UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

DATABÁZOVÉ SYSTÉMY V GIS

Zdena DOBEŠOVÁ

Olomouc
2004

Recenzenti: Doc. Dr. Ing. Jiří Horák
Ing. Josef Hnojil Ph.D.

1. vydání

ISBN 80-244-0891-0

Motto:

Kde je moudrost?

Ztracena ve znalostech.

Kde jsou znalosti?

Ztraceny v informacích.

T.S.Eliot

Kde jsou informace?

Ztraceny v datech.

Kde jsou data?

Ztracena v databázích.

Joe Celko

Obsah

Úvod

Toto skriptum je určeno pro studenty oboru geoinformatika. Skriptum je studijním textem předmětu „Prostorové databázové systémy“. Cílem tohoto skripta je získání základního přehledu znalostí z principů relačního databázového systému a znalostí z oblasti analýzy a návrhu databází. Stěžejní je probrání právě relačního modelu, neboť relační databázové systémy jsou dnes nejrozšířenějším typem databázových systémů pro ukládání dat.

V textu je odkazováno na praktické implementace obecných problémů v databázovém prostředí produktu MS Access, který je používán v praktických cvičeních tohoto předmětu. Tyto odkazy v textu usnadní studentům pochopení probírané látky a rovněž její procvičování.

Každý automatizovaný informační systém, tudíž i geografický informační systém se skládá z pěti základních součástí: lidí, postupů, hardwaru, softwaru a dat uložených v souborech nebo databázích. Z hlediska *ekonomického* bývají data nejnákladnější součástí informačního systému, hovoří se až o 80% z celkových nákladů na vybudování systému.

Z hlediska *časového* vykazují data nejdelší trvalost oproti hardwaru a softwaru. Samozřejmě hovoříme o datech, která jsou průběžně aktualizována. Můžeme říci, že data jsou platná dokud existuje realita, kterou popisují. V případě, že realita přestává existovat můžou data sloužit pro archivní účely. V průběhu využívání dat se data také zálohují a archivují. Při změně hardwaru a softwaru pro určitou aplikační oblast se data používají dále z původního systému. Pro nový systém se provádí potřebná konverze dat nebo často systémy v nové verzi zajišťují zpětnou kompatibilitu původních formátů dat.

Z těchto důvodů vidíme důležitou roli dat, kterou mají v geografických informačních systémech.

My se v tomto skriptu soustředíme zejména na atributová data. Způsoby ukládání těchto dat v GIS aplikacích prošly dlouhým vývojem. Původní souborové uložení dat přechází k vyžívání systémů řízení báze dat pro uložení atributových dat. V současné době umožňuje rozvoj multimediálních a objektových databází ukládat a zpracovávat grafická data společně s atributovými v databázovém prostředí bez nutnosti odděleného ukládání a správy grafické složky dat.

Děkuji doc. Dr. Ing. Jiřímu Horákovi z Institutu geoinformatiky, Hornicko-geologické fakulty Vysoké školy báňské Ostrava a Ing. Josefovi Hnojilovi Ph.D. z firmy Intergraph za pečlivé přečtení textu a za připomínky, které přispěly k jeho zdokonalení.

autorka

1. Informační systém

Informační systém je systém, který umožňuje účelné uspořádání sběru, uchování, zpracování a poskytování informací. Informační systém má dvě základní složky. První je databáze a ta druhá je komunikační systém. První složce se budeme věnovat v tomto skriptu.

Informační systémy (dále IS) prošly dlouhým vývojem. Základní mezníky v práci s daty lze pozorovat tyto: Prvním je oddělení datových struktur od programů a uložení dat v samostatných souborech. Dalším mezníkem je vypuštění popisu datové struktury z aplikačních programů a zajištění přístupu k datovým souborů prostřednictvím systémů řízení báze dat. Zároveň následuje přechod od uložení dat ve formě sekvenčních souborů k ukládání dat do relačních databází.

1.1. Co je to databáze

Databázi si můžeme představit jako místo, kam se ukládají všechny potřebné údaje. Přístup k údajům uloženým v databázi obstarává program, kterému se říká SŘBD - *Systém Řízení Báze Dat.* Tento poněkud krkolomný název vznikl přeložením původního anglického termínu DBMS - DataBase Management System. V tomto skriptu se budeme držet termínu SŘBD.

SŘBD je mezičlánkem mezi daty a aplikačním programem. SŘBD spravuje data a stará se o fyzické uložení dat. Dále zajišťuje sdílení dat nejefektivnějším způsobem tak, aby nebyla narušena integrita databáze. Řeší víceuživatelský, paralelní přístup k datům s možností nastavení uživatelských práv.

Mezi SŘBD patří takové programy jako Oracle, MS SQL Server, DB2, Sybase, Informix, Progress či InterBase. Nejedná se o programy nikterak levné. Jejich cena se pohybuje v desítkách a většinou spíše i ve stovkách tisíc korun. Na poli SŘBD existují programy šířené zdarma jako freeware - např. mySQL a PostgreSQL. Podmínky bezplatného používání je třeba pečlivě zjistit, zejména při komerčním používání.

Převážná většina dnes používaných SŘBD při uspořádání údajů v databázi vychází z *relačního modelu dat.* Název tohoto modelu vychází z relační algebry, což je matematický aparát, na kterém relační model dat staví. Bližší popis a vysvětlení je v kapitole 6.2. Relační model dat. V tomto modelu jsou údaje uspořádány do *relací.* Relace jsou reprezentovány tabulkami. Tabulka shromažďuje údaje o jednom druhu objektů.

1.2. Funkce a možnosti informačního systému

Pro prvotní představu si uveďme základní činnosti informačního systému, který pracuje s databází. Základní činností je operace obecně řečeno s evidencí, resp. můžeme říci s více evidencemi najednou. Takovou evidenci si jistě umíme všichni dobře představit. Evidenci na počítači dost často předcházela klasická papírová lístková kartotéka. Evidencí může být například seznam obyvatelstva města, evidence nemovitostí, seznam knih a map, kterými disponuje knihovna. Dalším příkladem je měření meteorologických veličin na meteorologické stanici, místenkové kanceláře pro vlaky, letadla, divadla a kina.

Jaké funkce budeme od informační systému pracující nad databázovým systémem požadovat a očekávat?

1. **Založení evidence**. To je první krok, při kterém si řekneme, co vlastně budeme evidovat, sledovat. Řekneme si například, že budeme chtít podchytit všechny občany jednoho města, přičemž budeme sledovat jejich jméno a příjmení, rodné číslo, bydliště, datum přihlášení k pobytu.
2. **Naplnění daty**. Máme-li definovánu strukturu evidence obyvatelstva, budeme do ní vkládat, zapisovat jednotlivé údaje - v našem případě budeme zapisovat informace o jednotlivých občanech.
3. **Měnit zapsaná data.** Občané se stěhují v rámci města a náš systém nám musí umožnit tyto změny provést v naší evidenci. Vedle individuálních změn (viz změna bydliště) můžeme snadno provádět i skupinové změny (například změna názvu určité ulice se musí provést u všech občanů, kteří zde bydlí).
4. **Doplnit další sledované údaje**. Na začátku jsme si řekli, co budeme naší evidencí sledovat. Ale po čase se nám může zdát, že jsme některé údaje zapomněli zařadit. Takže i toto by měl databázový program dokázat. Přidáme další atributy entit, rozšíří se struktura tabulky.
5. **Mazat data**. Občan se odstěhuje z města nebo zemře a my jej snadno vyřadíme z evidence. Opět můžeme vedle individuálního "vyškrtnutí" snadno provést i vyřazení celé rodiny (celé skupiny).
6. **Zapisovat nová data**. Narodí-li se v našem městě dítě nebo přistěhuje-li se nový občan, opět pomocí databázového programu jej snadno zaneseme do naší evidence. Přidáme nový záznam do tabulky.

7. **Vybírat údaje**. Dalším nejčastějším požadavkem je výběr dat jako odpověď na naše dotazy. Chceme zjistit, kteří občané mají 70 let věku, kteří občané jsou starší 18 let a tudíž voliči, kolik jich bydlí v centru, kolik procent je žen. Na všechny tyto dotazy by měl dobrý informační systém odpovědět (musíme ovšem požadovaný jev sledovat - datum narození, pohlaví).
8. **Řadit seznamy**. Máme údaje zaznamenány a budeme určitě požadovat seznam občanů seřazený podle abecedy, či podle věku (a to jak vzestupně tak sestupně), seznamy pro volební okrsky podle ulic abecedně. Seřadit evidenci bude opět jedním z nejčastějších požadavků.
9. **Vypočítávat další údaje**. Pro databázový program bude snadné vypočítat počty obyvatel v městských částech, ulicích. Pro meteorologickou stanici můžeme počítat průměrné denní teploty z evidence měření této meteorologické stanice. Získáváme tedy údaje, které nejsou přímo v databázi uložené. Nad daty můžeme provádět i složitější matematické funkce.
10. **Tvorba aplikační nadstavby**.
    a. **Formuláře**. Představte si situaci, kdy přecházíte z ruční evidence na počítačovou. Pracovnice, které mají zadávat data, se budou zdráhat s počítačem pracovat. Abychom jim přechod podstatně ulehčili, je velmi praktické vytvořit v počítači formuláře (například osobní karty evidence obyvatelstva), které budou k nerozeznání od papírových. Na obrazovce budeme mít formuláře, kde jednotlivá políčka budou na stejném místě jako na papíře a mohou být stejně zvýrazněna (barvou, podtržením). Formuláře se spíše vytváří v aplikačním programu (dnes je to často i webová stránka).
    b. **Tiskové sestavy**. Pochopitelně vše, co z informačního systému dostaneme (seřazený seznam, odpovědi na dotazy, osobní karty - formuláře), požadujeme snadno vytisknout. Dokonce můžeme vytvářet další a další zajímavé sestavy včetně např. skupinových součtů či procentuální počty věkových kategorií v jednotlivých městských částech. Tyto skupinové součty a průměry nikam nezadáváme ty si program vypočítá na základě našeho požadavku z osobních údajů jednotlivých občanů.
    c. **Export / import**. Při fungování systému se často jistě stane, že některé údaje již budou existovat, ale v jiném formátu - evidence bude vytvořena jiným programem. Jestliže se jedná některý ze standardů pro atributová data, pak ji do systému snadno převezmeme. A naopak, ze systému by měla být možnost snadno předat naše údaje do jiných systémů. Dnes se je k dispozici např. formát XML, který je pro výměnu dat určený. A nemusí se jednat pouze o přenos mezi databázovými systémy, ale i o přenos mezi textovými editory a tabulkovými procesory, statistickými aplikacemi, geografickými informačními systémy.

Tak toto jsou základní dovednosti informačního systému, který pracuje s databázovým systémem. Výše uvedené operace by měly být k dispozici běžnému uživateli, například pracovnici zadávající údaje. Pokud je na našem pracovišti počítačová síť, je rozumné, aby tato data byla k dispozici dalším vybraným uživatelům, avšak jen v režimu prohlížení. Zaznamenávat a měnit údaje např. o bydlišti a však má jen odpovědná referentka. Hovoříme o **zabezpečení databáze** a **nastavení práv** jednotlivých uživatelů. Přístupová práva můžeme členit i na několik požadovaných úrovní. Základní úrovně přístupových práv jsou čtení a modifikace dat. Tato práva mohou být udělena nebo odepřena. Práva se mohou vztahovat na celé tabulky nebo přímo na jednotlivé atributy.

O rozsáhlou databázi se již musí někdo pořádně starat - nestačí ji pouze občas zazálohovat. Tím je správce databáze, který kontroluje a testuje integritu dat, přiděluje práva, dále musí provádět pravidelné zálohování (každý den, týden) a archivaci (každý měsíc) a další akce.

1.3. Problémy práce se soubory

Způsob ukládání dat do souborů, který převažoval v sedmdesátých letech, je zastaralý a naráží na řadu problémů:

- Vyhledávání konkrétního záznamu nebo skupiny záznamů je obtížné. Používá se sekvenční zpracování - tj. začíná se na začátku a končí na konci souboru. Pokud bychom chtěli do středu souboru vkládat záznamy nebo ze středu mazat, bylo by to obtížné - museli bychom celý soubor načíst do paměti, upravit ho a pak znovu uložit. Jinak je třeba používat klíčové znaky, pevné délky záznamů, aby bylo možné něco najít. Pokud chcete najít určitý záznam (například lidi žijící v Uničově), musíte projít celý soubor a prohledávat každý záznam.
- Jakmile se soubor zvětší, je práce s ním pomalá.

- Zajištění společného přístupu k souboru je poměrně náročné. Lze sice zamykat soubory, ale při větším počtu uživatelů může dojít k situaci, kdy dlouho čekají na odemčení souboru. Nelze uzamykat na úrovni záznamu.
- Kromě omezení přístupových práv k souborům neexistuje způsob, jak omezit přístup k různým úrovním dat (všichni vidí všechno nebo nikdo nevidí nic).

Jak databáze řeší podobné problémy?

Databázový systém řeší všechny tyto problémy:

- Databáze poskytuje rychlejší přístup k datům než k datům uložených v souborech.
- Uživatel může bez problémů získávat data odpovídající určitým kritériím.
- Databáze mají zabudované mechanismy pro vyřešení současného přístupu k datům. Takže se o to programátor nemusí starat.
- Databáze mají zabudovaný systém přístupových práv.

Dnes je k dispozici celá řada (komerčních i nekomerčních) systémů řízení báze dat, které obsahují většinu funkcí, které od systému pro ukládání dat vyžadujete.

1.4. Modelování reality

Úkolem databáze je popsat část reálného světa. Popsat všechny vlastnosti spojitosti reality bývá dosti obtížné. Reálný svět je složitý a je v něm mnoho souvislostí a proto se budeme snažit jej pouze modelovat, popsat modelem. K části reálného světa vytváříme model, který se soustředí na určitou vybranou množinu objektů. Jen tak rozumně postihneme podstatné vlastnosti.

Při modelování musíme projít několika úrovněmi, které se liší podle míry abstrakce.

Tu nejvyšší úroveň tvoří **reálný svět** a z něj takové typy objektů a údajů o objektech, které souvisí se skutečnostmi, jež chceme zahrnout do informačního systému. Tuto úroveň má popisovat zadavatel aplikační úlohy. Vybrat podstatné a dostatečné informace pro informační systém zdaleka nebývá jednoduché a často zadavatel spolupracuje s analytikem IS.

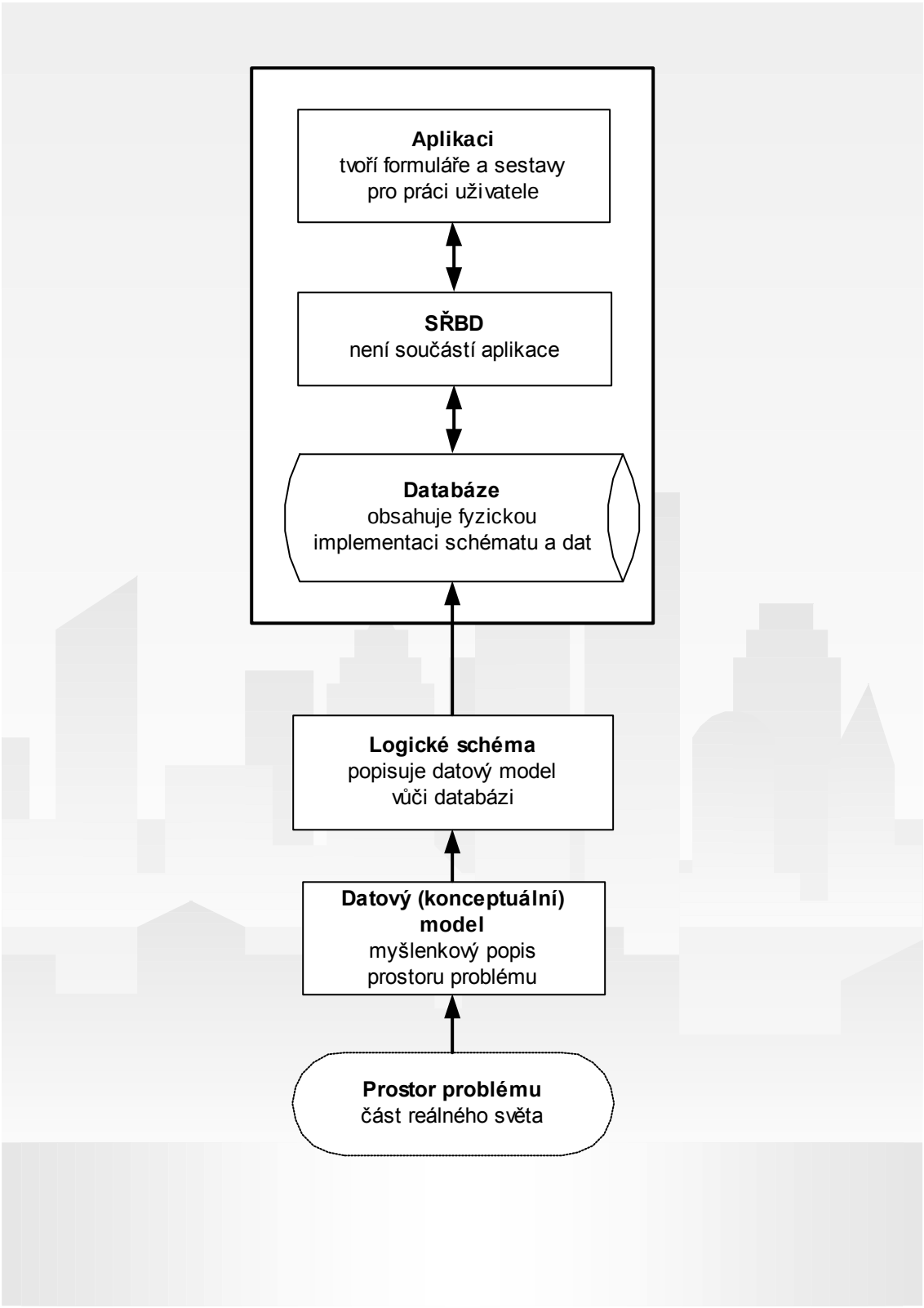
Analytik pak provádí datovou a funkční analýzu.

IS je používán obvykle řadou uživatelů, přičemž každý z nich „vidí“ jen část celé databáze. Pohledy jednotlivých uživatelů na databázi nazýváme **externí schémata**. Při zadání se obvykle vychází z požadavků jednotlivých uživatelů, tedy z externích schémat. Datový analytik musí provést integraci všech požadavků tak, aby překrývající se požadavky nevedly k násobnému výskytu entit a atributů.

Datová analýza je proces poznávání objektů reálného světa, jejich vlastností a vazeb: vytipování, které jsou potřebné pro zamýšlený informační systém, jakými entitami a atributy budou objekty popsány. Výsledkem datové analýzy (po integraci požadavků z externích schémat) je informační struktura zvaná **konceptuální** (myšlenkové) **schéma** databáze. Definují se entity (např. obyvatel města) a jejich atributy (příjmení, jméno, pohlaví, adresa).

Proces, který na základě požadavků na databázový systém plynoucích z reality definuje strukturu databáze, její rozdělení na externí schémata, se často nazývá **konceptuálním modelováním.** Jazyk použitý pro popis konceptuálního schématu je formalizovaný (používají se různé grafické prostředky viz. kapitola 2 Konceptuální modelování databází pomocí diagramů). Tyto prostředky bývají srozumitelné i pro zadavatele a budoucí uživatele.

Analýza chování objektů v reálném světě se nazývá **funkční analýza.** Popisuje jednotlivé akce prováděné nad objekty reálného světa, které jsou zaznamenány v konceptuálním schématu databáze. Obě analýzy vedou k různým popisům, obě však spolu úzce souvisejí. Výsledkem funkční analýzy je pojmenování a popis akcí, které se nad datovými strukturami provádějí.



Obr. 1 Postup a realizace datového modelování

Obě uvedená schémata jsou logickým popisem struktury a chování báze. Zatím není podstatné, jak bude báze implementována, jaký bude zvolen SŘBD a operační systém.

Datový model je druhou úrovní po nejvyšší úrovni, kterou byl reálný svět. Další úroveň tvoří **implementační vrstva.** Rozhodnutí o vlastní implementaci báze souvisí s výběrem použitého SŘBD a realizací popsaných datových struktur, jejich vazeb a akcí nad nimi prováděných. Tentýž konceptuální model je možno realizovat v různých SŘBD a jim odpovídajících datových modelech.

Realizací konceptuálního schématu v konkrétním SŘBD je **logické schéma**, tedy definice datových struktur a jejich vazeb pomocí prostředků použitého SŘBD.

Realizací externích schémat jsou data viděná jednotlivými **aplikačními programy** nad databázovým schématem, které provádějí akce jednotlivých uživatelů.

Interní schéma databáze popisuje nejnižší fyzickou úroveň uložení dat na médiu počítače. Definuje fyzické záznamy, fyzickou reprezentaci jejich položek, sdružování záznamů do souborů, charakteristiky těchto souborů. Souvisí bezprostředně s použitým SŘBD.

Popsané víceúrovňové schéma vývoje databáze odpovídá svou logikou životnímu cyklu vývoje informačního systému. Popis reality odpovídá zadání. Analýza externích schémat a jejich integrace do konceptuálního schématu odpovídá datové či objektové analýze, analýza požadavků na chování systému funkční analýze. Převod logického modelu do databázového odpovídá datovému návrhu. Realizace aplikačních úloh pak je implementací.

1.5. Terminologie relačních databází

Již dříve v textu byly použity odborné termíny z oblasti relačních databázových systémů. Proveďme jejich definovaní a vysvětlení.

Výskyt entity (instance) je jednoznačně rozlišitelný, identifikovatelný a samostatné existence schopný objekt z reálného světa. Entitou je například osoba Oldřich Šmíd s osobním číslem Z103. Entitou může být dále například řeka, stát, parcela, dům, měření fyzikálních veličin, výpůjčka, faktura, objednávka, bankovní účet atd.

Typ entity je skupina entit stejných atributů.

Atribut je funkce přiřazující entitám nebo vztahům hodnoty popisného typu. Popisným typem rozumíme jednoduchý datový typ, například reálné číslo 278.1, textový řetězec „Lípa srdčitá“, „trvalý travní porost“.

V atributech jsou obsaženy význačné vlastnosti entity či vztahu. Atribut může nabývat hodnot pouze určitého datového typu (z určitého oboru hodnot), např. celé číslo.

Typy – třídy entit si můžeme v relačním databázovém modelu představit jako **tabulky**, které jsou nazývány dle autora relačního modelu E. F. Codda **relace.**

V záhlaví sloupců tabulek bývá zvykem zobrazovat označení **atributu**. Často nazývané též pole.

Každý řádek představuje jeden **záznam** (record).

Do jedné tabulky ukládáme stejný typ entit.

tOsoba

OsobniCislo	Prijmeni	Jmeno	DatumNar	Vyska	RidPruk	RodneCislo
Z101	Škodová	Václava	10.8.1983	162	Ano	8358101841
Z102	Janků	Libuše	24.7.1982	164	Ne	8257242136
Z103	Šmíd	Oldřich	12.12.1959	170	Ano	5912124717

Tab 1 Relace tOsoba

Někdy se musíme podle požadavku na výslednou databázi rozhodnout, co bude entitou a atributem. Situace nemusí být jednoznačná. Může nastat i tento případ. Jednou je popelnice atributem u entity dům a ve druhém případě je popelnice entitou a dům určený adresou je atributem této entity. Rozhodnutí o tom, co a jak modelovat závisí na analytikovi, který navrhuje model databáze.

tDum

ID_Dum	Ulice	CisloPopisne	Popelnice_kusu
269	Vídeňská	17	2

tPopelnice

CisloPopelnice	Ulice	CisloPopisne
11	Vídeňská	17
12	Vídeňská	17

Tab 2 Rozdílné modelování entit a atributů

Doporučení a konvence:

Při pojmenovávání relací, atributů a dalších objektů je dobré dodržovat určité konvence a doporučení z důvodu srozumitelnosti a přenositelnosti. Všechny objekty musí být pojmenovány. Pro názvy se doporučují následující konvence.

Název atributu a **tabulky** volíme co nejvýstižnější. Doporučuje se, zvláště v návaznosti na případné programování či jiné operace, aby název atributu byl pokud možno co nejkratší. Je vhodné držet se limitu do 25 znaků vzhledem k převodu pod jiný databázový systém. Nejraději volíme názvy bez mezer, místo mezer použijeme podtržítka. Dále není vhodné používat znaky s diakritikou, i když to systém umožňuje. Zakázány jsou znaky !, ., [,], název nesmí začínat mezerou, %,?.

Příklad: Pro atribut "datum narození" jsou vhodné varianty Dat_nar, DatNar, DatumN, datum_narozeni a podobně.

Názvy objektů musí být v rámci jedné databáze jedinečné. Nemůžou existovat dvě tabulky stejného jména. Existují výjimky z tohoto pravidla, kdy název indexu může být shodný s názvem již existující tabulky. Výjimkou jsou názvy atributů v různých tabulkách. Zde bývá žádoucí použít stejného názvu v případě, že se jedná o cizí klíč.

