1. ALGORITMUS A ZÁKLADNÍ POŽADAVKY NA ALGORITMUS

Definice: Algoritmem rozumíme postup řešení, kterým lze řešit třídu problémů.
Každý algoritmus musí mít tyto vlastnosti:
Konečnost a resultativnost
ukončí se v reálném čase a s konkrétním výsledkem
Hromadnost
není pouze pro jeden problém, ale pro celou řadu (třídu) problémů (aritmetic. průměr)
Jednoznačnost

VLASTNOSTI ALGORITMU

Jednoznačnost

- přechod do následujícího stavu algoritmu je jednoznačně určen výsledkem stavu předchozího
- algoritmus složen z kroků
- každý krok je charakterizován jako přechod z jednoho stavu do jiného
- stav algoritmu je dán zpracovávanými daty (stavem dat po zpracování kroku) a to určí následující krok

3 TYPY ALGORITMICKÝCH PROBLÉMŮ

1. řešitelné za přijatelnou dobu a na realizovatelném počítači
2. řešitelné, ale nepřijatelná doba ale realizovatelný počítač
3. neřešitelné z principu

1. ŘEŠITELNÝ ALGORITMUS

požadavek na **skončení** algoritmu pro libovolný **legální** vstup (sčítání dvou písmen)
vrácení **správného** výsledku

Časová složitost
funkce počtu zpracovaných elementů **O=f (N)**
Prostorová složitost
kolik paměti pro provedení výpočtu (i dynamické struktury- zásobník)

DEFINICE SLOŽITOSTI

Je-li definice složitosti dána funkcí O:
$O = N^2 + 5N – 5$, pak říkáme, že složitost je řádu N^2 a zapisujeme $O(N^2)$,
kde N je počet zpracovaných elementů

obvyklé řády složitosti:
$1, Log_2N, N Log_2N, N, N^2, N^3, 2N, N!, N^n$

rozumně použitelné alg. s polynomiální složitostí, exponenciální složitost nejsou prakticky použitelné

NUMERICKÉ A NENUMERICKÉ ALGORITMY

numerické algr.
operace nad čísly,
řeší matematické problémy
nenumerické algr.
obecné úlohy,
vstup není množina čísel, ale musí se transformovat do numerické oblasti

PŘÍKLAD NENUMERICKÉ ÚLOHY

Bludiště
Jak lze najít cestu ven z bludiště, existuje-li?
Řešení:
Na každém rozcestí, kterého se dosáhne, se zjistí všechny možné cesty, jimiž lze pokračovat, vybere se jedna z nich, ostatní se zapamatují.
Tímto postupem se postupuje dopředu tak dlouho, pokud to jde nebo dokud se nedostane do situace, ve které se již bylo.
Nejde-li postupovat dál, návrat na poslední rozcestí, kde došlo k rozhodnutí a výběr jiné cesty, kterou se dosud nešlo.
Speciální případ algoritmického vyhledávání – prohledávání s návratem – backtraking.
nevýhoda – velká časová náročnost

ZPŮSOB ZÁPISU ALGORITMU

slovní popis (kuchařka)
vývojový diagram
strukturogram
zápis v programovacím jazyku
(pseudojazyku)

VÝVOJOVÝ DIAGRAM

Definice: Vývojový diagram je orientovaný graf sloužící pro zápis algoritmu, jehož uzly odpovídají jednotlivým krokům nebo stavům zpracování úlohy a hrany určují směr postupu výpočtu.

ČSN ISO/IEC 14598-1 (369028) Informační technologie - Hodnocení softwarového produktu
Symboly vývojových diagramů pro systémy zpracování dat

PSEUDOJAZYK

návrat k textovému popisu
odsazení textu doprava
každá dílčí struktura začíná a končí speciálními příkazy – podoba se skutečnými jazyky (BASIC, Pascal, Visual Basic)

PŘÍKLAD

Sestavte program pro výpočet poloviční hodnoty součtu a rozdílu vstupních čísel X a Y.

ŘÍDÍCÍ STRUKTURY – VĚTVENÍ PROGRAMU

PŘÍKLAD – NAJDI VĚTŠÍ ČÍSLO

Sestavte algoritmus, který ze dvou vstupních čísel vytiskne větší z nich.

```
čti X, Y
max = X
if max < Y then
    max = Y
end if
tisk X, Y, max
```

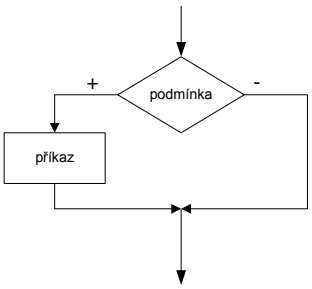
2. ŘÍDÍCÍ STRUKTURY V PROGRAMOVÁNÍ - VĚTVENÍ A CYKLY SE ZNÁMÝM A NEZNÁMÝM POČTEM PRŮCHODŮ

ŘÍDÍCÍ STRUKTURY

Programátor potřebuje ovládat tok provádění příkazů, jinak se příkazy provádějí v pořadí zleva doprava a shora dolů. Řídící struktury umožňují *větvení* programu. Řídící struktura mění pořadí příkazů a počet vykonání.
Dělíme je na **rozhodovací struktury** a **smyčky** (cykly).

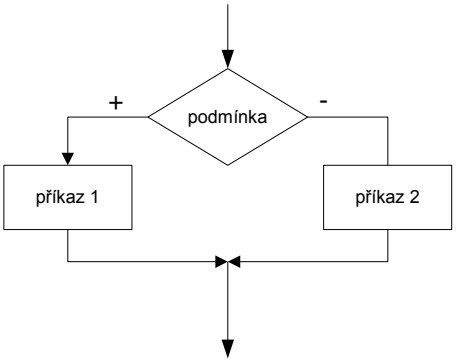
Rozhodovací struktury

If *podmínka* **Then** *příkaz*



If *podmínka* **Then**
příkazy
Else
příkazy

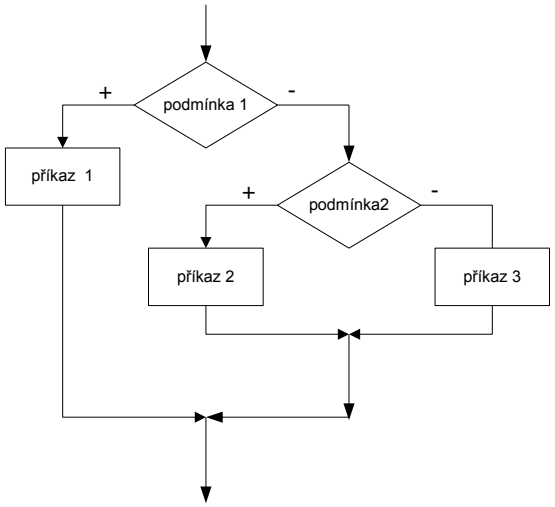
End If



If *podm1* **Then**
blok příkazů1
ElseIf podm2 **Then**
blok příkazů2

Else
blok příkazů 3

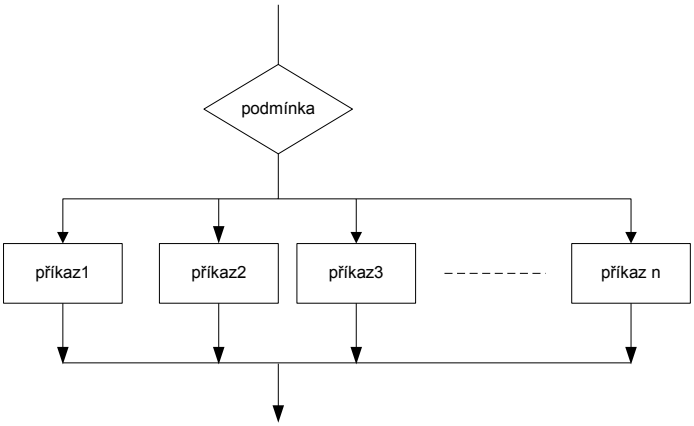
End If



Select Case testovací výraz
Case výraz2
příkazy1
Case výraz2
příkaz2

Case Else
Příkazn

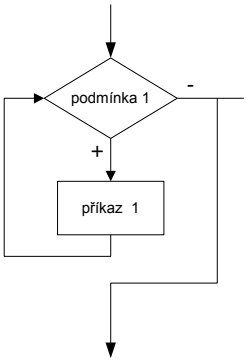
End Select



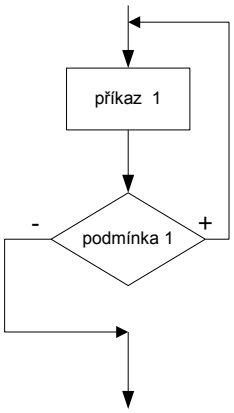
Struktury cyklu

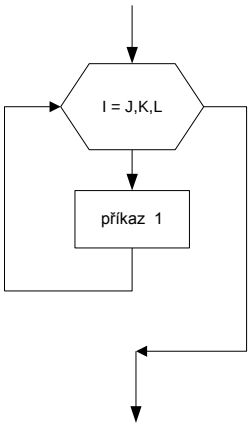
Do While podmínka
příkazy

Loop



Do
Příkazy
Loop While *podmínka*
'provede se aspoň jednou





For počítadlo=start **To** konec [Step přírůstek]
Příkazy

Next [počítadlo]

For Each prvek **In** skupina
Příkazy
Next prvek
Opuštění řídící struktury **Exit For, Exit Do**
Návěští - umožní skok na jiné (označené) místo programu.
příklad: Pokracuj: Sem: Znovu: Konec:
Go To Konec
Série příkazů s objektem:
With objekt
[příkazy]
End With

ŘÍDICÍ STRUKTURA - CYKLUS

důvod – opakování stejných příkazů
slovně : „Pokud platí B opakuj P.“ , kde
B je podmínka
P je skupina příkazů – tzv. tělo cyklu

známý počet opakování x neznámý počet opakování
rozlišujeme několik typů cyklů
1. Cyklus *While do*
předem neznámý počet průchodů
cyklus končí při nesplnění podmínky
nemusí se provést ani jednou, končí hned při nesplnění podmínky
2. Cyklus – *Repeat Until*
„opakuj B dokud platí P“
předem neznámý počet průchodů
cyklus končí při nesplnění podmínky
provede se aspoň jednou

Konečnost cyklu

chyba při spuštění
zacyklení – neustálé opakování stejných akcí
není projev navenek, obtížná identifikace
příklad

```
While I <> 0 Do
    J = I -1
End While
```

Tělo cyklu nemá vliv na hodnotu **I** !!!!
Příkazy v těle cyklu musí měnit hodnotu proměnné, které je v podmínce.
1. Konečnost cyklu
Je tam příkaz, který mění I, ale..
I < N – je funkční (neprojde ani jednou)
I >N - zacyklí se
Počáteční hodnoty proměnných mají vliv na počet průchodů, zda se vykoná cyklus.
Musí se měnit hodnota testované proměnné v těle.
Změna musí být vést k ukončení cyklu
2. Cyklus *FOR*
„N-krát opakuj P.“
Známý počet průchodů – N průchodů
Varianty:
Počet průchodů je pevný, je předem znám (není univerzální algoritmus)
Počet prvků se zjišťuje za běhu programu (je univerzální algoritmus)
Přepínač SELECT

Podle výsledku podmínky **P** proveď příslušný příkaz **PříkazN**.

3. DATOVÉ STRUKTURY: PROMĚNNÁ, POLE, VÍCEROZMĚRNÁ POLE. DATOVÉ TYPY

PROMĚNNÉ

Proměnné slouží k dočasnému uchování hodnot během vykonávání programového kódu.
Jsou definovány **názvem**, **datovým typem, rozsahem platnosti** a **dobou platnosti**.
Pravidla pro **názvy** proměnných a konstant:

- nepoužívat klíčová slova, nesmí obsahovat: mezeru, čárku, &, #, @, $, %, !
- musí začínat písmenem
- bez diakritiky
- jména proměnných, konstant a podprogramů musí být kratší než 255 znaků
- jména formulářů, ovladačů, tříd a modulů musí být kratší než 40 znaků – bez diakritiky, začínají písmenem, neklíčová slova

Datové typy:

Datový typ	T	**Rozsah hodnot**	**Paměť v bytech**
Byte		0-255	1
Boolean		True, False	2
Integer	%	-32 768 až 32 767	2
Long	&	-2 147 483 648 až 2 147 483 647	4
Single	!	-3,402 823E+38 až –1,401 298E-45; 1,401 298E-45 až 3,402 823E+38	4
Double	#	-1,797 693 134 862 32E+308 až –4, 940 656 458 412 47E-324; 4,94065645841247E-324 až 1,79769313486232E308	8
Currency	@	-922 337 203 685 477,5808 až 922 337 203 685 477,5807	8
Decimal		96bitová čísla bez znaménka	14
Date		1.1.100 až 31.12.9999	8
Object		Odkaz na jakýkoliv objekt v aplikaci nebo jiných aplikacích	4
String	$	0 až 2 miliardy znaků	10+délka řetězce
String*délka	$	1 až 65 400 znaků	délka řetězce
Variant		číselná proměnná až do rozsahu Double String proměnné délky	16 22+délka řetězce
Uživatelsky definovaný		rozsah pro každý prvek je shodný s rozsahem jeho datového typu	

Pole jakéhokoli datového typu zabírají 20B paměti a 4B pro každý rozměr pole, plus počet bytů zabraných vlastními daty.
Rozsah platnosti
Udává, ve které části – a v jak velké části – aplikace má jméno proměnné (konstanty) vazbu na tutéž entitu (proměnnou, objekt …).

- *Lokální* – platí v dané proceduře, či funkci. Deklarují se uvnitř procedury příkazem Dim.
- *Modální* – platí v daném formuláři. Deklarují se v záhlaví modulu formuláře příkazem Dim.
- *Globální* – platí v celém projektu. Deklarují se v basickém modulu příkazem Public.

Doba platnosti
Deklarace Dim – proměnná existuje pouze při běhu procedury, při novém spuštění se znovu inicializuje.
Static – hodnota se uchová i po skončení běhu procedury a vstupuje do ní i při jejím dalším volání.

Deklarace proměnné

Deklarace nejsou povinné, avšak po zápisu klíčového slova **Option Explicit** jsou vyžadovány. Proměnná se musí deklarovat před prvním použitím. Má tu výhodu, že předejdeme překlepům v názvech proměnných.

Není-li proměnná deklarována je typu Variant.

a) Jednoduchá proměnná

Deklarace se provádí zápisem
{Dim, nebo Private, nebo Public} jméno_proměnné [As typ]*
např.

Dim sestra As String
*Public jmeno As String * 20 'definována pevná délka 8 znaků*

b) Uživatelem definovaný typ

Deklaruje se pouze v deklarační části basického modulu nikoliv procedur.
Např.

```
Type zamestnanec
  Jmeno As string
  Prijmeni As string
  Vykonnost As integer
End type
```

Po vytvoření tohoto uživatelského typu – deklarace proměnné delnik:
Dim delnik As zaměstnanec
Použití:
delnik.jmeno = „Jan"

c) Pole

Proměnná na jejíž prvky se odkazujeme indexem. Jeho prvky mají stejný datový typ (výjimka je variant).
Deklarace: {Dim, nebo Private, nebo Public} jméno_pole[([indexy])] [As typ][1]

jednorozměrná pole

Např.:
Dim intPole (20) As Integer
Dim Pole(1 To 5) As String

Pole

aaaa	bbbb	cccc	dddd	eeee
Pole(1)	Pole(2)	Pole(3)	Pole(4)	Pole(5)

vícerozměrná pole - pro uložení matic
Např.
Dim matice (1 To 10, 1 To 10) As Integer
dynamická pole - pro případ, kdy neznáme rozměr a meze pole a jsou upřesněny až za běhu programu
deklarace : *Dim intPole() As Integer*
před prvním použitím klíčové slovo ReDim
N=20
ReDim intPole (1 To N) As Integer
ReDim intPole(0) ' odalokace paměti

d) Konstanty

Uživatel může definovat vlastní konstanty následující deklarací :
[Public][Private] Const názevkonstanty [As typ] = výraz
Const pi = 3.14159265358979
Const Sto=100
Sám VB má množství konstant, které jsou veřejné.

POČÁTEČNÍ HODNOTY PROMĚNNÝCH

Numerické proměnné => 0
Řetězce proměnné délky => řetězec nulové délky
Řetězce pevné délky => řetězec obsahující binární nuly
Variant => Empty

PROMĚNNÁ X POLE PROMĚNNÝCH

Proměnná – paměťové místo pro dočasné uložení hodnot.
Po skončení programu se obsah ztrácí.
Jednoduchá **proměnná** nestačí.
př.: A, I, J, K, Mocnina, SUMA, MIN, MAX, Prumer

Pole – datová struktura
Několik paměťových míst stejného jména
př.: A(I), B(I), SouradniceX (I), Cetnost(I)
Pozice v poli označena indexem

[1] Parametry uvedené v hranatých závorkách nejsou povinné, u parametrů ve složených závorkách musí být použit jeden z uvedených.

A(I) 1 2 3 4 ……N

4. PRINCIPY OBJEKTOVÉ ORIENTOVANÉ PROGRAMOVÁNÍ

ZÁKLADNÍ POJMY

Objekt, vlastnost, metoda, událost

Jazyk VB pracuje s objekty.

Objekt je určitá simulace (malé) části reálného nebo abstraktního světa, která je dána:

- **vlastnostmi** (properties)
- **metodami** (methods)
- **událostmi** (events).

Příkladem reálného objektu může být například *člověk*. Jeho vlastnostmi jsou: věk, barva očí, výška, … Kromě vlastností má člověk i určité „schopnosti" nebo „způsoby chování", například umí chodit. Z hlediska programátora mohou být vlastnosti objektu chápány jako specifické proměnné s daným typem objektu. Metody jsou pak podprogramy pracující nad těmito (ale i jinými) proměnnými.

Dalším příkladem objektu může být obyčejný *nafukovací balónek*. Ten má vlastnosti barvu, rozměr, stav nafouknutý (ano/ne), vypuštěný, stáří (což je neviditelná vlastnost). Metody, které můžeme na balónku uplatnit jsou na nafukování, vyfukování. Událost, která může nastat je propíchnutí špendlíkem. Reakcí na tuto událost je splasknutí a protržení balónku a rána. Je to reakce objektu na vnější událost.

Třída objektu

Třída objektu představuje *obecnou* definici objektu – jeho vlastností a metod a událostí. To znamená, že je například stanoveno, že auto má barvu (obecně) spolu s uvedením všech možných barev, které auto může mít. Třída objektu je vzorem (šablonou) pro vytváření *instancí* objektu.

Třídou objektu jsou okno operačního systému Windows.

Instance objektu

Instance objektu jedná se o konkrétní objekt - výskyt objektu. Tedy auto se státní poznávací značkou OVD 45 21, číslem motoru 445821E, … Vlastnosti instance objektu už tedy nabývají konkrétních hodnot a objekt je jednoznačně identifikovatelný. Taktéž dialogové okno Windows, které v kterém je dotaz na Restart, vypnutí počítače, má své rozměry, barvu, umístění na obrazovce, jméno.

Metoda

Objekty mají definované metody. Metody jsou spojovány s chováním objektu. Dynamická stránka objektu je vyjádřena jako množina operací, které objekt může provádět za určitých podmínek. Metody objektu jsou procedury nebo funkce, které vykonávají určitou činnost nad vnitřní pamětí objektu. Nad objektem lze vykonat jen metody, které má definované.

Událost

Běh programu, či jeho části je aktivován tzv. událostmi. Událost nastane po nějaké specifické akci uživatele nebo systému. Např. metoda zobrazení formuláře je reakcí na určitý podnět uživatele (kliknutí na tlačítko „Otevři", nebo stisknutí nějaké klávesy na klávesnici – událost na tlačítku, nebo na klávesnici). Běžící program čeká na událost. Poté co událost nastane, spustí se kód, který je zapsán do procedury (podprogramu) svázané s touto událostí.

PRINCIPY A VÝHODY OBJEKTOVĚ ORIENTOVANÉHO PROGRAMOVÁNÍ (DÁLE OOP)

1. **Opakovatelnost použití objektu** – je-li definována třída objektu, lze snadno vytvořit „nekonečně mnoho" instancí třídy. Např. je-li obecně nadefinováno tlačítko, které má určitý vzhled a chování, je možné na základě této definice vytvořit více tlačítek (tlačítko OK, Cancel, Retry, ...)
2. **Identita objektů** – každý objekt musí být jednoznačně identifikovatelný.
3. **Dědičnost** (inheritance)– jedna třída objektu může být odvozena od jiné – nadřazené třídy, dědí poté vlastnosti a metody nadřazené třídy. Vznikají určité hierarchie tříd objektů. Například je-li definována třída „Dopravní prostředek" s vlastnostmi „počet přepravovaných osob" a „maximální rychlost", mohou být od ní odvozeny nové třídy např. „letadlo", „lod'", „automobil" atd., které daní vlastnosti dědí. (ve VB je dědičnost realizována poměrně složitými postupy)
4. **Rozšiřitelnost** (extensibility**)** – definici tříd lze dále doplňovat a upravovat. Nová třída automobil by pak mohla mít rozšířena o vlastnosti jako: „počet kol", „počet válců" atd.
5. **Zapouzdření** (encapsulation)– objekt by měl být samostatnou jednotkou a měl by mít jasné hranice. Zapouzdřujeme k objektu jeho vlastnosti, metody a můžeme k tomuto objektu přistoupit jej přes tyto metody. Tímto způsobem vlastně objekt chráníme (zapouzdříme), aby jej nebylo možné ovládat jinak, než přes jeho vlastní metody.
6. **Polymorfismus –** Nová třída má stejné metody (jejich názvy) jako u nadřazené třídy podle principu dědičnosti. Avšak metoda stejného jména může mít jinou funkcionalitu. Platnost lokální metody má vyšší prioritu než zděděna metoda.

Visul Basic a objekty

Ve VB jsou předdefinované objekty nebo objekty nově definované uživatelem.

Všechny **ovládací prvky** jsou předefinované objekty.

Příklady **objektů** : formuláře (form), textová pole (Text Box)(pro zadávání i zobrazování), příkazová tlačítka (CommandButton), popisky (Label), menu, atd.

Pomocí předdefinovaných objektů komponent lze vytvořit program snadněji a rychleji. Jedním z nejčastěji používaných objektů v programech je **tlačítko**. Jestliže tedy máme k dispozici definici třídy tlačítko, které je nadefinováno jako obdélník s určitými rozměry, vzhledem, barvou, popisem atd., není třeba při každém použití tlačítka v programu znovu pracně definovat tyto vlastnosti. Navíc jsou tyto kódy již odladěny, takže nevznikají další chyby. Při použití programové komponenty uživatel nepracuje přímo s kódem, ale přes komunikační prostření programovacího jazyka přistupuje přímo k vlastnostem, metodám a událostem dané třídy. Definice třídy by měla být univerzální pro širší použití, ale lze nalézt i zcela úzce zaměřené komponenty. Korektní použití programové komponenty zajišťuje přesná dokumentace.
Představíme-li si formulář jako jedno z oken ve Windows, potom jako vlastnost můžeme uvést barvu, titulek, velikost okna, atd. … a metodu například zobrazení formuláře na obrazovce počítače.
Objekty mají také obvykle nějaké vztahy s okolím. Například okno jež je součástí nějaké aplikace ve Windows lze zavřít kliknutím na křížek v pravém rohu tohoto okna, ale také uzavřením celé aplikace, protože je vázáno na běh aplikace (je její součástí).

5. VISUAL BASIC

ÚVOD DO VISUAL BASIC

Ještě dnes si mnoho lidí spojuje Basic s Visual Basicem (dále jen VB). Je pravdou, že z něj VB původně vycházel. Dnes už mnoho společného nemají. VB je plnohodnotným programovacím jazykem. VB umožňuje rychle vytvářet a implementovat distribuované aplikace a programovat webové aplikace s využitím známých nástrojů a funkcí VB.
VB obsahuje nástroje a objekty pro používání databází (DAO, ADO, ODBC, OLEDB atd.). Součástí těchto nástrojů je také plnohodnotný jazyk SQL. VB také podporuje mechanizmy OLE, neboli OLE Automation (Object Linking and Embedding), pomocí nichž je možno používat objekty jiných aplikací (např. dokument Wordu), dále také mechanizmus DDE (Dynamic Data Exchange), který umožňuje výměnu dat mezi aplikacemi. Dále VB obsahuje mnoho objektů pro různé akce, např. komunikace v síti, s tiskárnami, grafické a multimediální prvky atd.
Dnes lze VB verze 6 koupit ve třech verzích, **Learning**, Professional a Enterprise. Verze Learning, jak napovídá název, je určena spíše pro učení se jazyku. Ze všech třech je nejlevnější. Navíc oproti dvěma dalším verzím má na CD interaktivní učebnici. Pokud však mají být programy prodávány, či jinak distribuovány, je nezbytné zvolit verzi Professional nebo Enterprise. Obě tyto verze umožňují vše, co je uvedeno výše, obsahují integrované nástroje pro přístup k databázím atd. Verze Enterprise obsahuje navíc různé nástroje pro vyvíjení aplikací client-server, pro programátorské týmy apod. Cenově se rovná asi dvojnásobku verze Professional.
Slovo Visual v názvu znamená vytváření programu vizuálně, tedy hlavně za pomocí myši a vyplňování různých dialogů. Není to až tak úplně pravda, za pomocí myši a dialogů vytvoříte grafické rozhranní aplikace, a maximálně nějaké základní úkoly. Pokud však chcete vytvořit trochu složitější aplikaci, bez psaní kódu se neobejdete.

