Česká zemědělská univerzita v Praze

Technická fakulta

Katedra technologických zařízení staveb

Bakalářská práce

Logické databázové modely

Jana Sladká

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jana Sladká

Informační a řídicí technika v agropotravinářském komplexu

Název práce

Logické databázové modely

Název anglicky

Logic structure of database models

Cíle práce

Cílem práce je shrnout a posoudit základní databázové modely (RDBMS, ORDBMS, ODBMS) z pohledu datových typů. Demonstrovat na příkladech jednoduchých kódů.

Metodika

1. Literární rešerše
2. Metodika
3. Logické databázové modely
4. Datové typy (Bachmanuv diagram, Hiearchický model, Relační model, Objektový model, XML)
5. Posouzení vhodnosti pro typické aplikace
6. Zhodnocení, cenové relace

Doporučený rozsah práce
30 – 40 stran včetně obrázků a tabulek

Klíčová slova
databázový model, XML, UML

Doporučené zdroje informací
AUER, D J. – KROENKE, D. *Databáze.* Brno: Computer Press, 2015. ISBN 978-80-251-4352-0.
Conolly,T.: Databáze, 2009, Computer Press, ISBN: 978-80-2512-328-7
LACKO, Ľ. *Databáze: datové sklady, OLAP a dolování dat s příklady v MicrosoSQL Serveru a Oracle.*.
MERUNKA, V.: Objektové modelování, 2008, Alfa Publishing, ISBN: 978-80-87197-04-2

Předběžný termín obhajoby
2018/19 LS – TF

Vedoucí práce
Ing. Zdeněk Votruba, Ph.D.

Garantující pracoviště
Katedra technologických zařízení staveb

Elektronicky schváleno dne 29. 1. 2018

doc. Ing. Jan Malaťák, Ph.D.
Vedoucí katedry

Elektronicky schváleno dne 30. 1. 2018

prof. Ing. Vladimír Jurča, CSc.
Děkan

V Praze dne 25. 03. 2019

Čestné prohlášení

„Prohlašuji, že jsem bakalářskou práci na téma: Logické databázové modely vypracovala samostatně a použila jen pramenů, které cituji a uvádím v seznamu použitých zdrojů. Jsem si vědoma, že odevzdáním bakalářské práce souhlasím s jejím zveřejněním dle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů, ve znění pozdějších předpisů, a to i bez ohledu na výsledek její obhajoby. Jsem si vědom/a, že moje bakalářská práce bude uložena v elektronické podobě v univerzitní databázi a bude veřejně přístupná k nahlédnutí. Jsem si vědoma že, na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, ve znění pozdějších předpisů, především ustanovení § 35 odst. 3 tohoto zákona, tj. o užití tohoto díla.“

V Praze dne _____ _____

Sladká Jana

Poděkování

Ráda bych touto cestou poděkovala svému vedoucímu práce panu Ing. Zdeňkovi Votrubovi, Ph.D za výběr tématu a možnost pracovat pod jeho vedením. Díky tomu jsem získala spoustu znalostí, které mohu uplatnit v praxi. Chtěla bych také poděkovat celé své rodině, která mi byla nejen při psané této práce, ale během celého studia mou největší oporou.

Logické databázové modely

Abstrakt

Práce se zabývá základními logickými datovými modely a jejich problematikou. Na začátku práce jsou databáze představeny z historického hlediska. Následuje teoretický rozbor databází, které jsou pro dnešní dobu nezbytné. Na databáze navazují logické databázové modely, kde jsou probrány jejich struktury a základní vlastnosti. Cílem je zde poukázat na jejich hlavní používání, které se projevuje v závěru práce. V praktické části je vysvětlen strukturově dotazovací jazyk SQL a následně pomocí něj vytvořená relační databáze. V závěru práce je vhodnost použití jednotlivých databázových modelů.

Klíčová slova: databázové modely, XML, UML, SQL

Logic structure of database models

Abstract

Bachelor thesis deals with basic logical data models and their problems. At the beginning of the work, the databases are discussed from a historical perspective. The following is a theoretical analysis of the databases necessary for today. The databases are followed by logical database models where their structures and basic properties are discussed. The aim is to point out their main use, which is reflected in the conclusion of the work. The practical part explains the structure query language SQL and then the relational database created by it. At the end of the work is the suitability of using individual database mode

Keywords: database models, XML, UML, SQL

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Již před naším letopočtem lidé používali databáze. Nepotřebovali k tomu počítače jako je tomu dnes, ale stačily jim pouhé hliněné tabulky. Z té doby už je patrné, že se lidé snaží data třídit, ale neumí s nimi dále nakládat jako je tomu nyní. Již v 19. století přichází první automaticky stroj zřízený panem Hollerithem. Databáze pak procházely několika letým vývojem. Až v roce 1969 E. F. Codd přišel s článkem o databázi založené na matematickém aparátu a predikátové logice. Jednalo se o první databáze, které pracovaly s pojmem schéma relace. Zavedl definice u pojmů, o kterých se pouze jen mluvilo. Relační model popisuje tabulky, jejich strukturu a s daty manipuluje pomocí matematických pojmů, především z matematické logiky. V relační databázi mohou vznikat vztahy mezi tabulkami. Dalším typem databází je Objektově relační databáze, která se snaží sjednotit základny rysy z relační i objektové databáze. Výhoda objektově relačních databází je, že již máme podporu pro zpracování komplexních dat. Posledním typem je objektová databáze, která je velice rozdílná oproti relační databázi. Ke své činnosti využívá objektový model dat. Vychází ze známých principů, objektově relačního programování. Základem je uvažovat o programu jako kolekci objektů na sobě nezávislých, které jsou sdružovány do tříd.

Po rozšíření objektového modelování se začal objevovat standard jazyka UML. Jedná se spíše o grafické řešení, které usnadňuje především komunikaci. Nejnovější datový model XML umožňuje zapsat data společně s jejich významem. Jedná se o průlomové řešení, protože nezáleží na tom, v jakém jazyce se to píše. XML umí pracovat se všemi jazyky a zároveň je velice jednoduchý. Již nyní spousta programů podporuje model XML.

V této elektronické době jsou databáze nezbytné pro všechny. Každý ať už si to uvědomuje nebo ne, databáze se využívají v hojném množství. Lidé si odjakživa chtěli práci jen usnadňovat a vývoj v databázích je tomu jasným příkladem. Nikdo už si nedovede představit, že by se používaly kartotéky a ručně všechno zapisovalo. Každý uživatel očekává snadný, rychlý a jednoduchý program s minimální námahou a nejlépe s okamžitým výsledkem. Proto databáze byly, jsou a budou potřeba v životě člověka.

2 Cíl práce a metodika

Cíl práce

Cílem práce je shrnout a posoudit základní logické databázové modely. Mezi dílčí cíle práce patří popsání databázových modelů od těch nejstarších po současně používané. Každý databázový model je doplněný o obrázek pro zjednodušení a lepší porozumění. V praktické části jsou uvedeny a porovnány databázové modely na jednotlivých kódech, a v závěru porovnání jednotlivých typů z hlediska vhodnosti. Práce dále poukazuje na výhody a nevýhody jak logických databází, tak i jednotlivých logických modelů.

Metodika

Teoretická část je zpracovaná pomocí deskriptivních metod založených na studiu odborné literatury. Dochází zde k určení přesných pojmů, s kterými je dále pracováno dle jejich popisu. Popsány jsou zejména logické databáze a databázové modely. V praktické části dochází ke komparaci jednotlivých typů. Nechybí zde ani praktická ukázka relačního a objektového modelování. V závěru jsou objasněné pojmy, které se často a snadno zaměňují.

3 Úvod do databází

Ke každé databázi je zapotřebí vytvoření modelu dat. Model dat je množina konceptů, kterou je možné použít k popisu struktury databází a jejich operací. Model dat je u většiny databázových řešení nutné nejdříve postupně formulovat, tzn. nejdříve vymezit jeho strukturu obecně.

3.1 Historie

Již za první doloženou zmínku o snaze uchovávat data je knihovna ve městě Ugaritu na území dnešní Sýrie. Zde se nalezlo větší množství hliněných tabulek s diplomatickým textem a literární tvorbou, které pocházejí již z 12. století před naším letopočtem. Tímto se tady ukázala snaha o to data sbírat, nikoliv však je třídit. První snaha o třídění se objektivně potvrdila až u knihovny nacházející se v římském Foru Romanu.

Za předchůdce počítačových databází je považována papírová kartotéka. Ta splňovala myšlenku dnešní doby – třídit data do nových položek a uspořádávat podle různých kritérií. Jejich správa byla velmi náročná, jelikož veškeré operace s informacemi prováděl člověk sám.

Dalším změna ve vývoji databází bylo zpracování dat strojem, kdy v roce 1880 jako první automatický stroj vytvořil Herman Hollerith. Pomocí elektromechanických strojů probíhalo zpracovávání informací, které se uchovávaly pomocí děrného štítku. Ten sloužil jako paměťové médium. Pro zpracování dat se elektromechanické stroje využívaly další půlstoletí. V roce 1911 se spojila jeho firma s několika dalšími vývojáři a došlo ke vzniku nové firmy zvané International Business Machines (IBM), která se dodnes zabývá výpočetní technikou. Během první světové války se používal systém děrných štítků. Bohužel taková metoda se ukázala jako neefektivní pro databázové úlohy. Tím vzniknul požadavek na vyšší jazyk pro zpracování dat.

V USA roku 1935 byla uzákoněna (Social Security Act) povinnost vést informace o zhruba 26 miliónech zaměstnancích. Kvůli tomu firma IBM vytvořila nové zařízení pro zpracováních podobných úloh. Tzv. UNIVAC I, byl vytvořen pro komerční využití první digitální počítač. Byl založený na státem podporovaném projektu Elektronic Discrete Variable Automatic Computer (EDVAC) z Pensylvánské Univerzity. V roce 1959 měl Pentagon více než 200 počítačů, jejichž základem byl hlavně děrný štítek.

V roce 1960 vzniklo uskupení, jehož výsledkem byl produkt Common Business-Oriented Language – COBOL. Stal se nejrozšířenějším jazykem, protože v té době uměl nejlépe hromadně zpracovávat data. Hlavní záměr bylo sestavovat programy s minimálním programovacím úsilím a v minimálním čase. Zápis programů v anglickém jazyce a snadným převodem programů na novější typy počítačů, byl označován jako COBOL-60. Že se tento jazyk bude rozvíjet bylo zřejmé už od počátku. Po provedených změnách a úpravách byl vydán v roce 1961 COBOL-61. V roce 1963 byl uskupením vydán COBOL-61 EXTENDED, který oproti jazyku COBOL-61 obsahoval příkazy pro generování tiskových sestav, pro více aritmetických příkazů, ale především příkazy pro třídění dat. Další navazující verzí byla COBOL-65. Ta byla doplněna o příkazy pro operace s daty v hromadných pamětech. Byl zaveden nový typ indexace příkazy pro vyhledávaní informací v tabulkách. Komunikaci programů a dělení se zbytkem zavedl COBOL-68. Bylo zjednodušeno využívaní knihovny programů, byly zrušeny nadbytečné editační popisy a provedeny další nezbytné úpravy.

Verze doplněna a rozšířena o příkazy pro manipulace s řetězy znaků pro komunikaci s koncovým zařízením byla COBOL-69. Zjednodušení se týkala specifikací. COBOL – 70 byl doplněn nejen o příkaz pro slučování souborů, ale i o příkaz pro obsazení položek, které se shodují s jejich popisem.

Další velká změna se týkala přechodu z magnetických pásek na magnetické disky. Důvod byl takový, že magnetický pásek umožňoval pouze sériový přístup k datům (což znamenalo značné ztížení při vyhledávání dat).

Jako první představil integrovaný datový sklad s jinými vlastnostmi Charles Bachman z General Electric. Bylo to v roce 1961 a už zde byl náznak DB managementu. V šedesátých letech, výzkumníci v čele s Bachmanem založili samostatnou skupinu Database Task Group (DBTG), která byla v rámci seskupení Codasyl. Seskupení pak dále publikovalo základní specifikace pro programovací jazyky (zvláště pak COBOL), které byly určené pro práci s databázemi. Na podobné specifikaci Codasyl posléze vznikla řada produktů od firem např. Honeywell Incorporated, Siemens AG a další.