Název objektu nemůže být shodný s klíčovým slovem jazyka SQL nebo jiným rezervovaným slovem používaného databázového prostředí.

Další konvencí je **nazývání tabulek**. Doporučuje se nazývat tabulky v jednotném čísle nikoliv množném. Tabulku všech měst nazvat mesto, nikoliv města , tabulku parcel nazvat parcela apod.

Často se v názvu tabulky objevuje jako první písmeno **t**, v názvu dotazu **d**. Rozlišíme takto objekt tabulka od objektu dotaz. Např. tOsoba je tabulka všech osob, dOsoba je dotaz na určitou hledanou osobu. V anglickém prostředí může být použito q jako query. V případě číselníků se dá použít **c**, např. cPSC.

1.6. Datové typy

U každého atributu musíme určit jeho **datový typ** (**obor hodnot**, relační teorie databází hovoří o **doméně**) a tím v podstatě naznačíme, které údaje budeme do něho zapisovat.

Norma ISO 9075 z roku 1992 pro jazyk SQL 92 definuje datové typy těchto klíčových slov: CHARACTER, CHARACTER VARYING, BIT, BIT VARYING, NUMERIC, DECIMAL, INTEGER, SMALLINT, FLOAT, REAL, DOUBLE PRECISION, DATE, TIME, TIMESTAMP, a INTERVAL.
Následující ISO norma z roku 1999 přidává předefinované datové typy CHARACTER LARGE OBJECT, BINARY LARGE OBJECT a BOOLEAN.

Databázové systémy se liší počtem a druhy datových typů. Neshodují se plně s ISO normou pro jazyk SQL. Nejběžnějšími základními datovými typy tedy jsou:

- **textový** (v systémech může být označen jako text, character, string, varchar apod.),
- **číselný** (numeric),
- **datumový** (date)
- **logický** (logical, boolean, YesNo, ano/ne).

Databázové systémy pak podporují různé další typy, které však už nemusí být obecně přenositelné ze systému do systému.

Datové typy se liší nejen tím, co popisují, ale i způsobem uložení v systému.

Pro grafická data, která jsou zvláštním druhem dat, se používají různé specifické datové typy. Nejprve se využívalo uložení do datových typů velkých rozměrů, příkladem může být datový typ **BLOB** (binary large object), např. v systémech MS SQL nebo ORACLE. Až databázové systémy posledních let řeší uložení těchto dat tak, aby kromě uložení bylo možné i vyhledávat v těchto datech i například topologické vztahy.

Pro názornost si uveďme varianty jednotlivých datových typů v MS Access. Typy jsou následující:

- **Text** – tento typ je nejobecnější a jeho obsahem může být libovolný řetězec znaků v maximální délce 255. To znamená, že konkrétní hodnota může obsahovat i mezery, speciální znaky, číslice, malá a velká písmena, interpunkční znaménka a podobně. Typickým představitelem je pole obsahující název města, jméno a příjmení, adresu bydliště, ale i číslo telefonu, a pokud hodláme uvádět, i směrovací číslo. Napří-

klad pro atribut Prijmeni bude většinou postačovat délka 20 znaků (příjmení v naších podmínkách nebývá delší než 20 znaků).

- **Memo** - nejčastěji se tento typ používá pro zaevidování poznámek, tedy těch údajů, které se vyskytují spíše sporadicky (pouze u některých vět) a nelze je v podstatě zařadit do jiného pole, protože jsou velice různorodé a mají velice rozdílnou délku. Například u evidence stromů v arboretu by bylo možné pomocí tohoto typu pole vpisovat poznámky: Rozdvojený kmen, Napaden jmelím, Obrostlý popínavkou do výšky 2,5m. Jiný příklad je pro nivelační bod v evidenci bodů České státní nivelační sítě. Poznámka může obsahovat: Dříve bod pořadu 568 DRHN Olomouc-Svitavy(1941), Nelze postavit svisle 3m lať, Výška nejistá (pokles 5 až 60 mm) atd.

  Poznamenejme ještě, že textový řetězec může mít maximálně 65 535 libovolných znaků. Tento datový typ se nevyskytuje běžně u jiných databázových systémů.

- **Číslo** - Tímto typem určíme, že obsahem pole bude právě jedno číslo. Navíc můžeme s čísly v tomto poli za celou tabulku (či její vybranou část) provádět matematické operace - sčítat, vypočítávat průměry, zjišťovat maximální či minimální hodnotu a případně jejich rozdíl či zjišťovat například počet kladných čísel. Lze zvolit následující "podtyp":

  ◊ **bajt** - do tohoto pole lze zadat pouze celá čísla od 0 do 255, velikost při uložení 1 byte

  ◊ **celé číslo** - pole bude akceptovat pouze celá čísla, a to od -32 768 do 32 767, velikost 2 byty

  ◊ **dlouhé celé číslo** – lze zadat celé číslo, a to od –2 147 483 648 do 2 147 483 647, velikost 4 byty

  ◊ **jednoduchá přesnost** - desetinné číslo v rozmezí čísla od –3,402823 E38 do –1,401298 E–45 pro záporné hodnoty a od 1,401298 E–45 do 3,402823 E38 pro kladné hodnoty, počet platných číslic je 7, velikost při uložení jsou 4 byty

  ◊ **dvojitá přesnost** – číslo v rozmezí –1,79769313486231 E308 do –4,94065645841247 E–324 pro záporné hodnoty a od 4,94065645841247 E–324 do 1,79769313486231 E308 pro kladné hodnoty, počet platných číslic je 15, velikost při uložení je 8 bytů

  ◊ **desetinné číslo** - číslo od –1 E28–1 do 1E28–1, počet platných číslic je 28, velikost při uložení je 12 bytů

  Používejte co nejnižší rozsah čísla, neboť data o menší velikosti lze rychleji zpracovávat a vyžadují méně paměti.

- **Datum/čas** - tento typ je vhodný například pro sledování data narození, data pořízení vkladu do katastru nemovitostí, času odběru vzorku a podobně. Poznamenejme, že Access dokáže velice jednoduše u polí tohoto typu například přičíst k datu jeden týden, měsíc či rok, případně ze zadaného data "vyjmout" pouze měsíc, a tudíž vyhledat z databáze všechna měření, která se prováděla například v měsíci březnu (na roce nezáleží).

- **Měna** - Peněžní částky a číselná data používaná v matematických výpočtech, ve kterých se vyskytují data s přesností od jednoho do čtyř desetinných míst. Maximální přesnost je 15 řádů před desetinnou čárkou a 4 řády za desetinnou čárkou, velikost při uložení je 8 bytů.

- **Automatické číslo –** je ekvivalentem datového typu dlouhé celé číslo. Toto je zvláštní typ, specifický pro Access. Access bude tuto položku u každého záznamu vyplňovat sám. Generuje se buď vzestupná posloupnost přirozených čísel (začne od jedničky) nebo se přiřazují jednotlivým záznamům čísla náhodně.

- **Ano/Ne -** alternativní položka, která může obsahovat pouze dvě varianty ano/ne, zapnuto/vypnuto, pravda/nepravda. V tabulce může být znázorněna zaškrtávacím políčkem. Jistě lze nalézt mnoho případů, kdy je vhodné použít tento datový typ - například u položek muž/žena, má/nemá přístup k editaci dat, řidičský průkaz má/nemá, smlouva je platná/neplatná, město má/nemá park.

- **Objekt OLE -** objekt, například list aplikace Microsoft Excel, dokument aplikace Microsoft Word, obrázek, zvuk nebo jiný typ binárních dat, který je propojený s tabulkou aplikace Microsoft Access nebo vložený do tabulky. Velikosti je omezena na 1 MB. Tak například evidence budov může díky tomuto typu obsahovat naskenované fotky. Mohou se zde uložit naskenované podpisové vzory k účtu u bankovního ústavu. Seznam GIS softwaru můžeme zpestřit jejich logy.

- **Výčet –** nelze považovat za samostatný datový typ, ale jako datový typ číslo nebo text s určitým výčtem možných hodnot, který zadám volbou Průvodce vyhledáváním. Výčet je stanoven pouze seznamem nebo

samostatnou zdrojovou tabulkou. Příkladem je klasifikace ploch podle zemědělského využití: orná půda, zahrada, ovocný sad, vinice, chmelnice, trvalý travní porost.

1.7. Klíče

Dříve v definici entity bylo řečeno, že entitou je jednoznačně identifikovatelný objekt reálného světa. Jeden řádek tabulky obsahuje jeden záznam jednoho výskytu entity. V každé tabulce (relaci) musí být každý záznam jedinečný. Má-li být splněna podmínka jedinečnosti, musí u každé relace existovat jeden nebo skupina atributů, která jednoznačně identifikuje jednotlivý záznam. Tento atribut nebo více atributů, který by mohl jednoznačně identifikovat záznam, se nazývá ***kandidátní klíč***. Příklad u osob rodné číslo, číslo občanského průkazu, evidenční číslo osoby. Dále ISBN u knihy, SPZ u osobního auta. Výběr kandidátních klíčů je třeba velice dobře zvážit. Číslo občanského průkazu se v průběhu života mění, rodné číslo nemají cizinci žijící u nás atd.

Tabulka může mít více kandidátních klíčů.

Pokud žádný z atributů není kandidátním klíčem zavedeme dodatečný atribut, který bude jednoznačně identifikovat záznam. Např. kód čtenáře, kód studenta apod. Tento dodatečný atribut nemá žádnou odpovídající reprezentaci v reálném světe, je umělý. U MS Access můžeme využít pro tento atribut automatické číslo. Access při tvorbě nové tabulky nabízí přímo vygenerování primárního klíče. Vloží nové pole s názvem ID a datovým typem automatické číslo.

Jednoduchý klíč je klíč složený z jediného atributu.

Složený klíč je složený z více atributů. Délku složeného klíče musíme také pečlivě zvážit. Složený klíč musí být *minimální* v tom smyslu, že nelze odebrat žádný atribut, aniž by to narušovalo identifikační vlastnost.

Klíč, který vybereme z kandidátních klíčů, se stává ***primárním klíčem***.

Primární klíč nesmí mít hodnotu NULL! Hodnota NULL znamená, že atribut je neznámý nebo neexistující. V případě, že více záznamů by mělo pro primární klíč hodnotu NULL, pak by již neplnil funkci jednoznačné identifikace záznamu. Z tohoto důvodu je to pochopitelný požadavek. Pro jiné atributy může být tato hodnota použita. Často se tak děje u údajů, které mají být doplněny časem, nebo nejsou tak podstatné.

Cizí klíč je atribut v tabulce, který přidáme do tabulky z jiné tabulky, ve které je tento atribut primárním klíčem. Tento atribut již může vykazovat duplicity. Může se zde objevit i hodnota NULL (u nepovinného zapojení).

tOsoba

OsobniCislo	Prijmeni	Jmeno	DatumNar	Vyska	RidPruk	RodneCislo
Z101	Škodová	Václava	10.8.1983	162	Ano	8358101841
Z102	Janků	Libuše	24.7.1982	164	Ne	8257242136
Z103	Šmíd	Oldřich	12.12.1959	170	Ano	5912124717

primární klíč této relace

tDite

ID_dite	OsobniCislo	Prijmeni	Jmeno	RodneCislo	Vaha
1	Z101	Škoda	Petr	9601211458	35
2	Z101	Škodová	Andrea	9851174569	27
3	Z103	Šmídová	Julie	9753016789	34


primární klíč této relace cizí klíč (primární klíč relace tOsoba)

Tab 3 Atributy, které jsou primárním a cizím klíčem v relaci tOsoba a tDite

Atribut OsobniCislo je primárním klíčem v tabulce tOsoba a v tabulce tDite je cizím klíčem. Vidíme, že osoba Škodová (OsobníCislo Z101) má dvě děti - Petra a Andreu.

1.8. Vztahy

Vztah je vazba mezi dvěma nebo více typy entit. Například osoba Oldřich Šmíd, osobní číslo Z103, může být ve vztahu „má-dítě“ k Šmídové Julii, identifikační číslo ID_dite 3. Vztah je nejčastěji realizován primárním klíčem v jedné tabulce a cizím klíčem v tabulce druhé. Pozor oba tyto atributy musí být **typově kompatibilní.** To znamená, že musí být ze stejného oboru hodnot (domény). Pokud je primární klíč text s 28 znaky, musí být cizí klíč též text s 28 znaky, nikoliv text s 11 znaky. Nebo pokud je primární klíč typu dlouhé celé číslo, taktéž cizí klíč musí být dlouhé celé číslo, nikoliv číslo s dvojitou přesností nebo dokonce text.

Vazba mezi dvěma nebo více entitami je vyjádřena pomocí vztahu (relace). V reálném světě registrujeme velké množství vztahů, které musí být v datovém modelu nějakým způsobem vyjádřitelné. Poměr prvků ve vztahu může být 1:1, l:N anebo M:N. Mluvíme o stupni či kardinalitě vztahu.

Vztah 1:1

Typickým příkladem je vztah 1 manžel : 1 manželka. Dalším příkladem je název kraje : název krajského města, 1 polygon : 1 parcela, atd. Vztah 1:1 může zahrnovat i vztah 1:0 a 0:1. Někdo např. nemá manželku.

Je třeba zvážit, zda zakládat pro takovéto entity samostatné tabulky. Lze je velice jednoduše sloučit do jedné tabulky. Ale může existovat i důvod, proč je chceme modelovat samostatnými tabulkami.

Vztah 1:N

Vztahový typ 1:N (nebo též 1 : nekonečnu) může teoreticky zahrnovat i vztahy 1:0, 0:1 a 1:1, které mohou být v případě potřeby z modelu vyloučeny vhodně zvolenými pravidly.

Příklady jsou: jedna budova má více pater, do jednoho úmoří spadá více řek, město má více městských částí, jeden stát má více letišť.

Vztah M:N

Vztah M:N je nejsložitější vztah. Zahrnuje i vztahy 1:0, 0:1, 1:1, 1:N, M:1.

Příkladem může být situace, kdy jedna parcela má více vlastníků a naopak, když jeden vlastník vlastní více různých parcel. Dále například jeden stát může mít více úředních jazyků a naopak úřední jazyky v některých státech mohou být shodné (angličtina, francouzština, němčina).

Vztah M:N je v relačním datovém modelu nutné **dekomponovat** (převést) na dva vztahy 1:N. Důvodem je, že je v relačních databázových systémech nelze přímo realizovat. Ukažme si to na příkladu. Máme entitu *stát* a entitu *úřední jazyk*.

tStat

ID_stat	Stat
101	Belgie
102	Lucembursko
103	Francie

tJazyk

ID_jazyk	UredJazyk
1	francouzština
2	nizozemština
3	němčina
4	lucemburština

Tab 4 Výchozí entity tStat a tJazyk

Musí se provést dekompozice vztahu stát - úřední jazyk takto: Přidá se další – vazební tabulka tVazba.

tStat

ID_stat	Stat
101	Belgie
102	Lucembursko
103	Francie

tVazba

ID_stat	ID_jazyk
101	1
101	2
101	3
102	1
102	3
102	4

tJazyk

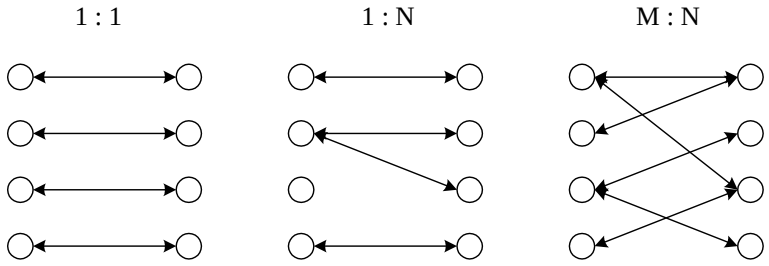
ID_jazyk	UredJazyk
1	francouština
2	nizozemština
3	němčina
4	lucemburština

103	1

Tab 5 Výsledek dekompozice vazby N:M

Tabulka tVazba má dva cizí klíče, primárním klíčem je složený klíč ID_stat+ID_jazyk. Tabulka tVazba je vazební (spojovací) tabulkou.

Graficky můžeme výše uvedené vztahy znázornit takto:



Obr. 2 Grafické znázornění druhů vazeb

Rekurzivní typ vztahu

Velmi častou je potřeba vyjádření typu vztahu, jehož účastníci jsou **entity stejného typu**. Tuto vazbu označíme jako rekurzivní nebo je někdy označována jako *self vazba.*

Například: Jedna OSOBA může vést několik OSOB (je vedoucím pracovníkem) a zároveň jedna OSOBA musí být vedena právě jednou OSOBOU.

Nebo VÝROBEK může být součástí VÝROBKŮ a zároveň VÝROBEK může být sestaven z daných VÝROBKŮ.

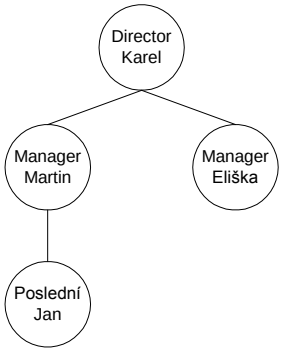
V následující tabulce je evidence osob, osoba je určena atributem CisloOsoby. Atribut JeVedenaOsobouCislo nám realizuje vazbu na CisloOsoby. Je zde vazba 1:N

tOsoba

CisloOsoby	Prijmeni	Jmeno	JeVedenaOsobouCislo
8	Director	Jan	Null
16	Manager	Martin	8
24	Manager	Eliška	8
32	Poslední	Jan	24

Tab 6 Relace s rekurzivní vazbou

Je-li kardinalita vztahu 1:N, můžeme říci, že tento typ vztahu zavádí do množiny výskytů tohoto typu stromovou *hierarchii*. Na vrcholu je jeden zaměstnanec, ten má několik podřízených na první úrovni hierarchie, ti zase mají každý své podřízené, atd. Hloubka hierarchie není předem omezena.



Obr. 3 Hierarchická struktura osob, která vyjadřuje jejich nadřízenost a podřízenost

Rekurzivní typ vztahu může být i stupně M:N. V tomto případě již nestačí jedna tabulka, jak byla uvedena výše, ale opět musí být vytvořena ještě vazební tabulka. Uvedeme to na příkladu výrobků.

tVyrobek

ID_Vyrobku	Vyrobek
1	výrobek1
2	výrobek2
3	výrobek3
4	výrobek4

tRozklad

IDVyrobku Nadrazeneho	IDVyrobkuPodrizeneho
1	2
2	3
2	4
3	4

Tab 7 Rekurzivní vazba M:N

Poznámka:

V češtině je vhodnější používat slovo vztah. V anglickém originále mu odpovídá slovo relation, nebo lépe relationship. Někdy i česká literatura a programy používají pro vztah označení relace a dokonce dochází k mylnému názoru, že odsud plyne název relačních databází. Relace je množina pro uložení entit představovaná tabulkou.

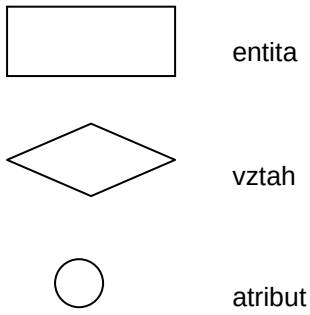
2. Konceptuální modelování databází pomocí diagramů

2.1. E-R diagramy

Při modelování databází napomáhá velice grafické vyjádření pomocí diagramů, neboť je srozumitelné a názorné. Metodu zobrazení do diagramů zavedl poprvé Peter Pin-Shan Chen v roce 1976. Tyto diagramy nazval E-R diagramy (modely). E-R modely jsou založeny na pojmu entita (Entity) a vztah (Relationship). Z počátečních písmen slov vyplývá i jejich název.

E-R model se vyskytuje v mnoha variantách. Ke dvěma základním konstrukcím často bývá přiřazován ještě třetí - atribut (Attribute), potom se setkáváme s označením ERA diagram. E-R model je jedním z mnoha modelů, které jsou pro konceptuální modelování k dispozici.

Pro reprezentaci grafického návrhu E-R modelu se používají tyto symboly. **Entita** se popisuje pomocí obdélníků, **vztahy** se znázorňují pomocí kosočtverců a **atributy** pomocí kružnic, tak jak vidíte na následujícím obrázku:

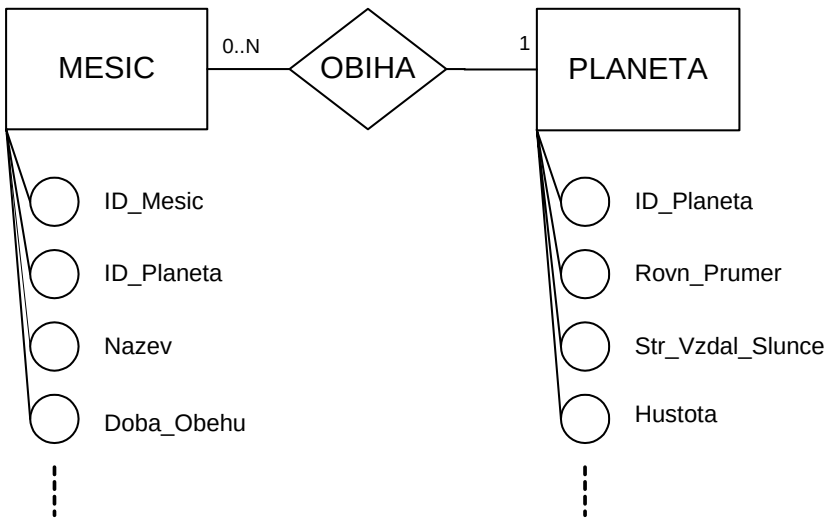


Obr 4 Grafické symboly ERA diagramů

Příklad: Zakresleme si ERA diagram pro databázi vesmírných objektů naší Sluneční soustavy. Navrhneme dvě entity PLANETA a MĚSÍC. Vztah je OBÍHÁ.

Entita bývá z hlediska kategorie českého jazyka podstatné jméno a vztah bývá reprezentován slovesem. Vymezení pojmu entita, vztah a atribut je dosti volné. Neexistuje jednoznačné pravidlo jak klasifikovat data jako entitu nebo jako vztah. Často závisí na úhlu pohledu analytika.

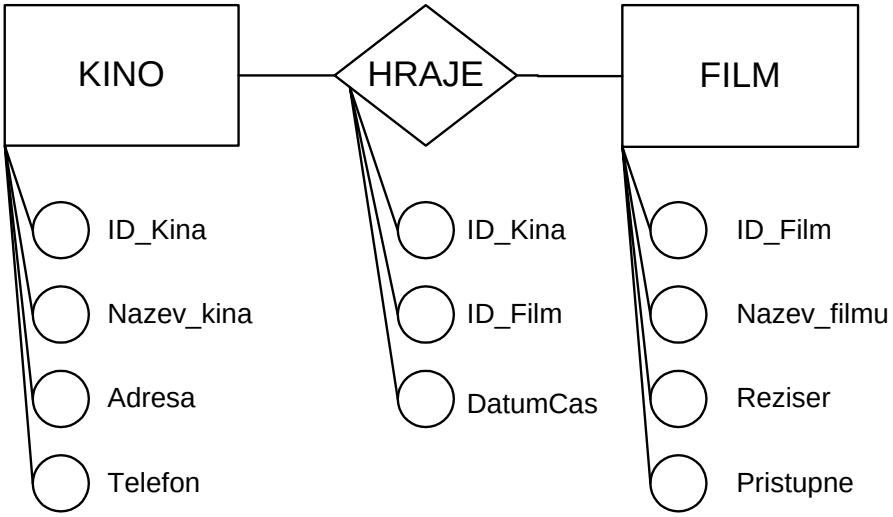
Planeta i měsíc mají řadu atributů: název, rovníkový poloměr, dobu oběhu, hustotu atd. V diagramu nejsou všechny vypsány.



Obr. 5 ER diagram pro databázi Sluneční soustavy

Vztah nemá v tomto případě žádné atributy.

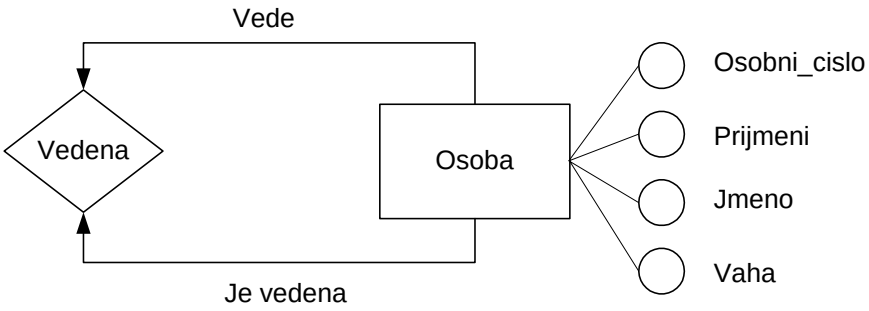
Následující model představuje databázi kin a filmů, které se promítají. Entita FILM má své atributy jako jsou název filmu, rok natočení, režisér, země atd. Entita KINO má atributy název kina, adresa kina, telefon. Vztah HRAJE již má také atributy. Jsou to údaje: ve kterém kině se film hraje, jaký film se hraje a časový údaj kdy (datum) a v kolik hodin se hraje.



