množství

Druhá otázka je nápovědou. Tato tabulka není normalizovaná. Porušuje 3NF. Naučíte se o ní později.

Povšimněte si, že můžete určit druh z odrůdy ve dvou tabulkách. Tak:

[Sklad]Odrůda -> [Sklad]Druh

a

[Plán krmení]Odrůda -> [[Plán krmení]Druh

To znamená, že je porušena 3NF

Sloupce [Sklad]Druh a [[Plán krmení]Druh jsou závislé v obou tabulkách na dvou sloupcích.

Příklad 5.2

V této stuktuře jsou tři opakované skupiny:

- Dohodnutá návštěva datum a čas

- Příznaky
- Diagnóza
- Kombinace logických (Ano/Ne) a datumových sloupců o nemocech a očkování (Tato je zde lstivě zakrývána různými názvy nemocí a očkování)

Existuje určitě více než jedno správné řešení struktury. Následující popisuje jedno z nich.

Dohodnutá návštěva a čas musí být vyděleny do zvláštní tabulky [Návštěvy], která bude ukládat informace o jednotlivých návštěvách. Tímto jsme odstranili omezení na počet návštěv.

Jestliže není požadavek na hledání určitých příznaků, může být skupina příznaků nahrazena textovým sloupcem. Protože příznaky se nejpravděpodobněji vztahují k jednotlivým návštěvám umístíme tento sloupec do tabulky [Návštěvy]. V jiném případě by musela tato skupina tvořit samostatnou tabulku.

Jestliže není požadavek pro vyhledávání určité diagnózy nahradíme tuto skupinu textovým sloupcem. Protože diagnózy se nejpravděpodobněji vztahují k jednotlivým návštěvám umístíme tento sloupec do tabulky [Návštěvy]. V jiném případě by měla tato skupina tvořit samostatnou tabulku.

Kombinace logických a datumových sloupců o nemocech a očkování bude tabulkou Historie pacienta, toto je největší zisk. Přesunutím těchto dat do samostatné tabulky můžeme reorganizovat a modifikovat historii pacienta kdykoliv. Ukládání informací o nemocech a očkování, keré pacient nikdy neměl je přepych. Tato tabulka nám umožní ukladádat i nemoci a očkování, na které doktor nemyslel v době vytváření databáze, možná ani neexistovaly a není třeba navrhovat databázi znovu a což teprve programovat. Pokud některá nemoc a očkování v záznamech chybí pacient je neměl.

Zde je nová struktura:

Záznam pacienta

Název sloupce	Typ dat	Klíč
ID pacienta	Longint	PK
Příjmení pacienta	Řetězec (45)	
Jméno pacienta	Řetězec (45)	
Datum narození	Datum	
Alergie	Text	

Návštěvy

Název sloupce	Typ dat	Klíč
ID pacienta	Longint	PK1 CK1
Datum návštěvy	Datum	PK2
Čas návštěvy	Čas	
Příznaky	Text	
Diagnoza	Text	

Historie

Název sloupce	Typ dat	Klíč
ID pacienta	Longint	PK1 CK1

Název položky	Řetězec (45)	PK2
Datum nemoci/očkování	Date	PK3

Systém jako je tento je nazýván řízená data. Je to jeden z nejúčinnějších rysů relační databáze.

Tento příklad ukazuje na nebezpečí automatizace papírové dokumentace. Zatímco je Váš doktor velice spokojen se svou papírovou dokumentací, kterou plně využívá, musí porozumět (nebo být doveden k porozumění), že papírový a počítačový systém jsou dvě zcela odlišné věci. Pouze „zpočítačovaný“ papírový systém bude nejpravděpodobněji pohroma. (Na druhou stranu fungující papírové formuláře poskytnou výbornou informaci o podstatě dat, tak jak jsme zde i viděli.)

Příklad 5.3

V navrhované datové struktuře jsou dvě opakované skupiny.

První je celkové prodeje – Celkem a průměr k sledovaný měsíčně, vytvořme pro ně tabulku [Měsíční data].

Druhá je čtvrtletní sledování Celkem a průměr k, vytvořme pro ně tabulku [Čtvrtletní data]

Tento druh chyby je velice běžný pro lidi, kteří přicházejí do styku pouze s tabulkovými editory a snaží se vytvořit návrh relační databáze.

Je více než jedno správné řešení. Jedno si popíšeme.

Zkombinujeme všechny tři tabulky do jedné tabulky, která bude sledovat data na základě datumů Od a Do. Pak pouze vložíme správné datumy a finanční oddělení může provádět analýzy na základě periody kterou si vybere (rok, více let, pololetí..)

Je více než jedno správné řešení, následující je jedno z nich:

Zde je normalizovaná struktura:

Opakovaná data

Název sloupce	Typ dat	Klíče
Datum od	Datum	PK1
Datum do	Datum	PK2
Celkové prodeje	Real	
Průměr prodejů	Real	

Jedna otázka, kterou určitě položíte je jak budeme provádět roční průběžný průměr na základě libovolné periody. Odpověď je jednoduchá.

Jestliže je perioda menší než 6 měsíců, rozdělíme minulý rok do stejných úseků dle této periody, jestliže nebude výsledek celočíselný ponecháme [Opakovaná data]Průměr prodejů nevyplněný

Pokud ano vypočteme data na základě této periody.

Je-li větší než 6 měsíců ponecháme [Opakovaná data]Průměr prodejů nevyplněný

Je to jiný případ programování řízených dat.

Příklad 5.4

Ať byl vývojářem pojišťovací společnosti kdokoliv, nerozuměl příliš normalizaci.

Zde jsou funkční závislosti:

Inforce

ID agenta-> Jméno agenta

ID agenta + Politika označení -> ID zákazníka

ID agenta + Politika označení -> Příjmení zákazníka

ID agenta + Politika označení -> Jméno zákazníka

ID zákazníka -> Příjmení zákazníka

ID zákazníka -> Jméno zákazníka

Vidíme, že byla porušena 2NF. Jméno agenta je závislé pouze na ID agenta a to není vnitřní primární klíč [Politika]. Takže tato tabulka není v 2NF.

Tabulka není v 3NF ještě i z jiného důvodu, že není v 2NF. Příjmení a Jméno zákazníka je závislé na ID zákazníka a tím je dosaženo zprostředkované závislosti na vnitřním klíči tabulky.

Zde jsou správně normalizované tabulky:

Politika

Název sloupce	Typ dat	Klíče
Carrier ID	Longint	PK1 CK1
Politika označení	Řetězec (10)	PK2
ID zákazníka	Longint	CK2

Agent

Název sloupce	Typ dat	Klíče
ID agenta	Longint	PK
Jméno agenta	Řetězec (45)	

Zákazník

Název sloupce	Typ dat	Klíče
ID zákazníka	Longint	PK
Příjmení zákazníka	Řetězec (45)	
Jméno zákazníka	Řetězec (45)	

Příklad 5.5

Tabulka je v 1NF, nejsou zde opakované skupiny.

Tabulka je v 2NF, je pouze jeden sloupec s primárním klíčem, takže všechny sloupce jsou závislé na vnitřním primárním klíči.

Zde jsou funkční závislosti:

ID zaměstnance -> Jméno zaměstnance

ID zaměstnance -> ID střediska

ID zaměstnance -> Název střediska

ID střediska -> Název střediska

Protože Název střediska je funkčně závislý na ID střediska je porušena 3NF. (3NF – žádný neklíčový sloupec není závislý na jiném neklíčovém sloupci)

Přemístěním sloupce názvu střediska do jiné tabulky tento problém vyřešíme.

Zde je normalizovaná struktura :

Zaměstnanci

Název sloupce	Typ dat	Klíče
ID zaměstnance	Longint	PK
Jméno zaměstnance	Řetězec (45)	
ID střediska	Řetězec (3)	CK1

Střediska

Název sloupce	Typ dat	Klíče
ID střediska	Řetězec (3)	PK
Název střediska	Řetězec (20)	

Příklad 5.6

První tabulka porušuje 3NF. Kapacita úlu je funkčně závislá na ID úlu, takže nepatří do této tabulky. Správná struktura je :

Včela

Název sloupce	Typ dat	Klíče
ID včely	Longint	PK
Jméno včely	Řetězec (20)	
ID úlu	Longint	CK1

Druhá tabulka porušuje 2NF. Jméno osoby je funkčně závislé pouze na ID osoby a proto nepatří do této tabulky. Správná struktura je:

Žihadla

Název sloupce	Typ dat	Klíče
ID včely	Longint	PK1
ID osoby	Longint	PK2
Počet žihadel	Integer	

Zde jsou další tabulky doplňující tuto normalizovanou strukturu:

Úly

Název sloupce	Typ dat	Klíče
ID úlu	Longint	PK
Kapacita úlu	Longint	
Telefon do úlu	Řetězec (10)	

Květiny

Název sloupce	Typ dat	Klíče
ID květiny	Longint	PK
Typ květiny	Řetězec (45)	
Barva	Řetězec (10)	

Opylení

Název sloupce	Typ dat	Klíče
ID včely	Longint	PK1 CK1
Datum opylení	Datum	PK2
Čas opylení	Čas	PK3
ID květiny	Longint	FK2

Osoby

Název sloupce	Typ dat	Klíče
ID osoby	Longint	PK
Jméno osoby	Řetězec (80)	

Tyto tabulky spolu s předchozími vytvářejí úplnou normalizovanou strukturu, protože jsou zde zahrnuty všechny funkční závislosti a žádné z pravidel normalizace není porušeno.