Pochopitelně Produkt Codasyl měla i firma IBM, která ho uvedla v roce 1968 pod názvem IMS. Pracovala na počítači Systém/360. Většina kompatibilních databází Codasyl používala síťový model dat, zatímco firma IBM používala hierarchický model.

Po zveřejnění článku E.F. Codda v 70. letech se začalo přecházet na nové databázové systémy založené především na relacích a relační algebře. Ty pohlížely na data jako na

tabulky. Veškeré operace s daty jsou uskutečňovány pomocí základních operací z relační teorie a ostatní operace jsou jen kombinace těchto operací.

První verze dotazovacího jazyka SQL se objevuje v roce 1974. V dalších letech se pak pracuje zcela jen s relační databází a jazyka SQL. Ten se postupně stal jediným nástrojem pro práci s takovým typem databázových systémů.

Začátky objektově orientované databáze se objevily v 90. letech. Jsou založené na objektovém programování. Výhodou je možnost uchovávání širokého spektra dat – od znakových, přes obrazová, zvuková data a videa. Dle předpokladů tyto databáze měly nahradit relační systémy. To se ale nestalo, a proto vznikla kompromisní objektově-orientovaná technologie. [1, 4]

3.2 Relační databáze

V roce 1969 doktor E. F. Codd vydal článek o databázi založené na predikátové logice a matematickém aparátu relačních množin. Databáze je založena na pojmu schéma relace. Schéma relace uvádí, jaký je název relace, počet sloupců a jak se nazývají domény, nebo-li přípustné hodnoty daného sloupce. Relační datový model popisuje databázi, její strukturu, vlastnosti a manipulaci s daty pomocí matematických pojmů. Především z matematické logiky a teorii množin. Přinesl přesné definice k pojmům, které dosud byly popisovány jen slovně. Popisuje chyby ve strukturách tabulek databáze a definuje správně navržené databáze. Relační model uspořádává data do tzv. relací (tabulek), což jsou dvourozměrné struktury tvoření záhlavím a tělem. Základ relační databáze tvoří relace. Tabulka je skladba záznamů s pevně nastavenými atributy (sloupci) a v řádcích tabulky jsou pak záznamy. Každý sloupec je definován jednoznačným názvem, typem a rozsahem. Řádek je řezem přes sloupce tabulky a slouží k vlastnímu ukládání dat. Pokud se v různých tabulkách shodují sloupce, pak mezi jednotlivými tabulkami mohou totožné sloupce vytvářet vazby.

Tabulky se plní konkrétními daty. [1, 7]

3.2.1 Vztahy v relační databázi

Rozlišujeme tři základní typy vztahů:

- **1:1**

  Vztah 1:1 je sdružení dvou tabulek, kdy hodnota primárního klíče každého záznamu primární tabulky, odpovídá shodnému poli či polí právě jednoho záznamu související tabulky.

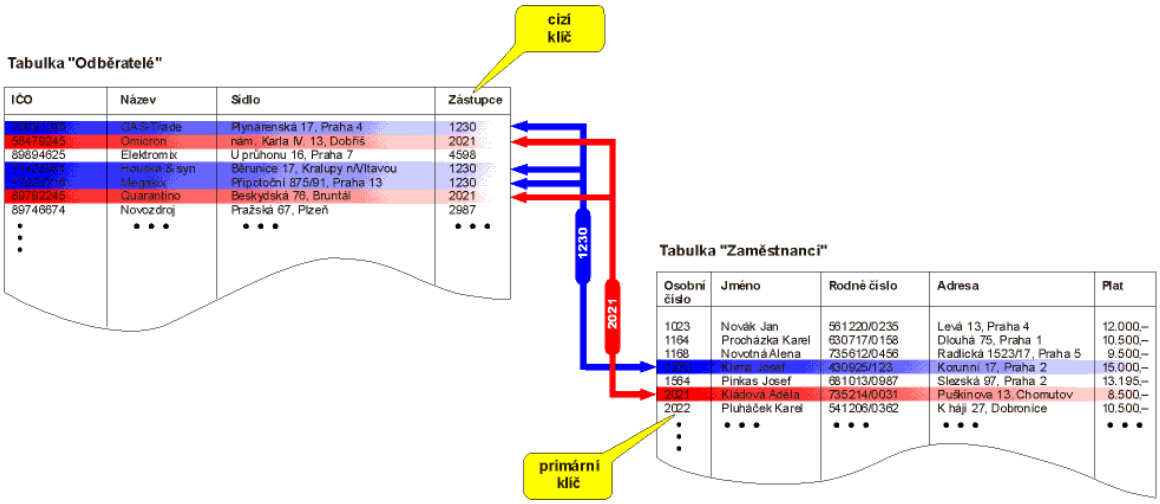
- **1:N**

  Vztah 1:N je sdružení dvou tabulek, kdy hodnota primárního klíče každého záznamu primární tabulky odpovídá hodnotě shodné pole či polí mnoha záznamů související tabulky. Takový vztah je vidět na obr. 1.

- **M:N**

  Vztah M:N je sdružení dvou tabulek, kdy může každý záznam kterékoliv z obou tabulek souviset s mnoha záznamy související tabulky. U této relace nám mohou vzniknout problémy. Jedním z nich může být vznik přebytečných dat, které porušují pravidla normalizace. Pro realizaci si musíme pomoci třetí tabulkou, jelikož pouze dvě nám nestačí. Vztah tak převedeme na tabulky vztahu 1:N, kde původní dvě tabulky plní roli primární k tabulce pomocné.[7]

Obr. 1 Ukázka vztahu 1:N



zdroj: https://www.kosek.cz/clanky/iweb/12.html

3.3 Objektově relační databáze

Zastánci relační databáze, tvrdí, že objektová databáze byla vyvinuta především k podoře programovacích jazyků a jeho interakcí s daty. Objektově relační zastánci však vidí hlavní slabost tradičních relačních systémů v neschopnosti podporovat komplexní data.

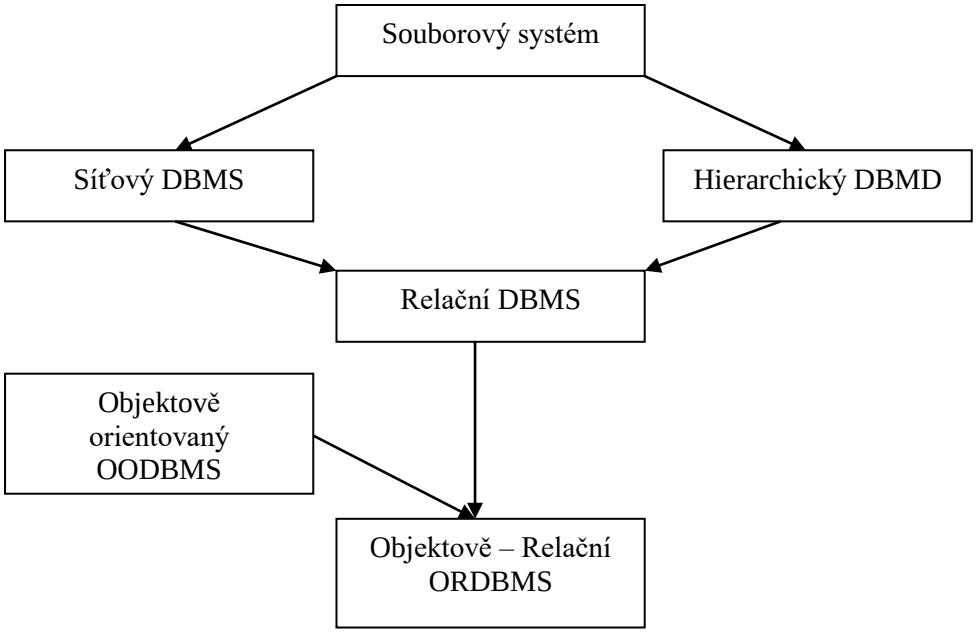
Objektově relační databáze se snaží sjednotit rysy jak relačních, tak objektových databází. Jak objektově relační databáze pracuje je patrné z obr. 2. Objektově relační databázový systém přidává podporu zpracování komplexních dat na již existujících SQL. Je nutné přidat podporu pro:

- rozšíření základních datových typů v rámci SQL;
- složité objekty v kontextu SQL;
- dědění v kontextu SQL;
- tvorbu systémových pravidel.

Objektově relační model využívají datový model tak, že přidávají tzv. objektovost do tabulek. Všechny trvalé informace jsou v tabulkách, ale některé položky mohou mít větší datovou strukturu. Tu nazýváme abstraktní datové typy (ADT).

Obvykle se mezi základní datové typy v tradičních relačních systémech řadí znak, řetězec, celé číslo, číslo s plovoucí desetinnou čárkou, datum a čas. [6, 7]

Obr. 2 Objektově relační databáze



Zdroj: vlastní

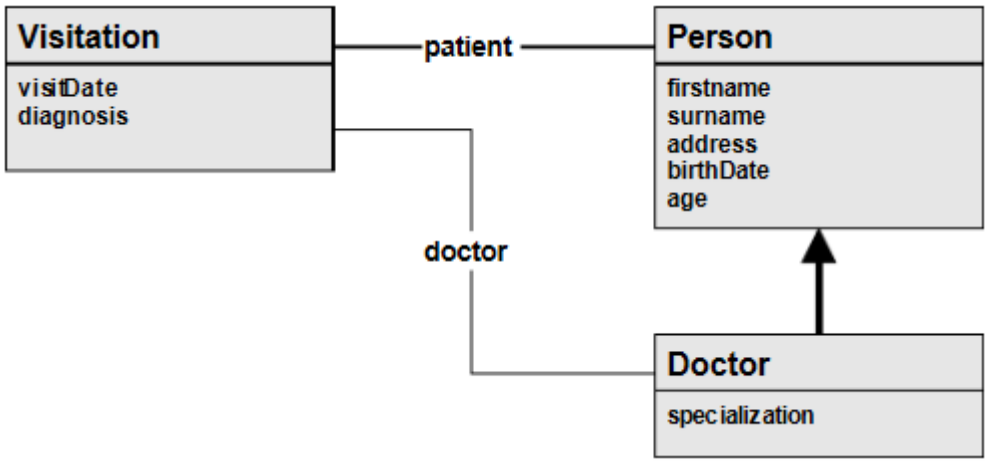
3.4 Objektová databáze

Ke své činnosti využívá objektový model dat. Vychází ze známých principů objektově orientovaného modelování a programování. Principem ODBMS je ukládat objekty do databáze se současným využitím užitečných prvků objektové technologie. Dále je však obohacena o techniky perzistence, prezentace vztahů, dotazování, transakčního přístupu apod. Systémy jsou pomocí svého objektového návrhu velmi pružné, jelikož nejsme limitováni datovými typy, ani dotazovacím jazykem, který je typický pro relační model. Základním a zásadním rysem objektově-orientovaných systémů je schopnost specifikace složitých objektů a tvorba vlastních operací pro manipulaci s objekty. Vzniká tak ekvivalence mezi objekty.

Základním principem objektového přístupu v programovacích jazycích je uvažovat o programu jako kolekci nezávislých objektů, které jsou sdružovány do tříd. Komunikují mezi sebou pomocí zpráv, které si zasílají. Objekty v programovacích jazycích existují pouze v době běhu programu. Naproti tomu objekty tomu v ODBMS mohou být vytvořeny jako perzistentní a mohou být sdíleny více programy.

„Vlastnosti objektové databáze budou prezentovány pomocí následující úlohy: Mějme jednoduchou databázi pro evidenci návštěv v ordinacích lékařů. U každého pacienta bude evidováno jeho jméno, příjmení, adresa, datum narození a věk. U každého doktora ještě navíc jeho specializace. U návštěvy pacienta u lékaře bude evidováno datum návštěvy a diagnóza. Není vyloučeno, aby se jeden lékař stal druhému lékaři pacientem.“ [12] Úlohu znázorňuje následující obr.3. [6, 10, 12]

Obr. 3 Objektová databáze



zdroj: http://docplayer.cz/665523-Objektovy-pristup-v-databazove-technologii.html

4 Datové modely

Model je schéma, které člení data do druhů a skupin. Současně taky vymezuje vztahy mezi daty (tabulkami). Pro správnou organizaci dat v databázi jsou důležité jednotlivé modely, do kterých se snažíme zachytit povahu procesů. Datový model většinou bývá základem či prerekvizitou pro konstrukci fyzického datového modelu.