Obr. 6 ER diagram kin a filmů

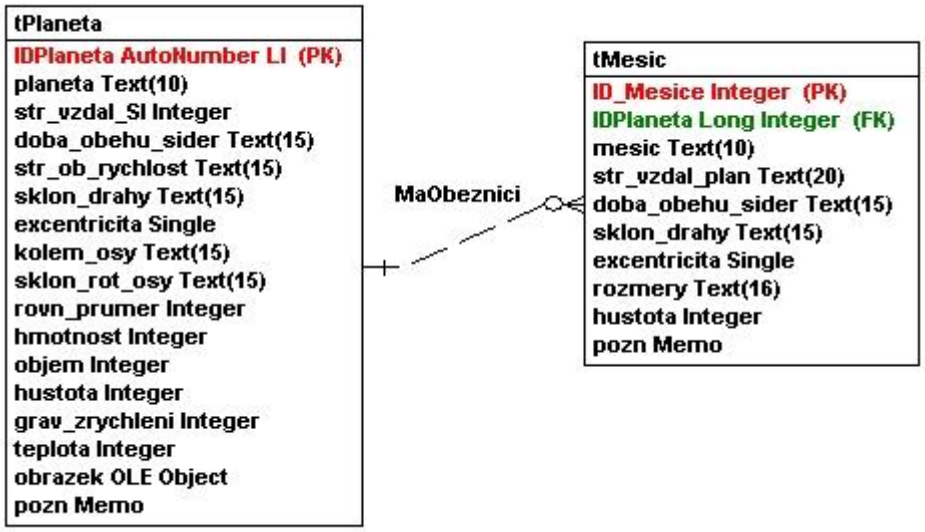
V datovém modelu je pak vazba HRAJE realizována samostatnou entitou – tabulkou s atributy. Primárním klíčem entity HRAJE je složený klíč ze tři atributů ID_Kina + ID_Film + DatumCas (PFK – primary foreign key).

ER modelem můžeme zakreslit i **rekurzivní vztah**. Je uveden na následujícím obrázku pro příklad nadřízených a podřízených zaměstnanců. V souvislostí s rekurzivním typem vztahu je užitečné zavést do E-R modelu ještě jeden konstrukt. Říká se mu **role.** Jmény rolí se ohodnocují jednotlivé hrany E-R schématu, které vedou od entity k vztahu. Označují roli, jakou účastnické entity hrají v daném typu vztahu. V našem personálním příkladu bude horní hrana ohodnocena rolí **Vede** a dolní hrana rolí **Je vedena**.



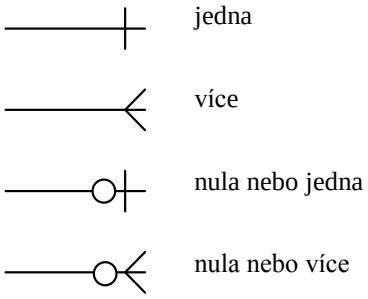
Obr. 7 ER diagram rekurzivní vazby

Způsob zákresu ER diagramů se vyvíjel a dnes se nejčastěji používá v následujícím provedení. Do obdélníků jsou zakresleny entity, v záhlaví je jejich název. V spodní části obdélníku je výčet atributů. Primární, cizí klíče jsou zvýrazněny (tučným písmem, barevně).



Obr. 8 Diagram entit a vztahů

Vysvětlíme si varianty zákresu grafických symbolů pro vztahy. Jsou následující:



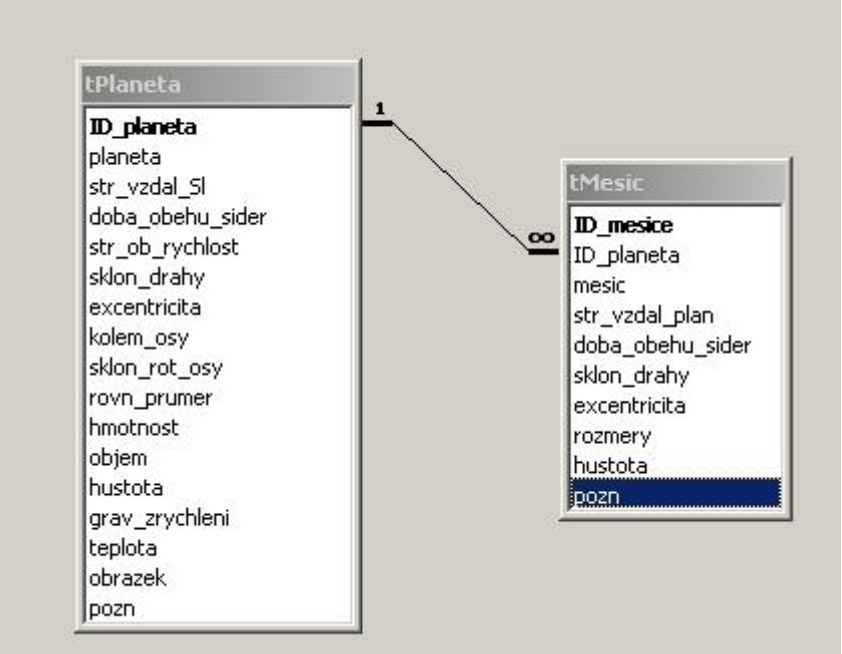
Obr. 9 Symbolika stupňů vztahů

Velkou výhodou těchto symbolů je jejich srozumitelnost. Použití vidíme např. u produktu CASE Studio 2 od firmy CharonWare, který je určen pro modelování databází. Opět jde o ukázku databáze Sluneční soustavy. Tyto grafické symboly mají lepší vypovídající schopnost, neboť opravdu vidíme, že jde o vazbu 1:N (s možností 1:0) a může tedy nastat případ, kdy planeta nemá žádný měsíc (Merkur, Venuše). Jinak mají planety jeden nebo více měsíců.

Dekomponovaná vazba M:N, která byla diskutována dříve v textu je zakreslena takto:

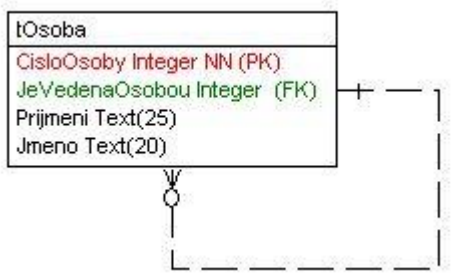
Obr. 10 Diagram dekomponované vazby M:N pro informaci o státě a jeho úředním jazyku

MS Access pro stupeň vztahů mezi entitami používá v okně relací (správněji vztahů) tuto symboliku. Nad spojnou čáru se píší symboly 1 a nekonečno ∞.



Obr. 11 Okno relací – vztahů mezi tabulkami v MS Access

Pro úplnost uveďme způsob zákresu rekurzní vazby:



Obr. 12 ER diagram rekurzivní vazby

Lineární textový zápis

Při modelování databází se používají ještě jiné zkrácené formy zápisu. V lineárním textovém zápisu se zapíše název typu entity a v kulatých závorkách její atributy. Primární klíč se píše tučně nebo se podtrhne. Můžeme psát entity takto:

E: Osoba (**OsobniCislo**, Prijmeni, Jmeno, DatumNar, RodneCislo, Vyska, RidPruk) nebo
Osoba (OsobniCislo, Prijmeni, Jmeno, DatumNar, RodneCislo, Vyska, RidPruk)

a vztahy zapíšeme:

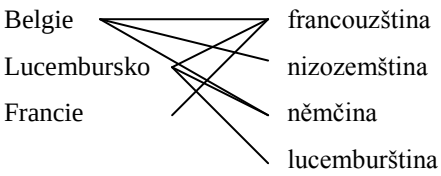
E: Planeta (**ID-planeta**, planeta, rovnikovy_prum, hustota,…)
Mesic (**ID_mesice**,ID-planeta, mesic, …)
V: Obiha (Planeta, Mesic)

Přehlednější je ovšem zápis formou tabulky, kde v záhlaví uvádíme název tabulky a ve sloupci pod sebou atributy. Opět použijeme pro primární klíč tučné písmo nebo podtržení (vhodné při ručním zápisu) nebo barevné zvýraznění.

Stanice
ID_Stanice
NazevStanice
Reka
1PovodnovyStupen
2PovodnovyStupen
3PovodnovyStupen

Výskytový zápis

Pro názornost a vysvětlení se často zobrazují entity a vazby ve formě výskytového diagramu. Tento způsob je vhodné používat jako pomocný při ujasňování stupně vztahu. Například rozeznání vztahu M:N pro stát a úřední jazyky.



Zápis tabulky přímo s ukázkami příkladů záznamů a položek tak, jak už bylo v textu několikrát uvedeno, také lépe objasní možné výskyty při navrhování entit.

Následující tabulka ukazuje tabulku měřících stanic na řekách. Každá stanice má stanoven limit výšky hladiny v milimetrech pro první, druhý a třetí stupeň povodňové aktivity.

ID_Stanice	NazevStanice	Reka	1StupPovodAktiv	2StupPovodAktiv	3StupPovodAktiv
21	Moravičany	Morava	200	250	300
22	Šumperk	Desná	170	220	260
23	Jarcová	Vsetínská Bečva	240	300	350

Tab 8 Ukázka tabulky pro měřící stanice na povodí řeky Moravy a stupňů povodňové aktivity v milimetrech

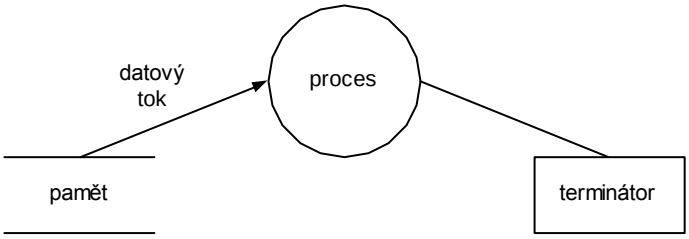
2.2. Funkční analýzy a diagram datových toků

V kapitole 1.4 Modelování reality je zmíněno provádění analýzy chování objektů v reálném světě. Tato analýza se nazývá **funkční analýza**. Výsledkem funkční analýzy je tzv. funkční model.

Diagram datových toků (Date Flow Diagram, DFD) je grafický prostředek pro návrh a zobrazení funkčního modelu systému. Podobně jako ERD je dostatečně jednoduchý a názorný i pro uživatele a může sloužit i k upřesňování zadání.

DFD popisuje dynamiku systému, vyjadřuje transformace dat z jedné formy do druhé; modeluje funkce systému pomocí grafu a přitom používá následujících základních grafických prvků:

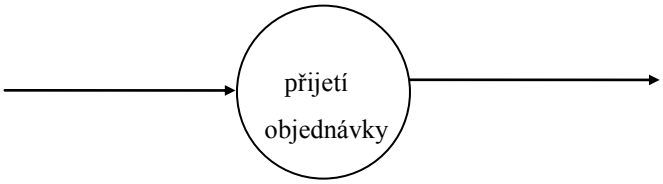
- procesy
- paměti
- terminátory
- datové toky



Obr. 13 Grafické prvky DFD

Proces (transformace, funkce)

provádí transformaci dat vstupních na data výstupní, realizuje nějakou funkci nad daty. Zakresluje se kruhovým uzlem v grafu (někdy elipsou, obdélníkem), v uzlu je zaznamenán název funkce a především její identifikátor (pod kterým lze v dokumentaci nalézt podrobný její popis).

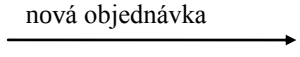


Datový proces vyjadřuje fyzickou transformaci dat, tj. změnu reprezentace dat nebo změnu stavu dat, modifikaci údajů, vznik nových údajů, zrušení údajů; úkolem datového procesu tedy je zpracovávat data.

Každý proces v DFD má svůj název a jednoznačné číslo. Čísla je vhodné přidělovat hierarchicky: součástí čísla je číslo nadřízené funkce, do jejíhož rozkladu popisovaná funkce patří, druhou část tvoří jednoznačné číslo v rámci úrovně rozkladu (nemusí mít žádný vztah k pořadí provádění funkce).

Datový tok

Datový tok vyjadřuje přesun dat nebo informací z jedné části systému do jiné, z okolí systému do systému nebo ze systému do jeho okolí. Znázorňuje se hranou (úsečkou, obloukem) opatřenou šipkou, znamenající směr toku. Je možné použít šipky i oboustranně, když jde o dialog a stejná data tečou oběma směry.

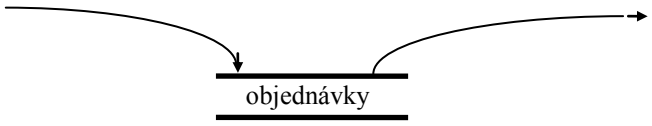


Datový tok musí mít známý obsah a je zase pojmenovaný. Datové toky obsahují data, která jsou do systému vkládána, systémem zpracovávána nebo ze systému vypouštěna. U programových systémů jsou obsahem datových toků zprávy, čísla, znaky, záznamy, bity. Datové toky jsou jednou z forem propojení (komunikace) procesů.

Datová paměť (Data store, zásobník, úložiště)

Paměť je místo dočasného uchování dat pro jejich pozdější využití.

Je to obecnější pojem než soubor. Může být implementován jako pole, soubor textový, soubor databázový, kniha, šanon a leccos jiného. Používá se tam, kde mezi procesy existuje časové zpoždění při předávání dat. Znázorňuje se pomocí dvou rovnoběžek, mezi nimi je název paměti.



Pro každou paměť musí existovat alespoň jeden datový tok směřující do paměti a jeden směřující z paměti. Datový tok může vyjadřovat 1 výskyt dat, více výskytů dat, část jednoho výskytu či část z více výskytů. Paměť je pasivní prvek, data do paměti i z paměti musí vždy procházet přes proces. Paměť je další formou propojení procesů.

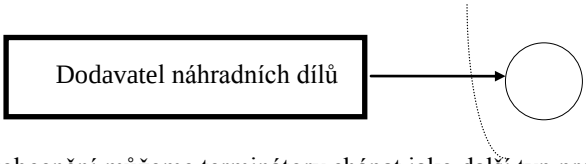
Terminátor

Terminátor znázorňuje externí zdroj nebo cíl dat, objekt vně systému, s nímž systém komunikuje. Může to být člověk, skupina lidí (oddělení), jiný systém ap. Platí pro ně :

- terminátory jsou vně systému, toky mezi nimi a systémem představují rozhraní mezi systémem a vnějším světem
- analytik nemá možnost měnit organizaci a chování entit vně systému ani změnit chování terminátorů

Vztahy mezi terminátory se v DFD nezachycují; mohou sice existovat, ale nejsou součástí navrhovaného systému.

Graficky se znázorňuje obdélníkem či čtvercem, opět má název.



Při mírném zobecnění můžeme terminátory chápat jako další typ propojení procesů.

Hierarchie DFD

Model systému vyjádřený pomocí DFD má hierarchickou strukturu. Jen velmi malý systém je možno vykreslit jediným diagramem. Proto dle podrobnosti rozkladu obvykle rozlišujeme několik úrovní DFD: vrchní (top), střední (middle), spodní (bottom).

Pokud se IS vyvíjí postupem shora dolů, začíná se od přehledového diagramu a pokračuje se ke stále detailnějším diagramům.

Na vrcholu hierarchie je pouze jeden DFD zvaný **kontextový**. Ten obsahuje celý systém jako jednu funkci, definuje hranice systému a všechny terminátory - zdroje systému a cílová místa dat. Systém je zde černá skříňka s definovanými vstupy a výstupy.

Bezprostředním rozkladem kontextového diagramu je DFD **úrovně 0**. Obsahuje základní funkce systému (často rozklad na subsystémy) a jejich vztahy vyjádřené prostřednictvím datových toků a pamětí. Terminátory systému jsou totožné s kontextovým diagramem.

Dále se postupuje v rozkladu funkcí obdobně. Každá funkce, kterou je možno dále rozložit, se rozkresluje novým diagramem nižší úrovně až na elementární úroveň. Nejnižší úroveň obsahuje primitivní uživatelsky dále nedělitelné funkce (stanovení, co je elementární funkce a co dále dělitelná, je věcí analytika). Platí však pro ně, že:

- činnosti se provádějí jako celek
- jsou buď celé ruční nebo automatizované
- jsou opakovatelné
- jsou elementární, nemají další podrobnější DFD.

Pravidla tvorby DFD :

Při číslování procesů se identifikuje jednak úroveň rozkladu, do něhož funkce patří, jednak proces v rámci úrovně (např. 2.4.3)

Názvy procesů má být stručné, výstižně vyjadřovat funkční náplň procesu (např. Vystavení faktury, ne Zpracování dat).

Složitost jednoho DFD má být taková, aby formát nepřesahoval velikost A4, aby neobsahoval velké množství uzlů (někdy se doporučuje jen 6 uzlů, rozhodně počet funkcí v rozmezí 3 - 9) a aby byl srozumitelný pro uživatele, analytika a návrháře

Diagram má být technicky správný (konzistentní), srozumitelný, přehledný a esteticky uspořádaný (bubliny stejně velké, toky se nekříží ap., překreslovat až do grafické dokonalosti).

Konzistence DFD, logická soudržnost diagramu. Ta není samozřejmá, protože jedna skutečnost je díky hierarchickému rozkladu rozkreslena více či méně podrobně na několika DFD.

Pravidla týkající se funkcí

Neznázorňují se žádné inicializační ani závěrečné procedury.

Neznázorňují se cykly mezi funkcemi.

Žádné dvě funkce nesmí mít stejný název.

Pravidla týkající se datových toků

Nesmí existovat proces generující výstupy bez pomoci vstupů (perpetum mobile).

Nesmí existovat proces, který má pouze vstupy a žádné výstupy (černá díra).

Datové paměti smějí být propojeny jen prostřednictvím funkce, tedy datový tok do / z paměti musí vycházet z / do procesu.

Datový tok z / do terminátoru musí procházet přes proces.

Datové toky mezi funkcemi znázorňují pouze přenášená data, nevyjadřují volání jedné funkce druhou ani předávání řízení.

Datový tok s týmž názvem může být v DFD použit na více místech, pokud název znamená skutečně tentýž datový tok se stejným obsahem.

Doporučuje se dodržovat označení datového toku z/do datové paměti:

- bez označení = přenáší se jeden výskyt
- označen datovou pamětí = přenáší se jeden nebo více výskytů
- označen jednoznačně jinak = přenáší se část výskytů.

Pravidla týkající se datových pamětí

Paměti se objeví až na té úrovni, kde jsou viditelné funkce do pamětí zapisující a z pamětí čtoucí.

Vyhledání pro aktualizaci paměti se chápe jako součást zápisu do paměti, nevyznačují se zvláštní šipkou; šipka dovnitř znamená jakékoliv provádění změn (vkládání dat, aktualizaci, rušení).

V paměti jsou uloženy výskyty dat se stejnou strukturou. Jestliže tok z / do paměti přenáší celý výskyt, nemusí se pojmenovávat, je určen obsahem a názvem paměti.

3. Integrita a integritní omezení (IO)

Samozřejmým požadavkem na databázi je, aby obsahovala **správná** a **aktuální data** pro daný okamžik. Hovoříme o integritě databáze. Integritní omezení jsou logická omezení na typy a hodnoty atributů, entit a vazeb tak, aby databáze co nejpřesněji odpovídala zobrazované realitě. Správnost údajů můžeme zajistit několika způsoby na různých místech.

Integrita entit

Integritu entit zajišťujeme požadavkem existence primárního klíče.

Atributová integrita

Atributovou integritu zajišťujeme již na počátku vhodnou *definicí oboru hodnot*. Tu dále můžeme zpřísnit. Například v evidenci studentů čtyřletého studia nedovolíme zadat jiné číslo ročníku (které je z oboru celých čísel) než v rozmezí 1 až 4. Kromě zúžení jednoho určitého atributu může entitové omezení zasahovat také do několika různých atributů. Příkladem takového omezení je například požadavek, podle něhož musí být DatumOdeslani požadovaného výrobku pozdější nebo rovno jako DatumObjednavky tohoto výrobku.

Zadáváme například adresu a to název ulice. U názvu ulice „Fr. Ondříčka“ může dojít často k chybnému zápisu, např. Františka Ondříčka, F. Ondříčka, Ondříčkova vlivem uživatele. Z pohledu databáze se jedná vždy o jinou ulici. Při zjišťování počtu lidí žijících v ulici Fr. Ondříčka dostaneme chybný údaj, neboť ti, kteří mají zapsánu adresu v jiném tvaru, nejsou započítáni. Vstup správných údajů zajistíme tak, že uživateli nabídneme pouze exitující možné názvy ulic pro dané město. Určíme tedy obor hodnot.

Stejně tak u evidence osob v atributu rodinný stav připadají v úvahu pouze čtyři možnosti: svobodný/á, ženatý/vdaná, vdovec/vdova, rozvedený/rozvedená.

Obor hodnot je tedy dán výčtem možností nebo je realizován samostatnou tabulkou, někdy označovanou jako číselníky. Můžeme uvést například číselníky Českého statistického úřadu pro kódování krajů, sčítacích okrsků atd. Existují různé oborové číselníky, či vnitropodnikové.

Při vstupu, nebo editaci údajů je dobré též kontrolovat správnost atributů eventuálně zamezit vstupu chybných údajů. Je vhodná i možnost automatického převodu vstupních údajů. Například u příjmení a jmen automaticky převádět první písmeno na velké nebo celé příjmení na velká písmena. V MS Accessu k tomu slouží nastavování tzv. vstupní masky. Příklady a význam masek pro MS Access naleznete v nápovědě programu.

V jiných složitějších případech je nutné ošetřit vstup a změnu přímo programovým kódem. Tyto „spouště“ jsou v programových produktech nazývány *trigger*.

Vstupní masky

Vstupní maska se používá v textových polích (v tabulkách a dotazech) a polích se seznamem (ve formulářích) k předběžné kontrole vkládaných dat a k formátování. Vstupní maska se skládá ze znakových literálů (mezery, tečky, čárky a uvozovky), které vymezují jednotlivé oddíly pro vkládaná data. Vstupní masky se používají především pro pole typu Text a Datum/Čas, ale lze je použít i pro pole typu Číslo a Měna.

Referenční integrita a její důsledky

Referenční integritu lze nastavit pro tabulky ve vzájemném vztahu 1 : N. Lze ji nastavit též pro vztah 1 : 1, který možno chápat jako zvláštní případ relace 1 : N. Zajišťuje, že se při práci se záznamy (vkládání a odstraňování záznamů) zachovávají definované vztahy mezi tabulkami.

Každému záznamu v tabulce na straně N s hodnotou cizího klíče <> NULL musí odpovídat právě jeden záznam v tabulce na straně 1. Jinými slovy, v tabulce na straně N nepřipouštíme osiřelé záznamy.

Vynutíme-li referenční integritu, SŘBD zabrání

- přidat záznamy do tabulky na straně N, jestliže v první tabulce neexistuje odpovídající záznam,
- změně hodnoty v tabulce na straně 1, která by mohla mít za následek vznik osiřelých záznamů v druhé tabulce,
- odstranění záznamů z tabulky na straně 1, jestliže druhá tabulka obsahuje příslušné související záznamy.

Ukažme si příklad evidence leteckých snímků. Snímkování se provádí na film. Každý film má určitý počet snímků. Tato databáze bude mít dvě tabulky tFilm a tSnimek. V tabulce tFilm jsou popsány vlastnosti filmu jako typ filmu, datum zpracování filmu a další údaje, které se týkají celého filmu. V tabulce tSnimek jsou informace o jednotlivých snímcích z filmu. Jsou to například souřadnice středu snímku, čas snímkování atd.

tFilm

CisloFilmu	NazevFilmu	NazevLokality	Meritko	PocSnimku	DruhF	TypF	DatumZprac
53	AGFA H100(8/01)	Olomouc	13500	76	NEG	B	2.8.2001
54	AGFA H100(8/01)	Litovel	13500	76	NEG	B	3.9.2001

tSnimek

CisloFilmu	KameroveCislo	DatumSnimkovani	Pozn	X JTSK	Y JTSK
53	192	31.7.2001 12:00	2.řada	538735	1117021
53	193	31.7.2001 12:00	2.řada	538821	1125605
54	27	1.9.2001 10:00	1.řada	560423	1109831

Tab 9 Databáze evidence leteckých snímků

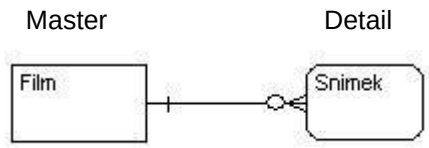
V tabulce tFilm je primární klíč CisloFilmu, v tabulce tSnimek je atribut CisloFilmu cizí klíč. Nadefinujeme-li referenční integritu ve vztahu mezi tabulkami tFilm a tSnimek, nemůžeme potom do tabulky tSnimek přidat záznam o jednom snímku, které nemá zaveden záznam o filmu v tabulce tFilm. To je přínos referenční integrity, neboť takový záznam by v tabulce tSnimek (jednotlivé snímky jednoho filmu) opravdu neměl být.