Objekty VB

- Form
- Command Button (Příkazové tlačítko)
- Text Box (Pole textu)
- Label (Nepřístupný text)
- Check Box (Možnost)
- Option Button (Volba mezi)
- Frame (Rámeček)
- Picture Box (Obrázek)
- Image (Obrazové tlačítko)
- Line (Čára)
- Shape (Tvar)
- List Box
- Combo Box
- Timer (Časovač)
- Horizontal Scroll Bar, Vertical Scoll Bar (Posuvníky)
- Drive List Box (Seznam jednotek)
- Directory List Box (Seznam adresářů)
- File List Box (Seznam souborů)
- Common Dialog (Standardní dialog)
- Dialogy Open a Save As

6. TEORIE DATABÁZOVÝCH SYSTÉMŮ

Data je výraz pro údaje, používané pro popis nějakého jevu nebo vlastnosti pozorovaného objektu. Data se získávají měřením nebo pozorováním, a lze je dělit na data spojitá a data atributivní. Data spojitá se přitom vztahují k nějaké spojité stupnici, zatímco data atributivní nikoliv.

Data jsou:

- vyjádření skutečností formálním způsobem tak, aby je bylo možno přenášet nebo zpracovat (např. počítačem)
- číselné nebo jiné symbolicky vyjádřené (reprezentované) údaje a hodnoty nějakých entit nebo událostí
- jakékoliv fyzicky (materiálně) zaznamenané znalosti (vědomosti), poznatky, zkušenosti nebo výsledky pozorování procesů, projevů, činností a prvků reálného světa (reality)
- surovina, z níž se tvoří informace

Databáze (neboli **datová základna**) je určitá uspořádaná množina informací (dat) uložená na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento software se v české odborné literatuře nazývá systém řízení báze dat (SŘBD). Běžně se označením *databáze* – v závislosti na kontextu – myslí jak uložená data, tak i software (SŘBD).

FUNKCE A MOŽNOSTI DATABÁZOVÉHO SYSTÉMU

Základní činností databázového programu je operace s evidencí, resp. můžeme říci s více evidencemi najednou. Takovou evidenci si jistě umíme všichni dobře představit Může se jednat o seznam pracovníků určité firmy, seznam studentů školy, seznam knih, kterými disponuje knihovna. Dále místenková kancelář vlaků, letecké společnosti, divadla. V návaznosti na to evidence výpůjček, seznam uchazečů a mnoho dalších příkladů.

Jaké informace budeme od dobrého databázového systému požadovat a očekávat?

1. **Založení evidence**. To je pochopitelně první krok, při kterém si řekneme co vlastně budeme evidovat, sledovat. Řekneme si například, že budeme chtít podchytit všechny studenty jedné střední školy, přičemž budeme sledovat jejich jméno a příjmení, rodné číslo, známky na vysvědčení, bydliště.
2. **Naplnění daty**. Máme-li kostru naší evidence, budeme do ní vkládat, zapisovat jednotlivé údaje - v našem případě budeme zapisovat informace o jednotlivých žácích.
3. **Měnit zapsaná data**. Život je změna, nic není v úplném klidu. Student se přestěhuje a náš program nám musí umožnit tyto změny provést v naší evidenci. Vedle individuálních změn (viz změna bydliště) můžeme snadno provádět i skupinové změny (například na začátku roku povýšit u studentů jejich třídu, ročník).
4. **Doplnit další sledované údaje**. Na začátku jsme si řekli, co budeme naší evidencí sledovat. Ale po čase se nám může zdát, že jsme některé údaje zapomněli zařadit - například údaje o rodičích. Takže i toto by měl databázový program dokázat. Změní se vlastně počet atributů entit, rozšíří se struktura tabulky.
5. **Mazat data**. Student nestačí na studium, je vyloučen a my jej snadno vyřadíme z evidence. Opět můžeme vedle individuálního "vyškrtnutí" snadno provést i vyřazení celého ročníku (celé skupiny).
6. **Zapisovat nová data**. Přejde-li na naši školu student z jiné školy, opět pomocí databázového programu jej snadno zaevidujeme zaneseme do naší evidence. Přidáme nový záznam do tabulky.
7. **Vypočítávat další údaje**.Sledujeme známky z jednotlivých předmětů. Pro databázový program bude snadné vypočítat celkový průměrný prospěch. A pro přísnější hodnoceni si můžeme vytvořit své vlastní hodnocení tím, že přiřadíme jednotlivým předmětům "váhu" (matematika a český jazyk budou mít váhu 5, zeměpis a dějepis 3 a tělocvik 1). Nad našimi daty můžeme tedy vytvářet složitější funkce.
8. **Řadit seznamy**. Máme údaje zaznamenány a budeme určitě požadovat seznam studentů seřazený podle abecedy, či podle prospěchu (a to jak vzestupně tak sestupně), seřadit po třídách a v rámci tříd abecedně. Seřadit evidenci bude jedním z nejčastějších požadavků.
9. **Vybírat údaje**. Nejčastějším požadavkem je výběr dat jako odpověď na naše dotazy. Chceme zjistit, kolik studentům má vyznamenání (průměrný prospěch 1,5), kolik jich bydlí v Pardubicích, kolik procent máme dívek, jaká je souvislost mezi vzděláním rodičů a prospěchem žáků a podobně. Na všechny tyto dotazy by měl dobrý databázový systém odpovědět (musíme ovšem požadovaný jev sledovat - například musíme sledovat dosažené vzdělání rodičů).
10. **Formuláře**. Představte si situaci, kdy přecházíte z ruční evidence na počítačovou. Pracovnice, které mají zadávat data, budou určitě rozladěné a nebudou chtít s počítačem pracovat. Abychom jim přechod podstatně ulehčili, je velmi praktické vytvořit v počítači formuláře (například osobní karty studentů), které budou k nerozeznání od papírových. Na obrazovce budeme mít formuláře, kde jednotlivá políčka budou na stejném místě jako na papíře a mohou být stejně zvýrazněna (barvou, podtržením).

11. **Tiskové sestavy**. Pochopitelně vše, co z databázového programu dostaneme (seřazený seznam, odpovědi na dotazy, osobní karty formuláře), požadujeme snadno vytisknout. Dokonce můžeme vytvářet další a další zajímavé sestavy včetně skupinových součtů či (průměrný prospěch za třídy, ročníky a nakonec za celou školu). Tyto skupinové součty a průměry nikam nezadáváme ty si program vypočítá na základě našeho požadavku z osobních údajů jednotlivých studentů.
12. **Export / import**. Při naší práci se často jistě stane, že některé údaje budeme již budou existovat, ale v jiném formátu - evidence bude vytvořena jiným programem. Pakliže se jedná některý ze standardů, pak ji do systému snadno převezmeme. A naopak, ze systému lehce předáme naše údaje do jiných systémů. A nemusí se jednat pouze o přenos mezi databázovými programy, ale i o přenos mezi textovými editory a tabulkovými procesory.
13. ***Makra, moduly***. Práci je možné zjednodušit pomocí maker a modulů. Makro je zaznamenaná sled několika jednoduchých úkonů, které často provádí. Tento sled si zapíšeme a uložíme pod vybranou kombinaci kláves. A pak stisknutím například kláves SHIFT-F4 se rychle provede seřazení databáze podle příjmení. Moduly to už je v podstatě programování, kdy definujeme složitější operace.

Tak toto jsou základní dovednosti evidenčního programu. Jelikož je Access poněkud propracovanější, nabízí i něco navíc. Výše uvedené operace jsou k dispozici standardnímu uživateli, například pracovnici zadávající údaje. Pokud je na našem pracovišti počítačová síť, je rozumné, aby tato data byla k dispozici vybraným uživatelům. Například seznam žáků by měl mít k dispozici každý uživatel. Přístup, zaznamenávat a měnit údaje o bydlišti a podobně však už jen třídní učitel a rovněž známky by měl mít možnost měnit pouze zkoušející (přednášející). Administrativní personál by do této evidence nemusel mít přístup vůbec.
Hovoříme o **zabezpečení databáze** (evidence) a **nastavení práv** jednotlivých uživatelů,
Nyní se přesuňme z příkladu naší školy do evidence obyvatelstva většího města. O rozsáhlou evidenci (databázi) se již musí někdo pořádně starat -nestačí ji pouze občas zazálohovat, či dokonce překopírovat na diskety. Správce databáze musí často kontrolovat a testovat integritu dat, musí provádět pravidelné zálohováni (každý týden) a další akce. I tyto činnosti, donedávna proveditelné pouze velkými databázovými systémy, dokáže Access zastávat.

DATABÁZOVÉ SERVERY -PŘEHLED PRODUKTŮ

MS SQL 2000
Microsoft SQL Server je výkonný databázový systém společnosti Microsoft. Databázi MS SQL 2000 mohou využívat výhradně virtuální servery běžící na platformě **Windows 2000**.
ORACLE 9i
Databáze Oracle patří k nejvýkonějším databázovým strojům, které jsou na trhu k dispozici. Dává vývojářům veškeré moderní prostředky pro práci s daty a pro pohodlnou správu databáze, nabízí pokročilé funkce dle standardu SQL (uložené procedury, triggery, uživatelské datové typy, transakční zpracování apod). Vysokou spolehlivost a maximální bezpečnost zaručuje certifikace dle normy ISO 15048. Realizace prostorových dat
DB2 IBM
Databáze DB2 je produkt firmy IBM.
MySQL
MySQL je rychlý a stabilní databázový systém. Nabízí bohatý a velmi užitečný soubor funkcí. Je podporován skripty PHP, CGI nebo ASP dle zvolené platformy virtuálního serveru. Databázi MySQL mohou využívat virtuální servery běžící na platformě **Linux/Unix i Windows 2000.**
PostgreSQL
PostgreSQL patří k robustním a na schopnosti bohatým databázím s rozsáhlými možnostmi rozšiřování uživatelskými typy a funkcemi, programování logiky databáze a transakčního zpracování. Databázi PostgreSQL mohou využívat virtuální servery běžící na platformě **Linux/Unix i Windows 2000.**

7. TEORIE RELAČNÍCH DATABÁZÍ

TERMINOLOGIE RELAČNÍCH DATABÁZÍ

Definice:

Entita je jednoznačně rozlišitelný, identifikovatelný a samostatné existence schopný objekt z reálného světa. Entitou je například osoba Karel Rychlý, osobní číslo D123-10254. Entitou může být kniha, řeka, stát, parcela, dům, měření fyzikálních veličin, výpůjčka atd.
Třída entit je skupina entit stejných vlastností. Jsou zaznamenány v jedné tabulce.???
Atribut je funkce přiřazující entitám nebo vztahům hodnoty popisného typu. Popisným typem rozumíme jednoduchý datový typ, například reálné číslo 278.1.
V atributech jsou obsaženy význačné vlastnosti entity či vztahu. Atribut může nabývat hodnot pouze určitého datového typu (z určitého oboru hodnot), např. celé číslo.
Entity jsou v relačním modelu ukládány do **tabulek**, které jsou nazývány dle autora relačního modelu E. F. Codda **relace**.
V záhlaví sloupců tabulek je definice **atributů**. Access jej nazývá pole.
Každý řádek představuje 1 **záznam** (record).

Do jedné tabulky ukládáme stejnou třídu entit.

tOsoba

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

Tab.1 :

Můžou nastat i takovéto dva případy. Jednou je popelnice atributem u entity dům a ve druhém případě je popelnice entitou a dům určený adresou a vlastníkem je atributem této entity. Rozhodnutí o tom, co a jak modelovat závisí na tvůrci databáze.

tDum

Dum	Ulice	CisloPopisne	Popelnice_kusu	Kontejner_kusu
269	Vídeňská	17	5	2

tPopelnice

CisloPopelnice	Ulice	CisloPopisne	Vlastnik
11	Vídeňská	17	Novák Jan
12	Vídeňská	17	Novák Jan

Doporučení a konvence:

Název pole (atributu) volíme co nejvýstižnější.

Doporučuje se, zvláště v návaznosti na případné programování či jiné operace s položkami, aby **název atributu** byl pokud možno co nejkratší. Někdy je vhodné držet se limitu 8 znaků vzhledem k převodu pod jiný databázový systém. Nejraději volíme názvy bez mezer, místo mezer použijeme podtržítka. Dále se nepoužívají znaky s diakritikou. Zakázány jsou znaky !, ., [,] , nesmí začínat mezerou, %,?.

Příklad : pro položku "datum narození" jsou varianty datum_narozeni, Dat_nar, DatNar, DatumN a podobně.

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

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

KLÍČE

V každé tabulce (relaci) musí být každý záznam jedinečný. Má-li být tato podmínka splněna, musí u každé relace existovat jeden nebo skupina atributů, která jednoznačně identifikuje jednotlivý záznam. Tento atribut nebo více atributů se nazývá ***kandidátní klíč***. Příklad u osob příjmení + jméno, (lépe příjmení + jméno+datum narození), rodné číslo, číslo občanského průkazu, evidenční číslo osoby. Dále ISBN u knihy, SPZ u osobního auta. Vhodnost kandidátních klíčů je třeba ale velice dobře zvážit. Číslo občanského průkazu se v průběhu života mění, rodné číslo nemají cizinci žijící u nás atd.

Tabulka může mít více kandidátních klíčů.

Pokud žádný z atributů není kandidátním klíčem zavedeme dodatečný atribut, který bude jednoznačně identifikovat záznam. Např. kód čtenáře, kód studenta apod. Můžeme u Accessu využít pro tento atribut datový typ automatické číslo. Access při tvorbě nové tabulky nabízí přímo vygenerování primárního klíče. Vloží nové pole s názvem ID, datového typu automatické číslo. Jedná se pak o vzestupnou posloupnost přirozených čísel od 1.

Jednoduchý klíč je klíč složený z jediného atributu.

Složený klíč je složený z více atributů. Délku složeného klíče musíme také pečlivě zvážit.

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

Primární klíč nesmí mít hodnotu NULL!

Hodnota NULL znamená, že atribut je neznámý nebo neexistující.

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

tDite

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

Cizí klíč Primární klíč

DATOVÉ TYPY

U každého atributu musíme určit jeho **datový typ** (**obor hodnot**, relační teorie databází hovoří o doméně viz. kapitola) a tím v podstatě naznačíme, které údaje budeme do něho zapisovat.
Jednotlivých varianty datových typů jsou v Accessu následující:

- **Text** – tento typ je nejobecnější a jeho obsahem může být libovolný řetězec znaků v maximální délce 255. To znamená, že konkrétní hodnota může obsahovat i mezery, speciální znaky, číslice, malá a velká písmena, interpunkční znaménka a podobně. Typickým představitelem je pole obsahující jméno a příjmení, adresu bydliště, ale i číslo telefonu, a pokud hodláme uvádět, i směrové číslo země či oblasti. Standardní délka je 50 znaků. Chceme-li ji změnit, pak na kartě Obecné vepíšeme odpovídající velikost do prvního políčka Velikost pole. Například pro pole Příjmení bude většinou postačovat délka 20 znaků (příjmení v naších podmínkách nebývá delší než 20 znaků).

- **Memo** - nejčastěji se tento typ používá pro zaevidování poznámek, tedy těch údajů, které se vyskytují spíše sporadicky (pouze u některých vět) a nelze je v podstatě zařadit do jiného pole, protože jsou velice různorodé. Například u evidence zaměstnanců by bylo možné pomocí tohoto typu pole vpisovat poznámky: Chodí často pozdě, Na schůzí musí sedět vedle Vlkové, Pozor - má zlého psa. Poznamenejme ještě, že textový řetězec může čítat maximálně 65 535 libovolných znaků. Tento datový typ se nevyskytuje běžně u jiných databázových systémů.

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

  ◊ **bajt** - do tohoto pole budeme moci zadat pouze celá čísla od 0 do 255

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

  ◊ **dlouhé celé číslo** - budeme moc zadat celé číslo, a to od -2 000 000 do 2 000 000

  ◊ **jednoduchá přesnost** - libovolné desetinné číslo v rozmezí od $-3 .10^{18}$ do $3 .10^{18}$

  ◊ **dvojitá přesnost** – číslo v rozmezí od - l,7.10^{308} do $1,7. 10^{308}$

- **Datum/čas** - tento typ je vhodný například pro sledování data narození, data pořízení vkladu do katastru nemovitostí, času odběru vzorku a podobně. Poznamenejme, že Access dokáže velice jednoduše u polí tohoto typu například přičíst k datu jeden týden, měsíc či rok, případně ze zadané ho data "vyjmout" pouze měsíc, a tudíž vyhledat z databáze všechny zaměstnance, kteří se narodili například v měsíci březnu (na roce nezáleží).
  Na kartě Obecné si ještě v prvním políčku Formát můžeme určit, v jak podobě budeme datum, resp. čas chtít zapisovat:

  datum (obecné) 19.6.1994 17:34:23
  datum (dlouhé) 19. června 1994
  datum (střední) 19-VI-94
  datum (krátké) 19.6. 1994
  čas (dlouhý) 17:34:23
  čas (střední) 5:34 odp.
  čas(krátký) 17:34

- **Měna** -tento typ použijeme v případě, kdy požadujeme, aby vkládaná čísla obsahovala znak naší měny (Kč). Je asi zřejmé, že není zrovna nejjednodušší, abychom u každého vkládaného čísla sami doplňovali měnu (navíc, pokud bychom měli položku definovanou jako číslo, pak by nám to program ani nezobrazil). Proto Access obsahuje tento typ, který sám automaticky za nás doplní zkratku měny. Na kartě Obecné můžeme ještě v poli Počet desetinných míst určit s kolika desetinnými místy se budou dané hodnoty vypisovat.

- **Automatické číslo** - toto je zvláštní typ, s jehož pomocí ponecháme vstup údajů plně na programu. Access nám totiž sám bude tuto položku u každé věty vyplňovat, a to buď vzestupnou posloupností přirozených čísel začne od jedničky nebo bude přiřazovat jednotlivým větám čísla náhodně. Tyto dvě varianty zvolíme na kartě Obecné v políčku Nové hodnoty.

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

- **Objekt OLE** - toto je nejobecnější typ položky, neboť může obsahovat cokoli, o čem se nám může zdát, a to do velikosti 1 MB. Můžeme sem pomocí technologie OLE vložit dokument z Wordu, Excelu, další tabulku Accessu, obrázek, zvuk. Tak například evidence zaměstnanců může díky tomuto typu obsahovat naskenované fotky.

Můžou se zde uložit naskenové podpisové vzory k účtu u bankovního ústavu. Seznam našich dodavatelů můžeme zpestřit jejich logy, evidenci CD disků s hudbou pak krátkými, pokud možno výstižnými úryvky.

- **Popis** - toto políčko je nepovinné a lze jej použít pro objasnění a doplnění názvu pole. Políčko Popis se vypisuje jako nápověda na posledním řádku obrazovky (okna) Accessu.

Databázové systémy se liší počtem a druhy datových typů. Všechny systémy zcela určitě podporují datové typy **textový** (text, charakter, string), **číselný** (numeric), **datumový** (date) a **logický** (logical, boolean, YesNo, ano/ne). Různé systémy pak podporují různé další typy, které však už nemusí být obecně přenositelné ze systému do systému.
Pro prostorová data se nejprve využívalo uložení do datového typu **BLOB** (binary large object).

INDEXY

Velmi názorně si rozdíl mezi tabulkou s indexy a bez indexů můžeme vysvětlit na příkladu předchůdců moderních databázových platforem - klasické papírové databance. Představme si kartotéku se zdravotními záznamy pacientů u obvodního lékaře. Při návštěvě lékaře jsme v minulosti viděli v činnosti systém řízení takovéto databáze - zdravotní sestru. Zdravotní záznamy mohou být uložené v kartotéce náhodně nebo mohou být uspořádány podle vhodného primárního klíče.Náhodné uspořádání - složky se zdravotními záznamy jsou poskládané v kartotéce náhodně, bez jakéhokoli systému. Abychom našli hledanou složku, musíme začít obsah kartotéky prohledávat od první složky až do místa, kde se nachází hledaná složka. V průměru tedy vždy projdeme přibližně polovinu složek.Naproti tomu vkládání nových složek je velmi rychlé - jednoduše vložíme složku, kamkoli nás napadne.
Uspořádání podle primárního klíče - složky jsou v kartotéce uspořádány podle určitého pravidla, například v abecedním pořadí podle příjmení pacienta. Vyhledání je podstatně rychlejší (ke zrychlení povede i používání zarážek s jednotlivými písmeny). Vkládání nových složek je ale o něco složitější a pomalejší, protože musíme novou složku umístit přesně na pozici, která jí podle abecedního pořadí náleží. Zdálo by se, že na tomto systému již není co vylepšovat. Není to ale pravda - složky totiž nejsou uložené v jednom nekonečně dlouhém regálu, ale ve více skříních celé kartotéky. I přes pravidelnou údržbu databáze (zdravotní sestra pravidelně přesouvá jednotlivé složky mezi regály) může dojít k situaci, kdy může vložení nového záznamu trvat o mnoho déle. Předpokládejme, že typická tloušťka záložky jednoho pacienta jsou tři centimetry a že "databáze" je dobře udržovaná, což znamená, že v jednotlivých regálech je dostatek místa pro vložení několika nových záznamů pro každé písmeno. Jenže co když k lékaři přijde hypochondr, takový pan Náhlovský. Podle vyprávění pana Náhlovského by tloušťka jeho složky musela být alespoň jeden metr. Takto tlustý záznam se nám do police mezi ostatní záznamy s písmenem N již nevejde, proto musíme popřesouvat záznamy v ostatních regálech a vytvořit potřebné místo.
Problémy ale mohou být také jiného druhu - pokud potřebujeme zvát pacienty na pravidelné preventivní prohlídky, musíme procházet celou kartotéku a ve všech složkách ověřit datum poslední prohlídky.

ZAVEDENÍ INDEXŮ

Naznačené problémy vyřeší přidání další malé kartotéky, která bude obsahovat lístky (zjednodušeně řešeno index) se základními údaji. Nyní již nemusíme vkládat nové složky na přesně určené místo. Abychom si případ s indexovanou kartotékou objasnili, popíšeme si ji co nejpodrobněji. Zavedeme si primární klíč, tedy jedinečné číslo, které napíšeme na viditelné místo na každé složce. Toto číslo budeme při vkládání nové složky automaticky zvětšovat o jedničku.
Mějme prázdnou databázi, v našem případě kartotéku lékaře, který si právě zařídil praxi; má prázdné kartotékové regály, krabici na minikartotéku s indexovými lístky a zdravotní sestru, která bude s kartotékou pracovat (samozřejmě hezkou a šikovnou). Nyní již čekáme jen na to, až se přijde zaregistrovat první pacient. Když tato chvíle nastane a jako první přijde například pan Tomeček, uložíme jeho záznam (složku) na začátek regálu, přidělíme mu číslo (primární klíč) 1 a napíšeme toto číslo na složku. Potom vyplníme indexový lístek, který bude obsahovat primární klíč, jméno a datum poslední prohlídky. Lístek uložíme do minikartotéky pod písmeno T. Postupně se přichází zaregistrovat další pacienti a regály se úspěšně zaplňují. Pokud nyní přijde pan Náhlovský, uložíme jeho záznam jednoduše na začátek dalšího regálu. Přidělíme mu pořadové číslo a vyplníme indexový lístek, který vložíme do minikartotéky pod písmeno N.
Zavedením primárního klíče a indexové minikartotéky jsme si vyhledávání patřičně usnadnili a ani zařazení nové složky není nijak složitou záležitostí. Pokud k nám přijde pacient, vyhledáme v minikartotéce jeho indexový lístek a zjistíme odpovídající primární klíč. Primární klíč nám v tomto případě přesně určuje polohu složky v regálu.
Problém zvaní pacientů na preventivní prohlídky jsme si ovšem až tak neusnadnili stačí však mít dvě minikartotéky tedy dva indexy. Tato druhá minikartotéka bude obsahovat indexové lístky seřazené podle data poslední preventivní prohlídky. Potom při zvaní pacientů k prohlídkám stačí procházet tuto minikartotéku odzadu a po absolvování této prohlídky zařadit příslušný indexový lístek na začátek minikartotéky .
Podobně můžeme zavést také třetí minikartotéku indexových lístků, například podle věku či váhy. Potom nebude žádný problém nabízet starším občanům možnost očkování proti chřipce. Při existenci více indexů nám vkládání zabere o něco více času. V našem případě nyní musí sestra vyplnit a zařadit tři indexové lístky pro každou nově vkládanou složku. Minikartotéka také pochopitelně zabere nějaké to fyzické místo.
Zavedením indexů ovšem výrazně zkrátíme přístupový čas k požadovaným datům, přičemž podstatně redukujeme množství diskových operací. Moderní databázové platformy používají pro indexy nejčastěji stromové struktury , například tzv. B-stromy. Základní princip vyhledávání v jednoduché stromové struktuře si ukážeme na jednoduchém příkladu hádání čísel.

Pokud máme uhodnout číslo v rozsahu 1 až 16, potřebujeme na to maximálně 4 pokusy. Pokud bychom hádali číslo v rozsahu 1 až 1 024, potřebovali bychom pouze 10 pokusů (210 = 1 024).
Mnohem dokonalejší vyhledávání nabízí tzv. B++ stromy.
Příklad databáze lékařských záznamů je výborný i pro pochopení relací (vztahů) mezi více tabulkami. Důležitým údajem je totiž také informace o zdravotní pojišťovně, u které je konkrétní pacient pojištěný, především pak její číslo a adresa. Tyto údaje mohou být uloženy přímo ve složce pacienta, není to ale příliš praktické. Problém může nastat například tehdy, když některá z pojišťoven změní svoji adresu. Řešení je poměrně jednoduché - použijeme dvě tabulky. V první budou údaje o zdravotních pojišťovnách. Každá pojišťovna má své identifikační číslo, které je jedinečné. Toto číslo bude primárním klíčem tabulky pojišťoven. V tabulce zdravotních záznamů uvedeme u každého pacienta v příslušném sloupci jen číslo pojišťovny.