4.1 Hierarchický model

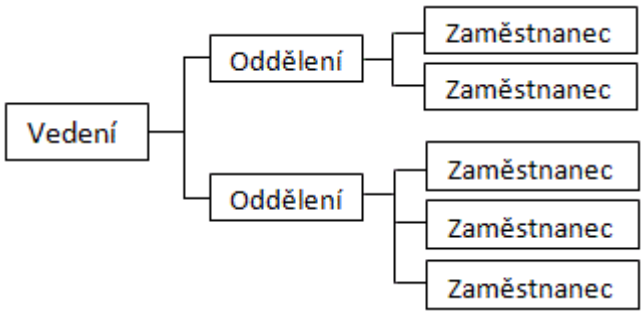
Základem modelu je stromová struktura, která je znázorněna na obr.4. Základním Stavebním prvkem je kořen a na větvích jsou umístěny uzly a listy. Kořenový uzel existuje v každém systému právě jeden. Vlastní datové struktury obsahují větve, avšak uzly mohou být také v rámci jedné úrovně. Nemusí tedy být jen v klasickém hierarchickém vztahu. To nám velice usnadňuje vyhledávání v databázi. Není totiž nutné prohledávat veškerá data umístěna v databázi. Pomocí navigace po větvích a listech nalezne požadovanou informaci. Omezení zůstává v propojení na jednosměrnost vazeb 1:N.

Tento databázový model je vhodný hlavně pro aplikace, které zpracovávají data postavená na hierarchické struktuře. K těm patří například skladové či organizační systémy.

Hlavním omezením hierarchického modelu je nutnost rekonstrukce celé struktury databáze, pokud dojde ke změnám požadavků. Nestačí pouze přidat či ubrat jednu položku. Mezi další nevýhodu hierarchického modelu, patří obtížné znázornění vztahu M:N, které se mohou vyřešit pomocí cyklických vztahů a redundantních přístupů.

V 60. letech byl společností IBM a NAA vyvinut systém IMS, což bylo nejznámější implementací tohoto modelu. [2]

Obr. 4 Hierarchický model

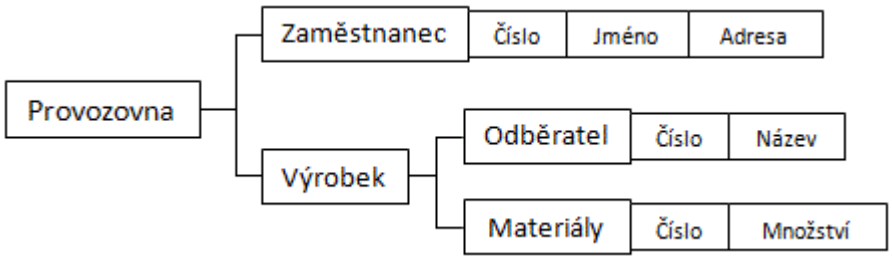


zdroj: http://www.databaze.chytrak.cz/modely.htm

4.2 Síťový model

Model je založen na použití ukazatelů vyjadřujících vztahů mezi jednotlivými databázovými položkami. Vychází z hierarchického modelu a rozšiřuje jej o mnohonásobné vztahy označované jako „C-množiny“ nebo „Sety“. Sety slouží k propojení záznamů, a to i různého typu. Oproti hierarchickému modelu síťový model odstraňuje omezení ve vztazích. V čistě hierarchickém modelu může být pouze jeden uzel rodičovský. Tedy je možné realizovat vztah pouze jeden k mnoha (jeden záznam typu A je spojen s mnoha záznamy typu B). V síťovém modelu však může uzel souviset s více než jedním uzlem, jako je to vidět na obr. 5. Mezi nevýhody modelu patří obtížná změna struktury.[2]

Obr. 5 Síťový model



zdroj: http://www.databaze.chytrak.cz/modely.htm

4.3 Relační model

Relační model přináší mnoho výhod, zejména snadnou definici dat a zpracování vazeb. Mnohdy přináší přirozenou prezentaci zpracovávaných dat.

Relační model s sebou přináší nové pojmy, které jsou základem relačního modelu. Jedná se o primární klíč a cizí klíč. U obr. 1 jsou názorně ukázané.

Primární klíč

Cizí klíč

Základní struktura:

Ke konstruování relační model používá pouze databázovou relaci. Z matematiky je dáno, že jsou-li $D_1 \ldots D_n$, $n \geq 1$ domény, relací je libovolná, obecně neuspořádaná podmnožina kartézského součinu domén $D_1 \times \ldots \times D_n$ prvky $(a_1 \ldots .a_n)$, kde hodnoty a_i jsou z D_i, pro i z <1,n>.

Pojem relace v matematice se od pojmu databázové relace liší ve dvou aspektech.

- Schéma relace se říká relaci, která je vybavena pomocnou strukturou. Schéma je složené ze jména relace, jmen atributů a domén.
- Množiny nebo-li jednotlivé prvky domén, ze kterých se berou jednotlivé komponenty prvků relace jsou atomické hodnoty (pojem atomická hodnota znamená, že hodnota nemůže být relačním SŘDB dekomponována do menších částí).

Lze tedy říct o schématu relace R nad množinou atributů $\Omega = \{A_1:D_1, \ldots, A_n:D_n\}$, kde D_i jsou domény a A_i jsou jména atributů. Atribut relace se říká dvojici $A_i:D_i$. Schéma relace R lze napsat jako R ($A_1:D_1\ldots..$, $A_n:D_n$).

Základní výhodou u relačních databází je relativně snadné propojování tabulek, s kterými se spojena možnost dotazů a snadná modifikace. Za nevýhodu lze považovat nízkou efektivnost zpracování. Projeví se to tak, že mnoho příkazů používá velké množství přístupů na disk, což zpomaluje zpracování.

Př:

Relační schéma: Zaměstnanec (ID_zamestnance, Jmeno, Prijmeni)

Relace: Zaměstnanec = {(111, Jana, Zámková) , (112, Jiří, Kába)}

Zobrazení relace pomocí tabulky 1:

Tab. 1 Zaměstnanec

ID_zamestnance	**Jmeno**	**Prijmeni**
111	Jana	Zámková
112	Jiří	Kába

Zdroj: vlastní

4.3.1 Relační algebra

Od té doby, co Codd v 70.letech navrhl použití relační algebry jako základ pro dotazovací jazyk, nachází výrazné uplatnění při vývoji dotazovacích jazyků. Relační algebra nám slouží pro manipulaci s relacemi. Mezi nejpopulárnější patří strukturovaný dotazovací jazyk SQL.

Relační algebra je procedurální dotazovací se základními operacemi. Mezi tyto operace se řadí množinové operace: sjednocení, rozdíl, průnik a kartézský součin. Kromě těchto základních operací se relační algebra skládá i z dalších pomocných nebo pouze

odvozených operací typických pro relační databáze. Patří sem selekce, projekce, přejmenování, spojení a rozdělení. Operace relační algebry jsou tedy operace teorie množin s dalšími operátory, které berou v potaz specifickou povahu vztahů.

Sjednocení

Definice: $r \cup s = \{n \mid n \in r \text{ or } n \in s\}$

Cílem sjednocení je sjednotit všechna fakta z argumentů. To znamená, že pokud se nacházejí v tabulkách shodné řádky, objeví se pouze jednou ve výsledné tabulce.

Rozdíl

Definice: $r - s = \{n \mid n \subset r \text{ and } n \not\subset s\}$

Výsledkem rozdílu je relace, která obsahuje všechny *n*-tice v R, ale neobsahuje je S.

Průnik

Definice: $r \cap s = \{n \mid n \in r \text{ and } n \in s\}$

Výsledkem průniku je relace, která zahrnuje všechny *n*-tice, které jsou v obou R a S.

Kartézský součin

Definice: $r \times s = \{n\ q \mid n \in r \text{ and } q \in s\}$

Kartézský součin je relace, která spojuje řádky z obou tabulek. Systém spojování je každý s každým. Počet sloupců z výsledné tabulky je dán součtem počtu sloupců obou vstupních tabulek. Počet řádků je dán součinem počtů řádků obou vstupujících tabulek.

V relační algebře jsou kro množinových operací definovány i speciální operace:

Projekce - $\Pi_{P1, P2, ..., Pk}(n)$, kde n je jméno relace a kde P_1, P_2 jsou jména atributů.

Výběr sloupců z relace A do relace B, kdy zvolené sloupce jsou dané jmenným seznamem. Nová tabulka vznikne tím, že z výchozí tabulky vyberou určité sloupce. V nově vzniklé tabulce se pak vymažou všechny duplicitní řádky.

Selekce - $\sigma_P(r)$

Def: $\sigma_p(r) = \{n \mid n \in r \text{ and } P(n)\}$

Výběr řádků z relace A do relace B vznikne tabulka se shodným záhlavím jako tabulka vstupující, ale s nižším počtem řádků a na základě definované podmínky. Řádky se vybírají podle určitého atributu, jehož hodnoty se porovnávají se srovnatelným atributem nebo konstantou.

Podmínka je zadaná Booleovským výrazem, pomocí logických spojek (and, or, not…) atomických formulí mající tvar *n1 θ n2* kde $\theta = \{<,>,=,\leq,\geq,<>\}$, t_i je buď jméno atributů nebo konstanta. Tyto formule se nazývají jednoduché podmínky.

Spojení

Jsou dvě relace A $(X_1,X_2,\ldots X_m,Y_1,Y_2,\ldots Y_n)$

B $(Y_1,Y_2,\ldots Y_n, Z_1,Z_2,\ldots Z_p)$

=> relace mají shodnou podmnožinu atributů Y $(Y_1,Y_2,\ldots Y_n)$, avšak relace A je obohacena o X $(X_1,X_2,\ldots X_m)$ a relace B je obohacena o Z $(Z_1,Z_2,\ldots Z_p)$.

Relace se záhlavím (X,Y,Z), je přirozené spojení relací A a B. Základem pro relace se stává součet (m+n+p)-tic, které mají shodnou hodnotu atributu Y jak v relaci A, tak v relaci B.

Vznikne AxB.

Pro každý atribut Y_i, který se nachází jak v relaci A tak v relaci B, vybere z AxB ty řádky, pro které platí $A.Y_i = B.Y_i$.

Zmizí pro každý atribut Y_i sloupec $B.Y_i$.[1, 2, 7]

4.4 Objektově relační model

Mezi základní datové typy v tradičních relačních systémech jsou znak, řetězec, celé číslo, číslo s plovoucí desetinou čárkou, datum a čas. Navrhovateli databáze ORDBMS umožňují definovat nové základní typy. Pro úplné využití rozšiřitelnosti datového typu, musí systém rovněž umožnit tvorbu uživatelem definovaných operátorů a funkcí.

Atributy v relačním modelu jsou tradičně atomové. ORDBMS podporují komplexní objekty, které tvoří shluky hodnot jiných datových typů. Existují zde mechanismy pro definici složitých objektů. V objektově relační databázi byla zavedena dědičnost, aby mohl být opět použit definovaný komplexní objekt a funkce definované uživatelem.