Příklad 5.7

Zde jsou funkční závislosti :

ID ošetřovatele -> Jméno ošetřovatele

ID ošetřovatele -> Velikost ošetřovatele

ID vepřína -> Ocenění vepřína

ID vepřína -> Kapacita vepřína

ID vepřína -> Swineherd ID

ID vepře -> ID vepřína

ID vepře -> Váha vepře

ID vepře -> Ocenění vepře

Zde je normalizovaná struktura:

Ošetřovatel

Název sloupce	Typ dat	Klíče
ID ošetřovatele	Longint	PK
Jméno ošetřovatele	Řetězec (80)	
Velikost	Integer	

Vepřín

Název sloupce	Typ dat	Klíče
ID vepřína	Longint	PK
Ocenění vepřína	Real	
Kapacita vepřína	Real	
ID ošetřovatele	Longint	CK1

Vepři

Název sloupce	Typ dat	Klíče
ID vepře	Longint	PK
Ocenění vepře	Real	
Váha vepře	Real	
ID vepřína	Longint	CK1

Tyto tabulky vytvářejí úplnou normalizovanou strukturu, protože jsou zde zahrnuty všechny funkční závislosti a žádné z pravidel normalizace není porušeno.

Příklad 5.8

Zde jsou funkční závislosti :

ID iluzionisty -> Jméno iluzionisty

ID iluzionisty -> Sazba iluzionisty

Název lóže -> Adresa lóže

Název lóže -> Telefon lóže

ID kouzla -> Název kouzla

ID kouzla -> Stupeň ohromení

ID kouzla + ID iluzionisty -> Úroveň provedení

Jsou zde ještě nějaké problémy. Protože se Název lóže může měnit, nemůže Název lóže sloužit jako primární klíč. Rovněž musíme vytvořit M:M vztah mezi [Iluzonisti] a [Lóže].

Zde je normalizovaná struktura dat:

Iluzionisti

Název sloupce	Typ dat	Klíče
ID iluzionisty	Longint	PK
Jméno iluzionisty	String (80)	
Sazba iluzionisty	Real	

Lóže

Název sloupce	Typ dat	Klíče
ID lóže	Longint	PK
Název lóže	String (80)	
Adresa lóže	String (80)	
Telefon lóže	String (10)	

Kouzla

Název sloupce	Typ dat	Klíče
ID kouzla	Longint	PK
Název kouzla	String (80)	
Stupěň ohromení kouzlem	Integer	

Lóže iluzionisty

Název sloupce	Typ dat	Klíče
ID iluzionisty	Longint	PK1 CK1
ID lóže	Longint	PK2 CK2

Kouzla iluzionisty

Název sloupce	Typ dat	Klíče
ID iluzionisty	Longint	PK1 CK1
ID kouzla	Longint	PK2 CK2
Úroveň provedení	Integer	

Tyto tabulky vytvářejí úplnou normalizovanou strukturu, protože jsou zde zahrnuty všechny funkční závislosti a žádné z pravidel normalizace není porušeno.

Příklad 5.9

Zde jsou funkční závislosti:

ID trenéra -> Jméno trenéra

ID trenéra -> Velikost trenéra

ID stáje -> Ocenění stáje

ID stáje -> Kapacita stáje

ID stáje -> ID trenéra

ID koně -> ID stáje

ID koně -> Váha koně

ID koně -> Ocenění koně

Zde je normalizovaná struktura:

Trenér

Název sloupce	Typ dat	Klíče
ID trenéra	Longint	PK
Jméno trnéra	String (80)	
Velikost trenéra	Integer	

Stáj

Název sloupce	Typ dat	Klíče
ID stáje	Longint	PK
Ocenění stáje	Real	
Kapacita stáje	Real	
ID trenéra	Longint	CK1

Kůň

Název sloupce	Typ dat	Klíče
ID koně	Longint	PK
Ocenění koně	Real	
Váha koně	Real	
ID stáje	Longint	CK1

Tyto tabulky vytvářejí úplnou normalizovanou strukturu, protože jsou zde zahrnuty všechny funkční závislosti a žádné z pravidel normalizace není porušeno.

Příklad 5.10

Zde jsou funkční závislosti:

ID řemeslníka -> Jméno řemeslníka

ID řemeslníka -> Sazba řemeslníka

Název party -> Adresa party

Název party -> Telefon party

ID řemesla -> Název řemesla

ID řemesla -> Obtížnost řemesla

ID řemesla + ID řemeslníka -> Řemeslná zručnost

Jsou zde ještě nějaké problémy. Protože se Název party může měnit, nemůže Název party sloužit jako primární klíč. Rovněž musíme vytvořit M:M vztah mezi [Řemeslníci] a [Parta].

Zde je normalizovaná struktura dat:

Řemeslníci

Název sloupce	Typ dat	Klíče
ID řemeslníka	Longint	PK
Jméno řemeslníka	String (80)	
Sazba řemeslníka	Real	

Parta

Název sloupce	Typ dat	Klíče
ID party	Longint	PK
Název party	String (80)	
Adresa	String (80)	
Telefon	String (10)	

Řemeslo

Název sloupce	Typ dat	Klíče
ID řemesla	Longint	PK
Název řemesla	String (80)	
Obtížnost řemesla	Integer	

Party řemeslníka

Název sloupce	Typ dat	Klíče
ID řemeslníka	Longint	PK1 CK1
ID party	Longint	PK2 CK2

Řemesla řemeslníka

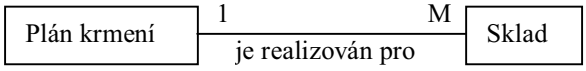
Název sloupce	Typ dat	Klíče
ID řemeslníka	Longint	PK1 CK1
ID řemesla	Longint	PK2 CK2
Řemeslná zručnost	Integer	

Tyto tabulky vytvářejí úplnou normalizovanou strukturu, protože jsou zde zahrnuty všechny funkční závislosti a žádné z pravidel normalizace není porušeno.

Kapitola 6

Příklad 6.1

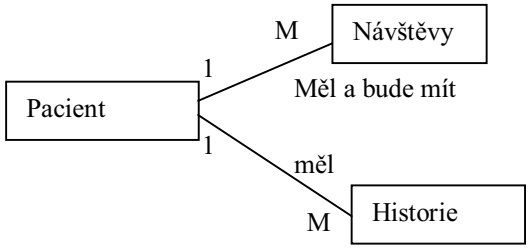
Zde je VRD:



Pamatujte, že tato struktura není normalizovaná. Podrobnosti viz řešení příkladu 5.1.

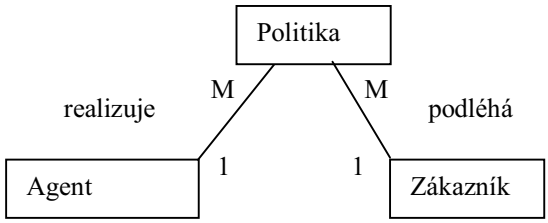
Příklad 6.2

Zde je VRD:



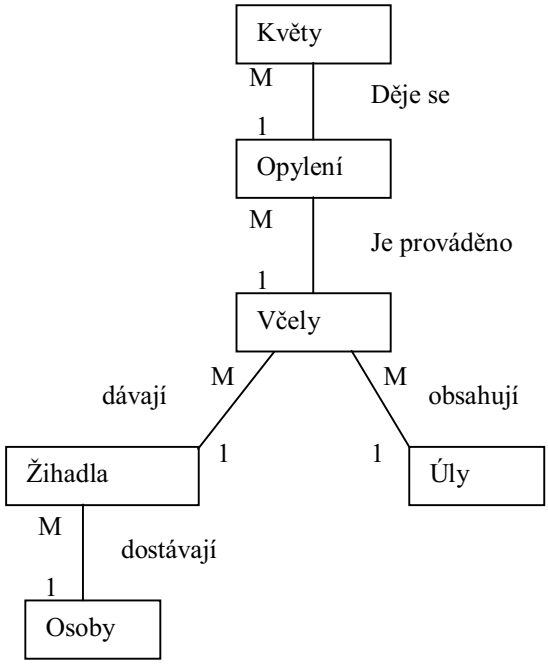
Příklad 6.3

Zde je VRD:



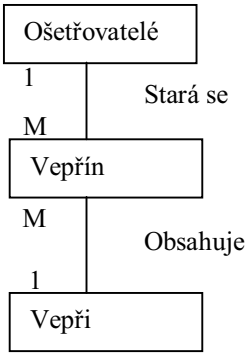
Příklad 6.5

Zde je VRD:



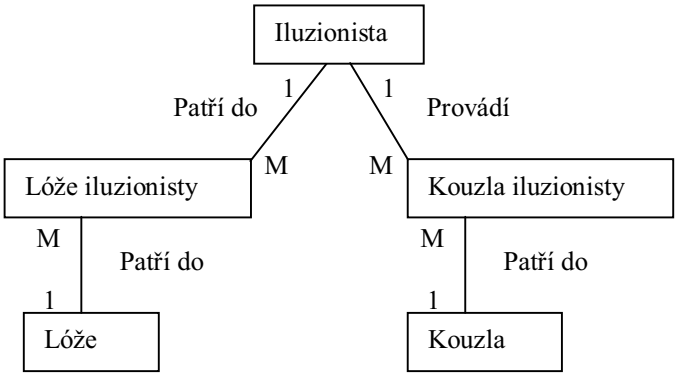
Příklad 6.6

Zde je VRD:



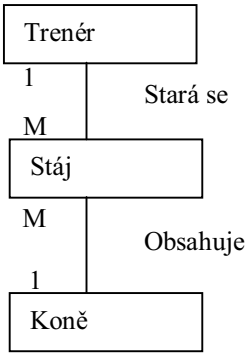
Příklad 6.7

Zde je VRD:



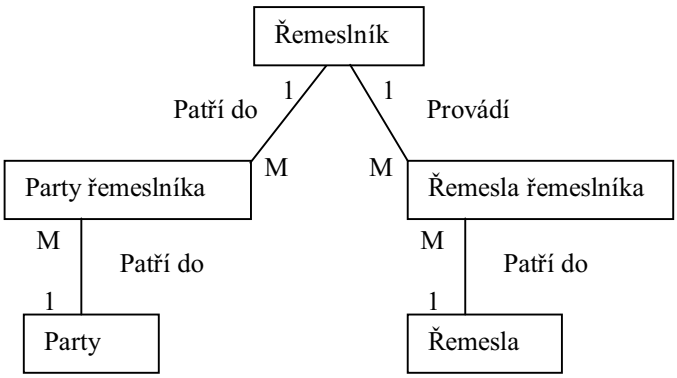
Příklad 6.8

Zde je VRD :



Příklad 6.9

Zde je VRD:



Příklad 6.10

Zde jsou funkční závislosti:

Přezdívka piráta -> Křestní jméno piráta

Přezdívka piráta -> Stupeň hrůzostrašnosti

Přezdívka piráta -> Datum prvního lupu

Přezdívka piráta -> Název pitátské lodi

Název pitátské lodi -> Počet stěžňů PL

Název pitátské lodi -> Počet kanónů PL

Název pitátské lodi -> Datum spuštění PL

Název pitátské lodi -> Kapitán pirátské lodi

Název obchodní lodi -> Výtlak OL

Název obchodní lodi -> Počet stěžňů OL

Název obchodní lodi + Název pirátské lodi+ Datum útoku -> Potopení obchodní lodi

Název obchodní lodi + Název pirátské lodi+ Datum útoku -> Počet mužů přes palubu

Zde je normalizovaná struktura:

Piráti

Název sloupce	Typ dat	Klíče
Přezdívka piráta	String (80)	PK
Křestní jméno piráta	String (80)	
Stupeň hrůzostrašnosti	Integer	
Datum první akce	Date	
Název pirátské lodi	String (80)	FK1

Pirátské lodi (PL)

Název sloupce	Typ dat	Klíče
Název pirátské lodi	Řetězec (80)	PK
Počet stěžˇmů PL	Integer	
Počet kanónů PL	Integer	
Datum spuštění PL	Datum	
Kapitán pirátské lodi	Řetězec (80)	CK1

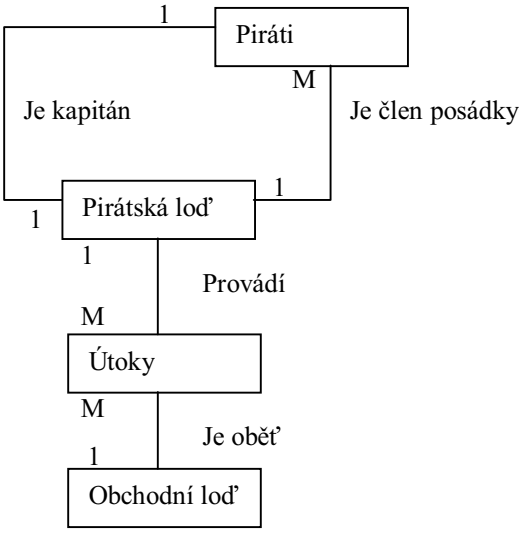
Obchodní lodi (OL)

Název sloupce	Typ dat	Klíče
Název OL	Řetězec (80)	PK
Počet stěžňů OL	Integer	
Výtlak OL	Real	

Útoky

Název sloupce	Typ dat	Klíče
Název OL	Řetězec (80)	PK1 CK1
Název PL	Řetězec (80)	PK2 CK2
Datum útoku	Datum	PK3
Potopení obchodní lodi	Boolean	
Počet mužů přes palubu	Integer	

A zde je odpovídající VRD:



Kapitola 7

Příklad 7.1

Existuje více než jedno řešení tohoto problému. Následující popisuje jedno ze správných řešení.

Lojza může být velice šikovný právník, ale jako návrhář databází na úrovni uklízečky. (Samozřejmě mu to nemůžete říci.)

Navrženy jsou složené sloupce, keré lze snadno rozdělit:

PSČ, město klienta

Jméno obhájce

Jméno pr.zást.druhé strany

Jméno doktora

Tyto sloupce musí být nahrazeny atomárními sloupci, obsahující tatáž data:

Město klienta

PSČ klienta

Jméno obhájce

Příjmení obhájce

Město obhájce

PSČ obhájce

Jméno PZ druhé strany

Příjmení PZ druhé strany

Jméno doktora

Příjmení doktora

Připomeňme si pravidlo pro výběr primárního klíče:

- Primární klíč musí jednoznačně identifikovat řadu
- Primární klíč musí být neměnný (t.j. stálý) po dobu celé životnosti řady
- Primární klíč musí být povinný (t.j. definovaný v každé řadě).

Lojza má problémy s druhým a třetím pravidlem.

Vezmeme například [Klient]. Výbraný primární klíč obsahuje kombinaci Příjmení a jméno klienta titulu. Tento výběr není vhodný z několika důvodů, mnoho klientů neužívá svůj titul, nebo jej nevíme, nebo je prostě nemají, takže tyto sloupce všechny nemohou být povinné. Lidé mění svá příjmení, některé ženy i docela často, takže nelze zaručit neměnnost klíče. Potřebujete tedy vytvořit umělý klíč tabulky [Klient]. Nazvete ho identifikace klienta ID klienta.

Tentýž problém máme s primárním klíčem tabulky [Inzerát]. Kolikrát mění noviny svůj název? Rozhodně se to stává.

Další problém je, že nemáme tabulku na uložení [Vydavatelství] a potřebujeme ji, abychom vytvořili klíč v tabulce [Inzerát], nejlépe umělý.

Zde je tabulka:

Vydavatelství

Název sloupce	Typ dat	Klíče
ID vydavatelství	Longint	PK
Název vydavatelství	Řetězec (30)	
Telefon inz.kanceláře	Řetezec (10)	

Po úpravách dostaneme tyto funkční závislosti :

ID klienta -> Příjmení klinta
ID klienta -> Jméno klinta
ID klienta -> Titul
ID klienta -> Adresa
ID klienta -> Město
ID klienta -> Okres
ID klienta -> PSČ
ID klienta -> Telefon práce
ID klienta -> Telefon domů
ID klienta -> Přišel na inzerát
ID klienta -> ID vydavatelství
ID klienta -> Název vydavatelství
ID klienta -> Telefon inz.kanceláře
ID klienta -> Inzerováno kdy
ID klienta -> Od doktora
ID klienta -> Příjmení doktora
ID klienta -> Jméno doktora

ID klienta -> Specializace doktora
ID vydavatelství -> Název vydavatelství
ID vydavatelství -> Telef.inzertní kanceláře
Příjmení doktora+Jméno doktora -> Specializace doktora
Číslo případu -> ID klienta
Číslo případu -> Příjmení klienta
Číslo případu -> Jméno klienta
Číslo případu -> Titul klienta
Číslo případu -> Telefon práce
Číslo případu -> Telefon domů
Číslo případu -> Částka ze případ
Číslo případu -> Příjmení obhájce
Číslo případu -> Jméno obhájce
Číslo případu -> Adresa obhájce
Číslo případu -> Město obhájce
Číslo případu -> Okres obhájce
Číslo případu -> PSČ obhájce
Příjmení obhájce + Jméno obhájce -> Adresa obhájce
Příjmení obhájce + Jméno obhájce -> Město obhájce
Příjmení obhájce + Jméno obhájce -> Okres obhájce
Příjmení obhájce + Jméno obhájce -> PSČ obhájce
Příjmení obhájce + Jméno obhájce -> PZ druhé strany příjmení
Příjmení obhájce + Jméno obhájce -> PZ druhé strany jméno
Příjmení obhájce + Jméno obhájce -> PZ druhé strany telefon
Číslo případu -> PZ druhé strany příjmení
Číslo případu -> PZ druhé strany jméno
Číslo případu -> PZ druhé strany telefon
PZ druhé strany příjmení + PZ druhé strany jméno -> PZ druhé strany telefon
Číslo případu -> Přišel na inzerát
Číslo případu -> ID vydavatelství
Číslo případu -> Název vydavatelství
Číslo případu -> Telefon inz.kanceláře
Číslo případu -> Inzerováno kdy
Číslo případu -> Od doktora
Číslo případu -> Příjmení doktora
Číslo případu -> Jméno doktora
Číslo případu -> Specializace doktora

Podíváme-li se na funkční závislosti je zde přibližně 20 porušení 3NF (Je to však typický případ databáze navrhované neodborníkem, účelem tohoto kurzu je naučit vás vyhnout se takovýmto návrhům).