Představme si však tuto situaci. Potřebuji změnit číslo filmu z 53 na 153. Tento údaj nelze změnit jen v tabulce tFilm, v tabulce tSnimek by se ocitli sirotci. I v tabulce tSnimek, která je na straně N, se musí změnit také atribut CisloFilmu na hodnotu 153.

Potřebujeme v rámci referenční integrity dovolit tzv. a*ktualizaci souvisejících polí v kaskádě*. Aktualizace cizích klíčů se pak provádí automaticky. V našem případě nám potom SŘBD nejenže nechá opravit nesprávné číslo filmu, ale současně je ještě sám opraví u záznamů tabulky na straně N souvisejících se záznamem v tabulce první.

Referenční integrita nám též neumožní odstranit z tabulky tFilm záznam o filmu, který má v tabulce tSnímek odpovídající záznamy o jeho snímcích. Z těchto záznamů by se totiž staly záznamy osiřelé, což by narušilo samu podstatu referenční integrity. Proto je ještě dovoleno v rámci referenční integrity povolit o*dstraňování souvisejících polí v kaskádě*. Potom nám SŘBD nejen dovolí odstranit z tabulky tFilm záznam o filmu, ale navíc ještě sám odstraní z tabulky tSnimek jeho snímky, čímž zde nedojde ke vzniku osiřelých záznamů.

Takové dvě relace, které spolu úzce souvisí přes primární a cizí klíče, se obvykle nazývají **master** a **detail** nebo parent a dependent. Česky bychom je mohli nazvat hlavní a závislá.



Obr. 14 Master a detail relace

Povinnost členství ve vztahu je další důležité integritní omezení. Do některých vztahů musí vstupovat každá entita třídy entit, do jiného ne, záleží na modelované realitě. Definujeme tedy dva druhy členství ve vztahu:

- povinné (obligatorní)
- nepovinné (fakultativní)

Přitom může mít jedna entita povinné členství a druhá nepovinné. Jde vlastně o připuštění vazby 0:1, 1:0 (nepovinné členství), nebo jeho vyžadování (povinné členství).

Přechodová integrita

Omezení přechodové integrity definují stavy, mezi kterými může hodnota atributu právoplatně přecházet.

Můžeme si sestavit názorný diagram stavů a přechodů, např. pro rodinný stav. Osoba musí přejít ze stavu svobodný pouze do stavu ženatý. Ze stavu ženatý do stavu rozvedený nebo do stavu vdovec. Nelze přejít přímo ze stavu svobodný do stavu rozvedený.

Přechodová omezení mohou být definována přes několik atributů a dokonce přes několik tabulek.

4. Fyzická organizace dat – indexy

Vraťme se opět k SŘBD. SŘBD se stará o přístup k datům na interní (fyzické) úrovni. Uživatel se nestará o způsob uložení dat o přístup k nim při vyhledávání, editaci, mazání apod. Ale představu o významu indexování a použití indexů by měl tvůrce databáze mít. Vysvětleme si jej proto v této kapitole.

Velmi názorně si rozdíl mezi tabulkou s indexy a bez indexů můžeme vysvětlit na příkladu předchůdců moderních databázových platforem - klasické papírové databance. Představme si kartotéku se zdravotními záznamy pacientů u obvodního lékaře. Při návštěvě lékaře jsme v minulosti viděli v činnosti systém řízení takovéto databáze - zdravotní sestru (Lacko, 2001).

Zdravotní záznamy mohou být uložené v kartotéce náhodně nebo mohou být uspořádány podle vhodného primárního klíče. Náhodné uspořádání - složky se zdravotními záznamy jsou poskládané v kartotéce náhodně, bez jakéhokoli systému. Abychom našli hledanou složku, musíme začít obsah kartotéky prohledávat od první složky až do místa, kde se nachází hledaná složka. V průměru tedy vždy projdeme přibližně polovinu složek. Naproti tomu vkládání nových složek je velmi rychlé - jednoduše vložíme složku, kamkoli nás napadne.

Uspořádání podle primárního klíče - složky jsou v kartotéce uspořádány podle určitého pravidla, například v abecedním pořadí podle příjmení pacienta. Vyhledání je podstatně rychlejší (ke zrychlení povede i používání zarážek s jednotlivými písmeny). Vkládání nových složek je ale o něco složitější a pomalejší, protože musíme novou složku umístit přesně na pozici, která jí podle abecedního pořadí náleží. Zdálo by se, že na tomto systému již není co vylepšovat. Není to ale pravda - složky totiž nejsou uložené v jednom nekonečně dlouhém regálu, ale ve více skříních celé kartotéky. I přes pravidelnou údržbu databáze (zdravotní sestra pravidelně přesouvá jednotlivé složky mezi regály) může dojít k situaci, kdy může vložení nového záznamu trvat o mnoho déle. Předpokládejme, že typická tloušťka záložky jednoho pacienta jsou tři centimetry a že "databáze" je dobře udržovaná, což znamená, že v jednotlivých regálech je dostatek místa pro vložení několika nových záznamů pro každé písmeno. Jenže co když k lékaři přijde hypochondr, takový pan Kučera. Podle vyprávění pana Kučera by tloušťka jeho složky musela být alespoň jeden metr. Takto tlustý záznam se nám do police mezi ostatní záznamy s písmenem K již nevejde, proto musíme popřesouvat záznamy v ostatních regálech a vytvořit potřebné místo.

Problémy ale mohou být také jiného druhu - pokud potřebujeme zvát pacienty na pravidelné preventivní prohlídky, musíme procházet celou kartotéku a ve všech složkách ověřit datum poslední prohlídky.

Zavedení indexů

Naznačené problémy vyřeší přidání další malé kartotéky, která bude obsahovat lístky (zjednodušeně řešeno index) se základními údaji. Nyní již nemusíme vkládat nové složky na přesně určené místo. Abychom si případ s indexovanou kartotékou objasnili, popíšeme si ji co nejpodrobněji. Zavedeme si primární klíč, tedy jedinečné číslo, které napíšeme na viditelné místo na každé složce. Toto číslo budeme při vkládání nové složky automaticky zvětšovat o jedničku.

Mějme prázdnou databázi, v našem případě kartotéku lékaře, který si právě zařídil praxi; má prázdné kartotékové regály, krabici na minikartotéku s indexovými lístky a zdravotní sestru, která bude s kartotékou pracovat. Nyní již čekáme jen na to, až se přijde zaregistrovat první pacient. Když tato chvíle nastane a jako první přijde například pan Tomeček, uložíme jeho záznam (složku) na začátek regálu, přidělíme mu číslo (primární klíč) 1 a napíšeme toto číslo na složku. Potom vyplníme indexový lístek, který bude obsahovat primární klíč, jméno a datum poslední prohlídky. Lístek uložíme do minikartotéky pod písmeno T. Postupně se přichází zaregistrovat další pacienti a regály se úspěšně zaplňují. Pokud nyní přijde pan Malý, uložíme jeho záznam jednoduše na začátek dalšího regálu. Přidělíme mu pořadové číslo a vyplníme indexový lístek, který vložíme do minikartotéky pod písmeno M.

Zavedením primárního klíče a indexové minikartotéky jsme si vyhledávání patřičně usnadnili a ani zařazení nové složky není nijak složitou záležitostí. Pokud k nám přijde pacient, vyhledáme v minikartotéce jeho indexový lístek a zjistíme odpovídající primární klíč. Primární klíč nám v tomto případě přesně určuje polohu složky v regálu.

Problém zvaní pacientů na preventivní prohlídky jsme si ovšem až tak neusnadnili, stačí však mít dvě minikartotéky tedy dva indexy. Tato druhá minikartotéka bude obsahovat indexové lístky seřazené podle data poslední preventivní prohlídky. Potom při zvaní pacientů k prohlídkám stačí procházet tuto minikartotéku odzadu a po absolvování této prohlídky zařadit příslušný indexový lístek na začátek minikartotéky .

Podobně můžeme zavést také třetí minikartotéku indexových lístků, například podle věku či váhy. Potom nebude žádný problém nabízet starším občanům možnost očkování proti chřipce. Při existenci více indexů nám vkládání

zabere o něco více času. V našem případě nyní musí sestra vyplnit a zařadit tři indexové lístky pro každou nově vkládanou složku. Minikartotéka také pochopitelně zabere nějaké to fyzické místo.

Stejným mechanismem jako zdravotní sestra bude pracovat se záznamy a indexy i databázový systém.

Databázový systém dokáže vytvářet tzv. index pro zvolený atribut. Vytvoří se zvláštní tabulka, s jejíž pomocí se třídí záznamy a při zpracování se rychle nalezne požadovaný záznam. Indexové zpracování je mnohem rychlejší než sekvenční zpracování, kdy musíme procházet celou tabulku, abychom nalezli hledaný záznam. Index se vytváří vždy aspoň pro primární klíč.

Na příkladu si uveďme konkrétní obsah datového a indexového souboru.

datový soubor

pořadí	adresa	Rozloha	Pozemek	Typ
4	1	3800	Nad rybníkem	louka
3	2	2500	U lesa	louka
5	3	3900	Na mokré	mokřad
1	4	1100	V lánech	orná půda
6	5	4500	Kocouří	mokřad
2	6	2000	U křížku	louka

indexový soubor

adresa	rozloha
4	1100
6	2000
2	2500
1	3800
3	3900
5	4500

Tab 10 Ukázka indexovaného a indexového souboru

Zavedením indexů výrazně zkrátíme přístupový čas k požadovaným datům, přičemž podstatně redukujeme množství diskových operací. Moderní databázové platformy používají pro indexy nejčastěji stromové struktury, například tzv. B-stromy. Pro B-stromy platí, že všechny cesty od kořene stromu do libovolného listu jsou stejně dlouhé.

Používání indexů

Na první pohled to vypadá tak, že čím více použijeme indexů, tím dosáhneme lepšího výkonu při práci s databází. Bohužel toto tvrzení není vždy pravdivé. Indexy bychom měli používat velmi rozvážně a zakládat je pouze v případě, kdy je to opravdu výhodné. Indexy totiž mají také své nevýhody, v některých případech mohou podstatně zpomalit databázovou aplikaci a výrazně zvýšit nároky na diskovou kapacitu. Indexové tabulky zabírají určité místo. Také je potřeba je vždy při změně aktualizovat indexové soubory a aktualizace mnoha indexových souborů při přidání jen jednoho záznamu může být časově náročné. Vzhledem k dnešním parametrům výpočetní techniky nebývá toto však problémem.

Stejně tak nedoporučujeme používat indexy nad atributy s nízkou variabilitou, například pokud máme ve sloupci pouze dvě možné hodnoty. Index nám v tomto případě příliš nepomůže, protože bude vyhledávací mechanismus směřovat na začátek tabulky nebo někam doprostřed (počátek dat odpovídající druhé hodnotě indexu závisí na poměru dat pro jednotlivé hodnoty indexu). Tento problém řeší sekundární indexy.

5. Logická organizace báze dat

Historický vývoj směřoval od prvotního zpracování dat na úrovni fyzických záznamů přes datové záznamy bez vzájemných vazeb až po báze dat obsahující v sobě popis dat včetně vazeb.

V současné době má smysl rozebírat dva modely logické organizace bází dat: hierarchický (také hierarchicko - síťový) model a relační model. Problematika hierarchicko – síťových modelů už má dnes malé použití a tak bude uvedena jen informativně pro všeobecný přehled.

5.1. Hierarchicko - síťový model

Model vychází z použité hierarchické struktury dat tak, jak byla kdysi zavedena pro potřeby jazyka Cobol pro zobrazení hodnot dat a jejich vzájemných vztahů (nadřízenosti a podřízenosti). Tento model se neopírá o matematickou teorii, i když přejímá část terminologie z **teorie grafů**. Přesto nalezl dříve v praxi široké uplatnění.

Hierarchická struktura je taková, kde záznamy jsou v hierarchickém vztahu nadřazenosti a podřízenosti. Přitom se používá "rodinná" terminologie **rodič** a **potomek** ve zřejmém významu.

V hierarchické struktuře má každý potomek jediného rodiče, existuje jediný rodič, který není potomkem a potomek v jednom vztahu může být rodičem v jiném vztahu.

Pokud je zapotřebí popsat, na kterém místě hierarchické struktury se nějaký záznam nalézá, používá se k tomu tzv. přístupová cesta. To je možno díky popsaným vlastnostem hierarchické struktury, které zaručují, že od kořene lze dojít k danému záznamu jediným způsobem.

Často se vyžaduje (většinou z ryze praktických důvodů např. sběru dat), aby v každém záznamu existoval klíč. V takovém případě lze přístupovou cestu popsat jednoduše jako posloupnost klíčů počínaje klíčem kořenového záznamu přes klíče všech nadřízených až po klíč daného záznamu včetně.

Zmíněné pojmy z teorie grafů se při popisu hierarchických struktur využívají v tomto smyslu:

záznam = uzel grafu

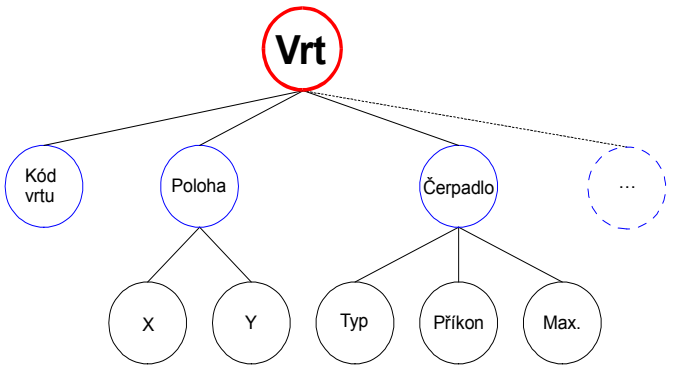
vztah rodič - potomek = hrana grafu

rodič a potomek = incidenční uzly hrany

hierarchická struktura = souvislý graf, který je stromem (graf bez cyklů)

báze dat hierarchického modelu = graf, který je les (tj. množina disjunktních stromů)

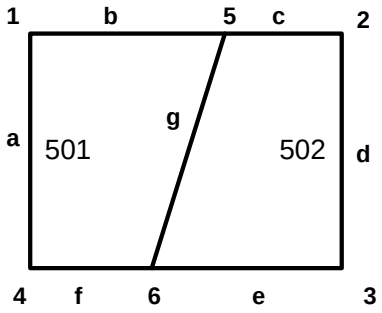
přístupová cesta k záznamu = cesta v grafu od kořene k danému uzlu.



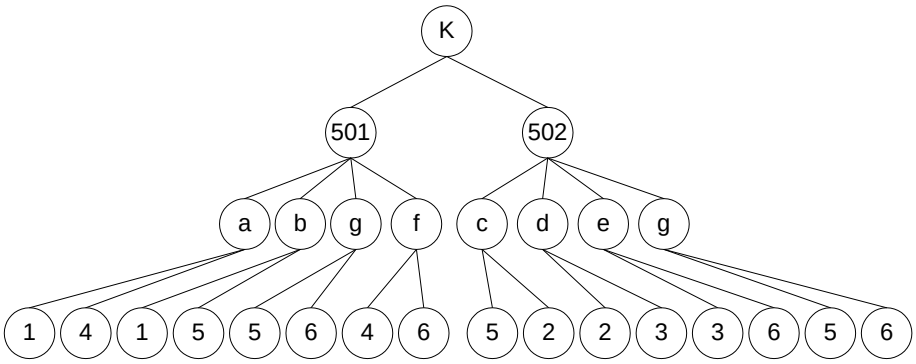
Obr. 15 Stromová struktura

Nevýhodou hierarchických systémů je velmi obtížná implementace odkazů. V takových případech se sice rozšiřují možnosti, snižuje redundance dat, ale současně dochází k nutnosti promíchávat otázky uložení na médiu s otázkami struktury modelu, ke znepřehlednění a zvláště ke snížení abstrakce při práci s daty.

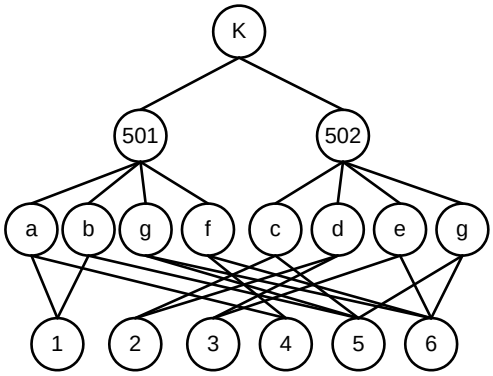
Uveďme si příklad uložení informací o plánu K, který má dvě parcely 501 a 502. Popis stran a vrcholů si provedeme tak, jak je patrné z obrázku. Následuje záznam informací do hierarchické, síťové a relační databáze.



Obr. 16 Plán K s parcelami 501 a 502



Obr. 17 Hierarchická databáze



Obr. 18 Síťová databáze

tPlan

ID_parcela
501
502

tParcela

ID_parcela	ID_hrana
501	a
501	b
501	g
501	f
502	g
502	c
502	d
502	e

tHrana

ID_hrana	od	do
a	1	4
b	1	5
c	5	2
d	2	3
e	3	6
f	4	6
g	5	6

tBod

ID_bod	x	y
1	…	…
2	…	…
3	…	…
4	…	…
5	…	…
6	…	…

Obr. 19 Relační databáze

5.2. Relační model

Jedním z nejjednodušších zápisů dat je zápis do klasické tabulky. Takto převážně vznikají zápisy při prvotním pořizování dat. Charakteristický pro tento zápis je její členění do sloupců, z nich každý má nadpis. Nadpis ve smyslu identifikace údajů naznačuje také typ údajů v daném sloupci.

Pevným počtem n sloupců tvoří data v řádcích uspořádané n-tice. Každý prvek n-tice, nazývaný pole, je toho typu, jaký odpovídá typu sloupce. Je tedy prvkem (konečné nebo nekonečné) jednoznačně určené množiny D_i (např. množiny všech datumů, množiny všech racionálních čísel, množiny {Ano;Ne} apod).

Relační datový model je spojen se jménem Edgara F. **Codda**, pracovníka firmy IBM. Teoretický popis modelu navrhl v roce 1970 publikací článku „A relational model of data for large shared data banks“.

Matematický základ relačního modelu

Množina všech uspořádaných n-tic $\mathbf{<a_1, a_2, ... , a_n>}$, kde A_i jsou libovolné neprázdné množiny, a_i je prvkem z A_i a n je přirozené číslo, je z teorie množin známa jako kartézský součin $\mathbf{K = A_1 \times A_2 \times ... \times A_n}$ a libovolná podmnožina R z K je známa jako n-ární relace v K. Je tedy možno pohlížet na každou tabulku, která je vytvořena obdobně jako tabulka shora, jako na n-ární relaci.

Pro určení relace R pro potřeby modelu báze dat je zapotřebí zadat konečnou množinu atributů F - což jsou jména polí s případnými dalšími specifikacemi (šířka sloupce, počet desetinných míst apod.) využitelných pro uživatelskou i programovou identifikaci domény A_i, tj. množiny možných hodnot každého pole podmnožinu kartézského součinu domén, tj. vlastní relaci (z hlediska tabulky je tím určen počet polí a pořadí sloupců).

Relaci je tedy možno definovat jako trojici $\mathbf{R = <F, D, T>}$, kde

F je konečná množina jmen atributů.
D je zobrazení, přiřazující každému atributu f_i z F doménu $D(f_i)$. Domény $D(f_i)$ jsou libovolné neprázdné množiny (konečné nebo nekonečné). Je-li f_i a f_j z F, f_i různé od f_j , nemusí být $D(f_i)$ různé od $D(f_j)$. Tzn. např. atribut jméno i příjmení jsou ze stejné domény - text.
T je konečná podmnožina kartézského součinu X [D(f)] všech domén atributů f z F.

Důležité vlastnosti relačního datového modelu jsou:

- data jsou uživateli zobrazena vždy ve formě tabulek
- data jsou uložena nezávisle na formě tabulek
- uživatel se nemusí starat o fyzickou strukturu a uložení dat
- tabulky jsou v databázi identifikovány jménem
- nezáleží na pořadí sloupců v tabulkách
- nezáleží na pořadí řádků v tabulkách.
- každé n-tici (resp. výskytu entity) odpovídá jeden řádek tabulky
- žádné dva řádky nejsou identické
- sloupec s atributem f z F v záhlaví obsahuje jen hodnoty z domény D(f).

Okolnost, že v praxi často nebývá splněna identičnost řádku, se řeší přidáním sloupce – umělého klíče.

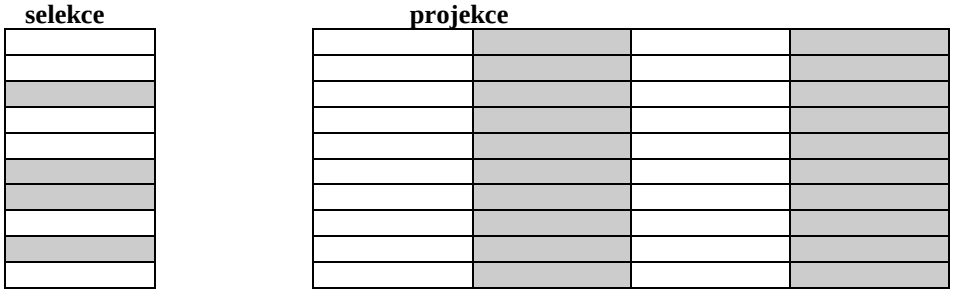
Relační algebra

Relační algebra je množina operací na relačním modelu. Skládá se z operátorů, jednotlivé relace zde vystupují jako operandy operace. Výsledkem množinové operace je opět vždy relace.

Operace dělíme na unární a n-ární.

Unární operace znamená, že do dané operace vstupuje pouze jedna relace.

Unární operace jsou :



Obr. 20 Operace selekce a projekce

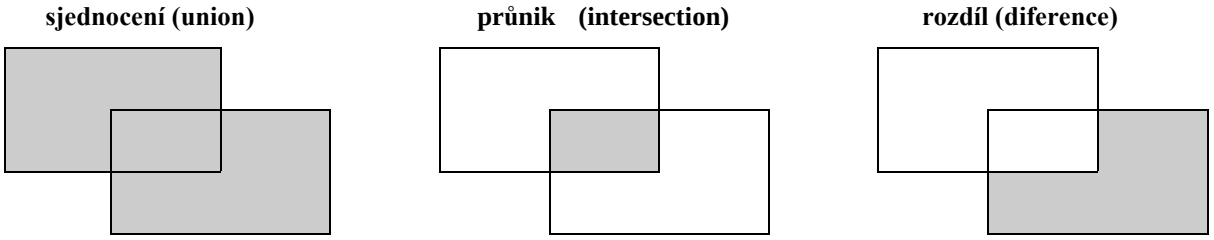
Selekce (někdy označovaná jako restrikce) vybírá pouze některé relace, které vyhovují zadané podmínce. Například vybereme všechny osoby žijící v Brně.

Projekce vybírá pouze některé atributy ze vstupní relace. Například vybereme pouze atribut příjmení a jméno z evidence osob.

Obě operace lze uplatnit i zároveň.

N-ární operace (vstupuje do nich *n* relací). Nejčastěji jsou to dvě relace, takže operaci můžeme označit jako *binární.*

Mezi n-ární operace řadíme množinové operace:



Obr. 21 Množinové operace

Dalšími operacemi jsou operace **spojení** (join). Spojení kombinuje dohromady n-tice vstupních relací.

Druhy spojení jsou následující:

- **vnější** - outer join

  vnější spojení se dělí dále na:

  1. **levé** - left outer join
  2. **pravé** - right outer join
  3. **plné** - full join

- **vnitřní** - inner join

Posledním druhem spojení je **křížové spojení** - cross join, to odpovídá **kartézskému součinu**. Kartézský součin (produkt) vrací relaci sestávající z kombinací všech n-tic obou vstupních relací.

Na dvou tabulkách RelaceX a RelaceY si ukážeme výsledky jednotlivých operací názorně.

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Levé vnější spojení X a Y

AtributA	AtributB	AtributC
A1	B1	C22
A2	B1	C22
A3	B2	C23
A4	B4	NULL

Obr. 22 Levé vnější spojení

Levé vnější spojení je takové, kdy se vezmou všechny záznamy levé tabulky (RelaceX) a k nim se vyhledají záznamy z pravé tabulky (RelaceY) podle společného atributu. Existuje-li společná hodnota atributu, bude záznam připojen. V případě, že neexistuje, bude záznam obsahovat pro další atributy (AtributC) hodnotu NULL

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Pravé vnější spojení X a Y

AtributA	AtributB	AtributC
A1	B1	C22
A2	B1	C22
A3	B2	C23
NULL	B3	C24

Obr. 23 Pravé vnější spojení

Pravé vnější spojení bere všechny záznamy z pravé tabulky (RelaceY) a k nim se vyhledávají záznamy z levé tabulky (Relace). Opět se dosadí hodnoty NULL u chybějících atributů.