Je také umožněno definovat podtyp stávajícího typu, kdy nový typ zdědí data a funkce od jeho nadtypu. Složitější a větší aplikace vyžadují integritní omezení, které je řešeno pomocí pravidel a ty jsou spojené s událostí. Pokud dojde k události, operace spojené s pravidlem se provedou. Pravidla se používají pro zajištění konzistentního stavu databáze.[2, 6, 12]

4.5 Objektový model

Poskytuje podporu objektům, které jsou modelované v databázových aplikacích. Jeden z nejzákladnějších pojmů objektového modelování dat je objekt. Ten představuje entitu zájmu v konkrétní aplikaci. Stav objektu popisuje specifické strukturální vlastnosti objektu. Chování objektu definuje metody, které se používají k manipulaci s objekty. Implementace metody lze změnit bez způsobu jakým je rozhraní používané v kódu aplikace. Vliv na změnu implementace metody nemá ani rozhraní třídy. Rozdíl mezi klasickým a objektovým přístupem tvorby modelu dat si lze vysvětlit následovně: *„Při tvorbě datového modelu klasickým způsobem se snažíme prvky reálného světa zobrazit do předem připravených struktur pevně daného druhu. U objektů je tomu obecně; pro prvky reálného světa si vytváříme nové objekty, které se jim podobají.“* [12]

4.5.1 Objektové normální formy

- **1 objektově normální forma**
  Pokud je třída v první objektově normální formě, pak neobsahuje skupinu opakujících se atributů v objektech. Tyto atributy se musí vyčlenit do objektů nové třídy. Je zapotřebí skupinu opakujících se atributů nahradit pomocí jedné vazby na kolekci objektů nové třídy.
- **2 objektově normální forma**
  Pokud je třída v druhé objektově normální formě, pak neobsahuje atribut ani skupinu atributů v objektech, které by mohly být sdílené s jiným objektem. Tyto sdílené atributy je třeba vyčlenit do objektů nové třídy. Je zapotřebí je nahradit ve všech objektech, kde se nacházeli jednou vazbou na objekt nové třídy.
- **3 objektově normální forma**
  Pokud je třída v třetí objektově normální formě, pak neobsahují atribut nebo skupinu atributů v objektech, které mají vlastní význam bez ohledu na objektu, ve které jsou umístěný. [2, 6, 10]

4.6 XML

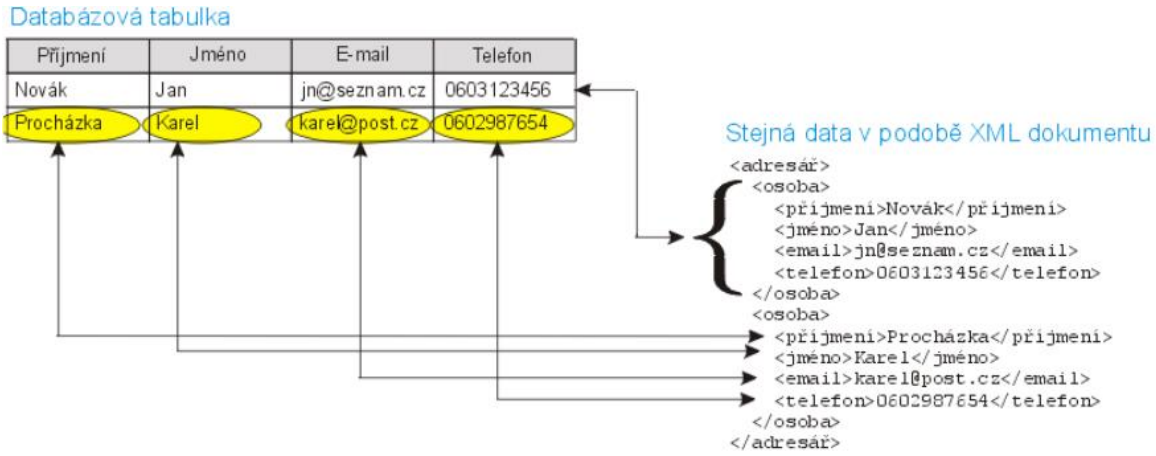
Jazyk XML je nový formát dokumentů. Předepisuje nám, jak zapsat data společně s jejich významem. Je jednoduchý formát (jednoduchý značkovací jazyk), je rozšířitelný (není zde omezení nějakou množinou elementů), je otevřený a je fakticky podporován jak

velkými, tak i malými firmami. Stává se z něho univerzální formát, který postupně nahrazuje nyní používané formáty.

XML rovnou počítá s podporou všech možných jazyků. Není tak úzce svázaná s angličtinou jako většina předchozích počítačových technologií. Jako znaková sada se používá ISO 10646, což je 32bitová znaková sada, která dokáže pojmout všechny dnes používané znaky všech jazyků. Můžeme zde vytvářet dokumenty, které mají texty v několika světových jazycích najednou – např. čeština, angličtina, němčina a ruština dle libosti. Pokud je dokument napsaný v pouze jednom jazyce, pak by zbytečně zabíral místo v ISO 10646. Kvůli tomu XML dokument je možné ukládat v jakémkoliv kódování (např. windows-1250). Syntaxe zápisů v XML oproti SGML je poměrně přísná, což umožní mnohem snazší a levnější vývoj aplikací, které umožňují pracovat s XML. Pochází z oblasti, která se zaměřuje na uchovávání a zpracování textových dokumentů. Přesně pro tento účel se skvěle hodí.

V dokumentu pomocí XML značek vyznačujeme význam jednotlivých částí textu, kdy říkáme „toto je název výrobku, tohle zase telefonní číslo a tohle je naše adresa“. Názorná ukázka je vidět na obr. 6. Dokumenty obsahují mnohem více informací, oproti prezentačnímu značení – tohle je písmem Times New Roman o velikosti 12 bodů zarovnaného na střed. [8]

Obr. 6 obsah tabulky relační databáze v XML kódu



Zdroj: https://www.kosek.cz/clanky/swn-xml/uvod.html#c38b1b4b6

4.7 UML

Po rozšíření objektového programování se hledal způsob, jak se z různých pohledů dívat na informační systémy. Během 90. let se firmě Rational Software podařilo pomocí sjednocení několika metodik vytvořit standart UML. Čas, ale ukázal, že o takový standard mají velké korporace skutečný zájem. Důvod je prostý. Jeden standard usnadní nejen samostatnou komunikaci, ale i dokumentaci projektů všichni pak rozumí. Nezaleží na tom o jaký typ projektu se jedná. Mezi velké výhody se řadí zejména náklady školení a komunikaci, které se pochopitelně výrazně sníží. Na specifikace standardu UML dohlíží mezinárodní konsorcium OMG (Object management Group).

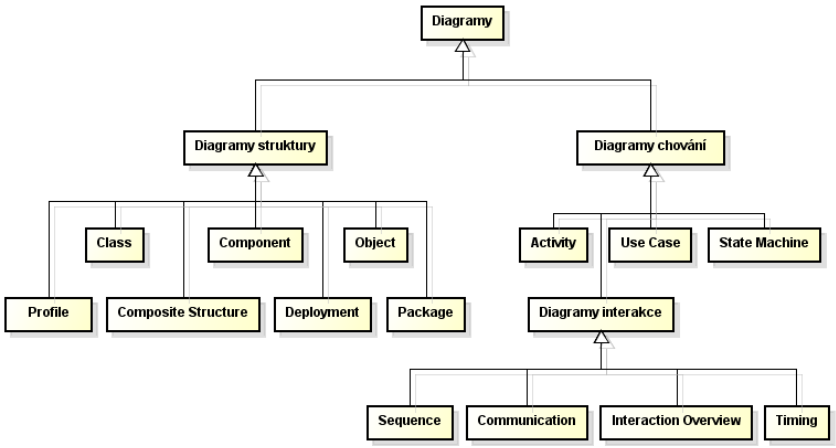
UML je jeden z všestranných nástrojů. Pomůže nám nejen ve fázi analýzy, kdy je zapotřebí řešit s klientem, co a jak teda budeme programovat, ale velmi dobře poslouží ve fázi designu. Konkrétně usnadňuje otázku, jak to vlastně naprogramovat.

UML lze použít jako:

- Náčrt
  - Obvykle se jedná o rukou nakreslené diagramy, které nám usnadňují komunikaci. Grafické vyjádření problémů nám lépe pomůže vyjádřit situaci. Velmi důležitá vlastnost diagramů je abstrakce. Jednotlivý diagram je vlastně určitý pohled na systém, z určitého úhlu. Zobrazíme pouze to co je v danou situaci potřebné a zbytek zanedbáme.
  - UML nám snižuje riziko, že v komunikaci něčemu jinak porozumíme.
- Plán
  - Je mnohem detailnější než samotný náčrt. Slouží jako plán implementace pro programátory a diagramy se vytvářejí v CAD nástrojích. Díky diagramům, se programátoři lépe orientují v systému. Pokus systém dokončíme, pak diagramy slouží dále jako dokumentace. UML je standardem, tzn., že i programátor začátečník se bude schopen orientovat v systému.
- Programovací jazyk
  - Pokud je UML diagram detailní, pak lze vygenerovat šablonu kódu, která slouží jako základ pro implementaci. Tyto modely se běžně používají v databázích pro vygenerování základních skriptů.

V současné době se UML skládá ze 14 diagramů, kde každý diagram plní svůj daný účel. Zobrazení všech diagramů je na obr. 7. S diagramy by se mělo pracovat, protože mají jistou přidanou hodnotu. [3]

Obr. 7 UML diagramy



Zdroj: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

5 Posouzení vhodnosti pro typické aplikace

5.1 Objektová databáze

Program pro napsání objektové databáze je zvolen Daskalos. Tento program vymyslel a napsal doc. Ing. Vojtěch Merunka, Ph.D. Je naprogramován jako samostatný balík systémů VisualWorks / Smalltalk verze 7.4. V Daskalosu je možné vytvářet třídy, programové metody a více objektů s reálnými daty. Umožňuje také testování objektů, kdy objekty a třídy objektů jsou demonstrovány podle standardu UML. S takovým obsahem zobrazení můžeme přímo pracovat.

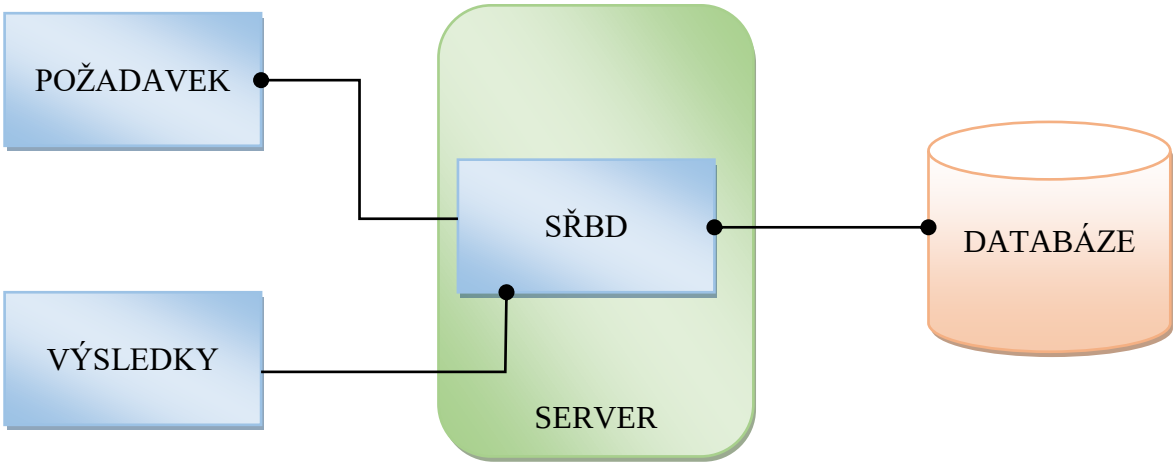
Objekty je možné zobrazit v samostatných oknech anebo přímo řešit atributy objektů metodou přetažení. U složitějších aplikací s objekty je možné použít pracovní panel, ve kterém můžeme vyhodnocovat výrazy a pracovat s jejich výsledky – např. nastavení dotazů na mnoho objektů. Objekty, které jsou nezbytné pro testování, mohou být generovány nejen vizuálně, ale také ze zdrojového kódu tradičnějším způsobem: třídy a mnoho objektů, s nimiž je manipulováno. Jsou demonstrovány formou třídních diagramů. Symboly tohoto diagramu, jejich obsahy a vazby mezi nimi jsou synchronizovány se skutečným obsahem objektů pracovního panelu. Projekty navržené v Daskalosu jsou uloženy v datovém souboru ve formátu XML, ale také je generován dokumentací obsaženého zdrojového kódu, dat, a také diagramů ve formátu HTML.[9]

Příklad takové databáze nalezete v přílohách.

5.2 SQL

Jazyk SQL (Structured Query Language) je nástroj pro správu, organizování a získávání dat obsažených v relační databázi. Pro získávání dat z databáze vytvoří dotaz v tomto jazyce a odesílá ho do systému řízení báze dat. V SŘBD (systém řízení báze dat) je zpracován, proveden a vrací se výsledná hodnota. Průběh dotazování je zobrazen na obr. 8.