POUŽÍVÁNÍ INDEXŮ

Na první pohled to vypadá tak, že čím více použijeme indexů, tím dosáhneme lepšího výkonu při práci s databází. Bohužel toto tvrzení není vždy pravdivé. Indexy bychom měli používat velmi rozvážně a zakládat je pouze v případě, kdy je to opravdu výhodné. Indexy totiž mají také své nevýhody, v některých případech mohou podstatně zpomalit databázovou aplikaci a výrazně zvýšit nároky na diskovou kapacitu. Ke zpomalení může dojít také z toho důvodu, že tzv. optimalizátor dotazu vyhodnotí nevhodný index jako nejlepší -jinými slovy s trochou nadsázky bude přebírat tak dlouho, až přebere. Stejně tak nedoporučujeme používat indexy nad atributy s nízkou variabilitou, například pokud máme ve sloupci pouze dvě možné hodnoty. Index nám v tomto případě příliš nepomůže, protože bude vyhledávací mechanismus směřovat na začátek tabulky nebo někam doprostřed (počátek dat odpovídající druhé hodnotě indexu závisí na poměru dat pro jednotlivé hodnoty indexu).

8. STUPNĚ RELACÍ V RELAČNÍCH DATABÁZOVÝCH SYSTÉMECH

(stupně relací, **modelování databáze - nástroje**)

VZTAHY

Vztah je vazba mezi dvěma nebo více entitami. Například entita uživatel Karel Rychlý, osobní číslo D123-10254 může být ve vztahu má-půjčenu k entitě kniha katalogové č. A32198.
Vazba mezi dvěma nebo více entitami je vyjádřena pomocí vztahu (relace). V reálném světě registrujeme velké množství vztahů, které musí být v datovém modelu nějakým způsobem vyjádřitelné. Poměr prvků ve vztahu může být 1:1, l:N anebo M:N. Mluvíme o stupni, kardinalitě vztahu.

VZTAH 1:1

Typickým příkladem je vztah 1 manžel : 1 manželka (aspoň v našich krajích). Dalším příkladem je název kraje : název krajského města, 1 polygon : 1 parcela, atd.
Je třeba zvážit zda zakládat pro takovéto entity samostatné tabulky. Lze je velice jednoduše sloučit do jedné tabulky. Ale může existovat i důvod proč je chceme mít v samostatných tabulkách.

VZTAH 1:N

Vztahový typ 1:N (nebo též 1 : nekonečnu) může teoreticky zahrnovat i korespondence,1:0, 0:1 a 1:1, které můžou být z modelu vyloučeny vhodně zvolenými pravidly.
Příklady jsou jedna kniha může být napsána více autory, jedna budova má více pater, do jednoho úmoří spadá více řek, město má více městských částí.

VZTAH M:N

Vztah M:N je nejsložitější vztah. Je to totéž jako m-krát opakovaný vztah 1:N.
Příkladem může být vztah 1 parcela má více vlastníků a naopak jeden vlastník může vlastnit více různých parcel. Dále například jeden stát může mít více úředních jazyků a naopak úřední jazyky v některých státech mohou být shodné (angličtina, francouzština, němčina).
Vztah M:N je v datovém modelu nutné **dekomponovat** (převést) na dva vztahy 1:N.
Ukažme si to na příkladě.V ukázkovém příkladě máme entitu *stát* a entitu *úřední jazyk*.
tStat

Stat
Belgie
Lucembursko
Francie

tJazyk

UredJazyk
Francouzština
Nizozemština
Němčina
Lucemburština

Musíme provést dekompozici vztahu stát - úřední jazyk takto:
Přidáme další – vazební tabulku tVazba. Vzniknou tak tři tabulky:

tStat

ID_stat	Stat
101	Belgie
102	Lucembursko
103	Francie

tVazba

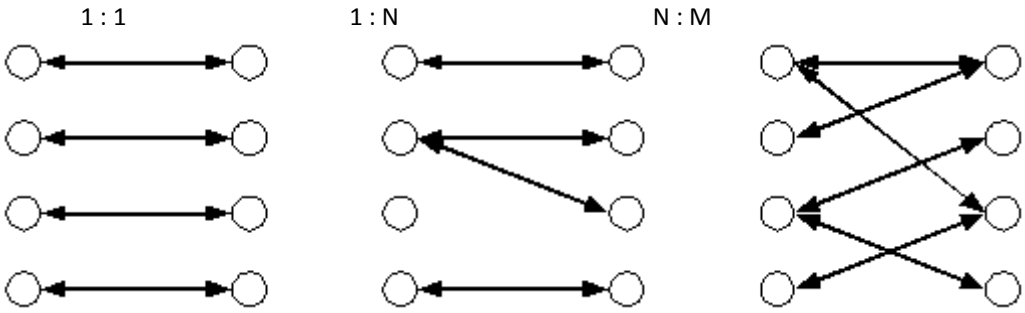
ID_stat	ID_jazyk
101	1
101	2
101	3
102	1
102	3
102	4
103	1

tJazyk

ID_jazyk	UredJazyk
1	francouština
2	nizozemština
3	němčina
4	lucemburština

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

Graficky můžeme vztahy znázornit takto:

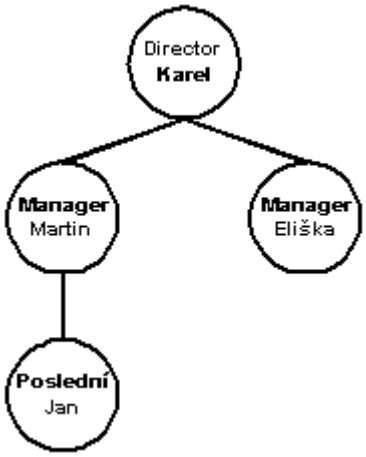


REKURZIVNÍ TYP VZTAHU

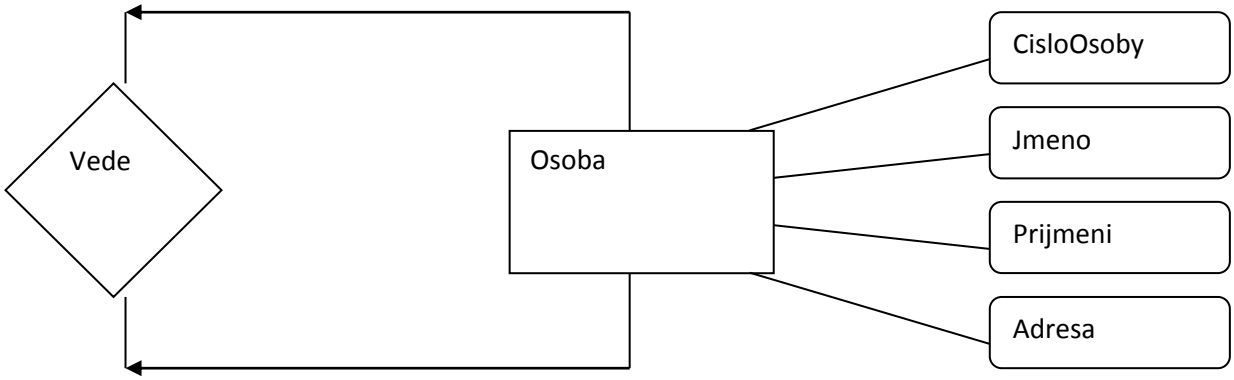
Velmi častou je potřeba vyjádření typu vztahu, jehož účastníci jsou entity stejného typu.
Například:
Jedna OSOBA může vést několik OSOB (je vedoucím pracovníkem) a zároveň jedna OSOBA musí být vedena právě jednou OSOBOU.
Nebo VÝROBEK může být součástí VÝROBKŮ a zároveň VÝROBEK může být sestaven z daných VÝROBKŮ.
tOsoba

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

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



Konceptuální model může takovouto situaci zobrazovat např. tak, jak je ukázáno na následujícím obrázku.



Zejména v souvislostí s rekurzivním typem vztahu je užitečné zavést do E-R modelu ještě jeden konstrukt. Říká se mu *role.* Jmény rolí se ohodnocují jednotlivé hrany E-R schématu, které vedou od etitních typů k typům vztahů. Označují roli, jakou účastnické entitní typy hrají v daném typu vztahu. V našem personálním příkladu bude horní hrana ohodnocena rolí *Vede* a dolní hrana rolí *Je_vedena*.

9. INTEGRITA DAT

INTEGRITA

Samozřejmým požadavkem na databázi je, aby obsahovala správná a aktuální data pro daný okamžik.
Hovoříme o integritě databáze. Správnost údajů můžeme zajistit několika způsoby na různých místech.

ENTITOVÁ INTEGRITA

Entitovou integritu zajišťujeme požadavkem existence primárního klíče.

ATRIBUTOVOU INTEGRITU

zajišťujeme již na počátku vhodnou *definicí oboru hodnot*. Tu dále můžeme zpřísnit. Například v evidenci studentů čtyřletého studia nedovolíme zadat jiné číslo ročníku (které je z oboru celých čísel) než v rozmezí 1 až 4. Kromě zúžení jednoho určitého atributu může entitové omezení zasahovat také do několika různých atributů. Příkladem takového omezení je například požadavek, podle něhož musí být DatumOdeslani požadovaného výrobku pozdější nebo rovno jako DatumObjednavky tohoto výrobku.
Zadáváme například adresu a to název ulice. U názvu ulice „Fr. Ondříčka“ může dojít často k chybnému zápisu, např. Františka Ondříčka, F. Ondříčka, Ondříčkova vlivem uživatele. Z pohledu databáze se jedná vždy o jinou ulici. Při zjišťování počtu lidí žijících v ulici Fr.Ondříčka dostaneme chybný údaj, neboť ti, kteří mají zapsánu adresu v jiném tvaru nejsou započítáni. Vstup správných údajů zajistíme tak, že uživateli nabídneme pouze exitující možné názvy ulic pro dané město. Určíme tedy obor hodnot.
Stejně tak u evidence osob v atributu rodinný stav připadají v úvahu pouze 4 možnosti: svobodný/á, ženatý/vdaná, vdovec/vdova, rozvedený/rozvedená.

Obor hodnot je tedy dán výčtem možností. Tyto možnosti jsou realizovány samostatnou tabulkou, někdy označovanou jako číselníky. Můžeme uvést například číselníky Českého statistického úřadu pro kódování krajů, sčítacích okrsků atd.
Při vstupu, nebo editaci údajů je dobré též kontrolovat správnost atributů eventuelně zamezit vstupu chybných údajů. Je vhodná i možnost automatického převodu vstupních údajů. Například u příjmení a jmen automaticky převádět první písmeno na velké nebo celé příjmení na velká písmena. V MS Accessu k tomu slouží nastavování tzv. vstupní masky. Příklady a význam masek pro MS Access naleznete v nápovědě nebo v příloze č?
V jiných složitějších případech je nutné ošetřit vstup přímo programovým kódem. Tyto „spouště" jsou nazývány v programových produktech *trigger*.

VSTUPNÍ MASKY

Vstupní maska se používá v polích (v tabulkách a dotazech) a v textových polích a polích se seznamem (ve formulářích) k formátování a předběžné kontrole vkládaných dat. Vstupní maska se skládá ze znakových literálů (mezery, tečky, čárky a uvozovky), které vymezují jednotlivé oddíly pro vkládaná data. Nastavení vlastnosti **Vstupní maska** se skládá z literálních a speciálních znaků určujících druh hodnot, které lze vložit do daného oddílu. Vstupní masky se používají především pro pole typu Text a Datum/Čas, ale lze je použít i pro pole typu Číslo a Měna.

REFERENČNÍ INTEGRITA A JEJÍ DŮSLEDKY

Referenční integritu lze nastavit pro tabulky ve vzájemném vztahu 1 : N (lze ji nastavit též pro vztah 1 : 1), který možno chápat jako zvláštní případ relace 1 :N . Zajišťuje, že se při práci se záznamy (vkládání a odstraňování záznamů), aby se zachovávaly definované vztahy mezi tabulkami.
Každému záznamu v tabulce na straně N s hodnotou cizího klíče <> NULL musí odpovídat právě jeden záznam v tabulce na straně 1. Jinými slovy, v tabulce na straně N nepřipouštíme osiřelé záznamy.
Vynutíme-li referenční integritu, program Microsoft Access zabrání

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

Předělat na vlastníky parcel.
Nadefinujeme-li referenční integritu ve vztahu mezi tabulkami tOsoba a tDite, nemůžeme potom do tabulky tDite přidat záznam dítěte, které nemá rodiče v tabulce tOsoba. To je přínos referenční integrity, neboť takový záznam by v tabulce tDite (děti našich zaměstnanců) opravdu neměl být.
Představme si však tuto situaci. V tabulce tOsoba je primárnf klíč rodné číslo. Přijde pan Novák, který má dvě děti a ohlásí nám, že jeho rodné číslo bylo nesprávné a přinese platné rodné číslo. Access jej však nedovolí změnit, neboť v tabulce tDite by se ocitli dva sirotci. Děti pana Nováka, které mají ve sloupečku Rodič ještě původně nesprávné rodné číslo svého otce. Access se samozřejmě nedá přemluvit; že tato situace je jen chvilková, neboť máme vzápětí v úmyslu tato dvě nesprávná.rodná čísla v tabulce tDite též opravit. Abychom se však nedostali do popsané patové situace, dovoluje Access po nadefinování referenční integrity ještě aktivovat možnost *Aktualizace souvisejících polí v kaskádě*.
V našem případě nám potom nejenže nechá opravit nesprávné rodné číslo, ale současně je ještě sám opraví u záznamů tabulky na straně N souvisejících se záznamem v tabulce první (tj. u dětí pana Nováka, kde je uvedeno jako cizí klíč).
Referenční integrita nám též neumožní odstranit z tabulky tOsoba záznam zaměstnance, který má v tabulce tDite odpovídající záznamy o jeho dětech. Z těchto záznamů by se totiž staly záznamy osiřelé, což by narušilo samu podstatu referenční integrity. Proto je po nadefinování referenční integrity ještě dovoleno aktivovat volbu Odstranění souvisejících polí v kaskádě. Potom nám Access nejen dovolí odstranit z tabulky tOsoba záznam zaměstnance, ale navíc ještě sám odstraní z tabulky tDite jeho děti, čímž zde nedojde ke vzniku osiřelých záznamů.

PŘECHODOVÁ INTEGRITA

Omezení přechodové integrity definují stavy, kterými může vektor hodnot právoplatně přecházet.
Můžeme si sestavit názorný diagram stavů a přechodů. Např. pro rodinný stav. Osoba musí přejít ze stavu svobodný pouze do stavu ženatý. Ze stavu ženatý do stavu rozvedený nebo do stavu vdovec. Nelze přejít přímo ze stavu svobodný do stavu rozvedený. Přechodová omezení mohou být definována přes několik atributů a dokonce přes několik tabulek.

TRANSAKCE

Transakce nám zajišťují kontinuitu a nedělitelnost změn na základě principu "Všechno, nebo nic!". To znamená, že není možné například provést srážku ze mzdy a nezaslat ji na účet u zdravotní pojišťovny. Při pozornějším přečtení předchozí věty se možná mnozí podnikatelé pousmějí. Vědí, že něco takového možné je, ale za to pak nemůže databázový server, ale lidský faktor. Proto bude asi lepší klasický příklad s převodem finanční částky z jednoho účtu na druhý. Pokud klient A převádí finanční částku na účet klienta B, musí se daná částka nejprve odepsat z účtu klienta A a následně připsat na účet klienta B. Je to triviální operace, ale jen za předpokladu, že vše proběhlo bez problémů. Představme si situaci, že klient B svůj účet již zrušil nebo klient A se spletl při zadávání čísla účtu klienta B. Při těchto (a dalších) problémech se částka z účtu klienta A sice

odepíše, nicméně není ji možné připsat na účet klienta B. Naštěstí se nic neděje, protože SŘBD takovouto transakci stornuje a peníze se bezpečně vrátí na účet klienta A.
Podobně jsou ošetřeny i chybové stavy, kdy dojde k technickým výpadkům. Takováto transakce tedy proběhne buď jako celek, což znamená, že peníze se z účtu odesilatele odepíší a na účet příjemce připíší, nebo se transakce jako celek zruší.
Průběh transakce řídíme SQL příkazy SAVE POINT, COMMIT a ROLLBACK. Příkazem COMMIT potvrzujeme platnost transakce, příkazem SAVE POINT název označujeme místo pro případný návrat a příkaz ROLLBACK [TO SAVEPOINT název] zruší všechny provedené změny, které byly v rámci aktivní transakce doposud provedeny.
Opatrný manažer firmy by případné dodatečné zdražení všech položek měl provést pro jistotu takto:
SAVE POINT pro_jistotu;
UPDATE ucastnici SET suma = suma * 1.5;
Jiný příklad, kde se musí ošetřit transakce je případ vložení majitele do katastru nemovitostí a jeho vlastnického vztahu. Vložím majitele do jedné tabulky a další údaje do tabulky vlastnictví. Když jedna z těchto operací ztroskotá,musíme odvolat i tu která se provedla. Zůstala by pak pouze třeba zaznamenaná nemovitost, ale údaj o existenci vlastníka by chyběl.

REPLIKACE DAT

Při replikaci dat se nachází kopie množiny objektů v každém uzlu, ve kterém je využívána. V systému tedy existuje několik kopií každého objektu. Výhodou tohoto řešení je kvalitní dostupnost, rychlý přístup ke každému objektu a menší nároky na komunikaci mezi uzly. Nevýhodou je problém duplicity, díky níž je nutné trvale zajišťovat konzistenci všech kopií.

Zajišťování konzistencem je klíčovým problémem replikace dat. Je nutné implementovat systém, který určí správné pořadí provedených operací a při replikaci rozdistribuje správnou kopii. Také je nutné zabránit současné modifikaci dvou kopií objektu. Správné pořadí provedených operací je určováno většinou časovými razítky, současné modifikaci dvou kopií jednoho objektu mohou zabránit klasické paralelní synchronizační prostředky (zámky, semafory, apod.). U transakčních systémů je možné razítky označovat jednotlivé transakce, výhodou je jednodušší a spolehlivější zajištění konzistence, nevýhodou pak nutnost archivovat všechny transakce až do doby další synchronizace. Jinak je možné označovat celé objekty, což má ale za následek zvýšení rizika poškození konzistence dat. Další variantou může být označování větších resp. menších celků, tj. např. skupiny objektů resp. jednotlivé atributy. V některých případech může jít o vhodnou variantu, to je však nutné pečlivě zvážit.

Doposud jsme předpokládali asynchroní replikaci, tj. že jsou operace pro zajištění konzistence prováděny nezávisle na modifikaci replikovaných objektů. Spolehlivější může být provádění operací pro zajištění konzistence bezprostředně před modifikací objetu nebo po ní. V prvním případě je před modifikací objektu nalezena jeho aktuální kopie, která je použita během modifikace. Tato metoda je spolehlivější než ostatní, problémem je však hledání aktuální kopie. V druhém případě je po každé modifikaci objekt zkopírován do všech uzlů, tj. každý uzel má vždy aktuální kopii každého objektu. Tato varianta je vhodná pro případ, kdy jsou objekty mnohem častěji pouze čteny, než modifikovány a kdy je nutné zajistit aktuálnost kopie objektu při každém čtení. Obecně však hrozí riziko přetížení komunikační sítě příliš častou výměnou kopií objektů.

Ani v těchto případech se však nezbavíme nutnosti vyloučit současnou změnu dvou kopií replikovaného objektu. To muže být zejména v geograficky rozsáhlejších systémech problém, který je nutné řešit dalšími prostředky. Mezi ně patří např. systémy přístupových práv, priorit či distribuce objektů pouze pro čtení.

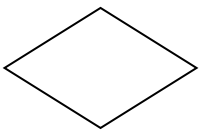
10. DATABÁZOVÉ MODELY

E-R DIAGRAMY

E-R konceptuální model (entita - relationship)
E-R konceptuální model (E/R či E-R model) je jedním z mnoha modelů, které jsou pro konceptuální modelování k dispozici. E-R model se vyskytuje v mnoha variantách. E-R modely jsou založeny na pojmu entita (Entity), vztah (Relationship). K těmto dvěma základním konstrukcím často bývá přiřazován ještě třetí - atribut (Attribute). Model navrhl a poprvé popsal v roce 1976 Peter Pin-Shan Chen.
Pro reprezentaci grafického návrhu E-R modelu se používají tyto symboly:


entita


atribut

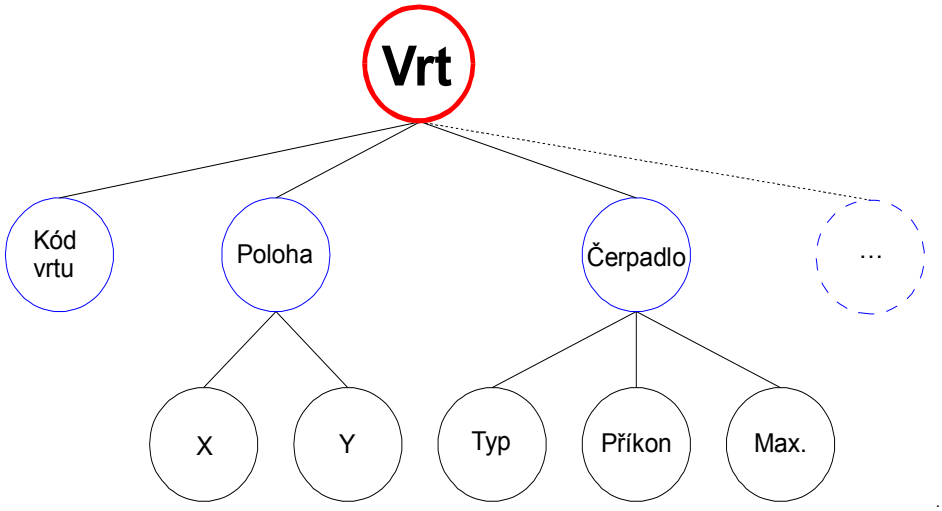

vztah

LOGICKÁ ORGANIZACE BÁZE DAT

Historický vývoj směřoval od prvotního zpracování dat na úrovni fyzických záznamů přes datové záznamy bez vzájemných vazeb (na úrovni dat) až po báze dat obsahující v sobě popis dat včetně vazeb.
V současné době má smysl rozebírat dva dominantní modely logické organizace bází dat: hierarchický (také hierarchicko - síťový) model a relační model.

HIERARCHICKO - SÍŤOVÝ MODEL

Model vychází z použité hierarchické struktury dat tak, jak byla kdysi zavedena pro potřeby jazyka Cobol pro zobrazení hodnot dat a jejich vzájemných vztahů (nadřízenosti a podřízenosti). Tento model se neopírá o matematickou teorii, i když přejímá část terminologie z *teorie grafů*. Přesto nalezl v praxi široké uplatnění.
Na hierarchickém modelu je založena řada systémů řízení báze dat.
Hierarchická struktura je taková, kde záznamy jsou v hierarchickém vztahu nadřazenosti a podřízenosti. Přitom se používá "rodinná" terminologie **rodič** a **potomek** ve zřejmém významu.
V hierarchické struktuře má každý potomek jediného rodiče, existuje jediný rodič, který není potomkem a potomek v jednom vztahu může být rodičem v jiném vztahu.
Pokud je zapotřebí popsat, na kterém místě hierarchické struktury se nějaký záznam nalézá, používá se k tomu tzv. přístupová cesta. To je možno díky popsaným vlastnostem hierarchické struktury, které zaručují, že od kořene lze dojít k danému záznamu jediným způsobem.
Často se vyžaduje (většinou z ryze praktických důvodů např. sběru dat), aby v každém záznamu existoval klíč. V takovém případě lze přístupovou cestu popsat jednoduše jako posloupnost klíčů počínaje klíčem kořenového záznamu přes klíče všech nadřízených až po klíč daného záznamu včetně.
Zmíněné pojmy z teorie grafů se při popisu hierarchických struktur využívají v tomto smyslu:
záznam = uzel grafu
vztah rodič - potomek = hrana grafu
rodič a potomek = incidenční uzly hrany
hierarchická struktura = souvislý graf, který je stromem
báze dat hierarchického modelu = graf, který je les (tj. množina disjunktních stromů)
přístupová cesta k záznamu = cesta v grafu od kořene k danému uzlu.



Obr.: Stromová struktura

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

RELAČNÍ MODEL

Jedním z nejjednodušších zápisů dat je zápis do klasické tabulky. Takto převážně vznikají zápisy dat v terénu, obecně v neautomatizovaných částech systému. Charakteristický pro tento zápis je její členění do sloupců, z nich každý má nadpis. To je velmi podstatné, protože nadpis ve smyslu identifikace údajů automaticky indukuje také typ údajů v daném sloupci. Sloupce jsou tedy "homogenní co do typu".

Pevným počtem n sloupců tvoří data v řádcích uspořádané n-tice. Každý prvek n-tice, nazývaný pole, je toho typu, jaký odpovídá typu sloupce. Je tedy prvkem (konečné nebo nekonečné) jednoznačně určené množiny Di (např. množiny všech datumů, množiny všech racionálních čísel, množiny {Ano;Ne} apod).
Relační datový model je spojen se jménem Edgara F. **Codda**. Teoretický popis modelu navrhl v roce 1970.