Zde je uveden seznam porušení 3NF:

- Název vydavatelství nepatří do [Klient], je funkčně závislé na ID vydavatelství. (To jsme již poznali, při opravě problémů s primárním klíčem).
- Telefon inz.kanceláře nepatří do [Klient], je funkčně závislé na ID vydavatelství.
- Specializace doktora je funkčně závislá na kombinaci Příjmení doktora a Jméno doktora, nepatří tedy do [Klient]
- Příjmení klienta je funkčně závislé na ID klinta, nepatří tedy do [Případ]. Totéž se Jménem klienta, telefony a titulem v tabulce [Případ].

- Adresa obhájce je funkčně závislá na kombinaci Příjmení a Jméno obhájce, nepatří tedy do [Případ]. Totéž s Městem, Okres a PSČ obhájce v tabulce [Případ].
- Telefon PZ druhé strany je funkčně závislé na kombinaci Příjmení PZ druhé strany a Jméno PZ druhé strany, nepatří tedy do [Případ].
- Přišel na inzerát je funkčně závislé na ID klienta, nepatří tedy do [Případ]. Totéž s ID vydavatelství a datumem inzerátu.
- Název vydavatelství je funkčně závislé na ID vydavatelství, nepatří tedy do [Případ].

Po odstraněnní všech porušení 3NF založením nových tabulek a přesunutím sloupců, přidáním umělých klíčů do nových tabulek dostáváme správnou normalizovanou formu tabulek:

Klient

Název sloupce	Typ dat	Klíče
ID klienta	Longint	PK
Příjmení	Řetězec (45)	
Jméno	Řetězec (45)	
Titul	Řetězec (45)	
Adresa	Text	
Město	Řetězec (45)	
Okres	Řetězec (2)	
PSČ	Řetězec (9)	
Telefon práce	Řetězec (10)	
Telefon domů	Řetězec (10)	
Přišel na inzerát	Logické	
ID vydavatelství	Řetězec (30)	CK1.1
Datum inzerátu	Datum	CK1.2
Od doktora	Logické	
ID doktora	Longint	CK2

Případ

Název sloupce	Typ dat	Klíče
Případ číslo	Longint	PK
ID klienta	Longint	CK1
ID obhájce	Řetězec (45)	CK2
ID PZ druhé strany	Řetězec (80)	CK3
Částka případu	Real	

Vydavatelství

Název sloupce	Typ dat	Klíče
ID vydavatelství	Longint	PK
Název vydavatelství	String (30)	
Tel. inzertní kanceláře	String (10)	

Inzerát

Název sloupce	Typ dat	Klíče
ID vydavatelství	Longint	PK1 CK1
Datum inzerátu	Date	PK2

Doktor

Název sloupce	Typ dat	Klíče
ID doktora	Longint	PK
Příjmení	Řetězec (45)	
Jméno	Řetězec (45)	
Specializace	Řetězec (45)	

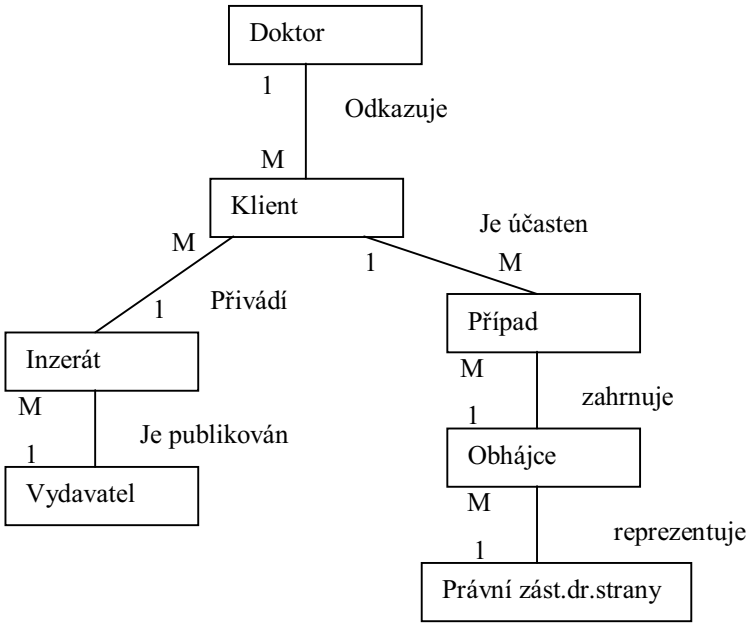
Obhájci

Název sloupce	Typ dat	Klíče
ID obhájce	Longint	PK
Příjmení obhájce	Řetězec (45)	
Jméno obhájce	Řetězec (45)	
ID PZ druhé strany	Longint	CK1

Právní zástupci druhé strany

Název sloupce	Typ dat	Klíče
ID PZ	Longint	PK
Příjmení PZ	Řetězec (45)	
Jméno PZ	Řetězec (45)	
Telefon PZ	Řetězec (10)	

Zde je VRD:



Tato struktura nebere do úvahy fakt, že může být více obhájců v případu, a že obhájce může reprezentovat více právních zástupců oba tyto aspekty byly Lojzou ignorovány. Jestliže však může tato situace nastat, je nutná datší normalizace.

Vzhledem k osobním problémům se zákazníkem, které máte, přesvědčit ho o tom, že se mýlí a vy máte pravdu může být trochu problém. Nejlepší přístup je rozhovor a na téma jak si Lojza sám vylepší návrh. Ukazujte na defekty jeden po druhém a zdůrazněte lepší způsob ukládání a šetření místem, to bude apelovat na jeho spořivost. Po desatém návrhu již Vám sám nejpravděpodobněji navrhne ať přinesete vlastní návrh. Jinak musíte pomalu projít všechny defekty jeden po druhém.

Je-li to pro Vás moc problémů najednou nejste dostatečně hladový návrhář a lépe je najít si dalšího zákazníka.

Příklad 7.2

Je samozřejmě více než jedno správné řešení. Jedno je zde popsáno.

Zde jsou pravděpodobné subjekty:

[Vydavatel]

[Reportáž]

[Reportér]

[Spolupráce]

[Předmět]

Zde jsou vlastnosti:

ID vydavatele

Název vydavatele
ID reportáže
Název reportáže
ID reportéra
Příjmení reportéra
Jméno reportéra
Datum zahájení spolupráce
Datum ukončení spolupráce
Klíčové slovo předmětu

Zde jsou funkční závislosti:

ID vydavatele -> Název vydavatele
ID reportáže -> Název reportáže
ID reportéra -> Příjmení rportéra
ID reportéra -> Jméno reportéra
ID reportéra + ID vydavatele +Datum zahájení spolupráce-> Datum ukončení spolupráce

Kromě toho můžeme rozeznat vztah M:M mezi [Reportáž] and [Předmět], kterou musíme zavést přes další subjekt [Klíčování reportáže].

Zde je věcný seznam vlastností :

Vydavatel

Název sloupce	Typ dat	Klíče
ID vydavatele	Longint	PK
Název vydavatele	String (30)	

Reportér

Název sloupce	Typ dat	Klíče
ID reportéra	Longint	PK
Příjmení reportéra	String (80)	
Jméno reportéra		

Reportáž

Název sloupce	Typ dat	Klíče
ID reportáže	Longint	PK
ID vydavatele	Longint	CK1
ID reportéra	Longint	CK2
Název reportáže	String (80)	

Předmět

Název sloupce	Typ dat	Klíče
Klíčové slovo předmětu	String (80)	PK

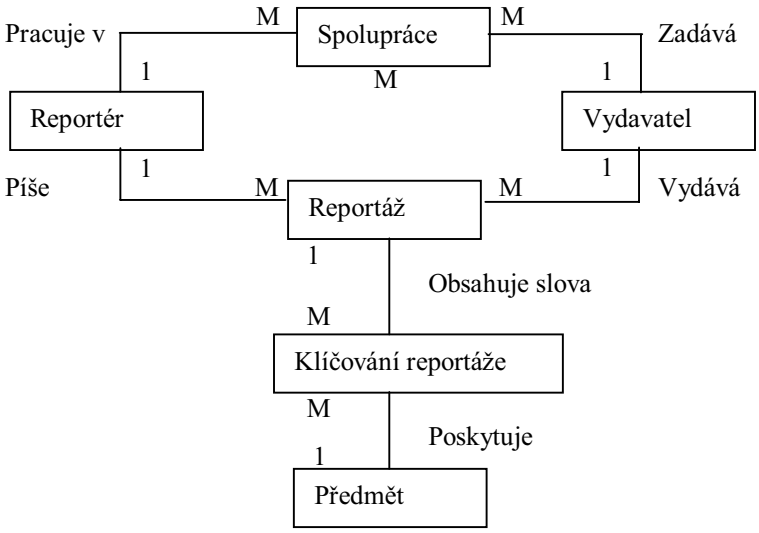
Klíčování reportáže

Název sloupce	Typ dat	Klíče
Klíčové slovo předmětu	String (80)	PK1 CK1
ID reportáže	Longint	PK2 CK2

Spolupráce

Název sloupce	Typ dat	Klíče
ID vydavatele	Longint	PK1 CK1
ID reportéra	Longint	PK2 CK2
Datum zahájení	Date	PK3
Datum ukončení		

Zde je VRD:



Kapitola 8

Příklad 8.1

Existuje více než jedno správné řešení. Následující popisuje jedno správné řešení.

Vyřešení tohoto problému potřebuje jeden dodatečný subjekt, kerý nazveme [Zařazení].

