Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Diplomová práce

Geomorfologická databáze

Plzeň, 2007 František Vracovský

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem předloženou diplomovou práci zpracoval samostatně a s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 25.5.2007 .....

František Vracovský

Poděkování:

Na tomto místě bych rád poděkoval Karlu Jedličkovi za jeho ochotu, cenné rady a připomínky k práci.

Klíčová slova:

Metodiky pro databázové modelování, GmIS, databázové modely, topologie, Model Builder, Visual Basic for Application.

Abstrakt:

Diplomová práce se zabývá popisem vybrané metodiky pro návrh geografické databáze. Nejdříve zachycuje vývoj jednotlivých generací GIS. Následně popisuje uložení prostorových a atributových dat v databázových modelech. V závislosti na vybraném modelu lze odvodit různý postup (metodiku) při tvorbě konkrétní databáze. Důležitou složkou při vytváření databáze jsou také prostorové vztahy, které byly v praktické části práce modelovány pomocí topologie.

Praktická část diplomové práce popisuje vytvoření automatizovaných procesů v geomorfologické databázi, které vytvářejí nové, vyšší hierarchické formy reliéfu. Tyto formy reliéfu jsou následně provázány topologickými pravidly. Jedná o úpravy a doplnění struktury databáze. Při jejich implementaci bylo využíváno postupů popsaných v teoretické části práce.

Keywords:

Metodics for database modelling, GmIS, database models, topology, Model Builder, Visual Basic for Application.

Abstract:

This diploma thesis describes methodic chosen for the concept of geographic database. First of all it explains the development of GIS generations. Than the description of placement of spatial and attribute data in database models follows. According to chosen model it is possible to derive different procedure (methodics) while creating factual database. While creating the factual database, the spatial relationships are also a very important component, which was mold in the practical part of this diploma thesis using topology.

The practical part of this diploma thesis describes creating of automated procedures in geomorphological database, which create new higher hierarchical forms of relief. These forms of relief are followed by topological rules. This rules are used in modification and addition of the structure of this database. By the implementation was possible to use specific methodics explained in the theoretical part of diploma thesis.

Seznam obrázků

Obr. 2.1: Atributová složka 1. generace GIS - zpracováno podle [Jedlička2007]..... 10
Obr. 2.2: Systém flat souborů - zpracováno podle [Jedlička 2007]..... 11
Obr. 2.3: Uložení dat v duálním systému - zpracováno podle [Jedlička 2007].....11
Obr. 2.4: Duální systém - zpracováno podle [Jedlička 2007]..... 12
Obr. 2.5: Princip komunikace GIS a databáze u integrovaného systému - zpracováno podle [Jedlička 2007]..... 12
Obr. 2.6: Integrovaný systém - zpracováno podle [Jedlička 2007].....13
Obr. 3.1: Diagram typické hierarchické struktury – zpracováno podle [Hernandez 2006].....15
Obr. 3.2: Diagram typické síťové databáze - převzato z [Pokorný 1999].....15
Obr. 3.3: Složky relačního databázového modelu.....17
Obr. 3.4: Ukázka odkazu cizího klíče..... 18
Obr. 3.5: Kardinalita vazeb - zpracováno podle [Vokounová 2003]..... 19
Obr. 3.6: Schéma feature tables využívajících předdefinované datové typy – převzato z [OGC 2006] 20
Obr. 3.7: Asociace – zpracováno podle [Tomlinson 2003].....23
Obr. 3.8: Agregace – zpracováno podle [Tomlinson 2003].....24
Obr. 3.9: Kompozice – zpracováno podle [Tomlinson 2003].....24
Obr. 3.10: Srovnání databázových modelů - převzato z [Batko 2007] 28
Obr. 4.1: Typový E-R diagram – převzato z [Pokorný 1999].....30
Obr. 4.2: Symbol třídy a abstraktní třídy podle klasického UML – zpracováno podle [Page-Jones 2001].....31
Obr. 4.3: Znázornění tříd a vztahů mezi nimi podle notace ESRI - převzato z [Andrade et al. 2002].....32
Obr. 4.4: Příklad třídy s atributy a metodami - převzato z [Tilton, et al. 2002].....32
Obr. 4.5: Symbol objektu (instance třídy) ve své úplné a zkrácené formě pomocí klasického UML – převzato z [Page-Jones 2001].....33
Obr. 4.6: Atributy podle klasického UML – převzato z [Page-Jones 2001]..... 34
Obr. 4.7: Metody podle klasického UML – převzato z [Page-Jones 2001]..... 34
Obr. 4.8: Asociace v ESRI notaci – převzato z [Andrade, et al. 2002].....35
Obr. 4.9: Dědičnost v ESRI notaci - převzato z [Andrade, et al. 2002].....35
Obr. 4.10: Konkretizace v ESRI notaci - převzato z [Andrade, et al. 2002].....36
Obr. 4.11: Agregace v ESRI notaci.....36
Obr. 4.12: Kompozice v ESRI notaci - převzato z [Andrade, et al. 2002].....36
Obr. 4.13: Topologické zobrazení.....37
Obr. 4.14: Princip cluster tolerance - převzato z [ESRI 2004].....39
Obr. 4.15: Princip set ranks v ESRI topologii - převzato z [ESRI 2004].....40
Obr. 4.16: Postavení LaserScan Radius Topology v DBMS Oracle – zpracováno podle [Louwsma 2003].....41
Obr. 4.17: Struktura tabulek modulu Laser-Scan Radius Topology - zpracováno podle [Louwsma 2003].....42
Obr. 4.18: Popis návrhu Personal Geodatabase – převzato z [Arctur 2004].....44
Obr. 4.19: Postup při návrhu GDB - převzato z [Vokounová 2003].....45
Obr. 4.20: Ukázka vyexportované prvkové třídy do UML pomocí programu „ESRI Diagrammer“ 47
Obr. 4.21: Pozice a funkce basic layers v GmIS – převzato z [Minár, et al. 2005]..... 50
Obr. 4.22: Logický model geomorfologické databáze – převzato z [Minár, et al. 2005]..... 52
Obr. 4.23: Fyzický model datové sady „BasicLayers“, vytvořený pomocí ESRI Geodatabase Diagrammer v Microsoft Visio - převzato z [Mentlík, et al. 2006]..... 53

Obr. 4.24: Detail fyzického modelu elementárních forem, vytvořený pomocí ESRI Geodatabase Diagrammer v Microsoft Visio – převzato z [Mentlík, et al. 2006]....54
Obr. 5.1: Výchozí geomorfologická databáze s elementárními formami okolí Prášilského jezera... 55
Obr. 5.2: Elementární formy u Prášilského jezera.... 56
Obr. 5.3: Atributová tabulka ElementaryForms – ukázka z programu ArcCatalog....57
Obr. 5.4: Význam skriptu Select By Unique na příkladu geomorfologické databáze.... 58
Obr. 5.5: Vyplněná standalone table.... 58
Obr. 5.6: Celkový průběh vyplnění hodnot atributů feature class ElementaryForms na příkladu geomorfologické databáze....60
Obr. 5.7: Princip nástroje Dissolve....62
Obr. 5.8: Vytvoření vyšších forem v rozšíření Model Builder....63
Obr. 5.9: Výsledná struktura datasetu BasicLayers s vyššími formami - vytvořená v programu Visio 64
Obr. 5.10: Princip hvězdicového přístupu – vytvořeno v programu Visio.... 65
Obr. 5.11: Princip Kaskádového přístupu – vytvořeno v programu Visio....65
Obr. 5.12: Ukázka vytvořeného submodelu v rozšíření Model Builder....67
Obr. 5.13: Logický model znázorňující jednotlivá topologická pravidla mezi vrstvami – vytvořeno v programu Visio....68
Obr. 5.14: Submodel vytvoření topologie vyšších hierarchických forem v rozšíření Model Builder 69
Obr. 5.15: Část výsledného modelu vytvářejícího topologii....70
Obr. 5.16: Část fyzického modelu vytvořené topologie mezi vrstvami, celý model je uložen v příloze a na přiloženém CD - vytvořeno pomocí programu Visio.... 70
Obr. 5.17: Příklady topologických chyb....71
Obr. 5.18: Vymazání pseudo-uzlů pomocí nástroje Error Inspector....72
Obr. 5.19: Příklad pseudonodes....73

Seznam tabulek

Tabulka 3.1: Vyjádření násobnosti vazby v objektově orientovaném databázovém modelu.... 23
Tabulka 5.1: Pravidla použitá u vrstev ElementaryForms.... 66
Tabulka 5.2: Pravidla použitá u vyšších hierarchických vrstev.... 68

Obsah

1 Úvod ... 9
2 Generace GIS ... 10
2.1 První generace ... 10
2.2 Druhá generace ... 11
2.3 Třetí generace ... 13
3 Databázové modely pro ukládání prostorových a atributových dat ... 14
3.1 Hierarchický databázový model ... 14
3.2 Síťový databázový model ... 15
3.3 Relační databázový model ... 16
3.3.1 Atributová složka dat ... 16
3.3.2 Základní názvosloví relačních databází ... 16
3.3.3 Prostorová složka dat ... 19
3.4 Objektově orientovaný databázový model ... 21
3.4.1 Hlavní části objektově orientovaného databázového modelu ... 22
3.4.2 Prostorová reprezentace objektově orientovaného modelu ... 24
3.5 Objektově-relační databázový model ... 26
4 Existující metodiky pro modelování geografické databáze ... 29
4.1 Grafické znázornění (geo)databázových modelů ... 29
4.1.1 E-R modely ... 29
4.1.2 Přístup pomocí jazyka UML ... 30
4.2 Prostorové vztahy - topologie ... 36
4.2.1 Matematická topologie ... 37
4.2.2 Topologie v GIS ... 38
4.2.3 Shrnutí topologie ... 43
4.3 Návrh geografické databáze ... 44
4.3.1 Konceptuální model ... 45
4.3.2 Logický model ... 46
4.3.3 Fyzický model ... 47
4.4 Možnosti vytvoření GDB ... 47
4.5 Ukázka návrhu na příkladu geomorfologického informačního systému ... 48
5 Praktická část diplomové práce ... 55
5.1 Vytyčení cílů práce ... 55
5.2 Vyplnění atributové tabulky prvkové třídy ElementaryForms ... 55
5.2.1 Skript „Select By Unique“ ... 57
5.2.2 Skript UpdateEF ... 59
5.3 Tvorba jednotlivých prvkových tříd vyšších forem ... 61
5.3.1 Vytvoření polygonových vrstev ... 62
5.3.2 Vytvoření liniových vrstev ... 62
5.3.3 Použití rozšíření „Model Builder“ ... 62
5.4 Topologické vztahy mezi vyššími formami ... 64
5.4.1 Submodel vytvoření topologie ElementaryForms ... 65
5.4.2 Submodel vytvoření topologie vyšších forem ... 68
5.4.3 Výsledný model ... 70
5.4.4 Skript „Delete Pseudonodes“ ... 72
6 Závěr ... 75
Seznam použité literatury ... 76
Příloha ... 80

Slovníček pojmů..... 80
Fyzický model topologie..... 81
Adresářová struktura CD..... 82

1 Úvod

Ještě před patnácti až dvaceti lety, byla většina geografických databází vytvářena digitalizací pouze analogových forem map. V následujících letech následoval u nás a ve světě velký rozmach zejména bezkontaktních metod sběru dat (letecká fotogrammetrie, DPZ), které našly svou cestu (v digitální podobě) na trhu s informacemi. Tyto metody mají svůj význam nejen při vizualizaci určitého území, ale zejména při zpřesnění původních dat, uložených v geografických databázích. V dnešní době jsou rastrová i vektorová data uložena ve velkých databázových systémech (např. Oracle), což vede k velkému nárůstu objemu databází a nutnosti řešit dříve méně potřebné záležitosti (indexace, optimalizace, rychlost, atd.). Vzhledem k odezvě, kterou mělo ukládání prostorové informace do databáze, se z klasických relačních databází staly objektově-relační. V posledních letech byl zaveden formát OGC pro ukládání prostorových dat, který využívá většina prostorových databází.

Za největším rozmachem GIS stojí bezesporu internet jako médium pro sdílení digitálních dat. Při rozhodování, zda hledaná digitální data stáhnout, je důležité si prověřit zejména historii a kvalitu datových sad. Mezi důležitá kritéria volby dat patří obsah, zdroj, stáří a podrobnost. Tyto informace poskytují distributoři většinou ve formě metadat (jinak též „data o datech“), která urychlují proces hledání určitých informací a zabraňují stažení nepotřebných dat. Již na těchto metadatech (struktura, obsah, apod.) lze usoudit, jak kvalitní stažená data budou.

2 Generace GIS

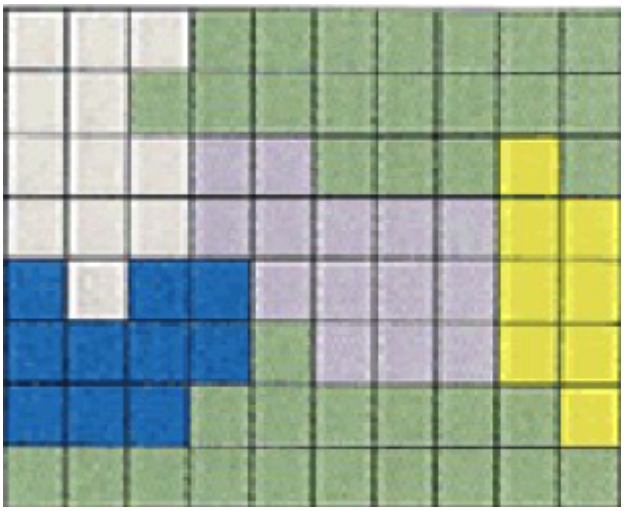
Tato kapitola byla zpracována podle přednášek „Úvod do GIS“ podle: [Jedlička 2007] a podle [Perchta 2007].

Rozeznáváme tři generace GIS podle jejich vývinu od minulosti po současnost. Jedním z faktorů, který ovlivnil rozvoj GIS byl spjatý s velkým rozmachem databázových systémů (zejména relačních). Do počátků vývoje GIS řadíme zejména souborově orientované databáze, které ovšem v GIS velké uplatnění nenašly. V další generaci se setkáváme s relačními databázemi pod správou RDBMS, které již našly své uplatnění v GIS a v současnosti patří mezi nejrozšířenější. Protože se tato generace tolik rozšířila, zabránila z části nástupu poslední generace, která má základ v objektové orientovaném přístupu. Zatím jako nejlepší řešení se jeví objektově-relační přítup.

2.1 První generace

Systém bez atributových dat

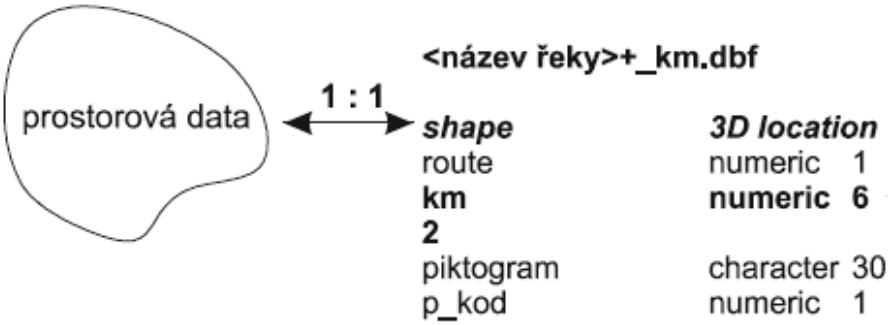
Do první generace GIS patří systém, který nepodporoval ukládání atributových dat. V dnešní době se zachoval pouze u rastrových dat, které nemají oddělený popisný a prostorový údaj. Jediným atributem, který lze do tohoto systému uložit je přímo hodnota buňky (viz Obr. 2.1). Mezi první průkopníky tohoto systému patřil ve svých počátcích GRASS.



Obr. 2.1: Atributová složka 1. generace GIS - zpracováno podle [Jedlička2007]

Systém „Flat“ souborů

Jedná se o tabulky, u kterých nelze vytvářet relace (využívají souborově orientovaný systém). Ke každému objektu je možné přiřadit pouze 1 tabulku (spojení prostorových a atributových dat je provedeno přes id_objektu, které funguje jako klíč, ve vztahu 1:1, viz Obr. 2.2). Tento způsob je velice nepraktický, pokud potřebujeme nějaký údaj využívat na více místech (neřeší paralelní přístup více uživatelů). Příkladem takového systému je třeba GIS IDRISI. Obecně se tyto GIS využívají dodnes převážně při rastrových analýzách.



Obr. 2.2: Systém flat souborů - zpracováno podle [Jedlička 2007]

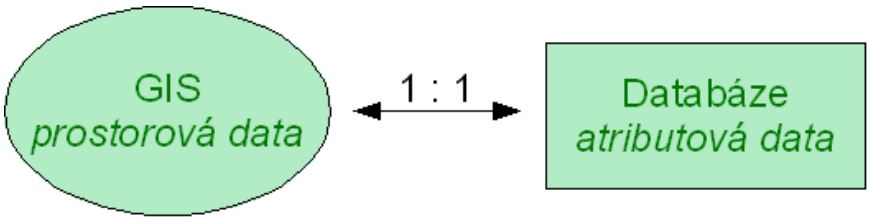
2.2 Druhá generace

Duální/hybridní způsob propojení prostorových a atributových dat

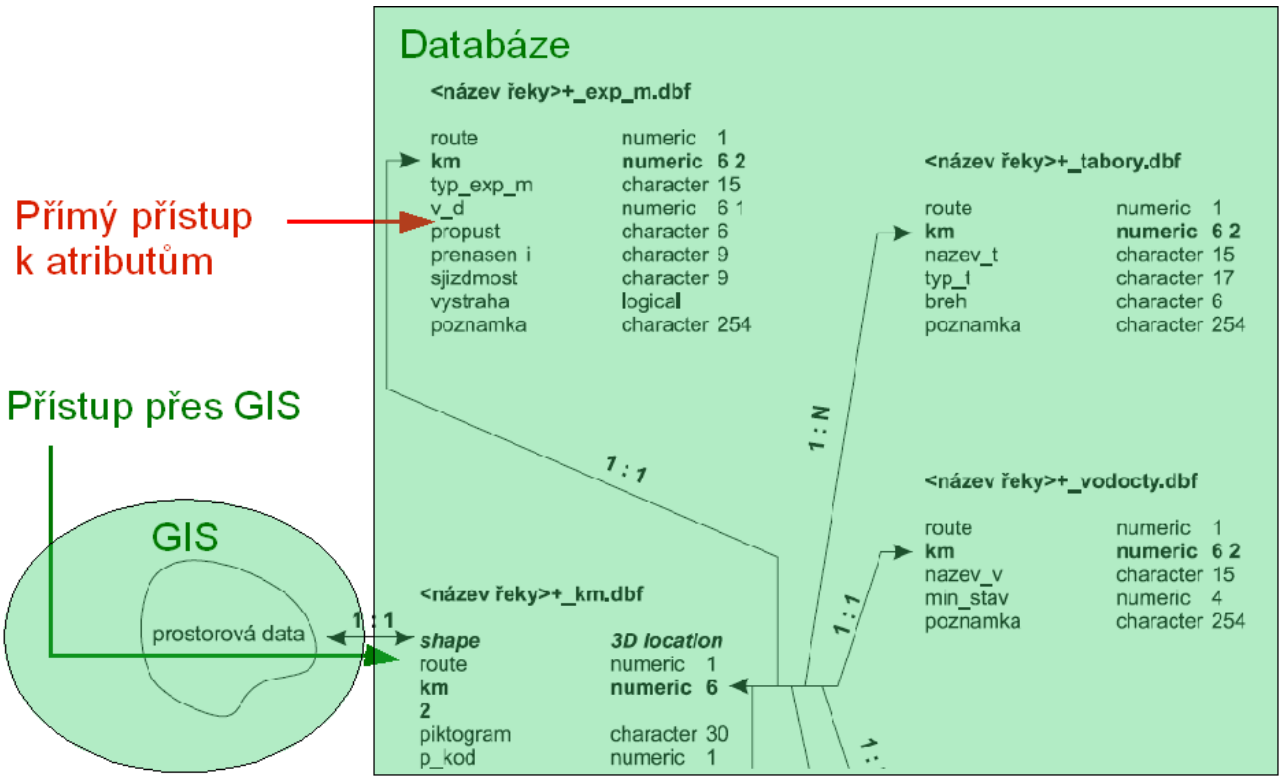
Je to velice rozšířený způsob použití relační databáze, kde grafika se zpracovává v jednom systému, atributy v DBMS (viz Obr. 2.4). Např., dřívější verze ARC/INFO, kde už z názvu je vidět, že jeden systém (ARC) zpracovává grafiku a druhý (INFO) se stará o popisné atributy. Dalším příkladem může být formát shapefile, který má prostorová data uložena v souboru s příponou *.shp, atributová data v souboru s příponou *.dbf a spojení mezi atributovými a prostorovými daty zajišťuje indexový soubor s příponou *.shx. Občas se ještě používá rozdělení na DBMS implementovaný do GIS a externí DMBS (zde je nutné podotknout, že i 1. případ umožňuje využívat externí DBMS).

Kvůli oddělení prostorové a atributové složky má však tento model problémy s integritou (konzistencí) dat (nelze zaručit, že někdo pomocí DBMS neobejde GIS a nemodifikuje data přímo v DB). Právě kvůli umístění prostorové složky na straně GIS může mít tento způsob problémy s ukládáním většího objemu dat. Proto je nutné data dělit podle zájmových území, či mapových listů.

Příkladem může být již zmiňované ARC/INFO; MGE; ArcGIS, pracující s formátem SHP a další systémy.



Obr. 2.3: Uložení dat v duálním systému - zpracováno podle [Jedlička 2007]



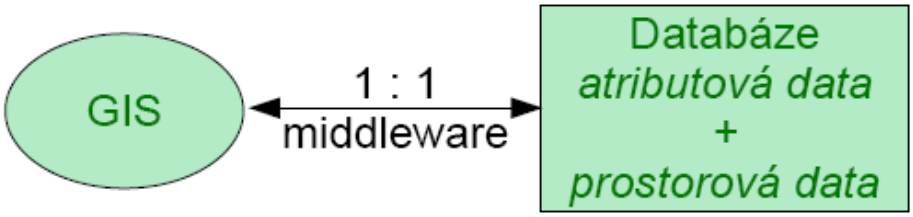
Obr. 2.4: Duální systém - zpracováno podle [Jedlička 2007]

Integrovaný systém

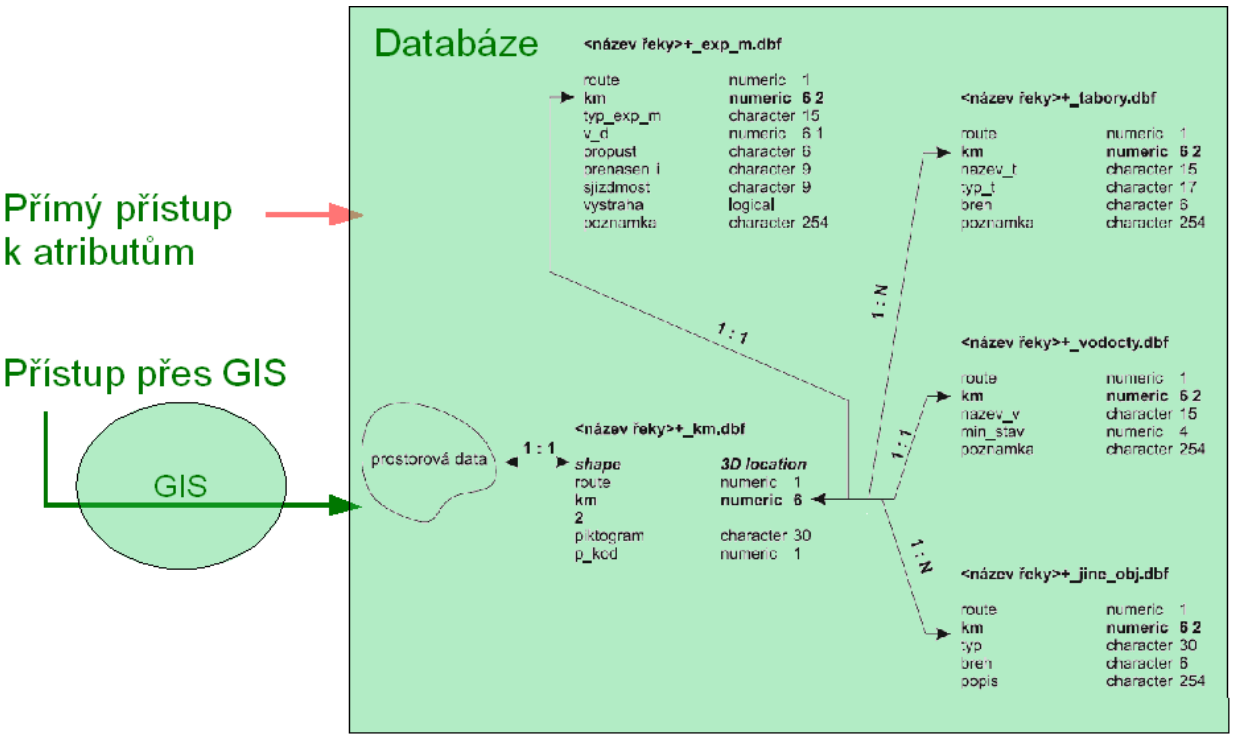
Tento model se rozšiřuje hlavně v poslední době. Vše je uloženo v jedné databázi (jak prostorová, tak popisná složka), ale využíván je pouze standardní relační model. Na Obr. 2.5 se o management všech geografických dat se stará tzv. middleware, což je produkt tvořící komunikační vrstvu mezi databází a GIS, nikoli tedy databáze samotná. Díky uložení prostorové části v databázi je možné používat bezešvá prostorová data a odpadá dělení prostoru na mapové listy.

Model je však relativně pomalý (hlavně kvůli tomu, že standardní relační databáze neumí efektivně ukládat prostorová data). Není definován standard přístupu k prostorovým datům v databázi (prostorová data jsou uložena jako binární posloupnosti (Binary Large Objects - BLOBs), což znemožňuje data zpracovávat již v DBMS či pomocí jiného SW).

Příkladem může být ARC/INFO, SDE, Geomedia.



Obr. 2.5: Princip komunikace GIS a databáze u integrovaného systému - zpracováno podle [Jedlička 2007]



Obr. 2.6: Integrovaný systém - zpracováno podle [Jedlička 2007]

Speciálním případem integrovaného modelu jsou pak prostorové databáze, které umožňují ukládat prostorová data ve standardizované podobě spolu s atributy. Díky tomu, že middleware se v podstatě přesunul do DBMS (je součástí DBMS), jsou odstraněny problémy s integritou dat a částečně i s výkonností.

Příkladem je Oracle Spatial.

2.3 Třetí generace

Za třetí generaci GIS se považuje systém, který ukládá prostorové a atributové údaje do jedné databáze jako prostorové objekty. Využívá k tomu objektově orientovaný přístup, který umožňuje manipulaci s prostorovými daty bez nutnosti použití specializované prostorové databáze nebo prostředků pro konverzi údajů. Dělí se na objektově orientované a objektově-relační modely, kterými se budu podrobněji zabývat v následující kapitole.

3 Databázové modely pro ukládání prostorových a atributových dat

Databázové modely se (podle vývoje) dělí na :

- hierarchické,
- síťové,
- relační,
- objektově orientované,
- objektově-relační.

Koncoví uživatelé aplikací se většinou tímto dělením, na rozdíl od návrhářů databází, nemusí zabývat. Pro návrháře je ale jedním z mnoha faktorů při rozhodování, který software do jaké firmy nasadit. Jádrem věci je, že logický model musí komplexně a co nejpřesněji popsat reálný svět v podobě databáze, protože cena (nijak zanedbatelná částka) a použitelnost dané databáze závisí na tom, jak přesně bude výsledná databáze modelovat realitu. Nasazení nejlepšího logického modelu v dané situaci a jeho následné vybudování totiž vyžaduje rozhodnutí hned na počátku tvorby dané databáze.

Protože hierarchické a síťové databázové modely stály na úplném počátku tvorby databází, bylo možné se s nimi setkat u první generace GIS. Jedná se sice o zastaralé modely, ale pro úplnost zde uvedu alespoň jejich stručný popis.

3.1 Hierarchický databázový model

V tomto modelu jsou (podle [Hernandez 2006]) data strukturována hierarchicky a obvykle se znázorňují jako obrácený strom. Jedna z tabulek slouží jako kořen obráceného stromu a ostatní jako větve vycházející z kořene. Obr. 3.1 ukazuje diagram typické hierarchické struktury databáze, kde se agent stará o několik bavičů a každý bavič má svůj časový harmonogram. Agent se také stará o svůj určitý počet zákazníků, jejichž požadavky na zábavu má za úkol plnit. Zákazník si domlouvá schůzky prostřednictvím agenta a tomu potom za jeho služby platí.

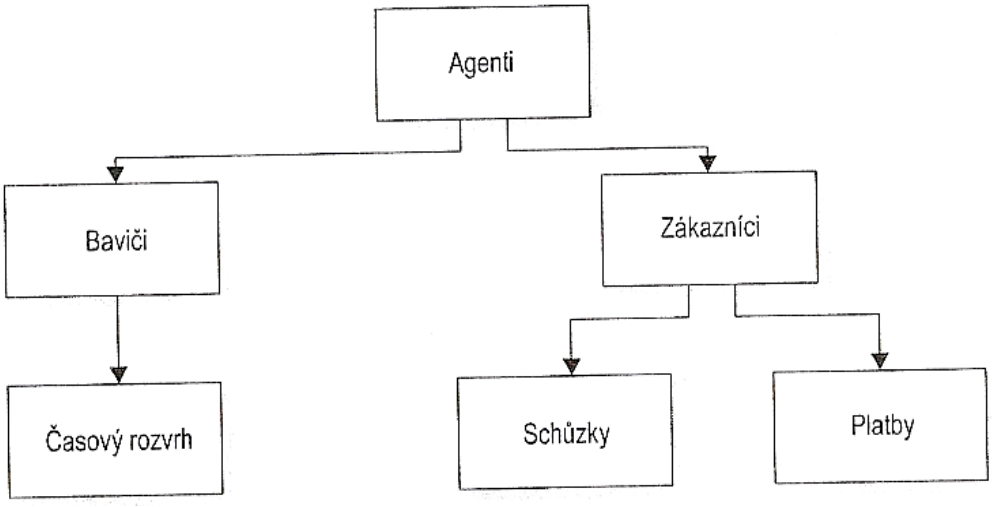
Vztah je v hierarchickém databázovém modelu reprezentován termíny rodič – potomek. V tomto typu vztahu může být tabulka rodiče přidružena k jedné, nebo více tabulkám potomků, ale tabulka potomka může být přidružena pouze k jedné tabulce rodiče. Uživatel může k datům přistupovat tak, že začne v kořenové tabulce a postupně se propracovává k hledaným datům. Uvedená metoda vyžaduje, aby uživatel dobře znal strukturu databáze.