Matematický základ relačního modelu

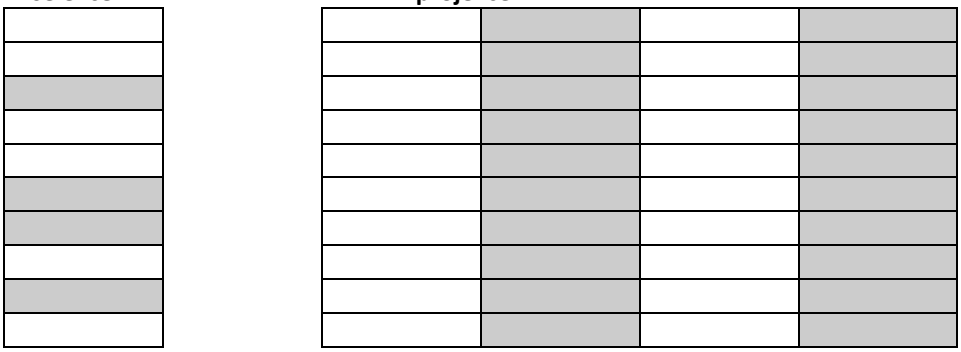
Množina všech uspořádaných n-tic $\mathbf{<a_1, a_2, ... , a_n>}$, kde A_i jsou libovolné množiny, a_i je z A_i a n je přirozené číslo, je z teorie množin známa jako kartézský součin $\mathbf{K = A_1 \times A_2 \times ... \times A_n}$ a každá podmnožina R z K je známa jako n-ární relace v K. Je tedy možno pohlížet na každou tabulku, která je vytvořena obdobně jako tabulka shora, jako na n-ární relaci.
Pro určení relace R pro potřeby modelu báze dat je zapotřebí zadat konečnou množinu atributů F - což jsou jména polí s případnými dalšími specifikacemi (šířka sloupce, počet desetinných míst apod.) využitelných pro uživatelskou i programovou identifikaci domény A_i, tj. množiny možných hodnot každého pole podmnožinu kartézského součinu domén, tj. vlastní relaci (z hlediska tabulky je tím určen počet polí a pořadí sloupců).
Relaci je tedy možno definovat jako trojici **R = <F, D, T>,** kde
F je konečná množina jmen atributů
D je zobrazení, přiřazující každému f z F doménu D(f) atributu f. Domény jsou libovolné neprázdné množiny (konečné nebo nekonečné); je-li f, g z F, f různé od g, nemusí být D(f) různé od D(g).
T je konečná podmnožina kartézského součinu X [D(f)] všech domén atributů, f je z F.
Tabulky dat, které reprezentují relace, mají - jak již bylo uvedeno - následující vlastnosti:

- každému prvku relace odpovídá jeden řádek tabulky
- žádné dva řádky nejsou identické
- sloupec s atributem f z F v záhlaví obsahuje jen hodnoty z domény D(f).

Okolnost, že v praxi často nebývá splněna vlastnost ad 2), se řeší "očíslováním řádků"; přidá se jeden sloupec, jehož doména je podmnožina přirozených čísel.

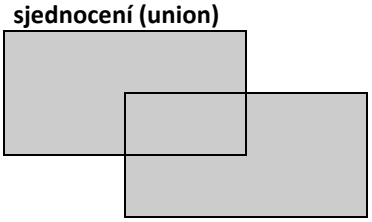
Relační algebra
Relační algebra je množina operací na relačním modelu. Skládá se z operátorů, jednotlivé relace zde vystupují jako operandy operace. Výsledkem množinové operace je opět vždy relace.
Operace dělíme na unární a n-ární.
Unární operace znamená, že do dané operace vstupuje pouze jedna relace.
Unární operace jsou :

selekce **projekce**

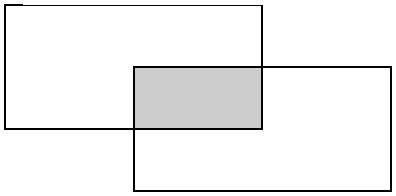


Selekce vybírá pouze některé relace, které vyhovují zadané podmínce. Projekce vybírá pouze některé atributy ze vstupní relace. Obě operace lze uplatnit i zároveň na vstupní relaci.
N-ární operace (vstupuje do nich *n* relací). Nejčastěji jsou to dvě relace, takže operaci můžeme označit jako *binární.* Jsou to známé množinové operace:

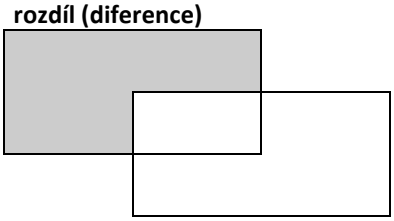
sjednocení (union)



průnik (intersection)



rozdíl (diference)



Čtvrtou operací je **kartézský součin (produkt)** vrací relaci sestávající z kombinací všech n-tic obou vstupních relací.

Terminologie relačních databází označuje tyto operace jako **přirozené spojení** (natural join). Přirozené spojení kombinuje dohromady n-tice vstupních relací, které mají stejné hodnoty ve společných sloupcích. Tato hodnota se objeví ve výsledku pouze jednou.

Druhy spojení jsou následující:

- **vnitřní** - inner join (odpovídá průniku)
- **vnější** - outer join

Vnější spojení se dělí dále na:
1.levé - left outer join (odpovídá rozdílu)
2.pravé - right outer join (odpovídá rozdílu)
3.plné - full join (odpovídá sjednocení)

Posledním druhem spojení je **křížové** spojení- cross join, to odpovídá kartézskému součinu.

Na dvou relacích Relace X a RelaceY si ukážeme výsledky jednotlivých operací.

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Levé vnější spojení X a Y

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

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Pravé vnější spojení X a Y

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

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Plné spojení X a Y

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

Relace X

AtributA	AtributB
A1	B1
A2	B1
A3	B2
A4	B4

Relace Y

AtributB	AtributC
B1	C22
B2	C23
B3	C24

Vnitřní spojení X a Y

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

Operace **přejemenování** atributů – neprovádí nic s daty, ale pracuje pouze se seznamem atributů.

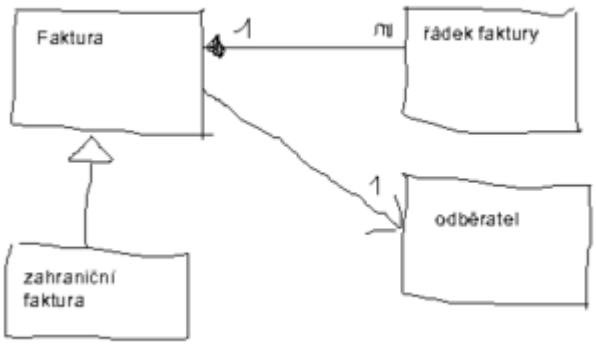
UML, **Unified Modeling Language** je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. UML nabízí standardní způsob zápisu jak návrhů systému včetně konceptuálních prvků jako jsou business procesy a systémové funkce, tak konkrétních prvků jako jsou příkazy programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty.

UML podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů. UML neobsahuje způsob, jak se má používat, ani neobsahuje metodiku(y), jak analyzovat, specifikovat či navrhovat programové systémy.

Standard UML definuje standardizační skupina Object Management Group (OMG).

Způsoby použití UML

Kreslení konceptu



Ukázka UML

Při tomto použití je UML podpůrným nástrojem pro komunikaci mezi vývojáři a pro zaznamenání myšlenek a návrhů.
Do diagramů se kreslí pouze věci podstatné pro grafické vyjádření návrhu, části návrhu před tím, než se začne programovat.

Důležitá je srozumitelnost, rychlost nakreslení a snadnost změny či navržení alternativ řešení.

Kreslení detailních návrhů

Cílem je zaznamenat kompletní návrh či kompletní realizaci. Při kreslení návrhu by měl analytik obsáhnout všechny prvky tak, aby programátor byl schopen vytvořit program bez velkého přemýšlení nad věcnou oblastí (pro programátora by neměla vzniknout potřeba konzultace s uživatelem). Při kreslení detailních návrhů se obvykle používají specializované programy (CASE), které jsou schopny sdílet informace mezi jednotlivými modely a kontrolovat konzistenci návrhu. Při dokumentaci programu se často používání nástroje pro generování diagramů z vlastního kódu aplikace.

UML jako programovací jazyk

Při tomto použití vývojář nakreslí UML diagramy, ze kterých se vygeneruje přímo spustitelný kód. Toto vyžaduje specializované nástroje a velmi přesné vyjadřování v UML diagramech. V této souvislosti se velmi často používá pojem Model Driven Architecture (MDA), což je další standard skupiny OMG, který se snaží standardizovat použití UML jako programovacího jazyka.

Metamodel

Tento pohled používají autoři UML a autoři CASE nástrojů - nedívají se na UML jako na diagramy, pro ně je základem UML metamodel (diagramy jsou pouze grafickou reprezentací metamodelu). Při tomto přístupu se často používá pojem model místo pojmu diagram, např. místo diagramu tříd se používá pojem model tříd. Metamodel se popisuje pomocí Meta-Object-Facility (MOF) - abstraktního jazyka pro specifikaci, vytváření a správu metamodelů (další standard OMG). Pro výměnu metamodelů se používá XMI - na XML založený standard (součást standardu UML).

11.NORMALIZACE DATABÁZÍ

NORMALIZACE A NORMÁLNÍ FORMY

Jedním z kritérií na vlastnosti relačních schémat jsou jejich normální formy. Normalizace vychází
požadavku efektivního uspořádání dat do paměti. Cílem je při současném zachování integrity a konzistence databáze. Po normalizaci musí zachován požadovaný informační obsah, zejména s ohledem na efektivní práci s relačním datovým modelem. Tím se rozumí, že v normální formy relací musí zaručit jednoznačnost a správnost odpovědí na dotazy.
Na začátku je nenormalizovaná struktura relací, jak si je většinou představí uživatel. Uživatelovy představy o obsahu databáze je nutné upravit do stavu, použitelného v relačním systému.
Proces normalizace znamená postupné rozložení původních "nenormalizovaných" tabulek do soustavy více menších tabulek tak, aby nedošlo ke ztrátě informací v původní tabulce obsažených.
Tento proces je označován jako ***bezztrátová dekompozice*** -je možné kdykoliv vytvořit si původní
tabulku nazpět.
S procesem normalizace databáze úzce souvisí funkční závislost, jedno z integritních omezení.
Funkční závislost je definována mezi dvěma množinami atributů v rámci jednoho schématu relace,
v němž vymezuje množinu přípustných relací.

PRVNÍ NORMÁLNÍ FORMA (1.NF)

Relace je v první normální formě (l.NF) , jestliže každý její atribut je prvkem jednoduché domény.
Každou relaci lze algoritmicky na l.NF převést tak, že doménu, která není jednoduchá, nahradíme odpovídajícími jednoduchými doménami.
Jinak řečeno: každý atribut musí být skalární.
objednávka map
Příklad nenormalizované relace:

CisloObjednavky	KodZakaznika	DatumObjednavky	Polozky	ObjednanoCelkem
1	CACTV	01.01.2001	5 klávesnice, 1 myš,	750 Kč
2	BSBEM	12.11.2001	45 CD-ROM, 45 obal	1012Kč
3	SUPRD	05.10.2001	2 myš, 1 mechanika FDD, 5 SDRAM	2457Kč

Atribut Polozky je ze složené domény. Je zde uveden počet kusů a druh výrobku. Jde též o chybu ve smyslu spojení dvou entit (objednávka a položek objednávky) do jedné relace.
Provedeme dekompozici na dvě relace.

CisloObjednavky	KodZakaznika	DatumObjednavky	ObjednanoCelkem
1	CACTV	01.01.2001	750 Kč
2	BSBEM	12.11.2001	1012Kč
3	SUPRD	05.10.2001	2457Kč

CisloObjednavky	CisloPolozky	PocetKs	Polozka	CenaZaKus
1	1	5	klávesnice	210
1	2	1	myš	125
2	1	45	CD-ROM	12
2	2	45	obal	9
3	1	2	myš	125

Častou chybou, kdy relace nevyhovuje 1.NF je též opakovaná skupina atributů.
Příklad - opakovaná skupina:

CisloObjednavky	KodZakaznika	Polozka1	Mnozstvi1	Polozka2	Mnozstvi2	Polozka3	Mnozstvi3	poloz4	Mnoz4
1	ANTON	Kabel	1	podložka	1	myš	3	CD	7
2	CVUYT	FDD	2	sluchátka	5	HDD	1	0	0

Předem při návrhu omezujeme nákup na maximálně 4 položky. Při nákupu jen jedné položky zůstávají ostatní položky s hodnotou NULL. Zbytečně narůstá objem databáze, která je nevyužita.
Další příklad opakované skupiny je tentokrát pro evidenci průtoku a stavu na řekách při měření na měřící stanici v průběhu dne v určitou hodinu:

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

Častou chybou jsou též atributy, které jsou různé složené kódy a příznaky. Např. kódování geomorfologických jednotek (viz.

Druhá normální forma (2.NF)

Druhá normální forma (2.NF) relace vznikne z l.NF, jestliže každý neklíčový atribut je úplně závislý na každém klíči.
Příklad relace, která nevyhovuje 2.NF

NazevVyrobku	JmenoDodavatele	NazevKategorie	TelefCisloDodavatele
Myš Treker	Treker Make	Polohovací zařízení	56879885
HP LaserJet 1200	Hewlet Packard	tiskárna	05236987
HP color Inkjet CP1700	Hewlet Packard	tiskárna	05236987

Výsledek bezztrátové dekompozice:
Relace Vyrobek

CisloVyrobku	NazevVyrobku	Kategorie
1	HP LaserJet 1200	Tiskárna
2	HP color Inkjet CP1700	Tiskárna
3	Myš Treter	Polohovací zařízení

Relace Dodavatel

CisloDodavatele	JmenoDodavetele	TelefCisloDodavatele
1	HP	05236987
2	Consigna	457998565
3	Autocont	125789564

TŘETÍ NORMÁLNÍ FORMA (3.NF)

Relace je v třetí normální formě (3NF), jestliže je v 2NF a navíc všechny její neklíčové atributy jsou vzájemně nezávislé. Příkladem je PSČ a název města. Tyto dva atributy jsou na sobě závislé a při změně jednoho se mění i druhý.
Příklad:

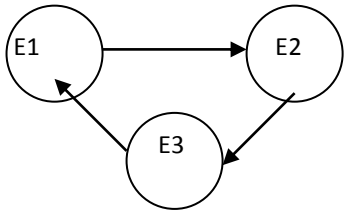
Firma	Adresa	Mesto	PSC
Autocont	Tř.1.máje 2	Olomouc	772 00
Konsigna	Tř. Svobody 42	Olomouc	779 00

Charakteristiky normalizačního procesu by se daly shrnout následovně
normalizovaný tvar

1. eliminace atributů, které obsahují jako prvky relace → l.NF
2. odstranění částečné závislosti neklíčových atributů na klíčích → 2NF
   (odstranění neúplných závislostí atributů na složených klíčích)
3. eliminace tranzitivní závislosti neklíčových atributů na klíčích → 3NF

DALŠÍ NORMÁLNÍ FORMY

První tři normální formy v roce 1972 definoval E. F. Codd. Společně s Raymondem F. Boycem později stanovil další normální formu, tzv. Boyce-Coddovu normální formu (BCNF). O relaci můžeme prohlásit, že je v BCNF, jsou-li všechny neklíčové atributy navzájem nezávislé, a navíc jsou všechny klíče nezávislé na klíčových atributech a úplně nezávislé navzájem.
Je-li relace v BCNF, pak je i v třetí normální formě.
Čtvrtá normální forma vyjadřuje princip : v jedné relaci se nesmí spojovat nezávislé opakované skupiny.
Pátá normální forma se týká poměrně vzácného případu *spojené závislosti*. Kdy entita E2 závisí na entitě E1 a entita E3 závisí na E2 a nakonec entita E1 závisí na entitě E3.



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

12. JAZYK SQL

DOTAZOVACÍ JAZYKY

Podstatnou část skupiny jazyků pro manipulaci s daty (JMD) tvoří dotazovací jazyky. Dotazovací jazyky slouží uživatelům k získání potřebných informací z databáze. Rozdělili jsme si JMD na procedurální a neprocedurální, a stejné rozdělení můžeme použít i pro dotazovací jazyky.
Procedurální dotazovací jazyky vyžadují algoritmický popis vyhledávání dat. Z tohoto důvodu obsahují prvky if-then-else, while-goto, umožňující přechody mezi prvky datové struktury.
V **neprocedurálních** jazycích jsou namísto popisu vyhledávání zadávány vymezující podmínky, kterým mají hledaná data vyhovovat. Tím se více **přibližují přirozenému jazyku**.
Dotazovací jazyk musí vedle základních uživatelských funkcí pro formátování výstupních dat obsahovat relační, porovnávací a aritmetické operátory, logické spojky, množinové funkce, aritmetické funkce, možnosti práce s časovými údaji. Z hlediska efektivnosti práce je kladen požadavek na možnost opětovného použití již vytvořených dotazů.
Kromě neprocedurálního dotazovacího **jazyka SQL**, jehož základy si popíšeme v následující kapitole, existuje mnoho úspěšných komerčních výrobků, které využívají dalších relačních dotazovacích jazyků, například QUEL či QBE. U jazyka QBE je dotaz vytvářen za podkladě grafického či polografického vyjádření databázových tabulek.
Dotazovací jazyk QBE (Query-By-Example) je druhým nejrozšířenějším relačním dotazovacím jazykem. Jak název napovídá, uživatel má při tvorbě dotazu k dispozici "příklad". Jazyk je určen
především pro koncové uživatele (úředníky, asistentky), kteří mají malé znalosti dotazovacího
jazyka. Například MS FoxPro či MS Access obsahují společně s možností vytvářet dotazy pomocí jazyka SQL i jazyk QBE.

VÝVOJ JAZYKA SQL

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

V letech 1973 až 1979 publikovali vědci firmy IBM velké množství publikací v akademickém tisku. Firma IBM však byla velmi pomalá ve svých obchodních strategiích a tudíž s prvním komerčně úspěšným relačním databázovým systémem podporujícím jazyk SQL přišla na trh firma ORACLE, založená v roce 1977.
V roce 1986 byl jazyk standardizován americkou ANSI, a o rok byl následován Mezinárodní organizací pro normalizaci (ISO). Verze jazyka pocházející z roku 1986 obsahuje původní dialekt jazyka SQL. SQL firmy IBM, a často se označuje jako SQL86. Jazykové prostředky SQL86 se jevily jako nedostatečné a proto pokračovaly práce na začleňování dalších konstrukcí. Výsledkem byla rozšířená verze jazyka, schválená organizací ISO v roce 1989: dnes označovaná jako SQL89. V roce 1992 byla standardizována zatím poslední verze jazyka, která je kvůli odlišení od předchozích verzí označována jako SQL92, případně SQL/92 (SQL-92, SQL2).
V roce 1993 se komise zabývající se jazykem SQL v ISO a ANSI, rozhodly, že další vývoj bude rozdělen do sedmi částí. Popis jednotlivých částí je uveřejněn na Internetu na webové domovské stránce SQL Standards na adrese <http: //www. jcc.com/sql_stnct.html>.
V roce 1996 pokračoval vývoj jazyka v podobě SQL3 a je velmi pravděpodobné, že ani zde se vývoj nezastavil a po SQL3 bude následovat SQL4. S novými verzemi jazyka stoupá i počet stran dokumentace. Zatímco popis SQL92 má přibližně 600 stran textu, u SQL3 je to již 1200 stran.
Jazyk SQL v současné době obsahuje více než jen původní dotazovací jazyk. Součástí jazyka jsou prostředky pro definici dat a jejich aktualizaci a dále definování přístupových práv k tabulkám.
Jazyk může být využíván v rámci některého programovacího jazyka (například PL/I, COBOL, C/C++). Tato verze SQL je označována jako hostitelská. Kromě ní existuje ještě interaktivní verze SQL (ISQL, interactive SQL). V případě využití hostitelské verze jsou SQL jsou příkazy vždy nějakým způsobem odděleny od příkazů programovacího jazyka. Implementace jazyka SQL se pro jednotlivé databáze mohou lišit. Výrobci se však snaží plnit existující normy. Například SQL server 2000 od firmy Microsoft nazývá svůj jazyk Transact SQL (T-SQL). Podobně databáze ORACLE 9i má své specifické řešení a tím je jazyk PL/SQL, který je rozšířením standardního jazyka o některé možnosti vyšších programovacích jazyků: větvení, skoky, cykly, vytváření procedur, funkcí a knihoven.

Důležité vlastnosti jazyka SQL lze odvodit z vlastností relačního datového modelu:

- data jsou uživateli zobrazena vždy ve formě tabulek
- data jsou uložena nezávisle na formě tabulek,
- uživatel se nemusí starat o fyzickou strukturu a uložení dat,
- tabulky jsou v databázi identifikovány jménem,
- nezáleží na pořadí sloupců v tabulkách,
- nezáleží na pořadí řádků v tabulkách.

Kromě tabulek je možné pro urychlení odezvy systému na uživatelské dotazy vytvářet tzv. **indexy**, což jsou soubory obsahující vhodně uspořádaná data pro snazší vyhledávání. (Indexy nejsou součástí relačního modelu dat.)
SQL nyní obsahuje prostředky pro:

1. Definici dat
2. Manipulaci dat
3. Výběr dat
4. Definice přístupových práv

Databáze v SQL se skládá ze souborů, obsahující tabulky a pohledy.
Každý příkaz SQL je zakončen středníkem.

PŘÍKAZY PRO DEFINICI DAT

Příkaz pro vytvoření databáze je to nejsnadnější. Jeho tvar je

CREATE DATABASE jmeno_databaze;
Místo jmeno_databaze napište konkrétní jméno databáze, například:

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

Vytvoření tabulky vypadá takto:

CREATE TABLE jmeno_tabulky

(seznam_prvků_ tabulky_oddělených_čárkami,

seznam_integritních_omezení);
Položku jmeno_tabulky nahraďte skutečným jménem tabulky, dále v kulatých závorkách definujte jména atributů a jejich datových typů. Atributy jsou odděleny čárkami.

Př. CREATE TABLE tZamestnanec

(cislo CHARACTER(6) NOT NULL PRIMARY KEY,

jmeno CHARACTER(15) NOT NULL,

vaha INTEGER,

datum DATE DEFAULT 1.1.1900);

Pozn.: *Access* menu Dotaz → Dotaz SQL → definiční

do okna editoru CREATE TABLE; DROP TABLE;

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

NOT NULL

DEFAULT počáteční_hodnota

UNIQUE (seznam_klíčových_atributů_oddělených_čárkou)

PRIMARY KEY (seznam_klíčových_atributů_oddělených_čárkou)

FOREIGN KEY

CHECK (plat > 4500 AND plat < 90000)

CONSTRAINT – integritní omezení

Příklad integritního omezení může vypadat i takto (*zase udělat na orná půda, vinice, …*

CHECK (VALUE IN 'svobodny','ženatý','rozvedeny','vdovec');

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

CREATE TABLE tVztahVlastnikNemov

(IDVlast INTEGER NOT NULL FOREIGN KEY,

IDNem INTEGER NOT NULL FOREIGN KEY,

PRIMARY KEY (IDVlast,IDNem)

);

Příkaz na smazání celé tabulky je

DROP TABLE jméno_tabulky;

Př. DROP TABLE tZamestnanec;

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

ALTER TABLE jmeno_tabulky zmena1[, zmena2…]**;**

Př. ALTER TABLE tZamestnanec ADD vyska INTEGER;

ALTER TABLE tZamestnanec ADD RodneCislo CHARACTER(11) UNIQUE;

ALTER TABLE tZamestnanec DROP vyska;

ALTER TABLE tZamestnanec ADD INDEX RodC (RodneCislo);

Vytvoří nový index pro atribut RodneCislo, který nazve RodC.

ALTER TABLE tZamestnanec ALTER vyska SET DEFAULT VALUE '170'); ???

PŘÍKAZY PRO MANIPULACI S DATY

Slouží k vkládání, odstraňování a aktualizaci dat.

INSERT INTO jmeno_tabulky (jmena_atributu) VALUES (seznam hodnot);

Př. INSERT INTO tZamestnanec VALUES ('101','Vydrová',65,'01.05.72');

Upozornění! Jeden příkaz INSERT vloží pouze jeden záznam.

DELETE FROM jmeno_tabulky WHERE podminka;

Smaže všechny záznamy vyhovující podmínce následující za WHERE.

UPDATE jmeno_tabulky SET atribut=nova hodnota WHERE atribut=stara_hodnota LIMIT

Př. UPDATE Zamestananci SET vaha=70 WHERE jmeno='Vydrová'

Aktualizuje atribut záznamu, který vyhovuje podmínce. Klausule LIMIT omezí celkový počet řádků, které příkaz aktualizuje.

PŘÍKAZY PRO VÝBĚR DAT

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

SELECT – FROM - WHERE

SELECT seznam_jmen_atributů

FROM seznam_jmen_tabulek

[WHERE podmínka]

[GROUP BY typ_skupiny]

[HAVING where_definice]

[ORDER BY atribut];

Klausule v hranatých závorkách jsou volitelné.

Př.

SELECT * FROM tZamestanaci;

SELECT Dite.Prijmeni, Dite.Jmeno, Dite.DatumNarozeni

FROM Dite

ORDER BY Dite.Prijmeni;

Př.

SELECT Zamestnanci.Prijmeni, Zamestnanci.Jmeno, Zamestnanci.DatumNarozeni INTO Studenti IN 'studenti.mdb'

FROM Zamestnanci;

Př.

SELECT Zamestnanci.Prijmeni, Zamestnanci.Jmeno, Stav.Stav

FROM Stav INNER JOIN Zamestnanci ON Stav.Stav = Zamestnanci.Stav

WHERE ((Stav.Stav) Like "vd*")

ORDER BY Zamestnanci.Prijmeni, Zamestnanci.Jmeno;

Při testování podmínkou where mohou být používány následující operátory:
Pozn.: Access Návrhové zobrazení Dotazu - Menu Dotaz→ Zobrazení SQL (editační okno)

PŘÍKAZY PRO DEFINICI PŘÍSTUPOVÝCH PRÁV

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

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

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

DROP	databáze, tabulky	odstranit databáze nebo tabulky

Obecná forma příkazu je:

GRANT {ALL PRIVILEGS |seznam_prav } [sloupce]

ON jmeno_objektu

TO {PUBLIC |uzivatel1 [, uzivatel2,…]} [WITH GRANT OPTION]

Př.: Výběr z tabulky tVyrobek povolen všem uživatelům

GRANT SELECT

ON tVyrobek

TO PUBLIC;

Př.:

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

GRANT DELETE, UPDATE (CenaVyrobku)

ON tVyrobek

TO Skladnik;

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

REVOKE seznam_prav [(sloupce)]

ON jmeno_objektu

FROM uzivatel

Pokud byla při přidělování práv uživateli použita možnost WITH GRANT OPTION, můžete ji zrušit pomocí:

REVOKE GRANT OPTION

ON jmeno_objektu

FROM uzivatel

13. PROGRAM A SKRIPT

ROZDÍL: SKRIPT X PROGRAM

Skript je v informatice zdrojový kód programu, který je tzv. interpretován, tj. čten a spouštěn za běhu speciálním procesem, tzv. interpretem.