Zde jsou nové funkční závislosti:

Přezdívka piráta + Název pirátské lodi + Počáteční datum -> Datum ukončení

Přezdívka piráta + Název pirátské lodi + Počáteční datum -> Název zařazení

Poslední funkční závislost nahrazuje z předchozího příkladu funkční závislost:

Přezdívka piráta -> Kapitán pirátské lodi

Protože nyní sledujeme celou historii a pirát může být do téže funkce zařazen více než jednou máme vztah 1:M mezi lodí a zařazením. Povšimněme si, že kapitán zde již není výslovně uveden jako typ najdeme ho jako jedno zařazení.

Zde je nový subjekt:

Zařazení

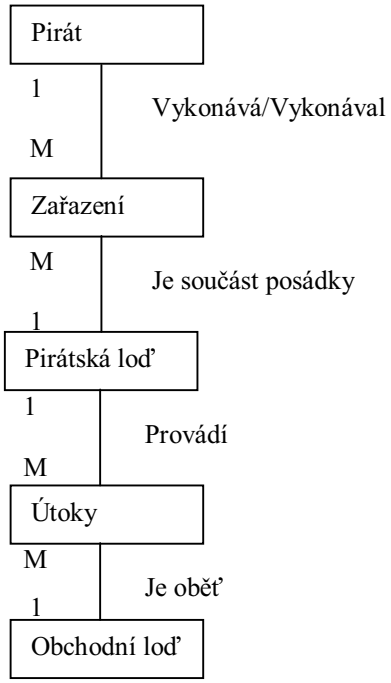
Název sloupce	Typ dat	Klíče
Přezdívka piráta	Řetězec (80)	PK1 CK1
Název pirátské lodi	Řetězec (80)	PK2 CK2
Datum počátku	Datum	PK3
Datum ukončení	Datum	
Zařazení	Řetězec (80)	

Zde je změněná tabulka [Pirátská loď]:

Piratské lodi

Název sloupce	Typ dat	Klíče
Název pirátské lodi	Řetězec (80)	PK
Počet stěžňů PL	Integer	
Počet kanónů PL	Integer	
Datum spuštění PL	Datum	

Zde je nový VRD:



Toto je velice běžná modifikace databáze. Zákazník objeví, že potřebuje ukládat historii dat a ne pouze současný stav. Databázi musíte navrhnout již s tímto uvážením. Jestliže vaše databáze bude normalizovaná, lze takovéto změny provést snadněji. Jinak to může být pěkně zapeklité.

Příklad 8.2

Tato databáze potřebuje cyklický přístup. Je to proto, že potomci jsou rovněž osoby. Existuje několik správných řešení. Následující je jedno z nich.

Zde je věcný seznam vlastností:

Osoba

Název sloupce	Typ dat	Klíče
ID osoby	Longint	PK
Příjmení osoby	String (80)	
Jméno osoby	String (80)	
Datum narození	Date	
Datum úmrtí		

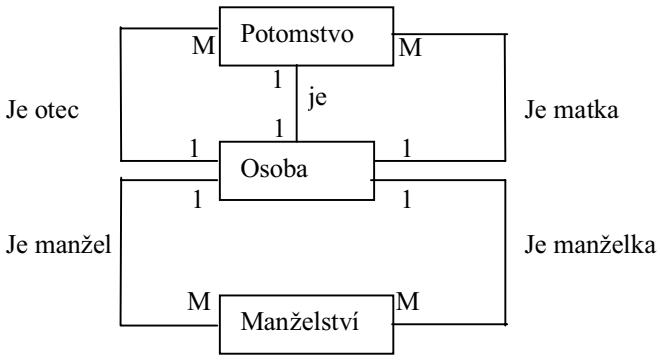
Manželství

Název sloupce	Typ dat	Klíče
ID muže	Longint	PK1
ID ženy	Longint	PK2
Datum sňatku	Date	PK3
Datum rozvodu	Date	

Potomstvo

Název sloupce	Typ dat	Klíče
ID potomka	Longint	PK
ID otce	Longint	CK1
ID matky	Longint	CK2

Zde je VRD:



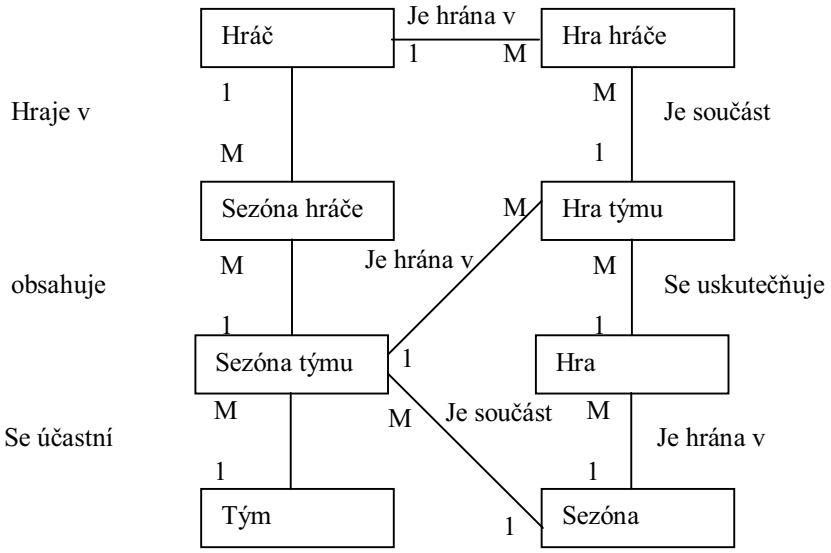
Povšimněme si některých věcí z tohoto návrhu. První je, že narození není vázáno na manželství, takže potomstvo je vázáno na své rodiče a nikoliv jejich sňatky, které ani nemusí být.

Také, že je třícestný vztah mezi [Osoba] a [Potomstvo]. Považujeme datum narození za vlastnost potomstva a nikoliv osoby, ale protože je zde vztah 1:1 je to věc vkusu.

Exercise 8.3

Existuje více než jedno správné řešení, následující je jedno z nich.

Zde je VRD:



Zde je věcný seznam vlastností:

Hráč

Název sloupce	Typ dat	Klíče
ID hráče	Longint	PK
Příjmení	String (80)	
Jméno	String (80)	
Celkem úderů (za život)	Integer	
Počet na pálce	Integer	
Celková účast	Integer	
Celkem chyb	Integer	
Celkem oběhů domů	Integer	
Celkem oběhů	Integer	

Tým

Název sloupce	Typ dat	Klíče
ID týmu	Longint	PK
Název týmu	String (80)	
Sídlo	String (80)	

Sezóna

Název sloupce	Typ dat	Klíče
Počátek sezóny	Date	PK
Konec sezóny	Date	

Sezóna týmu

Název sloupce	Typ dat	Klíče
ID týmu	Longint	PK1 FK1
Počátek sezóny týmu	Date	PK2 FK2
Celkem oběhů	Integer	
Celkem her	Integer	
Celkem vítězství	Integer	
Celkem proher	Integer	
Celkem remíz	Integer	

Sezóna hráče

Název sloupce	Typ dat	Klíče
ID hráče	Longint	PK1 FK1

ID týmu	Longint	PK2 FK2.1
Počátek sezóny hráče	Date	PK3 FK2.2
Celkem úderů za sezónu	Integer	
Počet na pálce	Integer	
Účast za sezónu	Integer	
Celkem chyb za sezónu	Integer	
Celkem oběhů za sezónu	Integer	
C. oběhů domů za sezónu	Integer	

Hra

Název sloupce	Typ dat	Klíče
ID hry	Longint	PK
Datum hry	Date	
Čas hry	Time	
Místo	String (80)	

Hra týmu

Název sloupce	Typ dat	Klíče
ID týmu	Longint	PK1 CK1
ID hry	Longint	PK2 CK2
Start	Logické	
Oběhů	Integer	

Hra hráče

Název sloupce	Typ dat	Klíče
ID hráče	Longint	PK1 CK1
Ball Club ID	Longint	PK2 CK2.1
ID hry	Longint	PK3 CK2.2
Počet na pálce	Integer	
Úderů	Integer	
Chyb	Integer	
Oběhů	Integer	
Oběhů domů	Integer	

Podívejme se jak jsme splnili požadavky zadání:

Statistika hráče

- Průměr na pálce (za sezónu) tabulka [Sezóna hráče].

  Údery hráče za sezónu děleno Počet na pálce za sezónu.
  Údery hráče za sezónu jsou potvrzovány (postupně načítány) z [Hra hráče]Úderů.
  Počet na pálce za sezónu hráče je potvrzováno z [Hra hráče]Počet na pálce.

- Průměr na pálce (za život) tabulka [Hráč].

  Celkem úderů děleno Počet na pálce z tabulky [Hráč].
  Celkem úderů je potvrzováno z [Hra hráče]Úderů.
  Počet na pálce je potvrzováno z [Hra hráče]Počet na pálce.

- Účast (za sezónu)

  [Sezóna hráče]Účast za sezónu je potvrzována z [Hra hráče] (zvýšením o 1 při uložení nového záznamu - jednoduše počet řad v [Hra hráče] pro danou sezónu).

- Účast (za život)

  [Hráč]Celková účast je potvrzována z [Hra hráče] (zvýšením o 1 při uložení nového záznamu - jednoduše počet všech řad v [Hra hráče])

- Průměr chyb na hru (za sezónu)

  Počet chyb za sezónu děleno Účast za sezónu
  Počet chyb za sezónu je potvrzován z [Hra hráče]Chyb.