Obr. 8 Příklad průběhu dotazování



Zdroj: vlastní

Jazyk SQL není pouze nástroj na dotazování. Lze jej použít k řízení všech funkcí, které databázový systém poskytuje jako je definice dat, manipulace s daty, řízení přístupu nebo řízení integrity dat. Datové typy SQL jsou vyjádřeny v tab. 2. Samotný jazyk SQL je tedy integrovanou součástí relačního databázového systému a slouží jako nástroj pro komunikaci s ním. Ve srovnání s tradičními programovacími jazyky jako je C, C++ nebo Java, je nutné říci, že SQL není úplným jazykem. Chybí mu například příkazy pro řízení běhu programu (např. cyklus FOR). Příkazy v SQL se učí velmi lehce jelikož vypadají jako jednoduché anglické věty. Je to dané tím, že příkazy popisují data, která se mají získat, ale nespecifikují, jak se data mají získat. Jazyk SQL nám tedy říká, co chceme, ale neříká nám jak toho dosáhnout. To je úkol pro samotné SŘBD.[5, 7, 10]

Datové typy SQL:

Tab. 2 Datové typy SQL

Příkaz	**Popis**
int	celá čísla v délce od -2147483648 do 2147483647
smallint	celá čísla v délce od -32768 do 32767
tinyint	celé číslo v rozsahu od 0 do 255
float	číslo s plovoucí řádovou čárkou
char(n)	Znakové řetězce v délce n znaků (maximálně však 255 znaků)
varchar(n)	znakový řetězec o maximální délce n (maximálně však 255 znaků)
decimal (n)	desetinné číslo s n platnými číslicemi
decimal(n,m)	desetinné číslo s n platnými číslicemi a s m desetinnými místy
datetime	údaj o datu a čase ve formátu RRRR-MM-DD HH:MM:SS
time	Časový okamžik dne HH:MM:SS
date	Datum ve formátu dle nastavení RRRR-MM-DD
blob, image	speciální typy pro ukládání obsáhlých binárních dat

Zdroj: vlastní

Jazyk SQL rozdělujeme do několika hlavních kategorií. K definování databázové struktury se používají příkazy jazyka DDL (Data Definition Language), které jsou vidět v tab. 3. Pro dotazování a k úpravě dat slouží příkazy jazyka DML (Data Manipulation Language) – zobrazené v tab. 4. Další jazyk je pro správu dat DCL a příkazy pro řízení transakcí TCC.

DDL

Tab. 3 Příkazy jazyka DDL

Příkaz	**Popis**
CREATE	Vytváření nových objektů
ALTER	Změny existujících objektů
DROP	Odstraňování objektů

Zdroj: vlastní

DML

Tab. 4 Příkazy jazyka DML

Příkaz	**Popis**
SELECT	Vybírá data z databáze, umožňuje podmnožiny a řazení dat
INSERT	Vkládá do databáze nová data
UPDATE	Edituje data v databázi
MERGE	Kombinace INSERT a UPDATE, data buď vloží nebo upraví
DELETE	Odstraňuje záznamy z databáze
EXPLAIN	Pomáhá uživateli optimalizovat příkazy tak, aby byly rychlejší
SHOW	Umožňuje zobrazit databáze, tabulky nebo jejich definicie

Zdroj: vlastní

Databáze často obsahují choulostivé informace, které je zapotřebí chránit. I zde lze nastavit kdo k těmto datům bude mít přístup a kdo ho bude mít odepřen. Takové příkazy se nacházejí v TCC příkazech a zobrazené jsou v tab. 5.[11]

TCC

Tab. 5 příkazy jazyka TCC

Příkaz	**Popis**
GRANT	Přidělení oprávnění k objektům
REVOKE	Odebrání práv uživateli

Zdroj: vlastní

5.2.1 Praktická ukázka SQL v Microsoft Access

V databázi je popsán zkráceně jak funguje takový internetový odchod. Ke správnému chodu jsou zapotřebí zaměstnanci, zákazníci, zboží a výrobci. To jsou názvy našich tabulek, s kterými jsme pracovali.

Pomocí klasického SQL příkazu CREATE TABLE *název tabulky,* se vytvoří tabulka.

```
CREATE TABLE Zakaznici
(
Z_ID int NOT NULL PRIMARY KEY,
Jméno varchar(50),
Příjmení varchar(50),
Ulice varchar(50),
ČP int,
PSČ int,
Město varchar(50),
email varchar(30),
telefon int,
ID_zbozi int
)
```

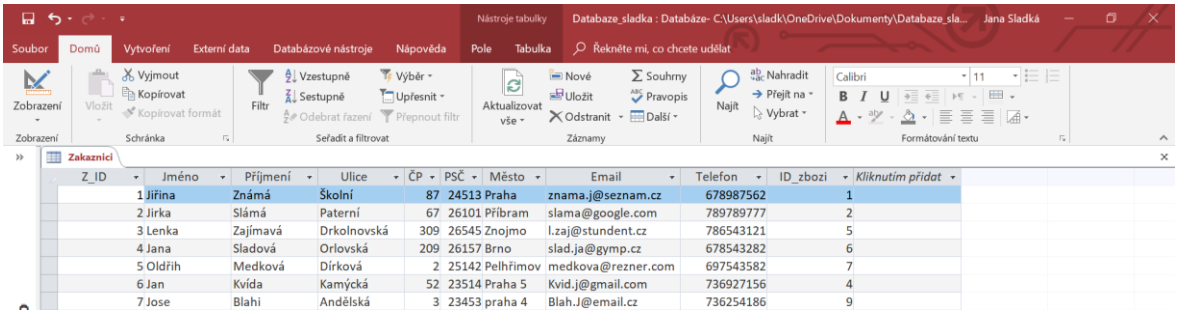
Tímto příkazem byla vytvořena tabulka s názvem Zakaznici, kde se nachází jméno, příjmení, bydliště, zboží a banka zákazníka. Takovéto údaje jsou nezbytné pro internetové obchody, aby mohly zboží dále připravit a odeslat. Pokud napíšeme k ID NOT NULL PRIMARY KEY, pak to znamená, že ID zákazníka musí být zadané a zároveň jde o primární klíč.

Po vytvoření takové tabulky se musí naplnit daty. K naplnění se použije příkaz INSERT.

INSERT INTO Zakaznici VALUES (1, 'Jiřina', 'Známá', 'Školní', 87, 24513, 'Praha', 'znama.j@seznam.cz', 678987562, 1);

Tímto příkazem se naplnil první řádek tabulky. Pro další zákazníky se použije stejný příkaz. Pokud se přidají všichni zákazníci, pak to vypadá jako obr.9.

Obr. 9 Tabulka Zakaznici s daty



Z_ID	Jméno	Příjmení	Ulice	ČP	PSČ	Město	Email	Telefon	ID_zbozi	Kliknutím přidat
1	Jiřina	Známá	Školní	87	24513	Praha	znama.j@seznam.cz	678987562	1	
2	Jirka	Slámá	Paterní	67	26101	Příbram	slama@google.com	789789777	2	
3	Lenka	Zajímavá	Drkolnovská	309	26545	Znojmo	l.zaj@stundent.cz	786543121	5	
4	Jana	Sladová	Orlovská	209	26157	Brno	slad.ja@gymp.cz	678543282	6	
5	Oldřih	Medková	Dírková	2	25142	Pelhřimov	medkova@rezner.com	697543582	7	
6	Jan	Kvída	Kamýcká	52	23514	Praha 5	Kvid.j@gmail.com	736927156	4	
7	Jose	Blahi	Andělská	3	23453	praha 4	Blah.J@email.cz	736254186	9	

Zdroj: vlastní

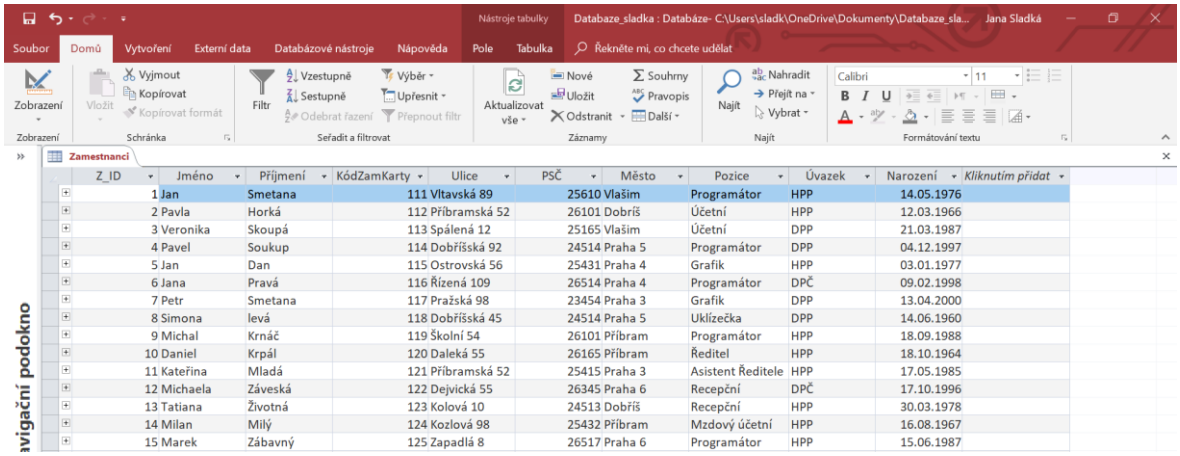
Stejným způsobem se vytvoří i zbývající tabulky. Vždy se nejdříve vytvoří tabulky a vzápětí se do nich vkládají data.

```
CREATE TABLE Zamestnanci
(
ID int NOT NULL PRIMARY KEY,        PSČ int,
Jméno varchar(50),                  Město varchar(30),
Příjmení varchar(50),               Pozice varchar(20),
KódZamKarty int,                    Úvazek varchar(20),
Ulice varchar(30),                  Narození date,
```

INSERT INTO Zamestnanci VALUES (1, 'Jan', 'Smetana', 111, 'Vltavska 89', 25610, 'Vlašim', 'Programátor', 'HPP', 1976-05-14);

Obrázek 10 zobrazuje kolik zaměstnanců ve firmě pracuje, základní informace o nich a druh úvazku, na který jsou zaměstnáni. K vytvoření tabulky a vložení záznamů, byly použité stejné přízkazy jako u tabulky Zakaznici.

Obr. 10 Tabulka Zamestnanci s daty



Z_ID	Jméno	Příjmení	KódZamKarty	Ulice	PSČ	Město	Pozice	Úvazek	Narození	Kliknutím přidat
1	Jan	Smetana		111 Vltavská 89	25610	Vlašim	Programátor	HPP	14.05.1976	
2	Pavla	Horká		112 Příbramská 52	26101	Dobříš	Účetní	HPP	12.03.1966	
3	Veronika	Skoupá		113 Spálená 12	25165	Vlašim	Účetní	DPP	21.03.1987	
4	Pavel	Soukup		114 Dobříšská 92	24514	Praha 5	Programátor	DPP	04.12.1997	
5	Jan	Dan		115 Ostrovská 56	25431	Praha 4	Grafik	HPP	03.01.1977	
6	Jana	Pravá		116 Řízená 109	26514	Praha 4	Programátor	DPČ	09.02.1998	
7	Petr	Smetana		117 Pražská 98	23454	Praha 3	Grafik	DPP	13.04.2000	
8	Simona	levá		118 Dobříšská 45	24514	Praha 5	Uklízečka	DPP	14.06.1960	
9	Michal	Krnáč		119 Školní 54	26101	Příbram	Programátor	HPP	18.09.1988	
10	Daniel	Krpál		120 Daleká 55	26165	Příbram	Ředitel	HPP	18.10.1964	
11	Kateřina	Mladá		121 Příbramská 52	25415	Praha 3	Asistent Ředitele	HPP	17.05.1985	
12	Michaela	Záveská		122 Dejvická 55	26345	Praha 6	Recepční	DPČ	17.10.1996	
13	Tatiana	Životná		123 Kolová 10	24513	Dobříš	Recepční	HPP	30.03.1978	
14	Milan	Milý		124 Kozlová 98	25432	Příbram	Mzdový účetní	HPP	16.08.1967	
15	Marek	Zábavný		125 Zapadlá 8	26517	Praha 6	Programátor	HPP	15.06.1987	