Program je protikladem k interpretovanému kódu. Program je přeložený (kompilovaný) do strojového kódu.

SHODNÉ VLASTNOSTI PGM. A SKRIPTU

Kód programu i kód skriptu vytvořen v programovacím jazyku (hovoří se o tzv. *skriptovacím jazyku*), který má přesně stanovenou formální gramatiku (tj. pravidla, syntaktické elementy, jazykové konstrukty atd).

Typické znaky skriptů

často zohledňuje rychlý vývoj programu a tak umožňují některé prvky, které v překládaných jazycích nenajdeme. Jedná se například o:

netypové proměnné,

automatické nastavení hodnot u nedefinovaných proměnných a konstant,

zotavení z chyb, které neústí v ukončení skriptu.

VÝHODY SKRIPTU

Výhody skriptu

Není nutné mít nainstalovaný kompilátor a provádět po každé změně kódu kompilaci.

Snadnější údržba, vývoj a správa kódu.

Skript často tvoří rozšiřitelnou (parametrickou) část nějakého softwarového projektu.

Mění se jen rozšiřující část, nemusí se pokaždé rekompilovat hlavní spustitelný soubor.

Z toho důvodu skripty najdeme u *her*, složitějších softwarových řešení nebo jako hlavní součást *dynamických internetových stránek*.

NEVÝHODY SKRIPTU

Pomalá rychlost vykonávání. Interpretace stojí určitý strojový čas a nikdy nebude tak rychlá jako spouštění přeloženého (a optimalizovaného) programu.
O trochu vyšší paměťová náročnost. Interpret musí být spuštěn a tedy zabírá určitou operační paměť.
Skriptovací jazyky mají většinou větší omezení než překládané programovací jazyky (např. co do přístupu do paměti, ovládání tzv. rukovětí procesů a kontextových zařízení apod).

KDE ZAPISOVAT KÓD SKRIPTU?

textový editor – Notepad,…
Python Command prompt
IDE (Integrated DeveLopment Enviroment): PythonWin, IDLE

Výhody IDE – vše z jednoho místa:
write, save, run, debug
instalováno s ArcGIS:
ArcGIS v. 9.1 – PythonWin 2.1
ArcGIS v. 9.2. – PythonWin 2.4
ArcGIS v. 9.3. – Python 2.5

14. METADATA A METAINFORMAČNÍ SYSTÉMY

METADATA

Metadata jsou určité charakteristiky, atributy, které slouží k popisu jiných dat.
Obecně jsou metadata také data a mohou mít své vlastní charakteristiky (atributy nebo též deskriptory) a tak vznikají vyšší úrovně metadat.
Rodíl mezi daty a metadaty je v jejich *použití*.
Metadata je možné ukládat nebo vybírat z archivu (repozitory) podobně jako ukládáme data do databází. Příkladem je popis jednotlivých datových položek databázových tabulek. Tyto popisy jsou často uloženy odděleně od vlastních dat. Příkladem je popis jednotlivých datových položek databázových tabulek. Tyto popisy jsou uloženy odděleně od vlastních dat.
K čemu slouží metadata?
Metadata jsou data, která mají za úkol především popsat data. Slouží pro přesnou a korektní identifikaci a interpretaci dat.
Metadata pro prostorová data by měla být organizována a spravována s využitím **metainformačních systémů**.
Metainformační systém tak podporuje efektivní využívání samotných dat. Důležitou složkou metadat kromě popisu vlastních dat bývají i údaje o kontaktním místě (popř. osobě).

PŘÍKLAD METADAT

Velice důležitou složkou dat (samozřejmě i prostorových) je jejich popis.Tento popis označovaný obvykle jako metadata je často podceňován a opomíjen. Uvažme následující data: (1, 100, 150, 0.0064), (2, 120, 145, 0.0044), atd. Tato data nám byla dodána panem Dvořákem a jsou důležitá pro naši práci. Pan Dvořák předtím než odjel na dovolenou (kde není k zastižení) sdělil pouze, že tato data dostal od někoho z firmy Velmi Velká Firma, a.s. a, že jsou to koncentrace kadmia v podzemní vodě z vrtů v námi sledované oblasti. Po prohlédnutí takovýchto dat však zjistíme, že jsou pro nás naprosto bezcenná. Můžeme se dohadovat, že čísla 0.0064, 0.0044, atd. jsou koncentrace kadmia v mg/l (neboť taková koncentrace je v naší lokalitě očekávána). Dále můžeme čísla 1, 2 atd. vyhodnotit jako identifikátory jednotlivých vrtů. Čísla 100, 150, 120, 145, atd. by mohly být souřadnice daných vrtů. Soubor obsahuje hodnoty z 1500 vrtů. My však evidujeme pouze 500 vrtů. Používáme souřadnicový systém S-JTSK (Jednotná trigonometrická síť katastrální). Pro identifikaci vrtů používáme následující alfanumerické kódy: 1-1, 1-2, 11-1 , IV-4, atd. Data, která máme k dispozici, jsou pro naši potřebu velmi špatně identifikovatelná, ne-li neidentifikovatelná a tudíž pro naši práci bezcenná.
Některý z následujících popisů dat (metadata) nám umožní původně bezcenná data využít.
Metadata1 : Položky jednotlivých záznamu: Identifikátor vrtu, souřadnice x v místním souřadnicovém systému, souřadnice y v místním souřadnicovém systému, koncentrace kadmia v mg/l. Místní souřadnicový systém má osy orientované stejně jako S-JTSK a počátek ve vrtu č. 8, který má souřadnice S-JTSK: x=11 00303.22, y=455938.44. Měření byla prováděna 11.4.2000 v 9:00 hod.
Metadata2: Veškeré dostupné informace o daných datech má k dispozici pan Novák. Pan Novák je k dispozici na telefonním čísle 5861295472 nebo e-mail adrese
novak@mojeposta.cz.
Metadata3: Položky jednotlivých záznamů: Identifikátor vrtu, souřadnice x v místním souřadnicovém systému, souřadnice y v místním souřadnicovém systému, koncentrace kadmia v mg/l. Místní souřadnicový systém má osy orientované stejně jako S-JTSK a počátek ve vrtu č. 8, který má souřadnice S-JTSK: x=11 00303.22, y=455938.44. Měření byla prováděna 11. 4. 2000 v 9:00 hod. Veškeré další dostupné informace o daných datech má k dispozicipan Novák. Pan Novák je k dispozici na telefonním čísle 5861295472 nebo e-mail adrese naovak@mojeposta.cz.

Zatímco metadata1 nám umožní přímé využití dat (po několika úpravách), metadata2 nám umožní kontaktovat osobu (kontaktní místo), která může, ale také nemusí poskytnout potřebné informace pro využití dat. Metadata2 však mohou umožnit získání jiných doprovodných informací o poskytnutých datech. Ještě lepší variantou je kombinace obou popisů do metadat3.
Tento stručný a názorný příklad poukazuje na několik skutečností, které se týkají metadat. Z příkladu vyplývá především to, že bez vhodných metadat jsou data často bezcenná nebo obtížně použitelná. Druhá část příkladu demonstruje to, že často tou nejdůležitejší složkou metadat je aktuální kontakt na zodpovědnou osobu.

STANDARTIZACE METADAT

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

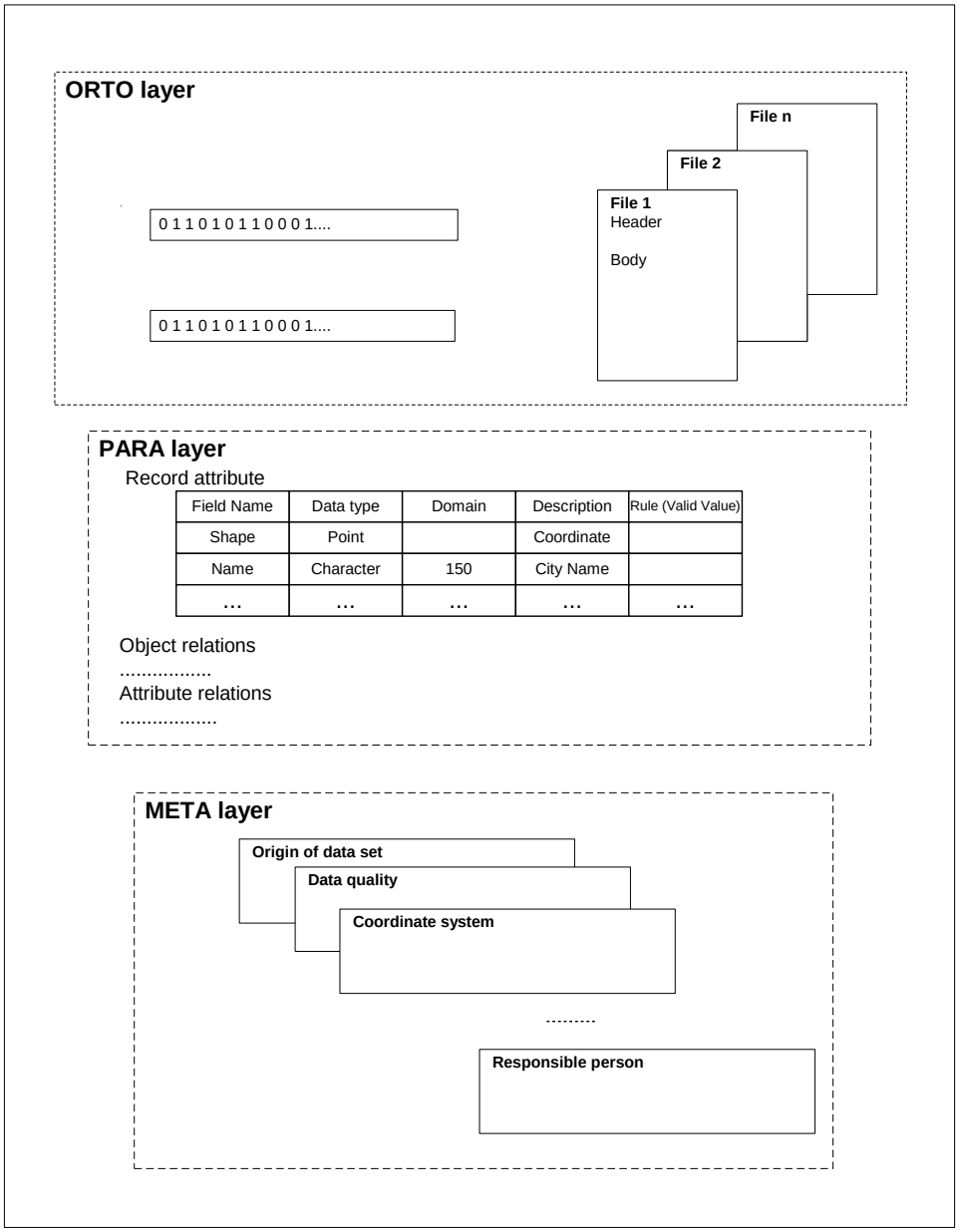
DATOVÁ SADA

Data bývají obvykle sdružována do kolekcí, které jsou označována jako datové sady (data sets) nebo datové soubory (data files). Pojem "datová sada" je obecnější oproti pojmu "datový soubor". V dalším textu bude termín "datový soubor" představovat konkrétní soubor v souborovém systému paměťového média. Pojem "datová sada" bude představovat data tvořící logický celek v rámci určitého informačního systému či datové báze. Může se tedy jednat o jeden datový soubor či kolekci těchto souborů.
Pojem datová sada nebude omezován jen na digitální data, ale může jím být označena i analogová forma dat (např. papírová mapa, atlas map, tištěný seznam majitelů parcel v okrese Nový Jičín).

METADATOVÉ HLADINY

Metadata jsou doplňkovou informací k datům. Metadata stojí na rozhraní mezi daty a informacemi a jsou prostředníkem k interpretaci, verifikaci a užití dat. Metadata mohou být chápána na více úrovních a mohou a mohou plnit různý účel v interpretaci dat. Z hlediska teorie zpracování dat můžeme na data pohlížet na třech úrovních:
1. **fyzická (orto)**
2. **logická (para)**
3. **meta**

Fyzická úroveň se zabývá tím, jak jsou data uspořádána na paměťovém médiu.
Logická úroveň se zabývá popisem obsahu dat. Hodnoty těchto vlastností jsou pak součástí samotných dat. Také zde bývají údaje o vztazích.
Meta úroveň se zaměřuje na datový soubor jako celek. Meta úroveň se zabývá popisem typu datové sady, důvodem jejího vzniku, historií, kvalitou dat, identifikací datové sady v rámci nějakého systému, podmínkami šíření datové sady, vztahem osob a organizací k datům. Meta úroveň může obsahovat i prvky logické úrovně.



Obr č.: Metadatové hladiny

METAINFORMAČNÍ SYSTÉM (METAIS)

= IS, který dokáže sofistikovaně pracovat s metadaty a vyřizovat požadavky na něho kladené.

- důležitými charakteristiky MetaIS:
  - Identifikace metadat (schopnost zpracovat neomezené množství metadat).
  - Verifikace metadat (ověření správnosti testováním).
  - Interpretace metadat (zpracování obsahu metadat, validita).
  - Distribuce metadat (prezentace ve vhodné formě).
- Uživatelé MetaIS mohou vyhledávat metadata o požadovaných datových sadách s využitím standardních vyhledávacích mechanismů.
- MetaIS většinou publikuje svá metadata prostřednictvím služby WWW

MIDAS - GeoNetwork

- http://gis.vsb.cz:8080/geonetwork2
- postaven na GeoNetwork OpenSource a umožňuje snadné sdílení geografických informací mezi organizacemi.

MetIS (Metainformační systém spol. T-mapy)

- http://www.tmapy.cz/
- aplikace vyvinuta na základě požadavků a potřeb orgánů veřejné správy ČR

MicKA (HELP SERVICE - REMOTE SENSING spol. s.r.o.)

- http://bnhelp.netart.cz/
- komplexní systém po práci s metadaty

DUBLIN CORE - ISO 15836

= standard určený k vytváření metadat dokumentů na internetu

- vychází z knihovnických systémů
- definuje velice omezené množství základních položek s možností jejich další strukturalizace ve stromové struktuře podle potřeb uživatelů.
- položky jsou pouze textového charakteru a pro popis geografických dat jsou nedostatečné
- jednoduchý systém složený z několika položek s prefixem "DC".
- možno vytvářet další podtřídy rozšířením předdefinovaných pomocí tečky
- metadata je možno vkládat do hlaviček HTML dokumentů.
- existují vyhledávače, které je umí zpracovávat a tak poskytovat tříděné informace na rozdíl od "textových" webových vyhledávačů.
- standard DC obsahuje základní sadu patnácti prvků *(Dublin Core Metadata Element Set)*, z nichž žádný není povinný

ISO 19115/19119 METADATA

- norma ISO 19115 standardizuje metapopis prostorových dat
- počítá se s tím, že bude postupně nahrazovat současně používané standardy: FGDC (USA), ANZLIC (Austrálie), CEN (EU).
- výrobci softwaru i producenti dat postupně na tuto normu přecházejí
- současné aktivity EU v oblasti prostorové infrastruktury (INSPIRE) počítají s touto normou jako jedinou pro popis prostorových dat.
- v ČR zrušeny předběžné normy CEN a do češtiny byla přeložena norma ISO - přijata za normu ČSN
- byla inspirována stávajícími metadatovými normami
- je však značně rozsáhlejší
- snaží se implementovat řadu číselníků, které omezují zadávání volného textu
- měla by přispět k ujednocení interpretace jednotlivých položek.

MIDAS – ČESKÝ ON-LINE KATALOG GEODAT

Midas je metainformační katalog geodat v české státní správě.
Z hlediska výše uvedené teorie obsahuje metadata nejvyšší úrovně (tj. meta úrovně).

HISTORIE :

- Byl vyvíjen CAGI od roku 1999, nejdříve jako pilotní projekt. Na vývoji se podíleli též pracovníci VŠB Ostrava.
- MIDAS je provozován na serveru Institutu geoinformatiky VŠB –TU OSTRAVA. Je přístupný přes Internet na adrese http://gis.vsb.cz/midas.
- Obsahuje více než 3 tisíce datových sad o geodatech převážně vytvořených a používaných ve státní správě- zejména na okresních úřadech.
- Údaje o datových sadách se pořizovali přímo na okresních úřadech pověřenými pracovníky.
- K plnění se používal program MIDAS lite, který je určen pro jednoduché plnění daty a jednoduché vyhledávání. Je to lokální metainformační systém. Viz obr.3.
- Se změnou organizace státní správy k 1.1.2003 (zrušení okresních úřadů, vznik krajských úřadů a pověřených obcí) je nachystán projekt MIDAS-kraj. V rámci tohoto projektu se provede aktualizace údajů v MIDASu, zejména kontaktních míst a odpovědných osob.

Struktura:
MIDAS popisuje datové sady těmito složkami:

- Identifikace datové sady
- Základní popis
- Prostorové schéma – popis geometrické struktury
- Prostorové referenční systémy
- Rozsah dat – časový a prostorový
- Kvalita dat – původ, užití, časový údaj, úplnost,…

Příklad vyhledávání si uveďmě na vyhledávání datových sad sloužících k dálkovému průzkumu země:
V záložce *Klasifikace* lze zadat check "Prostová data".
Při výběru podle této klasifikace lze nalézt 93 datových sad pro ČR. Jsou nalezeny záznamy o leteckých snímcích, družicových snímcích. V údajích *Popis vzniku* lze nalézt například pro letecké snímky lesů Kutná Hora, kdo je klasifikoval, jakém sw byly zpracovány atd. V údajích o polohové přesnosti lze nalézt i přesnost snímku v metrech. Datové sady z dálkového průzkumu jsou z hlediska prostorového schématu označeny jako rastrový obraz.

MICKA

Micka je systém pro práci s metadaty prostorových dat
Podporované standardy
Systém je utvořen tak, že je možno implementovat libovolný standard, který je reprezentován XML dokumentem. Pro vložení struktury standardu existuje zvláštní správcovský modul. Ve verzi 1.0 jsou podporovány tyto standardy:
Jazykové prostředí
Uživatelské prostředí je vícejazyčné, v současné době je podporována čeština a angličtina, další jazyk je možno přidat vyplněním tabulky termínů v databázi. Přepínat mezi jazyky je možno ťuknutím na odpovídající vlaječky v horní liště programu.
Samotná metadata je možno paralelně vyplňovat v různých jazycích podle nastavení uživatele. Ve všech případech je používána kódová stránka UTF-8.

15. GEOPROCESSOR NA BÁZI ESRI

- **geoprocessor** nabízí několik možností volání – přímo z ArcToolboxu, z modelu skriptu, ze skriptu (verze 9 nabízela ještě volání z příkazové řádky), v ArcGISu nově přímo knihovna ArcPy
- **geoprocessing**
  - různé druhy geograficky založených operací
  - všechny operace, kde je vstup a výstup (i přidání atributu do tabulky)
  - projekce, konverze (CAD - geoDB), datamanagement, prostorové analýzy
- existují tři možné přístupy k nástrojům geoprocessoru – toolbox, toolset, tool
- nástroje jsou trojího druhu – systémové, model a skript
- od verze 9.1 je snaha umísťovat všechny nástroje do toolboxů → výhoda v tom, že všechyn nástroje pak lze použít ve skriptech
- dostupnost geoprocessoru přímo ze skruptu pomocí zápisu

```
import arcgisscripting
gp = arcgisscripting.create()
```

- každý objekt má **vlastnosti** (charakterizují objekt) a **metody** (činnost objektu, reakce na událost), interakce s objekty pomocí vlastností a metod; programový model geoprocessoru model zobrazuje všechny objekty a metody (vztahy), které mohou být využity při skriptování, vlastnosti jsou v něm zaznačeny jako činka nebo půlčinka, metody pak jako šipka; všechny nástroje jsou metody, všechny proměnné prostředí jsou vlastnosti
- **volání vlastností** - gp.Workspace, gp.CellSize; nastavení metod - gp.Buffer_analysis(), gp.Clip_management()
- navíc má Geoprocessor vlastnosti a metody, které nejsou proměnné prostředí nebo existující nástroje (Tools) - gp.Usage(), gp.Toolbox, gp.Exist(), gp.AddToolbox()
- **nastavení úrovně licence** – gp.SetProduct(productCode - ArcView, ArcEditor, ArcInfo, Engine, EngineGeoDB, ArcServer); kontrola licence – gp.CheckProduct(productCode)
- **přiražení hodnoty do vlastnosti** – Object.Property = Value
- **použizí metody** – Object.Method (arg, arg)
- **kontrola existence dat** – gp.Exists ()
- **přepsání výstupu** – gp.OverwriteOutput (defaultně nastaveno na „false")
- metody **DESCRIBE**
  - vrací popisné informace o datech (Shape type, Spatial reference, Maen cell size, Band count a další)
  - proměnná dscFC (dscFD, dscRD), dotaz na třídu prvků

    - vrací objekty - ty závisí na typu popsaného objektu (jiné pro třídu prvků, jiné pro dataset)
    - nutné pro řízení chodu skriptu (ověření vhodnosti vybrané třídy prvků)
    - vlastnosti: FD – Describe, DatasetType, MExtent; FC – Describe, ShapeType, OIDFieldName, MExtent; RD – Describe, BandCount, Width, Extent, Format
- metody **ENUMERATION**
    - slouží k výčtu nebo vyjmenování seznamů a také k dávkovému zpracování
    - každá metoda vrací list (seznam) dat
    - vlastnosti: ListFields, ListIndexes, ListDatasets, ListFeatureClasses, ListRasters
- metody **CURSOR**
    - každá metoda Cursor vrací objekt; pracuje s řádky v tabulce nebo třídě prvků
    - přidání záznamu (řádce), čtení hodnot v záznamu (řádce), změna hodnot v záznamu (řádce)
    - metody: InsertCursor (pro vložení nové řádky), SearchCursor (pro čtení hodnot v záznamu, řádce), UpdateCursor (pro změnu hodnot v záznamu, maže záznam), UpdateField
- **tvorba dynamických skriptů**
    - skripty mohou být dynamické nebo statické (ze statického se dynamický stává přidáním argumentu)
    - existují dva různé způsoby, jak vytvářet argumentu - Python má funkci sys.argv [] - je třeba importovat modul sys, první argument začíná číslem 1; geoprocessor má metodu GetParametsAsText - není nutný import sys modulu, první argument začíná 0
- **Model Builder** – slouží k vytvoření modelu s využitím nástrojů z Toolboxů; lze vytvořit i parametrický model, který nabízí možnost opakovaného spouštění

16. SKRIPTOVÁNÍ V JAZYCE PYTHON PRO ARCGIS

(charakteristika jazyka Python, postup tvorby skriptu, volání knihoven, definice uživatelského nástroje, předávání parametrů, struktura try: except:)

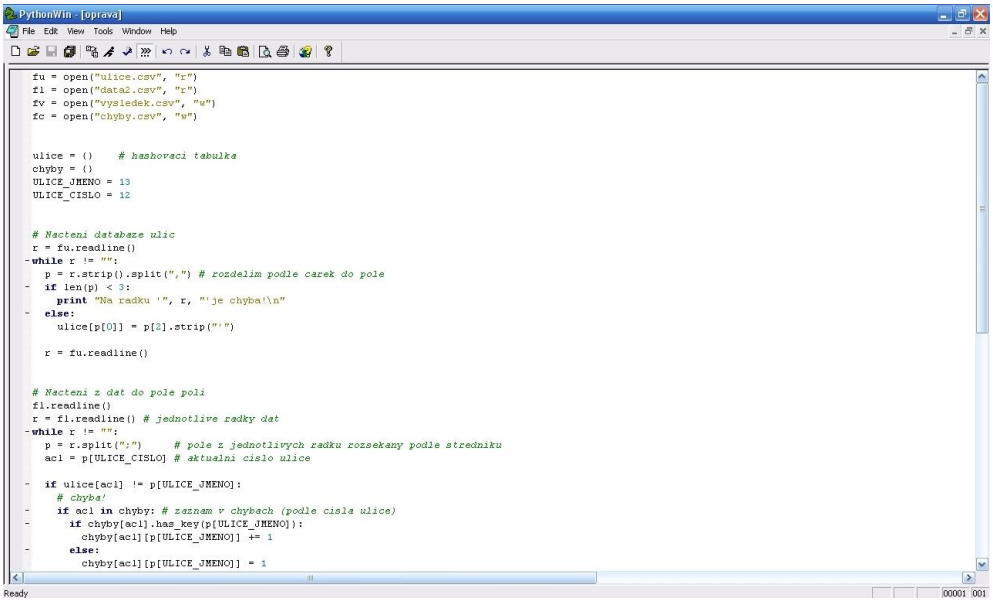
Python je skriptovací jazyk se zaměřením na objektově orientované programování. Je také interpretovaný, tudíž ho lze překládat do spustitelného souboru. Byl navržen v roce 1990 Guido van Rossumem v Matematickém centru Stitching v Nizozemí. Python je vyvíjen jako open source a nabízí zdarma instalační balíky pro většinu běžných platforem (Unix, Windows, Mac OS). Název Python nemá přímou souvislost s hady (python=krajta), ale i v nižších verzích můžeme vidět, že se i přesto používá v logu (viz obrázek 2).

Obr. 2: Logo Python

Prostředí jazyka můžeme snadno rozšířit pomocí funkcí a datových typů napsaných v jazycích C nebo C++. Python můžeme použít také, jako skriptovací jazyk pro aplikace v jiných jazycích.

Zdrojové kódy a standardní knihovny jsou volně ke stažení z domovské stránky Pythonu (http://www.python.org/). Poslední verze 3.1.1 vyšla 17.8.2009.

Jazyk Python má výrazně jednoduchou a čitelnou syntaxi kódu. Zajímavě je vyřešena bloková struktura kódu. Je rozdílná od jiných jazyků, kde se blokové struktury (tzn. určení např. těla cyklu, podmínky nebo funkce) definují uzavíráním kódu mezi závorky. V Pythonu se využívá mezer a odsazování kódu od začátku řádku. Také se nepoužívá středníků na konci příkazu. Uživatelské prostředí (viz obrázek 3) je jednoduché a přehledné.