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Plné spojení X a Y

AtributA	AtributB	AtributC
A1	B1	C22
A2	B1	C22
A3	B2	C23
A4	B4	NULL
NULL	B3	C24

Obr. 24 Plné spojení

Plné spojení bere všechny záznamy z obou tabulek a u chybějících atributů dosazuje hodnotu NULL.

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Vnitřní spojení X a Y

AtributA	AtributB	AtributC
A1	B1	C22
A2	B1	C22
A3	B2	C23

Obr. 25 Vnitřní spojení

Při vnitřní spojení se spojí záznamy podle společné položky (AtributB) a to jen tehdy, když hodnota položky existuje v obou tabulkách. Záznamy bez odpovídajících hodnot atributů (B3, B4) nejsou ve výsledné relaci.

Relace XX

AtributA
A1
A2
A3

Relace YY

AtributC
C22
C44

Křížové spojení XX a YY

AtributA	AtributC
A1	C22
A1	C44
A2	C22
A2	C44
A3	C22
A3	C44

Obr. 26 Křížové spojení – kartézský součin

Křížové spojení vrací relaci obsahující kombinaci všech n-tic obou vstupních relací. Spojují se spolu všechny záznamy z jedné tabulky se všemi záznamy z druhé tabulky. Má-li první tabulka m záznamů a druhá tabulka n záznamů, má výsledná tabulka m x n záznamů.

Odpovídající klíčová slova jazyka SQL pro jednotlivá spojení jsou uvedena v kapitole 7.5 o jazyku SQL.

6. Normalizace a normální formy

Při návrhu relačních databázových modelů je prvním úkolem návrh samotných relací. Cílem je minimalizace všech redundancí a problémů s nimi spojených. Redundance plýtvají jednak prostředky a velice obtížná je aktualizace opakovaných informacích na všech místech. Např. v Tab. 16 je opakovaně telefon dodavatele. Při jeho změně se musí aktualizovat všechny záznamy, kde je tato informace. Jedná se o redundantní informaci. Návrh musí být takový, aby model byl schopen správně odpovědět na různé dotazy.

Jinak řečeno: každá informace se vyskytuje v systému pokud možno jen jednou.

Na začátku návrhu je nenormalizovaná struktura relací, jak si je většinou představí uživatel. Uživatelovy představy o obsahu databáze je nutné upravit do stavu, použitelného v relačním systému. Napomáhá mu proces normalizace.

Proces normalizace znamená postupné **rozložení** původních "nenormalizovaných" tabulek do soustavy více menších tabulek tak, aby nedošlo ke ztrátě informací v původní tabulce obsažených.*(Hnojil, 1999)*

Tento proces je označován jako **bezztrátová dekompozice** - je možné kdykoliv vytvořit si původní relaci nazpět.

S procesem normalizace databáze úzce souvisí funkční závislost jedno z integritních omezení.

6.1. První normální forma (1.NF)

Relace je v první normální formě (l.NF), jestliže každý její atribut je atomický. Pojem atomický není přesně definován, myslí se jím atributy, jejichž domény jsou v jistém smyslu množiny jednoduchých hodnot, čísel, znaků, slov v nějaké abecedě apod., logicky dále nedělitelných (Šarmanová, 1997).

Každou relaci lze na l.NF převést tak, že neatomické atributy nahradíme odpovídajícími jednoduchými atomickými atributy nebo dekompozicí na více relací.

Ukažme si relaci, která obsahuje objednávky digitálních map od ČÚZK

tObjednavka

CisloObj	KodZakaznika	DatumObj	Polozka	CenaCelkem
10	CACTV	01.10.2001	Mapa ČR 1 : 500 000 - vodstvo, lesy, polohopis	4500 Kč
20	BSBEM	12.11.2001	RZM 200 celá ČR, Mapa ČR 1 : 500 000- železnice	61500 Kč

Tab 11 Příklad nenormalizované relace se neatomickým atributem Polozka

Atribut Polozka je zde neatomický atribut. Jsou zde uvedeny všechny objednané mapy. Jde též o chybu ve smyslu spojení dvou entit (objednávka a položek objednávky) do jedné relace. Provedeme dekompozici na dvě relace.

tObjednavka

CisloObjednavky	KodZakaznika	DatumObjednavky
10	CACTV	01.10.2001
20	BSBEM	12.11.2001

tDetailObjednavky

CisloObjednavky	CisloPolozky	Polozka	CenaZaKus
10	1	Mapa ČR 1 : 500 000 – vodstvo	1500
10	2	Mapa ČR 1 : 500 000 – lesy	1500
10	3	Mapa ČR 1 : 500 000 – polohopis	1500
20	1	RZM 200 celá ČR	60000
20	2	Mapa ČR 1 : 500 000 - železnice	1500

Tab 12 Výsledek dekompozice

Častou chybou je též **opakovaná skupina atributů** (Riordan 2000). Příkladem je rozpis objednávky hardwarových komponent.

CisloObj	KodZakaznika	Polozka1	Mnozstvi1	Polozka2	Mnozstvi2	Poloz3	Mnoz3	Poloz4	Mnoz4
1	ANTON	kabel	1	podložka	1	myš	3	CD	7
2	CVUYT	FDD	2	sluchátka	5	HDD	1	0	0

Tab 13 Příklad opakované skupiny atributů

Předem při návrhu omezujeme nákup na maximálně 4 položky. Při nákupu jen jedné položky zůstávají ostatní položky s hodnotou NULL. Zbytečně narůstá objem databáze, která je nevyužita.

Další příklad opakované skupiny je tentokrát relace pro evidenci průtoku a stavu vody na řekách. Měření na měřící stanici se provádí v průběhu dne v určitou hodinu. Tento návrh již předem omezuje počet a typ měření. Není možné uložit více měření za den nebo v jinou hodinu.

Obtížně budeme sestavovat dotaz, kterým bychom chtěli zjistit všechny stanice, kde se změnil v libovolném čase průtok o 10%.

Merici stanice	Prutok 0:00	Stav 0:00	Prutok 5:00	Stav 5:00	Prutok 12:00	Stav 12:00
Moravičany	6,55	87	6,60	89	6,65	92
Kašava	0,05	35	0,05	35	0,08	36

Merici stanice	Datum	Cas	Prutok	Stav
Moravičany	17.5.2003	0:00	87	6,60
Kašava	17.5.2003	5:00	0,05	35

Tab 14 Stavy a průtoky na řekách nenormalizované a normalizované relace

Častou chybou jsou též atributy, které jsou různé **složené kódy** a příznaky. Tyto kódy jsou také neskalárními hodnotami. Například značení geomorfologických jednotek na území ČR je takovým případem. Kód okrsku Nemanická vrchovina je IA-1A-b. V tomto kódu je význam následující:

kód	název	typ geomorfologické jednotky
I	Šumavská soustava	soustava
IA	Českoleská podsoustava (oblast)	podsoustava
IA-1	Český les	celek
IA-1A	Čerchovský les	podcelek
IA-1A-a	Haltravská hornatina	okrsek
IA-1A-b	Nemanická vrchovina	okrsek

Tab 15 Význam kódu v geomorfologickém členění

Při ukládání do databáze je vhodnější uložit geomorfologické jednotky do samostatných tabulek (soustava, podsoustava, celek, podcelek,…). Nebo je možné uložení v jedné tabulce s rekurzní vazbou, kde je vyjádřen vztah nadřazenosti. Detailněji je tento problém diskutován v Dobešová(2002).

Uplatnění 1.NF se zdá na první pohled velice jednoduché, ale někdy může být právě tím nejobtížnějším krokem.

6.2. Druhá normální forma (2.NF)

Relace je v druhé normální formě (2.NF) pokud je v první normální formě a navíc všechny její neklíčové atributy jsou závislé na každém atributu složeného klíče.

Příklad relace o dodávkách digitálních prostorových dat, která nevyhovuje 2.NF. V této relaci je primárním klíčem složený klíč *NazevVyrobku+JmenoDodavatele*. Atribut TelefCisloDodavatele je však závislý pouze na části klíče a to na atributu JmenoDodavatele.

tDodavka

NazevVyrobku	JmenoDodavatele	Kategorie	TelefCisloDodavatele
Chráněná území přírody ČR	T-MAPY	vektorová data	495 513 335
Silniční síť ČR 1: 50 000	T-MAPY	vektorová data	495 513 335
RZM 10	ČÚZK Praha	rastová základní mapa	284 041 561

Tab 16 Nenormalizovaná relace se složeným klíčem

Výsledkem bezztrátové dekompozice jsou dvě relace tVyrobek a tDodavatel.

tVyrobek

NazevVyrobku	Kategorie	Dodavatel
Chráněná území přírody ČR	vektorová data	T-MAPY
Silniční síť ČR 1: 50 000	vektorová data	T-MAPY
RZM 10	rastová základní mapa	ČÚZK Praha

tDodavatel

Dodavetel	TelefCisloDodavatele
T-MAPY	495 513 335
ČÚZK Praha	284 041 561

Tab 17 Dvě relace, které již splňují druhou normální formu

6.3. Třetí normální forma (3.NF)

Relace je v třetí normální formě (3.NF), jestliže je v 2.NF a navíc všechny její neklíčové atributy jsou tranzitivně nezávislé na primárním klíči.

Příkladem je PSČ a název města v tabulce tFirma. Tyto dva atributy jsou na sobě závislé a jsou neklíčové. Při změně města se mění i údaj PSČ. Úprava, aby byla splněna 3.NF vede opět na vytvoření samostatné entitity PSČ. Zda v tomto případě uplatnit 3.NF je třeba uvážit. Činíme tak, jen tehdy pokud se údaje často mění, což není případ PSČ.

tFirma

Firma	Adresa	Mesto	PSC
Autocont	Tř.1.máje 2	Olomouc	772 00
Konsigna	Tř. Svobody 42	Olomouc	779 00
TechData	Na Radosti 414	Praha 5 - Zličín	155 21

Tab 18 Relace se závislými neklíčovými atributy Mesto a PSC.

Další příkladem je relace, kde si eviduji mnou zakoupené listy vybraných digitálních map vydávaných Českým úřadem zeměměřickým a katastrálním. V tomto návrhu jsou na sobě závislé atributy TypMapy a Meritko.

tMapa

ID_Mapy	TypMapy	Popis	Meritko
501	RZM 10	Rastrová Základní mapa ČR	1:10000
502	RZM 25	Rastrová Základní mapa ČR	1: 25000
502	RZM 50	Rastrová Základní mapa ČR	1: 50000
504	RZM 200	Rastrová Základní mapa ČR	1:200000
505	SM5	Státní mapa	1: 5000
506	SMO5	Státní mapa odvozená	1: 5000

Tab 19 Relace se závislými neklíčovými atributy TypMapy a Meritko.

Shrnutí

Postup normalizačního procesu by se dal shrnout následovně :

1. nenormalizovaný tvar
2. eliminace atributů, které obsahují jako prvky relace → l.NF
3. odstranění částečné závislosti neklíčových atributů na klíčích → 2.NF
   (odstranění neúplných závislostí neklíčových atributů na atributech složeného primárního klíče)
4. eliminace tranzitivní závislosti neklíčových atributů na primárním klíči → 3.NF

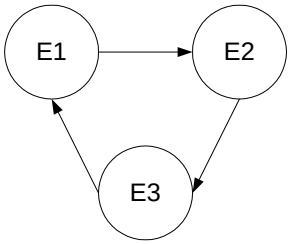
6.4. Další normální formy

První tři normální formy v roce 1972 definoval E. F. Codd. Společně s Raymondem F. Boycem později stanovil další normální formu, tzv. Boyce-Coddovu normální formu (BCNF). Používá se pro speciální případy relací s více kandidátními klíči. O relaci můžeme prohlásit, že je v BCNF, jsou-li všechny neklíčové atributy navzájem nezávislé, a navíc jsou všechny kandidátní klíče úplně nezávislé navzájem.

Je-li relace v BCNF, pak je i v třetí normální formě.

Čtvrtá normální forma vyjadřuje princip: v jedné relaci se nesmí spojovat nezávislé opakované skupiny.

Pátá normální forma se týká poměrně vzácného případu spojené závislosti. Kdy entita E2 závisí na entitě E1 a entita E3 závisí na E2 a nakonec entita E1 závisí na entitě E3.



Tyto zbývající tři normální formy se používají pouze ve výjimečných případech.

7. Jazyk SQL

7.1. Databázové jazyky

Pro práci s SŘBD se většinou nepoužívá některý z klasických programovacích jazyků, ale SŘBD má definován vlastní jazyk.

Příkazy jazyka se dělí podle druhu své činnosti na:

1. Příkazy jazyka pro definici dat (JDD) neboli Data Definition Language (DDL)

2. Příkazy jazyka pro manipulaci s daty (JMD) neboli Data Manipulation Language (DML)

Podstatnou část skupiny jazyků pro manipulaci s daty (JMD) tvoří **dotazovací jazyky**. Dotazovací jazyky slouží uživatelům k získání potřebných informací z databáze. JMD dělíme na procedurální a neprocedurální, a stejné rozdělení můžeme použít i pro dotazovací jazyky.

Procedurální dotazovací jazyky vyžadují algoritmický popis vyhledávání dat. Z tohoto důvodu obsahují prvky if-then-else, while-goto, umožňující přechody mezi prvky datové struktury.

V **neprocedurálních** jazycích jsou namísto popisu vyhledávání zadávány vymezující podmínky, kterým mají hledaná data vyhovovat. Tím se více **přibližují přirozenému jazyku.**

Dotazovací jazyk musí obsahovat porovnávací a aritmetické operátory, logické spojky, množinové funkce, aritmetické funkce, možnosti práce s časovými údaji a funkce pro formátování výstupních dat. Z hlediska efektivnosti práce je kladen požadavek na možnost opětovného použití již vytvořených dotazů.

Kromě neprocedurálního dotazovacího **jazyka SQL**, jehož základy si popíšeme v následujících kapitolách, existuje mnoho úspěšných komerčních výrobků, které využívají dalších relačních dotazovacích jazyků, například QUEL či QBE. U jazyka QBE je dotaz vytvářen za podkladě grafického či vyjádření databázových tabulek.

Dotazovací jazyk QBE (Query-By-Example) je druhým nejrozšířenějším relačním dotazovacím jazykem. Jak název napovídá, uživatel má při tvorbě dotazu k dispozici "příklad". Jazyk je určen především pro koncové uživatele (úředníky, asistentky), kteří mají malé znalosti dotazovacího jazyka. Například MS Access obsahuje společně s možností vytvářet dotazy pomocí jazyka SQL i jazyk QBE.

7.2. Historie jazyka SQL

Jazyk SQL je standardním neprocedurálním relačním dotazovacím jazykem, podporovaným většinou dostupných databázových systémů.

První návrh jazyka pochází z počátku sedmdesátých let (1974), kdy byl jeho prototyp součástí databázového systému System/R. V té době zněl jeho název SEQUEL (Structured English Query Language). Později byl zkrácen na SQL (Structured Query Language). System/R byl úspěšný a tak na jeho základě společnost IBM vytvořila další systémy SQL/DS, QMF a DB2.

V letech 1973 až 1979 publikovali vědci firmy IBM velké množství publikací v akademickém tisku. Firma IBM však byla velmi pomalá ve svých obchodních strategiích a tudíž s prvním komerčně úspěšným relačním databázovým systémem podporujícím jazyk SQL přišla na trh firma ORACLE, založená v roce 1977.

V roce 1986 byl jazyk standardizován americkou ANSI, a o rok později byl následován Mezinárodní organizací pro normalizaci (ISO). ISO norma pro SQL jazyk nese označení ISO 9075. Dále je v označení oddělující dvojtečka a rok uvedení verze.Verze jazyka pocházející z roku 1986 obsahuje původní dialekt jazyka SQL. Často se označuje jako SQL86. Jazykové prostředky SQL86 se jevily jako nedostatečné a proto pokračovaly práce na začleňování dalších konstruktorů. Výsledkem byla rozšířená verze jazyka, schválená organizací ISO v roce 1989 (ISO/IEC 9075:1989), je označována jako SQL89. V roce 1992 byla standardizována verze jazyka, která je kvůli odlišení od předchozích verzí označována jako SQL92.

V roce 1993 se komise, zabývající se jazykem SQL v ISO a ANSI, rozhodly, že další vývoj bude rozdělen do sedmi částí. V roce 1996 pokračoval vývoj jazyka v podobě SQL3. V roce 1999 byl zveřejněn SQL/MultiMedia označovaný jako SQL/MM. V roce 1999 bylo zveřejněno 5 částí ISO 9075:1999 pro jazyk SQL. Poslední normou pro jazyk SQL je z roku 2003 (ISO/IEC 9075:2003).

Jazyk SQL v současné době obsahuje více než jen původní dotazovací jazyk. Součástí jazyka jsou prostředky pro definici dat a jejich aktualizaci a dále definování přístupových práv k databázím, tabulkám a atributům.

Jazyk může být využíván v rámci některého programovacího jazyka (například PL/I, COBOL, C/C++). Tato verze SQL je označována jako hostitelská. Kromě ní existuje ještě interaktivní verze SQL (ISQL, interactive SQL). V případě využití hostitelské verze jsou SQL jsou příkazy vždy nějakým způsobem odděleny od příkazů programovacího jazyka.

Implementace jazyka SQL se pro jednotlivé databáze liší. Výrobci se však snaží plnit ISO normy. Například SQL server 2000 od firmy Microsoft nazývá svůj jazyk Transact SQL (T-SQL). Podobně databáze ORACLE 9i má své specifické řešení a tím je jazyk PL/SQL, který je rozšířením standardního jazyka o některé možnosti vyšších programovacích jazyků: větvení, skoky, cykly, vytváření procedur, funkcí a knihoven.

SQL obsahuje příkazy pro:

1. **Definici dat**
2. **Editaci dat**
3. **Výběr dat**
4. **Definici přístupových práv**
5. **Transakce**

Databáze v SQL se skládá ze souborů, obsahující tabulky, a pohledů. Pohledy se jeví jako virtuální tabulky.

Některé systémy vyžadují zakončení SQL příkazu středníkem.

Příkazy uvedené v následujících kapitolách jsou podle normy ISO/IEC 9075:1992.

7.3. SQL příkazy pro definici dat

Příkaz pro vytvoření databáze je

CREATE DATABASE *jméno_databáze*

Místo *jméno_databáze* napište konkrétní jméno databáze, například:

CREATE DATABASE povodi

Základními výrazy pro definici tabulek jsou CREATE TABLE, ALTER TABLE a DROP TABLE.

Vytvoření tabulky vypadá takto:

CREATE TABLE *jméno_tabulky*

(seznam_atributů_ tabulky_oddělených_čárkami,

seznam_integritních_omezení)

Jméno_tabulky nahraďte skutečným jménem tabulky, dále v kulatých závorkách definujte jména atributů a jejich datových typů. Atributy jsou odděleny čárkami.

Př. CREATE TABLE tOsoba

(cislo CHARACTER(6) NOT NULL PRIMARY KEY,

jmeno CHARACTER(15),

prijmeni CHARACTER(25) NOT NULL,

vaha INTEGER,

datum DATE DEFAULT 1.1.1900);

V rámci definice tabulky se příkazy nastavují i integritní omezení. Jsou následující:

NOT NULL – nepovoluje prázdnou hodnotu NULL

DEFAULT – počáteční hodnota

UNIQUE - všechny hodnoty ve sloupci musí být unikátní

PRIMARY KEY – primární klíč

FOREIGN KEY – cizí klíč

CHECK – zadaný logický výraz definuje přídavné integritní omezení pro hodnoty atributu

CONSTRAINT – integritní omezení

Příklad integritního omezení může vypadat takto:

```
CHECK (rozloha > 4500 AND rozloha < 90000)
CHECK (VALUE IN 'orná půda','zahrada ','chmelnice ','vinice','ovocný sad','trvalý travní porost')
```

Příklad definice tabulky, kde je definován složený primární klíč musí být deklarován samostatnou klauzulí takto:

```
CREATE TABLE tVztahVlastnikNemov
    (IDVlast INTEGER NOT NULL FOREIGN KEY (IDVlast) REFERENCES tVlastnik,
    IDNem INTEGER NOT NULL FOREIGN KEY (IDNem) REFERENCES tNemovitost,
    PRIMARY KEY (IDVlast,IDNem))
```

Příkaz na smazání celé tabulky je

DROP TABLE jméno_tabulky

Př. DROP TABLE tOsoba

Může nastat též požadavek na dodatečnou změnu struktury tabulky. Příkaz, který to provede je

ALTER TABLE jmeno_tabulky zmena1[, zmena2…]

Př. ALTER TABLE tOsoba ADD vyska INTEGER - přidá nový sloupec vyska

ALTER TABLE tOsoba ADD RodneCislo CHARACTER(11) UNIQUE

ALTER TABLE tOsoba DROP vyska - odstraní sloupec vyska

ALTER TABLE tOsoba MODIFY jmeno CHARACTER(20) - změní rozsah datového typu

ALTER TABLE tOsoba ADD INDEX RodC (RodneCislo)

- vytvoří nový index pro atribut RodneCislo, který nazve RodC.

Požadavek na vytvoření indexů se realizuje příkazem:

CREATE INDEX [UNIQUE] jméno_indexu ON název_tabulky sloupec [ASC|DESC]

Index lze definovat pro jeden atribut i pro kombinaci atributů. Implicitně jsou hodnoty uspořádány vzestupně, pokud není udáno DESC. Pro jednu tabulku lze zadat libovolné potřebné množství indexů, které se musí samozřejmě lišit jménem. Index lze zadat jako UNIQUE, což znamená, že jde o primární index.

Př. CREATE INDEX OsobCislo ON tOsoba cislo

Index lze zrušit příkazem **DROP INDEX** jméno_indexu. Např. tedy pro zrušení výše zavedeného indexu napíšeme:

DROP INDEX OsobCislo

Pohledy umožňují vidět aplikaci uživatelskýma očima. Vytváří se virtuální tabulky, které obsahují jen některé atributy nebo záznamy. Můžeme se dívat na část dat, aniž bychom přesně znali strukturu zdrojových tabulek. Pohledy se implementuje externí schéma. Užitečnými jsou případy, kdy chceme uživatele odstínit od nějakých dat z bezpečnostních důvodů, v příkladu neuvidíme váhu zaměstnanců. Požadavek na vytvoření pohledů se realizuje příkazem:

CREATE VIEW název_pohledu [pojmenování_sloupců] AS select

Př. CREATE VIEW VerejneInfo AS SELECT cislo, prijmeni, jmeno FROM tOsoba

Př. CREATE VIEW InformaceZam (CisloZamestnance, prijmeni, jmeno)

AS SELECT cislo, prijmeni, jmeno FROM tOsoba

V pohledu InformaceZam došlo k přejmenování položky cislo na CisloZamestnance.

7.4. SQL příkazy pro editaci dat

Slouží k vkládání, odstraňování a aktualizaci dat. Jsou to příkazy INSERT, UPDATE, DELETE.

Příkaz pro vložení jednoho řádku (pozor opravdu jen jednoho) je:

INSERT INTO jméno_tabulky ([jména_sloupců]) VALUES (seznam hodnot)

Př. INSERT INTO tOsoba VALUES (‘101‘,‘Jana‘,‘Vydrová‘,65,’01.05.72‘)

Atributy typu text, datum se musí uzavřít do apostrofů, číselné hodnoty ne.

Můžeme vložit jen některé hodnoty v případě, že ostatní nejsou dosud známé (zde neznáme váhu a datum narození) :

INSERT INTO tOsoba (cislo, prijmeni, jmeno) VALUES (‘102‘,‘Kosíková‘,’Jana‘)

Změnu dat provádí příkaz:

UPDATE jmeno_tabulky SET atribut=nova_hodnota WHERE atribut=specificka_hodnota

Př. UPDATE tOsoba SET prijmeni=“Kotoučková“ WHERE cislo=‘101‘

Aktualizuje atribut záznamu, který vyhovuje podmínce. Takových záznamů může podmínce vyhovovat i více. Potom se jedná o hromadnou změnu.

DELETE FROM jmeno_tabulky WHERE podminka

Smaže všechny záznamy vyhovující podmínce následující za WHERE.

7.5. SQL příkazy pro výběr dat

Základním příkazem jazyka SQL pro výběr dat je příkaz SELECT. Je to nejpoužívanější příkaz a jeho syntaxe je nejkošatější.

Příkaz pro výběr provádí výběr záznamů odpovídajících specifikovaným kriteriím. Tyto příkazy mají většinou strukturu:

SELECT seznam_jmen_atributů
FROM seznam_jmen_tabulek
[WHERE podmínka]
[GROUP BY typ_skupiny]
[HAVING where_definice]
[ORDER BY atribut]

Klausule v hranatých závorkách jsou volitelné. Místo seznamu jen některých atributů může být znak * nebo slovo ALL, který znamená všechny atributy, což je výhodné u velkého počtu atributů.