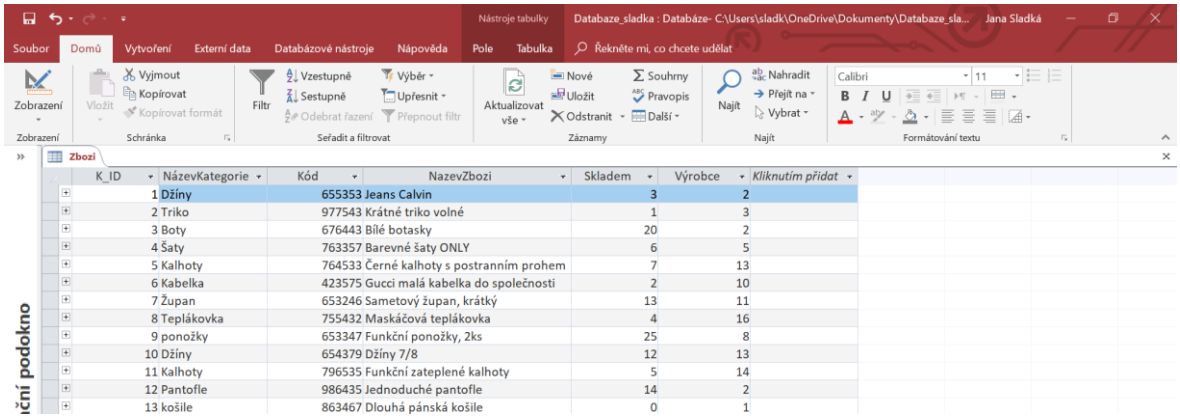
Zdroj:vlastní

```
CREATE TABLE Zbozi
(
Zb_ID int NOT NULL PRIMARY KEY,
NazevKategorie varchar(50),
Kod int,
NazevZbozi varchar(200),
Skladem int,
Vyrobce int
)
```

INSERT INTO Zbozi VALUES(1, 'Džíny', 655353, 'Jeans Calvin', 3, 2)

Zboží je základ veškerého e-shopu. Ke vkládání dat opět použijeme příkaz INSERT. Pouze s jedním zbožím by eshop dlouho nefungoval. Je tedy potřeba mít od každého kus. Jaké zboží eshop nabízí se na obr. 11.

Obr. 11 Tabulka Zbozi s daty



K_ID	NázevKategorie	Kód	NazevZbozi	Skladem	Výrobce	Kliknutím přidat
1	Džíny	655353	Jeans Calvin	3	2	
2	Triko	977543	Krátné triko volné	1	3	
3	Boty	676443	Bílé botasky	20	2	
4	Šaty	763357	Barevné šaty ONLY	6	5	
5	Kalhoty	764533	Černé kalhoty s postranním prohem	7	13	
6	Kabelka	423575	Gucci malá kabelka do společnosti	2	10	
7	Župan	653246	Sametový župan, krátký	13	11	
8	Teplákovka	755432	Maskáčová teplákovka	4	16	
9	ponožky	653347	Funkční ponožky, 2ks	25	8	
10	Džíny	654379	Džíny 7/8	12	13	
11	Kalhoty	796535	Funkční zateplené kalhoty	5	14	
12	Pantofle	986435	Jednoduché pantofle	14	2	
13	košile	863467	Dlouhá pánská košile	0	1	

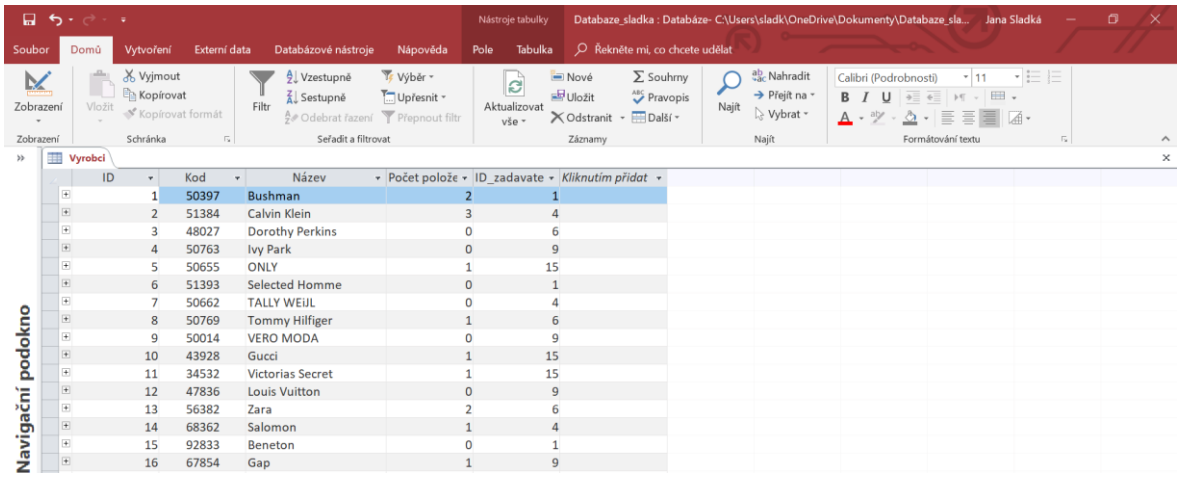
Zdroj: vlastní

Mnoho lidí vyhledává na internetu podle svého oblíbeného výrobce. Tudíž tabulka s výrobci je nezbytná. Tabulka s Vyrobci je na obr. 12. Jeden výrobce nestačí, tudíž využitím příkazu INSERT naplníme tabulku všemi našimi výrobci.

```
CREATE TABLE Vyrobci
(                                   PočetPoložek int,
V_ID int NOT NULL PRIMARY KEY,      ID_zadavatele int
Kod int,                            )
Název varchar(200),
```

```
INSERT INTO Vyrobci VALUES (1, 50397, 'Bushman', 2, 1)
```

Obr. 12 tabulka Vyrobci s daty



ID	Kod	Název	Počet polože	ID_zadavate	Kliknutím přidat
1	50397	Bushman	2	1	
2	51384	Calvin Klein	3	4	
3	48027	Dorothy Perkins	0	6	
4	50763	Ivy Park	0	9	
5	50655	ONLY	1	15	
6	51393	Selected Homme	0	1	
7	50662	TALLY WEiJL	0	4	
8	50769	Tommy Hilfiger	1	6	
9	50014	VERO MODA	0	9	
10	43928	Gucci	1	15	
11	34532	Victorias Secret	1	15	
12	47836	Louis Vuitton	0	9	
13	56382	Zara	2	6	
14	68362	Salomon	1	4	
15	92833	Beneton	0	1	
16	67854	Gap	1	9	

Zdroj: vlastní

Dotazy

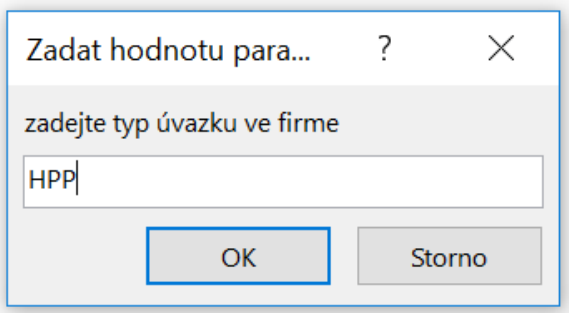
1. Typ úvazku zaměstnance

   Příkaz slouží k tomu, aby si zaměstnavatel nemusel vybírat z dlouhé databáze, pokud potřebuje pouze zaměstnance, kteří pracují na HPP, DPP nebo DPČ, což je zobrazeno na obr. 13. Po zadaní typu uvazku ve firmě vyjede pouze tabulka, kde jsou požadovaní zaměstnanci, a výsledek je vidět na obr. 14.

   Je to uspořádané tak, aby jako první byl kód zaměstnanecké karty, dále dochází ke sjednocení políček Jméno a Příjemení, mezi které je přidaná mezera pro přehlednost. Další sloupce jsou narození, celá adresa a pozice zaměstnance. Sjednocení bylo použito i v adrese, což slouží pro opěr pro přehlednost.

```
SELECT
    Zamestnanci.KódZamKarty, [Zamestnanci]![Jméno]+" "+[Zamestnanci]![Příjmení]
    AS [Jméno zamestnance], Zamestnanci.Narození, [Zamestnanci]![Ulice]+",
    "+[Zamestnanci]![Město] AS Adresa, Zamestnanci.Pozice
FROM
    Zamestnanci
WHERE
    (((Zamestnanci.Úvazek)=[zadejte typ úvazku ve firme]));
```

Obr. 13 Zadání typu úvazku ve firmě



Zdroj: vlastní

Obr. 14 Tabulka po zadání typu úvazku ve firmě

Úvazek_zaměstnance

KódZamKarty	Jméno zamestnance	Narození	Adresa	Pozice
115	Jan Dan	03.01.1977	Ostrovská 56, Praha 4	Grafik
119	Michal Krnáč	18.09.1988	Školní 54, Příbram	Programátor
120	Daniel Krpál	18.10.1964	Daleká 55, Příbram	Ředitel
121	Kateřina Mladá	17.05.1985	Příbramská 52, Praha 3	Asistent Ředitele
123	Tatiana Životná	30.03.1978	Kolová 10, Dobříš	Recepční
124	Milan Milý	16.08.1967	Kozlová 98, Příbram	Mzdový účetní
125	Marek Zábavný	15.06.1987	Zapadlá 8, Praha 6	Programátor
111	Jan Smetana	14.05.1976	Vltavská 89, Vlašim	Programátor
112	Pavla Horká	12.03.1966	Příbramská 52, Dobříš	Účetní

Zdroj: vlastní

2. Spojování tabulek, vytvoření relace

Dotaz JOIN slouží k propojení tabulky zbozi s tabulkou Vyrobci obr. 15. Pokud rozklineme řádek, pak se nám zobrazí informace o výrobcích. Propojení je následně zobrazené v relaci. Relace je 1:N nebo-li Vyrobci:Zbozi. V tabulce Vyrobci je ID jako primární klíč, zatímco v tabulce Zbozi je ve sloupci Výrobce cizí klíč viz. Obr. 16.

```
SELECT
    Zbozi.K_ID, Zbozi.NázevKategorie, Zbozi.Kód, Zbozi.NazevZbozi, Zbozi.Výrobce
FROM
    Vyrobci INNER JOIN Zbozi ON Vyrobci.ID = Zbozi.Výrobce
WHERE
    (((Zbozi.Výrobce)=[Vyrobci].[ID]) AND ((Zbozi.Skladem)>1))
ORDER BY
    Zbozi.K_ID;
```

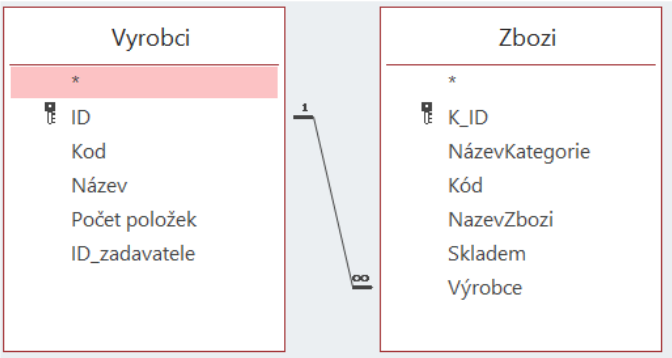
Obr. 15 propojení tabulky Vyrobci s tabulkou Zbozi

03_02_join

K_ID	NázevKategorie	Kód	NazevZbozi	Výrobce
1	Džíny	655353	Jeans Calvin	2
3	Boty	676443	Bílé botasky	2
4	Šaty	763357	Barevné šaty ONLY	5
5	Kalhoty	764533	Černé kalhoty s postranním prohem	13
6	Kabelka	423575	Gucci malá kabelka do společnosti	10
7	Župan	653246	Sametový župan, krátký	11
8	Teplákovka	755432	Maskáčová teplákovka	16
9	ponožky	653347	Funkční ponožky, 2ks	8
10	Džíny	654379	Džíny 7/8	13
11	Kalhoty	796535	Funkční zateplené kalhoty	14
12	Pantofle	986435	Jednoduché pantofle	2