Obr. 3: Uživatelské prostředí PythonWin

Počínaje verzí 9.0 zavádí Python ve svých produktech ArcGIS jako skriptovací jazyk. Pro prostorové zpracovávání geografických dat (tzv. geoprocessing) je u ArcGIS desktopu k dispozici několik skriptovacích jazyků, ESRI víceméně doporučuje Python, jako jazyk s jednoduchou a srozumitelnou syntaxí a koncepcí, snadno pochopitelný, objektově orientovaný, snadno integrovatelný s jazyky C/C++ a Java, volně dostupný.

Podpora pro skriptování v ArcGIS je zajištěna pomocí rozhraní tzv. COM objektů (Component Object Model), což je jakýsi model či standard popisující možnost komunikace mezi objekty různých aplikací vytvořených v různých programovacích jazycích. Funkčnost prostorového zpracování dat zajišťuje ArcObjects komponenta geoprocessor. Ke všem jejím funkcím a nástrojům má Python přístup právě přes rozhraní COM. Uživatel má možnost v Pythonu využít všech nástrojů, kterými disponuje ArcGIS ToolBox.

Skriptovací jazyky umožňují definovat opakování určitých vybraných operací, Pythonu je tedy možno v prostředí ArcGIS vhodně využít k dávkovému zpracovávání dat, například nastavit souřadnicový systém u všech položek datového skladu (shp soubory v adresáři, položky v geo-databázi SDE, …), dále možnost odstraňování, selekce, úprava datových objektů podle zadaných vlastností či atributů. Lze manipulovat s rastrovými i vektorovými daty, transformovat je do jiných datových formátů. Více informací lze získat v dokumentaci k produktu ArcGIS, nebo na domovských internetových stránkách společnosti ESRI, například v sekci virtuální výuky, kde jsou k dispozici obrazové on-line kurzy, dále v sekci pro podporu zákazníků lze nalézt spoustu více i méně užitečných skriptů[5].

PROČ PSÁT SKRIPTY?

- Automatizace práce
  - Kopie všech vstupních dat do geodatabáze
  - Vykonání operace projekce, cplip, buffer na řadu data setů
- Možnost spustit skript v určený čas
  - Windows plánovač úloh
  - Windows AT příkaz
- Snadné předávání kódu
  - Skript je samostatný soubor

PYTHON SCRIPTING LANGUAGE

- Možno použít různé skriptovací jazyky, které podporují COM technologii
- (VBScript, Jscript, Perl, VBA, C++,…)
- Není třeba se učit proprietární jazyk jako AML nebo Avenue
- Python – možné psát v notepad
- Python command prompt
- Podporuje obsáhlé objekty

- Debugging tools

17. ESRI GEODATABÁZE

(dělení single x multi user geodatabáze, ArcSDE, třídy prvků, domény, podtypy, anotace, relace, pravidla topologie, geometrická síť,)

Přednášky

18. ZOBRAZENÍ DAT V POČÍTAČI

BINÁRNÍ SOUBOR

Binární soubor je v informatice počítačový soubor, který obsahuje jakákoliv data, která jsou následně zpracovávána počítačovým programem. Obsahem souboru jsou čísla v binární soustavě (řetězec nul a jedniček), která reprezentují jistým způsobem uloženou informaci, což může být zvuk, obrázek, video, ale i formátovaný text, databáze a podobně. Při čtení binárního souboru je proto nutné vědět, jak uložená data interpretovat.
Opakem binárních dat je čistý text (anglicky plain text), což je přímo zobrazitelný text (řada písmen je zakódována do čísel podle tabulky – znakové sady).

OBSAH BINÁRNÍHO SOUBORU

Binární soubor je označení pro data, která je nutné po přečtení nějakým způsobem interpretovat. Data mohou být uložena jinak, než ve formě řady za sebou uložených bajtů (osm bitů) a při jejich čtení je nutné je správně interpretovat.

Hlavička

Pokusíme-li se zobrazit uložený zvuk jako obrázek, zobrazíme nejspíše chaotickou změť bodů, která nevytváří rozumný obrázek (platí samozřejmě i obráceně). Proto binární soubory někdy obsahují hlavičku (záhlaví), která uložená data popisuje (tzv. metadata). Pomocí formátu záhlaví lze pak obvykle zjistit, jaká data jsou v souboru uložena. Například EXE soubory začínají dvěma charakteristickými znaky (MZ, NE, LE, NX, PE). V unixových systémech existuje nástroj file, který podle těchto charakteristických znaků dokáže určit, jaká data jsou v souboru uložena.[1]

Vlastní data

Za hlavičkou následují vlastní data obsažená v binárním souboru. Jejich organizace je závislá na tom, jak je příslušný program zapisuje (a čte). U některých formátů je všeobecně známo, jakou mají strukturu, u jiných to známé být nemusí, případně to ani není nutné (jsou-li k dispozici nástroje, které umí binární soubor zpracovat, prohlížet a upravovat).
Data mohou být čtena po jednotlivých bajtech, ale i po více bajtech nebo naopak po jednotlivých bitech. Mohou být interpretována jako text, obraz, zvuk a podobně.
Mezi binární soubory se známým formátem patří multimediální soubory (například GIF, JPEG, MPEG, …), spustitelné soubory (EXE, ELF, …) a další. Formát souborů pro uchování dokumentů textového editoru Microsoft Word (.DOC) je převážně známý, avšak jeho interpretace přesto není dokonalá, protože význam jednotlivých údajů je jen odhadován (viz otevírání těchto souborů v OpenOffice.org).

E-mail

Při posílání binárních souborů pomocí e-mailu je nutné jejich obsah transformovat do textové podoby (Base64, Quoted-printable, Uuencoding a podobně), protože starší systémy pro přepravu elektronické pošty umí posílat jen čistý text. Konvertovaná podoba má však typicky větší délku, než původní binární data.

Binární a textový režim

Microsoft Windows umožňuje při otevírání souboru specifikovat parametr systémového volání, který určuje, je-li soubor textový nebo binární. Unixové systémy tyto režimy nerozlišují a nakládají se všemi soubory jako s binárními, což odráží fakt, že rozdělení souborů na textové a binární je do určité míry subjektivní.

ČTENÍ BINÁRNÍCH SOUBORŮ

Pokud otevíráte binární soubor v textovém editoru, každá skupina osmi bitů bude typicky přeložená jako jeden znak. Uvidíte (pravděpodobně nelogicky) zobrazení textových znaků tak, jako by byl tento soubor otevřen v jiné aplikaci. Tato aplikace bude mít svůj vlastní význam pro každý bajt: je možné, že daná aplikace bude nakládat s každým bajtem jako s číslem, a pošle na výstup tok čísel v rozpětí 0 až 255. A nebo je možné, že bude interpretovat tato čísla v bajtech jako barvy, a zobrazí odpovídající obrázek. Jestliže se se samotným souborem nakládá jako s vykonavatelným a spustitelným souborem, tak se počítač pokusí interpretovat tento soubor jako sérii strojových instrukcí.
Na čtení hexadecimálních (možné dokonce i decimálních) hodnot pro zodpovídající bajty binárního souboru se může použít hex editor. S bajty se nakládá stejně jako s jejich hexadecimálními hodnotami v hexadecimálním editoru.

ZNAKOVÁ SADA NEBO TAKÉ KÓDOVÁNÍ

(také kódová stránka) je kód, který páruje sekvence znaků z dané množiny (abecedy) s jejich jinou reprezentací, jako je sekvence přirozených čísel, bytů nebo elektrických pulzů, za účelem ukládání textu v počítači nebo přenosu textu telekomunikačními sítěmi.

Příkladem může být Morseova abeceda, která kóduje písmena latinky (a další znaky) pomocí sérií dlouhých a krátkých stisků telegrafního klíče. Dalším příkladem je znaková sada ASCII, která kóduje 128 znaků americké abecedy s číslicemi a dalšími symboly jako 7bitová čísla.
Kódování je také prostředkem pro kompresi (tj. zmenšení) anebo šifrování (tj. utajování) dat.
Proces standardizace začal zavedením **znakových sad** ASCII (1963) a EBCDIC (1964). Omezení těchto sad brzy vyplavala na povrch a vedla tak k vytvoření mnoha ad-hoc rozšíření. Potřeba podporovat více skriptovacích systémů včetně rodiny CJK – východoasijských skriptů si vyžádala ve srovnání s předchozími ad-hoc experimenty podporu mnohem většího množství znaků a rovněž systematický přístup k jejich kódování.

ASCII

ASCII je nepochybně základním kódováním, z kterého vychází v euro-americkém prostoru ostatní standardy. Drtivá většina osmibitových znakových sad pouze rozšiřuje ASCII tím, že přidávají významy kódům 128-255, které se v ASCII nepoužívají.

Kódování češtiny

Sada	Popis
CP852	osmibitové kódování češtiny v systému MS-DOS
ISO 8859-2	osmibitové kódování češtiny v UNIXových systémech
Windows-1250	osmibitové kódování češtiny používané v systémech Microsoft Windows
Kód Kamenických	osmibitové kódování češtiny částečně kompatibilní s CP437 (zachovává semigrafické znaky)
Unicode	současná celosvětová znaková sada, používaná v současných OS

Standardy ISO 8859-x

Standardy ISO/IEC 8859-x definují osmibitové znakové sady, používané zvláště v UNIXu/LINUXu. Novější unixové distribuce již přecházejí na Unicode s kódováním UTF-8.

Sada	Skript
ISO 8859-1	Latin-1, Západoevropský
ISO 8859-2	Latin-2, Východoevropský
ISO 8859-3	Latin-3, Jihoevropský
ISO 8859-4	Latin-4, Baltský
ISO 8859-5	Cyrilice
ISO 8859-6	Arabský
ISO 8859-7	Řecký
ISO 8859-8	Hebrejský
ISO 8859-9	Latin-5, Turecký
ISO 8859-10	Latin-6, Nordický
ISO 8859-11	Thaiský
ISO 8859-13	Latin-7, Baltský
ISO 8859-14	Latin-8, Keltský
ISO 8859-15	Latin-9, Západoevropský
ISO 8859-16	Latin-10, Jihovýchodoevropský

Vybrané kódové stránky Windows

Operační systémy rodiny Microsoft Windows používají několik osmibitových kódování, více nebo méně podobných standardům ISO8859-x. Kromě toho interně používají pro definici fontů šestnáctibitovou znakovou sadu WGL-4 (windows glyph 4), která obsahuje všechny znaky ze zde zmiňovaných kódových stránek Windows-1250 až Windows-1258.

Sada	Skript
Windows-1250	Latin-2, Středoevropský (podobný na ISO8859-2)
Windows-1251	Cyrilice
Windows-1252	Latin-1, Západoevropský (rozšiřuje ISO 8859-1)

Windows-1253	Řecký (podobné ISO 8859-7)
Windows-1254	Turecký (shoduje se s ISO 8859-9)
Windows-1255	Hebrejský (rozšiřuje ISO 8859-8)
Windows-1256	Arabský (částečně shodný s ISO 8859-6)
Windows-1257	Latin 13, Baltský (téměř shodný s ISO 8859-13)
Windows-1258	Vietnamský (velmi podobný Windows-1252)

Vícebytové standardy

Kódování	Popis
UTF-8	kódování unicode s nejmenší kódovou jednotkou délky osm bitů. Nezávisí na tom zda je systém „little-endian“ nebo „big endian“, je kompatibilní s ASCII.
UTF-16	jiné kódování unicode používané ve Windows a v jazyku Java. Nejmenší kódová jednotka má délku šestnáct bitů, má varianty „little-endian“ a „big-endian“, není kompatibilní s ASCII.
GB18030	oficiální standard pro kódování čínštiny. Jde o netriviálně přemapovaný unicode, tak aby byl kompatibilní s GBK. Nejmenší kódová jednotka má délku osm bitů, je kompatibilní s ASCII.
GBK	microsoftí kódování čínštiny. Nejmenší kódová jednotka má délku osm bitů, je kompatibilní s ASCII.
UCS-2	překonaný šestnáctibitový kód implementující pouze základní rovinu unicode, je nahražen UTF-16. Používal se ve Windows NT a Windows 2000.
WGL4	panevropská šestnáctibitová microsoftí sada fontů

OPRÁVNĚNÍ: Sada atributů určujících druh přístupu uživatele k datům nebo objektům v databázi.)

Oprávnění	Povolují uživateli
Otevírání/spouštění	Otevření databáze, formuláře nebo sestavy nebo spuštěnímakra v databázi
Výhradní přístup	Otevření databáze pro výhradní přístup
Čtení návrhu	Zobrazení tabulek, dotazů, formulářů, sestav nebo maker v návrhovém zobrazení
Změna návrhu	Zobrazení a změny návrhu tabulek, dotazů, formulářů, sestav nebo maker nebo odstranění těchto objektů
Správa	Pro databáze změny hesla databáze, replikace databáze a změna vlastností po spuštění Pro tabulky, dotazy, formuláře, sestavy a makra plný přístup k těmto objektům včetně možnosti přidělovat oprávnění
Čtení dat	Zobrazení dat v tabulkách a dotazech
Aktualizace dat	Zobrazení a změna, nikoli však vložení nebo odstranění dat v tabulkách a v dotazech
Vkládání dat	Zobrazení a vložení, nikoli však změna nebo odstranění dat v tabulkách a v dotazech
Odstraňování dat	Zobrazení a odstranění, nikoli však změna nebo vložení dat v tabulkách a v dotazech

- Některá oprávnění automaticky implikují nastavení dalších oprávnění. Oprávnění Aktualizace dat pro tabulku automaticky implikuje například oprávnění Čtení dat a Čtení návrhu, protože jsou nutná k úpravě dat v tabulce. Oprávnění Změna návrhu a Čtení dat implikují oprávnění Čtení návrhu. Pro makra oprávnění Čtení návrhu implikuje oprávnění Otevírání/spouštění.
- Chcete-li v prostředí s více uživateli navrhovat formuláře, sestavy, makra a moduly, musíte otevřít databázi aplikace Access s výhradním přístupem. To znamená, že v databázi aplikace Access musíte mít oprávnění Otevřít s výhradním přístupem.
- Chcete-li, aby uživatelé měli přístup k propojené tabulce, pak jim v serverové databázi udělte oprávnění Čtení dat a Čtení návrhu. Oprávnění Změna návrhu na propojení tabulky definujte v uživatelské databázi, takže uživatelé mohou pohodlně změnit propojení tabulky. Pokud chcete uživatelům zamezit přístup ke koncové tabulce, ale chcete, aby uživatelé mohli prohlížet data a opětovně propojovat tabulky, pak odeberte všechna oprávnění z koncové tabulky a použijte dotazy v uživatelské databázi s jejich vlastností **Práva pro spuštění**nastavenou na hodnotu **Vlastník**.

19. ARCHIVACE, ZÁLOHOVÁNÍ A OBNOVA DAT

SOUBOROVÝ A DATABÁZOVÝ PŘÍSTUP

Datová základna - samostatná část informačního systému (IS)
měla by co nejlépe obstát při změnách v IS (IS se skládá z technologie a neformální části IS;hardware+software+datová základna)
návrh datové základny - pohled na to, co datová základna obsahuje-odpovídá pohledu na skutečnost, ve které se odehrává činnost podniku nebo organizace (data musí být stabilní)
základní pojmy databázových struktur

- záznam (record) - množina údajů v datové základně, které se týkají jednoho reálného objektu (věcí, jevů, osob, děj, např. záznam o konkrétním druhu zboží)
- **atribut** - zaznamenává vlastnost reálného objektu, např. název zboží, jeho váha, rozměry, barva, materiál, datum výroby

SOUBOROVÝ PŘÍSTUP

Historicky první - aplikace ukládá svá data do 1 či několika datových souborů (data file). Soubor obsahuje záznamy o 1 typu objektů VU formě datových vět. Hodnoty atributů v 1 záznamu se nazývají položky
omezení: každá aplikace si udržuje svá data (může je měnit). Problémové využívání týchž dat pro různé aplikace (např. adresa bydliště - nutnost opakovaných změn dat v řadě souborů při přestěhování)
organizace vět v souboru - položky tvoří strukturu záznamu (v aplikačním programu). Záznamy se ukládají jako věty do souborů, hodnoty atributů v 1 záznamu se nazývají položky.
omezení souborového přístupu - koncipován pro jednoúkolové zpracování. Těsná vazba struktury dat na aplikační program.

DATABÁZOVÝ PŘÍSTUP

Základní princip: koncepcí oddělení dat od aplikací a svěření jejich správy do databáze. Databáze spravuje a řídí datovou základnu. Jednotlivé aplikace, pokud chtějí nějaká data uložit nebo přečíst žádají o tuto složku databáze

Požadavky na databázový systém
sdílení dat - odstranění redundance a paralelní přístup. Každý údaj v databázi je pouze jedenkrát a mohou k němu přistupovat různé aplikace; různé aplikace obecně mohou do databáze přistupovat paralelně (zároveň)
nezávislost aplikace - na změnách ve fyzickém uložení dat, abstraktní pohled na data, možnost definicí datových typů, centrální popis dat
typické uložení dat upravuje systém řízení báze dat (SŘBD)-pro přístup k datům nabízí aplikacím a uživatelům nástroje, s jejichž pomocí mohou vyjádřit, jaká data požadují.
ochrana dat před neoprávněným přístupem a poškozením, různí uživatelé mají různá přístupová práva do databáze, databáze má být schopná ochránit data i před výpadky elektřiny
Souborová struktura ukládání dat

Způsob ukládání údajů lze provádět různým způsobem. Nejjednodušší je ten, jak ho známe z běžného používání počítače. Jednotlivé soubory se ukládají do složek a tyto se ukládají případně do dalších složek až do počtu vrstev, které považujeme za

dostatečné, abychom jednoznačně rozlišili zařazení příslušného souboru na to místo, kam dle našeho uvážení patří. Tvoření souborové struktury je jednoduché a pro uživatele, který má smysl pro pořádek je i dostatečné.
Ovšem jenom do chvíle, kdy nepotřebujeme, aby se uložené soubory mezi sebou začaly propojovat a své dílčí informace nabídly celku. Pak nastává několik problémů, které musíme řešit v rámci konkrétních, používaných aplikací. Je to zejména definování místa, kde jsou soubory uložené a jak k nim mají aplikace přistupovat. Další problém je v přístupu do dat z několika aplikací najednou, ale i sdílení a editace dat více uživateli najednou. Některé problémy tímto způsobem ani nelze řešit, lépe řečeno jejich řešení je neefektivní. Proto přecházíme na jiný způsob organizace dat.

Databázový systém
Soubor programů, které byly navrženy podle dohodnutých kritérií tak, aby soustředil data do jednoho celku, obhospodařoval je a umožňoval přístupy k různým aplikacím a různým uživatelům najednou, se jmenuje systém řízení báze dat. V informatice se pro tento systém používá známá zkratka DBMS z anglického database management system. Používání DBMS má v informatice velký význam pro všechny zúčastněné strany.
Programátor aplikačního softwaru se nemusí soustředit tolik na zajištění přístupů k datům, protože je má na jednom místě. Správce dat má o datech dokonalý přehled a při dodržení stanovených pravidel může data rychle a přesněji propojovat. Řadový uživatel se zase dostává k datům vždy, kdy to potřebuje, může je editovat a nemusí čekat, až mu „uvolní“ přístup jiný uživatel. Základní požadavky na DBMS doplňuje možnost různého způsobu vyhledávání, bezpečnost před neoprávněným přístupem a také dostatečné prostředky pro správu dat.
Na to, abychom mohli DBMS využívat, potřebujeme databázi. Databáze je v zásadě několik souborů, jednoho předmětu zájmu. Pro příklad můžeme uvést, že předmětem zájmu je sledování stavu úseku silnice po časových intervalech. Jeden soubor dat tvoří poloha silnice je definovaná souřadným systémem atributy x,y, tyto jsou konstantní. Druhý soubor dat tvoří kvalita povrchu definována atributem „q“ a ten se každým rokem mění. K silnice lze vytvářet další soubory dat, které budou obsahovat jiné atributy (dopravní značení, materiál silnice, stáří) a každý soubor dat lze vhodně vázat z jiným.
Jak bude vypadat struktura dat, určují katalogy dat. Nemusí se vypracovávat přímo pro aplikační program, ale fungují zcela samostatně. Lze konstatovat, že systém řízení báze dat a databáze tvoří databázový systém. Práci s databází nám umožňují databázové jazyky. Pro definici dat používáme jazyk, který vychází z použitého datového modelu. Pro vlastní práci s daty používáme ve většině případů dotazovací jazyk – SQL.

Databázové modely
Informativně vzpomeňme existenci hierarchického datového modelu. Jeho princip je postavený na hierarchii vztahů nadřazených a podřazených typů entit.
Je využívána asociace 1:1 a 1:N. Vztahy M:N neexistují, tudíž nelze data libovolně propojovat. Jiný, síťový model, připouští i typ M:N, ale jenom za cenu náročnější a rozsáhlejší databáze.
Hierarchické a síťové datové modely se v dnešní době prakticky nevyužívají.

Relační model je postaven na matematickém přístupu. Každá entita (objekt) obsahuje libovolné atributy. Pro názornost si představme tabulku, která má řádky, jako jednotlivé entity a sloupce jako jednotlivé atributy.
Alespoň jeden atribut je vždy unikátní nebo je vytvořen definovanou kombinací vybraných atributů. Takovému atributu se říká klíč. Pomocí klíčů lze vytvořit všechny typy asociací – 1:1, 1:N, M:N.
Pojem relace nám vlastně uvádí, že s jednotlivými tabulkami lze provádět úkony, kterých výsledkem je vytvoření nové tabulky. Takto lze na základě matematických poznatků vytvářet sjednocení, průnik, rozdíl a součin tabulek, dle požadovaných parametrů. Hlavním cílem při budování relačního systému je zachovat stabilitu datové struktury a současně eliminovat nadbytečnost záznamů a tím šetřit i hardwarové prostředky (paměť, místo na disku).
Pro geografické informační systémy není relační model úplně dostatečný. Je výhodný pro manipulaci s neprostorovými daty. Popisování objektů reality prostřednictvím tabulek v relačním modelu je značně složité. Geoobjekty se zde musí náročně definovat pomocí několika tabulek, primární klíče vyžadují zase pečlivou definici a unikátnost. Proto se využívá objektově orientovaný model. Tento je vytvořen na základě několika zásad.
Každý objekt je modelován s vlastní identitou a ta je po celou jeho existenci trvalá. Současně je zapouzdřen, má vlastní atributy a chování. Jeho vnitřní vztahy nás nezajímají, zajímá nás jenom jeho vztah k okolí. Objekty, které mají stejné atributy a stejné chování, se popisují jako třída objektů. Tyto třídy se mohou dělit na subtřídy dle specializace, přičemž dědí atributy a chování své nadtřídy. Dá se stručně říci, že snahou objektově orientovaného modelování, je přenést svět z relačních tabulek do větší reality, vytvářet objekty a seskupovat je podle všech údajů do tříd.

Prohledávání databáze
Vycházíme z toho, že uživatel ukládá různé údaje proto, aby je mohl v budoucnu využívat buď přímo nebo pro vytvoření nových údajů. Vzhledem k tomu, musí vždy provést dotaz pomocí SQL k vyhledání požadovaných dat dle vybraného kritéria. Obecně má dotaz tři části. Specifikaci vybíraných údajů, formulaci zvolených podmínek a pokyn, co se má s vybranými daty provést.
Pro GIS je základním dotazem prohledávání databáze. Můžeme vznášet prostorový dotaz na databázi v otázce – kde se nachází dané místo? Druhý typ dotazu na databázi může být atributový v otázce – které objekty mají požadovanou vlastnost? Častěji potřebujeme zvolit dotaz, který bude kombinací na prostor i atributy. V této souvislosti je nutné zvážit výběr

odpovídající databáze.

Jakou databázi vybrat?
Složitější dotazy vyžadují kvalitnější softwarové vybavení. Přesto lze konstatovat, že je nutné zodpovědně zvážit výběr aplikačního i databázového softwaru, aby odpovídal jednak velikosti databáze, rozsahu požadovaných úkonů a počtu přistupujících uživatelů. Současně by měl splňovat i finanční možnosti pořizovatele, nároky na bezpečnost a v neposlední řadě i nároky na aktivní obsluhu. Na trhu se dnes nachází open source databáze PostgreeSQL, MySQL. Tyto v řadě případů stačí, navíc ve spojení s operačním systémem Linux vytváří finančně velmi zajímavou kombinaci.
V klasické kategorii lze vzpomenout databázové stroje MS SQL ve verzi Express, Workgroup, Standard a Oracle ve verzi Express, Standard, Standard One, pro velké databáze zejména MS SQL Enterprise a Oracle Enterprise a produkty IBM – Informix Dynamic server a DB2. Tímto není seznam databází uzavřen, vzpomenul jsem jenom nejpoužívanější databáze na českém trhu. Při rozhodování je nutné vycházet i ze samotné GIS aplikace, kterou chceme používat a dbát na doporučení jeho tvůrce, který za portaci na vybrané databáze a operační systémy, ručí. Jestli tomu tak není, zvolte si raději jinou GIS aplikaci.

ZÁLOHA

Záloha nebo **záložní kopie** (anglicky backup) je kopie dat uložená na jiném nosiči (nebo i místě). Záložní data jsou využívána v případě ztráty, poškození nebo jiné potřeby práce s daty uloženými v minulosti. Zálohování probíhá nepravidelně (např. v domácnostech) nebo pravidelně podle rozvrhu (např. ve firmách).

ZÁLOHOVÁNÍ DAT

Při zálohování většího množství dat se obvykle používá specializovaný program (například i v systému Microsoft Windows je součástí instalace), který celý proces zálohování usnadňuje (viz níže). Pro zálohování většího množství dat je možné použít také specializovaná zařízení (hardware), která pracují poloautomaticky nebo plně automatizovaně.
V poslední době je využíváno komplexních zálohovacích systémů, které umožňují efektivně zálohovat mnoho počítačů propojených počítačovou sítí nebo naopak na mnoho počítačů propojených v síti data zálohovat (tzv. *úložný cluster*).