Klady hierachického databázového modelu:

- rychlý přístup k datům,
- referenční integrita, která je automaticky zabudována ve stromové struktuře (vymazání rodiče způsobí vymazání potomků).

Zápory hierachického databázového modelu:

- nepodporuje komplexní vztahy,
- redundance dat,
- nemožnost spojování typů entit na stejné úrovni hierarchie.

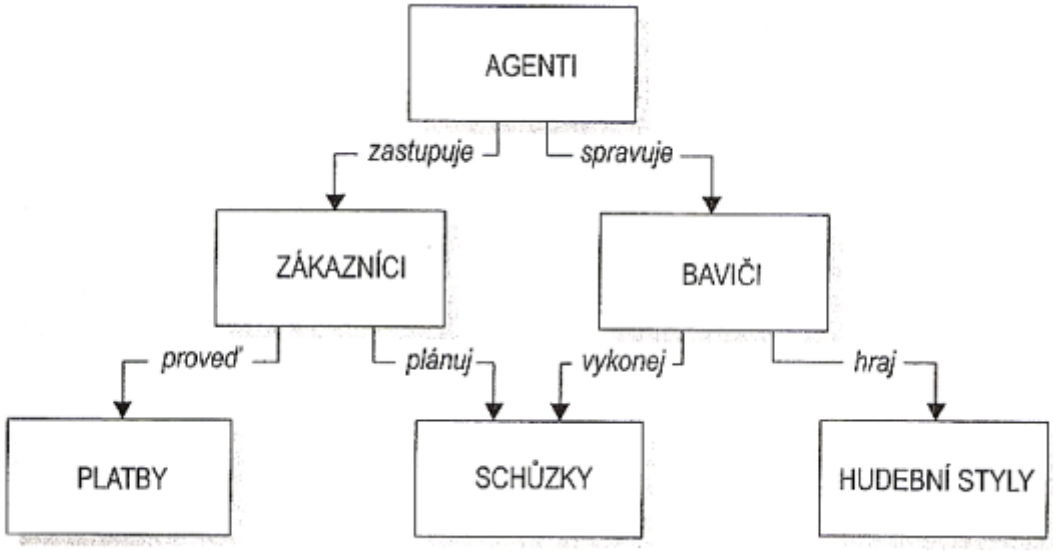


Obr. 3.1: Diagram typické hierarchické struktury – zpracováno podle [Hernandez 2006]

3.2 *Síťový databázový model*

Síťová databáze byla (podle [Hernandez 2006]) vyvinuta hlavně jako pokus o vyřešení problémů hierarchické databáze. Struktura síťového databázového modelu je vyjádřena v pojmech uzlů a množinových struktur, uvedených dále.

Na Obr. 3.2 zastupuje agent určitý počet klientů a určitý počet bavičů. Každý klient si naplánuje libovolný počet schůzek a platí agentovi za jeho služby. Každý bavič absolvuje určité množství schůzek a může hrát více různých hudebních stylů.



Obr. 3.2: Diagram typické síťové databáze - převzato z [Pokorný 1999]

Uzel reprezentuje soubor záznamů a množinová struktura reprezentuje a zřizuje vztah v síťové

databázi. Je to snadno pochopitelná konstrukce, která vytváří vztah mezi dvěma uzly tak, že jeden uzel je definován jako vlastník a druhý jako prvek (tato metoda je značným vylepšením vztahu rodič/potomek). Množinová struktura podporuje vztah 1:N, neboli jeden záznam v uzlu vlastník může být v relaci k jednomu, nebo více záznamům v uzlu člen. Na druhé straně, jeden záznam v uzlu člen je ve vztahu pouze k jednomu záznamu v uzlu typu vlastník. Záznam v uzlu typu člen navíc nemůže existovat, aniž by byl ve vztahu k nějakému záznamu v odpovídajícím uzlu typu vlastník. Např., klient musí být přiřazen k agentovi, ale agent bez klientů může v databázi být.

Mezi dvěma uzly může být definována jedna, nebo více množin (spojení), a libovolný uzel může být součástí dalších množin s jinými uzly v databázi. Např., na Obr. 3.2 je uzel ZÁKAZNÍCI ve vztahu k uzlu PLATBY prostřednictvím množinové struktury *Proveď*. Je také ve vztahu k uzlu PLATBY ve vztahu k uzlu SCHUZKY prostřednictvím množinové struktury *Plánuj*. Uzel SCHUZKY je zároveň ve vztahu k uzlu BAVIČI prostřednictvím množinové struktury *Vykonej*.

Klady síťového databázového modelu:

- rychlý přístup k datům,
- komplexnější dotazy než u hierarchického modelu,

Zápory síťového databázového modelu:

- uživatel musí znát strukturu databáze, aby mohl pracovat s množinovými strukturami
- nesnadné změnit strukturu databáze bez nutnosti změny aplikace, které s ní pracují,
- ztrácí se přehled nad databází (časem velice složité struktury).

3.3 Relační databázový model

3.3.1 Atributová složka dat

Většina geografických digitálních dat je (podle [Tomlinson 2003]) v dnešní době uložena v relačním databázovém modelu. Je to soubor tabulek (nazývaných relace), které jsou mezi sebou propojeny pomocí primárních a cizích klíčů. Sloupce tabulky se označují jako atributy a řádky bývají označovány, buď jako řádky, nebo n-tice. Průsečíkem sloupce a řádky je záznam relace. Každý sloupec tabulky obsahuje pouze jeden datový typ (např. date, string, integer, long, short, atd.). Ukládání dat do relačních databází velmi ulehčují specializované softwary, jako např. „Spatial Database Engine“ (SDE) od firmy ESRI, který umožňuje čtení a zapisování dat uložených v klasických databázích RDBMS (např. Oracle).

3.3.2 Základní názvosloví relačních databází

(zpracováno podle [Kaliková 2007])

Relace

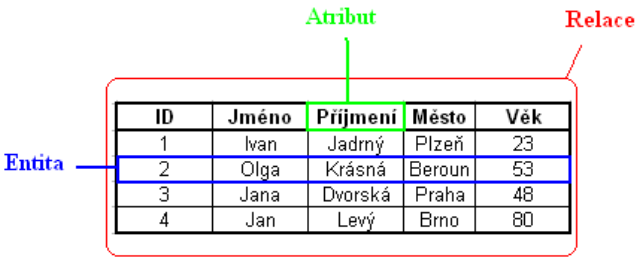
Relace, někdy nazývaná *relační množina, entitní typ* nebo *entitní množina*, je množina záznamů (dvourozměrná tabulka), kde jednotlivé řádky tabulky jsou unikátní (nesmí obsahovat 2 či více stejných řádků).

Entita

Entita je objekt reálného světa, který je schopen nezávislé existence a je jednoznačně odlišitelný od ostatních objektů. Z pohledu informatika je to cokoli, o čem potřebujeme v systému uchovat nějaké informace (řádka v tabulce).

Atribut

Atribut je skutečnost, kterou o dané entitě evidujeme v systému (např. u entity VÝROBKY budeme evidovat atribut JMÉNO VÝROBKU). Každému atributu lze přiřadit jeden ze základních datových typů (např., date, string, integer, long, short, atd.).



Obr. 3.3: Složky relačního databázového modelu

Klíč

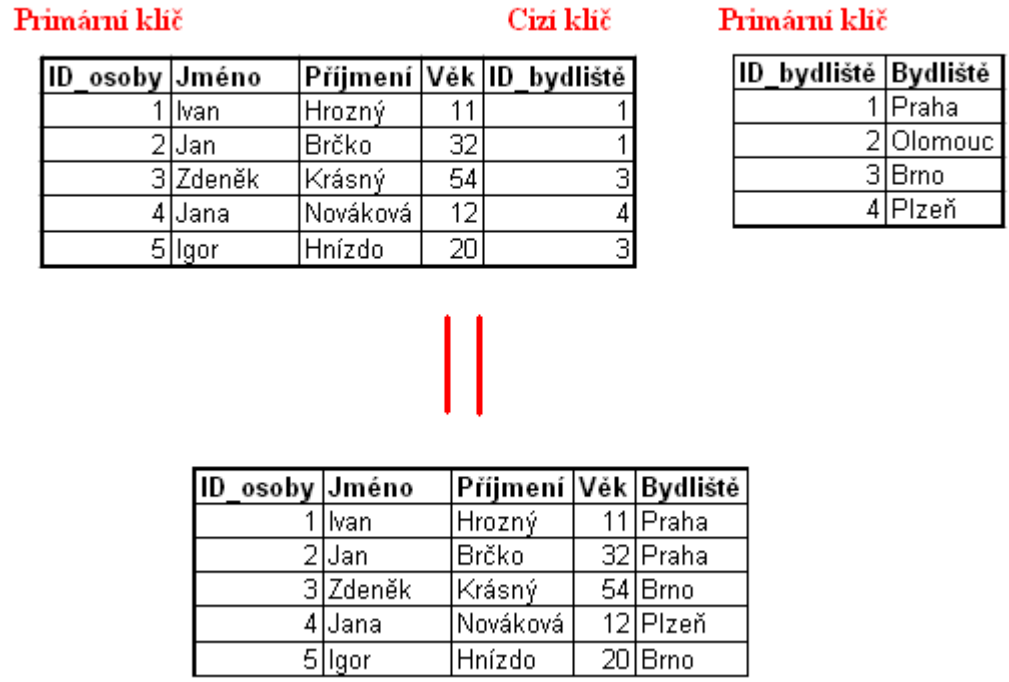
Klíč je podle [Tuček 1998] obecně unikátní atribut nebo kombinace atributů. Klíče mají v relačním databázovém modelu dvě důležité úlohy:

1. Reprezentují asociace mezi typy entit. To znamená, že slouží jako vyhledávače mezi různými relacemi, tj.spojují tabulku s jinou tabulkou. V tomto smyslu jsou klíče relativními vyhledávači, protože budují spojení přes atributové hodnoty a ne přes registry adres v paměti. Výhoda relativních vyhledávačů je naproto zřejmá – pokud se změní fyzická struktura (např. jiný operační systém), relační model se měnit nemusí.
2. Kromě tohoto se klíče mohou použít k jednoznačné identifikaci entit. Pomocí klíče můžeme identifikovat jednotlivý datový záznam v tabulce (řádek). Takové klíče se nazývají primární.

Primární klíč (Primary key) je množina atributů v tabulce, která má tyto vlastnosti:

1. Je jednoznačná – v relaci neexistuje druhá n-tice, která by měla pro tuto množinu atributů stejné hodnoty.
2. Je minimální – není možné žádný atribut vypustit, aby se neporušilo pravidlo č.1.

Cizí klíč (Foreign key) je množina atributů databázové tabulky, která odkazuje na množinu atributů jiné nebo stejné tabulky. Hodnoty takového sloupce musí být shodné s některým atributem, ke kterému je klíčem. Vytváří se tak reference – odkaz (viz Obr. 3.4).



ID_osoby	Jméno	Příjmení	Věk	ID_bydliště
1	Ivan	Hrozný	11	1
2	Jan	Brčko	32	1
3	Zdeněk	Krásný	54	3
4	Jana	Nováková	12	4
5	Igor	Hnízdo	20	3

ID_bydliště	Bydliště
1	Praha
2	Olomouc
3	Brno
4	Plzeň

ID_osoby	Jméno	Příjmení	Věk	Bydliště
1	Ivan	Hrozný	11	Praha
2	Jan	Brčko	32	Praha
3	Zdeněk	Krásný	54	Brno
4	Jana	Nováková	12	Plzeň
5	Igor	Hnízdo	20	Brno

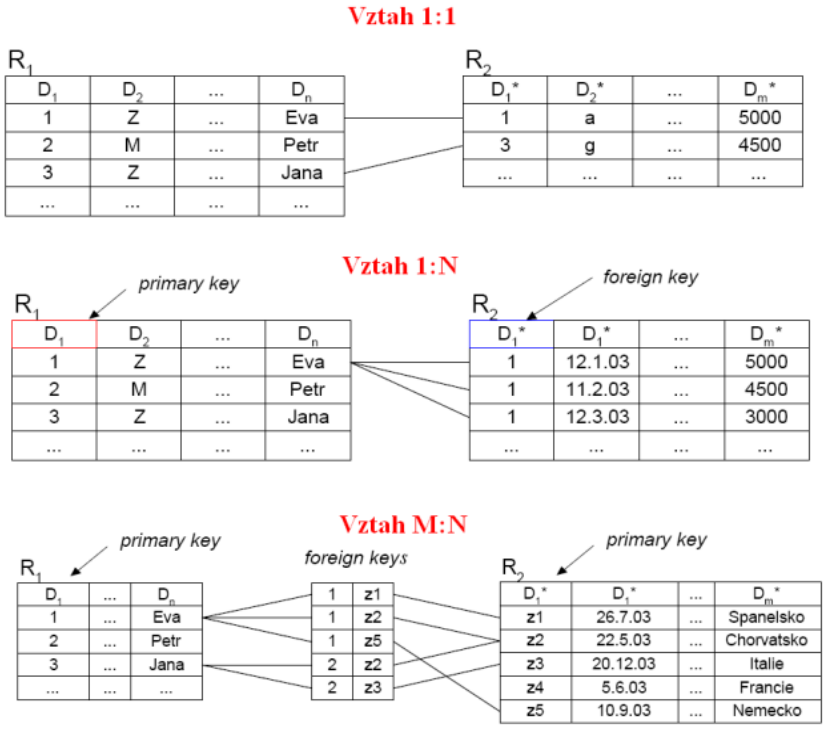
Obr. 3.4: Ukázka odkazu cizího klíče

Vztahy mezi relacemi

Kardinalita vazby

Kardinalita vazby vyjadřuje, kolik entit jedné relace přísluší entitám druhé relace (viz Obr. 3.5).

1. Vztah 1:1 – na vazbě se podílí pouze jedna entita z relace R_1 a jedna entita z relace R_2.
2. Vztah 1:N – jedné entitě z R_1 může příslušet více entit z R_2, ale jedné entitě z R_2 může příslušet pouze 1 entita z R_1.
3. Vztah M:N - jedné entitě z R_1 může příslušet více entit z R_2 a naopak jedné entitě z R_2 může příslušet více entit z R_1.



Obr. 3.5: Kardinalita vazeb - zpracováno podle [Vokounová 2003]

Existuje několik dělení vztahů mezi relacemi, které jsou stejně důležité jako kardinalita vazby (např., povinnost výskytu, vazby x-ární, aj.). Detailní popis vazeb lze nalézt např. v: [DB1], [Hernandez 2006], [Pokorný 1999].

3.3.3 Prostorová složka dat

Základem pro ukládání prostorových dat je jejich převod do formátu, který lze uložit v databázi. Existuje několik formátů uložení prostorových dat v geografické databázi, které se liší použitým datovým typem pro uložení dat.

Zde bych rád uvedl formát OGC pro ukládání prostorových dat s jeho základním popisem. Podrobnější informace lze nalézt např. v: [OGC 2006].

Formát OGC

Nejnovější specifikací OGC standardu je verze 1.2.0 dostupná z [OGC 2006]. Architektura formátu OGC je založena na SQL implementaci předdefinovaných datových typů pro správu relačních tabulek. V těchto tabulkách jsou uloženy informace o geoprvcích (feature table), geometrii (Geometry) a souřadnicovém systému (Spatial Reference System) – viz Obr. 3.6, kde:

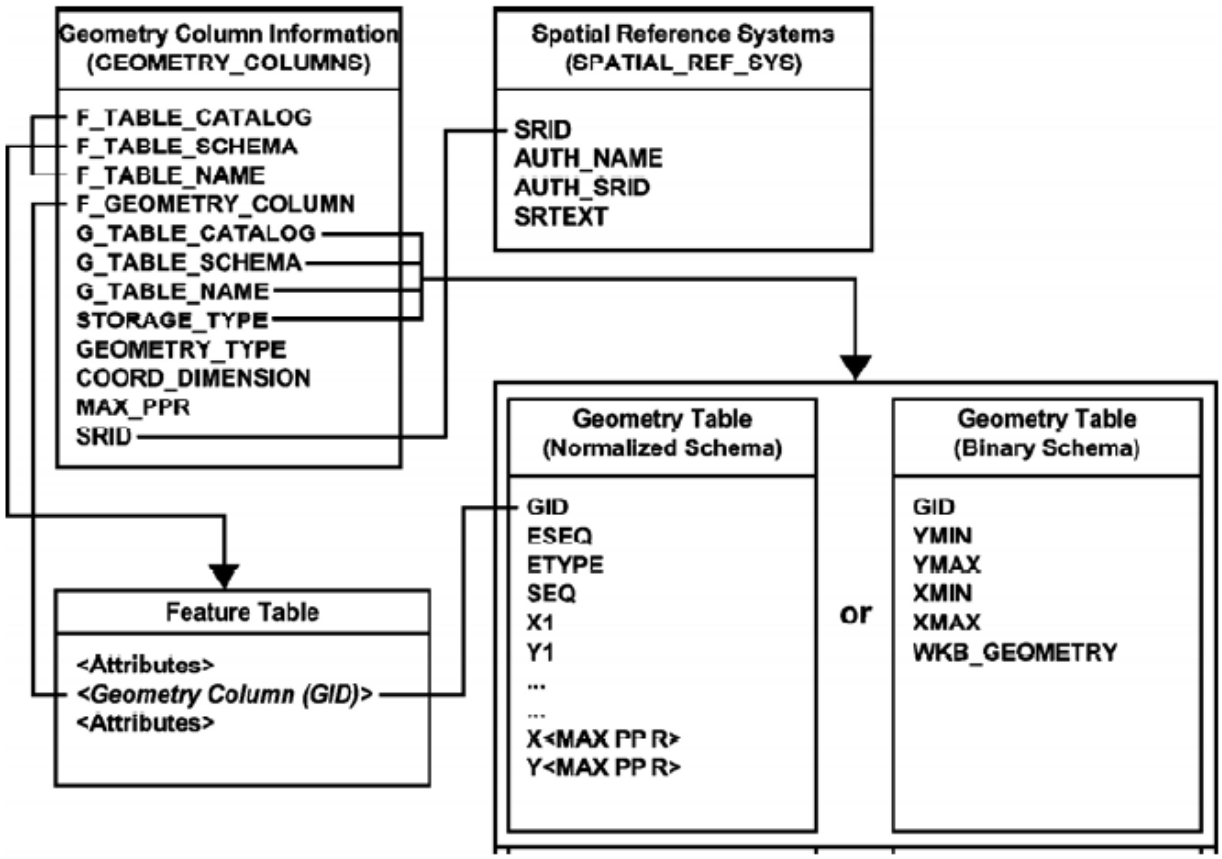
1. Tabulka GEOMETRY_COLUMNS popisuje dostupné feature tables a jejich geometrické (Geometry) vlastnosti.
2. Tabulka SPATIAL_REF_SYS popisuje souřadnicový systém a transformace tabulky Geometry.

3. Do tabulky FEATURE_TABLE se ukládají všechny prvky (features). Sloupce tabulky reprezentují atributy prvků a řádky identifikují vždy jeden prvek. Geometrie každého prvku je implementována jako cizí klíč v atributu *Geometry Column* pomocí klíče *GID*.
4. Tabulka GEOMETRY_TABLE ukládá geometrické objekty, které jsou jednoznačně identifikovány pomocí *GID*. Může být implemetována standardem SQL ve dvou typech:

   a) numeric

   b) binary

   Rozdíl mezi těmito dvěma reprezentacemi je v tom, jak získávají jednotlivé souřadnice. Starší, *Numeric Type*, musí opakovaně přistupovat do GEOMETRY TABLE, odkud získává jednotlivé souřadnice v podobě posloupnosti čísel. *Binary Type*, který je novější, ukládá geometrický objekt ve formátu WKB (Well-known Binary representation) jako jednu hodnotu.



Obr. 3.6: Schéma feature tables využívajících předdefinované datové typy – převzato z [OGC 2006]

.

Pro úplnost je nově uvedena reprezentace pomocí schématu SQL/MM, kde geometrické atributy prvků mohou být specifikovány pomocí tohoto rozšíření.

V dnešní době je tento formát podporován všemi známými společnostmi, které vytváří relační databáze podporující ukládání prostorových dat. Patří mezi ně např., Oracle, IBM DB2, IBM Informix, PostgreSQL, MYSQL, ad.

Formát Personal Geodatabase

Protože jsem se při praktické části mé diplomové práce zabýval zejména Personal Geodatabase, je nutné zmínit popis ukládání dat v tomto formátu. Jedná se o databázi, která není uložena na databázovém serveru, ale v binárním formátu *.mdb, který používá i aplikace Microsoft Access. Vzhledem k tomu, že je celá databáze obsažena v jediném souboru, má omezenou velikost, danou maximální velikostí souboru v daném operačním systému. Další nevýhodou je nemožnost paralelní editace více uživateli. Na druhou stranu má tato databáze spoustu předností, jako např., dostačující správa méně objemných dat, přenosnost, rychlost přístupu, vytváření topologie, aj.

Klady relačního databázového modelu

- Strukturám tabulky je jednoduché porozumět.
- Intuitivní, jednoduché uživatelské rozhraní.
- Mnoho nástrojů pro koncové uživatele (makra, skripty).
- Jednoduchá modifikace a přidávání nových vztahů, dat a záznamů.
- Poskytuje nezávislost dat z aplikace, dobrá pro jednoduché vztahy.
- Přímý přístup k datům zajišťuje rychlý a efektivní výkon.
- Nezávislost dat na aplikaci.
- Velké množství zkušených vývojářů, vývojových nástrojů, dokumentace a specialistů.

Zápory relačního databázového modelu

- Nejsou schopny pracovat na úrovni jednotlivého prvku, protože výsledkem je vždy tabulka.
- Problémem jsou složitá data, která mají proměnlivou délku – tato data se rozloží do mnoha tabulek – definice pak vznikne spojením tabulek (např. adresa) => pomalý přístup.
- Obtížné modelování složitých vztahů.

3.4 Objektově orientovaný databázový model

Pro pochopení objektově orientovaného databázového modelu se tato část nejdříve zaměřuje na obecnou charakteristiku, jak ji vysvětluje [Tomlinson 2003], pak následuje popis objektové struktury podle [Tomlinson 2003]. Uvedl jsem také jednu z možných prostorových reprezentací objektů na příkladu podle [Rigaux 2002] a nakonec klady a zápory objektově orientovaného databázového modelu. Pro znázornění jednotlivých komponent objektově orientovaného modelu byl použit jazyk UML (Unified Model Language), který bude popsán dále.

Objektově orientovaný databázový model umožňuje bohatý a komplexní popis reálného světa. Objekty mohou být modelovány jako entity reálného světa (např., kanály, ohně, lesy, budovy, vlastníci, atd.) a lze jim nadefinovat určitá chování, která je napodobují nebo modelují z hlediska jejich funkce v reálném světě. Jedním z rozdílů mezi relačním a objektovým databázovým modelem je, že objekt (viz dále) v sobě uchovává všechny své atributy a metody a je jednoznačně určen svým *oid* (object identifier), který je uložen v systému po celou dobu existence objektu. Další výhodou je možnost vytváření tzv. „Abstraktních datových typů“.

Abstraktní datové typy (ADT)

Abstraktní datový typ je (podle [McClure 1997]) datový typ, který je vytvořen kombinací

základních datových typů. Podpora ADT je široká, protože operace a funkce přiřazené novému datovému typu mohou být použity k indexaci, ukládání a obnově záznamů, založených na obsahu tohoto nového datového typu (např. multimédia nebo právě prostorové složky geografických dat).

3.4.1 Hlavní části objektově orientovaného databázového modelu

Tato kapitola byla zpracována podle [Tomlinson 2003].

Objekty (Objects)

Objekty reprezentují entity reálného světa (např., budovy, řeky, nebo i bankovní účty). Objekty mají vlastnosti, které definují jejich stavy, a metody, definující přechody mezi vztahy. Objekty mezi sebou komunikují zprávami, které vyvolávají určité chování.

Vlastnosti (Attributes, Properties)

Vlastnosti (jinak též atributy) definují stavy objektu, jako např., maximální dovolená rychlost určité silnice, název vlastníka budovy, atd.

Metody (Behavior, Methods)

Chování objektů nazýváme metodami nebo operacemi, které může objekt vykonávat. Např., ulice může spočítat zvestup času potřebný k jejímu překonání ve špičce nebo účet je chopen odečíst vyzvednutou sumu peněz. Toto chování může být též použito při poslání zprávy jinému objektu, aby změnil, uložil nebo spočítal hodnoty (vlastnosti) tohoto objektu.

Zprávy (Messages)

Objekty mezi sebou komunikují pomocí zpráv. Zprávy jsou činy jednoho objektu, které vyvolají určité chování druhého objektu. Zpráva se skládá z názvu objektu následovaným názvem metody, se kterou ví, jak pracovat. Objekt, který inicializuje zprávu se nazývá Odesílatel, zatímco objekt, který zprávu obdržel je Příjemce.

Třídy (Classes)

Třídou rozumíme seskupení objektů, které sdílí stejné vlastnosti a metody. Objekty jednotlivých tříd se označují jako instance těchto tříd. Třídy mohou být vloženy do určitého stupně hierarchie a dědičnost (viz. dále) zajistí jejich začlenění a přístup k dalším vlastnostem a metodám. Určování potřebných tříd je důležité při návrhu databáze.

Abstraktní třídy a metody

Abstraktní třídy jsou (podle [Mrozek 2006]) třídy, které nemohou vytvářet instance (objekty). Mohou být ale děděny a jejich potomci po implementaci potřebných metod instance vytvářet mohou. Metody označené jako abstrakní nemohou definovat obsah, pouze specifikují název a případné argumenty.

Vztahy (Relationships)

Vztahy popisují, jak jsou objekty mezi sebou propojeny. Definují pravidla pro vytváření, modifikaci a odstraňování objektů. Existuje hned několik druhů vztahů, které se používají

v objektově orientovaném databázovém modelu. Jsou to:

1. **Dědičnost (Inheritance)**

   Dědičnost dovoluje jedné třídě dědit vlastnosti a metody jedné nebo několika jiných tříd. Třída, která dědí se nazývá podtřída (subclass), „rodičovská“ (nadřazená) třída se nazývá supertřída (superclass). Kromě chování, které dědí, mohou podtřídy také přidat zděděné vlastnosti a metody. Dá se též říci, že supertřída je určité zevšeobecnění (generalizace) podtřídy a naopak podtřída je specializací supertřídy.

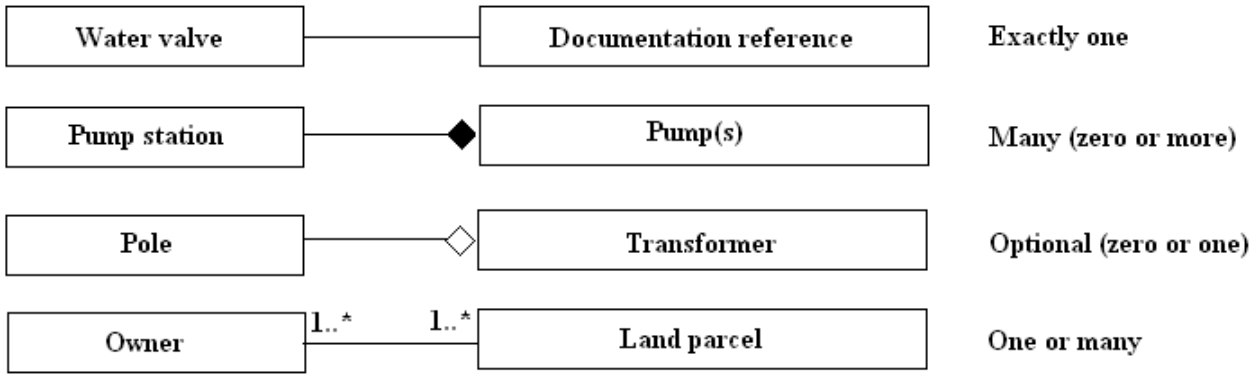
   Např., rodinný dům je specializace budovy a budova je zevšeobecněním rodinného domu. Třída dům pak může dědit vlastnosti a metody třídy budova, jako jsou: počet podlaží, pokojů nebo konstrukční typ.

2. **Asociace (Association)**

   Asociace je základním vztahem mezi objekty (viz Obr. 3.7). Mezi objekty existuje, stejně jako u relačního modelu, násobnost vazby (Multiciplity), která definuje, kolik objektů dané třídy je spojeno s objekty jiné třídy. Většinou se značí stejně jako v Tab. 3.1.

Multiplicity	Význam
1	pouze 1
0..1	0 nebo 1
M..N	od M do N
* or..*	od 0 do nekonečna
1..*	od 1 do nekonečna

Tab. 3.1: Vyjádření násobnosti vazby v objektově orientovaném databázovém modelu



Obr. 3.7: Asociace – zpracováno podle [Tomlinson 2003]

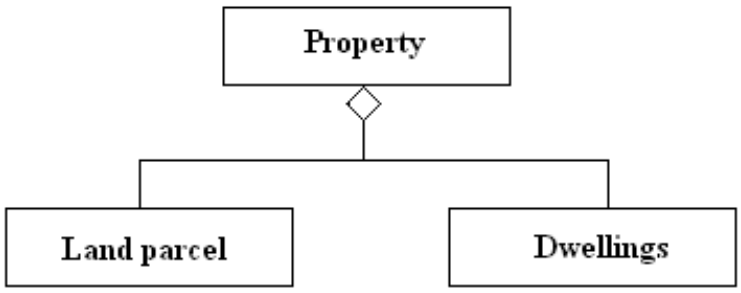
Následující typy vztahů (*Agregace* a K*ompozice*) jsou speciálními typy asociace.

a) **Agregace (Aggregation)**

   Objekty mohou obsahovat jiné objekty, proto je agregace jednoduše soubor odlišných objektových tříd uložených v tzv. agregační třídě (*Aggregate class*), která může vytvořit nový objekt. Tento nově vytvořený objekt je důležitý, protože

reprezentuje komplexnější strukturu než jednoduchý objekt.

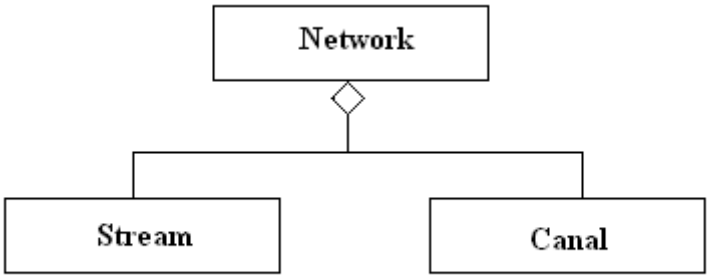
Na Obr. 3.8 může být agregační třída *Majetek (Property)* vytvořena agregací třídy *Parcela* (*Land parcel)* a *Obydlími (Dwellings)*.



Obr. 3.8: Agregace – zpracováno podle [Tomlinson 2003]

b) Kompozice (Composition)

Je silnějším asociačním vztahem než agregace, kde existenci podřízených objektových tříd řídí třída nadřízená. Např., budovy jsou složené ze základů, zdí a střech. Jesliže bude smazána třída budova, musí se automaticky smazat i třídy základy, zdi a střecha. Podobně je to na Obr. 3.9, kde městská *Síť vodního potrubí (Network)* se dělí na *Vodovod (Stream)* a *Kanál (Canal)*. Pokud bude smazána třída *Network*, musí být smazány také třídy *Stream* a *Canal*.