Zdroj: vlastní

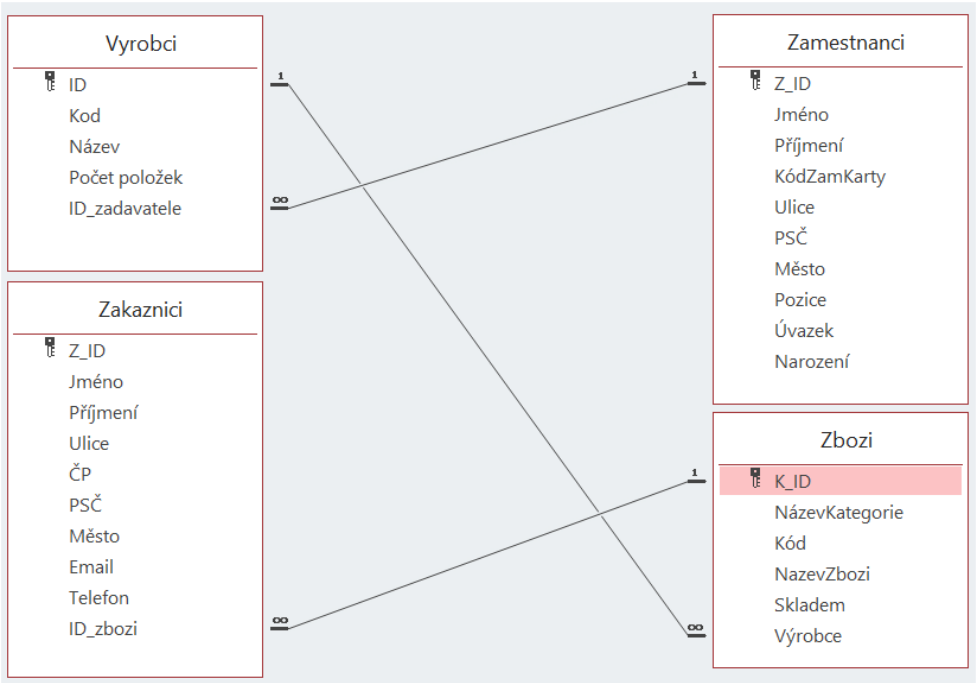
Obr. 16 Primární klíč Vyrobci je cizím klíčem tabulky Zbozi



Zdroj: vlastní

Pomocí příkazu JOIN jsme propojili i další tabulky. V těchto relacích jsou použité pouze vztahy 1:N. V tabulce výrobci je zapotřebí, aby bylo vidět jaký zaměstanec se o daného výrobce stará a vyřizuje komunikaci. V tabulce zákazníci najdeme informace o zboží, které si zákazník zakoupil. Propojení tabulek je možné vidět na obr. 17.

Obr. 17 Propojení tabulek pomocí příkazu JOIN



Zdroj: vlastní

5.3 Porovnání datových modelů

Základní charakteristiky objektového datového modelu v databázích:

1) Na rozdíl od relačního datového modelu, kde je relační tabulka jediným „druhem“ kolekce, tak objektová databáze podporuje více typů kolekcí objektů. V některých databázových systémech to může být až několik různých typů tak, jak jsou známy z knihoven objektových programovacích jazyků. Například *Set, MultiSet, Bag, Array* atd.
2) *Polymorfismus* je vlastnost programovacího jazyka, které umožňuje objektům volání jedné metody se stejným jménem, ale s různými parametry. Nachází se pouze v objektově orientovaném programování. Nevzniká pouze děděním tříd, pokud mají objekty společné atributy a jejich třídy se mezi sebou nedědí, jsou polymorfní.
3) Objektová databáze rozlišuje rozdíl mezi pojmem třída objektů a kolekce objektů, kdy třída objektů je jen realizace datového typu objektů a kolekce je jen uložiště pro objekty. Například, můžeme mít více kolekcí objektů stejného typu i množinu obsahující objekty z různých tříd, pokud takové objekty mají díky polymorfismu nějaké společné atributy, pak nám nic nebrání je držet pohromadě a použít např. selekci.
4) Objekty se skládají z metod a z vnitřních datových složek, což mohou být opět jiné objekty. Metody představují funkční stránku každého objektu. Známe nejen přístupové metody, ale i ty složitější. Přístupové metody jsou ty, které jen přímo manipulují s datovými složkami objektu, tzn. zapisují nové hodnoty datových složek nebo čtou hodnoty datových složek. Složitější metody vypočítávají z objektů taková data, která v objektu nebyla uložena jako jedna z jeho datových složek. Pro objektově orientované databáze je nutné si uvědomit, že mezi atributy objektů patří nejen jejich takové složky (jako je to u Relačních databází), ale i metody poskytující další data.
5) Každý objekt má svoji vlastní identitu, což znamená, že v rámci jednoho paměťového prostoru má každý objekt jednoznačně přidělený identifikátor označován jako OID (Object IDentifier). OID každého objektu zůstává stejný a plní úlohu ukazatele do virtuální paměti. I když se v objektu změní všechny jeho datové

složky nebo metody zachová se stejně. Nezmění se ani pokud dochází ke změnám na fyzické úrovni, jako je např. změna umístění v operační paměti. Vzhledem k existenci konceptu OID musíme rozlišovat rozdíl mezi pojmy rovnost dat objektu a totožnost objektu, jelikož dva objekty se stejnými daty zdaleka neznamenají, že jsou totožné. V objektové databází není zapotřebí objektům vytvářet primární klíče. Relační datový model toto nezná, jelikož identita záznamů je daná hodnotou atributů.

6) V objektové databázi lze pracovat v bázi dat i s takovou soustavou objektů, která je sama o sobě aplikací. Objektová databáze neslouží pouze jako uložiště dat, které pracuje s externím programem, ale algoritmy programu lze „rozpustit“ v metodách objektů. Metody jsou uloženy přímo v objektové databázi. Vytvoření databázové aplikace je pak pro klienta velice zjednodušená. V extrémním případě se muže jednat jen o prezentační rozhraní výpočetního systému. Celý pracuje uvnitř objektového databázového serveru.
7) Rozdíl od běžných objektových programovacích jazyků je, že objekty v objektové databázi mohou migrovat mezi různými třídami a v systému mohou současně existovat třídy s více verzemi. Podle přístupových práv mohou mít různí uživatelé dostupné různé atributy na stejných objektech.

Výhody relačního modelu:

1) Flexibilní struktura, kterou lze měnit i za běhu, pouhým odebráním či přidáním tabulky a relace
2) Pomocí relačního vztahu lze snadno vybrat z několika tabulek najednou
3) Relační databázové modely jsou velice rozšířené a podporované
4) Široká možnost podpory nejrůznějšími produkty
5) Pro přístup k datům se využívá jazyka SQL

Charakteristika objektově relačního modelu a objektově orientovaného modelu:

1) Objektově relační model doplňuje relačně datový model o možnost práce s některými datovými strukturami. Takové struktury známe z objektově orientovaného programování. Tuto variantu si zvolila většina výrobců velkých relačních databázových systémů jako je např. Oracle. V jádru, ale zůstává na

principech původního relačního datového modelu. Ve vývoji se představuje objektově relační model jako evoluční trend vývoje.

2) Objektově orientovaný model představuje revoluční trend vývoje. Jedná se o nový model, který vznikl z původního síťového datového modelu, který je doplněn o možnosti práce s objekty, tak jak je zvykem u objektového programování.

V tabulce 6 je porovnání relačního datového modelu a objektového datového modelu.

Tab. 6 Porovnání relačního a objektového datového modelu

Relační datový model	**Objetkový datový model**
Záznam	Objekt
Tabulka	Třída objektů (jako datový typ) Kolekce objektů (i z různých tříd)
Atribut	Datová složka objektu Metoda objektu, která vypočítává data
Primární klíč	OID (identifikátor objektu)
Propojení dvou záznamů dvou tabulek tak, že hodnota primárního klíče u jednoho záznamu je shodná s hodnotou cizího u druhého záznamu.	Skládání objektů, datovou položkou objektu je celý objekt a ne pouze hodnota jeho „klíče“.

Zdroj: vlastní

Nejrozšířenější technologií je objektově relační. Ovšem „nerelační“ objektově orientovaný model má několik přednostní. Patří mezi ně:

1) Lepší podpora datové struktury. Není zde třeba datové struktury tolik transformovat, aby jej bylo možné uložit do databáze. Prakticky již existují použitelné systémy, které dovolují zpracovávat objekty stejně jako je tomu v objektových programovacích jazycích. Mezi použitelné systémy se řadí např. Gemstone, ObjectStore, Versant a další.
2) Pokud srovnáme model s relačním datovým modelem, tak má předpoklady pro efektivnější způsoby zpracování dotazů . „Nerelační“ objektový model vychází ze síťového datového modelu. Tato vlastnost se projeví zejména u složitých datových struktur, které by se podle relačního datového modelu musely rozkládat do několika vzájemně provázaných relačních tabulek.

Větší rozšířenost relačních databází oproti objektovým může být následující:

1) Prakticky malá znalost objektových databází v komunitě tvůrců softwarů.
2) Cena systémů a jejich dostupnost na trhu. Velké relačně objektové databáze jsou levnější než ty objektové. Příkladem toho je cena objektového programovacího systému Gemstone, která je vyšší než objektově relačního systému Oracle. [6]

6 Závěr

Logické databázové modely jsou v dnešní počítačové době velice rozšířená věc a každý takovou databázi používá. Cílem bylo poukázat na fakt, že relační databáze jsou stále nejrozšířenější a stále se hledají způsoby, jak nakládat s daty efektivněji, rychleji a spolehlivěji.

V úvodu jsem se věnovala historii databází. Dále pak vhodnému popsání základních logických databázových modelů, mezi které patří relační, objektový a objektově relační databáze. Relační databáze nejsou vhodné pro ukládání objektů a naprogramovat rozhraní pro ukládání objektů je velice složité. Jsou dobré pro řízení velkého množství dat a vyhledávání dat. Bohužel poskytují nízkou podporu pro manipulaci s daty. Základem jsou dvourozměrné tabulky a vztahy mezi daty jsou vyjadřovány porovnáváním hodnot, které jsou v nich uložené. Objektová databáze umožňuje snadno pracovat s objekty. Jsou výborné pro manipulaci s daty, a navíc, pokud opomeneme programátorskou stránku, tak lze říct, že některé dotazy jsou efektivnější než u relačních databází. Je tomu tak díky dědičnosti a referencím.

Nyní se hojně využívá a prosazuje XML kód. Je vhodný jak pro začátečníka, tak i pro pokročilého uživatele programovacích technik. Výhodou XML je jeho všestranná použitelnost. Nezáleží na tom, zda ho píšete v rodném jazyce nebo v cizím jazyce. Struktura je stejná pro všechny a velice jednoduchá na přečtení, oč se vlastně jedná.

V praktické části je vidět, že použití databáze založené na relačním modelu, je velice snadné a přehledné. Zaleží zde na pořadí vkládání jednotlivých dat. Strukturovaný dotazovací jazyk, nám hezky posloužit a ukázal, jak pracuje. Bohužel u SQL nelze použít příkazy pro řízení běhu programu. Avšak předností je, že tabulky lze propojit za běhu.

Na praktické části byly dostatečně ukázány rozdíly mezi relační databází a objektovou databází. Byly také uvedené dostatečné rozdíly mezi pojmy, které se častou zaměňují.

7 Seznam použitých zdrojů

1. Historie relačních databází. [Online] Květen 2008. http://www.root.cz/clanky/historie-relacnich-databazi/.

2. databázový svět. [Online] www.dbsvet.cz.

3. Čapka, David. IT Network. [Online] 2013. https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy/,.

4. Molhanec, Martin. [Online] http://prog-story.technicalmuseum.cz/images/dokumenty/Programovani-TSW-1975-2014/1999/1999-23.pdf.