Př. SELECT * FROM tOsoba

Př. SELECT prijmeni, jmeno, datum FROM tOsoba

Tento příkaz vrátí příjmení, jméno a datum všech osob. Ve výsledku dotazu máme příjmení a jméno odděleno, zapsáno každé v jiném sloupci. Často potřebujeme jméno jako jednu položku. Takové zřetězení můžeme udělat již v dotazu na databázi.

Př. SELECT prijmeni & ' ' & jmeno AS JmenoOsoby, datum FROM tOsoba

Příjmení a jméno je nyní spojeno v jednom sloupci, který dostal jméno JmenoOsoby pomocí aliasu – fráze:

AS JmenoOsoby.

Pokud uživateli předkládáme nějaký jmenný seznam, bývá dobrým zvykem jej předkládat setříděný podle abecedy. To se zajistí frází

ORDER BY atribut1, atribut2, …

Př. SELECT prijmeni, jmeno, datum FROM tOsoba

ORDER BY prijmeni, jmeno

Tento příkaz vrátí příjmení, jméno a datum všech osob seřazené abecedně podle příjmení a jména. Má-li být třídění opačné Z-A, přidává se ještě klíčové slovo DESC. Implicitní řazení je ASC (ascending).

Vložení klíčového slova **DISTINCT** způsobí, že se vrátí tabulka, ve které se neopakují záznamy, výsledkem je relace.

Př. SELECT DISTINCT jmeno FROM tOsoba

Příkaz vrátí tabulku křestních jmen vždy s pouze jedním výskytem jednoho konkrétního křestního jména. I když v databázi bude 20 osob s křestním jménem Jana, bude ve výsledné tabulce jeden záznam s jménem Jana.

Podmínka WHERE

Podmínkou WHERE se provádí zúžení výběru na záznamy vyhovující podmínce. Realizuje se jí operace selekce a spojení

Př. SELECT * FROM tOsoba WHERE cislo='101'

Př. SELECT * FROM tOsoba WHERE datum>'31.12.1960' AND datum<'1.1.1965' ORDER BY datum

Tento příkaz vrátí všechny osoby narozené mezi 31.12.1960 a 1.1.1965 seřazené podle data narození.

Při testování podmínkou WHERE mohou být používány následující operátory:

relační operátory : = <> < <= > >=

logické operátory: NOT AND OR

další operátory : BETWEEN dolní mez AND horní mez (meze jsou zahrnuty)

IN (seznam_prvků_množiny)

IS NULL

LIKE vzor - pro porovnání řetězců podobně jako u hvězdičkové konvence:

% odpovídá skupině znaků

_ podtržítko zastupuje jeden znak

Pozor! Chceme-li získat všechny záznamy, které mají v určitém atributu hodnotu NULL, musí podmínka mít tvar
WHERE vaha IS NULL nikoliv chybně WHERE vaha=NULL

Uvědomme si tuto neobvyklou možnost, že lze vyhledat i záznamy, které mají v některé položce hodnotu NULL.

Grupovací funkce

Tabulku můžeme uspořádat tak, že vzniknou skupiny řádků se stejnou hodnotou třídícího klíče. Také pro tyto skupiny můžeme provádět operace (částečné počty, součty ap.). Vytvoření skupin se provede klauzulí

GROUP BY atribut

Př. SELECT vyska
FROM tOsoba
GROUP BY vyska

Agregační funkce

V SQL jsou k dispozici tyto agregační funkce COUNT, SUM, MAX, MIN, AVG a další.

Funkce **COUNT** na celou tabulku počítá záznamy včetně jejich duplicit.

Př. SELECT COUNT(vyska) AS PocetOsob FROM tOsoba

Př. SELECT COUNT(cislo) AS PocetOsob FROM tOsoba

Tyto dva selecty nemusí vrátit stejnou hodnotu pro tutéž tabulku, neboť atribut vaha může mít někde hodnotu NULL. Atribut cislo je primární klíč a tudíž není přípustný výskyt hodnoty NULL. Funkce Count() a všechny další podobné ignorují, přeskakují NULL hodnoty.

U tohoto příkazu musíme znát název některé položky a ještě by neměla mít žádnou hodnotu NULL. Existuje naštěstí varianta count(*), která vrací počet všech záznamů v tabulce, aniž uvedeme jakoukoli položku.

Př. SELECT COUNT(*) AS PocetOsob FROM tOsoba

Př. SELECT COUNT(vyska) AS PocetOsobNad180
FROM tOsoba
WHERE vyska > 180

Výsledkem je tabulka s jedním nově pojmenovaným sloupcem PocetOsob jedním číslem udávající počet osob větších než 180 cm.

Př. SELECT Avg (vyska) AS AvgOfVyska,
Min(vyska) AS MinOfVyska,
Max(vyska) AS MaxOfVyska
FROM tOsoba

Tento příkaz vrátí tři spočítané hodnoty (průměrnou výšku, minimální výšku, maximální výšku) AvgOfVyska, MinOfVyska a MaxOfVyska, které byly spočítány z dat v tabulce.

Agregační a grupovací funkce lze spojit, jak ukazuje následující příklad.

Př. SELECT vyska, COUNT(vyska) AS PocetOsob
FROM tOsoba
GROUP BY vyska
ORDER BY vyska

Výsledkem je tabulka s pojmenovaným sloupcem PocetOsob a sloupcem s hodnotami výšek, která udává kolik osob má danou výšku. Např.

vyska	PocetOsob
170	2
175	1
180	4

Tab 20 Výsledek příkazu SELECT s grupovací a agregační funkcí

Pokud pracujeme se skupinami a chceme formulovat podmínku pro celou skupinu, nejen pro jednotlivé řádky původních tabulek, nedává se tato podmínka za WHERE, ale za HAVING:

```
SELECT {seznam_atributů | *}
    FROM seznam_tabulek
    [WHERE podm]
    [GROUP BY seznam-atributů
    [HAVING podm_pro_skupinu] ]
```

Zapamatujte si, že podmínka WHERE omezuje výběr záznamů z databáze. Použijeme-li agregace (sdružování, grupování - vysvětleno v předchozím textu) záznamů, můžeme výslednou tabulku omezit podmínkou ve frázi HAVING.

Př.
```
SELECT vyska, Count(vyska) AS CountOfVyska
    FROM tOsoba
    GROUP BY Vyska
    HAVING (Count(tOsoba.Vyska)>2)
```

Zjistili jsme všechny výšky osob, kde počet všech osob dosahující tuto výšku je větší než 2. Výsledek z předešlé tabulky je tedy následující:

vyska	PocetOsob
180	4

Výběr z více tabulek

Zajímavější dotazy se týkají případů, kdy jsou data do výsledku kombinována z více tabulek. Potom se tečkovou notací udává název tabulky a název atributu. Před tečkou je název tabulky, za tečkuje název atributu.

Př.
```
SELECT tOsoba.prijmeni, tOsoba.jmeno, tStav.stav
FROM tStav, tOsoba ON tOsoba.ID_Stav = tStav.ID_Stav
WHERE ((tStav.Stav) Like "vdaná")
ORDER BY tOsoba.prijmeni, tOsoba.Jmeno
```

Tento příkaz vrátí jména, příjmení a stav všech osob, které mají stav vdaná seřazené podle příjmení a jména.

Dlouhé názvy tabulek, které jsme nuceni nyní psát pro kvalifikaci atributů si můžeme zkrátit pomocí alias takto:

Př.
```
SELECT O.prijmeni, O.jmeno, S.stav
FROM tStav AS O, tOsoba AS S ON O.ID_Stav = S.ID_Stav
WHERE ((S.Stav) Like "vdaná")
ORDER BY O.prijmeni, O.Jmeno
```

Způsob spojení tabulek lze definovat klíčovými slovy: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN.

Vnitřní spojení

```
SELECT seznam_atributů
FROM první_tabulka
INNER JOIN druhá_tabulka
ON podmínka
```

Vnitřní spojení tabulek je takové, kdy se spojí záznamy podle společné položky a to jen tehdy, když hodnota položky existuje v obou tabulkách.

Vnější spojení

```
SELECT seznam
FROM levá_tabulka
LEFT JOIN pravá_tabulka
ON podmínka
```

Levé vnější spojení tabulek je takové, kdy se vezmou všechny záznamy levé tabulky a k nim se vyhledají záznamy z pravé tabulky podle společné položky. Existuje-li společná hodnota, bude záznam připojen; neexistuje-li, bude záznam nahrazen hodnotou NULL.

RIGHT JOIN jen změní směr pohledu na tabulky.

Plné spojení

```
SELECT seznam
FROM levá_tabulka
FULL JOIN pravá_tabulka
ON podmínka
```

Plné vnější spojení tabulek je takové, kdy se vezmou všechny záznamy levé tabulky a k nim se připojí všechny záznamy z pravé tabulky. Existuje-li společná hodnota společné položky, budou záznamy připojeny; neexistuje-li, budou chybějící položky ve společném záznamu nahrazeny hodnotou NULL.

Křížové spojení

```
SELECT seznam
FROM jedna_tabulka
CROSS JOIN druhá_tabulka
```

Křížové spojení je takové, kdy se spolu spojí všechny záznamy jedné tabulky se všemi záznamy druhé tabulky. Nemá žádnou spojovací podmínku. Množinově bychom mluvili o kartézském součinu. Má-li první tabulka m záznamů a druhá tabulka n záznamů, má výsledná tabulka m krát n záznamů.

Poddotazy

Příkazy SELECT lze vnořovat i ve více úrovních. Lze je řetězit a pro formulaci hlavního dotazu je možno použít výsledků jiného dotazu tzv. poddotazu.

Př. `SELECT * FROM tOsoba WHERE OsobCis NOT IN (SELECT OsobCis FROM tDite )`

Tento dotaz vybere všechny osoby, které nemají děti v tabulce tDite. Vrací se relace o n-řádcích a proto se porovnává operátorem IN.

Výsledkem poddotazu může obecně být jediná hodnota (relace o 1 řádku a 1 sloupci), nebo n-tice hodnot (relace o 1 řádku), nebo relace o n-řádcíh (a 1 sloupci).

Poddotazy lze použít i za klausulí WHERE v příkazech UPDATE a DELETE.

7.6. SQL příkazy pro definici přístupových práv

Příkazy pro přidělení práv jsou dva: GRANT, který uděluje práva a REVOKE, který práva odnímá.

Práva pro uživatele se vlastně vztahují k určitým příkazům SQL a určují, zda je uživatel může nebo nesmí provádět. Nejčastější práva jsou uvedena v následující tabulce:

Právo	Aplikuje se na	Povoluje uživateli
SELECT	tabulky, sloupce	vybírat záznamy z tabulky
INSERT	tabulky, sloupce	vkládat nové záznamy do tabulky
UPDATE	tabulky, sloupce	upravovat hodnoty v záznamech tabulky
DELETE	tabulky	mazat záznamy v tabulce
INDEX	tabulky	vytvářet a odstraňovat indexy tabulek
ALTER	tabulky	upravovat strukturu tabulky (přidávat, přejmenovávat atributy, měnit jejich typ)
CREATE	databáze, tabulky	vytvářet nové databáze a tabulky
DROP	databáze, tabulky	odstranit databáze nebo tabulky

Tab 21 Seznam práv

Obecná forma příkazu je:

```
GRANT [seznam_práv]  [atributy]
        ON jméno_objektu
        TO {PUBLIC |uživatel1 [, uživatel2,…]}
```

Př. Výběr z tabulky tVyrobek povolen všem uživatelům

```
GRANT SELECT
        ON tVyrobek
        TO PUBLIC
```

Př. Právo mazat a měnit obsah atributu CenaVyrobku v tabulce tVyrobek povolen uživateli Skladnik

```
GRANT DELETE, UPDATE (CenaVyrobku)
        ON tVyrobek
        TO Skladnik
```

Příkaz REVOKE je přímým opakem příkazu GRANT. Je používán k odebírání práv uživateli.

Jeho syntaxe je

```
REVOKE seznam_prav [(sloupce)]
        ON jmeno_objektu
        FROM uzivatel
```

7.7. SQL příkazy pro transakce

Transakce nám zajišťují kontinuitu a nedělitelnost změn na základě principu "Všechno, nebo nic!". Klasický příklad představuje převod finanční částky z jednoho účtu na druhý. Pokud klient A převádí finanční částku na účet klienta B, musí se daná částka nejprve odepsat z účtu klienta A a následně připsat na účet klienta B. Je to triviální operace, ale jen za předpokladu, že vše proběhlo bez problémů. Představme si situaci, že klient B svůj účet již zrušil nebo klient A se spletl při zadávání čísla účtu klienta B. Při těchto (a dalších) problémech se částka z účtu klienta A sice odepíše, nicméně není ji možné připsat na účet klienta B. Naštěstí se nic neděje, protože SŘBD takovou transakci stornuje a peníze se bezpečně vrátí na účet klienta A.

Podobně jsou ošetřeny i chybové stavy, kdy dojde k technickým výpadkům a nekorektnímu uzavření databáze. Taková transakce tedy proběhne buď jako celek, což znamená, že peníze se z účtu odesilatele odepíší a na účet příjemce připíší, nebo se transakce jako celek zruší.

Průběh transakce řídíme SQL příkazy SAVE POINT, COMMIT a ROLLBACK.

Příkazem SAVE POINT uložíme aktuální stav dat v databázi. Příkazem SAVE POINT název označujeme též místo pro případný návrat, příkazem COMMIT potvrzujeme platnost transakce a příkaz ROLLBACK [TO SAVEPOINT název] zruší všechny provedené změny, které byly v rámci aktivní transakce doposud provedeny.

Opatrný programátor by zdražení všech položek (kterých může být třeba několik tisíc) měl provést pro jistotu takto:

```
SAVE POINT pro_jistotu
UPDATE zbozi SET cena = cena * 1.05
```

Jiný příklad, kde se musí ošetřit transakce, je případ vložení nového majitele do katastru nemovitostí a jeho vlastnického vztahu. Vloží se nový záznam o majiteli do jedné tabulky a další nové údaje do tabulky o vlastnictví nemovitosti. Když se jedna z těchto operací neprovede, musíme odvolat i tu, která se provedla. Zůstal by pak pouze třeba zaznamenán jen nový majitel, ale údaj o nemovitosti, kterou vlastní, by chyběl.

7.8. Řazení, filtry a dotazy v MS Access

Operace pro manipulaci s daty a výběr dat z databáze jsou realizovány v MS Access tak, jak je uvedeno v následujícím textu. Jde o ukázku jazyka QBE.

Filtry

Při práci s tabulkou je často třeba seřadit všechny záznamy podle určitého kritéria (podle určitého atributu). Tento požadavek patří k základním operacím s databázemi.

Access má k dispozici z menu volbu **Záznamy/ Seřadit** a dále vybereme **Seřadit vzestupně** či **Seřadit sestupně**. Řazení se provádí podle sloupce, který jsme vybrali. Při řazení se přesunují celé řádky. Takto řadit lze pouze podle *jednoho* sloupce. Poslední řazení se uloží s objektem tabulky a uplatní se při následujícím zobrazení.

Vedle řazení je častou operací **filtrování** záznamů, tj. zobrazení záznamů, které splňují určitou podmínku. Provádíme operaci **selekce** nad relací. V MS Accessu je možné realizovat tuto operaci dvěma způsoby: filtrací a dotazem. Jednodušší operací je **filtrace**. Možnosti filtrace jsou *Filtrace podle výběru*, *Rozšířený filtr nebo řazení* a *Filtr podle formuláře*. Velkou nevýhodou filtru je, že nelze uložit natrvalo.

POZOR! Informace z více tabulek lze zobrazit současně pouze strukturovaným dotazem nikoliv prostým filtrováním.

Dotazy

Nejmocnějším prostředkem k získávání informací z tabulek databáze a zároveň prostředkem k dodatečné změně těchto tabulek jsou **dotazy**. Výhodou dotazů oproti filtrům je, že mohou být uloženy trvale a tudíž se mohou opakovaně použít a nemusí se znovu sestavovat (Kořínek, 2000).

Dotaz je vlastně příkazem specifikujícím práci (výběr, modifikace) s daty, data samotná dotaz neobsahuje. Pracuje vždy s aktuálními daty tabulek či vybraných dotazů a jím získané výsledky mohou být použity jako základ pro tvorbu dalších dotazů, tabulek, formulářů a sestav. Dotazy lze sestavovat v návrhovém zobrazení (jedná se o QBE) nebo přímo v editoru SQL příkazů. Mezi zobrazením lze přepínat.

Rozeznáváme dva základní typy dotazů.

1. Pasivní neboli **výběrové dotazy**, které pouze vybírají a zobrazují data, patří sem i **agregační** dotazy
2. Aktivní neboli **akční dotazy** – provádějí změny hodnot atributů, vytváří nové tabulky

Je možné vytvářet též **parametrické** dotazy. Vstupní hodnota pro podmínku výběru se udává jako volitelný parametr. Po spuštění takového dotazu se objeví výzva na vložení požadované hodnoty parametru.

Výběrové dotazy

Výběrovým dotazem provádíme relační operaci selekce nebo projekce (někdy i obě zároveň).

Příklad: vyber všechny zaměstnance určitého příjmení, seřaď zaměstnance podle data narození, vyber jen zaměstnance určitého oddělení atd.

Agregační dotazy zjišťují počty, průměry, maxima, minima atd. záznamů, vyhovujících určitému kritériu. Viz. příklady SQL agregačních dotazů.

Akční dotazy

Mezi akční dotazy patří

- ***aktualizační dotaz*** – mění hodnoty některých atributů (zvýší číslo ročníku studenta, změní cenu výrobku,…)
- ***vytvářecí dotaz*** – vytváří novou tabulku ve stejné nebo jiné databázi
- ***přidávací dotaz*** – přidá nové záznamy nebo přidá určité záznamy z jedné tabulky do druhé tabulky
- ***odstraňovací dotaz*** – odstraňuje záznamy podle kriteria (absolvent ukončil školu, atd.)

Příklad: V evidenci studentů na začátku školního roku použijeme *aktualizační dotaz* na zvýšení čísla ročníku o 1 (studenti postoupili o ročník výš). Studenty absolventy *přidávacím dotazem* přidáme z tabulky řádných studentů do jiné archivní tabulky absolventů a nakonec je *odstraňovacím dotazem* z tabulky řádných studentů odstraníme.

Příklad:

Pro zdražení ceny všech výrobků o 20% sestavíme *aktualizační dotaz*, kde cenu násobíme koeficientem 1,2.

7.9. Jazyk SQL pro prostorová data

V současné době směřuje vývoj geografických informačních systémů do stavu, kdy je snaha ukládat do databází společně jak atributová tak geometrická data. GIS nejsou jediné aplikace, které využívají prostorové objekty. Při počítačovém konstruování strojů, stavebních objektů či elektronických obvodů jsou také vytvářeny prostorové objekty. Pouze použitá měřítka jsou různá. Stále častější je potřeba ukládat i obrazová data.

Použití standardních SŘBD pro uložení prostorových dat je velice obtížné, protože tyto jazyky neposkytují pro prostorová data žádnou podporu. Jsou těsně svázány s vlastnostmi lexikálních dat, pro která byly původně navrženy.

I přes tyto těžkosti existuje mnoho pokusů aplikovat existující SŘBD na prostorová data. Pro uložení geometrických dat se používá datový typ BLOB (binary large object), což je nestrukturovaný rozsáhlý objekt. Důsledkem tohoto omezení je skutečnost, že uživatel musí přesně znát strukturu, ve které se prostorová data ukládají. Je to nešikovné a nekonzistentní. Znalosti o uložení jsou v nadstavbové aplikaci a ta zprostředkovává jejich ukládání a načítání.

Situace se změnila k lepšímu, když oficiální standardizační skupina v roce 1999 ISO/IEC/JTC1/SC32WG4 rozšířila standard SQL a vytvořila novou verzi umožňující zacházet se složitými a netabulkovými daty. Vznikl SQL/MultiMedia označovaný jako SQL/MM. SQL/MM se skládá z pěti částí, zahrnujících kromě Prostorových objektů (Spatial) např. také úplné texty (Full Text). V tomtéž roce byl také uveden standard SQL1999 (původně SQL3), který do relačního prostředí vnesl objektovou orientaci. Pro implementaci prostorových objektů se tak otevřely další možnosti.

Základní ideou je uložit geometrické objekty do tabulek. Atribut obsahující geometrická data se nazývá prostorový atribut a tabulka s prostorovým atributem se nazývá prostorová tabulka. Prostorové tabulky mají jiné vlastnosti než konvenční tabulky, neboť odkazují na topologické a metrické vztahy příslušných objektů. Nad prostorovými daty jsou k dispozici speciální operace.
Prostorové typy dat včetně odpovídajících operací a funkcí jsou integrovány do relačního SŘBD. Toho lze dosáhnout různým způsobem, přičemž zřejmě nejpokročilejším řešením jsou dnes moduly připojené k objektově relačnímu DBMS (ORDBMS).
Rozšíření relační databáze o **prostorové typy dat** znamená přidat nové datové struktury a odpovídající nové operace. Důležitá je také podpora speciálních indexových struktur.

Moduly pro řízení prostorových objektů, jako jsou tzv. cartridges u SŘBD ORACLE, DataBlades u Informix a extenders u DB2, jdou ještě dále. Funkce modelující chování objektů jsou vnořeny do SŘBD, což umožňuje lepší optimalizaci dotazů. Využívají se uživatelem definovaných typech (UDT) a funkcích (UDF), které mají obecnější použití v SŘBD, nejen pro geometrické struktury. Mechanismus jejich vytváření je součástí jazyka (v daném případě SQL).

V původním SQL3 se používá pro UDT a UDF univerzálnější pojem abstraktního datového typu - ADT. ADT zapouzdřuje data a funkce neboli metody nad těmito daty. Relační DBMS poskytuje jednak nové konstruktory pro tvorbu datových typů (např. ARRAY), jednak konstrukt CREATE TYPE pro vytváření ADT. Pomocí polí lze snadno ukládat souřadnice prostorových objektů. Odkazováním pomocí typu REF lze objekty, a tedy i prostorové objekty, vázat do složitějších prostorových objektů.

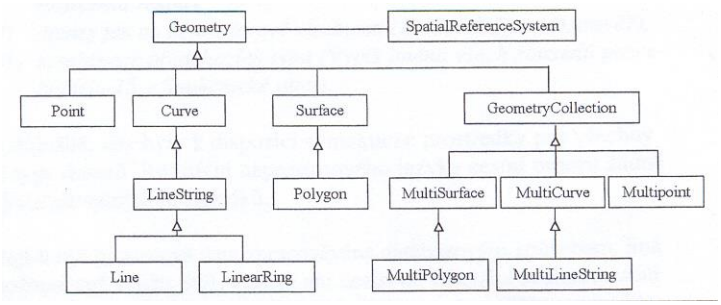
Taktéž konsorcium OpenGIS (dále OGC) definovalo standardní SQL schéma (OGC99) pro uložení, výběr, dotazování a aktualizaci kolekcí jednoduchých prostorových objektů. OGC udržovalo úzký vztah se skupinou WG4 a docílilo tak již ve verzi z r. 1998 kompatibility s jádrem části standardu SQL/MM-Spatial.

Nespornou výhodou uložení veškerých prostorových informací do relačních SŘBD je využití mechanismů transakčního zpracování, nastavování práv, přístupů a zabezpečení integrity. SŘBD jsou určeny pro práci s velkými objemy dat (v řádu teraBytů).

Geometrický model

OGC popisuje prostorové objekty **geometrickým modelem** založeným na hierarchii tříd.
Třída Geometry je abstraktní třída geometrických objektů. Každý geometrický objekt je asociován se systémem prostorových referencí (souřadnic). Mezi jednodimenzionální geometrické objekty patří body a křivky. Dvoudimenzionální jsou povrchy. Kolekce je vždy míněna jako multimnožina, tj. např. v kolekci bodů mohou být dva body stejné. Křivka je dána posloupností bodů, přičemž podtřídy třídy *Curve* určují způsob interpolace. V případě linie (lomená čára) jde o lineární interpolaci. Objekty třídy *Line* (úsečka) jsou dány dvěma body. Křivka je jedno-

duchá, jestliže se sama se sebou nikde neprotíná. Uzavřené křivky mají společný počáteční a koncový bod. Třída *LinearRing* obsahuje pouze jednoduché a uzavřené linie. Polygony jsou, stručně řečeno, definovány vnější hranicí a několika vnitřními hranicemi, které tvoří "díry". Tyto hranice (boundary) jsou dány pomocí objektů ze třídy *LinearRing*.



Obr. 27 Třídy

Pro jednotlivé třídy jsou specifikovány metody. Na úrovni Geometry to jsou základní metody, např. Dimension, Boundary, a metody pro porovnání objektů Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps, Relate. Výsledkem metody pro porovnání je TRUE v případě, že výsledek porovnání je shodný.

Metody pro porovnávání objektů jsou založeny na binárních prostorových predikátech. Další metody podporují prostorovou analýzu. Patří sem Distance, Buffer, ConvexHull, Intersection, Union, Difference a SymDifference. Metoda Envelope pro daný geometrický objekt (v 2D) vrací jeho minimální ohraničující obdélník (MOO).