Obr. 3.9: Kompozice – zpracováno podle [Tomlinson 2003]

Zapouzdření (Encapsulation)

Zapouzdření je jádrem objektově orientovaného databázového modelu, kde objekty tohoto modelu zapouzdřují (schovávají) svoje vlastnosti a metody. Data obsažená v objektu mohou být zpřístupněna pouze prostřednictvím metod objektu. Zapouzdření tak chrání data před zničením jinými objekty a oddděluje tím vnitřní detaily objektu od systému.

3.4.2 Prostorová reprezentace objektově orientovaného modelu

Z hlediska GIS to vypadá tak, že v databázi jsou obvykle definovány základní třídy geografických objektů (bod, linie, polygon, rastr, ...), a od nich jsou pak pomocí dědičnosti odvozeny další, složitější a specializovanější třídy. Například, z primitivní třídy linie je oddděna

třída komunikace (má již specifické chování a atributy nutné pro komunikaci, jako je povrch, počet pruhů, …) a z třídy komunikace pak třída dálnice (jedná se o speciální případ komunikace se speciálními atributy a metodami). Díky objektově orientovanému řešení je obvykle řešena i problematika různé grafické reprezentace téhož objektu (například elektrické vedení bude jinak vizualizováno ve schématické sítí, než v mapě). Tato interpretace je od GIS v poslední době často požadována.

Příklad – administrativní jednotky (zpracováno podle [Rigaux2002]):

Class *Country*

tuple *(country_name:string,*

geometry: Region,

states: **set***(State))*

Class *State*

tuple *(state_name: string,*

country_in_state: Country, // Composition reference

geometry: Region,

counties: **set***(County))*

Class *County*

tuple *(county_name: string,*

populations: integer,

state_in_county: State, // Composition reference

geometry: Region)

Toto schéma popisuje dělení administrativních jednotek v USA na: Zemi (Country), Stát (State) a Okres (County). Např., objekt třídy *Country* je složený z objektů třídy *State.* Atribut *states* ve třídě *Country* má jako hodnoty objekty třídy *State*, přesněji – soubor odkazů na objekt (*oids*) třídy *State*. Stejně tak kraje států jsou odkázány pomocí atributu *counties* ve třídě *State*. Na druhé straně odkazují v hierarchii níže postavené třídy na vyšší jinými atributy. Např., třída *State* odkazuje na třídu *Country* pomocí atributu *country_in_state* a třída *County* odkazuje an třídu *State* pomocí atributu *state_in_county*. Takovým odkazům říkáme *„Composition references"*. K reprezentaci geometrie na všech úrovních hierarchie slouží atribut *geometry: Region*, jež se označuje jako prostorová třída (Spatial Class) a lze objektově zapsat např. takto:

Class *Region*

set *(***list** *(***tuple** *(x:***real***, y:***real***)))*

Příkladem systému je Smallworld (viz http://www.smallworld.co.jp/ff2005/index.html), využívaný správci inženýrských sítí, nebo Intergraph MG Dynamo (http://www.intergraph.com).

Klady objektově orientovaného databázového modelu

- Zapouzdření umožňuje přístup k datům objektu pomocí metod bez nutnosti znát vnitřní strukturu objektu.

- Umožňuje ucelenou a intuitivní reprezentaci reálného světa.
- Podporuje vícenásobnou úroveň zobecnění, seskupení a spojení.
- Udržuje historii v databázi.
- Dobře slučitelný se simulačními modelovacími technikami.
- Vícenásobné, paralení aktualizace (Versioning).
- Kvalitnější kód s méně systémovými chybami a nižšími náklady na údržbu.
- U složitějších datových typů zajišťuje vyšší stupeň datové integrity (vstup nových dat probíhá přes metody).

Zápory objektově orientovaného databázového modelu

- Komplexnější data jsou náročnější na návrh a vytvoření databáze.
- Obtížné propojení s relační databází.
- Import a výměna dat mezi jinými typy databází je obtížná.
- Některé aplikace nepodporují objektově orientovaný model.
- Velké a komplexní modely jsou náročné na výkon počítače.
- Z hlediska modelovaného problému je závislý na dokonalém popisu reálného světa (mimořádně obtížné).
- Objektově orientované databáze vyžadují pro analýzy objektově orientované programovací jazyky, které nejsou dodnes standardizované.

3.5 Objektově-relační databázový model

Převzato z: [Tomlinson 2003].

Tento databázový model využívá schopností objektově orientovaného databázového modelu, ale na rozdíl od něho ukládá data do relačních databázích. Relační databáze je rozšířena o software, který umožňuje modelu objektově orientované chování, ale data nejsou zapouzdřena jako u předchozího modelu. Data jsou stále uložena v tabulkách, ale některé atributy mohou obsahovat bohatší datové struktury (viz str. 21 - ADT).

Objektově-relační model má své výhody zejména v rychlosti (důležité ve velkých databázích) a schopnost ovládat databázovou integritu pomocí objektově orientovaného přístupu. To má za následek další výhodu v podpoře různých rozšíření pro SQL (Structured Query Language) a možnost přístupu přes standardní RDBMS. Tato výhoda je důležitá zejména v rozsáhlých podnikových systémech, kde ostatní obchodní aplikace potřebují přístup do geografické databáze.

Příkladem je Oracle Spatial Cartridge, ArcGIS a Geodatabase.

Klady objektově-relačního databázového modelu

- Rychlé výpočty.
- Jedno úložiště geografických dat; umožňuje využití odkazů a klasických databází.
- Rychlejší vstup a editace dat.
- Vysoká integrita dat (vstup nových dat probíhá přes metody).

- Uživatel může pracovat s více intuitivními datovými objekty.
- Pokles potřeby programových aplikací k modelování komplexních vztahů.

Zápory objektově-relačního databázového modelu

- Kompromis mezi objektově orientovaným a relačním modelem..
- Omezená podpora pro objektové vztahy.
- Komplexní vztahy jsou náročnější na modelování na rozdíl od výhradně objektově orientovaného databázového modelu.

V podstatě lze pro modelování geografických databází využít tři posledně zmiňované modely, proto se v další části mé diplomové práce již hierarchickým a síťovým modelem zabývat nebudu. Na Obr. 3.10 jsou shrnuté výhody a nevýhody jednotlivých modelů z hlediska definovaných standardů.

Srovnání databázových systémů			
Kriterium	**RDBMS**	**ORDBMS**	**ODBMS**
Definovaný standard	SQL2 (ANSI X3H2)	SQL3/4 (in process)	ODMG-V2.0
Podpora pro objektově orientované programování	Špatná; programátoři stráví 25% času kódování mapováním objektového programu do databáze	Omezená hlavně na nové datové typy	Přímá a rozsáhlá
Jednoduchost používání	Strukturám tabulky je jednoduché porozumět; mnoho dostupných nástrojů pro koncové uživatele	Totéž co RDBMS, navíc s nějakými matoucími rozšířeními	OK pro programátory; nějaký SQL přístup pro koncové uživatele
Jednoduchost vývoje	Poskytuje nezávislost dat z aplikace, dobrá pro jednoduché vztahy	Poskytuje nezávislost dat z aplikace, dobrá pro jednoduché vztahy	Objekty jsou přirozenou cestou k modelu; může vyhovět širokým rozsahem typů a vztahů
Rozšiřitelnost a obsah	Žádná	Omezená hlavně na nové datové typy	Může pracovat s libovolnou složitostí; uživatelé mohou psát metody a jakékoliv struktury
Složité datové vztahy	Pro model obtížné	Pro model obtížné	Může pracovat s libovolnou složitostí; uživatelé mohou psát metody a jakékoliv struktury
Výkon versus spolupracovatelnost	Úroveň bezpečnosti se mění s dodavatelem, je třeba vzájemně porovnat; dosáhnutí obého vyžaduje rozsáhlé testování	Úroveň bezpečnosti se mění s dodavatelem, je třeba vzájemně porovnat; dosáhnutí obého vyžaduje rozsáhlé testování	Úroveň bezpečnosti se mění s dodavatelem; většina ODBMSs dovoluje programátorům rozšířit funkčnost DBMS definováním nových tříd
Distribuce, replikace, a spojené databáze	Rozsáhlá	Rozsáhlá	Podle dodavatele; pár jich poskytuje rozsáhlou podporu
Vyspělost produktu	Velmi vyspělé	Nezralé; rozšíření jsou nová, stále se definují a jsou relativně neprozkoušená	Relativně vyspělé
Podpora pro lidi a univerzálnost SQL	Široká podpora nástrojů a trénovaných vývojářů	Může využívat výhod nástrojů RDBMS a vývojářů	Vybaveno SQL, ale určeno pro objektově orientované programování.
Softwarové ekosystémy	Poskytováno hlavními RDBMS společnostmi	Poskytováno hlavními RDBMS společnostmi	ODBMS výrobci začínají emulovat RDBMS výrobce, ale žádný nenabízí velký obchod jiným ISV
Životaschopnost výrobce	Očekávaná pro hlavní zaběhnuté RDBMS výrobce	Očekávaná pro hlavní RDBMS výrobce; UniSQL bojuje	Menší než se čekalo; stále se očekává zmenšování
Zdroj: International Data Corporation, 1997			

Obr. 3.10: Srovnání databázových modelů - převzato z [Batko 2007]

4 Existující metodiky pro modelování geografické databáze

Stejně jako se pro vytváření relačních databází využívají určité metodiky, lze obecně odvodit i několik návodů pro tvorbu geografických databází, které jsou v dnešní době převážně objektově-relačními. Vzhledem k tomu, že pro vytváření geografických databází nebyly vytvořeny žádné normy, bude tato kapitola vycházet spíše z vlastních zkušeností. Bude částečně odlivněna firmou ESRI, protože jsem se za dobu studia nejčastěji setkal s tvorbou „Personal Geodatabase". Protože je tato firma největší distributor GIS nástrojů, jedná se částečně i o vývojáře nových technologií tvorby geografických databází.První část kapitoly je zaměřena na popis grafického znázornění struktury geografické databáze, protože se jedná o jazyk, jakým návrhář komunikuje s klientem. Důležitou složkou při návrhu geografické databáze jsou prostorové vztahy (topologie), proto je nutné se tímto problémem také zabývat. Další část je pak zaměřena na obecný popis návrhu geografické databáze, jak ho popisuje [Vokounová 2003], která tuto část zpracovala podle [Longley 2001]. Dále se zaměřím na možnosti vytvoření Personal Geodatabase a na konci uvedu příklad části návrhu geomorfologické databáze.

V předcházející části, věnované databázovým modelům, byl zmíněn popis jednotlivých částí databáze z hlediska její struktury. V této kapitole se budu věnovat spíše modelování prostorových vztahů (geometrických a topologických).

4.1 Grafické znázornění (geo)databázových modelů

Grafickému znázornění jednotlivých databázových modelů je nutné věnovat podstatnou část návrhu databáze. Jedná se zde totiž o jazyk, kterým návrhář databáze komunikuje s klientem. Při správné vizualizaci problému by měl mít klient představu o tom, jak bude výsledná databáze vypadat a co bude dělat. Na druhé straně pomůže návrháři (návrhářům) při vlastní tvorbě tím, jak si práci rozvrhnout a čemu se vyvarovat.

Následující kapitola je rozdělena na dvě části. V první části se jedná o znázornění databázových modelů pomocí E-R diagramů. Tyto diagramy mají stále své využití při návrhu relačních databází kvůli své jednoduchosti a přehlednosti. V poslední době ovšem došlo k velkému rozmachu jazyka UML (Unified Modeling Language), který způsobil zvrat v přípravě návrhů jakýchkoli procesů (nejen databází). V této části je uvedena i poněkud odlišná notace UML firmy ESRI, které se budu více věnovat v dalších částech mé diplomové práce.

4.1.1 E-R modely

Entity-relationship(E-R) model znázorňuje primárně pouze entity a relace. Někdy se lze setkat ještě s označením entity-relationship-attribute (E-R-A) model, který navíc umožňuje zápis i atributů jak u entit, tak u relací. V zásadě jsou oba přístupy shodné, proto uvedu pouze stručný popis E-R modelu.

Pravidla pro kreslení typových E-R diagramů jsou jednoduchá. Entitní typy (relace) jsou reprezentovány obdélníky, vztahové typy pomocí kosočtverců. Hrany grafů ukazují, které entitní typy jsou zapojeny do jednotlivých typů vztahů. Každému uzlu grafu je přiděleno jméno.

Pojmenování vztahového typu může být někdy problém. Na Obr. 4.1 se díváme na vztah ve směru od studenta k předmětu. Zvolený název vztahu je v tomto směru srozumitelný. (daný STUDENT MÁ_ZAPSÁN daný PŘEDMĚT). Zde nám nic nebrání hledět na tento typ vztahu z druhé strany. Potom by vhodnější název pro tento typ vztahu byl JE_ZAPSÁN (daný PŘEDMĚT JE_ZAPSÁN daným STUDENTem). Někdy je vztah obtížněji pojmenovatelný. Často je vhodné zvolit neutrální identifikátor (např. —).

Obr. 4.1: Typový E-R diagram – převzato z [Pokorný 1999]

O E-R modelech bylo napsáno již dost literatury, proto se dalšímu popisu vyhnu. Navíc bych se již dosti vzdálil od tématu mé diplomové práce, proto zde uvedu pouze odkaz např. na : [Pokorný 1999].

4.1.2 Přístup pomocí jazyka UML

UML (Unified Modeling Language) je (podle [Page-Jones 2001]) jazyk zaměřený na vytváření objektově orientovaných databázových modelů (viz. str. 21).

Protože jsem při tvorbě geomorfologického informačního systému programoval v programu Visual Basic for Application (dále VBA), rozhodl jsem se pro částečný popis struktury systému ESRI. Protože je veškerý software firmy ESRI postaven na COM (Component Object Model), využívá ke znázorňění objektů, tříd a vztahů mezi nimi notaci jazyka UML s určitými modifikacemi. Znázorňuje tak zejména strukturu tzv. ArcObjects, pomocí níž jsou naprogramovány všechny programy této firmy, ale částečně i návrh databází. Pro pochopení COM je nutné tuto modifikaci také zmínit a vyhnout se tak následným rozporům ve vyjádření klasického a modifikovaného UML.

Protože se v UML dodržují určité konvence pro psaní názvů jednotlivých komponent, uvedu zde ty základní (zpracováno podle [Jedlička 2005]). Následovat bude popis jednotlivých komponent klasického UML, který se bude prolínat s modifikovanou ESRI notací UML.

Obecné konvence psaní velkých a malých písmen

Obecné konvence psaní velkých a malých písmen, které zde uvedu, se týkají podle [Jedlička 2005] konvencí při tvorbě geodatabáze firmy ESRI. Toto označení se ovšem dá rozšířit i na názvy některých komponent UML, které uvádí Page-Jones ve své knize [Page-Jones 2001].

Užívání velkých a malých písmen

1. Pro pojmenování prvků UML se používá jednotné číslo.
2. Pro oddělování jednotlivých slov v názvech komponent se v UML používá stejné politiky jako v Personal Geodatabase:
   - První velká (PPV) - „PrvniPismenaVelka“,
   - Všechna velká (VPV) - „VSECHNAPISMENAVELKA“,
   - Všechna malá (VPM) - „vsechnapismenamala“.

   V UML se objevuje ještě jedno označení názvů, které se v [Jedlička 2005] nevyskytuje, ale které jsem pro úplnost doplnil:
   - První malé, ostatní velká (PPMOV) - „prvniPismenoMaleOstatniVelka“

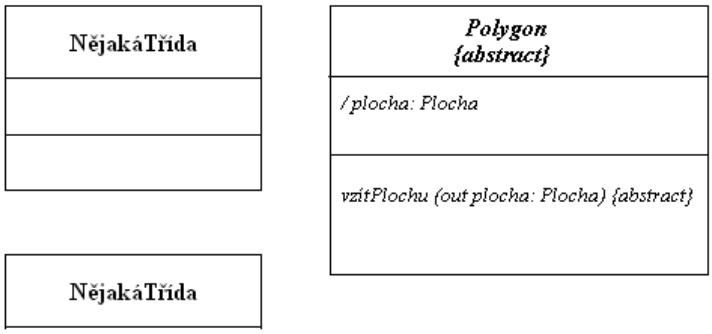
Znázornění jednotlivých komponent pomocí jazyka UML

Hlavní rozdíl mezi klasickým UML a ESRI notací UML je v tom, že v ESRI notaci se vůbec nesetkáme se znázorněním objektu. Klasické UML, které je obecnější, dovoluje i zakreslování objektů. U ESRI se objekty vytvářejí z tříd, proto není třeba objekty nějak zakreslovat.

Třída

Symbol třídy na Obr. 4.2 je základem jakékoli aplikace UML. Horní část symbolu zachycuje název třídy (NějakáTřída). Její název se řídí konvencí PPV (PrvniPismenaVelka). Je možné do této části přidat další informace o třídě na úkor přehlednosti diagramu. Prostřední část pak zachycuje atributy třídy a dolní část její metody. K těmto třem standardním částem lze přidat další, jimž lze přiřadit uživatelem definované názvy (např., konstanty a podmínky). Zkrácený symbol (viz Obr. 4.2 vlevo dole) je užitečný, chceme-li zobrazit jen třídu a její název v přehledovém modelu.

Speciálním případem znázornění třídy je znázornění abstraktní třídy. Na Obr. 4.2 vpravo je zachycen ***Polygon*** jako příklad abstraktní třídy. Název abstraktní třídy je zapsaný kurzívou a většinou se pod tímto názvem nachází označení ***{abstract}*** ve složených závorkách. Abstraktní metoda *vzítPlochu,* která zajistí výpočet plochy ohraničené daným tvarem polygonu, je také zapsána kurzívou a vlastnost *{abstract}* je opět ve složených závorkách za jejím názvem.



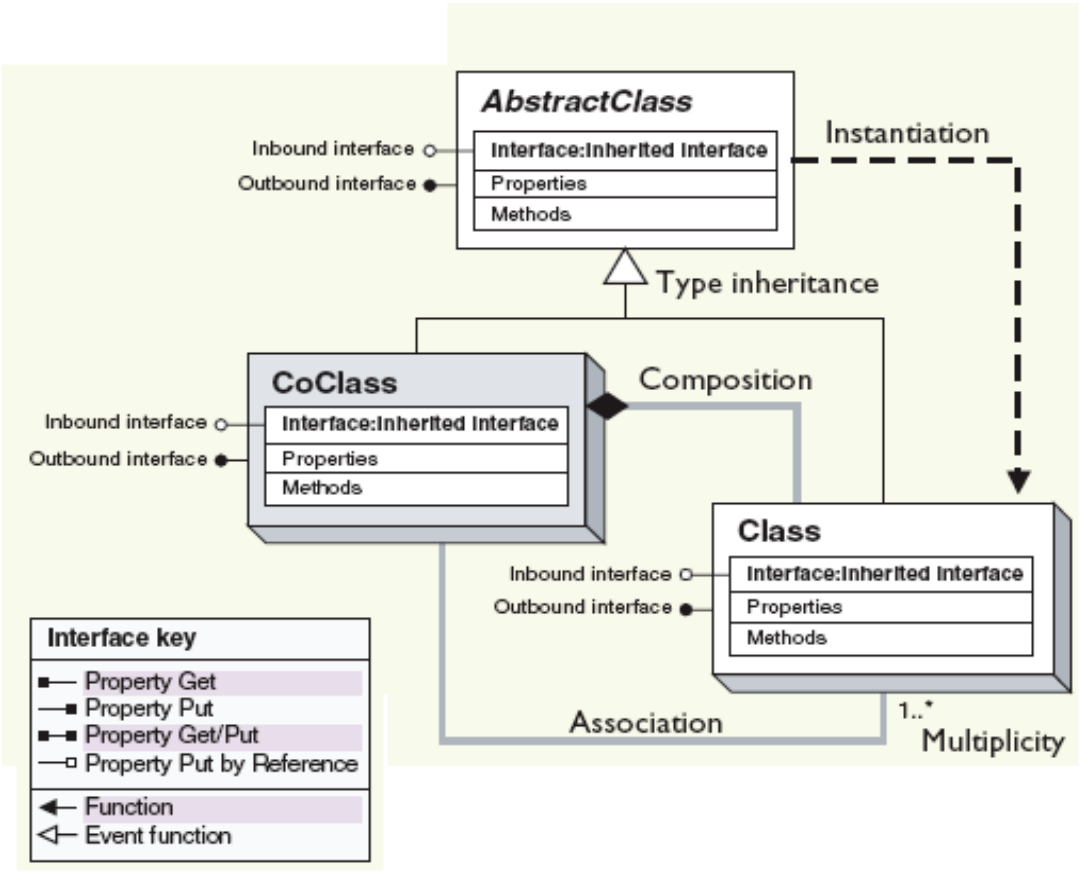
Obr. 4.2: Symbol třídy a abstraktní třídy podle klasického UML – zpracováno podle [Page-Jones 2001]

ESRI rozlišuje na rozdíl od klasického UML 3 typy tříd (viz Obr. 4.3).

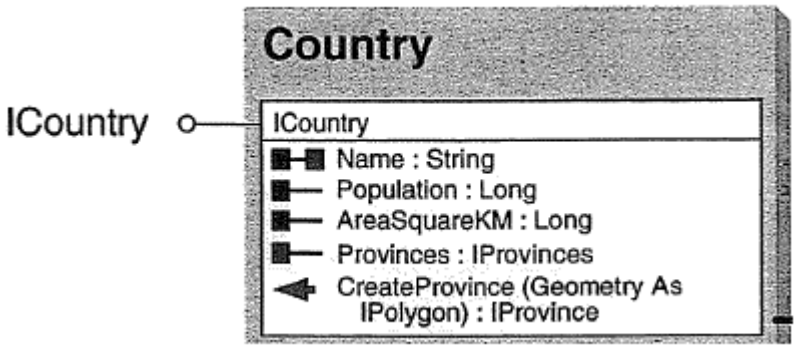
Abstraktní třída (AbstractClass) se znázorňuje stejně jako v klasickém UML. Značí se jako bezbarvá 2D tabulka.

Třída (Class) nemůže přímo vytvářet objekty, ale objekty mohou být vytvořeny pomocí vlastností nebo metod z jiných tříd. Značí se jako bezbarvá 3D tabulka.

CoClass je třída reprezentující objekty, která lze přímo vytvořit použitím objektové deklarace ve vývojovém prostředí. Značí se jako 3D obarvená tabulka. Ve VBA se obyčejně vytváří syntaxí: *Dim pFoo As New FooObject.*



Obr. 4.3: Znázornění tříd a vztahů mezi nimi podle notace ESRI - převzato z [Andrade et al. 2002]



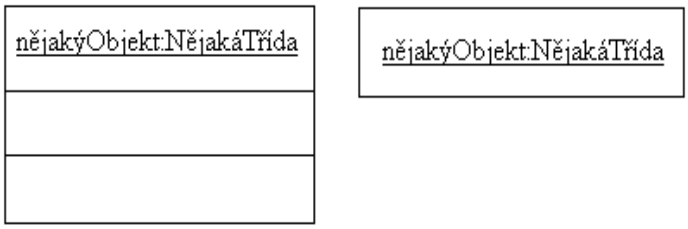
Obr. 4.4: Příklad třídy s atributy a metodami - převzato z [Tilton, et al. 2002]

Objekt

Obr. 4.5 Znázorňuje odpovídající symboly popisu objektu. Ze str. 21 víme, že objekt zrozený z nějaké třídy, obdrží strukturu, která je identická se strukturou dané třídy. Je tedy přirozené, že UML používá pro objekt stejný symbol jako pro třídu.

Největší rozdíl mezi zápisem objektu a jeho třídy je patrný v části názvu. Oproti stylu použitému pro název třídy je název objektu podtržený a není zapsán tučným písmem. Tyto typografické

rozdíly umožňují odlišení tříd od objektů. Navíc má syntaxe názvu objektu konvenci PPMOV (PrvniPismenoMaleOstatniVelka) - názevInstance:NázevTřídy. Ve většině kontextů návrhu, jako například když objekt odesílá jinému objektu nějakou zprávu, jsou objekty známy pod určitým názvem (např., **účetTohotoKlienta** nebo **levéKřídlo**). Existují ovšem i výjimky, které povolují anonymní název (např., **:NázevTřídy**).



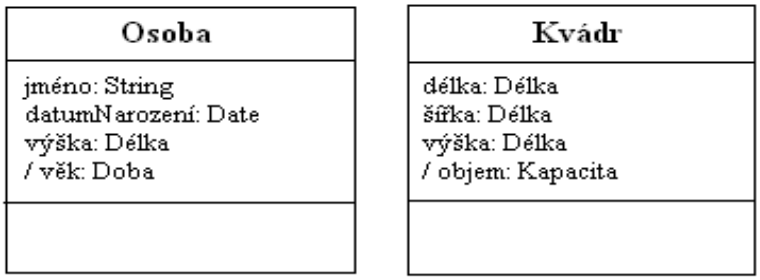
Obr. 4.5: Symbol objektu (instance třídy) ve své úplné a zkrácené formě pomocí klasického UML – převzato z [Page-Jones 2001]

Atribut (Property)

Nejdříve je nutné uvést, co to je a jak se značí vstupní (jinak též write, write-only, nebo Put), výstupní (jinak též read, read-only, nebo Get) a vstupně-výstupní (jinak též read-write, nebo Get-Put) atributy. Z názvu „Vstupní atribut" lze odvodit, že se jedná o atribut, který je určen pouze k zápisu, nikoli ke čtení. Názornější pro představu je spíše „Výstupní atribut", který je určen pouze pro čtení. Je to atribut, který nelze přímo měnit (přiřadit mu přímo jinou hodnotu – např., systémový čas). Jedinou možností, jak ho změnit, je, že v tomto objektu existuje metoda, která to umožňuje, nebo je tento atribut odvozený z jiného.

Existuje mnoho označení, ale já uvedu pouze značení klasického UML a ESRI notace UML. Názvy atributů mají opět konvenci PPMOV (PrvniPismenoMaleOstatniVelka). Na Obr. 4.5 je znázorněna třída **Osoba**, která má ve své střední části uvedeny atributy: **jméno**, **datumNarození** a **věk.** Po každém názvu atributu je za dvojtečkou uveden jeho datový typ. Před „Výstupními atributy" je vložen znak **„/"** např., **/ věk**. To samé platí i o druhém příkladu **Kvádr**, kde poslední atribut **/ objem** není přímo nastavitelný.

ESRI notace se odlišuje pouze vyjádřením dříve uvedených vstupních, výstupních nebo vstupně-výstupních atributů (viz Obr. 4.3 vlevo dole), pro které používá čtverečkové vyjádření (viz Obr. 4.4). Navíc odlišuje ještě speciální „Vstupní atributy", které lze zadávat pouze odkazem (Put By Reference).

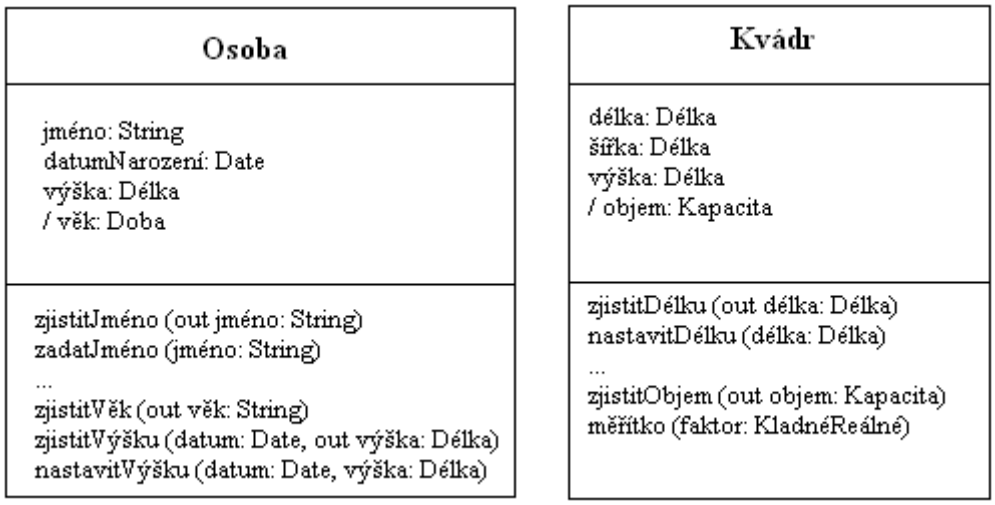


Obr. 4.6: Atributy podle klasického UML – převzato z [Page-Jones 2001]

Metoda (Method)

Na Obr. 4.7 se metody (operace) objevují v dolní části s plnými formálními podpisy. Každý formální podpis se skládá z názvu metody společně se seznamem formálních vstupních a výstupních argumentů dané operace. Standard UML požaduje určení směru každého argumentu klíčovými slovy **in** a **out**, přičemž označení **inout** určuje argument, který je zároveň **in** i **out**. Většinou se jedno nebo obě slova kvůli přehlednosti vynechávají (v uvedeném případě **in**). Opět se využívá konvence PPMOV (PrvniPismenoMaleOstatniVelka).

ESRI notace se odlišuje pouze vyjádřením funkcí, které je stejné jako u většiny známých programovacích jazyků: *NázevFunkce(parametr As TypParametru):TypFunkce*. Navíc je před každou funkcí použit symbol šipky (viz Obr. 4.4).



Obr. 4.7: Metody podle klasického UML – převzato z [Page-Jones 2001]

Rozhraní (Interface)

Podle [Andrade, et al. 2002] je rozhraní specifikací vlastností a metod, které umožňuje vysoký stupeň spolupráce a sdílení metod mezi objekty. Jinými slovy, rozhranní je balík vlastností a metod, který může být spojen s kteroukoli třídou. Výčet rozhraní v ESRI diagramu se nachází v první části

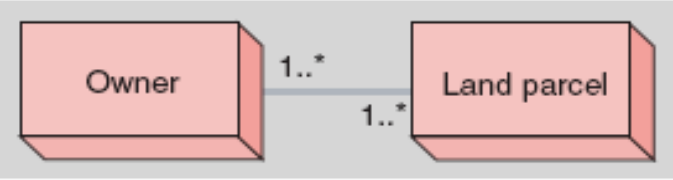
tabulky (viz Obr. 4.3). Mnoho CoClasses může implementovat stejné rozhraní. S tímto termínem je úzce spojen také polymorfismus (Polymorphism). Bližší informace lze nalézt např. v [Tilton, et al. 2002] nebo [Andrade, et al. 2002].

Vztahy

Označení vztahů klasického UML a ESRI notace UML je velice podobné, proto bude popsána pouze ESRI notace. Tam, kde je označení jiné, uvedu i klasickou notaci.