- Průměr chyb na hru (za život)

  [Hráč]Celkem chyb děleno [Hráč]Celková účast.
  [Hráč]Celková účast je potvrzováno z [Hra hráče]Chyb.

- Oběhů (za sezónu)

  [Sezóna hráče]Celkem oběhů za sezónu je potvrzováno z [Hra hráče]Oběhů.

- Oběhů (za život)

  [Hráč]Celkem oběhů je potvrzováno z [Hra hráče]Oběhů.

- Oběhů domů (za sezónu)

  [Sezóna hráče]C. oběhů domů za sezónu je potvrzováno z [Hra hráče]Oběhů domů.

- Oběhů domů (za život)

  [Hráč]Celkem oběhů domů je potvrzováno z [Hra hráče]Oběhů domů.

Statistika týmu

- Celkem oběhů (za sezónu)

  [Hra týmu]Oběhů je potvrzováno z [Hra hráče]Oběhů.
  [Sezóna týmu]Celkem oběhů je potvrzováno z [Hra hráče]Oběhů.

- Průměr oběhů na hru (za sezónu)

  [Sezóna týmu]Celkem oběhů děleno [Sezóna týmu]Celkem her.
  děleno [Sezóna týmu]Celkem her je potvrzováno z [Hra týmu] (jednoduše počet [Hra týmu] pro danou sezónu).

- Celkem vítězství (za sezónu)

  Z dvou řad [Hra týmu] vztažených k libolné hře [Hra], ta s největším počtem oběhů [Hra týmu]Oběhů je vítěz. Pro tuto řadu, zvětšit hodnotu v sloupci Celkem vítězství o jedna.

- Celkem proher (za sezónu)

  Z dvou řad [Hra týmu] vztažených k libolné hře [Hra], ta s menším počtem oběhů [Hra týmu]Oběhů je prohrávající. Pro tuto řadu, zvětšit hodnotu v sloupci Celkem proher o jedna.

- Celkem remíz (za sezónu)

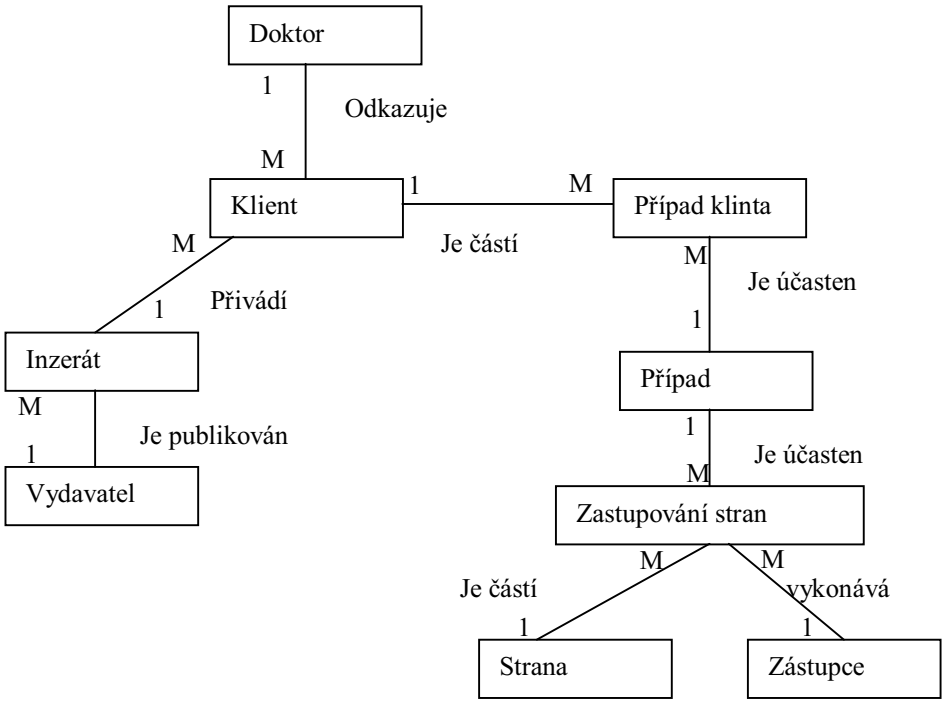
  Jestliže z dvou řad [Hra týmu] vztažených k libolné hře [Hra], je hodnota oběhů [Hra týmu]Oběhů stejná. Pro obě řady zvětšit hodnotu sloupce Celkem remíz o jedna.

Kapitola 9

Příklad 9.1

Je více než jedno správné řešení tohoto problému. Následující popisuje jedno z nich.

Věcný relační diagram :



Slovník dat:

Subjekty:

[Klient]

Je individuum, které je nyní nebo bylo v minulosti zastupováno firmou v případě.

[Případ]

Jedna záležitost, která byla, nebo je snaha aby byla řešena soudní cestou. Firma reprezentuje navrhovatele a protistrana je žalovaná strana.

[Případ klienta]

Účast klienta na konkrétním případě. Je to vazební tabulka pro vytvoření vztahu M:M mezi [Případ] a [Klient].

[Inzerát]

Jeden výskyt inzerce, jednoho druhu, běžící u jednoho vydavatele. Inzerce může v jednom dni běžet vícekrát, např v televizi, a je považována firmou za jeden sledovaný inzerát v tomto dni.

[Doktor]

Lékař, praktik, kerý léčí pacienta a doporučuje mu firmu k zastupování při vymáhání náhrad škody.

[Strana]

Fyzické a právnické osoby, které jsou, budou, nebo byli účastníky soudního řízení, buď jako navrhovatelé nebo jako žalovaní, ve přích, kde alespoň jeden navrhovatel je klient firmy. Jsou to osoby, které nejsou klienty firmy.

[Zástupce]

Právní zástupce zastupující stranu.

[Zastupování stran]

Účast strany v případu. Je to vazební tabulka pro vytvoření vztahu M:M mezi [Strana], [Zástupce] a [Případ].

Vlastnosti:

ID klienta

Jednoznačná identifikace [Klient]. Ukládáno jakoLongint.

Příjmení klienta

Příjmení klienta, ukládáno jako textový řetězec délky 45 – Řetězec(45).

Jméno klienta

Jméno klienta, ukládáno jako textový řetězec délky 45 – Řetězec(45).

Titul

Titul klienta, ukládáno jako Řetězec(45).

Adresa klienta

Uvelení ulice a čísla domu pro trvalou adresu klienta, ukládáno jako text (nestrukturovaný).

Město klienta

Město z trvalé adresy klienta, ukládáno jako Řetězec(45).

Okres klienta

Okres klienta z trvalé adresy klienta, ukládáno jako Řetězec(45).

PSČ klienta

PSČ z trvalé adresy klinta, ukládáno jako Řetězec(5).

Telefon práce

Telefonní číslo, na kterém může být klient zastihnut během normálních pracovních hodin. Ukládáno jako Řetězec(10).

Telefon domů

Telefonní číslo, na kterém může být klient zastihnut mimo normální pracovní hodiny. Ukládáno jako Řetězec(10)

Přišel na inzerát

Pravda, jestliže klientův příchod do firmy je výsledkem inzerce.Jinak Nepravda. Ukládáno jako Logické.

Od doktora

Pravda, jestliže klientův příchod do firmy je výsledkem doporučení firmy doktorem.Jinak Nepravda. Ukládáno jako Logické.

Případ číslo

Jednoznačná identifikace [Případ]. Ukládáno jakoLongint.

Částka za případ

Částka vyjádřená v korunách, kterou firma odhaduje, že získá za zastupování klienta, není nezbytně tatáž, která bude nakonec získána. Ukládáno jako Real (reálné číslo).

ID vydavatele

Jednoznačně identifikuje [Vydavatel]. Ukládána jako Longint.

Název vydavatele

Název vydavatele inzerátu, jestliže jsou to noviny nebo časopis, je zde uveden název novin nebo časopisu. Jestliže je to rádio nebo televize, je to jméno stanice. Jestliže je to telefonní společnost (zlaté stránky) je to jméno společnosti a region (Tele... Jižní Čechy). Ukládáno jako Řetězec(30).

Telefon inzertní kanceláře

Telefonní číslo do inzertní kanceláře vydavatele pro pracovní hodiny. Ukládáno jako Řetězec(10).

Datum inzerce

Datum, kde byla inzerce zveřejněna, nebo běžela (radio,televize). Ukládáno jako Datum.

ID doktora

Jednoznačně identifikuje [Doktor]. Ukládáno jako Longint.

Příjmení doktora

Příjmení doktora. Ukládáno jako Řetězec(45).

Jméno doktora

Jméno doktora. Ukládáno jako Řetězec(45).

ID Strany

Jednoznačně identifikuje stranu, jinou než klienta, v [Strana]. Ukládáno jako Longint.

Příjmení strany

Příjmení strany. Ukládáno jako Řetězec(45).

Jméno strany

Jméno strany. Ukládáno jako Řetězec(45).

Titul strany

Titul strany. Ukládáno jako Řetězec(45).

ID zástupce

Jednoznačně identifikuje [Zástupce]. Ukládáno jako Longint.

Příjmení zástupce

Příjmení zástupce. Ukládáno jako Řetězec (45).

Jméno zástupce

Jméno zástupce. Ukládáno jako Řetězec (45).

Titil zástupce

Titul zástupce. Ukládáno jako Řetězec (45)

Obhájce

Pravda, jestliže je to strana žalovaná, jinak nepravda. Ukládáno jako Logické.