TYPY ZÁLOH

Pro různé podmínky se používají různé strategie zálohování. Volba správné strategie je závislá na tom, jestli je potřeba se zálohami pracovat velmi často nebo je naopak požadována maximální délka archivace zálohovaných dat. Existují i další kritéria, která odrážejí konkrétní specifické podmínky.

Nestrukturovaná
: Nestrukturovaným úložištěm může být větší množství disket, CD, DVD medií s minimem informací o záloze. Tento způsob je nejjednodušší, ale není příliš oblíben u větších firem.

Úplná + Inkrementální
: Tento model má za cíl vytvořit více kopií zálohovaných dat vhodnějším způsobem. Nejdříve je provedena úplná záloha všech dat. Posléze je prováděna inkrementální záloha (ukládány jsou pouze soubory, které se změnily od předešlé úplné nebo inkrementální zálohy). Hlavní nevýhodou je, že při obnovení zálohy je potřeba pracovat s úplnou zálohou a následně se všemi inkrementálními zálohami až k požadovanému okamžiku zálohy, což může být velmi náročné na pracovní prostor.

Úplná + Rozdílová
: Rozdíl oproti předešlé metodě je v tom, že po úplné záloze se každá částečná záloha zachytí všechny soubory vytvořené nebo změněné od vytvoření úplné zálohy, třebaže některé už jsou obsaženy v předešlé částečné záloze. Výhodou je, že obnova zahrnuje obnovení pouze poslední úplné zálohy a potom její překrytí poslední rozdílovou zálohou, takže je proces obnovení více odolný vůči defektu média se zálohou.

Zrcadlová + Reverzně přírůstková
: Tento model obsahuje zrcadlo reflektující stav systému po poslední záloze a historii přírůstkových záloh. Výhodou je, že máme neustále k dispozici aktuální plnou zálohu a ukládáme pouze historii změn. Každé zálohování se automaticky promítá do zrcadla zrcadla a soubory, které byly změněny, jsou přesunuty do přírůstkové zálohy. Tato metoda se nehodí pro přenosná media, protože každá záloha musí být provedena pomocí srovnání se zrcadlem.

Průběžná ochrana dat
: Tato metoda využívá místo plánovaných periodických záloh okamžitý zápis každé změny do žurnálu změn (logu). Provádí se ukládáním bytů nebo celých bloků dat místo ukládání celých změněných souborů. Průběžný záznam změn v žurnálu umožňuje získat obraz dat v minulosti. Naproti tomu prosté zrcadlení dat na druhý disk (např. RAID 1) stav v minulosti nezachycuje.

PROČ ZÁLOHOVAT A ZPŮSOBY ZTRÁTY DAT

Zálohování není jen výsadou firemní sféry, kde je tento proces esenciální nutností bez které by žádná firma nemohla přežít, ale stejně by si měl počínat každý uživatel počítače. Naprostá většina z nás zálohuje, jen tomu neříká zálohování, ale vypalování (ukládání dat na CD a DVD). Kvůli kterým situacím je třeba data pravidelně zálohovat?
Způsoby poškození a ztráty dat
Tuto kapitolku si můžeme rozdělit do několika skupin:

Poškození vlivem lidského faktoru

neúmyslné smazání
nesprávné používání nebo manipulace
nedbalost

Poškození způsobené selháním systému

výpadky elektrického proudu, přepětí, podpětí
výpadek operačního systému
selhání pevných disků
zničení vlivem programové chyby

Poškození s úmyslem data zničit

virová nákaza
sabotáž
krádež

Poškození fyzikálními a přírodními vlivy

kouř
požár
voda
zásah blesku
Jak vidno, možností jak přijít o svá drahocená data je mnoho a nikdy není možné veškeré jevy eliminovat. Je ale nutné vznik všech krizových situací MINIMALIZOVAT. A právě teď je čas na jednotlivá řešení zálohování.

MÉDIA PRO UKLÁDÁNÍ DAT

Magnetická páska
: Magnetická páska je již po dlouhou dobu nejvíce používané medium pro zálohování a archivaci dat. Některé nové pásky jsou již dnes rychlejší (čtení/zápis) než pevné disky. Nevýhodou je vysoká pořizovací cena páskové jednotky, výhodou pak nízká cena médií.

Pevný disk
: Poměr kapacita/cena disku se čím dál více zlepšuje. To dělá pevný disk soupeřem pro magnetické pásky. Výhodou disku je nízká přístupová doba, kapacita a snadnost použití.

NAS
: Network Attached Storage je pevný disk nebo pole pevných disků, které je připojeno k lokální síti. Může se jednat o jednoúčelové zařízení nebo server, jehož úlohou je skladování dat.

Optický disk
: Výhodou u těchto medií je hlavně cena a dostupnost pro všechny počítače s optickou mechanikou. Dalšími používanými formáty jsou CD, DVD, DVD-RAM. Nověji se používají také HD DVD a Blu-ray disky, které nabízejí mnohem větší kapacitu pro zápis, avšak jejich nevýhodou je zatím vysoká cena.

Disketa
: Dnes již prakticky „muzejní záležitost“. Používána v devadesátých letech minulého století.

Ostatní paměťová media
: Používají se například USB flash disky nebo různé druhy paměťových karet (Secure Digital, Memory Stick apod.)

Vzdálená zálohovací služba
: Vysokorychlostní internet se stává již běžnou součástí firem i domácností, proto popularita zálohovací služby přes internet roste. Tato varianta zálohování zabraňuje možnému zničení záloh v důsledku požáru, povodní či jiných nenadálých situací. Nevýhodou naopak může být pomalejší průběh zálohování v porovnání s klasickými paměťovými

medii a v neposlední řadě také zneužití citlivých dat ze záloh třetí osobou (hacker), která se může k těmto datům nelegální cestou dostat.

MANIPULACE S DATY

Při zálohování je dobré zároveň data různými způsoby zpracovat, zrychlí se tím rychlost zálohování, rychlost obnovy a bezpečnost dat.

Komprese

Existuje mnoho metod ke zmenšení velikosti zálohy a tím k ušetření diskového prostoru. Komprese je často používána u zálohování *magnetickými páskami*.

De-duplikace

Tato metoda umožňuje odstranit ze zálohy duplicitní složky a soubory. Pokud je velké množství podobných systémů zálohováno do stejného místa, existuje zde možnost, že zde bude nadbytek zálohovaných dat. Například pokud je zálohováno 20 systémů Windows, pak určitou část dat mají všechny tyto systémy společnou a není je tak třeba ukládat vícekrát, ale pouze jednou.

Duplikace

Principem je vytvoření dvou záloh na dvou různých mediích a na různých místech. Zvýší se tím rychlost při obnově dat a ochrana zálohy před poškozením.

Šifrování dat

Používáno u *datových nosičů*, kde je důležité zabezpečení dat a omezení nebo znemožnění přístupu k datům nežádoucí osobě. Nevýhodou je zpomalení procesu zálohování a také fakt, že šifrovaná data nemohou být efektivně zkomprimována.

ZÁSADY ZÁLOHOVÁNÍ DAT

- postupy zálohování volíme v závislosti na konkrétní situaci (interval změn dat, denní objem nových dat, důsledky ztráty dat aj.)
- kontrola záloh – většina programů (kompresní, vypalovací atd.) následně umožňuje kontrolu archivu
- popisujeme zálohy - co obsahují, datum vytvoření
- z instalačních médií by měla být pořízena alespoň jedna kopie, originální média by měla být ihned po pořízení kopií uložena na bezpečném místě (včetně instalačních hesel a čísel!), vlastní instalace probíhá z pořízených kopií
- ukládání záloh na fyzicky různá místa – důležité zálohy by neměly být uloženy u počítače (požár atd.)
- zajištění důvěrnosti dat (fyzicky, nebo alespoň zaheslováním zálohy)
- volba média (CD, DVD, Flash …) – médium volíme podle: rychlosti zálohování (čtení), pořizovací a provozní ceny, spolehlivosti média, spolehlivost obnovení, doby uchovávání dat, kompatibility

PRINCIP ZÁCHRANY DAT

Poškození paměťových médií a následnou ztrátu dat dle příčiny dělíme na dva základní typy

MECHANICKÉ:

Většinou fyzické poškození média, záchrana dat bývá obtížnější a často časově náročnější. Úspěšnost obnovy je často nižší než u softwarového poškození.

Nejčastější příčiny mechanického poškození:
- vadná elektronika
- odřený povrch
- vadné hlavy
- zadřená ložiska

Obnova dat HW poškození:
V případě mechanického poškození disku většinou nestačí softwarové nástroje. Pevný disk je třeba rozebrat v bezprašném prostředí. Zjistí se zda jde vyměnit pouze určitá část disku nebo je potřeba za pomoci servisních informací znovu načítat data ze samotných ploten disku. Pokud jsou poškozené plotny, musí ze zkontrolovat a zanalyzovat. Následně se provede opětovné čtení pomocí nových čtecích hlav a pokus o načtení servisních dat z nepoškozených ploten.

SOFTWAROVÉ:

Softwarové poškození se odehrává pouze na datech, médium samotné není poškozeno a lze ho po opravě datové struktury opět používat.

Nejčastější příčiny mechanického poškození:
- poškozený file system
- naformátování
- smazání dat
- napadení virem
- přepsaná servisní data

Obnova dat SW poškození:
Provádí se speciálním nízkoúrovňovým softwarem. Většinou je potřeba obnovení a zrekonstruování souborového systému. Při větším poškození se pravděpodobně nezachovají původní názvy souborů a adresářů.

20. PRINCIPY PUBLIKOVÁNÍ WWW

K čemu je dobré znát HTTP
Pro normální tvorbu stránek je znalost http protokolu vcelku **k ničemu**. Jakmile ale děláte složitější věci nebo se zajímáte o pokročilejší optimalizaci pro vyhledávače, je lepší vědět, co se mezi serverem a klientem vlastně děje.

Z http hlaviček můžete vyčíst např. informace o přesměrování, kešování, o cookies, o komprimaci nebo třeba o referreru.

Pokud píšete serverové skripty nebo složitější webové programy, je dobré vědět, kdy a jak posílat kterou http odpověď.

Jak funguje http protokol

Povídá si klient se serverem. Klient něco chce a server mu to dá.

- **Klientem** je nejčastěji internetový prohlížeč (Explorer, Mozilla, Opera), ale může jím být třeba i vyhledávací robot nebo jiný program.
- **Http server je program** běžící někde v serverovně na nějakém počítači (ten počítač se náhodou též označuje jako "server", ale nikoli jako "http server"). Nejpoužívanějším http serverem je program zvaný Apache.

Takže http protokol je takový jakoby jazyk, kterým si povídají dva programy. Povídají si po síti, nejčastěji po Internetu.

Klient většinou chce nějakou stránku -- připojí se na server a požádá server o URL stránky. Tato žádost je formulována v HTTP protokolu (samotné připojení na server je přes TCP protokol). Server tuto http žádost zpracuje a pošle zpátky odpověď, která je taktéž v HTTP protokolu. Například pošle klientovi http hlavičky a za nimi text stránky v HTML. Klient odpověď přijme, hlavičky si přečte, a stránku zobrazí.

Příklad http komunikace
Čtenář si chce přečíst například tuto stránku. Tato stránka má URL http://www.jakpsatweb.cz/server/http-protokol.html.

1. Čtenář toto URL zadá do prohlížeče.
2. Prohlížeč (tedy klient) si vyhodnotí doménu, přes DNS si zjistí, jaké IP adresy se má ptát.
3. Přes TCP protokol naváže spojení se serverem na zjištěné IP adrese. Teprve nyní začíná HTTP.
4. Prohlížeč pak pošle na server toto HTTP volání:

WORLD WIDE WEB WWW

- aplikace internetového protokolu http - myšlena soustava propojených hypertextových dokumentů
  = informační systém pro práci s hypertextovými dokumenty, ve kterých jsou odkazy na internetovské zdroje uváděny pomocí adresy
- nejrozšířenější služba v současném internetu

Tvorba a správa webových stránek přestala být pouze záležitostí informatiků
webmaster - spravuje obsah a strukturu Webu
webdesigner - určuje grafický vzhled Webu
webdeveloper - programuje dynamické webové stránky

- v roce 1989 definoval Tim Berners-Lee hypertextový systém pro švýcarské výzkumné středisko CERN
- v roce 1990 B-Lee vytvořil
  - hypertextový editor *WorlDwidEweb*
  - první webový server
  - publikoval na něm specifikaci základních standardů:
    - UDI (nyní URI - Unique Resource Identifier)
    - HyperText Markup Language (HTML)
    - HyperText Transfer Protokol (http)

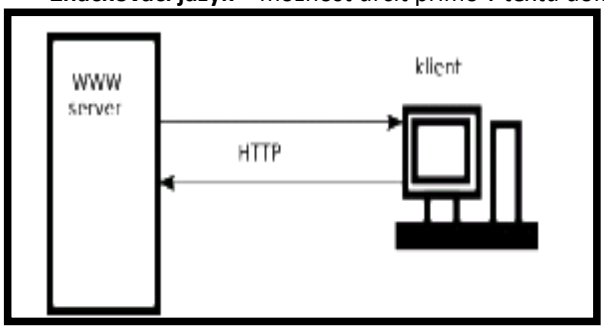
- v roce 1992 v provozu okolo padesáti webových serverů
- vznikají první grafické webové prohlížeče: Midal a Viola
- v roce 1993 vzniká Mosaic a architektura prohlížeče Mozilla
- pro standardizaci webových technologií - organizace W3C *(WWW Consortium)*
- v roce 1995 je na světě kolem 100 000 webových serverů

„Celosvětová pavučina" (web)

- v češtině se slovo web se používá:
  - pro označení celosvětové sítě dokumentů
  - pro označení jednotlivé soustavy dokumentů dostupných na tomtéž webovém serveru nebo na téže internetové doméně nejnižšího stupně (**=internetové stránce**).
- dokumenty umístěné na počítačových serverech jsou adresovány pomocí URL
- protokol HTTP je dnes již používán i pro přenos jiných dokumentů, než jen souborů ve tvaru HTML => World Wide Web se stává synonymem pro internetové aplikace
- nedostatkem je archivace zveřejněných dokumentů – lze je libovolně měnit a nelze zaručit, že se někdo nebude pokoušet falšovat soubor či, že dříve zveřejněný dokument bude na internetu stále k dispozici a k nalezení

STAVEBNÍ KAMENY WWW

- **Hypertext** – možnost navigace mezi dokumenty či jejich částmi
- **Identifikace zdrojů** – schopnost jednoznačně identifikovat každý zdroj (počítač, dokument atd.)
- **Klient-server architektura**
- **Značkovací jazyk** – možnost určit přímo v textu dokumentu jak mají jeho jednotlivé části vypadat (HTML).



HYPERTEXT

- strukturovaný elektronický text, obsahující odkazy na jiné texty, obrázky, zvuky, animace, video.
- používá se na internetu, ale i lokálně (encyklopedie, nápovědy atd.)
- informační systém, který zobrazuje informace v textu, který obsahuje návěstí odkazující na upřesnění nebo zdroje uváděných informací tzv. hyperlinky neboli česky (hypertextové) odkazy.
- odkazuje i na jiné informace v systému a umožňuje snadné publikování, údržbu a vyhledávání těchto informací.

UNIFORM RESOURCE LOCATOR URL

= jednotný lokátor zdrojů

- řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na Internetu
- definuje doménovou adresu serveru, umístění zdroje na serveru a protokol, kterým je možné zdroj zpřístupnit.
- jednotlivá pole v URL: protokol, doménové jméno, port, specifikace souboru, parametry
- některá pole jsou nepovinná – buď nemají význam, nebo se předpokládá předdefinovaná hodnota, závislá např. na protokolu (např. pro HTTP je implicitní port 80), nebo na aplikaci (pro webový prohlížeč se předpokládá protokol HTTP).

-příklad pro WWW stránku:

http://cs.wikipedia.org:80/w/wiki.phtml?title=URL&action=edit
protokol: http
server: cs.wikipedia.org
port: 80 – jelikož pro http je port 80 implicitní, není ho třeba v tomto konkrétním případě uvádět
dokument: /w/wiki.phtml – je uveden včetně cesty (adresáře) v rámci serveru
parametry: první parametr se jménem „title" a hodnotou „URL", druhý se jménem „action" a hodnotou „edit"

- pomocí URL lze zadat také autentizační informace:

mezi protokol a doménové jméno je možno vložit uživatelské jméno a případně i heslo oddělená navzájem dvojtečkou a od následující domény zavináčem

http://vilem:mojetajneheslo@www.ukazka.cz/

Webový prohlížeč

= browser - počítačový program, který slouží k prohlížení World Wide Webu (WWW)

- umožňuje komunikaci s HTTP serverem a zpracování přijatého kódu (HTML, XHTML, XML apod.), který podle daných standardů zformátuje a zobrazí webovou stránku
  - **textové prohlížeče -** zobrazují stránky jako text, obvykle velmi jednoduše formátovaný
  - **grafické prohlížeče -** umožňují složitější formátování stránky včetně zobrazení obrázků.
- pro zobrazení některých zvláštních součástí stránky, jako jsou Flashové animace nebo Javové applety, je třeba prohlížeč doplnit o specializované zásuvné moduly.
- nejznámější webové prohlížeče patří grafické Internet Explorer, Mozilla Firefox, SeaMonkey, Opera, Konqueror a Safari a textové Links a Lynx.

WEB 2.0

- charakterizován jako posun od centralizovaného zpracování / služeb k decentralizaci.

Web2.0 je přeměna webu dokumentů ve web dat, v platformu pro sdílení dat, kde si uživatel kontroluje svá data, z různých zdrojů.

Pokus o stručnou definici náplně Web 2.0:

- Blogy
- RSS
- Sociální síť a tagy
- API + AJAX (rozhraní + nadstavby)

Návrhové vzory:

- Long tail - mnoho malých zdrojů
- Uživatelé přidávají hodnotu
- Vlastnictví jedinečných dat je klíčem k úspěchu
- Agregování dat o způsobu chování uživatelů
- Uvolnění práv k dokumentům (creative commons)
- Stále se vyvíjející webové služby
- Kooperace místo řízení
- Software pro mnoho různých zařízení - platforma

Technologie realizace

- **Tagy** - označování obsahu uživateli, neboli folksonomy (jako protiklad k taxonomy - což je klasické třídění odborníky, např v knihovnách.
- **Sociální vztahy**, příslušnost ke skupinám, souvisí s tagováním Software, které bude poskytovat tyto služby, bude tím lepší, čím více bude uživatelů - to platí hlavně pro sociální systémy a folksonomy
- **RSS** - syndikace obsahu. Uživatel nebude muset hledat, co je nového ale informace k němu potečou přes zvolené kanály.
- **Long tail** - blogy, rss, Google adsense - samoobsluha pro zákazníky, hodně malých zdrojů
- **Blogy a Wikipedia** - Uživatel jako přispěvatel a tvůrce obsahu. Blogy jako zdroj obsahu spolu s RSS znamenají rozšíření obsahu, uživatelé se na něm podílejí - píší zprávy a úvahy, které byly dříve jen v centrálních mediích
- Programové nadstavby pro zpracování dat : **AJAX** - technologie, která umožňuje asynchronní přenos dat, aplikace nad obsahem databází a webů, webová rozhraní **(API)** pro přístup nadstavbových aplikací třetích stran.

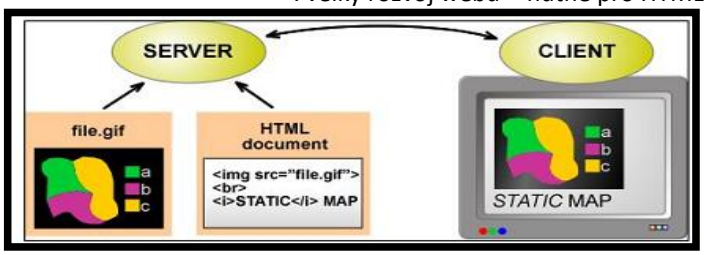
- Vznikne něco jako webové služby, které dokáží propojit data, např RSS a další webové služby pro přenos dat (SOAP a REST , přenos přes xml formát)

21. ZNAČKOVACÍ MAKROJAZYKY PRO DEFINOVÁNÍ STRUKTURY

HYPERTEXT MARKUP LANGUAGE HTML

- značkovací jazyk pro hypertext
- jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu
- jazyk je podmnožinou dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (*Standard Generalized Markup Language*)
- vývoj ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka
- v roce 1990 navržen jazyk HTML a protokol pro jeho přenos v síti – http
- v roce 1991 CERN zprovoznil svůj web.
- současně NCSA iniciovalo vyvinutí prohlížeče Mosaic
- vyvinut v roce 1993 pro počítače PC a Macintosh

- první prohlížeč s grafickým uživatelským rozhraním
    - -. velký rozvoj webu - nutné pro HTML definovat standardy



VERZE JAZYKA

Verze 0.9 – Byla vydána zhruba v roce 1991. Nepodporuje grafický režim

Verze 2.0

- zachycuje stav jazyka v polovině roku 1994. Standard vydala komunita IETF
- první verze, která odpovídá syntaxi SGML
- přidává k původní specifikaci interaktivní formuláře, podpora grafiky

Verze 3.2

- vydána 14. ledna 1997 a zachycuje stav jazyka v roce 1996.
- připravovaná verze HTML 3.0 nebyla nikdy přijata jako standard - byla příliš složitá a žádná firma nebyla schopna naprogramovat její podporu ve svém prohlížeči.
- standard už vydalo konsorcium W3C, stejně jako následující verze
- přidává k jazyku tabulky, zarovnávání textu a stylové elementy pro ovlivňování vzhledu

Verze 4.0

- vydána 18. prosince 1997
- do specifikace jazyka přidala nové prvky pro tvorbu tabulek, formulářů a nově byly standardizovány rámy (frames).
- verze se snaží dosáhnout původního účelu - prvky by měly vyznačovat význam (sémantiku) jednotlivých částí dokumentu, vzhled má být ovlivňován připojovanými styly. - některé prezentační elementy byly zavrženy.

Verze 4.01

- vydána 24. prosince 1999
- opravuje některé chyby verze předchozí a přidává některé nové tagy.
- původně se myslelo, že se bude jednat o poslední verzi a budoucnost bude patřit pouze XHTML.

Verze 5.0

- 7. března 2007 založena nová pracovní skupina HTML s cílem vývoje nové verze
- v květnu 2007 odhlasováno, že základem nové specifikace se stanou Web Applications 1.0 a Web Forms 2.0 ze specifikace WHATWG
- specifikace by měla být hotova v roce 2010

- Jazyk HTML je od verze 2.0 aplikací SGML

=> charakterizován množinou značek a jejich atributů pro danou verzi definovaných

- mezi **značky (tagy)** se uzavírají části textu dokumentu a tím se určuje význam (sémantika) obsaženého textu
- názvy jednotlivých značek se uzavírají mezi úhlové závorky ("<" a „>“).
- část dokumentu uzavřená mezi značkami tvoří **element (prvek) dokumentu**.
- součástí obsahu elementu mohou být další vnořené elementy
- **Atributy** jsou doplňující informace, které upřesňují vlastnosti elementu.
- **tagy** jsou **obvykle párové** - rozlišujeme počáteční a koncové značky
    - koncová značka má před názvem značky znak lomítko

př. pro označení odstavce: Text odstavce

-Některé značky jsou nepárové - neobsahují žádný text.

př. pro vykreslení vodorovné čáry <hr>

- tagy mohou obsahovat atributy popisující jejich vlastnosti nebo nesou jinou informaci.

odkaz

př. odkaz (tag a), jehož atribut href říká, kam se uživatel po kliknutí na něj dostane

- pro každou verzi existuje **definice pravidel DTD** (*Document Type Definition*).
- od v4.01 musí být odkaz na deklaraci v dokumentu uveden pomocí direktivy DOCTYPE
- DTD definuje pro určitou verzi, které elementy je možné používat a s jakými atributy
- dokument může mimo značkování obsahovat další prvky:
    - **Direktivy** - začínají znaky „<!“, jsou určeny pro zpracovatele dokumentu (prohlížeč)

- **Komentáře** - pomocné texty pro programátora, nejsou součástí obsahu dokumentu a nezobrazují se (prohlížeč je ignoruje).
- **Kód skriptovacích jazyků**
- **Definice událostí a kód pro jejich obsluhu**

STRUKTURA DOKUMENT V JAZYKU HTML

- **Deklarace DTD** - uvedena direktivou <!DOCTYPE.
- **Kořenový element** - element html (značky <html> a </html>) reprezentuje celý dokument.

- nepovinný, ale doporučeno ho používat

- **Hlavička elementu** - jsou to metadata, která se vztahují k celému dokumentu

- definují název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl ...