Associace (Association)

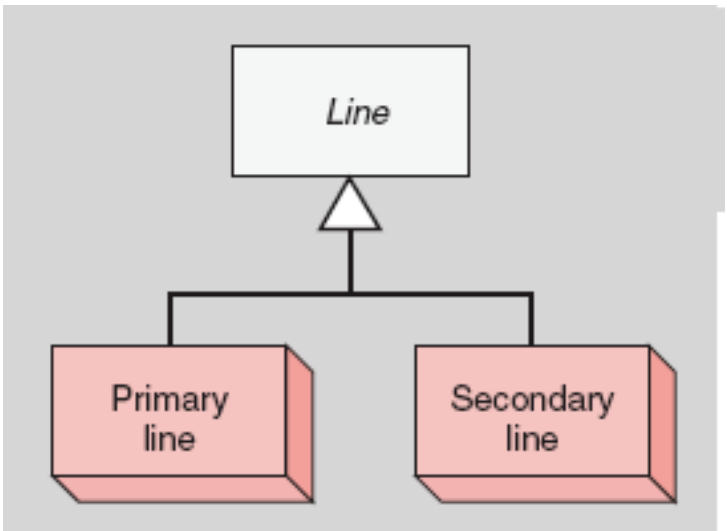
Podle 3.4.1 víme, že asociace reprezentuje vztahy mezi všemi typy tříd a definuje na obou koncích násobnost vazby (Multiciplity). Na Obr. 4.8, vlastník (Owner) může vlastnit jednu nebo více parcel, a parcelu (Land parcel) může vlastnit jeden nebo více vlastníků.



Obr. 4.8: Asociace v ESRI notaci – převzato z [Andrade, et al. 2002]

Dědičnost (Inheritance)

Jak bylo uvedeno dříve (viz 3.4.1), dědičnost definuje specializované třídy (podtřídy), které sdílejí vlastnosti a metody supertříd a přidávají svoje vlastní (většinou se čte jako: podtřída *je typem* supertřídy). Na Obr. 4.9 je *Primary (Secondary) Line* (CoClasses) typem *Line* (Abstract Class).



Obr. 4.9: Dědičnost v ESRI notaci - převzato z [Andrade, et al. 2002]

Konkretizace (Instantiation)

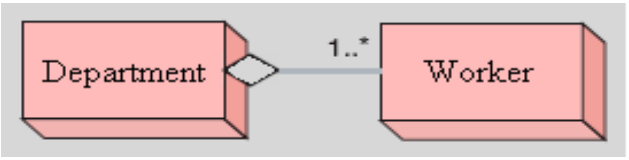
Konkretizace se v klasickém UML vůbec nevyskytuje, proto zde uvedu pouze ESRI notaci. Konkretizace specifikuje, že jeden objekt nějaké třídy má metody, kterými může vytvořit objekt jiné třídy. Na Obr. 4.10 má objekt *sloup (Pole)* metody pro vytvoření objektu *transformátor*

(Transformer). Tento typ spojení může propojovat CoClasses mezi sebou (jako na obrázku) nebo CoClass s Class.

Obr. 4.10: Konkretizace v ESRI notaci - převzato z [Andrade, et al. 2002]

Agregace (Aggregation)

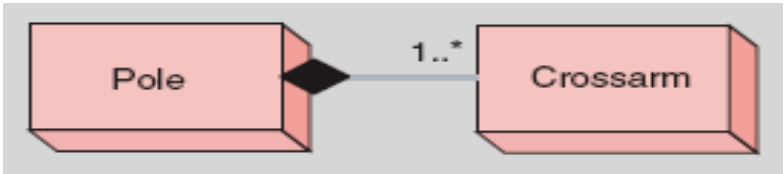
Podle 3.4.1 je agregace soubor odlišných objektových tříd uložených v tzv. agregační třídě (*Aggregate class*), která může vytvořit nový objekt. Na Obr. 4.11 pracují *pracovníci (Worker)* v určitém *oddělení (Department)*. Přestože budou všichni pracovníci propuštěni, bude oddělení stále existovat. To je rozdíl mezi kompozicí a agregací.



Obr. 4.11: Agregace v ESRI notaci

Kompozice (Composition)

Podle 3.4.1 je kompozice silnější formou agregace, kde objekty jedné třídy kontrolují životnost objektů jiné třídy. Na Obr. 4.12 obsahuje *sloup (Pole)* jeden nebo více *konzolí (Crossarms)*. Při vymazání *sloupu* nemůže existovat ani jedna *konzole. Sloup* řídí životnost *konzole*. Tento typ spojení se uskutečňuje většinou mezi CoClasses.



Obr. 4.12: Kompozice v ESRI notaci - převzato z [Andrade, et al. 2002]

4.2 Prostorové vztahy - topologie

Důležitou složkou při návrhu geografické databáze je možnost uložení topologie. Nejprve popíši topologii obecně (matematická topologie) a následně význam topologie v GIS.

Při konkrétním návrhu topologie záleží, v jakém systému je databáze navrhována. Protože jsem se při praktické části diplomové práce setkal s tvorbou topologie firmy ESRI, popíši v článku její strukturu a porovnám s jiným přístupem – systémem Oracle.

4.2.1 Matematická topologie

Topologie obecně

(z řeckého topos - místo a logos – studie)

Vlastnosti prostoru můžeme rozdělit např. podle [Ullmann 1983] na:

- kvantitativní (metrické) - související s měřením vzdáleností, úhlů, ploch
- kvalitativní (topologické) - související pouze se vzájemnými vztahy objektů

Některé definice topologie:

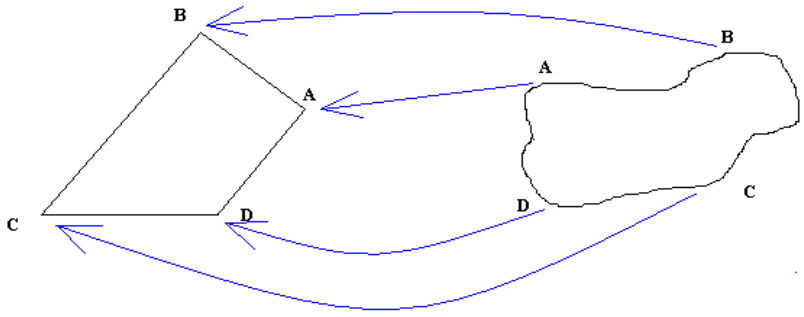
- úsek geometrie zkoumající vzájemný vztah polohy geometrických útvarů [Příroda 2006],
- matematická disciplína, studující prostorové vlastnosti množin [Swiki 2007],
- kvalitativní geometrie [Ullmann 1983].

Topologie se vyskytuje ve všech oborech matematiky (zjména v teorii grafů). Kvalitativní geometrii (topologii) lze chápat jako geometrii bez velikostí. Zabývá se vnitřními vzájemnými vztahy bodů, množin a objektů. Studuje vlastnosti geometrických objektů, které se při různých deformacích nemění. Základy topologie definují a studují vlastnosti prostorů jako je souvislost, spojitost, počet rozměrů, omezenost, neomezenost a pod.

Topologii nezajímá přesný tvar objektů, ale spíše to, jak jsou objekty spojeny dohromady. Například, čtverec a kružnice mají společné vlastnosti: jsou jednorozměrné a dělí prostor na plochu uvnitř a vně – jsou tudíž topologicky ekvitalentní. Topologie je proto obecnější stránkou geometrie.

Topologické zobrazení

Topologické zobrazení (viz Obr. 4.13) je takové zobrazení, při kterém se obecně nezachovávají úhly ani délky. Dochází při něm ke změně tvaru geometrických útvarů, což se označuje jako topologická transformace (někdy se také hovoří o topologické deformaci). Při topologické transformaci se zachovává příslušnost (incidence) bodu k dané křivce a také uspořádání bodů na křivce. Z toho vyplývá, že při topologické transformaci zůstává uzavřená křivka uzavřenou a neuzavřená se transformuje opět na neuzavřenou křivku. Také poloha bodu vzhledem k uzavřené křivce (tzn. zda je uvnitř nebo vně křivky) se také zachovává.



Obr. 4.13: Topologické zobrazení

Při topologické transformaci dochází pouze k deformaci útvarů, nikoli však k přerušení nebo vytvoření nových hran.

Homeomorfismus

Intuitivně lze dojít k tomu, že dva prostory jsou topologicky ekvivalentní, když jeden z nich lze různými deformacemi převést na druhý bez dělení nebo spojování jejich hranic. Pod pojmem homeomorfní útvary si lze představit například kružnici, elipsu, čtverec nebo trojúhelník, které lze pomocí topologického zobrazení deformovat vzájemně mezi sebou (např. z kružnice lze deformacemi získat velmi jednoduše elipsu, ale také čtverec i trojúhelník). Samozřejmě jsou si ekvivalentní různě velké útvary stejného typu (různě velké poloměry kružnice, nebo různě velké čtverce).

4.2.2 Topologie v GIS

Převzato z [Glos 2006], [ESRI 2004] a [Hoel 2001].

1) Obecně prostorová topologie

Prostorovou topologií, též databázovou topologií v GIS, rozumíme definici prostorových vazeb mezi objekty uvnitř geografické databáze. Prostorová topologie zajišťuje integritu prostorové složky objektů, které spolu souvisejí. Představme si plánek místností patra budovy. Sousedící místnosti mají obvykle alespoň jednu svoji stěnu společnou. Pokud tuto stěnu posuneme, jistě dojde ke změně v půdorysu obou místností a to tak, že společný půdorys zůstane zachován. Stejně tak pokud změníme polohu celé budovy, jistě očekáváme, že se odpovídajícím způsobem změní i poloha jednotlivých místností. Jak bylo naznačeno, pro definici prostorové topologie se využívá jak prostorových vztahů mezi objekty (sousedící místnosti) tak i vztahů definovaných hierarchií objektů (místnost je definována pomocí stěn místnosti, patro budovy je definováno pomocí místností budovy v tomto patře).

A k čemu slouží prostorová topologie? Dovoluje zachytit prostorové vazby mezi objekty a pomáhá udržovat správnou prostorovou lokalizaci objektů. Dále umožňuje provádět analýzy např. typu: „vyhledej vlastníky pozemků sousedících s pozemky vybranými pro stavbu sjezdovky“.

2) Topologie v různých systémech

Správa a ukládání topologie je důležitou složkou GIS pro udržení konzistence dat v databázi. Mezi největší přednosti využití topologie patří podle [Baars, et al 2004]:

- vyhnutí se redundantnímu ukládání dat (více kompaktní než uložení celé geometrie),
- jednodušší správa konzistence dat po editaci,
- je to přirozený model pro některé aplikace,
- efektivní při vizualizaci a dotazovacích operacích (např. sousedství),
- detekce topologických chyb

Podle [Baars, et al 2004] existují dvě různé implementace pro ukládání topologických struktur. První z nich se nazývá *rule-based topological structure* a je zastoupena firmou ESRI. Druhou implementaci využívá většina objektově-relačních databází a nazývá se *explicit topological structure*.

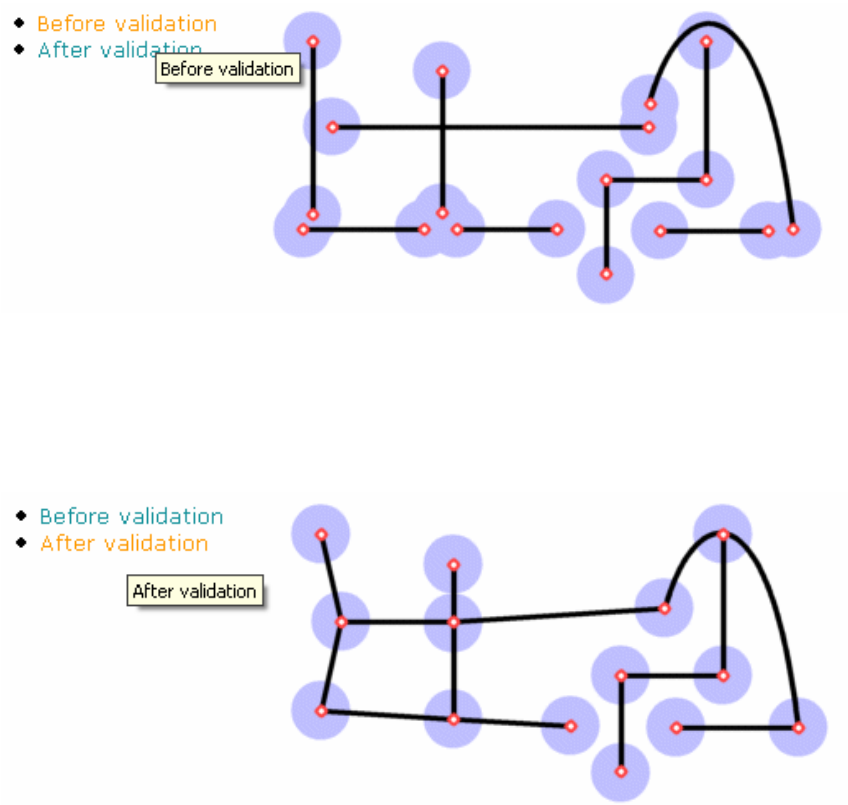
a) ESRI topologie

Největším rozdílem mezi tvorbou ESRI topologie a topologií ostatních systémů (např. Oracle) je,

že jednotlivá topologická primitiva nevytváří nové (speciální) geoprvky (features). Vyplývá z toho, že původní geometrie je zachována a topologická primitiva jsou vytvořena přímo nad touto geometrií.

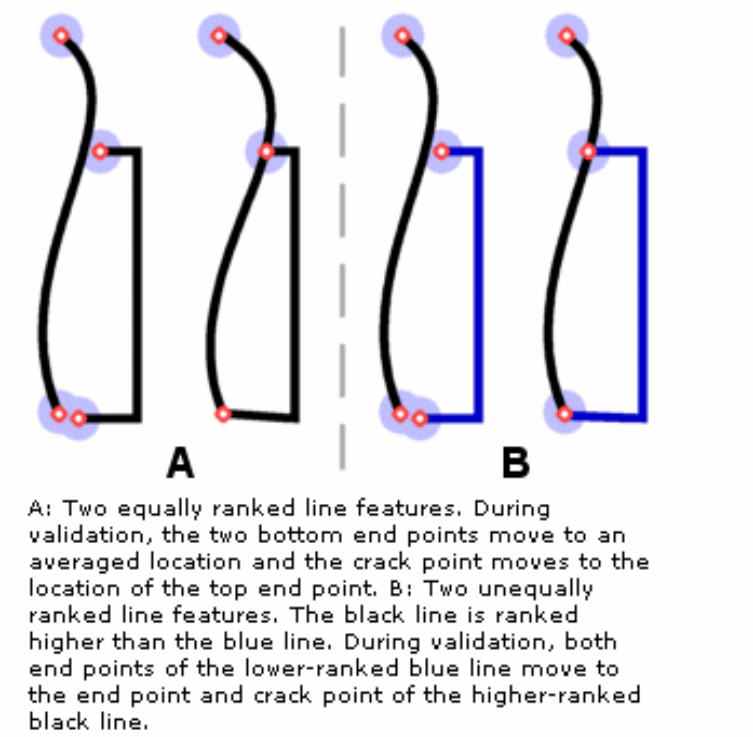
Samotná topologie je pak definována sadou topologických pravidel. Topologická pravidla se využívají k definování omezení přípustných topologických vztahů mezi geoprvky v jedné nebo více prvkových třídách (feature class) uvnitř datové sady (datasetu). Topologická pravidla jsou vybrána na základě toho, které topologické vztahy jsou pro uživatele nejdůležitější a poté uložena v topologii. V dnešní době existuje 25 topologických pravidel, ze kterých si může uživatel vybrat potřebný počet pro jednotlivé prvkové třídy. Jejich výčet lze nalézt např. na: [ESRI 2007b].

Mimo jiné lze u ESRI topologie nastavit tzv. *Cluster tolerance*, což lze přeložit jako shluková tolerance. Jedná se nastavení okolí kolem každého vrcholu (Vertex) u liniových a polygonových vrstev jednotlivých prvků (features), ke kterým se přichytí jiný prvek, který se nachází v tomto okolí. Pro názornost viz Obr. 4.14.



Obr. 4.14: Princip cluster tolerance - převzato z [ESRI 2004]

Další možností, kterou lze v ESRI topologii zadat při vstupu jednotlivých prvkových tříd (feature classes) do topologie je tzv. *Set Ranks*, přeloženo jako nastavení pořadí. Možný popis je takový, že pořadí určuje relativní stupeň polohové přesnosti. Jinými slovy lze říci, že určuje, který prvek se přichytne ke kterému. Pro lepší představu je uveden opět obrázek.



Obr. 4.15: Princip set ranks v ESRI topologii - převzato z [ESRI 2004]

Proces kontroly (validation process) je základní operací nad topologií, kterou vykonávají topologické nástroje. Tímto procesem se všechna vytvořená pravidla uvedou v platnost a vygenerují se topologické chyby v místech, kde byla pravidla porušena. Není nutno kontrolovat vždy všechny prvky prvkových tříd a stačí definovat pouze část prostoru, kde chceme data zkontrolovat.

Podmínky vytváření topologie a pravidel

- topologie musí být ve stejném datasetu jako prvkové třídy, kterých se týká,
- lze kdykoli přidat či odebrat pravidlo nebo celou topologii,
- dataset může obsahovat více topologií,
- pravidlo může existovat mezi jednou nebo dvěma prvkovými třídami,
- k prvkové třídě může být připojeno 0, 1 i více pravidel,
- lze definovat prostor v prvkové třídě, kde má probíhat kontrola topologie,
- topologie existuje, je-li zkontrolována (validated).

b) Topologie v systému Oracle

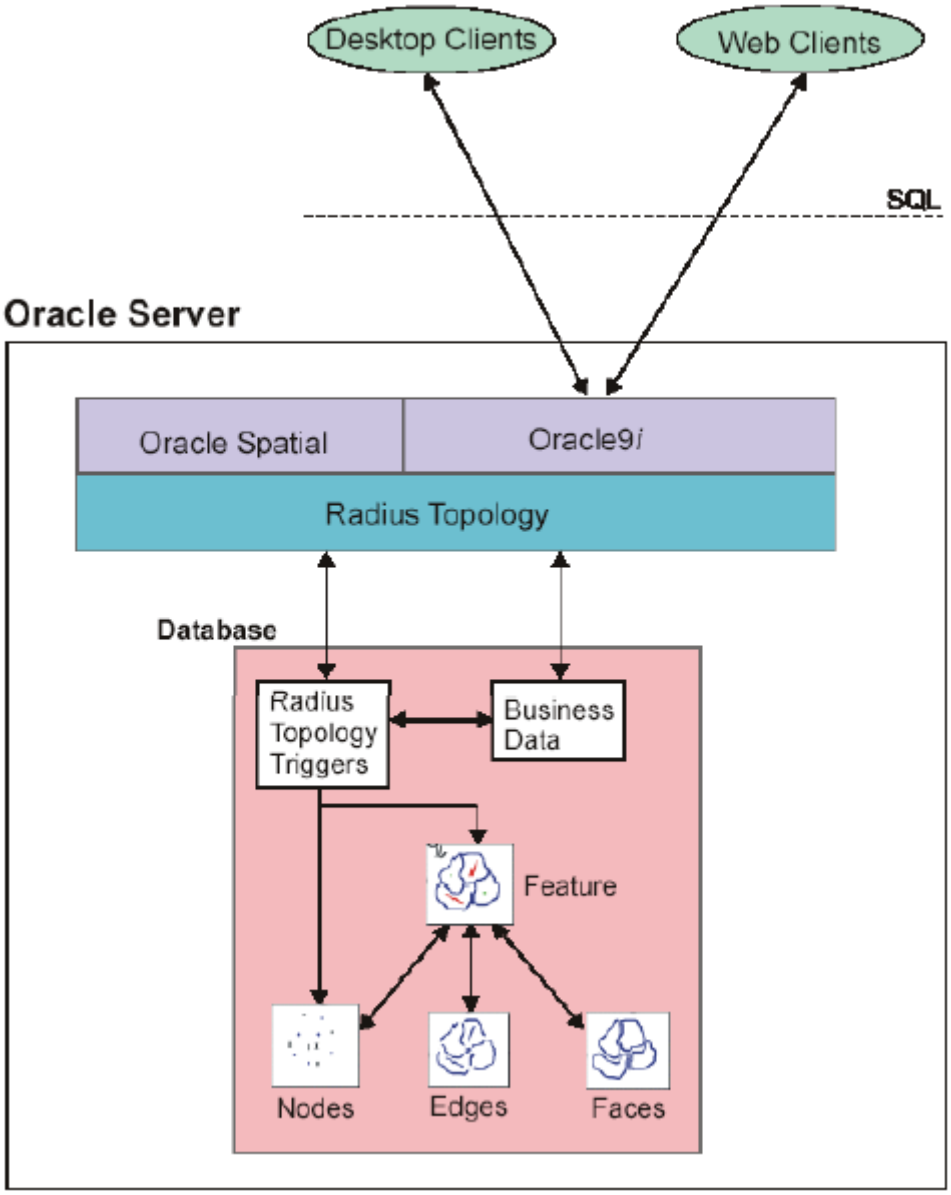
V dnešní době existují pro správu topologie v systému Oracle dva moduly. Jedná se o modul *SDOTOPO*, který byl vytvořen přímo firmou Oracle, a *Laser-Scan Radius Topology* (http://www.laser-scan.com), který se začíná nově prosazovat. *SDOTOPO* je modul, který je přítomen přímo v databázi Oracle Spatial a udržuje v ní topologicky čistá data. Jeho rozšíření *Radius Topology* má již více možností z hlediska editace a správy prostorové složky dat. Z hlediska ukládání topologie jsou oba moduly velice podobné a k pochopení rozdílu od ukládání topologie

firmou ESRI stačí popsat pouze jeden z nich.

Laser-Scan Radius Topology

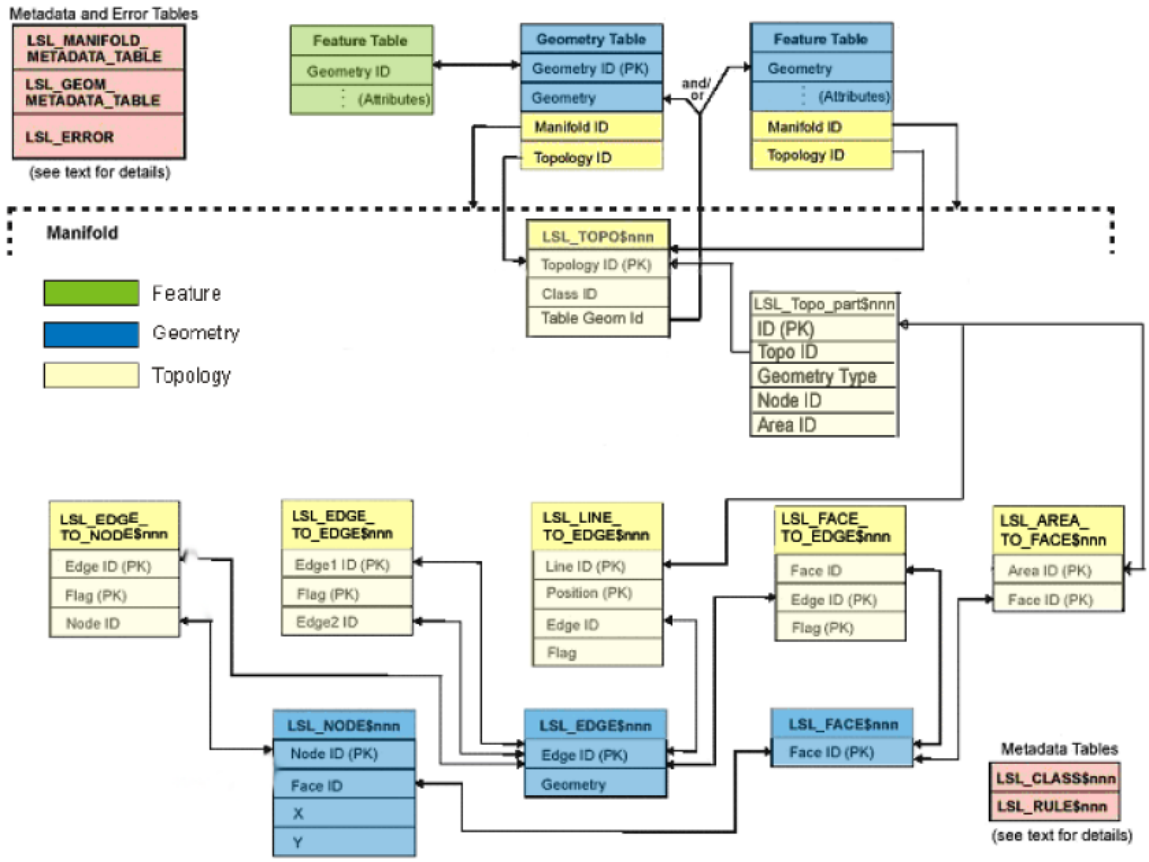
Jak již bylo řečeno, je koncept Laser-Scan Radius Topology (viz Obr. 4.16) poněkud odlišný od ESRI Geodatabase.

Klasická data jsou v DBMS uložena jako celistvá, ale Radius Topology je schopen tato data bez topologie převést do souboru topologických primitiv a vztahů mezi těmito primitivy. Primitiva jsou zde uložena v klasických relačních tabulkách, které jsou propojeny právě topologickými vztahy. Na Obr. 4.16 lze nalézt Radius Topology Triggers, které zabraňují vytvoření topologických chyb při editaci.



Obr. 4.16: Postavení LaserScan Radius Topology v DBMS Oracle – zpracováno podle [Louwsma 2003]

Prvkové tabulky (feature tables) a geometrické tabulky (geometry tables), které již v databázi existují, mohou být převedeny do topologicky strukturovaných datových tabulek. Během tohoto procesu jsou vytvořeny v původních tabulkách (feature a geometry) primární klíče manifold ID a Topology ID. Pomocí těchto ID se lze odkazovat na tabulky vytvořené Radius Topology. Struktura tabulek vytvořených pomocí Radius Topology je vidět na Obr. 4.17.



Obr. 4.17: Struktura tabulek modulu Laser-Scan Radius Topology - zpracováno podle [Louwsma 2003]

Občas je nutné vytvoření a udržení nezávislosti sad topologických vztahů v rámci jedné datové sady. Příkladem může být kombinace reálných hranic a administrativních hranic. Řešení zde zajišťují tzv. *manifolds*, které vytváří určité uspořádání nezávislých sad vztahů. Každý manifold má svoje ID. Bližší informace lze nalézt např. v: [Louwsma 2003], [Baars, et al 2004].

c) Porovnání obou přístupů

Zpracováno podle: [Baars, et al 2004].

Výhody topologie ESRI Geodatabase:

- Topologická struktura není explicitně uložena. Uživatel může rozhodnout o tom, které pravidlo je pro něho důležité.
- Uživatel rozhodne, co se má stát s prostorovými objekty, když nevyhovují všem podmínkám. Topologické chyby nejsou řešeny automaticky, ale manuálně uživatelem.

- Proces kontroly (validation process) dat trvá relativně krátkou dobu. V porovnání s Radius Topology nezabere kontrola datové sady tolik času. Musí být ovšem poznamenáno, že chyby jsou pouze označeny a nejsou řešeny.
- Po editaci určitých oblastí nemusí být kontrolována celá datová sada. Editovaná místa jsou označena jako dirty areas a lze kontrolovat pouze tato místa.
- Topologii ESRI Geadatabase je jednoduché použit. K použití topologie není potřeba speciálních znalostí. Použití průvodce při vytvoření topologických pravidel je intuitivní.

Nevýhody topologie ESRI Geodatabase:

- Horší udržitelnost integrity. Existuje redundance - v polygonových vrstvách jsou všechny sousedící hrany uloženy dvakrát.
- Pokud se objeví objekt, který nevyhovuje podmínkám, uživatel si musí být vědom, že to neznamená existenci topologické chyby. Je možné, že podmínka není definována správně.
- Soubor nástrojů v ArcGIS k řešení topologických chyb nemá moc rozšíření. Některé chyby sice mohou být vyřešeny, ale hodně chyb vyžaduje pokročilejší nástroje.

Výhody Radius Topology

- Topologie je uložena explicitně. Uživatel může přidat pouze topologicky čistá data, nebo data, která lze automaticky opravit.
- Po vytvoření topologické struktury dat se uživatel nemusí o topologii již více starat. Toho může být využito např., když společnost prodá datovou sadu zákazníkovi. Zákazník a prodávající jsou si jisti, že topologie je v pořádku.
- Stále je možné se dotazovat do databáze přes DBMS pomocí jazyka SQL .
- Navzdory pevné struktuře Radius Topology, lze definovat některé elementy topologické struktury v manifoldu - obdoba set ranks a cluster tolerance. Např. uživatel může definovat, který topologický model využívá, jaké priority přiřadil objektu (který objekt se přiřadí (snap) kterému), může definovat toleranci, aj.
- Linie je uložena v databázi pouze jednou.

Nevýhody Radius Topology

- Kontrola topologie je časově náročná, ale vyváří se pouze jednou.
- Zabírá mnoho místa na disku.
- Elementy topologické struktury, které lze definovat uživatelem jsou limitovány.

4.2.3 Shrnutí topologie

GIS topologie se nezabývá stavbou geometrie pouze jednoho objektu uvnitř jedné vrstvy, ale řeší i vzájemné vztahy mezi objekty více vrstev. Právě tato možnost poskytuje GIS nové možnosti uplatnění analýz, jako je např. řešení sousednosti. Další nespornou výhodou GIS topologie je udržení konzistence geografické databáze.

Rozlišují se dvě různé implementace topologie – *Rule-based* a *Explicit*. Každá implementace má svá pozitiva a negativa. Pokud požadujeme pouze uchování topologicky čistých dat bez změny geometrie, je vhodné použít *Explicit* implementaci. Pokud na začátku víme, že geometrie objektů se

bude často měnit, je lepší použít *Rule-based* implementaci. Samozřejmě při tom záleží také na systému, v kterém budou výsledná data uložena. Nejlepším řešením, které ovšem často není možné je kombinace obou implementací. Příkladem může být kombinace ESRI ArcSDE Geodatabase propojená s databází Oracle, kdy se naimportovaná data nejdříve opraví topologickými nástroji firmy ESRI a topologicky čistá data se pak uloží do databáze Oracle.

4.3 Návrh geografické databáze

Velmi názorný popis návrhu geografické databáze sepsala ve své diplomové práci Lucie Vokounová (viz [Vokounová 2003]), který obecně platí pro návrh jakékoli databáze. Speciálnější popis návrhu databáze, zaměřená přímo na ESRI Personal Geodatabase, pak sepsali Arctur a Zeiler v knize [Arctur 2004] - viz Obr. 4.18. Protože se v základu oba dva přístupy neliší, uvedu první, obecnější způsob návrhu.