Vztahy:

Odkazuje

Událost, kdy doktor doporučil firmu klientovi, a tento přišel. Založeno na ID doktora.

Přivádí

Událost, kdy klient přišel na základě údajů z inzerátu. Založeno na ID vydavatelství a Datumu inzerátu.

Je účasten

Událost, kdy klient nebo jiná strana se účastní případu. Název vztahu je použit znovu, protože jeho charakter je tentýž. V případě [Klient] je založen na ID klienta a v případě [Strana] je založen na ID strany.

Je částí

Událost, kdy klient, nebo strana je částí (stranou) jednoho konkrétního případu a je tím účastna procesu na své straně. Je to pouze vazební vztah pro vyjádření skutečností, že na straně firmy (její klienti) i stran (nezastupované, žalované) je možná účast více osob (klientů firmy, ostatních..). V případě [Client] je vztah založen na ID klienta a v případě [Strana] na ID strany.

Vykonává

Událost, kdy zástupce vykonává zastupování strany v případu. Založeno na ID zástupce

Věcný seznam vlastností:

Klient

Název sloupce	Typ dat	Klíče
ID klienta	Longint	PK
Příjmení	Řetězec (45)	
Jméno	Řetězec (45)	
Titul	Řetězec (45)	
Adresa	Text	
Město	Řetězec (45)	
Okres	Řetězec (2)	
PSČ	Řetězec (9)	
Telefon práce	Řetězec (10)	
Telefon domů	Řetězec (10)	
Přišel na inzerát	Logické	
ID vydavatelství	Řetězec (30)	CK1.1
Datum inzerátu	Datum	CK1.2
Od doktora	Logické	
ID doktora	Longint	CK2

Případ

Název sloupce	Typ dat	Klíče
Případ číslo	Longint	PK
Částka případu	Real	

Případ klienta

Název sloupce	Typ dat	Klíče
ID klienta	Longint	PK1 CK1
Případ číslo	Longint	PK2 CK2

Vydavatelství

Název sloupce	Typ dat	Klíče
ID vydavatelství	Longint	PK
Název vydavatelství	String (30)	
Tel. Inzertní kanceláře	String (10)	

Inzerát

Název sloupce	Typ dat	Klíče
ID vydavatelství	Longint	PK1 CK1
Datum inzerce	Date	PK2

Doktor

Název sloupce	Typ dat	Klíče
ID doktora	Longint	PK
Příjmení	Řetězec (45)	
Jméno	Řetězec (45)	
Specializace	Řetězec (45)	

Strana

Název sloupce	Typ dat	Klíče
ID strany	Longint	PK
Příjmení strany	Řetězec (45)	
Jméno strany	Řetězec (45)	
Titul strany	Řetězec (45)	

Zástupce

Název sloupce	Typ dat	Klíče
ID zástupce	Longint	PK
Příjmení zástupce	Řetězec (45)	
Jméno zástupce	Řetězec (45)	
Titul zástupce	Řetězec (45)	
Telefon zástupce	Řetězec (10)	

Zastupování stran

Název sloupce	Typ dat	Klíče
Případ číslo	Longint	PK1 CK1
ID strany	Longint	PK2 CK2
ID zástupce	Longint	PK3 CK3
Obhájce	Logické	

Seznam zatížení tabulek:

Klient

Současný stav	3 500
Přírůstek	1 050
Mazání	0
Změna stavu	1 050
Úpravy	1 000
Nejvyšší počet úkonů za měsíc	170

Případ

Současný stav	5 000
Přírůstek	1 500
Mazání	1 000
Změna stavu	500
Úpravy	60 000
Nejvyšší počet úkonů za měsíc	6 125

Případ klienta

Současný stav	5 000
Přírůstek	1 500
Mazání	1 000
Změna stavu	500
Úpravy	0
Nejvyšší počet úkonů za měsíc	1 125

Vydavatelství

Současný stav	35
Přírůstek	1
Mazání	0
Změna stavu	1
Úpravy	0
Nejvyšší počet úkonů za měsíc	1

Inzerát

Současný stav	60
Přírůstek	60
Mazání	60
Změna stavu	0
Úpravy	0
Nejvyšší počet úkonů za měsíc	65

Doktor

Současný stav	18
Přírůstek	1
Mazání	0
Změna stavu	1
Úpravy	0
Nejvyšší počet úkonů za měsíc	1

Strana

Současný stav	5 000
Přírůstek	1 500
Mazání	1 000
Změna stavu	500
Úpravy	5 000
Nejvyšší počet úkonů za měsíc	1 541

Zástupce

Současný stav	100
Přírůstek	10
Mazání	0
Změna stavu	10
Úpravy	100
Nejvyšší počet úkonů za měsíc	9

Zastupování strany

Současný stav	5 000
Přírůstek	1 500
Mazání	1 000
Změna stavu	500
Úpravy	5 000
Nejvyšší počet úkonů za měsíc	1 541

Seznam omezení tabulek:

Případ

Název sloupce	Minimum	Maximum	Výchozí hodn	Výběrový seznam
Částka případu	100 000			

Seznam hledání a třídění:

Klient

Sloupce hledání	Frekvence
ID klienta	Denně +
Příjmení + Jméno klienta	Denně +
Přišel na inzerát	Měsíčně
Od doktora	Týdenně

Sloupce třídění	Frekvence
ID klienta	Denně +
Příjmení + Jméno klienta	Denně +
PSČ klienta	Měsíčně
Název vydavat.+Datum inz	Měsíčně
Příjmení+Jméno doktora	Měsíčně

Případ

Sloupce hledání	Frekvence
Číslo případu	Denně +
Částka případu	Denně +

Sloupce třídění	Frekvence
Číslo případu	Denně +
Částka případu	Denně +

Případ klienta

Sloupce hledání	Frekvence
ID klienta	Denně +
Číslo případu	Denně +
ID klinta+Číslo případu	Denně +
Příjmení klienta	Denně +
Jméno klienta	Denně +
Příjmení+Jméno klienta	Denně +

Sloupce třídění	Frekvence
ID klienta	Denně +

Číslo případu	Denně +
ID klinta+Číslo případu	Denně +
Příjmení+Jméno klienta	Týdenně

Vydavatelství

Sloupce hledání	Frekvence
ID vydavatelství	Denně +
Název vydavatelství	Denně +

Sloupce třídění	Frekvence
ID vydavatelství	Denně +
Název vydavatelství	Denně +

Inzerát

Sloupce hledání	Frekvence
ID vydavatelství	Denně +
Datum inzerátu	Denně +

Sloupce třídění	Frekvence
ID vydavatelství	Denně +
Datum inzerátu	Denně +
Název vydav.+Datum inzre	Měsíčně

Doktor

Sloupce hledání	Frekvence
ID doktora	Denně +
Příjmení doktora	Denně +
Jméno doktora	Denně +
Příjmení + Jméno doktora	Denně +

Sloupce třídění	Frekvence
ID doktora	Denně +
Příjmení doktora	Denně +
Jméno doktora	Denně +
Příjmení + Jméno doktora	Denně +

Strana

Sloupce hledání	Frekvence
ID strany	Denně +
Příjmení strany	Denně +
Jméno strany	Denně +
Příjmení+Jméno strany	Denně +

Sloupce třídění	Frekvence
ID strany	Denně +
Příjmení strany	Denně +
Jméno strany	Denně +
Příjmení+Jméno strany	Denně +

Zástupce

Sloupce hledání	Frekvence
ID zástupce	Denně +
Příjmení zástupce	Denně +
Jméno zástupce	Denně +
Příjmení+Jméno zástupce	Denně

Sloupce třídění	Frekvence
ID zástupce	Denně
Příjmení zástupce	Denně +
Jméno zástupce	Denně +
Příjmení+Jméno zástupce	Denně +

Zastupování strany

Sloupce hledání	Frekvence
ID strany	Denně +
Příjmení strany	Denně +
Jméno strany	Denně +
Příjmení+Jméno strany	Denně +
ID případu	Denně +
ID zástupce	Denně +
Příjmení zástupce	Denně +
Jméno zástupce	Denně +
Příjmení+Jméno zástupce	Denně +

Sloupce třídění	Frekvence
ID strany	Denně +

Příjmení strany	Denně +
Jméno strany	Denně +
Příjmení+Jméno strany	Denně +
ID případu	Denně +
ID zástupce	Denně +
Příjmení zástupce	Denně +
Jméno zástupce	Denně +
Příjmení+Jméno zástupce	Denně +

Seznam přístupových práv:

Klient

Přístup	Skupina
Číst	All
Přidávat	Hlavní společník
Upravovat	All
Mazat	Hlavní společník

Případ

Přístup	Skupina
Číst	Všechny
Přidávat	Všechny
Upravovat	Všechny
Mazat	Hlavní společník

Případ klienta

Přístup	Skupina
Číst	Všechny
Přidávat	Všechny
Upravovat	Všechny
Mazat	Hlavní společník

Vydavatelství

Přístup	Skupina
Číst	Všechny
Přidávat	Hlavní společník
Upravovat	Hlavní společník
Mazat	Hlavní společník

Inzeráty

Přístup	Skupina
Číst	Všechny
Přidávat	Všechny
Upravovat	Hlavní společník
Mazat	Hlavní společník

Doktor

Přístup	Skupina
Číst	Všechny
Přidávat	Hlavní společník
Upravovat	Hlavní společník
Mazat	Hlavní společník