5. Jaroslav, Pokorný. Dotazovací jazyky. Veletiny : Science, 1994.

6. Merunka, Vojtěch. Objektové modelování.: Alfa Publishing, 2008. 978-80-87197-04-2.

7. David Kroenke, David J. Auer. databáze. Brno : Computer Press, 2015. 978-80-251-4352-0.

8. Kočí, Michal. Interval. [Online] 2000. https://www.interval.cz/clanky/co-je-xml/.

9. Merunka, Vojtěch. [Online] https://sites.google.com/site/vmerunka/daskalos.

10. Kosek, Jiří. PHP tvorba interaktivních internetových aplikací. Grada, 1999.

11. Groff, J., R., Weinberg, P., N. SQL kompletní průvodce. CP Books a.s., 2005.

12. Merunka, Vojtěch. DocPlayer. [Online] 2016. http://docplayer.cz/665523-Objektovy-pristup-v-databazove-technologii.html.

Přílohy

Příloha 1: Objektová databáze

Jako první při psaní objektové databáze je za potřebí zvolit si objekty. V našem případě to jsou: Personál, Zákazník a Zboží

- OPersonal :Set
- OZakaznik :Set
- OZbozi :Set

I. Třídy

OZbozi := Set new.

OPersonal := Set new.

OZakaznik := Set new.

Jakmile jsou vytvořené třídy, začnou se plnit daty. U Zboží je zapotřebí napsat popis, cenu, hodnocení a počet zboží na skladě. Pro práci s objekty je potřeba mít několik záznamů. Vždy se začíná malým písmenem např. počátečního slova třídy.

z1 := Zbozi new. z1 popis: 'klavesnice'. z1 cena: 200.	z1 hodnoceni: 'vyborne'. z1 pocet: 100. OZbozi add: z1.
z2 := Zbozi new. z2 popis: 'mys'. z2 cena: 50.	z2 hodnoceni: 'mys se seka, zadna dobra kvalita'. z2 pocet: 50. OZbozi add:z2.
z3 := Zbozi new. z3 popis: 'pocitac'. z3 cena: 20000.	z3 hodnoceni: 'lepsi pocitac jsem nemel, lehky, rychly'. z3 pocet: 10. OZbozi add:z3
z4 := Zbozi new. z4 popis: 'sluchatka'. z4 cena: 300.	z4 hodnoceni: 'Momentalne neni hodnoceni, zadnym uzivatelem'. z4 pocet: 5. OZbozi add:z4.

Stejným způsobem se postupuje i u dalších tříd. U personálu se jedná o jméno, příjmení, výše platu, data narození, jaké zboží připravoval a jestli je v práci.

```
p1 := Personal new.                          p1 datumNarozeni: '5 7 1990' asDate.
p1 jmeno: 'Jan'.                             p1 pripravil: z1.
p1 prijmeni: 'Delnik'.                       p1 jsouvPraci: 'Ano'.
p1 plat: 13000.                              OPersonal add:p1.
```

```
p2 := Personal new.                          p2 datumNarozeni: '7 6 1954' asDate.
p2 jmeno: 'Daniel'.                          p2 pripravil: z2.
p2 prijmeni: 'Ludsky'.                       p2 jsouvPraci: 'Ano'.
p2 plat: 10000.                              OPersonal add:p2.
```

```
p3 := Personal new.                          p3 datumNarozeni: '6 12 1968' asDate.
p3 jmeno: 'Simona'.                          p3 pripravil: z3.
p3 prijmeni: 'Kratka'.                       p3 jsouvPraci: 'Ne'.
p3 plat: 14000.                              OPersonal add:p3.
```

Po zadání údajů personálu zadáváme údaje zákazníků. Informace, které jsou potřeba zadat jsou následující: jméno, příjmení, adresa, telefonní číslo, email, jaké zboží si koupil, jak zakoupené zboží ohodnotil, jaký má rozpočet na zákaznickém učtu.

```
k1 := Zakaznik new.                          k1 koupil: z3.
k1 jmeno: 'Kristyna'.                        k1 hodnotil: 'splnilo to ocekavani, jsem
k1 prijmeni: 'Skoupa'.                       velice spokojena'.
k1 adresa: 'Rybarska21, Praha10, 25101'.     k1 rozpocet: 25000.
k1 telefonnicislo: 602789716.                k1 zbozi: OZbozi.
k1 email: 'skoupaK@email.cz'.                OZakaznik add:k1.
```

```
k2 := Zakaznik new.                          k2 koupil: z2.
k2 jmeno: 'Karel'.                           k2 hodnotil: 'mys nefunguje jak ma, už
k2 prijmeni: 'Koupal'.                       bych ji nekoupil'.
k2 adresa: 'Karetni11, Plzen, 31202'.        k2 rozpocet: 100.
k2 telefonnicislo: 732567824.                k2 zbozi: OZbozi.
k2 email: 'karelkou@gmail.com'.              OZakaznik add:k2.
```

```
k3 := Zakaznik new.
k3 jmeno: 'Ludek'.
k3 prijmeni: 'Karel'.
k3 adresa: 'Moravska52, KarlovyVary,
56789'.
k3 email: 'karelkou@gmail.com'.
k3 koupil: z3.
k3 hodnotil: ' '.
k3 rozpocet: 0.
k3 zbozi: OZbozi.
OZakaznik add:k3.
```

```
k4 := Zakaznik new.
k4 jmeno: 'Jana'.
k4 prijmeni: 'Karasova'.
k4 adresa: 'Dlouha 76, Praha 5, 43256'.
k4 telefonnicislo: 603975109.
k4 email: 'karaska@gmail.com'.
k4 koupil: z4.
k4 hodnotil: ' '.
k4 zbozi: OZbozi. k4 rozpocet: 1000000.
OZakaznik add:k4
```

II. Metody

Po nadefinovaných datech je zapotřebí si zvolit nějaké metody, které budou v databázi probíhat. Jednotlivé proměnné a metody jsou vidět v class diagramu na obr.18.

- Pro Zboží jsou to tyto metody:
  - Celková cena za zboží
    - Jedná se o celkovou cenu zboží na skladě
  - Dostatek zboží na skladě
    - Pokud dojde k tomu, že počet zboží je menší nebo rovný nule, pak vypíše, že není dostatek zboží na skladě
  - Levné zboží
    - Jestli cena zboží se rovna nebo je menší než tisíc, vypíše to, že zboží je levné

CelkovaCena

```
^pocet * cena
```

JeDostatek

```
pocet isNil ifTrue: [^nil].
^pocet <= 10
```

Levne

```
cena isNil ifTrue: [^nil].
^cena <= 1000
```

- Pro Personál jsou to tyto metody:
    - Hodinová mzda zaměstnance
        - Hodinová mzda zaměstnance se vyjádří z podílu platu zaměstnance a počet odpracovaných hodin
    - Plat zaměstnance
        - Pokud je plat vyšší než 10 000, dá se říct, že se má skvěle, pokud má plat nižší pak by měl více pracovat
    - Věk zaměstnance
        - Datum narození ve vztahu k aktuálnímu datu

hodinovaMzda

```
^plat // 160
```

PlatebniPodminky

```
plat > 10000 ifTrue: [^'Má se skvěle']
ifFalse: [^'Měl by se více snažit']
```

Vek

```
^Date today year - datumNarozeni year
```

- Pro zákazníka jsou to tyto metody:
    - Co si může zákazník koupit z kreditu
        - Pomocí tohoto příkazu zákazník zjistí, zda si to zboží, které si vybral může zakoupit z rozpočtu, jaký má na zákaznickém účtu nebo musí připlatit
    - Doplatek
        - Pokud cena za zboží je vyšší než jeho rozpočet, pak se vypíše přesná částka jaká je potřeba doplatit
    - Zasílání SMS nebo E-mailu

CoSiMuzeKoupit

```
^zbozi select: [:c | c cena <= self rozpocet]
```

Doplatek

```
self rozpocet > koupil cena
ifTrue: [^'0']
ifFalse: [^koupil cena - rozpocet]
```

ZaslatSMS

```
telefonnicislo = nil ifTrue: [^'Pošli mail']
ifFalse: [^'Pošli sms']
```

III. Zobrazení tříd a metod

Zde je možné vidět, jaké mají jednotlivé objekty třídy a jaké používají metody. Třídy definují jednotlivé metody. V kapitole 8.1.1 se vytvořily základní proměnné jednotlivých tříd. Je možné si povšimnout kde se použila proměnná String, Number, Date a hlavně Object. String se používá pro definovaní textového datového typu. Number se použije vždy u číselného datového typu. Date označuje datum. Object se používá při práci s objekty. U personálu se Object nachází u „pripravil“, protože je důležité vědět, kdo z personálu zboží připravil. U zákazníka se proměnná Object objevuje u „hodnotil“, „koupil“ a u „zboží“. Zákazník hodnotí zboží, jaké zboží si zakoupil, a název zakoupeného zboží.

Pod proměnnými se nachází metody, které jsou blíže popsané v kapitole II.

Zbozi
instance variables cena :Number hodnoceni :String pocet :Number popis :String
methods **CelkovaCena** cena cena: hodnoceni hodnoceni: initialize **JeDostatek** **Levne** pocet pocet: popis popis:

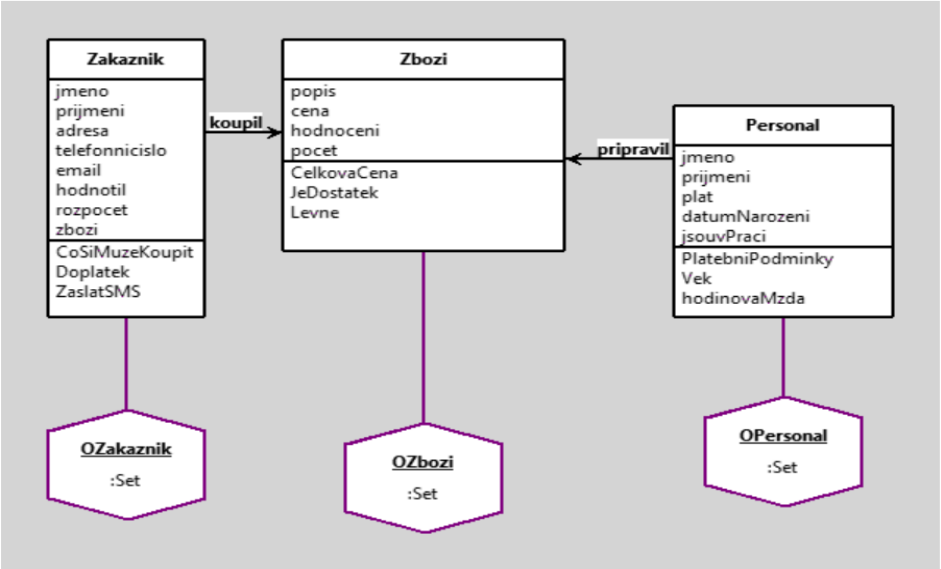
Personal
instance variables datumNarozeni :Date jmeno :String jsouvPraci :String plat :Number prijmeni :String pripravil :Object
methods datumNarozeni datumNarozeni: **hodinovaMzda** initialize jmeno jmeno: jsouvPraci jsouvPraci: plat plat: **PlatebniPodminky** prijmeni prijmeni: pripravil pripravil: **Vek**

Zakaznik
instance variables adresa :String email :String hodnotil :Object jmeno :String koupil :Object prijmeni :String rozpocet :Number telefonnicislo :Number zbozi :Object
methods adresa adresa: **CoSiMuzeKoupit** **Doplatek** email email: hodnotil hodnotil: initialize jmeno jmeno: koupil koupil: prijmeni prijmeni: rozpocet rozpocet: telefonnicislo telefonnicislo: **ZaslatSMS** zbozi zbozi:

IV. UML diagram

V HTML kódu vždy bude u tříd vytvořený diagram, kde jsou zobrazené instance a metody, které jsme použili. Na obr. 1 je vidět UML class diagram. Po nadefinování jednotlivých tříd objektů, se nadefinovaly jednotlivé atributy, a v závěru u každé třídy jsou použité metody. Z class diagramu je možné vyčíst, tzv. Zakaznik_na_zbozi. Tedy kolik zboží si zákazník zakoupil. Také si lze povšimnout, tzv. Personal_na_zboží, tedy kdo z personálu dané zboží připravoval. Pokud je šipka tatko zakreslena znamená to vztah „0:N“. Takový vztah vyjadřuje, že personál může být „prázdný“ a nebo N druhů zboží (teoreticky nekonečno).

Obr. 1 UML diagram



Zdroj: vlastní