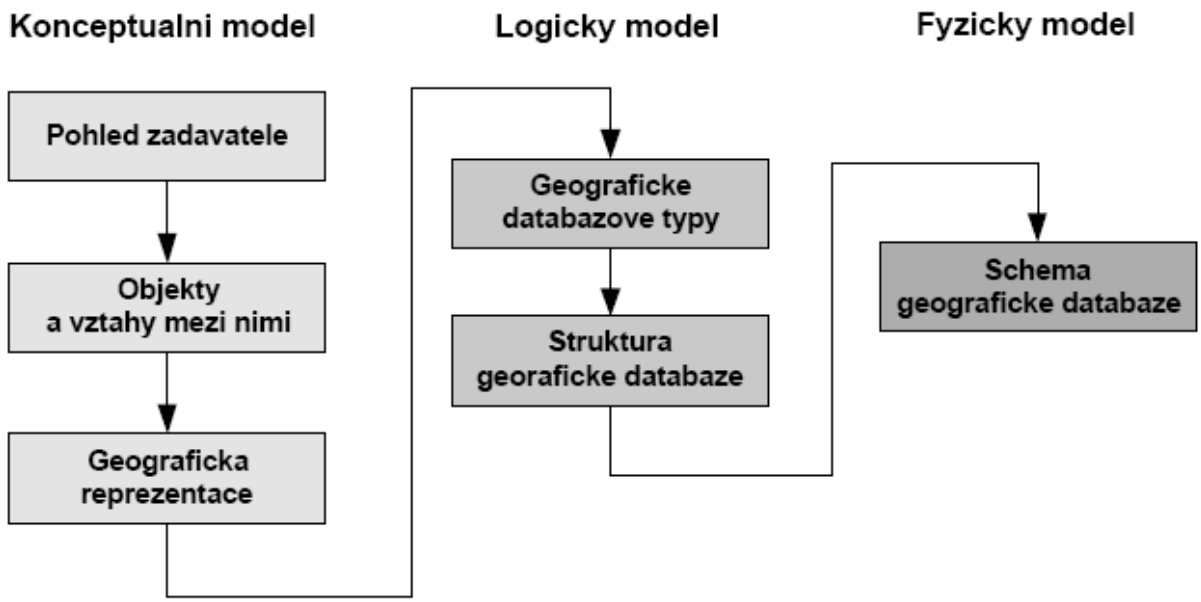
Obr. 4.18: Popis návrhu Personal Geodatabase – převzato z [Arctur 2004]

Protože jsou (podle [Vokounová 2003]) v geografických databázích (dále GDB) uloženy především prostorové objekty, které představují objekty reálného světa, probíhá návrh GDB pomocí

objektové analýzy. Objektová analýza se obecně skládá ze tří hlavních nástrojů: *dekompozice, abstrakce, specializace v hierarchii*.

Samotný návrh geografické databáze prochází třemi stupni vývoje. Postupně, jak dochází ke konkretizaci problému, jsou vytvořeny tři modely – **konceptuální, logický a fyzický**, pomocí výše uvedených nástrojů.

Fyzický model je konečným produktem návrhu. Obsahuje již specifikované datové typy a vztahy. Svým sestavením již odpovídá budoucí databázi. Při cestě k fyzickému modelu databáze bychom měli projít následujícími šesti kroky (viz Obr. 4.19):



Obr. 4.19: Postup při návrhu GDB - převzato z [Vokounová 2003]

4.3.1 Konceptuální model

Databázový model umožňuje (podle [Ressler 2006]) zobrazit a popsat objekty v databázi a vztahy mezi nimi z hlediska jejich významu a chování. Výsledkem konceptuálního modelování je implementačně nezávislé databázové schéma, tj. schéma obecně aplikovatelné v jakémkoli technicko-programovém prostředí. Nejčastěji se znázorňuje v podobě diagramu případů použití (Use Case), který definuje požadavky na funkcionalitu systému, a datového diagramu (E-R diagramu nebo diagramu tříd), který definuje třídy prvků, jejich atributy a vztahy mezi nimi.

Pohled zadavatele

V tomto první kroku se (podle [Vokounová 2003] a [Longley 2001]) stanoví základní funkce, které by měla výsledná databáze plnit. Zároveň se určí druh dat, která budou sloužit k jejímu naplnění. Následně se tato data roztřídí do skupin, jakýchsi budoucích zárodků tříd. Pro toto první přiblížení stačí jen papír a tužka.

Důležitou podmínkou v této fázi vývoje je, aby i zadatatel měl jasno v tom, co od budoucí databáze může očekávat. Návrhář proto musí být schopen zadataveli objasnit, základní principy databází. Na druhou stranu je nutné mít o zadavatele (popř. budoucího uživatele) zadanou jasnou specifikaci uživatelských požadavků.

Objektové vztahy

V druhém kroku se (podle [Vokounová 2003] a [Longley 2001]) jasně určí objektové typy (třídy) a funkce. Popíší se vztahy mezi třídami. Výsledkem tohoto kroku je objektový model (diagram) vytvořený podle přesných pravidel pro zápis objektové analýzy. Takovým standardním jazykem pro její zápis je UML.

V této fázi se samozřejmě stále nelze obejít bez zadavatele. Právě na základě vytvořeného modelu lze konfrontovat uživatelské požadavky s funkcionalitou databáze. V tomto kroku je možné zadavateli předvést strukturu budoucí databáze, vysvětlit jednotlivé části modelu a popřípadě nastínit funkcionalitu databáze. Ze schůzky (schůzek) pak výsledně vyplynou další připomínky, které lze v této fázi bez problémů řešit doplněním nebo upravením modelu. Otázkou zůstává, jak daný model vytvořit, aby ho zadavatel pochopil. S tím souvisí otázka, jak zadavatel ovládá UML a jak je zběhlý v informačních technologiích.

Geografická reprezentace

Poté následuje (podle [Vokounová 2003] a [Longley 2001]) krok třetí, ve kterém se určuje geografická reprezentace objektů. Správný výběr je velmi důležitý, souvisí totiž přímo s rychlostí a efektivností výsledné databáze. Samozřejmě můžeme provádět přetypování, ale tato operace je náročná na výpočetní čas počítače, zároveň také dochází k nevratné ztrátě informací.

Návrh geografické reprezentace objektů již musí návrhář udělat sám. Měla by ovšem vyplynout z pohledu zadavatele a to zejména z funkcionality databáze. Reprezentací se rozumí rozdělení objektů to objektových tříd a u každé třídy určit zda se bude jednat o bodovou, liniovou nebo polygonovou třídu. Každá reprezentace je vhodná k jiným účelům a analýzám a zvolení špatné reprezentace má za následek špatnou funkcionalitu databáze. Proto hraje tato část návrhu velice důležitou roli.

4.3.2 Logický model

Logický model umožňuje (podle [Ressler 2006]) zobrazit a popsat objekty v databázi a vztahy mezi nimi s ohledem na jejich implementaci v konkrétním technicko-programovém prostředí daném strukturou (organizací) datové základny a typem systému řízení báze dat. Logický model je rozšířením konceptuálního modelu o podrobnosti specifické pro dané prostředí, např. datové typy, realizaci vazeb mezi daty, integritní omezení. Neobsahuje popis konkrétní fyzické organizace a uložení dat na záznamovém médiu.

Geografické databázové typy

Čtvrtý krok (podle [Vokounová 2003] a [Longley 2001]) spočívá v nalezení vhodných geografických databázových typů, které budou maximálně vyhovovat reprezentaci objektů a zároveň budou podporovány GIS a jejich databázemi. Datový model GIS je totiž závislý na způsobu uložení, tedy na databázovém software, se kterým daný GIS spolupracuje (např., Oracle, MS Access, PostGIS, PostgreSQL atd.).

Pro ukládání prostorové složky dat není tento krok až tak závislý na návrháři databáze, protože daný systém má většinou přednastavené databázové typy, které nelze měnit (např. formátem OGC – viz [OGC 2006]). S atributovou složkou dat už je to ovšem něco jiného. Např. ESRI při vytvoření určitého objektu automaticky dovytvoří v atributové tabulce atributy: OID (primární klíč), Shape (reprezentace), Shape_Length (délka hranice u polygonové a liniové vrstvy), Shape_Area (plocha polygonu u polygonové vrstvy). Ostatní, pro zadavatele důležité, atributy je dobré v modelu také zahnout, ale jejich přidání nebo modifikaci lze provést v již hotové databázi kdykoli.

Struktura GDB

Pátým krokem je (podle [Vokounová 2003] a [Longley 2001]) uspořádání struktury geografické databáze. To zahrnuje definování způsobu uložení topologických vztahů, pravidel, volbu vhodného prostorového indexu, atd..

Topologie by se, podle mého názoru, měla vyskytnout v každé geografické databázi. Většinou se totiž jedná o znázornění částí krajiny, ve které se vyskytují určité prostorové vztahy. Proti tomu stojí fakt, že vybudování topologie a následné odladění chyb je poměrně časově náročná záležitost. Proto je nutné na začátku zvážit, jestli se budování topologie opravdu vyplatí.

Popis dvou možných způsobů uložení topologie je popsán na str. 38. Při své praktické části diplomové práce jsem se setkal s vytvářením ESRI topologie. Z mých zkušeností vyplývá, že vytvoření topologie není až tak časově náročná záležitost, aby omezovala uživatele databáze. Něco jiného už je pak následná oprava chyb pomocí nástrojů *topology*, která je podstatně složitější (viz praktická část mé práce), ale ta není cílem tohoto kroku.

4.3.3 Fyzický model

Jedná se o vytvoření výsledné struktury v konkrétní geografické databázi. Zahrnuje nejen strukturu databáze, ale i definování různých integritních omezení, topologické vztahy, jednotlivé atributy a jejich typy, atd.

Schéma GDB

Šestým a posledním krokem je vytvoření konečného schématu geografické databáze. Tj. konečné struktury databáze tak, jak budou data uložena přímo v databázi. Je vytvořena pomocí databázového jazyka, který podporuje daný DBMS. V poslední době nejpopulárnějším jazykem je SQL/3, který má v sobě zabudovanou i práci s geografickými datovými typy.

Další možností vytvoření výsledného schématu v ESRI technologiích je použití nadstavby ArcCatalog „ESRI Diagrammer“, která vygeneruje z již hotové databáze jednotlivé prvkové třídy ve formě tabulek (viz Obr. 4.20). Bohužel nelze touto cestou modelovat speciální vztahy mezi prvky databáze (např. topologické vztahy), které je nutno následně dokreslit ručně.

Simple feature class
GeomorphologicalIndividual_polygon

Geometry *Polygon*
Contains M values *No*
Contains Z values *Yes*

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
Shape	Geometry	No					
GeomorphologicalIndividual	String	Yes					100
Shape_Length	Double	Yes			0	0	
Shape_Area	Double	Yes			0	0	

Obr. 4.20: Ukázka vyexportované prvkové třídy do UML pomocí programu „ESRI Diagrammer“

4.4 Možnosti vytvoření GDB

Možností pro vytvoření geografické databáze je celá řada. Na začátku je nutné prozkoumat, jestli se již danou problematikou někdo nezabýval. Např. na stránkách firmy ESRI [ESRI 2007a] lze nalézt několik případů předvytvořených modelů (tzv. Case Studies). Jedná se většinou o konceptuální modely, které byly vytvořeny odborníky z jednotlivých oborů. Dají se zde nalézt modely geologie, zemědělství, atmosféry, adres, povrchové vody, aj. Spolu s těmito modely se zde

vyskytují již některé hotové prázdné databáze ve formátu „Personal geodatabase“.

V poslední době patří mezi nejrozšířenější návrhy nejen geografických databází přístup pomocí jazyka UML. Tento jazyk využívá např. program Microsoft Visio nebo Rational Rose. MS Visio má stejné uživatelské rozhraní jako většina produktů této firmy. Výhodou je, že k němu lze doprogramovat různé pluginy a knihovny, které pomáhají při tvorbě GIS. Touto cestou se dala i firma ESRI, která si vytvořila vlastní dialekt UML „ArcInfo UML Model“ s možností exportu do formátu XML, který je dále přenositelný a zobrazitelný v jiných grafických nástrojích, které nepodporují UML.

V knize [Arctur 2004], která je zaměřena zejména na vytvoření Personal Geodatabase, lze vyčíst několik způsobů vytvoření databáze. Domnívám se, že tyto způsoby tvorby databáze lze rozšířit na tvorbu geografických databází obecně.

Způsoby vytvoření Personal Geodatabase

- **Vytvoření prázdné Personal Geodatabase**

  Jedním ze způsobů, který napadne každého na prvním místě, je vytvoření prázdné geografické databáze od základu. Jde o způsob, který je univerzální ovšem nejpracnější. Výhoda tohoto způsobu je, že není potřeba jiných speciálních programů nebo skriptů a takovou, lze říci kostru, správně schéma databáze, lze v budoucnu použít při tvorbě jiných modelů. U firmy ESRI stačí při tomto způsobu tvorby databáze pouze program ArcCatalog.

- **Úprava struktury již existující databáze**

  Další možností vytvoření databáze je modifikace již existující databáze. Tento způsob se většinou používá při tvorbě databáze, která má podobnou strukturu jako původní. Z toho vyplývá, že se jedná o tvorbu databáze ze stejné nebo velmi podobného oboru (např. vytvoření silniční ze železniční databáze).

- **Modifikace databázového schématu**

  V poslední době se rozšířil návrh databází pomocí jazyka UML. Programy, ve kterých lze lépe modelovat objekty a vztahy mezi nimi většinou přímo nepodporují jazyk UML. Používaným formátem pro převod jiných formátů mezi sebou je XML Stejné je to i s jazykem UML, který je často ukládán ve zmiňovaném formátu. Strukturu databázového schématu uloženou v XML lze pak importovat do programu, který UML podporuje (např. Visio). Ve formátu UML je možné pak strukturu modifikovat, doplnit a případně pomocí speciálních rozšíření vytvořit celou databázi.

4.5 Ukázka návrhu na příkladu geomorfologického informačního systému

Geomorfologický informační systém (dále GmIS) je (podle [Minár, et al. 2005] a [Mentlík, et al. 2006]) speciálním typem GIS zaměřený na sběr, správu a analýzy geomorfologických informací. Implementace GmIS byla vytvořena na příkladu Prášilského jezera na Šumavě. Popis vytvoření fyzického modelu geomorfologické databáze je nutnou podmínkou pro implementaci geomorfologických analýz v prostředí GmIS. Stejně jako u jiných návrhů databází, je nutné se zabývat před vytvořením fyzického modelu návrhem konceptuálního a logického modelu.

Z hlediska vytváření modelu je nutné rozlišovat GmIS ze dvou pohledů:

1. Z pohledu geomorfologa – je GmIS definován jako prostředí pro určité analýzy georeliéfu (většinou geomorfologické analýzy) a rovněž jako místo pro uložení geomorfologických dat a dat potřebných pro geomorfologický výzkum.

2. Z pohledu technika – je GmIS chápáno jako prostředí pro ukládání a správu geomorfologických dat, a jako možnosti pro částečně předdefinované analýzy nad daty.

V následující části je uveden popis jednotlivých částí konceptuálního a logického modelu z pohledu geomorfologa, jak je uvádí [Minár, et al. 2005].

Jedná se o:

- *adopted layers* – primární „negeomorfologické" vrstvy
- *basic layers* – základní geomorfologické vrstvy
- *special layers* – speciální geomorfologické vrstvy

Adopted layers

Adopted layers jsou pro geomorfologa nejdůležitějším zdrojem informací, které byly původně vytvořeny k jinému účelu. Jsou použity pro vytvoření základních, ale i speciálních geomorfologických vrstev. Patří sem:

1. Topografické mapy (Topographic maps) – většinou v měřítku 1:10 000 nebo 1:25 000 s vrstevnicemi zakreslenými po 1 až 10 m.
2. Ortofotomapy (Orthophotomaps) – nabízejí velmi přesnou informaci o poloze geomorfologických objektů a jejich hranicích.
3. Říční síť (River network) – jsou velice důležité vrstvy, které jsou nutné k vytvoření povodí.
4. Geologická data (Geological data) – jsou důležitá pro geomorfologický výzkum. V ČR je celé území pokryto geologickou mapou 1:50 000.
5. Půdní mapy (Soil maps) – jsou důležitým zdrojem informací, zvláště pro výzkum čtvrtohorní morfodynamiky.
6. Ostatní – patří sem např. historické mapy, mapy využití země, mapy vegetace, triangulace a vyrovnání sítí, aj.

Basic layers

Základní geomorfologické vrstvy jsou plné nebo částečné nezávislé struktury zakladních geomorfologických informací. Patří mezi nejčastěji využívané informace a měly by být základem každého GmIS (viz Obr. 4.21). Patří sem:

1. Digitální model terénu (Digital elevation model - DEM) – je jedním z nejčastěji využívaných vstupů pro prostorové analýzy. Slouží jako vstup pro vytvoření odvozených morfometrických ukazatelů (jako jsou sklon, orientace, křivost), pro hledání společných vlastností terénu (např. úvodí nebo elementárních forem) a modelování (např. hydrologických funkcí, toku energií a požárů lesů).
2. **Elementární formy (Elementary forms)** – reprezentují základní (geometricky, geneticky a dynamicky stejné) části povrchu. Jsou definovány konstatní hodnotou nějaké významné morfometrické veličiny (výšky a jejích odvozenin) a liniemi nespojitostí těchto veličin. Konstantní hodnota morfometrických veličin je spojena s jakýmkoli druhem genetické, chronologické nebo dynamické homogenity a nespojitosti pak reprezentují přírodní hranice terénu, kde jsou tyto homogenity porušeny.

   Definice elementárních forem podle [Mentlík 2007]: Georeliéf se skládá (v daném rozlišovacím měřítku) z **elementárních forem georeliéfu**. Jedná se o geometricky homogenní plochy, ohraničené vůči jiným elementárním formám výraznými liniemi

(hranami), narušujícími jejich homogenitu. Na každé konkrétní elementární formě předpokládáme shodný průběh recentních geomorfologických procesů a jednotnou genezi všech jejích částí. Proto je nazýváme **geneticky stejnorodé plochy.** Tyto plochy mají určité **morfometrické vlastnosti (sklon, orientaci a křivost).**

3. Povodí (Basins) – jsou přírodní hydro-geomorfologické jednotky, které vyjadřují prostorovou organizaci nejdůležitějších exogenních geomorfologických procesů. Vymezení povodí reprezentuje přírodní části povrchu země odlišně než elementární formy.
4. Dokumentace (Documentation materials) – reprezentují směs informací o určitých specifikách dané oblasti, např. odkryvu země, půdě, povrchu, vegetaci nebo hydrologických vlastnostech.
5. Morfodynamické jevy (Morphodynamic phenomena) – současné geomorfologické procesy nejsou pouze o tvarování reliéfu, ale některé mohou představovat rizika pro lidskou společnost, proto by měly být zahrnuty v GmIS.
6. Geomorfologická síť (Geomorphic network) – je vytvořena pro vizualizaci lineárních útvarů reliéfu (např. různých nespojitostí, jako jsou údolnice, hřbetnice, příkopy, aj.), které vytváří pravidelnou síť.
7. Genetické skupiny povrchu (Genetic groups of landforms) – vytváří specifické průřezy geomorfologickými vrstvami sjednocující: vybrané elementární formy, povodí, dokumenty, ale také ostatní elementy na bázi původu.

Name	Generation	Function
DEM and its derivatives	Generated from adopted layers (topographic maps, orthophotomaps, geodetic nets and river networks) by mathematical modelling.	Generation of other features (elementary forms, basins, geomorphic network) and computation of their morphometric characteristics.
Elementary forms	Defined morphometrically on the basis of DEM and verified and modified by geomorphological (GPS) mapping.	Basic part of a GmIS. Other features just provide extra information for them. Extrapolation of genetic, chronological and dynamic properties and geomorphological regionalization.
Basins	Derived from DEM (with use of river network layer) and verified by geomorphological mapping.	Basic actual morphosystems. Modelling of actual processes. Boundaries provide information about each form, which they intersect.
Documentation materials	Received by geomorphological (GPS) mapping or from the adopted layers.	Connection of morphogenetically relevant information with other features stored.
Morphodynamic phenomena	Received by stationary research and field mapping as well as from adopted layers.	Connection of morphodynamically relevant information with other features stored.
Geomorphic network	Derived from DEM and orientation of geomorphological features.	Used mainly for morphostructural analysis and general genetic interpretations.
Genetic groups of landforms	They connect parts of georelief according to their genesis. Derived from elementary forms and field survey using adopted layers and morphodynamic phenomena.	Thanks to the genetic groups of geomorphological forms it is possible to create special maps of genetic forms of the area of interest and other specific layers.

Obr. 4.21: Pozice a funkce basic layers v GmIS – převzato z [Minár, et al. 2005]

Special layers

Při procesu vytváření basic a adopted layers může být vytvořeno mnoho speciálních vrstev. Patří mezi ně vrstvy, vytvořené speciálními geomorfologickými analýzami, jako např. morfostrukturní analýzy (morphostructural analysis), komplexní geomorfologické analýzy (comprehensive geomorphological analysis), geomorfologické vyhodnocení rizika (geomorphological hazard evaluation).

Popis návrhu geomorfologické databáze

Pro ukládání a správu GmIS byla vybrána databáze hlavně kvůli její schopnosti bezpečně ukládat a manipulovat s velkým množstvím dat. Přesněji byla vybrána *Personal Geodatabase* z důvodu jejího širokého uplatnění a možnosti vytvoření topologie mezi vrstvami. Mezi další přednosti patří fakt, že umožňuje import a export dat do formátu shapefile, který se „de facto“ stal standardem pro výměnu geodat mezi geografickými databázemi.

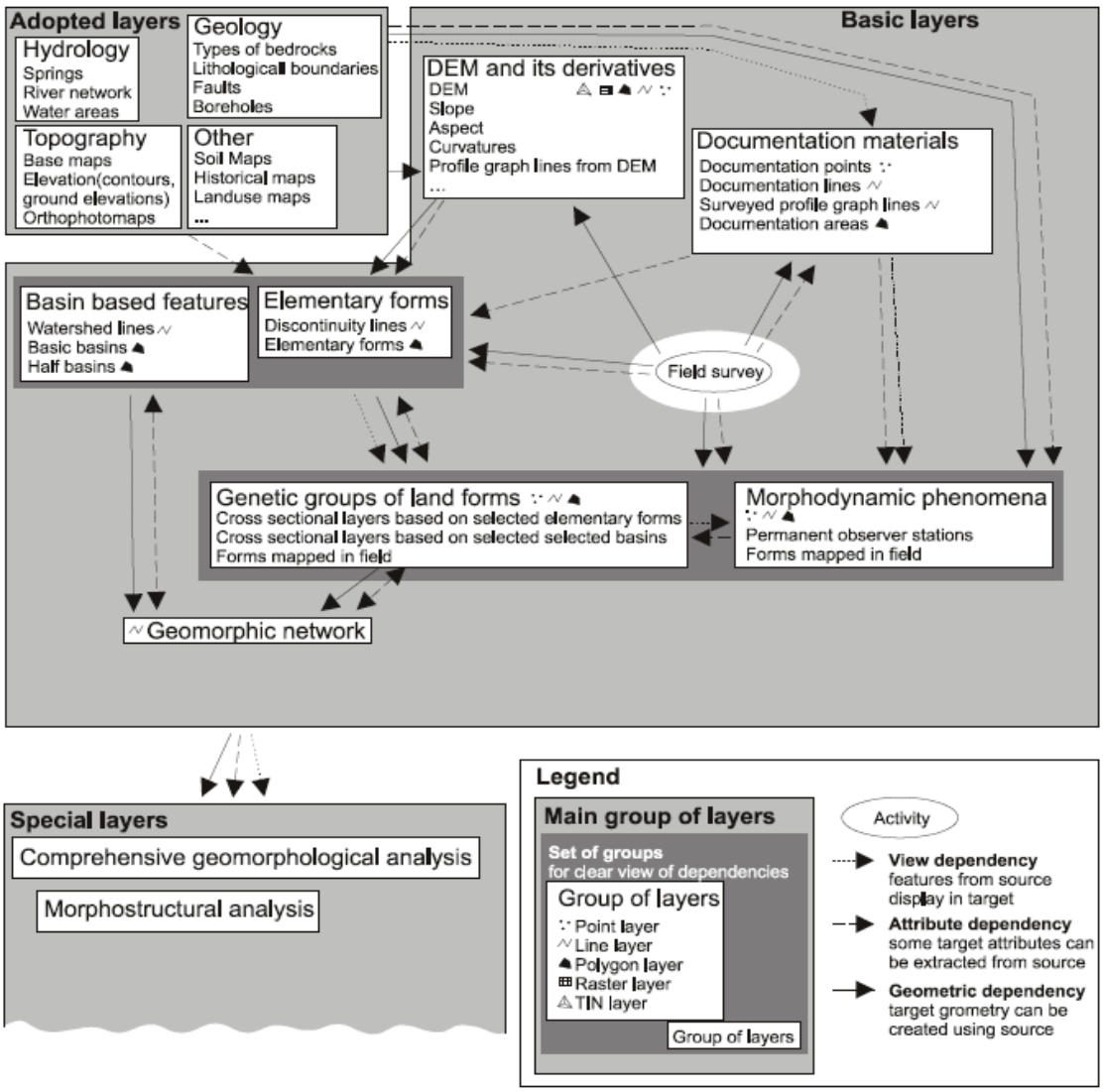
Konceptuální model geomorfologické databáze

Jak již bylo uvedeno, konceptuální návrh se zabývá zejména získáváním širšího rozhledu návrháře v dané oblasti zejména od potenciálních uživatelů budoucího IS. Měl by znázorňovat skupiny prvků, kterým bude v budoucnu (při vytváření logického a fyzického modelu) možné porozumět jako zárodkům vrstev, skupin vrstev a jejich vzájemných vztahů.

Konkrétně geomorfolog jako zadavatel vytvořil nástin tak, že rozdělil ukládání dat do tří předchozích částí podle jeho uvážení. Stanovil také určitý vztah jednotlivých vrstev mezi sebou (určitá data vytvářejí jiná) tak, že odborník přes GIS by měl být schopen vytvořit základní konceptuální model databáze. K tomuto návrhu postačí pouze papír a tužka.

Logický model geomorfologické databáze

Návrhář databáze je pak schopen na základě konceptuálního návrhu vytvořit logický model. Důležitou součástí je určení reprezentace jednotlivých vrstev, tzn. výběr bodových, liniových, polygonových a jiných reprezentací prostorových dat a příslušných atributových datových typů (např. number, string, date, aj.) do atributových tabulek jednotlivých vrstev. Spolu s tím je nutné blíže specifikovat existující vztahy mezi elementy jako jsou relace, topologie a funkční závislosti. Příklad logického modelu s vyznačenými funkčními závislostmi a základním rozdělení do vrstev a skupin vrstev je vidět na Obr. 4.22.



Obr. 4.22: Logický model geomorfologické databáze – převzato z [Minár, et al. 2005]

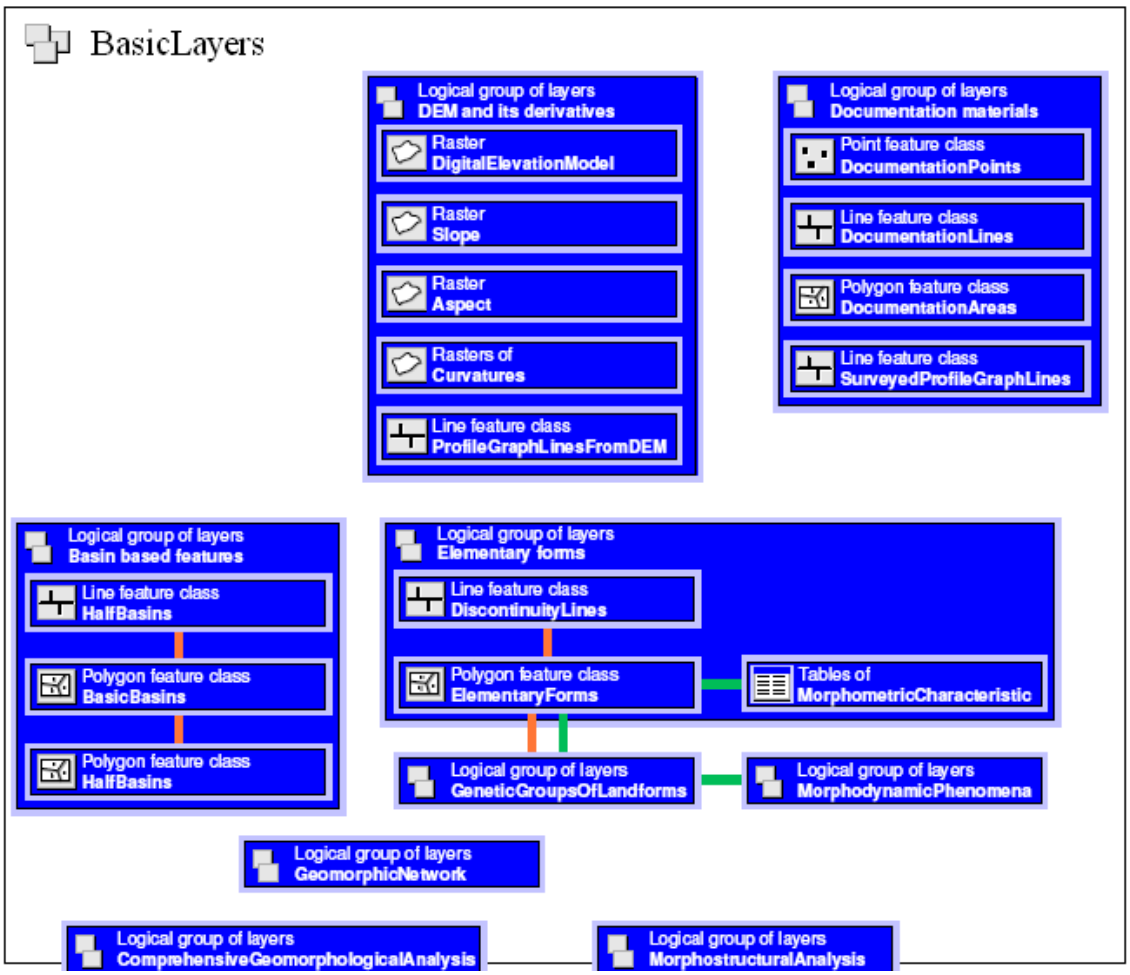
Fyzický model gemorfologické databáze

Logický model musí mít dobře definované jádro, které je základem vytvoření fyzického modelu. V tomto případě je dáno pevným rozdělením logických skupin vrstev. Logická struktura GmIS by měla zůstat otevřená, protože bude využívána na různých zájmových oblastech a odlišných geomorfologických školách. Musí umožňovat zaplnění databáze jinými vrstvami než je tomu v případě Prášilského jezera, nebo naopak musí umožňovat vynechání některých vrstev. Jednotlivá rozhodnutí pak musejí být zvlášť řešena nad určitým územím. Nicméně, rozdělení do předešlých skupin by mělo zůstat neměnné.

Fyzický model geomorfologické databáze již souvisí s konkrétním uložením databáze v DBMS. Jinými slovy lze říci, že dokud není vytvořen fyzický model, můžou být a většinou jsou očekávány změny ve struktuře jak konceptuálního, tak logického modelu, které by ovšem neměly být zásadního charakteru.