Strana

Přístup	Skupina
Číst	Všechny

Přidávat	Všechny
Upravovat	Všechny
Mazat	Hlavní společník

Zastupování strany

Přístup	Skupina
Číst	Všechny
Přidávat	Všechny
Upravovat	Hlavní společník
Mazat	Hlavní společník

Kapitola 10

Příklad 10.1

Produkty

Název sloupce	Typ dat	Klíče
Číslo produktu	Řetězec (10)	PK
Titul	Řetězec (30)	
Cena	Real	
Rok	Integer	
Kategorie	Řetězec (15)	

Zákazník

Název sloupce	Typ dat	Klíče
Název společnosti	Řetězec (25)	PK
Jméno kontaktní osoby	Řetězec (10)	
Příjmení kontaktní osoby	Řetězec (20)	
Adresa	Řetězec (25)	
Město	Řetězec (20)	
Okres	Řetězec (2)	
PSČ	Řetězec (10)	
Telefon	Řetězec (10)	
Důležitý	Logické	

Faktury

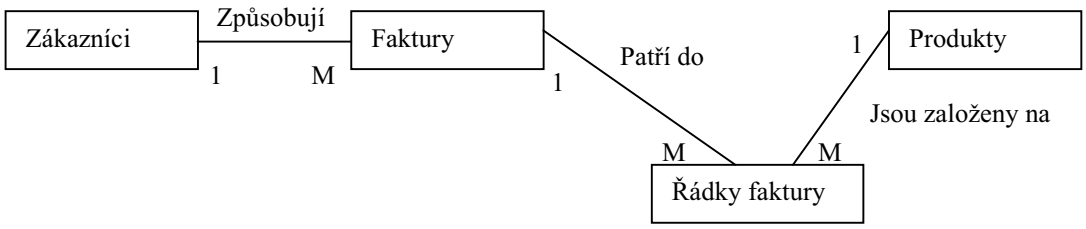
Název sloupce	Typ dat	Klíče
Faktura číslo	Longint	PK
Název společnosti	Řetězec (25)	CK1
Datum faktury	Date	

Částka celkem	Real
Placena	Logické
Způsob platby	Řetězec (15)

Řádky faktury

Název sloupce	Typ dat	Klíče
ŘF číslo	Longint	PK1 CK1
ŘF ID produktu	Řetězec (10)	PK2 CK2
Množství	Integer	
Aktual jednotková cena	Real	
Cena celkem položky	Real	

Zde je VRD:



Příklad 10.2

Zboží

Název sloupce	Typ dat	Klíče
Číslo zboží	Řetězec (10)	PK
Název	Řetězec (30)	
Prodejní cena	Real	
Rok	Integer	
Množství	Longint	
Sazba daně z PH	Real	

Zákazníci

Název sloupce	Typ dat	Klíče
Název společnosti	Řetězec (25)	PK
Příjmení	Řetězec (10)	
Jméno	Řetězec (20)	
Adresa	Text	
Město	Řetězec (20)	
Okres	Řetězec (2)	
PSČ	Řetězec (10)	
Telefon	Řetězec (10)	
Důležitý	Logické	
Zahraniční	Logické	

Faktury

Název sloupce	Typ dat	Klíče
Číslo faktury	Longint	PK
Faktura název společnosti	Řetězec (40)	CK1
Datum faktury	Date	

Částka bez DPH	Real
DPH	Real
Celkem Faktura	Real
Datum splatnosti	Date
Placena	Logické
Způsob platby	Řetězec (15)
Poznámka	Text

Položky faktury

Název sloupce	Typ dat	Klíče
PF číslo faktury	Longint	PK1 CK1
PF id zboží	Řetězec (10)	PK2 CK2
Množství	Integer	
PF celkem bez DPH	Real	
DPH položky	Real	
PF celkem s DPH	Real	

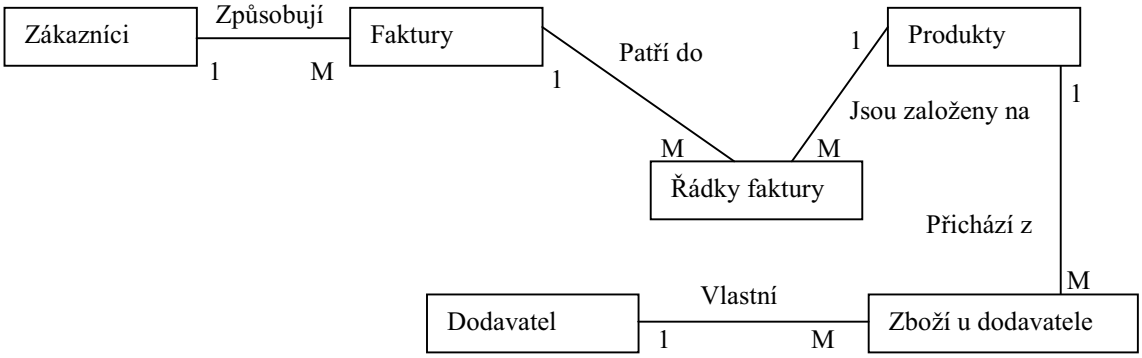
Dodavatelé

Název sloupce	Typ dat	Klíče
ID Dodavatele	Longint	PK
Název společnosti Dod	Řetězec (25)	
Jméno kontaktní osoby	Řetězec (10)	
Příjmení kontaktní osoby	Řetězec (20)	
Adresa	Text	
Město	Řetězec (25)	
Okres	Řetězec (2)	
PSČ	Řetězec (10)	
Telefon	Řetězec (10)	
Poznámka	Text	

Zboží u dodavatele

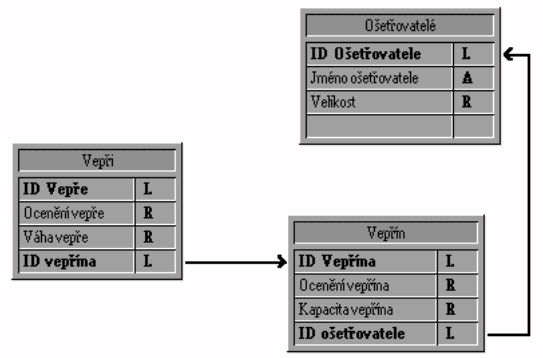
Název sloupce	Typ dat	Klíče
ID dodavatele	Longint	PK1 CK1
Zboží ID	Řetězec (10)	PK2 CK2
Cena dodavatele	Integer	
Kód dodavatele	Řetězec (20)	
Poznámka kvality	Text	

Zde je VRD :



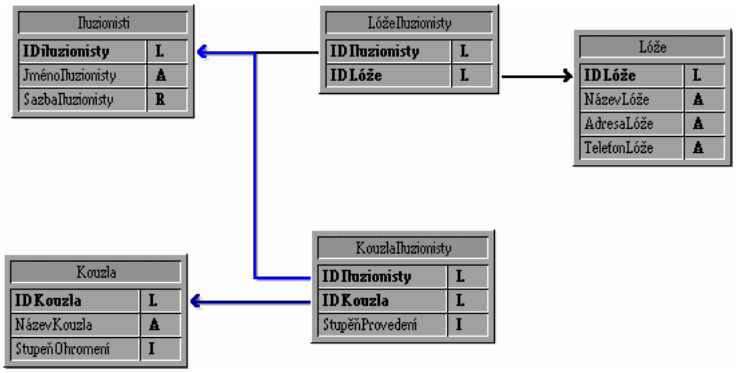
Kapitola 11

Příklad 11.1



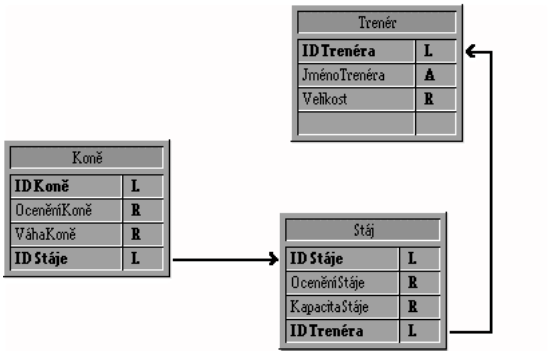
Stukturu příkladu ve 4D najdete ve složce školící materiály

Příklad 11.2

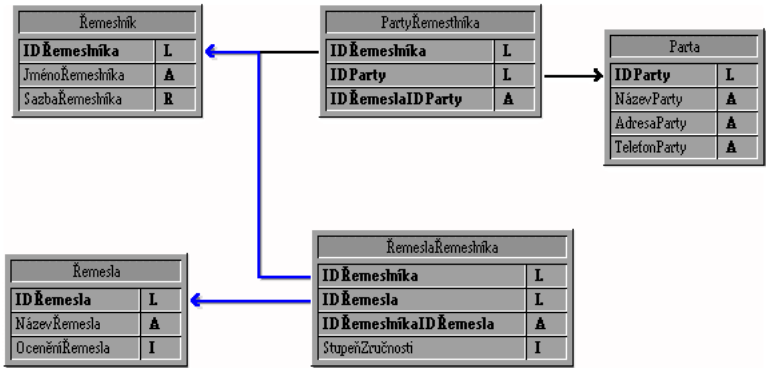


Stukturu příkladu ve 4D najdete ve složce školící materiály

Příklad 11.3



Příklad 11.4



Kapitola 12

Příklad 12.1

Existuje mnoho řešení. Je to vaše závěrečná práce, proto zde není žádné popsáno. Po skončení kurzu a diskuzi, jedno ze správných řešení dostanete na disketě.