Použití abstraktních datových typů pro prostorové objekty

V objektově relačním SQL:1999 existují dva hlavní rysy objektového rozšíření relačního modelu dat:

- abstraktní datové typy - ADT,
- typy řádků.

Uživatel může formulovat typy a funkce pro řádky tabulek. Ty mohou být základního typu, ADT, typu REF (odkazy), nebo typu, který modeluje vícehodnotové atributy. Ty lze zatím realizovat pouze jako pole hodnot (ARRAY). Současné implementace ORDBMS však již nabízejí i množinu (SET) či multimnožinu (BAG).

Tabulku lze definovat pomocí příkazu CREATE TABLE buď přímo výčtem atributů (viz příklad 1) nebo se nejprve vytvoří typ řádku (pomocí příkazu CREATE ROW TYPE T) a pak tabulka jako CREATE TABLE R OF T.

Příklad 1: Příkazem CREATE TABLE je vytvořena tabulka ZAHRADA.

```
CREATE TABLE ZAHRADA (
        JMENO       VARCHAR(30),
        ROZLOHA     INTEGER,
        SKLENIKY    VARCHAR(30) ARRAY[10],
        UMISTENI    ST_GEOMETRY)
```

Skleníky jsou modelovány polem řetězců (jmen skleníků), UMISTENI je prostorový atribut, jehož typ je z vrcholu hierarchie geometrických typů. Předpona ST_je dána standardem (označují se s ní UDT). Atribut UMISTENI je typu Point s určením X a Y souřadnice. Vložení řádku do tabulky ZAHRADA je realizováno následujícím programem.

```
BEGIN
    DECLARE umisteni_zahrady ST_POINT;
    SET umisteni_zahrady  =
    NEW ST_POINT(100.0, 200.0)
    INSERT INTO ZAHRADA VALUES (
        'Botanická zahrada'.
        32400,
        ARRAY ['Palmový skleník'],
        umisteni_zahrady)
END
```

Dotazy na geometrické vlastnosti objektů umístěné v tabulkách se formulují pomocí běžných prostředků SQL spolu s využitím metod.

Dotaz 1. Najdi x-souřadnici zahrady ‘Botanická zahrada‘.

```
SELECT UMISTENI.X
    FROM  ZAHRADA
    WHERE JMENO = ‘Botanická zahrada ‘
```

Příklad 2: Nechť abstraktní datový typ Point má implementované funkce pro vstup konstant typu Point a výstup výsledků typu Point. Nechť je k dispozici funkce pro testování sousedství dvou bodů JeVSousedstvi. V tabulce STUDNA je atribut UMISTENI typu Point a atribut HLOUBKA studny. Nyní lze formulovat dotaz:

Dotaz 2: Najdi hloubku studny spolu s umístěním studny, v jejichž sousedství žije Novák.

```
SELECT S.HLOUBKA, 0.UMISTENI
FROM STUDNA AS S, OSOBY  0
WHERE JeVSousedstvi (S.UMÍSTĚNÍ, 0.UMÍSTĚNÍ)
AND 0.JMENO = 'Novák'
```

Uvažujme dále umístění měst modelovaná pomocí polygonů. Pro polygony předpokládejme metodu vzdalenost (dvou bodů) a centroid (vrací těžiště polygonu).

Dotaz 3: Najdi jména osob, která bydlí ve vzdálenosti od centra Ostravy menší než 25 km.

```
SELECT O.JMENO
FROM OSOBY AS 0, MESTA AS M
WHERE M.JMENO = 'Ostrava'
AND vzdalenost(O.Umisteni, M.Umisteni.centroid) < 25
```

Potřebné je i modelování vývoje prostorových atributů v čase. To vede i ke změnám prostorových vztahů prostorových objektů. Pohybující se body či měnící se regiony lze chápat jako časoprostorové objekty. Existuje návrh jazyka STSQL (Spatio-Temporal SQL), který umožňuje formulovat dotazy na časové a prostorové souvislosti.

8. Metadata

Metadata jsou určité charakteristiky, atributy, které slouží k popisu jiných dat.

Obecně jsou metadata také data a mohou mít své vlastní charakteristiky (atributy nebo též deskriptory) a tak vznikají vyšší úrovně metadat.

Rozdíl mezi daty a metadaty je v jejich *použití*.

Metadata je možné ukládat nebo vybírat z archivu (repozitory) podobně jako ukládáme data do databází. Tyto popisy jsou často uloženy odděleně od vlastních dat. Příkladem je v DBMS popis jednotlivých datových položek databázových tabulek.

K čemu slouží metadata?

Metadata jsou data, která mají za úkol především popsat data. Slouží pro přesnou a korektní identifikaci a interpretaci dat.

Metadata pro prostorová data by měla být organizována a spravována s využitím **metainformačních systémů**. Metainformační systém tak podporuje efektivní využívání samotných dat. Důležitou složkou metadat kromě popisu vlastních dat bývají i údaje o kontaktním místě (popř. osobě).

8.1. Slovník dat

Metapopis struktur databáze nese slovník dat. Slovník dat (data dictionary) popisuje strukturu všech objektů uložených v databázi. Jsou zde uloženy informace o existujících tabulkách a jejich sloupcích, popisy omezení pro data v tabulkách, popis definovaných indexů atd.

Jednou z podmínek relačních databázových systémů je, že i tyto informace jsou uloženy v relačních tabulkách a jsou dosažitelné pomocí jazyka SQL. V každém databázovém systému se proto můžeme setkat s různým počtem tzv. systémových tabulek. Zjistit jejich obsah lze pomocí příkazu SELECT. Systémové tabulky lze číst, ale je zakázáno měnit přímo jejich obsah příkazy INSERT, UPDATE, DELETE. Názvy tabulek začínají ve většině případů předponou SYS. Názvy systémových tabulek a jejich popis je součástí dokumentace k příslušnému databázovému prostředí.

8.2. Příklad metadat

Velice důležitou složkou dat (samozřejmě i prostorových) je jejich popis. Tento popis označovaný obvykle jako metadata je často podceňován a opomíjen. Uvažme následující data:

(1, 100, 150, 0.0064),

(2, 120, 145, 0.0044), atd.

Tato data nám byla dodána panem Dvořákem a jsou důležitá pro naši práci. Pan Dvořák předtím než odjel na dovolenou (kde není k zastižení) sdělil pouze, že tato data dostal od někoho z firmy Velmi Velká Firma, a.s. a, že jsou to koncentrace kadmia v podzemní vodě z vrtů v námi sledované oblasti. Po prohlédnutí takovýchto dat však zjistíme, že jsou pro nás naprosto bezcenná. Můžeme se dohadovat, že čísla 0.0064, 0.0044, atd. jsou koncentrace kadmia v mg/l (neboť taková koncentrace je v naší lokalitě očekávána). Dále můžeme čísla 1, 2 atd. vyhodnotit jako identifikátory jednotlivých vrtů. Čísla 100, 150, 120, 145, atd. by mohly být souřadnice daných vrtů. My však používáme souřadnicový systém S-JTSK Soubor obsahuje hodnoty z 1500 vrtů. My však evidujeme pouze 500 vrtů. Pro identifikaci vrtů používáme následující alfanumerické kódy: I-1, I-2, II-1, IV-4, atd. Data, která máme k dispozici, jsou tedy pro naši potřebu velmi špatně identifikovatelná, ne-li neidentifikovatelná a tudíž pro naši práci bezcenná.

Některý z následujících popisů dat (metadata) nám umožní původně bezcenná data využít.

Metadata1 : Položky jednotlivých záznamů: Identifikátor vrtu, souřadnice x v místním souřadnicovém systému, souřadnice y v místním souřadnicovém systému, koncentrace kadmia v mg/l. Místní souřadnicový systém má osy orientované stejně jako S-JTSK a počátek ve vrtu č. 8, který má souřadnice S-JTSK: x=1100303.22, y=455938.44. Měření byla prováděna v dubnu až září roku 2000.

Metadata2: Veškeré dostupné informace o daných datech má k dispozici pan Novák. Pan Novák je k dispozici na telefonním čísle 5861295472 nebo e-mail adrese novak@mojeposta.cz.

Metadata3: Položky jednotlivých záznamů: Identifikátor vrtu, souřadnice x v místním souřadnicovém systému, souřadnice y v místním souřadnicovém systému, koncentrace kadmia v mg/l. Místní souřadnicový systém má osy orientované stejně jako S-JTSK a počátek ve vrtu č. 8, který má souřadnice S-JTSK: x=1100303.22, y=455938.44. Měření byla prováděna v dubnu až září roku 2000. Veškeré další dostupné informace o daných datech má k dispozici pan Novák. Pan Novák je k dispozici na telefonním čísle 5861295472 nebo e-mail adrese novak@mojeposta.cz.

Zatímco *metadata1* nám umožní přímé využití dat (po několika úpravách), *metadata2* nám umožní kontaktovat osobu (kontaktní místo), která může, ale také nemusí poskytnout potřebné informace pro využití dat. Metadata2 však mohou umožnit získání jiných doprovodných informací o poskytnutých datech. Ještě lepší variantou je kombinace obou popisů do *metadat3*. (Růžička, 2002)

Tento stručný a názorný příklad poukazuje na několik skutečností, které se týkají metadat. Z příkladu vyplývá především to, že bez vhodných metadat jsou data často bezcenná nebo obtížně použitelná. Druhá část příkladu demonstruje to, že často tou nejdůležitější složkou metadat je aktuální kontakt na zodpovědnou osobu.

8.3. Standardizace metadat

V následujícím textu jsou vymezeny základní pojmy datová sada a metadatové hladiny. Vymezení těchto pojmů je nezbytné pro porozumění dalšímu textu, který se týká především oblasti standardizace metadat.

Datová sada

Data bývají obvykle sdružována do kolekcí, které jsou označována jako datové sady (data sets) nebo datové soubory (data files). Pojem "datová sada" je obecnější oproti pojmu "datový soubor". V dalším textu bude termín "datový soubor" představovat konkrétní soubor v souborovém systému paměťového média. Pojem "datová sada" bude představovat data tvořící logický celek v rámci určitého informačního systému či datové báze. Může se tedy jednat o jeden datový soubor či kolekci těchto souborů.

Pojem datová sada nebude omezován jen na digitální data, ale může jím být označena i analogová forma dat (např. papírová mapa, atlas map, tištěný seznam majitelů parcel v okrese Nový Jičín).

Metadatové hladiny

Metadata jsou doplňkovou informací k datům. Metadata stojí na rozhraní mezi daty a informacemi a jsou prostředníkem k interpretaci, verifikaci a užití dat. Metadata mohou být chápána na více úrovních a mohou a mohou plnit různý účel v interpretaci dat. Z hlediska teorie zpracování dat můžeme na data pohlížet na třech úrovních (Růžička, 2002):

1. **fyzická (orto)**

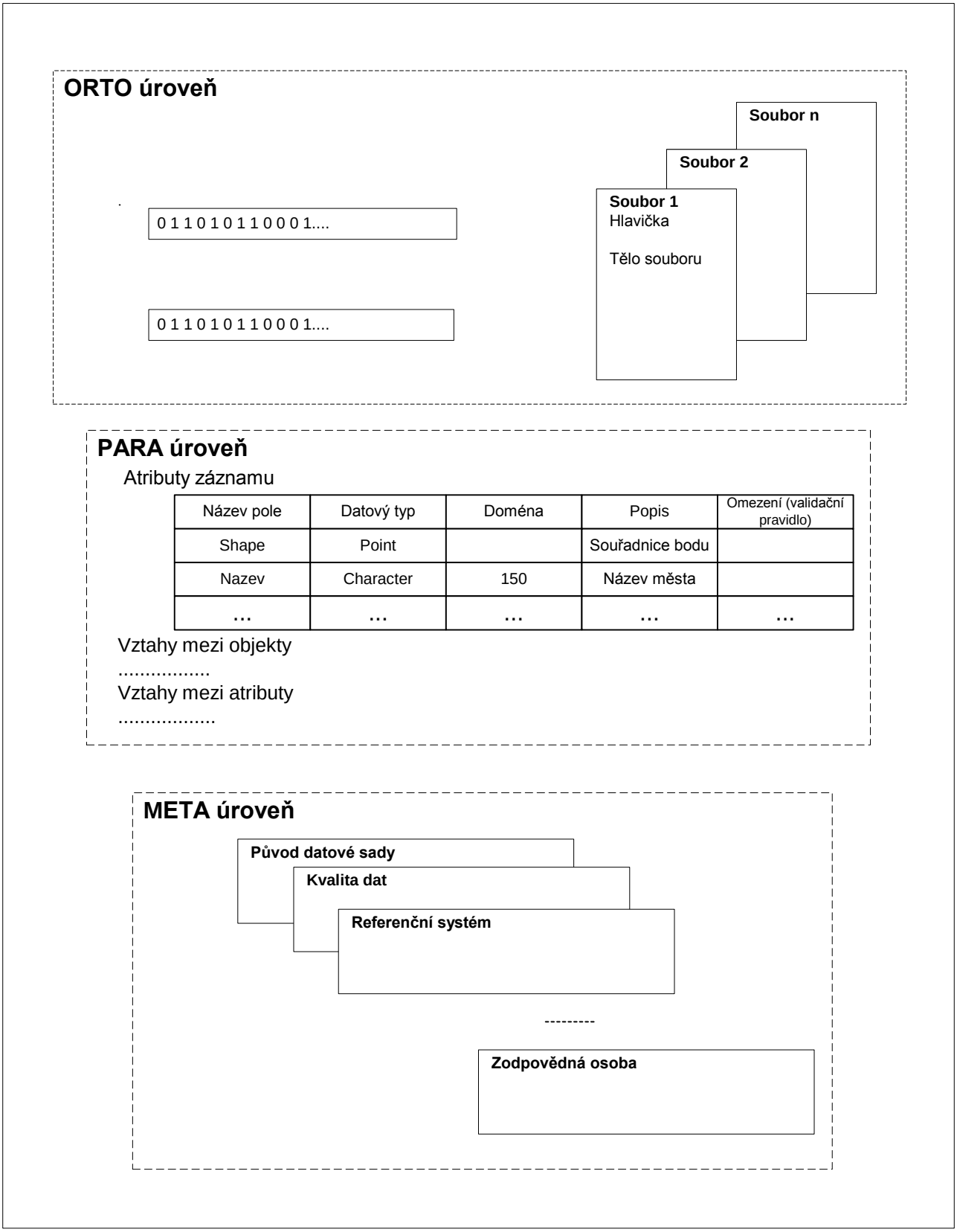
2. **logická (para)**

3. **meta**

Fyzická úroveň se zabývá tím, jak jsou data uspořádána na paměťovém médiu.

Logická úroveň se zabývá popisem obsahu dat. Hodnoty těchto vlastností jsou pak součástí samotných dat. Také zde bývají údaje o vztazích, integritní omezení.

Meta úroveň se zaměřuje na datový soubor jako celek. Meta úroveň se zabývá popisem typu datové sady, důvodem jejího vzniku, historií, kvalitou dat, identifikací datové sady v rámci nějakého systému, podmínkami šíření datové sady, vztahem osob a organizací k datům. Meta úroveň může obsahovat i prvky logické úrovně.



Obr. 28 Metadatové hladiny (Růžička, 2002)

8.4. MIDAS – Český on-line katalog geodat

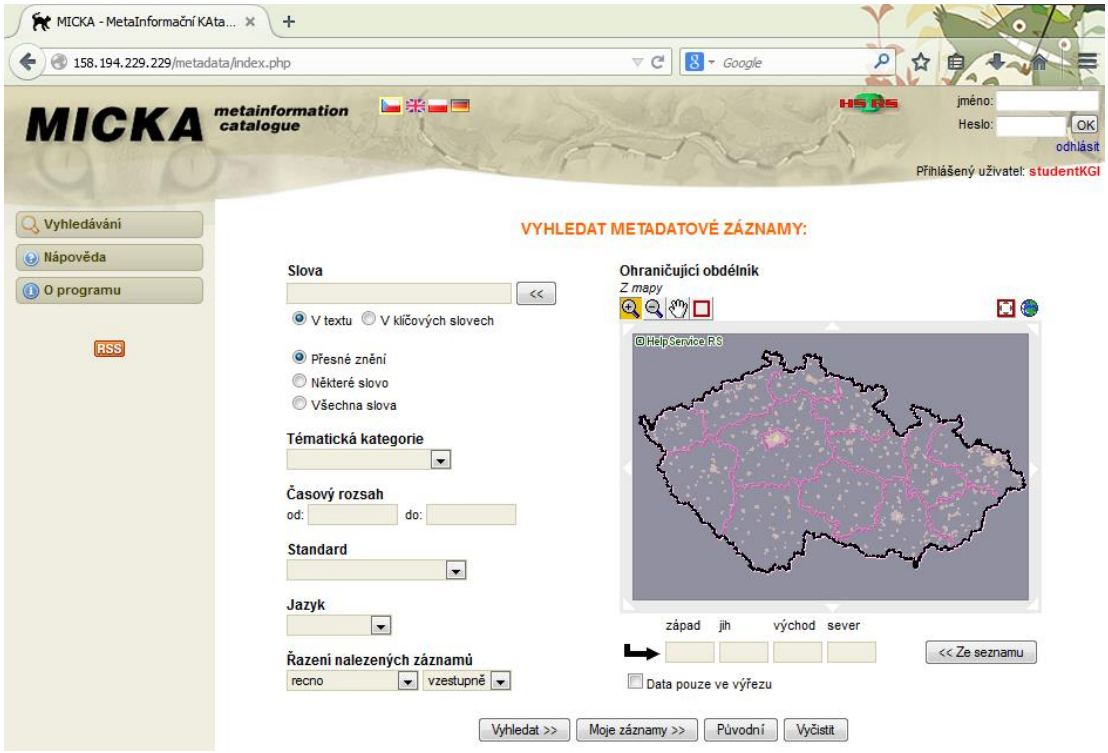
Midas byl metainformační katalog geodat v české veřejné správě. Z hlediska výše uvedené teorie obsahuje metadata nejvyšší úrovně (tj. meta úrovně). Byl vyvíjen Českou asociací pro geoinformace (CAGI) od roku 1999. Na vývoji se podíleli též pracovníci VŠB Ostrava Institutu geoinformatiky VŠB –TU Ostrava.

Obsahoval více než 3 tisíce datových sad o geodatech převážně vytvořených a používaných ve státní správě a samosprávě - zejména na okresních úřadech. Údaje o datových sadách se pořizovaly přímo na okresních úřadech pověřenými pracovníky. Se změnou organizace státní správy k 1.1.2003 (zrušení okresních úřadů, vznik krajských úřadů a pověřených obcí) byl nachystán projekt MIDAS-kraj. S nástupem standardů INSPIRE byl vývoj a používání ukončeno.

8.5. MICKA

MICKA je metainformační katalog dodávaný firmou Help Service Remote Sensing z Benešova. Pracuje s metadaty prostorových dat podle standardů ISO, OGC a INSPIRE.
Katedra geoinformatiky PřF UP v Olomouc provozuje vlastní katalog, kde shromažďuje metadata:

- o datech ze studentských prací - bakalářských, diplomových a disertačních,
- projektů řešených na katedře,
- zakoupených datech na katedru.



Obr. 28 Rozhraní Micky Katedry geoinformatiky PřF UP

9. Rozhraní ODBC

V této kapitole se budeme věnovat přístupu k datům uložených v různých databázích. Pro platformu Microsoft Windows (a dnes nejen pro ni) existuje univerzální rozhraní přístupu k datům - **ODBC** (Open Database Connectivity), které slouží jako mezičlánek pro spojení klientské aplikace a databázového systému.

Přístup prostřednictvím rozhraní ODBC je unifikovaný, což znamená, že ze strany klienta se k němu přistupuje stejným způsobem, nezávisle na tom, s jakým databázovým systémem pracujeme. Hlavní nevýhodou rozhraní ODBC byl nižší výkon ve srovnání s nativními ovladači. První verze ODBC tvořily jen jakýsi mezistupeň mezi aplikací a nativním ovladačem databázového systému, novější verze ODBC jsou již navržené tak, že přistupují k databázovému systému přímo, takže ztráta výkonu je ve srovnání s nativními ovladači takřka zanedbatelná. Pokud je aplikace dobře navržená, klient má na základě takřka okamžité odezvy dojem, že je vše jednoduché a bezproblémové. Přes rozhraní ODBC se naváže spojení s databázovým systémem. V tomto kroku už musí být požadavek na data převeden do podoby dotazu v jazyku SQL. Data z databázového systému putují ke klientovi nejčastěji ve formě tabulky. Omezující podmínkou použití ODBC je délka názvů na 25 znaků.

Kapitola vypuštěna.

10. CASE nástroje

10.1. Obecný popis

CASE nástroje (Computer Aided Software Engineering) jsou programy pro podporu návrhu a programování. Usnadňují práci a umožňují automatizovat zejména analýzu a návrh, někdy i implementaci konkrétního díla.

Nástroje CASE patří k poněkud skryté oblasti IT: rozhodně je nepotřebuje každý uživatel. Své uplatnění nacházejí při návrhu od malých až po velké informační systémy. Ve své oblasti jsou ovšem CASE nástroje nezastupitelné - od jisté velikosti bez nich informační systém dost dobře navrhnout nejde.

Pomocí *grafického návrhu* se navrhuje struktura systému, co má systém dělat s jakými daty. V tomto návrhu se nezabýváme způsobem, jak to bude uděláno. Velice často jsou CASE systémy použity pro navrhování databází. Označují se jako "Database modeler" či "Database designer".
V CASE nástroji je možné provádět datovou analýzu, datové modelování, návrh databází a funkční analýzu, návrh funkcí.
Dalším přínosem, kromě automatického generování struktur, je hlavně podpora strukturovaných analytických návrhových přístupů a také dále řízení verzí programů, automatická tvorba dokumentace o strukturách.

10.2. CASE Studio 2 CZ

CASE Studio patří do skupiny jednodušších nástrojů CASE. Pochází od české společnosti Charonware (starší verze od společnosti RK-Soft,). Informace jsou dostupné na www.casestudio.com. Podporována je tvorba ER (entitně relačních) a DF (data flow) diagramů. Podporuje verifikaci modelů ERD a DFD, také kontroluje oba modely mezi sebou navzájem a umožňuje dodržování konzistence celého návrhu.

CASE Studio 2 je program pro vizuální navrhování databázových struktur. Pracuje pod operačním systémem Windows.

Možnosti:

- Entitně relační diagramy (**ERD**)
- Data Flow Diagramy (**DFD**) (diagram datových toků)
- Reverse Engineering - umožňuje vytvořit model struktury již existující databáze
- Správce verzí - umožňuje porovnávat jednotlivé verze modelů
- Velice detailní logické i fyzické HTML reporty
- Galerie - pro uložení nejčastěji používaných částí modelů
- Podpora uživatelů, uživatelských skupin a uživatelských práv
- Uživatelsky definované šablony
- Možnost zapisování akcí do "To do" listu
- Datový slovník
- Referenční integrita (deklarativně i za pomocí triggerů)
- Podpora JScipt i VBScript
- Submodely

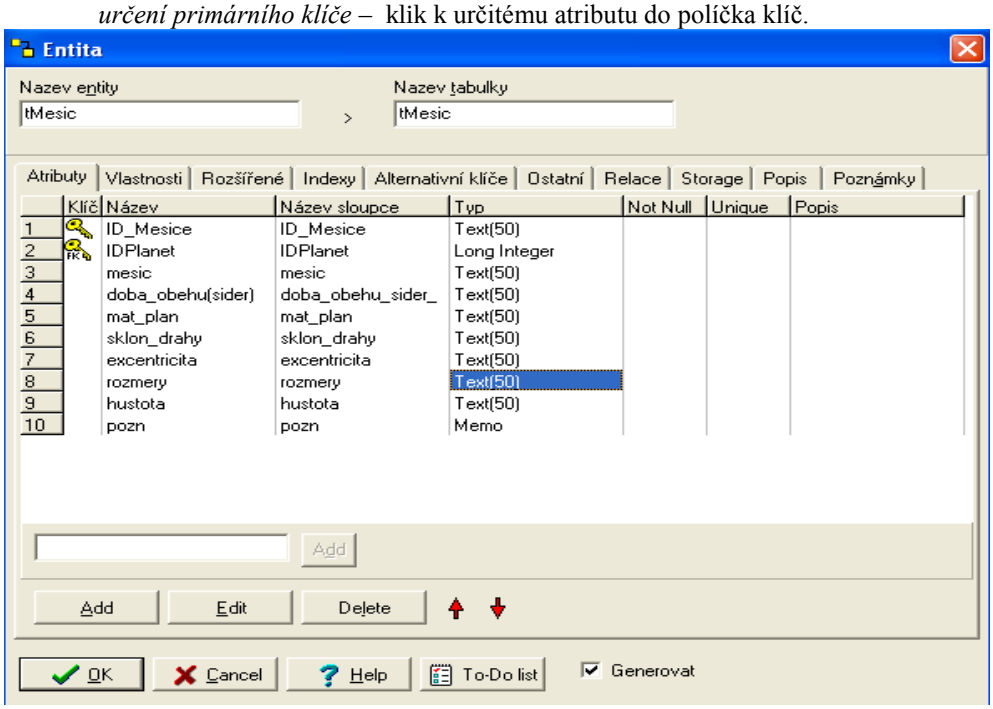
ER diagramy

Tvorba ER diagramů je v CASE Studiu jednoduchá. V plovoucím menu nástrojů si vyberete, co chcete vložit, a kliknutím myši vybraný prvek vložíte do modelu. Můžete si vybrat následující prvky: entity, identifikační relace, neidentifikační relace, relace M:N, informační relace (také self-relace), poznámku a razítko. V rámci jedné entity lze definovat atributy, indexy, alternativní klíče, relace a další doplňující vlastnosti. U vztahů můžeme určit název, typ, kardinalitu, parcialitu, způsob propojení, referenční integritu (none, restrict, cascade, set null, set default), klíče a další. Při složitějším návrhu je možné k celkovému modelu vytvořit více submodelů, které mohou být v hlavním modelu odlišeny barvou.