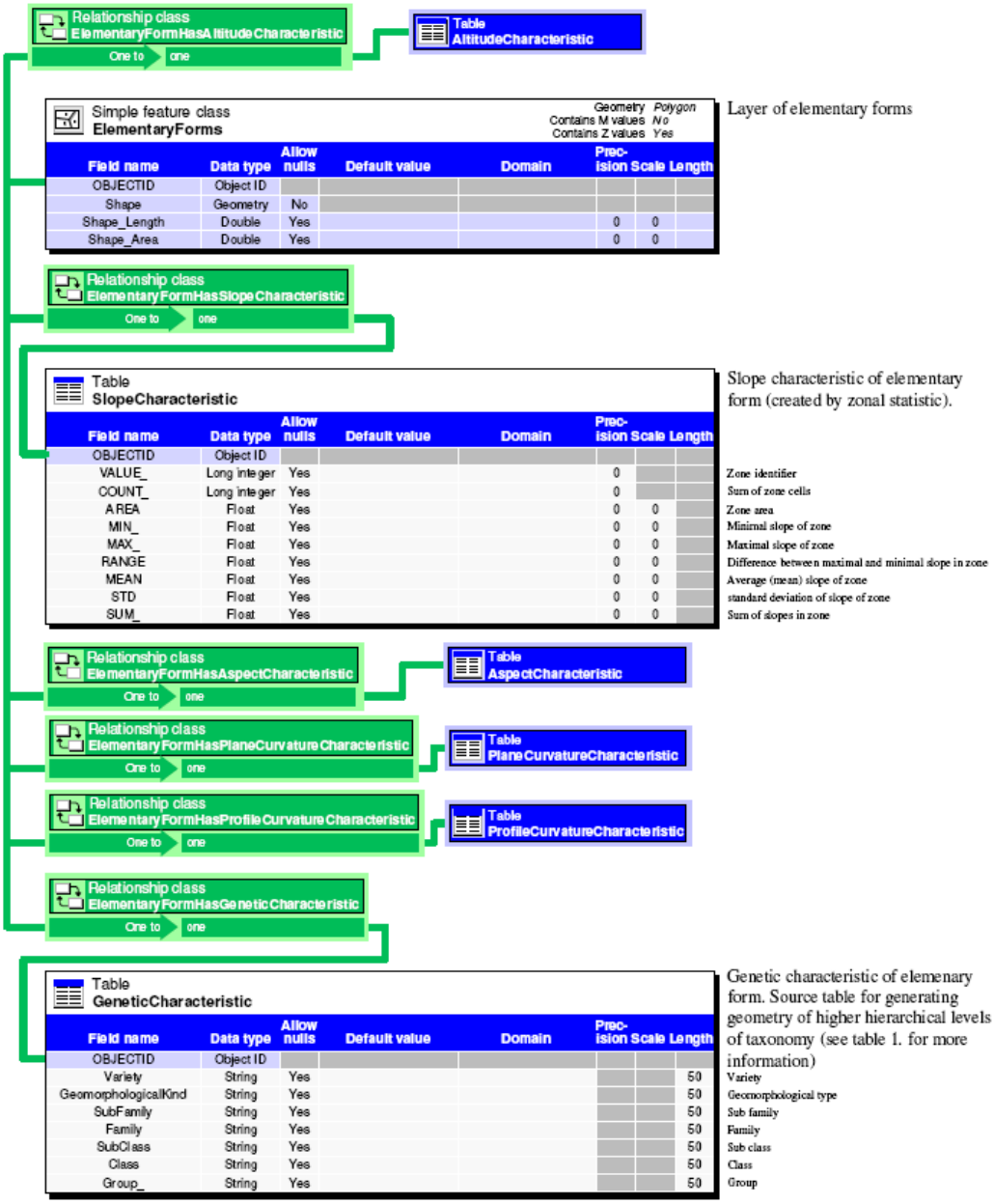
Jak již bylo uvedeno, základem celé geomorfologické databáze je skupina vrstev „basic layers“, v [Mentlík, et al. 2006]. Protože se již bavíme o fyzickém uložení dat v Personal Geodatabase, vytváří „basic layers“ dataset. Na Obr. 4.23 je znázorněn zjednodušený fyzický model datové sady

„BasicLayers“ bez atributů.



Obr. 4.23: Fyzický model datové sady „BasicLayers“, vytvořený pomocí ESRI Geodatabase Diagrammer v Microsoft Visio - převzato z [Mentlík, et al. 2006]

Následující obrázek (viz Obr. 4.24) znázorňuje detail elementárních forem a relacemi připojených vrstev fyzického modelu. Tabulky, které popisují charakter elementárních forem (výšku, sklon, orientaci a křivost) se spojují s elementárními formami pomocí relací. Na obrázku je v detailu pro přehlednost znázorněna pouze tabulka *Sklon (SlopeCharacteristic)*.



Obr. 4.24: Detail fyzického modelu elementárních forem, vytvořený pomocí ESRI Geodatabase Diagrammer v Microsoft Visio – převzato z [Mentlík, et al. 2006]

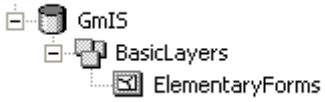
5 Praktická část diplomové práce

5.1 Vytyčení cílů práce

Cílem praktické části diplomové práce byla úprava struktury goemorfologické databáze, která spočívala v usnadnění vytváření prvkových tříd a topologie vyšších hierarchických forem reliéfu částečnou automatizací procesů. Cíle práce se dají rozdělit do několika bodů:

- Vyplnění atributové tabulky prvkové třídy ElementaryForms – jedná se o to, že jednotlivé části povrchu (elementární formy) lze slučovat do větších celků, které jsou si podobné z hlediska např. způsobu vzniku, složení hornin a půd, viz dále. Aby nemusel geomorfolog vytvářet jednotlivé prvkové třídy vyšších forem ručně, lze tyto formy vytvořit automaticky. Nutností je ovšem nejdříve vyplnění atributové tabulky ElementaryForms.
- Tvorba jednotlivých prvkových tříd (polygonových a liniových) vyšších forem – z atributové tabulky se dají pomocí nástrojů ArcGIS rozpustit hranice sousedících polygonů a vytvořit tak prvkové třídy vyšších forem. Po aplikaci jiného nástroje lze vytvořit i prvkové třídy hranic těchto forem.
- Vytvoření topologie mezi formami – jednotlivé prvkové třídy bylo nutné mezi sebou provázat pomocí topologie, aby se předcházelo nekonzistinci dat v databázi..
- Ladění topologických chyb – při výskytu topologických chyb bylo nutné určitý typ automatizovaně odstranit.

Výchozím bodem pro mne byla geomorfologická databáze okolí Prášilského jezera vytvořená pomocí „Personal Geodatabase“, na které jsem testoval své výsledky. Konkrétně se jednalo o prvkovou třídu ElementaryForms (definici elementárních forem lze nalézt na str. 49), ze které jsem vycházel. Pro lepší orientaci jsem si vytvořil novou databázi a do ní zkopíroval pouze potřebná data z původní databáze (viz Obr. 5.1).



Obr. 5.1: Výchozí geomorfologická databáze s elementárními formami okolí Prášilského jezera

Výsledkem mé práce je geomorfologická databáze, která má podobnou strukturu jako databáze okolí Prášilského jezera a je v ní uložen tzv. Toolbox s jednotlivými použitými modely (viz dále). Součástí výsledku je pak ještě projekt „GmIS.mxd“, ve kterém jsou uloženy vytvořené skripty (v programu Visual Basic for Application).

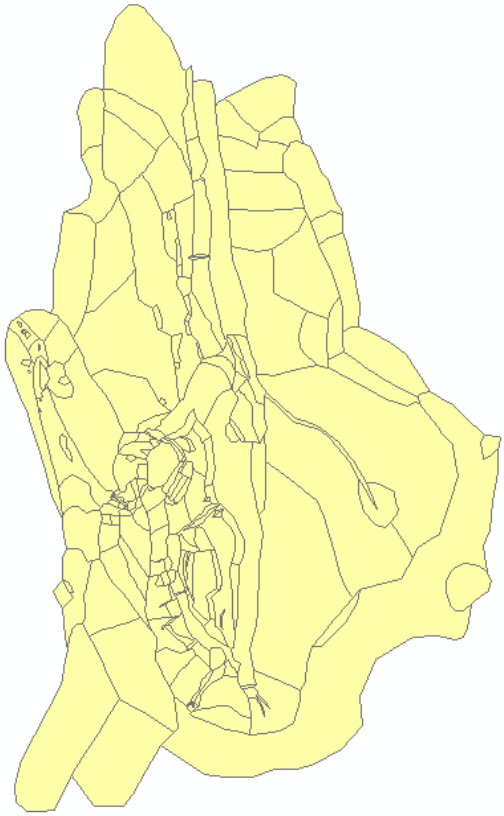
5.2 Vyplnění atributové tabulky prvkové třídy ElementaryForms

Jak již bylo uvedeno, je základní vrstvou pro vytvoření vyšších hierarchických forem vrstva elementárních forem ElementaryForms (viz Obr. 5.2). Tuto vrstvu musí většinou vytvořit geomorfolog. Takto rozdělený povrch lze následně seskupovat do větších celků na základě např. způsobu vzniku, složení hornin a půd atd.

Mezi vyšší hierarchické formy patří:

- group (skupina),

- class (třída),
- subclass (podtřída),
- family (rodina),
- subfamily (podrodina),
- geomorphological individual (geomorfologické individuum na úrovni druhu),
- variety (varieta),
- subgroup (podskupina),
- subgroups (podskupiny),
- morphochronological position (morfochronologická pozice).



Obr. 5.2: Elementární formy u Prášilského jezera

Na Obr. 5.3 je ukázka atributové tabulky elementárních forem Prášilského jezera. Jednotlivé atributy tabulky popisují, ke které vyšší hierarchické formě každá elementární forma patří.

K usnadnění zaplnění tabulky slouží vytvořené skripty „Select By Unique“ a „UpdateEF“. Skripty byly vytvořeny pomocí programu Visual Basic for Application (dále VBA) a uloženy v projektu GmIS.mxd. Základem skriptů je předpoklad, že elementární formy, které mají hodnotu atributu (nejčastěji GeomorphologicalIndividual) stejnou, se řadí do stejných vyšších hierarchických forem.

Contents | Preview | Metadata

GeomorphologicalIndividual	Variety	morphochronological_position	Subfamily
etchplain	nivation hollow	fossil form	destruction
pediment		ancient form	destruction
pediment	dellen	phase 3	destruction
pediment		phase 3	destruction
pediment		phase 3	destruction
erosion-denudation slope		phase 2	destruction
erosion-denudation slope		phase 2	destruction
pediment		phase 3	destruction
pediment		phase 3	destruction
etchplain		fossil form	destruction
erosion-denudation slope	quarry	phase 1	destruction
erosion-denudation slope		phase 1	destruction
outwash plain	gully	present-day	destruction
outwash plain		fossil form	construction
pediment		phase 3	destruction
pediment		phase 3	destruction
pediment		ancient form	destruction
pediment		phase 3	destruction
pediment		phase 3	destruction
erosion-denudation slope		phase 2	destruction
bottom of corrie	nivation hollow	fossil form	destruction
wall of corrie		Q3	destruction
wall of corrie	permanent slope erosio	present-day	destruction
wall of corrie	permanent slope erosio	present-day	destruction
wall of corrie	nivation hollow	fossil form	destruction
etchplain - dissected	residual knob	fossil form	destruction
dellen	permanent slope erosio	present-day	destruction
dellen	peatbog	present-day	construction
wall of corrie		Q3	destruction
wall of corrie		Q3	destruction
etchplain - dissected	residual knob	fossil form	destruction
etchplain - protected	cryoplanation surface	fossil form	destruction
etchplain - dissected	cryoplanation surface	fossil form	destruction
wall of corrie		Q3	destruction

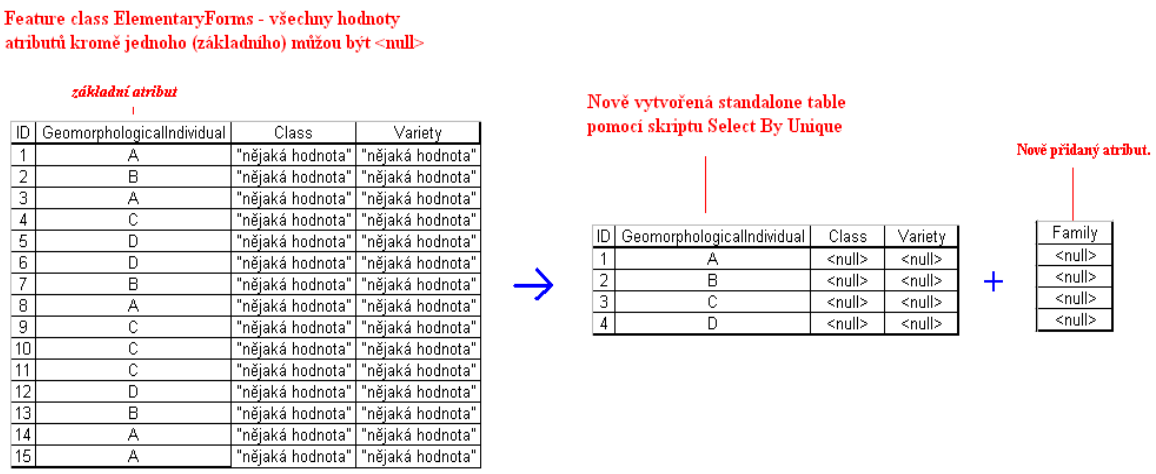
Record: 1 Show: All Selected Records (of 220) Options

Obr. 5.3: Atributová tabulka ElementaryForms – ukázka z programu ArcCatalog

5.2.1 Skript „Select By Unique“

Tento skript vytváří novou tabulku „standalone table“, která je uložena ve stejné databázi jako feature class ElementaryForms a má stejnou strukturu jako ElementaryForms. Význam skriptu spočívá v redukci počtu řádků podle zvoleného atributu tak, aby jednotlivé hodnoty ve zvoleném atributu byly unikátní. Jinými slovy lze říct, že neexistují v tabulce dvě stejné hodnoty vybraného atributu. Hodnoty ostatních atributů jsou nastaveny jako <*null*> (viz Obr. 5.4).

Navíc skript umožňuje přidání nového atributu do standalone table, který se aplikací skriptu „UpdateEF“ posléze objeví i v původní feature class ElementaryForms.



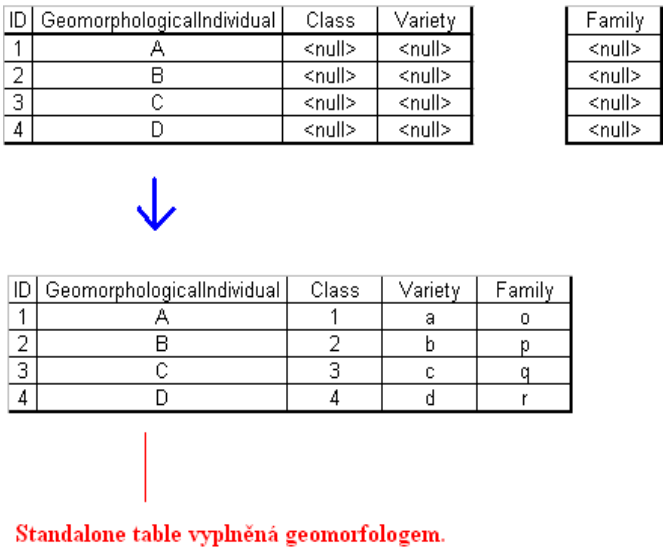
ID	GeomorphologicalIndividual	Class	Variety
1	A	"nějaká hodnota"	"nějaká hodnota"
2	B	"nějaká hodnota"	"nějaká hodnota"
3	A	"nějaká hodnota"	"nějaká hodnota"
4	C	"nějaká hodnota"	"nějaká hodnota"
5	D	"nějaká hodnota"	"nějaká hodnota"
6	D	"nějaká hodnota"	"nějaká hodnota"
7	B	"nějaká hodnota"	"nějaká hodnota"
8	A	"nějaká hodnota"	"nějaká hodnota"
9	C	"nějaká hodnota"	"nějaká hodnota"
10	C	"nějaká hodnota"	"nějaká hodnota"
11	C	"nějaká hodnota"	"nějaká hodnota"
12	D	"nějaká hodnota"	"nějaká hodnota"
13	B	"nějaká hodnota"	"nějaká hodnota"
14	A	"nějaká hodnota"	"nějaká hodnota"
15	A	"nějaká hodnota"	"nějaká hodnota"

ID	GeomorphologicalIndividual	Class	Variety
1	A	<null>	<null>
2	B	<null>	<null>
3	C	<null>	<null>
4	D	<null>	<null>

Family
<null>
<null>
<null>
<null>

Obr. 5.4: Význam skriptu Select By Unique na příkladu geomorfologické databáze

Dalším krokem při tvorbě vyšších hierarchických forem je vyplnit nově vzniklou standalone table. Tento krok musí opět provést geomorfolog.



ID	GeomorphologicalIndividual	Class	Variety
1	A	<null>	<null>
2	B	<null>	<null>
3	C	<null>	<null>
4	D	<null>	<null>

Family
<null>
<null>
<null>
<null>

ID	GeomorphologicalIndividual	Class	Variety	Family
1	A	1	a	o
2	B	2	b	p
3	C	3	c	q
4	D	4	d	r

Obr. 5.5: Vyplněná standalone table

Jádro skriptu „Select By Unique“:

```
"skript projíždí jednotlivé řádky atributové tabulky ElementaryForms

For rows1 = 1 To pElementaryFormsTable.RowCount(Nothing)
    ' projíždí atributovou tabulku standalone table, jestli již neexistuje stejná hodnota základního (base) atributu jako je
    'v atributové tabulce ElementaryForms
    If Not SearchFields(p1Row, intField1, intField2) Then
        'vytvoří nový řádek a zapíše hodnotu základního atributu
        Set p2Row = pSATable.CreateRow
        p2Row.Value(intField2) = p1Row.Value(intField1)
        p2Row.Store
```

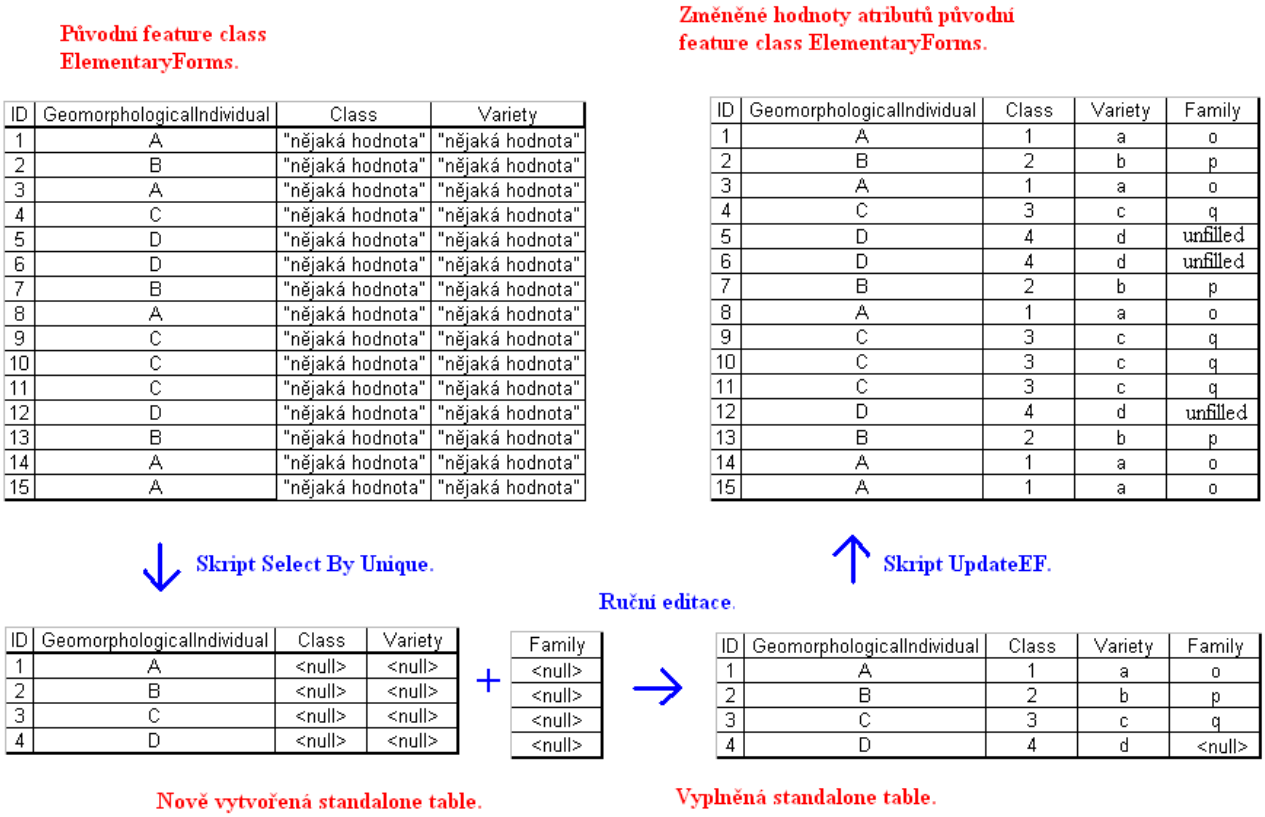
```
    End If
    'další řádek
    Set p1Row = pCursor.NextRow
Next rows1
```

```
' projíždí atributovou tabulku standalone table, jestli již neexistuje stejná hodnota základního (base) 'atributu jako je v atributové tabulce ElementaryForms
Private Function SearchFields(pEFTRow As IRow, i As Integer, j As Integer) As Boolean
 'přístup k řádkům přes ITableSort -> ICursor -> IRow
 Dim p2TabSort As ITableSort
 Set p2TabSort = New TableSort

 With p2TabSort
    .Fields = "OBJECTID"
    .Ascending("OBJECTID") = True
    Set .Table = pSATable
    .Sort Nothing

 End With
 Dim p2Cursor As ICursor
 Set p2Cursor = p2TabSort.rows

 'nastavení kurzoru na první řádek
 Dim pSATRow As IRow
 Set pSATRow = p2Cursor.NextRow
 'hledá stejnou hodnotu v základním atributu
 Do While Not pSATRow Is Nothing
  If pEFTRow.Value(i) = pSATRow.Value(j) Then
   'pokud nalezne, vrací hodnotu true
    SearchFields = True
    Exit Function
  End If
 'další řádka
  Set pSATRow = p2Cursor.NextRow
 Loop
 'pokud nenalezne, vrací hodnotu false
 SearchFields = False
End Function
```

5.2.2 Skript UpdateEF

Skript UpdateEF změní hodnoty atributů původní feature class ElementaryForms na nové hodnoty ve standalone table. Pokud byl ve skriptu „Select By Unique“ přidán nový atribut, přidá se tento atribut i do původní feature class ElementaryForms (viz Obr. 5.6).

Na žádost geomorfologů byla vytvořena nadstavba, která zajišťuje, že pokud bude mít hodnota jakéhokoli atributu typu *string* ve standalone table hodnotu <null>, bude v atributové tabulce ElementaryForms změněna na <unfilled>. Podobně, pokud bude mít atribut *„Variety“* ve standalone table hodnotu <null>, přepíše se v atributové tabulce ElementaryForms na hodnotu *„typical <GeomorphologicalIndividual>“*.



ID	GeomorphologicalIndividual	Class	Variety
1	A	"nějaká hodnota"	"nějaká hodnota"
2	B	"nějaká hodnota"	"nějaká hodnota"
3	A	"nějaká hodnota"	"nějaká hodnota"
4	C	"nějaká hodnota"	"nějaká hodnota"
5	D	"nějaká hodnota"	"nějaká hodnota"
6	D	"nějaká hodnota"	"nějaká hodnota"
7	B	"nějaká hodnota"	"nějaká hodnota"
8	A	"nějaká hodnota"	"nějaká hodnota"
9	C	"nějaká hodnota"	"nějaká hodnota"
10	C	"nějaká hodnota"	"nějaká hodnota"
11	C	"nějaká hodnota"	"nějaká hodnota"
12	D	"nějaká hodnota"	"nějaká hodnota"
13	B	"nějaká hodnota"	"nějaká hodnota"
14	A	"nějaká hodnota"	"nějaká hodnota"
15	A	"nějaká hodnota"	"nějaká hodnota"

ID	GeomorphologicalIndividual	Class	Variety	Family
1	A	1	a	o
2	B	2	b	p
3	A	1	a	o
4	C	3	c	q
5	D	4	d	unfilled
6	D	4	d	unfilled
7	B	2	b	p
8	A	1	a	o
9	C	3	c	q
10	C	3	c	q
11	C	3	c	q
12	D	4	d	unfilled
13	B	2	b	p
14	A	1	a	o
15	A	1	a	o

ID	GeomorphologicalIndividual	Class	Variety
1	A	<null>	<null>
2	B	<null>	<null>
3	C	<null>	<null>
4	D	<null>	<null>

Family
<null>
<null>
<null>
<null>

ID	GeomorphologicalIndividual	Class	Variety	Family
1	A	1	a	o
2	B	2	b	p
3	C	3	c	q
4	D	4	d	<null>

Obr. 5.6: Celkový průběh vyplnění hodnot atributů feature class ElementaryForms na příkladu geomorfologické databáze

Stačí, aby prvková třída ElementaryForms měla vyplněn pouze jeden z atributů vyšších hierarchických forem (většinou GeomorphologicaIndividual). Pomocí skriptu „Select By Unique“ se pak dají dovytvořit a zaplnit zbývající atributy. Na příkladu Prášilského jezera ovšem tyto atributové sloupce v prvkové třídě ElementaryForms existovaly, takže nebylo potřeba vytvářet žádné jiné.

Další možností, jak vyplnit zbývající hodnoty atributů je, že standalone table bude již v databázi vytvořena. Jinými slovy lze říci, že standalone table bude jakýmsi vzorem pro vyplnění prvkové třídy ElementaryForms. Pak stačí pouze poopravit některé hodnoty atributů ve standalone table, které se pro dané území odlišují, a aplikací skriptu „UpdateEF“ pouze změnit hodnoty atributů prvkové třídy ElementaryForms.

Jádro programu „Update EF“:

```
'projíždí všechny řádky atributové tabulky ElementaryForms

For rows1 = 1 To pElementaryFormsTable.RowCount(Nothing)
   'projíždí všechny řádky standalone table
   Do While Not p2Row Is Nothing
      'pokud se rovnají hodnoty hodnoty v základní atributu
      If p1Row.Value(intField1) = p2Row.Value(intField2) Then
         'projíždí obě tabulky po řádcích
         For i = 0 To pElementaryFormsTable.Fields.FieldCount - 1
            For j = 0 To pStandaloneTable.Fields.FieldCount - 1
             'nemění hodnoty v atributech jako jsou např. OID, Shape*
             If TestFields(i, pElementaryFormsTable) Or TestFields(j, pStandaloneTable) Then
```

```
            'pokud se rovnají názvy atributů
            ElseIf pElementaryFormsTable.Fields.Field(i).name = pStandaloneTable.Fields.Field(j).name Then
              'pokud je v standalone table atribut Variety nastaven jako <null>
               ValueIsNull = IsNull(p2Row.Value(j))
               'vyplní hodnotu v atributové tabulce ElementaryForms jako: typical <GeomorphologicalIndividual>
           If (p2Row.Value(j) = "") And (pStandaloneTable.Fields.Field(j).name = "Variety") Or _
                (ValueIsNull) And (pStandaloneTable.Fields.Field(j).name = "Variety") Then
                 p1Row.Value(i) = "typical " & p2Row.Value(intField2)
                 p1Row.Store
       'pokud je hodnota ve standalone table <null>, uloží do atributové tabulky ElementaryForms hodnotu unfilled
                 ElseIf ((p2Row.Value(j) = "") Or ValueIsNull) And pElementaryFormsTable.Fields.Field(i).Type =_
                 esriFieldTypeString Then
                    p1Row.Value(i) = "unfilled"
                    p1Row.Store
              'jinak změní hodnotu atributu v atributové tabulce ElementaryForms hodnotou ve standalone table
                 Else
                 p1Row.Value(i) = p2Row.Value(j)
                 p1Row.Store
               End If
             End If
           Next j
         Next i
       End If
      'další řádka ve standalone table
       Set p2Row = p2Cursor.NextRow
   Loop
   'nastavení ukazatele ve standalone table na první řádku
   Set p2Cursor = p2TabSort.rows
   Set p2Row = p2Cursor.NextRow
   'další řádka v atributové tabulce ElementaryForms
   Set p1Row = p1Cursor.NextRow
Next rows1
```

5.3 Tvorba jednotlivých prvkových tříd vyšších forem

Cílem této části diplomové práce bylo vytvoření samostatných prvkových tříd vyšších hierarchických forem uložených v datasetu „BasicLayers“. Jednalo se o vytvoření polygonových vrstev vyšších forem spolu s hranicemi (liniové vrstvy) těchto forem. Pro řešení jsem si vybral rozšíření ArcGIS „Model Builder“.

Jedná se o rozšíření, které je instalováno spolu s produkty firmy ESRI. Vybral jsem si ho zejména kvůli jednoduchosti ovládání a přehlednosti grafického znázornění jednotlivých komponent. Jeho další předností je vytváření komplexních modelů, které po spuštění fungují jako celek. Základem programu je grafické prostředí, které vytváří modely (podobné jako vývojové diagramy) – více viz např. ESRI 2005. Seskupením jednotlivých modelů lze vytvořit tzv. sadu nástrojů (Toolbox), kterou je možno uložit přímo do Personal Geodatabase.

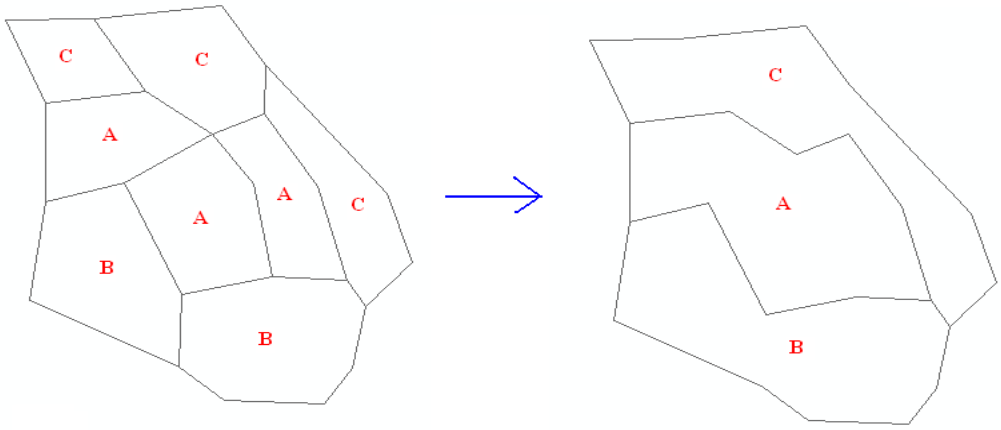
Klady programu „Model Builder“:

- řetězení jednotlivých nástrojů podle potřeby,
- intuitivní práce s programem,
- lehká modifikace modelů,
- vytváření komplexních modelů,
- vytváření submodelů,

- uložení do Personal Geodatabase,
- snadné vytvoření dokumentace.

5.3.1 Vytvoření polygonových vrstev

Základem vytvoření vyšších hierarchických forem je již zmíněná feature class ElementaryForms. Po vyplnění všech hodnot atributů bylo nutné pro každý atribut pomocí nástroje „Dissolve“ rozpustit hranice sousedících polygonů (viz Obr. 5.7). Tímto způsobem vznikly polygonové vrstvy vyšších forem.



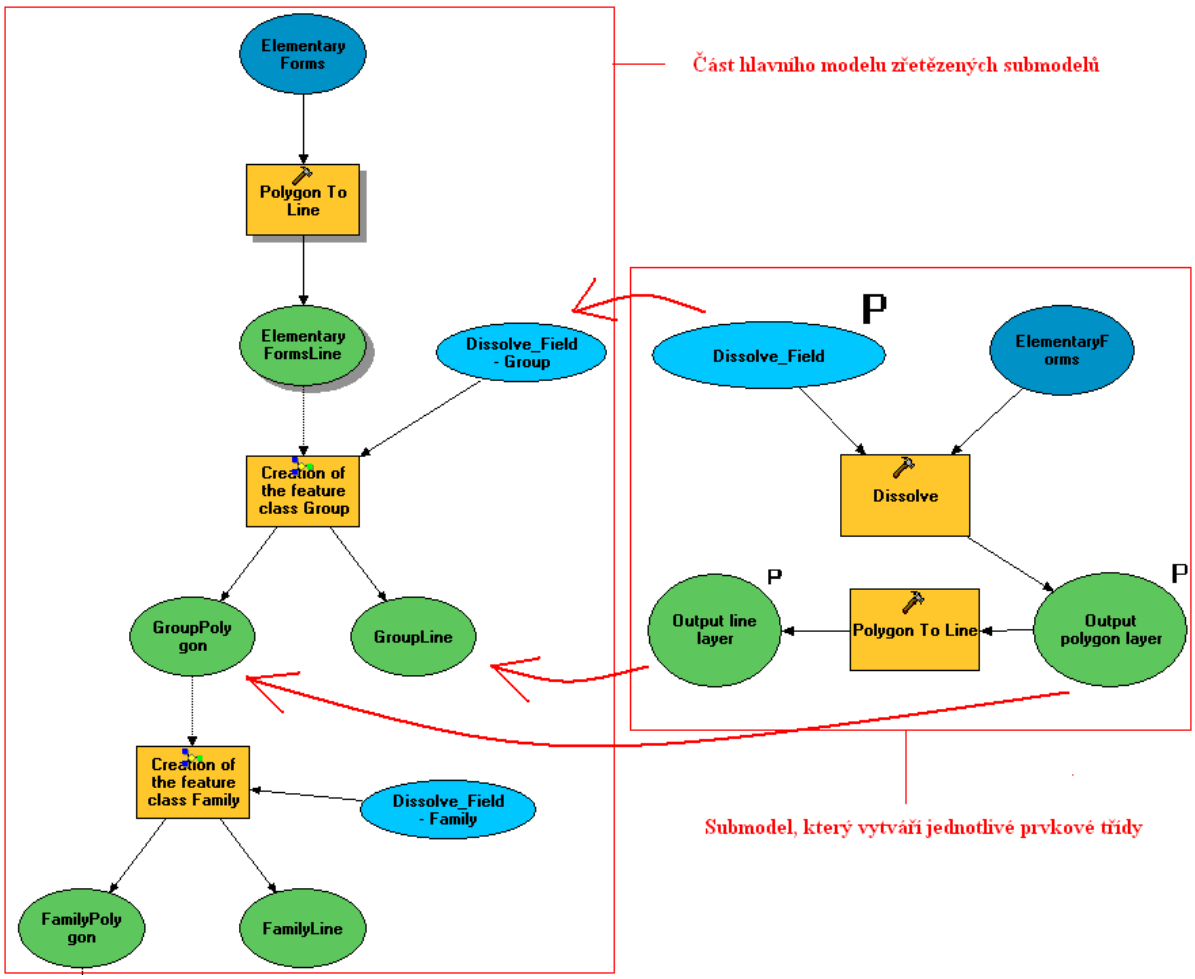
Obr. 5.7: Princip nástroje Dissolve

5.3.2 Vytvoření liniových vrstev

Po vytvoření polygonových vrstev se pomocí nástroje „Polygon To Line“ vytvoří hranice jednotlivých forem.

5.3.3 Použití rozšíření „Model Builder“

Jak již bylo uvedeno, lze pomocí programu řetězit jednotlivé modely. Celou automatizaci vytvoření prvkových tříd vyšších forem jsem rozdělil na jeden hlavní model a jeden submodel (model, který vstupuje do jiného modelu) – viz Obr. 5.8. Hlavní model slouží pouze ke zřetězení jednotlivých submodelů, kde do každého submodelu vstupují vždy jiné vstupní parametry (obdoba vstupních parametrů u funkcí). Tyto parametry jsou v modelu označeny písmenem **P** a lze je obecně měnit před spuštěním procesu. Můj případ byl ovšem trochu jiný v tom, že parametry byly součástí submodelu. V tomto případě je lze označit jako parametry, které propojují jednotlivé modely a jsou pevně dané (nelze je před spuštěním procesu měnit). Je to dáno tím, že byly stanoveny jisté konvence v pojmenování výstupních vrstev (*<NázevPříslušnéhoAtributu>Polygon* a *<NázevPříslušnéhoAtributu>Line*), tudíž jsou všechny názvy předem zvoleny.



Obr. 5.8: Vytvoření vyšších forem v rozšíření Model Builder

Pro uložení modelů je možné využít faktu, že Personal Geodatabase ukládá i Toolboxy, které si lze představit jako adresáře jednotlivých nástrojů a modelů. Výsledná struktura pak vypadá jako na Obr. 5.9.

Obr. 5.9: Výsledná struktura datasetu BasicLayers s vyššími formami - vytvořená v programu Visio

5.4 Topologické vztahy mezi vyššími formami

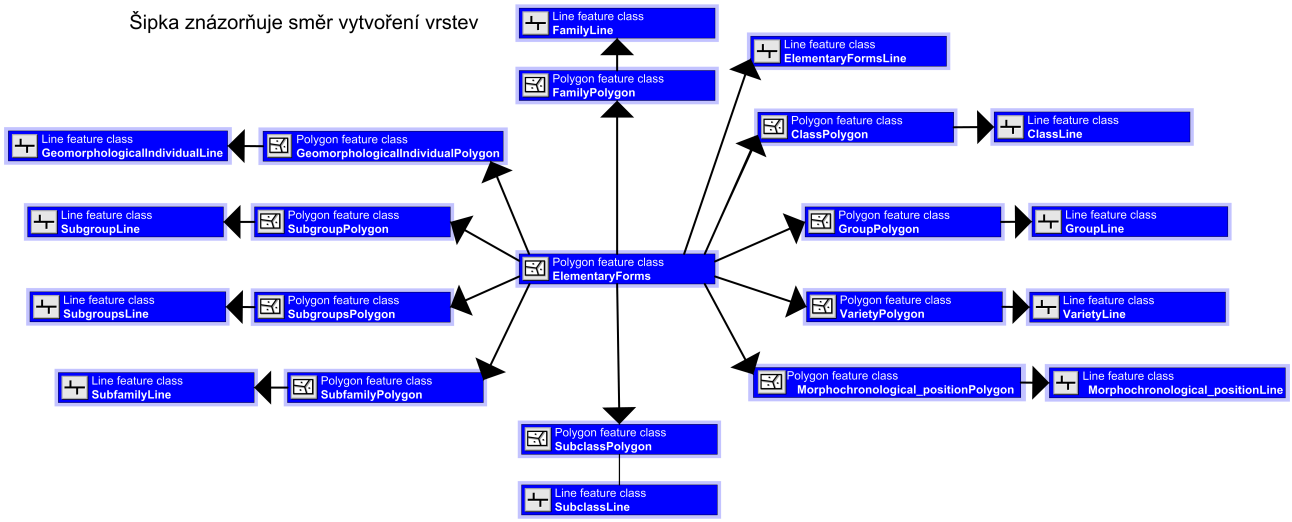
Personal Geodatabase používá pro vytváření topologie topologických pravidel. Jak bylo uvedeno na str. 38, lze vytvářet pouze topologická pravidla uvnitř jednoho datasetu (zde „BasicLayers“) mezi jednotlivými vrstvami nebo v rámci jedné vrstvy.

Kroky při vytváření topologie v Personal Geodatabase:

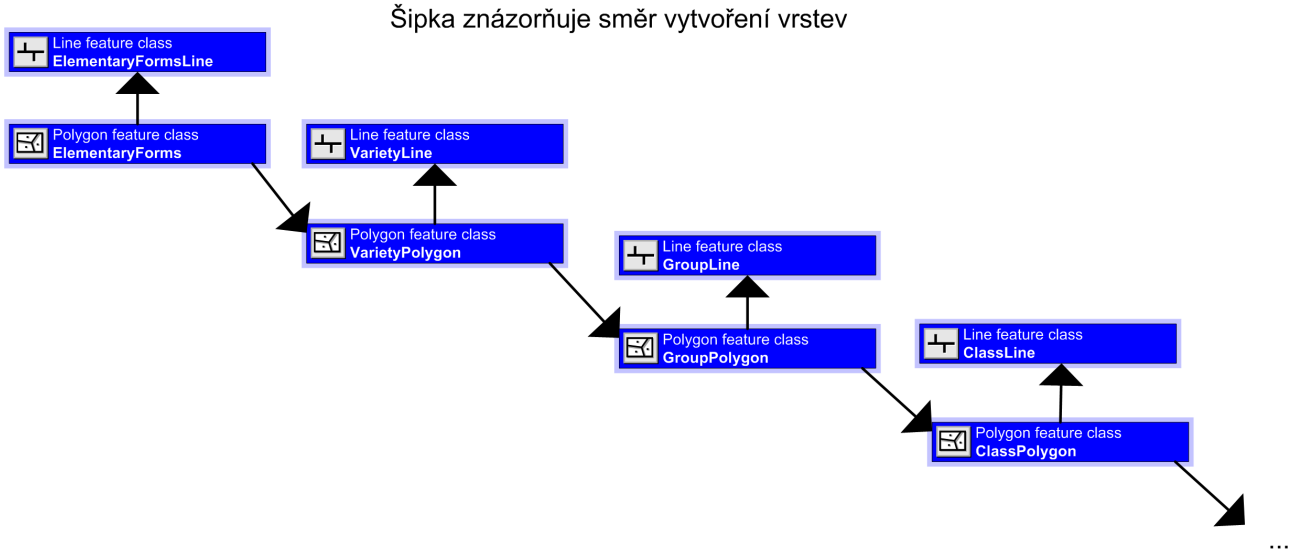
1. Vytvoření topologie (funkce „*Create Topology*“) - musí být obsažena ve stejném datasetu jako vrstvy, kterých se bude týkat.
2. Vložení feature class do topologie (funkce *„Add Feature Class To Topology“*) - vložení vrstev, pro která budou pravidla tvořena.
3. Vytváření jednotlivých pravidel (funkce *„Add Rule To Topology“*) - pomocí rolovacích menu je možno vybrat vstupní vrstvu, pravidlo, případně i závislou vrstvu.
4. Validace topologie (funkce *„Validate Topology“*) - topologie musí být zkontrolována, aby odhalila chyby.

Vzhledem k tomu, že jednotlivé prvkové třídy se vytvářejí z atributů atributové tabulky ElementaryForms a není zaručeno, že budou vždy vyplněny všechny hodnoty atributů, bylo nutné použít tzv. hvězdicový přístup (star approach), nikoli kaskádový (ladder). Při nevyplněné hodnotě atributu by u hierarchického přístupu skončil proces chybou. Rozdíl je v závislosti jednotlivých úrovní vrstev v hierarchii. U hvězdicového přístupu (viz Obr. 5.10) se všechny odvozené vrstvy vážou k jedné centrální (ElementaryForms). U kaskádového přístupu (viz Obr. 5.11) jsou jednotlivé

vyšší formy odvozovány postupně, takže tvoří skutečnou hierarchii. Od tohoto okamžiku již nelze hovořit o hierarchických vyšších vrstvách, ale pouze vyšších vrstvách.



Obr. 5.10: Princip hvězdicového přístupu – vytvořeno v programu Visio



Obr. 5.11: Princip Kaskádového přístupu – vytvořeno v programu Visio

Celou automatizaci vytvoření topologie jsem rozdělil na jeden hlavní model a dva submodely. Hlavní model opět slouží pouze ke zřetězení jednotlivých submodelů. Stejně jako v předchozím případě vstupují do submodelu vždy jiné vstupní parametry a také je nelze měnit.

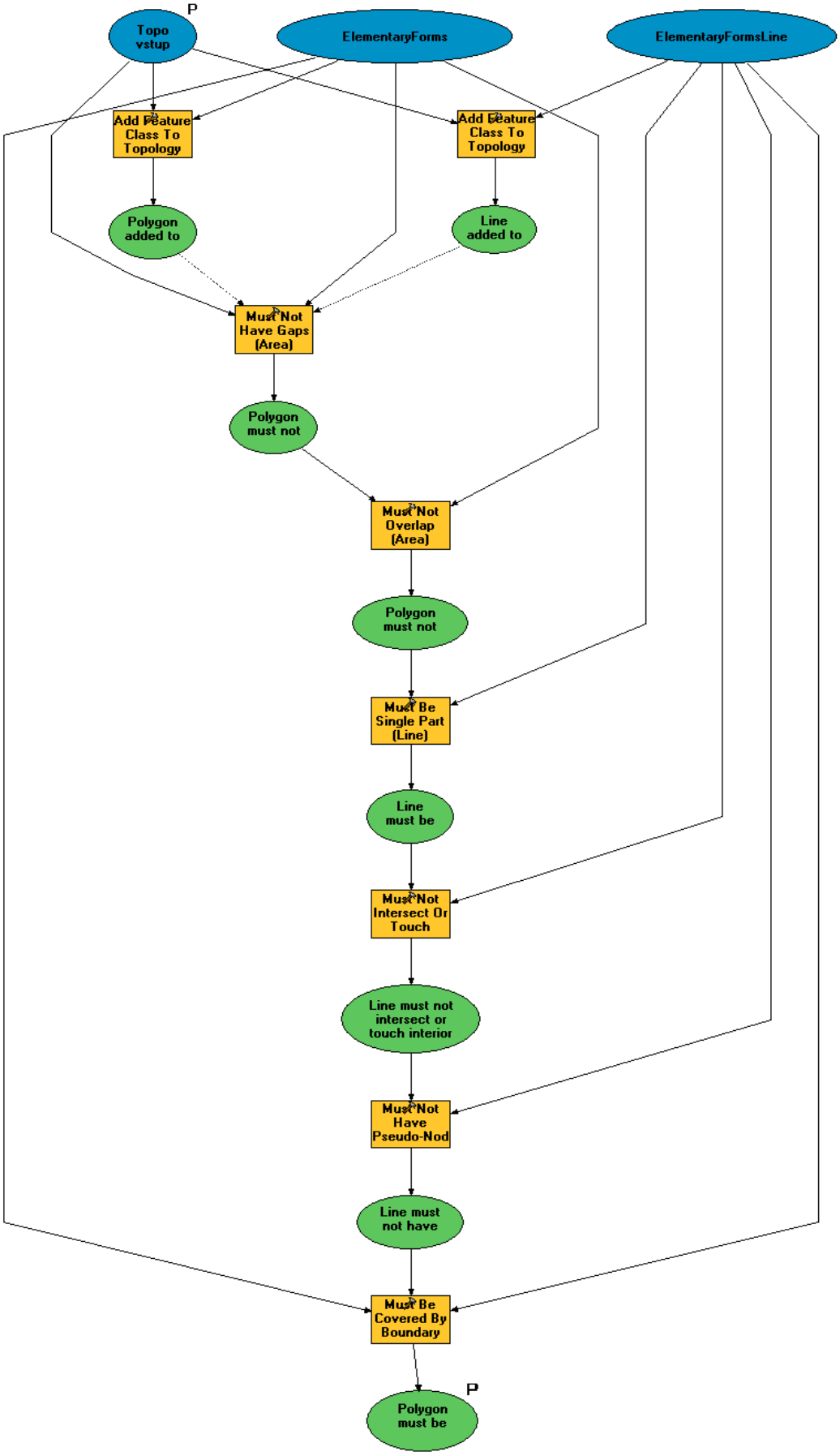
5.4.1 Submodel vytvoření topologie ElementaryForms

Protože topologie základní polygonové a liniové vrstvy ElementaryForms je trochu odlišná od ostatních vyšších forem, bylo pro její automatizované vytvoření nutné napsat speciální submodel. Následující Tab. 5.1 znázorňuje pravidla, která byla použita při vytváření topologie ElementaryForms. Na Obr. 5.12 je pak znázorněna část vytvořeného submodelu v rozšíření „Model

Builder“.

1. vrstva	Pravidlo (anglicky)	Pravidlo (česky)	2. vrstva
polygon	Must Not Have Gaps	Nesmí obsahovat mezery	
polygon	Must Not Ovelap	Nesmí přesahovat	
linie	Must Not Intersect Or Touch Interior	Nesmí se překrývat, protínat ani dotýkat	
linie	Must Be a Single Part	Musí mít jedinou část	
linie	Must Be Covered By Boundary	Musí ležet na hranicích polygonů	polygon

Tab. 5.1: Pravidla použitá u vrstev ElementaryForms



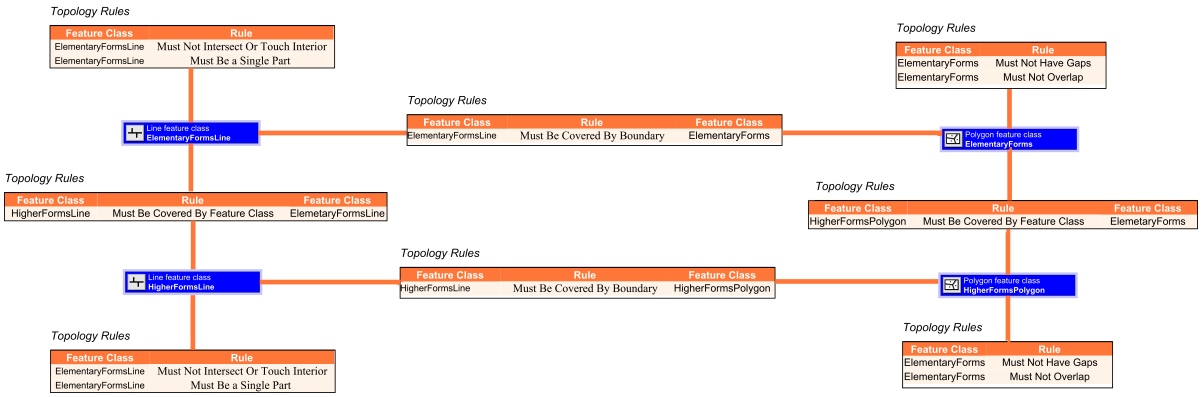
Obr. 5.12: Ukázka vytvořeného submodelu v rozšíření Model Builder

5.4.2 Submodel vytvoření topologie vyšších forem

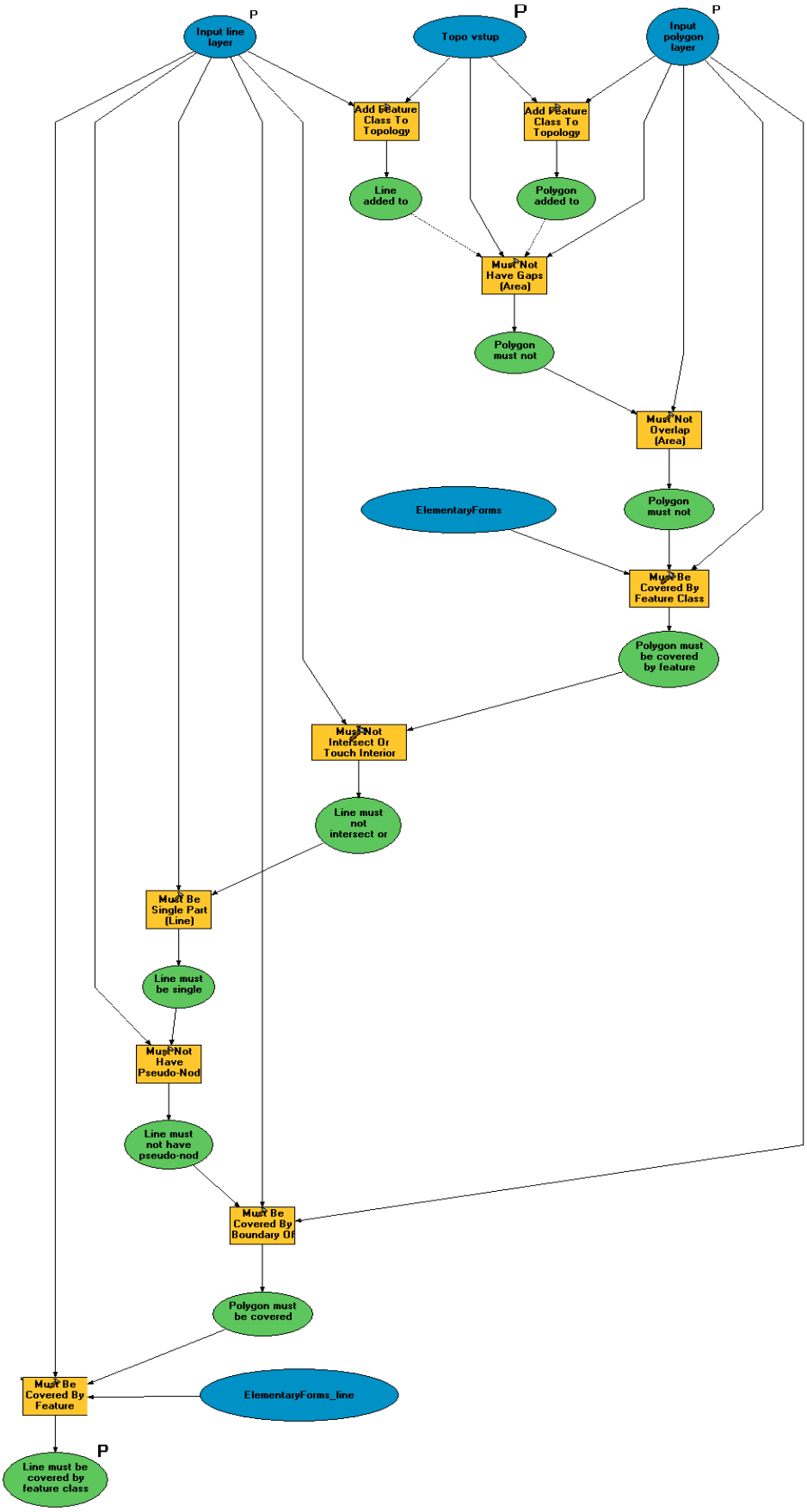
Pro automatizované vytvoření vyšších forem pomocí submodelu bylo nutné přidání dvou pravidel, která svazují tyto formy se základními vrstvami ElementaryForms a ElementaryFormsLine. V Tab. 5.2 jsou znázorněna jednotlivá pravidla pro vyšší hierarchické formy a na Obr. 5.13 je znázorněn logický model znázorňující topologická pravidla mezi jednotlivými vrstvami. Obr. 5.14 pak znázorňuje vytvořený submodel pro vytvoření topologie vyšších hierarchických forem.

1. vrstva	**Pravidlo (anglicky)**	**Pravidlo (česky)**	**2. vrstva**
polygon	Must Not Have Gaps	Nesmí obsahovat mezery	
polygon	Must Not Ovelap	Nesmí přesahovat	
polygon	Must Be Covered By Feature Class	Musí být pokryty třídou prvků	ElementaryForms_polygon
linie	Must Not Intersect Or Touch Interior	Nesmí se překrývat, protínat ani dotýkat	
linie	Must Be a Single Part	Musí mít jedinou část	
linie	Must Be Covered By Boundary	Musí ležet na hranicích polygonů	polygon
linie	Must Be Covered By Feature Class	Musí být pokryty třídou prvků	ElementaryForms_line

Tab. 5.2: Pravidla použitá u vyšších hierarchických vrstev



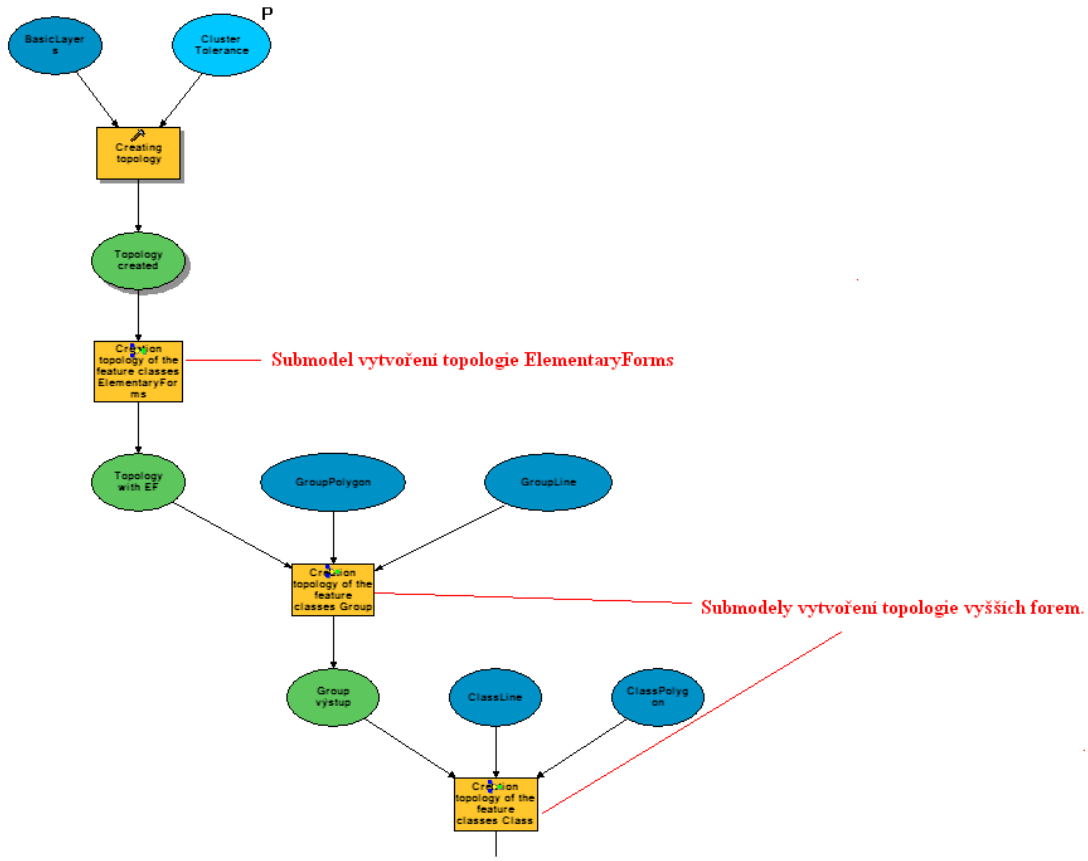
Obr. 5.13: Logický model znázorňující jednotlivá topologická pravidla mezi vrstvami – vytvořeno v programu Visio



Obr. 5.14: Submodel vytvoření topologie vyšších hierarchických forem v rozšíření Model Builder

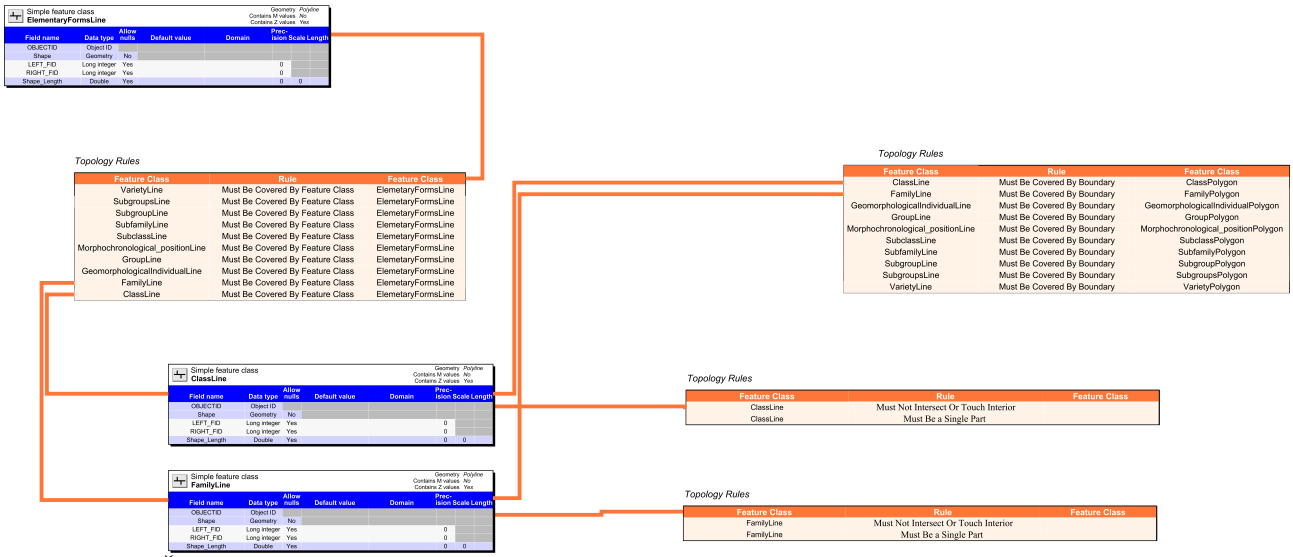
5.4.3 Výsledný model

Výsledný model vznikl opět zřetězením jednotlivých submodelů (viz Obr. 5.15).



Obr. 5.15: Část výsledného modelu vytvářejícího topologii

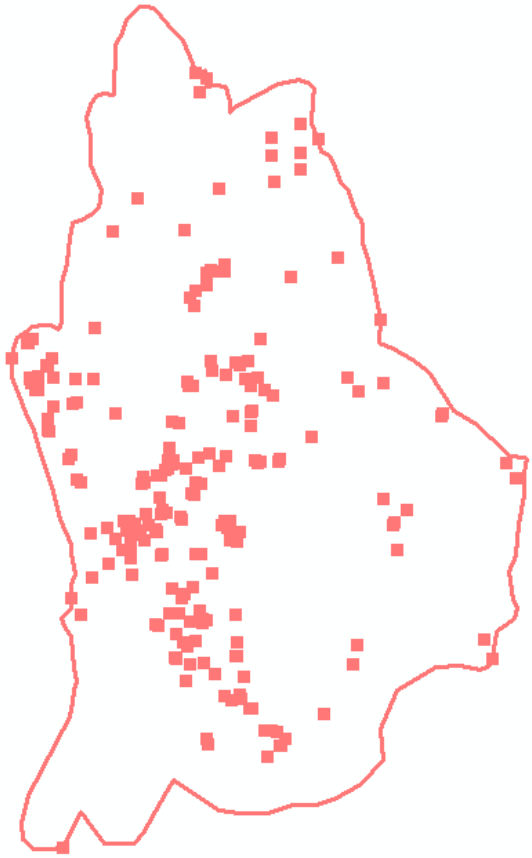
Na Obr. 5.16 je znázorněna část fyzického modelu topologie pomocí programu Visio. Protože znázornění celého modelu by bylo nečitelné, bude uložen v příloze a na přiloženém CD pod názvem *topologie.pdf*.