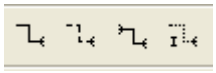
Prostředí

Entitu vytvoříme tlačítkem  . Dvojklikem na této vytvořené entitě určíme vlastnosti:

název entity
název atributů (typ atributu)
určení primárního klíče – klik k určitému atributu do políčka klíč.

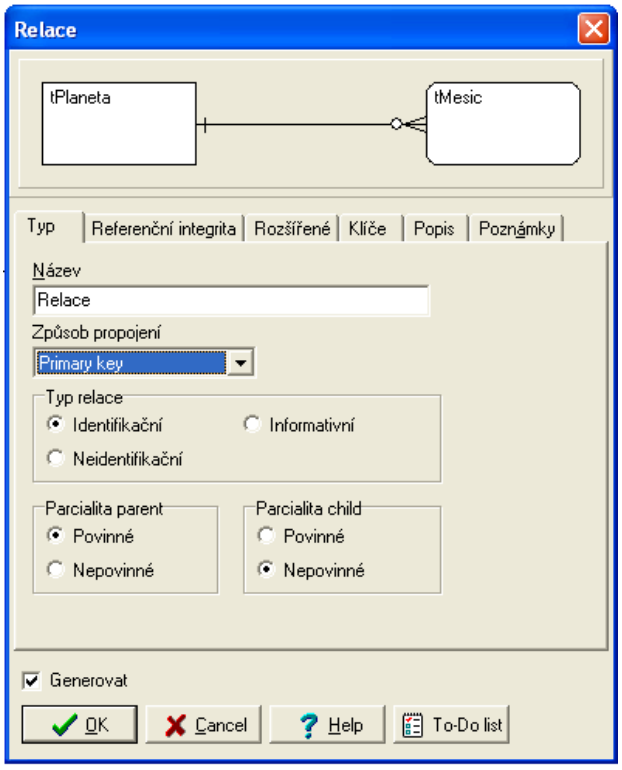


Obr. 38 Definice atributů entity

Takto můžeme nadefinovat více entit, mezi kterými poté budou nějaké vazby (relace). Tyto relace můžeme vytvořit pomocí tlačítek  , která znamenají:

Identifikační relace
Neidentifikační relace
Relace M:N
Informativní relace

Dvojklikem na relaci ji poté můžeme, stejně jako entitě, určit vlastnosti.

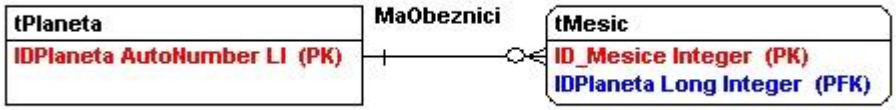


Obr. 39 Nastavení relací

Funkce zobrazení:

E – zobrazí pouze entity

P – zobrazí primární klíče



K – zobrazí všechny klíče

A – zobrazení všech atributů

F – zobrazení fyzických názvů s datovým typem atributu

I – zobrazení s indexy

DF diagramy

Vytváření DF diagramů je stejně jednoduché jako v případě ER diagramů.

K dispozici jsou čtyři prvky: data store, terminátor, proces, datový tok a řídící tok  . Návrh začíná kontextovým diagramem, který se zde nazývá Hlavní proces. Každý proces je možné detailněji popsat na nižší úrovni. Celkový diagram je zobrazen v levé části okna jako stromová hierarchická struktura. V celém DFD se lze díky stromu velice rychle orientovat a jedním kliknutím myši si můžete zobrazit požadovaný proces. Jednotlivé data story musí mít vazbu na entitu v ER diagramu. V DFD musíte dodržovat základní zásady jeho tvorby.

Generování skriptů

Důležitou funkcí CASE Studia je generování skriptů pro databáze. Ve stávající verzi je podporována většina dostupných databázových strojů jako Access 2000, Access 97, Clipper 5.0, DB2 v.7 (UDB), DB 2 v.8.1 (UDB), Informix, Ingres, Interbase 4-7, MS SQL2000, 6.5, 7, MySQL3.23,4, Oracle 9i, 8, 7, Paradox; Postgre SQL 7.3, Sybase Anywhere, ASE 12.5 a DBI-SAM 3. Pro připojení k databázím lze využít standardy ODBC, ADO, nativní připojení, popř. speciální připojení (např. BDE k databázi Interbase). Funkce umožňuje generovat domény, tabulky, primární klíče, indexy, referenční integritu, triggery z referenční integrity, uživatelské triggery, procedury, pohledy, alternativní klíče, role, práva rolí, práva uživatelů.

Tvorba reportů

CASE Studio poskytuje velmi schopnou dokumentaci k vytvořenému modelu. Dokáže vygenerovat reporty jak ER diagramů, tak i DF diagramů, a to buď ve formátu HTML, nebo RTF. Veškeré reporty jsou velmi podrobné. Reporty lze vytvářet jak pro celkový model, tak i pro jednotlivé submodely. Lze si jednoduše vytvořit i vlastní sestavy. Vzhled reportů HTML můžete upravit dle vlastních představ. Stačí vytvořit CSS soubor, ve kterém nadefinujete příslušné barvy, fonty, obrázky apod. Velkou výhodou je možnost vytváření vlastních jazykových verzí reportů za pomoci příslušných slovníkových šablon.

Další funkce

Z vytvořených databází lze zpětně vytvořit ER model. Funkcionalita se však omezuje jen na některé typy databází. Načtení entit, atributů, relací, indexů, triggerů, procedur a rovněž dalších objektů závisí právě na patřičné databázi. Ne všechny tyto položky jsou podporovány všemi databázemi. Občas potřebujeme uchovat část modelu, nebo třeba jenom jednu entitu někde stranou pro pozdější použití v jiných modelech. K tomu slouží funkce galerie. Každá entita má v galerii svůj název a popis. CASE Studio podporuje vytváření uživatelských rolí. Každá role má svůj název a jsou jí nastavena jednotlivá práva ke každému objektu. Takto nastavené role se pak mohou přiřazovat jednotlivým uživatelům.

K dispozici je i správce verzí, který obsahuje několik pro verzování důležitých funkcí. Hlavním cílem je udržet si přehled o jednotlivých verzích a zjišťování odlišností jednotlivých verzí. Podporované funkce jsou vkládání modelu do správce verzí, ukládání revizí a porovnávání verzí. Pomocí slovníku lze vytvářet vlastní uživatelské typy proměnných, které se mohou použít ve vlastním návrhu ER diagramu.

10.3. Další CASE nástroje

Řada velkých produktů obsahuje vlastní CASE nástroje. Většina z nich podporuje zavedené metodologie jako je Unified Modelling Language (UML) nebo dvouúrovňový návrh databáze.

Databáze Oracle 9i má **Oracle Designer**, který je součástí vývojového balíku Oracle 9i Developer Suite. Tento balík podporuje hlavně vývoj webových aplikací. Je plně integrován do ostatních produktů řady 9i.

Společnost Sybase vyvíjí řadu let **Power Designer**. Současná verze 9.5 obsahuje nástroje pro obchodně orientovanou procesní analýzu a dále prostředí pro datovou a objektovou analýzu informačních systémů.

Společnost Rational Software produkuje software **Rational Rose** a Rational XDE. Tato firma stála u zrodu UML, je také tvůrcem metodiky pro vývoj informačního systému Rational Unified Process. Produkty pokrývají celý životní cyklus projektu od analýzy přes objektový a datový návrh až k vlastnímu vývoji, testování a konfiguračnímu a změnovému řízení. Automaticky vytvářejí kód v C++, Visual Basic, Javě atd.

Firma SPARX Systems je producentem softwaru **Enterprise Architekt** v.4., který je určen pro modelování v UML.

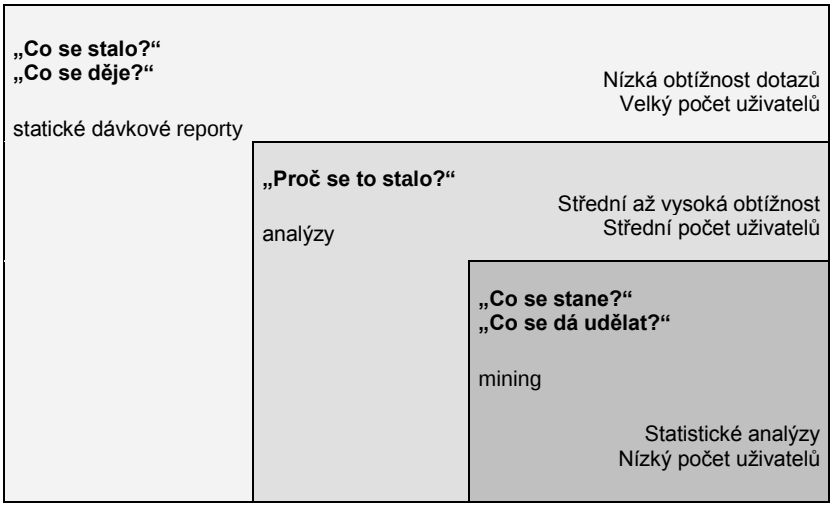
Pro návrh geodatabáze v ArcGIS od firmy ESRI se používá pro modelování **Microsoft Visio Enterprise** s komponentou **ArcGIS CASE Tools**. Možné je použít i Rational Rose. Opět se používá metodika UML.

11. Analýza dat

V této kapitole se budeme zabývat dalším využitím již existujících dat uložených v databázích. Analýzou dat zde nemyslíme úvodní analýzu dat pro potřebu modelování a návrh struktur. Zde je na mysli analýza existujících dat za účelem získání nových informací. Existující data je třeba přeměnit dál v informace nezbytné pro řízení a rozhodování. Datová základna sama o sobě není nic, pouhé skladiště dat. Data je třeba transformovat na informace tím, že na ně lidé prostřednictvím různých nástrojů aplikují své zkušenosti a znalosti. Kategorie práce s informacemi můžeme rozlišit na několik způsobů:

- Nejjednodušší využívání dat prostřednictvím pravidelného prohlížení statických reportů, bez dalšího detailního zkoumání.
- Vyšší a výkonný manažer vyžaduje denní obecný přehled o stavu a nezajímá se o detaily.
- Analytik využívá nejprve data tradičním způsobem - začíná na úrovni připravených dotazů a postupně proniká do většího detailu v závislosti na obdrženém výsledku.
- Průzkumy za účelem hledání neobvyklých vztahů mezi daty a na základě výsledků formulování hypotéz pro budoucí data.

Analytický přístup k datům lze rozdělit podle existence či neexistence fyzických vazeb ve zkoumaných datech. Pokud jsou zkoumána data s existujícími fyzickými vazbami, hovoří se nejčastěji o termínu **OLAP** - On-Line Analytical Processing. Do druhé kategorie analýz zkoumající data mimo rámec fyzických existujících vazeb lze zařadit **data mining** (dolování dat).



Obr. 40 Kategorie dotazů a uživatelů

OLAP analýza

Instituce *OLAP Council* definuje OLAP jako technologickou kategorii umožňující uživatelům interaktivně přistupovat k datům prostřednictvím různých pohledů na informace získané transformací zdrojových dat, a tím získat dimenzionální obraz zkoumaného prostředí. Pro snadnější pochopení je prostor popisován formou vícerozměrné kostky, datové krychle.

Multidimenzionální koncept uložení a manipulace s daty realizuje uložení dat v kombinaci různých definovaných dimenzí. Je možné definovat výpočty uvnitř dimenzí i mezi dimenzemi. OLAP umožňuje uživateli různé pohledy na data podle dimenzí (řezů) a podle potřeby pohledy dynamicky měnit. OLAP umožňuje pohyb ve směru vyššího detailu hodnot a zpět. Uložení výsledků OLAP se děje mimo zdrojová data.

Příklad: Jednoduchá relační databáze obsahuje demografické údaje pro území od úrovní obcí a v časových obdobích několika let. Z této databáze můžeme vytvořit datovou kostku. Hodnoty územní dimenze počtu obyvatel se sdruží do hierarchie: obec, pověřená obec, okres, kraj, země. V kostce můžeme přecházet od nejnižších úrovně počtu obyvatel obce ve směru vyšším na počty obyvatel pověřených obcí, okresů, krajů až na úroveň státu. V dimenzi času můžeme přecházet např. v počtech narozených dětí od nejnižšího údaje dne, přes týden, měsíc, rok. Další dimenzí může být věková struktura obyvatel. Operace přechodu na hierarchicky vyšší, obecnější úroveň

se nazývá *roll-up* – zobrazované údaje mají podobu souhrnů. Opačná operace přechodu na podrobnější pohled na data se nazývá *drill- down*. Datová krychle obsahuje jak datová pole, tak dílčí souhrny (např. počty, nebo průměry).

Data mining

Data mining ("dolování dat") lze obecně charakterizovat jako *proces extrakce relevantních, předem neznámých nebo nedefinovaných informací z velmi rozsáhlých databází*. Pro podporu data mining existuje již řada produktů. Umožňují identifikovat skryté korelace, analýzy vazeb ve velkých objemech dat, vytvářet modely pro predikci různých situací a procesů apod. Jde především o tzv. "data-driven" analýzy, tj. analýzy odvozované z obsahu dat, nikoli předem specifikované uživatelem nebo implementátorem.

Data mining je postaven na několika technikách, jejichž stručná charakteristika je:

- Shlukování (clustering) - znamená rozdělení databází, takže záznamy jsou seskupovány podle obdobných charakteristik (např. objemů prodeje, komodit, lokalit apod.),
- klasifikace - vytváří profily každé skupiny objektů definováním podstatných atributů těchto objektů,
- predikce - odhaluje závislosti hodnot jednoho atributu na hodnotách ostatních atributů v tomtéž záznamu a predikuje specifické hodnoty atributu pro nové záznamy,
- asociace - odhaluje veškeré asociace mezi transakcemi, tj. odvozuje z hodnot jedné transakce možnosti jejich výskytu v jiných transakcích,
- odhalování sekvenčních vzorů - nachází obdobné vzory v transakcích odvozováním z podobností logiky posloupností jejich operací nebo položek,
- odhalování obdobných časových sekvencí - nachází obdobné časové sekvence činností nebo procesů zaznamenaných v databázích.

Data mining je ve vazbě s datovými sklady významným analytickým nástrojem integrujícím v sobě možnosti rozsáhlých databází a jejich zpracování na bázi výkonné techniky a velmi sofistikovaných algoritmů, dnes v běžných aplikacích nedostupných.

12. Databázové systémy – stručný přehled produktů

Informace o jednotlivých produktech můžete nalézt na následujících internetových odkazech.

Databázový produkt	Odkaz
MS Access	http://office.microsoft.com
MS SQL server	http://www.microsoft.com/sql/
ORACLE	http://www.oracle.com
DB2	http://www-306.ibm.com/software/data/db2/
INFORMIX	http://www-306.ibm.com/software/data/informix/
SYSBASE	http://www.sybase.com
MySQL	http://www.mysql.com/
PostgreSQL	http://www.postgresql.org

12.1. Microsoft ACCESS

Je součást softwarového balíku Microsoft Office. První verzí byl Access verze 2, následovala verze Access 97, Access 2000, Access XP a nyní je nejnovější verzí Access 2003. Produkt je lokalizovaný do češtiny. SŘBD je stroj Microsoft Jet Database Engine (MS JDBE). Na tomto systému je založen i MS Visual Basic a další aplikace, které využívají Microsoft Data Access Object (DAO). Význam MS JDBE je tedy zejména v jeho rozšířenosti. Je určen pro databáze menšího rozsahu, limit na velikost databázového dokumentu je omezen na 2 GB. Databáze je uložena ve formě jednoho souboru s koncovkou *.mdb.

12.2. MS SQL

Microsoft SQL Server je výkonný databázový systém společnosti Microsoft. Databázi MS SQL 2000 mohou využívat výhradně servery běžící na platformě Windows. Jedná se o hlavní databázový produkt firmy Microsoft.

12.3. ORACLE

Databázový systém ORACLE je produkt firmy Oracle Corporation, jehož historie stejně dlouhá jako relační databáze a jazyk SQL. Databáze ORACLE patří k nejvýkonnějším databázovým strojům, které jsou na trhu k dispozici. Nejnovější verze je ORACLE 10g. Stále je běžně používaná i verze ORACLE 8, 9i. Historie produktu sahá až do roku 1977.

Je k dispozici pro všechny hlavní softwarové i hardwarové platformy - od kapesních počítačů, přes osobní počítače až po velké superpočítače.

ORACLE dává vývojářům veškeré moderní prostředky pro práci s daty a pro pohodlnou správu databáze, nabízí pokročilé funkce dle standardu SQL (uložené procedury, triggery, uživatelské datové typy, transakční zpracování apod.). Vysokou spolehlivost a maximální bezpečnost zaručuje certifikace dle normy ISO 15048. Většina produktů pro vývojáře je zdarma, platí se jen licenční poplatky za komerční nasazení a využití.

Tato databáze je zajímavá z hlediska geografických informačních systémů, neboť má zásuvný modul (cartridge) Oracle Spatial, který je určen pro ukládání prostorových dat. To umožňuje společné uložení a zpřístupnění atributových i geometrických dat v jediném datovém zdroji.

12.4. DB2 IBM

Databáze DB2 je produkt společnosti IBM. Je vhodná pro velkou podnikovou aplikaci. Existuje na velkém množství operačních systémů: Kromě operačních systémů IBM podporuje DB2 rovněž Linux, Windows, Solaris a HP-UX. DB2 dodržuje řadu standardů: ANSI/ISO SQL99, ODBC, ISO CLI atd. Jako základ WebServices dodržuje standardy XML, SOAP a UDDI. Časová náročnost ladění dotazů je malá díky patentovanému cost-based optimizeru. Pro řízení prostorových objektů je dodáván modul – DB2 Spatial Extender.

12.5. MySQL

MySQL je rychlý a stabilní databázový systém. Nabízí bohatý a velmi užitečný soubor funkcí. Často je použit u webových aplikací. Je podporován skripty PHP, CGI nebo ASP dle zvolené platformy virtuálního serveru. Databázi MySQL mohou využívat virtuální servery běžící na platformě Linux/Unix i Windows 2000. Velkou oblibu tento systém získal díky své licenci open source. Jen v případě prodeje komerční aplikace postavené na MySQL je třeba zakoupit licenci, která není drahá. Jinak lze využívat zdarma.

12.6. PostgreSQL

PostgreSQL patří k robustním a na schopnosti bohatým databázím s rozsáhlými možnostmi rozšiřování uživatelskými typy a funkcemi, programování logiky databáze a transakčního zpracování. Databázi PostgreSQL mohou využívat virtuální servery běžící na platformě Linux/Unix i Windows 2000. PostgreSQL je opět databází, která má též řešení pro prostorová data. Opět její používání je zdarma.

Existuje celá další řada produktů.

13. Příklady existujících geoinformačních databází

13.1. UIR-ADR a RÚIAN

Územně identifikační registr adresních míst České republiky byl vytvořen a spravován Ministerstvem práce a sociálních věcí ČR. Tento registr byl veřejně přístupný pro vyhledávání na webových stránkách ministerstva http://www.mpsv.cz/.

Registr byl 1. července 2012 nahrazen systémem základních registrů RÚIAN v souladu se zákonem č.111/2009Sb. o základních registrech.

RÚIAN je zkratka pro REGISTR ÚZEMNÍ IDENTIFIKACE, ADRES A NEMOVITOSTÍ, který je dostupný na adrese http://www.ruian.cz/.

Registr UIRADR obsahoval následující celostátní číselníky :

- oblastí
- krajů
- okresů
- obvodů obcí s rozšířenou působností (OPR)
- obvodů pověřených obecních úřadů (POÚ)
- obcí
- pražských obvodů
- NUTS4 – obvodů
- správních obvodů
- městských částí/městských obvodů
- částí obce
- ulic a veřejných prostranství
- stavebních objektů
- adresních míst
- dodávacích pošt

Číselníky oblastí, krajů, okresů, obvodů ORP, obvodů POÚ, obcí, pražských obvodů, NUTS4-obvodů a městských částí/městských obvodů jsou do UIR-ADR přebírány z Českého statistického úřadu, číselník správních obvodů z Magistrátu hl. m. Prahy, číselník částí obce z Ministerstva pro místní rozvoj, číselník adresních pošt z České pošty s.p. a číselníky ulic a veřejných prostranství, stavebních objektů a adresních míst jsou udržovány Ministerstvem práce a sociálních věcí na základě hlášení z obecních úřadů. Souřadnice X, Y definičních bodů adresních míst jsou do UIR-ADR poskytovány společností CEDA. Údaje v číselnících respektují verzi 4.1 Standardu ISVS k prostorové identifikaci. Poznámka: v číselníku stavebních objektů jsou evidovány všechny stavební objekty, které mají číslo popisné nebo evidenční, tj. i ty, kde nikdo trvale nebydlí.

14. Seznam literatury

1. Bejček, V.: Databázové systémy, nakladatelství VUT, fakulta strojní Brno 1992, ISBN 80-214-0422-1
2. Dobešová, Z.: Normalization of the Database of Regional Division of the Relief, Acta University Palackého, fac.rer. nat., Geographica 37, 2002, ISBN 80-244-0544-X, http://publib.upol.cz/~obd/fulltext/Geographica37/geogr37-1.pdf
3. Dobešová, Z.: Nebojme se databází, Geografické rozhledy 4, ročník 12, Praha 2003, ISSN 1210-3004
4. Dobešová, Z.: Remote Sensing Data in Metainformation Systems in Czech Republic, Proceedings of 4th international symposium Remote Sensing of Urban Areas, June 27-29, 2003, Regensburg, CD-ROM, ISSN 1682-1777
5. Farana, R.: Tvorba relačních databázových systémů, VŠB TU Ostrava 1999, ISBN 80-7078-706-6
6. Fikáček, I., Rozehnal I., Fikáček M.: Microsoft Access 2000, podrobný průvodce začínajícího uživatele, Grada 1999, ISBN 80-7169-879-2
7. Hnojil, J.: Operační a databázové systémy, ČVUT, fakulta stavební, Praha 1999
8. Kosek, J.: XML pro každého, Grada, Praha 2000, ISBN 80-7169-860-1
9. Kořínek, M.: Microsoft Access 2000, Koop, České Budějovice 2000, ISBN80-7232-100-5
10. Lacko, L.: Web a databáze, programujeme internetové aplikace, Computer Press Praha 2001, ISBN 80-7226-555-5
11. Pokorný, J., Halaška, I.: Databázové systémy, Vydavatelství ČVUT Praha, 1998
12. Pokorný, J., Halaška I.: Databázové systémy, vybrané kapitoly a cvičení, nakladatelství UK Praha 1998
13. Pokorný, J.: SQL ve třech lekcích, Příloha časopisu Geoinfo č.2,3,4/2002
14. Pokorný, J.: Objektově relační databáze, sborník konference DATAKON Brno 2002, ISBN 80-210-2958-7
15. Rigaux,P., Scholl, M., Voisard, A.: Spatial Databases with application to GIS, Morgan Kaufman Publishers, San Francisco 2002, ISBN 1-55860-588-6
16. Riordan, R. M.: Vytváříme relační databázové aplikace, Computer Press Praha 2000, ISBN 80-7226-360-9
17. Růžička, J.: Metadata pro prostorová data, disertační práce, Institut geoinformatiky, VŠB-TU, Ostrava 2002
18. Šarmanová, J.: Teorie zpracování dat, VŠB TU, FEI, Ostrava 1997, ISBN 80-7078-491-1
19. Šimůnek, M.: SQL kompletní průvodce, Grada, Praha 1999, ISBN 80-7169-692-7
20. Voženílek, V.: Geografické informační systémy I, historie a pojetí, nakladatelství UP Olomouc 2000, ISBN 80-7067-802-X
21. Welling, L., Thompson, L.: PHP a MySQL rozvoj a tvorba webových aplikací, SoftPress, Praha 2002, ISBN 80-86491-20-8