Obr. 5.16: Část fyzického modelu vytvořené topologie mezi vrstvami, celý model je uložen v příloze a na přiloženém CD - vytvořeno pomocí programu Visio

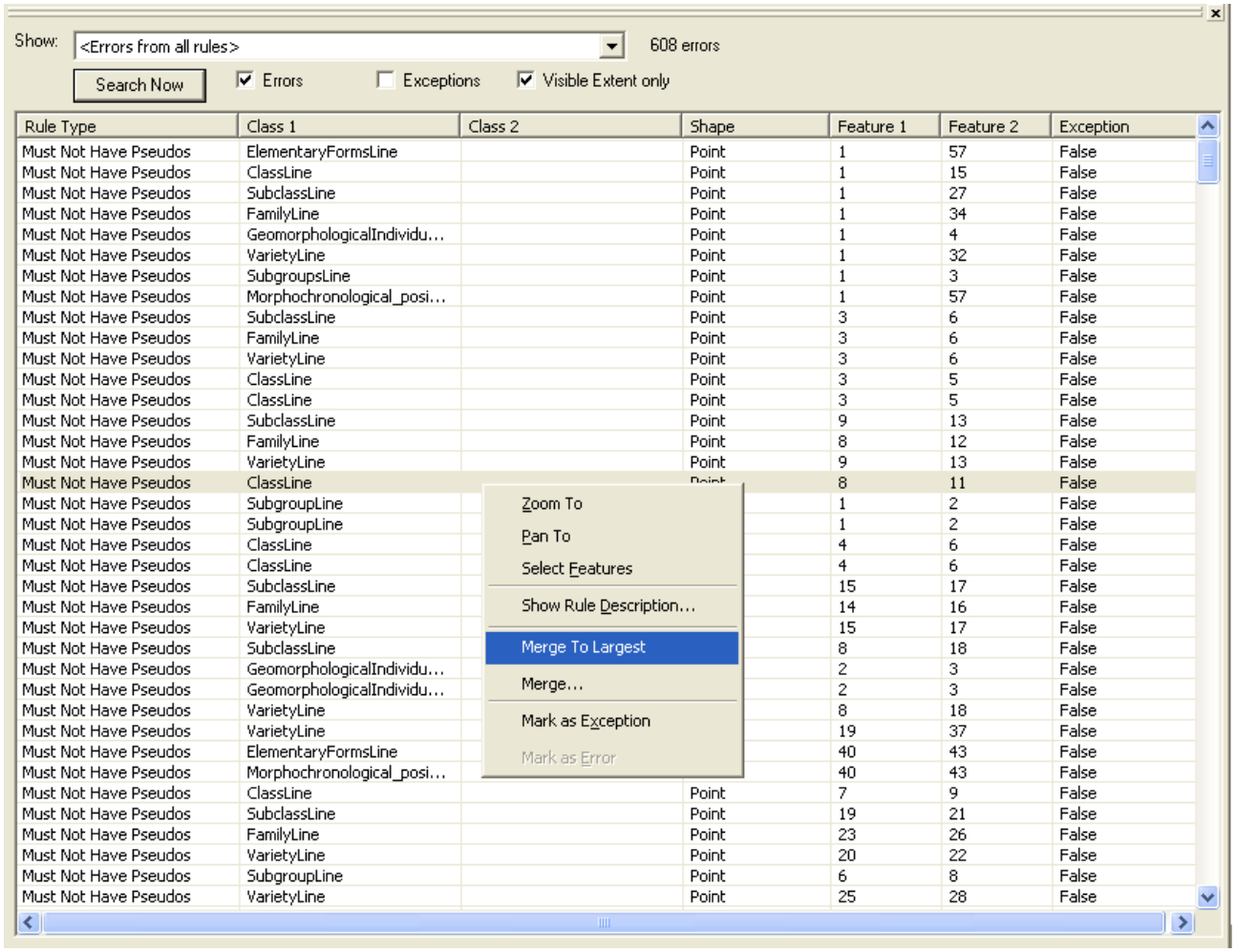
Po aplikaci výsledného modelu na příkladu Prášilského jezera se objevilo 608 topologických chyb (viz Obr. 5.17). Z toho:

- 10 chyb (Must Not Have Gaps) – hranice okolo celého území. Tento problém je standardní záležitostí, tudíž lze tyto chyby označit jako výjimky (Exceptions).
- 598 chyb (Must Not Have Pseudonodes) – všechny body na mapě.



Obr. 5.17: Příklady topologických chyb

Chyby „Must Not Have Pseudonodes“ se pomocí nástroje „Error Inspector“ (viz Obr. 5.18), který je umístěn v sadě nástrojů „Topology“ upravují poloautomaticky. Nástroj „Error Inspector“ nalezne všechny pseudo-uzly. Uživatel pak musí označit ty, které chce smazat a aplikací funkce „Merge to Largest“ je smaže. Princip skriptu je podobný se skriptem, který jsem vytvořil, ovšem můj skript je plně automatizovaný a na ovládání jednodušší. Navíc je možno následující skript publikovat na http://support.esri.com, kde jeho funkci někteří uživatelé možná ocení.



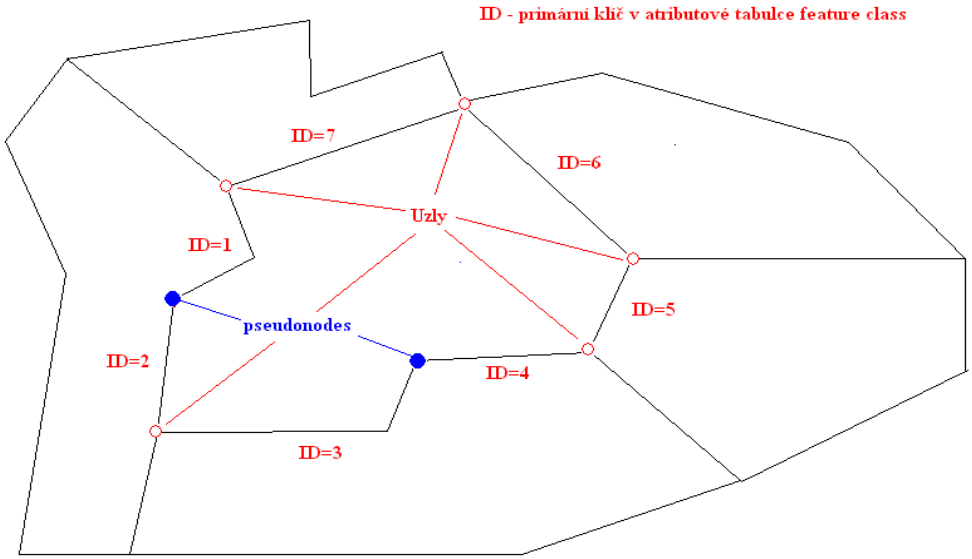
Obr. 5.18: Vymazání pseudo-uzlů pomocí nástroje Error Inspector

5.4.4 Skript „Delete Pseudonodes“

Skript „Delete Pseudonodes“ má za úkol upravit geometrii liniových vrstev tak, aby se v ní nevyskytovaly žádné pseudo-uzly.

Co jsou pseudo-uzly?

Pseudo-uzly vznikají tam, kde se linie dělí na více částí (viz Obr. 5.19). Jinými slovy lze říci, že segment linie od uzlu (node) k uzlu, který by měl být jedním prvkem se skládá ze dvou a více prvků. Důvod této chyby mi není znám, ale domnívám se, že problém zapříčinil nástroj „Polygon To Line“ (viz *Vytvoření liniových vrstev* na str. 62), který podle mého názoru rozděluje linie náhodně bez ohledu na to, jestli končí nebo začínají v uzlu.



Obr. 5.19: Příklad pseudonodes

Princip skriptu

Výhodou každého pseudo-uzlu je, že má v sobě uložen jednoznačný odkaz na prvkovou třídu (feature class ID) a ID jednotlivých prvků, kterých se týká (na Obr. 5.19 např. ID=1 a ID=2). Pomocí těchto ID lze jednoznačně identifikovat linii v atributové tabulce. Princip skriptu spočívá v tom, že geometrii druhého prvku sjednotí (Union) s geometrií prvního prvku. Protože obě linie mají stejné hodnoty v atributové tabulce, není nutné provádět žádné jiné změny. Pomocí sjednocení se navíc sečtou délky obou linií a automaticky se přepíší v prvním prvku.

Jádro programu „Delete pseudonodes“:

```
'nalezne všechny pseudo-uzly
Set pEnumTopologyErrorFeature = pErrorContainer.ErrorFeaturesByRuleType(pGeoDS.SpatialReference,
esriTRTLineNoPseudos, Nothing, True, False)
'nastavení ukazatele na první chybu
Set pTopoErrFeature = pEnumTopologyErrorFeature.Next

'pomocné pole pro ukládání feature class ID, uložené feature a smazané feature
Dim MyArray()
ReDim MyArray(ErrorCount, 3)

'prochází jednotlivé pseudo-uzly
For i = 0 To ErrorCount – 1
   'při chybě pokračuj
   On Error Resume Next
   'obdržení feature class ID, feature1 a feature2, mezi kterými je pseudo-uzel
   Set pFeatureClass1 = GetFeatureClass1(pTopoErrFeature.OriginClassID)
   Set pFeature1 = pFeatureClass1.GetFeature(pTopoErrFeature.OriginOID)
   Set pFeature2 = pFeatureClass1.GetFeature(pTopoErrFeature.DestinationOID)
   Dim j
   'feature, ke které se přiřadí geometrie smazané feature
   Dim pHelpFeature As Ifeature
  'feature, která bude smazána
   Dim pDelFeature As IFeature
   Set pHelpFeature = pFeature1
   Set pDelFeature = pFeature2
```

```
        'testuje existenci již smazané geometrie v MyArray
        Dim blnExists As Boolean
        blnExists = False
        'prochází pozpátku MyArray a hledá, jestli již nebylo s některou feature něco děláno
        For j = i - 1 To 0 Step -1
'hledá, jestli feature1 nemá již přidělenou geometrii jednou, pokud ano, přidá k ní geometrii feature2
            If (MyArray(j, 1) = pFeatureClass1.ObjectClassID And MyArray(j, 2) = pHelpFeature.OID) Then
               Set pHelpFeature = pFeatureClass1.GetFeature(MyArray(j, 2))
               Set pDelFeature = pFeature2
               Call AddFeature(pHelpFeature, pDelFeature)
               blnExists = True
               Exit For
'hledá, jestli feature2, která se má smazat nemá přidělenou geometrii jiného prvku, pokud ano, bude se mazat feature1
            ElseIf (MyArray(j, 1) = pFeatureClass1.ObjectClassID And MyArray(j, 2) = pDelFeature.OID) Then
               Set pHelpFeature = pFeatureClass1.GetFeature(MyArray(j, 2))
               Set pDelFeature = pFeature1
               Call AddFeature(pHelpFeature, pDelFeature)
               blnExists = True
               Exit For
'hledá, jestli feature1 nebyla již smazána, pak přidá geometrii feature2 do geometrie feature uložené v poli MyArray [2]
            ElseIf (MyArray(j, 1) = pFeatureClass1.ObjectClassID And MyArray(j, 3) = pHelpFeature.OID) Then
               Set pHelpFeature = pFeatureClass1.GetFeature(MyArray(j, 2))
               Set pDelFeature = pFeature2
               Call AddFeature(pHelpFeature, pDelFeature)
               blnExists = True
               Exit For
'hledá, jestli feature2 nebyla již smazána, pak přidá geometrii feature1 do geometrie feature uložené v poli MyArray [2]
            ElseIf (MyArray(j, 1) = pFeatureClass1.ObjectClassID And MyArray(j, 3) = pDelFeature.OID) Then
               Set pHelpFeature = pFeatureClass1.GetFeature(MyArray(j, 2))
               Set pDelFeature = pFeature1
               Call AddFeature(pHelpFeature, pDelFeature)
               blnExists = True
               Exit For
          'další uložený záznam v MyArray
           End If
        Next j
'uložení feature, které byla přidána geometrie a smazané feature (nemusí se jednat vždy o feature1 a feature2) do
MyArray
        MyArray(i, 1) = pFeatureClass1.ObjectClassID
        MyArray(i, 2) = pHelpFeature.OID
        MyArray(i, 3) = pDelFeature.OID
        'pokud nenastal zádný jiný případ, bude smazána feature2 a její geometrie přidána do feature1
        If blnExists = False Then
           Call AddFeature(pHelpFeature, pDelFeature)
        End If
        'další pseudo-uzel
        Set pTopoErrFeature = pEnumTopologyErrorFeature.Next
    Next i
```

```
'spojí geometrie obou features do pHelpFeature a smaže pFeature
Private Sub AddFeature(pHelpFeat As IFeature, pFeature As IFeature)

  Dim pGeom1 As IGeometry
  Dim pGeom2 As IGeometry
  Dim pOutputGeometry As IGeometry
  Dim pTmpGeom As IGeometry
  Dim pTopoOperator As ITopologicalOperator
  'kopie geometrií
  Set pGeom1 = pHelpFeat.ShapeCopy
  Set pTopoOperator = pGeom1
```

```
  Set pGeom2 = pFeature.ShapeCopy
  'sjednocení geometrií
  Set pOutputGeometry = pTopoOperator.Union(pGeom2)
  Set pTmpGeom = pOutputGeometry
  Set pHelpFeat.Shape = pTmpGeom
  'uložení geometrie
  pHelpFeat.Store
  'smazání feature
  pFeature.Delete

End Sub
```

6 Závěr

Cílem teoretické části diplomové práce bylo popsání metodiky pro vytváření geografických databází. Nutným základem pro popis metodik bylo zjištění vývoje geografických databázích v jednotlivých generacích GIS. Aby bylo možné si udělat představu o tom, jak různé typy prostorových databází fungují, bylo nutné zjistit způsoby ukládání atributové a prostorové složky dat v jednotlivých databázových modelech. Pro úplnost (podle vývoje) jsem zahrnul do popisu i hierarchický a síťový databázový model, které nebylo možné pro vytvoření geografických databází použít.

Následuje popis jednotlivých metodik pro různé databázové modely. Důležitou složkou modelování je jazyk, kterým se návrhář dorozumívá se zadavatelem, a ve kterém v dalších fázích vývoje vytváří výsledný fyzický model. Standardem se „de facto “ v tomto směru zejména u objektově orientovaného a objektově-relačního modelu stal jazyk UML. Ani vytváření modelů u relačních databází v E-R diagramech neztratilo vzhledem ke své jednoduchosti a rychlosti svůj význam.

Důležitou složkou při návrhu databáze je zahrnutí prostorových vztahů, modelovaných pomocí topologie, která je často opomínána. Samozřejmě záleží na konkrétním systému, jak topologii ukládá, proto je nutné myslet na tuto záležitost od začátku návrhu modelu. Např. ESRI topologie nedovoluje vytváření topologických pravidel napříč datovými sadami (datasets), z tohoto důvodu musí být myšleno na správné rozdělení prvkových tříd (feature classes) v jednotlivých datových sadách na počátku návrhu.

Teoretická část dále popisuje konkrétní návrh databáze na příkladu geomorfologické databáze, která tvoří základ geomorfologického informačního systému (GmIS). GmIS je celý postaven na produktech firmy ESRI a data jsou spolu s topologickými pravidly a sadami nástrojů (Toolboxes) uložena v ESRI Personal Geodatabase.

Praktická část se zabývá popisem částečné automatizace procesů pro úpravu struktury geomorfologické databáze a její provázání topologickými pravidly. Dva vytvořené skripty („Select By Unique“ a „UpdateEF“) zajišťují vyplnění atributové tabulky elementárních forem a dva vytvořené modely v rozšíření „Model Builder“ zajišťují zmíněnou změnu struktury databáze přidáním nových, vyšších hierarchických vrstev a provázání těchto vrstev topologickými pravidly. Poslední skript („Delete Pseudonodes“) pomáhá při čištění topologie vymazáním pseudo-uzlů. Všechny vytvořené skripty jsou publikovány pod licencí GNU/GPL.

Seznam použité literatury

[ABCLinuxu] ABCLinuxu: Výkladový slovník – RDBMS, 2007, [ONLINE], <http://www.abclinuxu.cz/slovnik/rdbms>

[Andrade, et al. 2002] Andrade J., et al.: *Exploring ArcObjects*, ESRI, 2002, ISBN: 1-58948-001-5

[Arctur 2004] Arctur D., Zeiler M.: *Designing Geodatabases (Case Studies in GIS Data Modeling)*, ESRI Press, 2004, ISBN: 1-58948-021-X

[Baars, et al. 2004] Baars M., et al.: *Rule-based or explicit storage of topology structure: a comparison case study*, TU Delft, 2004, [ONLINE] , <http://giscenter.isu.edu/training/ppt/AdvGIS/Topology_comparison_Geodatabase.pdf>

[Batko 2007] Batko: *Relační vs. objektově-relační vs. objektové databáze*, Masarykova univerzita v Brně, Fakulta informatiky, 2007, [ONLINE] , <http://www.fi.muni.cz/~xbatko/oracle/compare.html>

[DB1] *Přednášky z předmětu - Databázové systémy, definice, struktura*, imaturita.cz , 2003, [ONLINE] , <www.imaturita.cz/kky/IRS1/db.pdf>

[ESRI 2004] ESRI: *Creating and Editing Geodatabase Topology with ArcGIS 9 (for ArcEditor and ArcInfo)*, 2004, [ONLINE] , <http://campus.esri.com>

[ESRI 2005] ESRI: *Geoprocessing with ArcGIS Desktop*, 2005, [ONLINE] , <http://campus.esri.com>

[ESRI 2007a] ESRI: *Support center - Downloads for Data Models*, 2007, [ONLINE], <http://support.esri.com/index.cfm?fa=downloads.dataModels.matrix>

[ESRI 2007b] ESRI: ArcGIS 9.2 Desktop Help - Topology rules, 2007, [ONLINE], <http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Topology_rules>

[GIS 2007]: Prezentace – Úvod do GIS, Univerzita Pardubice, 2007, [ONLINE], <http://webak.upce.cz/~uozp/gis4/GIS-iszp2.pdf>

[Glos 2006] Glos P.: *Technologie GIS*, Masarykova univerzita, Ústav výpočetní techniky, 2006,

[ONLINE] , < http://www.ics.muni.cz/to.en.cgi/bulletin/articles/276.html>

[Hernandez 2006] Hernandez M. J.: *Návrh databází*, Grada, 2006, ISBN: 80-247-0900-7

[Hoel 2001] Hoel E., Menon S., Morehouse S.: *Building a Robust Relational Implementation of Topology*, ESRI, 2001, [ONLINE], <gis.esri.com/esripress/shared/images/66/Building_Topologies.pdf>

[Jedlička 2005]: Jedlička K.: *Konvence v pojmenovávání geodatabáze*, Západočeská univerzita v Plzni, Katedra matematiky, 2005

[Jedlička 2007] Jedlička K.: *Úvod do geografických informačních systémů*, ZČU v Plzni, 2007, [ONLINE] , <http://www.gis.zcu.cz/studium/ugi/e-skripta/ugi.pdf>

[Kaliková 2007] Kaliková: *Podklady k předmětu - Databázové a prezentační systémy*, ČVUT - Fakulta dopravní, 2007, [ONLINE], <http://daps.fd.cvut.cz/slovnicek_pojmu.pdf>

[Kennedy 2001] Kennedy H.: *Dictionary of GIS Terminology*, ESRI Press, Redlands, California, 2001, ISBN: 1-879102-78-1

[Longley 2001] Longley P. A., Goodschild M. F., Maguire D.J., Rhind D.W.: *Geographic Information Systems and Science*, John Wiley & Sons, Ltd., 2001, ISBN: 0-471-89275-0

[Louwsma 2003] Louwsma J.H.: *Topology versus non-topology storage structures*, TU Delft, 2003, [ONLINE], <http://www.gdmc.nl/publications/2003/Topology_storage_structures.pdf>

[McClure 1997] McClure S.: *Object Database vs. Object-Relational Databases*, International Data Corporation, 1997, [ONLINE] , <http://www.geog.ubc.ca/courses/geog470/notes/Object%20Database%20vs_%20Object-Relational%20Databases.htm>

[Mentlík, et al. 2006] Mentlík P., et al.,: *Geomorphological information system - physical model and options of geomorphological analysis*, Západočeská univerzita v Plzni, 2006

[Mentlík 2007] Mentlík P.: *Geomorfologie - základní pojmy (určeno studentům bakalářského studia)*, Západočeská univerzita v Plzni, Katedra geografie, 2007, [ONLINE], <http://www.kge.zcu.cz/pesonal/PERSON/mentlikp/vyuka/vyuka.html>

[Minár et al. 2005] Minár j., et al.,: *Geomorphological information system - idea and options of practical implementation*, Komenského univerzita v Bratislavě, 2005

[Mrozek 2006] Mrozek J.: KnihovnaPHP – Příručka PHP pro každého, Zoner software, 2006, [ONLINE], <http://php.interval.cz/clanky/abstraktni-trida/>

[OGC 2006]: Open Geospatial Consortium Inc., *OpenGIS Implementation Specification for Geographic information - Simple feature access - Part2: SQL option*, 2006, [ONLINE], <http://www.opengeospatial.org/standards/sfs>

[Page-Jones 2001] Page-Jones M.: *Základy objektově orientovaného návrhu v UML*, Grada, 2001

[Perchta 2007] Perchta: *Geografické informační systémy - přednášky*, Vysoké učení technické v Brně, 2007, [ONLINE] , <perchta.fit.vutbr.cz/vyuka-gis/uploads/1/gis4db.pdf>

[Pokorný 1999] Pokorný J.: *Konstrukce databázových systémů*, Vydavatelství ČVUT, 1999, ISBN: 80-01-01935-7

[Příroda 2006] PŘÍRODA.cz: *Odborný ekologický a přírodovědný slovník, pojmy na písmeno T*, , 2006, [ONLINE], <http://www.priroda.cz/slovnik.php?detail=876>

[Ressler 2006] Ressler M.: *Elektronické vydání - Informační věda a knihovnictví : Výkladový slovník české terminologie z oblasti informační vědy a knihovnictví. Výběr z hesel v databázi TDKIV*, Vysoká Škola Chemicko-Technologická v Praze ve spolupráci s Národní knihovnou ČR, 1. vyd. 2006, [ONLINE] , <http://vydavatelstvi.vscht.cz/knihy/uid_es-005/motor/main.obsah.html>

[Rigaux 2002] Rigaux P., Scholl M., Voisard A.: *Spatial databases (with application to GIS)*, Morgan Kaufmann Publishers, 2002, ISBN: 1-55860-588-6

[Rigaux2002] Rigaux P., Scholl M., Voisard A.: *Spatial databases (with application to GIS)*, Morgan Kaufmann Publishers, 2002, ISBN: 1-55860-588-6

[Swiki 2007] Swiki: *Co je to topologie?*, 2007, [ONLINE] , <http://perchta.fit.vutbr.cz:8000/uits-seminar/uploads/15/Topologie.pdf>

[Tilton et al.2002] Tilton T., et al.: *Introduction to Programming ArcObjects with VBA - exercise 6*, ESRI, 2002

[Tomlinson 2003] Tomlinson R.: *Thinking About GIS*, ESRI Press, 2003, ISBN: 1-58948-070-8

[Tuček 1998] Tuček J.: *Geografické informační systémy (Principy a praxe)*, Computer Press, 1998, ISBN: 80-7226-091-X

[Ullmann 1983] Ullmann V.: *Geometrie a topologie prostoročasu*, Klinika nukleární medicíny FNsP v Ostravě, 1983, [ONLINE], < http://astronuklfyzika.cz/Gravitace3-1.htm>

[Vokounová 2003] Vokounová L.: *Návrh struktury datového modelu pro správu elektrických distribučních sítí ZČE v GIS analýzou mezinárodního datového modelu ArcFM*, Západočeská univerzita v Plzni, Katedra matematiky, 2003

Příloha

Slovníček pojmů

Většina pojmů je zpracována podle [Kennedy 2001]. Citace z jiných zdrojů je uvedena u příslušného pojmu.

Layer (Vrstva) – je sada vektorových dat, organizovaná podle určitého tématu, jako jsou silnice, řeky, nebo hranice států, a sada rastrových dat, reprezentujících část zemského povrchu, např., letecké nebo satelitní snímky.

Feature (Geoprvek) – se označuje objekt v krajině nebo na mapě, nebo reprezentace prostorové datové vrstvy (bod, linie, polygon), která je geografickým objektem.

Dataset (Datová sada) – soubor dat společného tématu.

Feature class (Prvková třída) – soubor prostorových dat stejné reprezentace (bod, linie, polygon), uložených ve formátech shapefile, coverage nebo Geodatabase.

Feature data set – soubor feature classes, které sdílí stejný souřadnicový systém.

Uzel (Node) – je (podle [GIS 2007]) bod na konci linie nebo bod, kde se linie kříží.

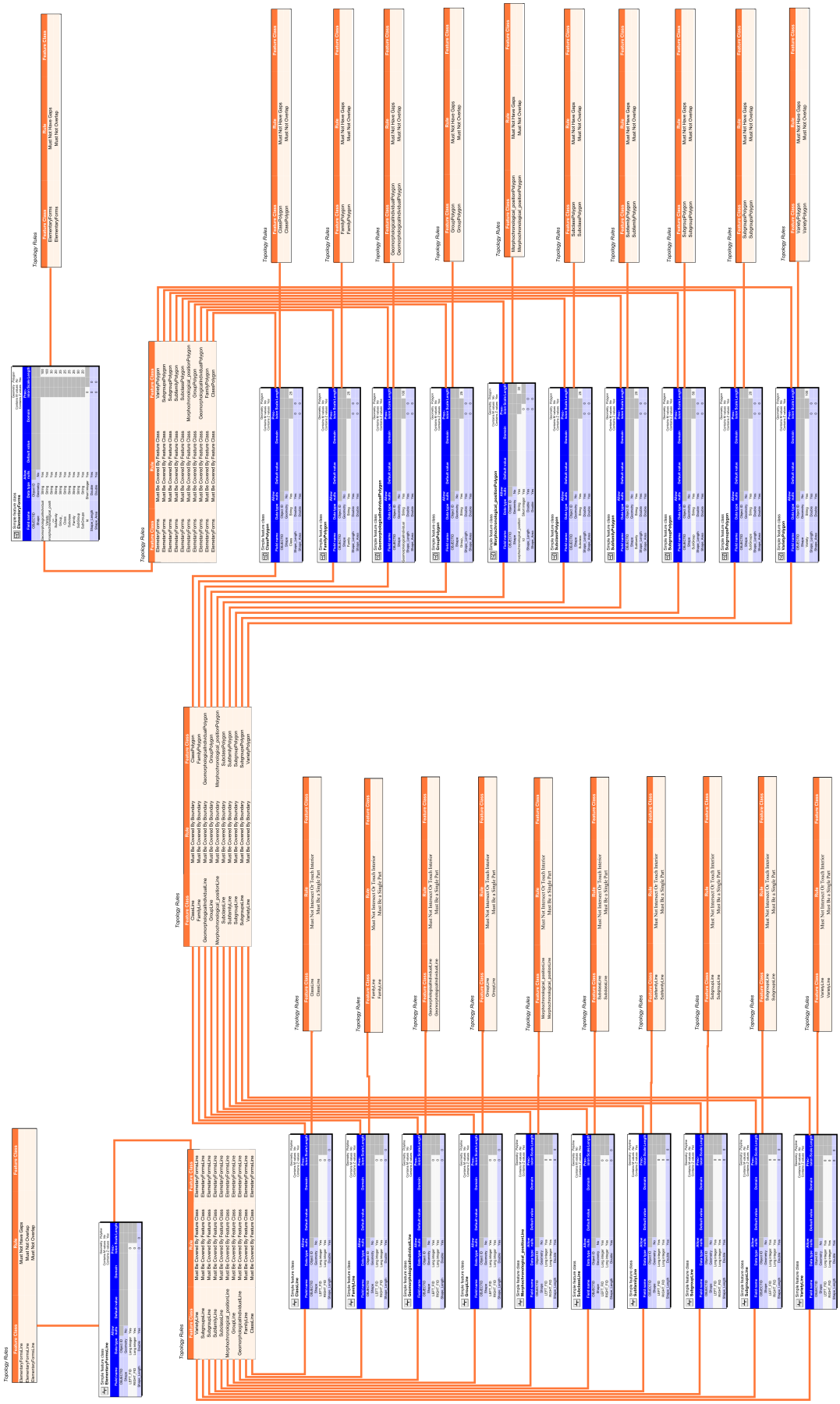
Database management systém (DBMS) – je soubor počítačových programů, které organizují informace v databázi a poskytují nástroje pro vstup, ověření a ukládání dat.

Relational database management system (RDBMS) – je (podle [ABCLinuxu]) databázový server, který spravuje databáze, komunikaci s klienty (lokálními nebo vzdálenými), vstupy a výstupy dat a jejich integritu. (Příklady: MySQL, Postgre SQL, Oracle, DB/2, Sybase, MS SQL Server, aj.).

Geodatabase (Geodatabáze) – je formát pro ukládání dat. Reprezentuje geografické prvky a atributy jako objekty a je přítomen v RDBMS.

Geographic database (Geografická databáze) – soubor prostorových dat a jejich atributů, efektivně organizovaný a snadno přístupný.

Fyzický model topologie



Adresářová struktura CD

-- Databaze

- puvodni
  - GmIS.mdb – databáze před vytvořením vyšších forem
  - GmIS.mxd – projekt s uloženými skripty
  - Documentation – help ke skriptů
- hotova
  - GmIS.mdb – vytvořená databáze se všemi vyššími formami
  - GmIS.mxd – projekt s uloženými skripty
  - Documentation – help ke skriptům

-- Prilohy

- zip – komprimovaný adresář se všemi skripty
- topologie.pdf – logický a fyzický model vytvořené topologie

-- TextDp

- Vracovsky_Geomorfologicka_databaze_DP.pdf – text diplomové práce

-- OnlineZdroje – online zdroje použité v diplomové práci

ERRATA

str. 56

Obr. 5.2: Elementární formy u Prášilského jezera vymezené v práci [Mentlík 2006].

str. 57

Obr. 5.3: Atributová tabulka ElementaryForms – ukázka z programu ArcCatalog. Atributy vyšších hierarchických forem byly vymezený v [Mentlík 2006].

str. 77

[Mentlík 2006] Mentlík, P. (2006): *Geomorfologická analýza a tvorba GmIS pro okolí Prášilského jezera a jezera Laka na Šumavě* (Česká republika) [disertační práce]. Bratislava. Univerzita Komenského v Bratislave. 252 s.