- **Hlavička** je uzavřena mezi značky <head> a </head>
- **Tělo dokumentu** - obsahuje vlastní text dokumentu. Vymezuje se značkami <body> a </body>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<!-- toto je komentář -->
   <head>
      <title>Titulek stránky</title>
   </head>
<!-- tělo dokumentu -->
   <body>
      <h1>Nadpis stránky</h1>
      <p>Toto je tělo dokumentu</p>
   </body>
</html>
```

Druhy značek (tagů)

- **Strukturální značky**

- rozvrhují strukturu dokumentu
- př. jsou odstavce (<p>), nadpisy (<h1>, <h2>)

- **Popisné (sémantické) značky**

- popisují povahu obsahu elementu
- př. nadpis (<title>) nebo adresa (<address>).
- současný trend je orientován právě na sémantické značky
  - usnadňují automatizované zpracovávání dokumentů a vyhledávání informací v záplavě dokumentů na webu – **sémantický web**
- vrcholem je jazyk XML.

- **Stylistické značky**

- určují vzhled elementu při zobrazení.
- př. značka pro tučné písmo (<b>).

- tento druh značek se **nedoporučuje** používat, trendem je používání **kaskádových stylů** oddělených od obsahu dokumentu
- umožňují definovat rozdílné zobrazení pro různá zařízení

Parsování v prohlížečích

- webové prohlížeče - programy s účelem prezentovat dokument na zobrazovacím zařízení - převážně monitoru počítače
- dokument je prohlížečem načítán a rozkládán (**parsován**, syntaktická analýza) na jednotlivé elementy.
- prohlížeč obsahuje tabulku značek, které podporuje - tabulku možné omezit typem dokumentu (DTD), je-li deklarován.
- každému elementu je poté přiřazen styl (způsob zobrazení)
- styly mohou být uvedeny ve stylovém předpisu
- vlastnosti stylů, které nejsou předepsány, doplní prohlížeč podle implicitního stylu, který má zabudován
- některé prohlížeče umožňují uživateli implicitní styly definovat

Novější prohlížeče pracují obecně ve dvou základních režimech

- **Standardní režim**
  - dodržována specifikace verze HTML, deklarovaná v dokumentu.

- elementy v dokumentu musí odpovídat verzi, neznáme elementy jsou považovány za chybu a nezobrazují se
- netolerují se syntaktické chyby

- **Quirk mód** - nestandardní režim
  - prohlížeč se snaží „napravovat“ chyby v syntaxi dokumentu,
  - domýšlí si chybějící koncové párové značky
  - neznámé elementy zobrazuje implicitním formátem.

- režim se stanoví podle typu dokumentu - rozhodování je dost složité *doctype sniffing*
- prohlížeče mají různé implicitní styly zobrazování a chovají se různě v quirk módu

=> stejný dokument se tedy může v různých prohlížečích zobrazovat různě

- rozdíly jsou i mezi verzemi prohlížeče stejného výrobce

- současné chování prohlížečů = výsledek vývoje - výrobci prohlížečů si přizpůsobovali definici HTML dle svých potřeb a prohlížeče podporovaly nestandardní elementy a syntaxi
- řada těchto „vylepšení“ byla následně přejímána do standardů, a některé zase v dalších verzích vyřazeny

XHTML EXTENSIBLE HYPERTEXT MARKUP LANGUAGE

- značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý konsorciem W3C
- původní předpoklad: nástupce jazyka HTML, jehož vývoj byl verzí 4.01 ukončen. (v roce 2007 nová skupina pro verzi HTML5.0)
- stále paralelně vyvíjeno (ver 1.1) a nyní se pracuje na verzi 2.0.
- XHTML 1.0 - první specifikace, jejíž cílem bylo převedení staršího jazyka HTML tak, aby vyhovoval podmínkám tvorby XML dokumentů a přitom byla zachována zpětná kompatibilita.
  - **XHTML 1.0 Strict** - používá se pro strukturovaný dokument osvobozený od formátovacích značek souvisejících s rozvržením stránky. Předpokládá se jeho užití společně s CSS, které vám umožní dosáhnout potřebných grafických efektů.
  - **XHTML 1.0 Transitional** - přechodným DTD pro webové stránky, který vám umožní používat překonané tagy. Je vhodný pro formátování stránek vytvářených pro staré prohlížeče, které nerozumí kaskádovým stylům CSS.
  - **XHTML 1.0 Frameset** - umožňuje používat zastaralé značky jako XHTML 1.0 Transitional a přidává podporu pro rámce

ROZDÍLY XHTML OPROTI HTML

- v XHTML na rozdíl od HTML musí být **všechny tagy ukončené** a to včetně nepárových jako jsou <meta>, <link>,
, <hr> nebo <img>

- zápis pak vypadá:
  - v XHTML musí být všechny tagy a jejich atributy **zapsány malými písmeny**
  - všechny **hodnoty atributů** musí být **uzavřeny do uvozovek**
  - dokument **musí začínat XML hlavičkou**. Její použití není povinné, pokud je dokument kódován v UTF-8.

  - některé tagy jako například <font> nebo , které byly označeny za zastaralé nejsou povolené.
  - většina HTML atributů (např.border, bgcolor, background, align, name, a jiné) je zakázaných. Ve striktních DTD (1.0 Strict, 1.1) jsou pravidla ještě přísnější
  - atribut target tagu a je povolený pouze v Transitional verzi. Lze jej nahradit pomocí JavaScriptu

EXTENSIBLE MARKUP LANGUAGE XML

- obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C
- umožňuje vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat
- určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů
- umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí => nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí
- prezentace dokumentu (vzhled) se definuje připojeným stylem
- pomocí různých stylů lze provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML
- nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů)
- syntaxe podstatně přísnější než HTML

VLASTNOSTI XML

- **standardní formát pro výměnu informací**

- založen na jednoduchém textu
- zpracovatelný (v případě potřeby) libovolným textovým editorem

- specifikace XML konsorcia W3C je zdarma přístupná všem
=> každý může bez problémů do svých aplikací implementovat podporu XML

- **mezinárodní podpora**

- jako znaková sada se implicitně používá ISO 10646 (Unicode).
- současně je přípustné i jiné libovolné kódování (např. windows-1250, iso-8859-2), musí však být v každém dokumentu přesně určeno

- **vysoký informační obsah**

- XML značky (tagy) vyznačují v dokumentu význam jednotlivých částí textu
=> obsahují více informací, než kdyby se používalo značkovaní zaměřené na prezentaci (vzhled) – definice písma, odsazení a podobně

- **snadná konverze do jiných formátů**

- XML samo o sobě žádné prostředky pro definici vzhledu nenabízí
- existuje několik stylových jazyků, které umožňují definovat, jak se mají jednotlivé elementy zobrazit

styl = soubor pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu
- jeden vytvořený styl lze aplikovat na mnoho dokumentů stejného typu

stejně tak můžeme na jeden dokument aplikovat několik různých stylů
- výsledek např. PostScriptový soubor, HTML kód nebo XML s obsahem původního dokumentu

KASKÁDOVÉ STYLY (CSS)

- asi nejznámější jazykový styl
- lze použít pouze pro jednoduché formátování – vhodné pro zobrazení na obrazovce

rodina jazyků XSL *(eXtensible Stylesheet Language)*
- umožňuje dokument různě upravovat a transformovat
- vybírat části dokumentu nebo generovat obsahy a rejstříky

- **automatická kontrola struktury dokumentu**

- XML neobsahuje předdefinované značky (tagy)
=> třeba definovat **vlastní značky**, které budeme používat
- značky možné (nepovinně) definovat v souboru **DTD** *(Document Type Definition)*
=> automaticky kontrolovat, zda vytvářený XML dokument odpovídá této definici

parser = program, který tyto kontroly provádí
- DTD není jediný definiční jazyk pro XML
- neobsahuje možnost kontrolovat typy dat (čísla, měnové údaje, údaje o datu a čase)!
- pro různé standardní aplikace - vytvořena schémata pro konkrétní typy dokumentů např.

DocBook - definuje struktury pro vytváření knih, článků, vědeckých publikací
- s definičními soubory DTD je dodávána sada stylů (XSL souborů) pro následné zpracování, přípravu pro tisk, převod do jiných standardních tvarů (PostScript, HTML)

- **automatická kontrola struktury dokumentu**

- XML neobsahuje předdefinované značky (tagy)
=> třeba definovat **vlastní značky**, které budeme používat
- značky možné (nepovinně) definovat v souboru **DTD** *(Document Type Definition)*
=> automaticky kontrolovat, zda vytvářený XML dokument odpovídá této definici

HYPERTEXT A ODKAZY

- XML umožňuje vytváření odkazů v rámci jednoho dokumentu i mezi dokumenty
- má možností
- možné vytvářet i vícesměrné odkazy, které spojují více dokumentů dohromady
- tvorba odkazů je popsána ve třech standardech

- **XPath** *(XML Path Language)* - umožňuje adresovat jednotlivé části dokumentu
- **XPointer** *(XML Pointer Language)* - rozšířením Xpath

- používá k určování jednotlivých částí dokumentu ve stylu: „zajímá mě první odstavec třetí kapitoly“

- **XLink** *(XML Linking Language)* - samotný jazyk pro tvorbu odkazů

- jednotlivé dokumenty se určují pomocí jejich URL adresy, za kterou lze uvést ještě XPointer pro přesnější určení části dokumentu

Součásti XML dokumentu

Tagy
Elementy
Atributy
Znakové a textové entity
CDATA
Komentáře

Procesní instrukce
Hlavička dokumentu
Každý XML dokument se skládá z elementů , které jsou do sebe navzájem vnořené.
Elementy se v textu vyznačují pomocí tagů – počáteční a ukončovací

XML Namespaces

- Jeden dokument – více typů značení

(více značek pro elementy a atributy definovaných v různých schématech)

- Seskupení příbuzných prvků XML dokumentů z důvodů snažšího rozeznávání

xmlns:prefix="URI sady značek"
xmlns:svg="http://www.w3.org/2000/svg"
xmlns:gml="http://www.opengis.net/gml"
<svg:circle cx="0" cy="0" r="100px"/>
<mapa:linie meta:titulek="Linie1">...obsah
elementu...</mapa:linie>
<elementBezNS xmlns=""/>

Aplikace XML

- **XHTML** – nástupce jazyka HTML.
- **RDF** – *Resource Description Framework* - specifikace, která umožňuje popsat metadata, např. obsah a anotace HTML stránky.
- **RSS** – rodina XML formátů, sloužící pro čtení novinek na webových stránkách
- **SMIL** – *Synchronized Multimedia Integration Language* - popisuje multimedia pomocí XML.
- **MathML** – *Mathematical Markup Language* - značkovací jazyk pro popis matematických vzorců a symbolů pro použití na webu
- **SVG** – *Scalable Vector Graphics* - jazyk pro popis dvourozměrné vektorové grafiky, statické i dynamické (animace)
- **DocBook** – sada definic dokumentů a stylů pro publikační činnost
- **Jabber** – protokol pro Instant messaging
- **SOAP** – protokol pro komunikaci mezi Webovými službami
- **OpenDocument** – Souborový formát určený pro ukládání a výměnu dokumentů vytvořených kancelářskými aplikacemi

Scalable Vector Graphics SVG

- značkovací jazyk a formát souboru, který popisuje dvojrozměrnou vektorovou grafiku pomocí XML
- výhled: základní otevřený formát pro vektorovou grafiku na Internetu
- definuje 3 základní typy grafických objektů:
  - **vektorové tvary** (vector graphic shapes) – obdélník, kružnice, elipsa, úsečka, lomená čára, mnohoúhelník a křivka)
  - **rastrové obrazy** (raster images)
  - **textové objekty**
- objekty různě seskupeny, formátovány pomocí atributů nebo stylů CSS a polohovány pomocí obecných prostorových transformací
- podporuje ořezávání objektů, alpha masking, interaktivitu, filtrování obrazu a animaci
- ne všechny SVG prohlížeče však umí všechny tyto vlastnosti

22. WEBOVÉ FORMULÁŘE

Webové formuláře jsou to, na čem částečně svět internetu stojí. Webové formuláře nejsou pouze otravná část webových stránek, která Vás zdržuje a nutí něco vyplňovat, ale díky nim fungují rozsáhlé aplikace, všemožné filtry na webových stránkách apod...
V našem systému se však dají použít na mnoho operací - nechceme po Vás, abyste se ponořovali do světa HTML kódu a norem, které musí dotyčný člověk při tvorbě znát. Nabízíme Vám webové formuláře v té nejčistší formě, v takové formě, kdy vy pouze vyberete pole, jak se má jmenovat a jaký má být jeho typ (viz dále), přidáte jej do výsledněhho formuláře a uložíte.

Proč webové formuláře

Zapomeňte nyní na ty špatné chvíle, kdy po Vás někdo chtěl, abyste byli nuceni vyplnit formulář na celou stránku monitoru, a zkuste se vžít do role někoho jiného - například zákazníka, kterému se opravdu líbí Vaše nabídka, webové stránky či zkrátka firma jako taková, a chtěl by Vás kontaktovat. Pokud je někde na cestě, v internetové kavárně, či zkrátka na jiném, než svém počítači, nemá přístup ke své emailové schránce, a to, že vy jste zveřejnil svoji emailovou adresu, mu v danou chvíli není k ničemu.
Právě v tuto chvíli se hodí webové formuláře, které Vám nabízíme v každé verzi systému Onas.cz naprosto zdarma. Jednoduše, bez použití HTML či jiného ne až tak jednoduchého způsobu, můžete sami tvořit webové formuláře, které vypadají

profesionálně, vy je můžete 100% ovládat a především - fungují.
Za každý pracovní den projde skrze systém Onas.cz několik set poptávek / objednávek, což svědčí nejen o mimořádné kvalitě a prosperitě systému Onas.cz, ale zkrátka také o tom, že TO funguje.

Je webové formuláře ovládat?
Ano, webové formuláře jsou kompletně ve Vaší správě - od výběru jednotlivých polí, která má dotyčný vyplnit, přes možnost zaškrtnutí těch polí, která musí být vyplněna, až po možnost určení, kam (= na jakou emailovou adresu) má vyplněný webový formulář dojít po odeslání. Nebojte se formuláře vyzkoušet. To nejhorší, co můžete udělat je, že formulář nebude fungovat, a tak jej prostě smažete. Popis jednotlivých polí zde není třeba, navíc veškeré funkce jsou vysvětleny přímo v nápovědě programu.
Nebojte se využít formuláře, jak jen to jde. Pokud dodržíte základní estetická pravidla, a nebudete po svých návštěvnících vyplnit každou osobní informaci, jistě Vám budou k užitku.

Formuláře
Pro vytvoření opravdové aplikace v prostředí Internetu potřebujeme do hry aktivně zapojit uživatele. Aplikace musí od uživatele získat vstupní informace, aby mu mohla poskytnout požadované údaje. Jazyk HTML proto obsahuje podporu formulářů. Součástí webové stránky se tak stanou různá vstupní pole a tlačítka — vznikne nám něco velice podobné dialogovým oknům. Uživatel do formuláře zadá data, která jsou odeslána skriptu na serveru ke zpracování. Skript získaná data nějakým způsobem zpracuje a poskytne uživateli odpověď. Vidíme, že bez formulářů by uživatel neměl příliš šanci aktivně vstoupit do dění. V této kapitole se proto podrobně seznámíme s tvorbou formulářů v jazyce HTML a se zpracováním dat z formulářů právě pomocí skriptů v PHP. Stranou nenecháme ani možnosti kontroly dat vkládaných uživatelem do formuláře.

Byrokratické zásady
Pro vložení formuláře do stránky slouží HTML element FORM. Pro správnou funkci celého formuláře jsou nezbytné dva atributy — ACTION a METHOD. Atribut ACTION určuje URL skriptu, který se použije pro zpracování dat z formuláře. V PHP je obvykle stránka s formulářem i skript v jednom adresáři
a s výhodou použijeme samotné jméno skriptu, které v tomto případě odpovídá relativnímu URL.
Pomocí atributu METHOD určujeme způsob, jakým budou data z formuláře předána zpět serveru. K dispozici jsou dvě metody — GET a POST. Metoda GET je vhodná pro přenášení kratších dat z malých formulářů, protože data formuláře se připojí na konec URL ukazujícího na obslužný skript.
Metoda POST se hodí pro odesílání větších formulářů, které obsahují mnoho dat. Údaje z formuláře jsou v tomto případě přenášeny v těle HTTP požadavku. Z hlediska psaní skriptů v PHP není mezi oběma metodami žádný rozdíl — data jsou vždy přístupná stejným způsobem. Výše zmíněné poznatky shrneme a ukážeme si, jak má vypadat základ kaž- dého formuláře:
<FORMACTION="≪URL obslužneho skriptu≫"METHOD="≪metoda≫">
≪definice formulaře≫ </FORM>

V definici samotného formuláře můžeme použít téměř libovolné elementy HTML — styly písma, tabulky, obrázky a mnoho dalšího. Kromě toho můžeme použít i speciální elementy, které slouží pro vytvoření různých vstupních polí a tlačítek.
Formuláře jsou součástí HTML již dlouho (od verze 2.0) a dnes je podporují snad všechny prohlížeče. Ve formulářích můžeme používat následující
tři elementy:
• element INPUT slouží pro definici většiny prvků — vstupních polí, polí pro zadání hesla, zaškrtávacích polí (checkboxes), přepínacích tlačítek (radio buttons), tlačítek pro odeslání a smazání formuláře, skrytých polí, odesílání souboru a tlačítek s obrázkem;
• SELECT umožňuje vytvořit seznamy, ze kterých je možno vybírat jednu i více položek;
• TEXTAREA slouží k vytvoření vstupního pole pro víceřádkový text.

Vývoj se však nezastavil a jak jistě víte, dnes aktuální verze HTML nese číslo 4.0. Právě HTML 4.0 přidalo několik novinek pro tvorbu formulářů—budeme se jim věnovat v samostatné sekci, protože je dnes ještě většina prohlížečů nepodporuje. Již nyní si můžeme říci, že stránky vyhovující standardu HTML 4.0 mohou ve formuláři navíc obsahovat elementy OPTGROUP, LABEL, FIELDSET a BUTTON.
Nyní se podíváme na to, jak můžeme pomocí tří výše zmíněných elementů INPUT, SELECT a TEXTAREA vytvářet všechny možné formulářové prvky.

Základní prvky formulářů
Nejčastěji používaným elementem ve formulářích je INPUT. Je to mimo jiné dáno tím, že může vystupovat v mnoha různých podobách v závislosti na hodnotě atributu TYPE. Tento atribut určuje typ vstupního prvku, který se ve formuláři objeví.
Všechny prvky formuláře však mají jednu shodnou vlastnost — mají své jméno. Prvku jméno přiřadíme pomocí atributu NAME. Pro nás je důležité, že
pod tímto jménem je obsah prvku dostupný v obslužném skriptu, jak si za chvíli ukážeme.
Podívejme se však na jednotlivá vstupní pole.

Profesionální formuláře

Při vytváření webovských formulářů máme poměrně velkou volnost. Není tak problém vytvořit formulář, který je pro uživatele nepřehledný, nebo naopak formulář, který je radost vyplňovat. Vytvořit přehledný formulář, zvláště pokud má obsahovat mnoho vstupních polí, je někdy umění. I přesto existuje několik zásad, jejichž dodržování nám pomůže vytvářet pro uživatele přehledné formuláře.

Přehledný formulář má jednotlivá vstupní pole jasně označena a přehledně zarovnána pod sebou. K dosažení tohoto efektu lze využít tabulku — první sloupec bude obsahovat popis vstupního pole a druhý samotné vstupní pole. Podobného efektu můžeme dosáhnout i použitím kaskádových stylů.

Celkem příjemné je, když uživateli napovíme, v jaké formě má data do jednotlivých vstupní polí zadávat. Do tabulky můžeme například přidat třetí sloupec, který bude znázorňovat ukázkově vyplněná data. Druhou možností je zobrazení ukázkových dat přímo ve vstupním poli s využitím atributu VALUE. Uživatel pak pouze přepíše předpřipravené údaje. Pro uživatele je příjemné, pokud vidí celý vyplňovaný formulář najednou na obrazovce. Pokud potřebujeme od uživatele tolik údajů, že se formulář nevejde najednou na obrazovku, je lepší vytvořit několik stránek s menšími formuláři,
které postupně od uživatele vyzvědí požadované údaje. Mnoho formulářů na Webu (např. různé registrace) vyžadují vyplnění pouze některých povinných částí formuláře. V těchto případech je vhodné povinná pole nějak odlišit — např. použitím výraznějšího písma nebo barevného písma
u popisky pole.

23. ZNAČKOVACÍ MAKROJAZYKY PRO DEFINOVÁNÍ VZHLEDU

- Každý text má obsah a formu.
- Formát (forma) webových stránek - barva a velikost písma, pozadí, zarovnání atd., prostě všechno, co nepatří do obsahu.
- jazyk HTML se vyvíjel - časem různé způsob jak formátovat text
- ⇨ dnes dva odlišné způsoby, jak v HTML třeba obarvit písmo nebo ztučnit text.

1. Starší způsob používá přímo HTML tagy.

(Například kurzíva se dělá pomocí tagů a : kurzíva). Pomocí tagů se některé věci nedají udělat.

2. Novější způsob -- CSS styly

- používá tag <style> a obecný atribut "style", kterému se přiřazuje nějaká definice.
- Je to složitější, ale užitečnější a všeobecné.

- CSS - Cascading Style Sheets(kaskádové styly) – vznikly kolem roku 1997
- kolekce metod pro grafickou úpravu webových stránek.
- kaskádové, protože se na sebe mohou vrstvit definice stylu - platí jen ta poslední
  - Už je na světě návrh CSS 3, vylepšené a složitější formy stylů

NÁSTIN MOŽNOSTÍ CSS

- Nastavit libovolnou a přesnou velikost písma, prokládání, kapitálky
- Udělat odsazení prvního řádku odstavce, zvětšit řádkování
- Zrušit nebo zvětšit prázdný prostor po odstavci
- Automaticky formátovat nadpisy (například je všechny udělat zelené)
- Zvýrazňovat odkazy po přejetí myší
- Udělat automaticky grafické odrážky
- Určité části textu zneviditelnit, zprůhlednit nebo nezobrazit
- Předefinovat grafický význam běžných tagů (například všechno, co je kurzívou, udělat i tučně)
- Nastavit pozadí čehokoliv, stránky, tabulky ale třeba i odstavce; pozadí se nemusí opakovat a může mít přesnou pozici!
- Umístit nějaký objekt (třeba kus textu) kamkoliv do stránky, může se to i překrývat
- Přidat k čemukoli rolovací lišty, oříznout to, orámovat, nastavit okraje
- V kombinaci se skripty je dnes CSS nejmocnější zbraň pro "rozhýbání" stránek.
- Hlavní význam CSS spočívá v tom, že fungují hodně automaticky, přičemž se vzhled celého webu deklaruje jedním souborem.

TROJÍ POUŽITÍ CSS

- Styl se může nadeklarovat třemi způsoby.
  - **přímý styl**
- přímo v textu zdroje u formátovaného elementu pomocí **atributu style="..."**
- nešikovné, ale občas se to používá.
  - **2.stylopis (angl. "stylesheet")**

- jakýsi seznam stylů - je v něm obecně napsáno, co má být jak zformátováno, například že nadpisy mají být zelené.
- do stránky se píše mezi **tagy <style> a </style>**
  - **externí stylopis**
- soubor *.css, na který se stránka odkazuje **tagem <link>**
- v souboru umístěný stylopis.
- v**ýhoda:** na jeden takový soubor se dá nalinkovat mnoho stránek => všechny vypadají podobně

Přímý zápis

Chci udělat odstavec červeným písmem pomocí CSS.

- do zdroje se napíše tato deklarace odstavce:
  - Tento odstavec bude červený.
  - **Vysvětlení:** je značka vymezující odstavec; z anglického paragraph. Atribut "style" je obecný atribut použitelný u každého prvku. Color znamená barva a red je červená.

Stylopis

- do hlavičky dokumentu se napíše stylopis uzavřený mezi tagy <style></style>
  - <style>
  - p {color: red}
  - </style>
- do těla stránky se mohou psát odstavce:
  - Tento odstavec bude červený.
  - Tento mimochodem také, protože červené budou všechny.

Externí CSS souborem

- vytvoří se soubor, který se pojmenuje třeba styly.css. V něm bude pouze tento text:
  - p {color: red}
- do hlavičky html dokumentu, který chci stylem ovlivnit - napsat odkaz na tento soubor:
  - <link rel="stylesheet" type="text/css" href="styly.css">
- V těle dokumentu pak budou opět všechny odstavce červené

SYNTAXE

- CSS nejsou součástí HTML, a tak se zapisují zcela jiným způsobem

Přímý styl:	<tag style="zápis vlastností">stylovaný element</tag>
Ve stylopisu:	<style> tag {zápis vlastností} 2.tag {zápis vlastností} </style>
Zápis vlastností zjednodušeně:	vlastnost: hodnota; 2.vlastnost: 2.hodnota
Zápis vlastností obecně:	vlastnost: hodnota [, hodnota2] [; další zápis vlastností]

- **!** kde jsou uvozovky, dvojtečky, složené závorky, středníky a čárky
- pokud si některé znaménko popletete, nebude to pravděpodobně fungovat.

Příklad správného zápisu:

- H2 je **selektor** = jméno tagu, jehož formátování se mění
- {} složené závorky - vymezují deklaraci formátu onoho selektoru
- color je **vlastnost** a blue jeho **hodnota** (barva: modrá), vlastnost a hodnota jsou odděleny dvojtečkou a mezerou
- font-style je další vlastnost a italic je její hodnota (řez-fontu: kurzíva)
- dvě deklarace se oddělují středníkem.

- mezery a konce řádků - nehrají roli => mohou se přidávat a vypouštět
- velikost písmen nehraje roli
- Hodnoty (překlepy) které prohlížeč nezná, ignoruje
- komentáře ve stylopisech - /* a */

PSEUDOTŘÍDY

- pomocí stylopisů se dají formátovat libovolné HTML tagy, ne pouze nadpisy.

- tag A (jako jediný) má pseudotřídy (různé stavy) - umožňují různé zobrazení podle toho, zda už je odkaz navštívený nebo zda po něm jede myš.
- mezi a a dvojtečkou není mezera!

```
<style type="text/css">
a        {text-decoration: none}
a:link    {color: green}
a:visited {color: navy}
a:active  {color: black}
a:hover   {color: red; text-decoration: underline}
</style>
```

* text-decoration: none znamená, že odkazy nebudou podtrhávané
* a:link znamená nenavštívený odkaz (bude zelený)
* a:visited je už navštívený (tmavě modrý)
* a:active je ten, na který se zrovna kliklo (černý), nebo ten, po kterém jede tabulátor
* a: hover je ten, přes který se jede myší (červený podtržený)
* text-decoration: underline znamená podtržení.

Externí stylopis

- myšlenka CSS počítá s tím, že se definice stylů umisťuje do externího souboru, který se pak už jenom odkazuje.
- soubor pokus.css.

Záludnosti

- Externí stylopis v samostatném souboru nezačínejte tagem <style>.
- Externí css soubor správně začíná rovnou první deklarací bez jakýchkoli hlaviček, např. body {color: black}
- Zkoušíte-li nové externí stylopisy tak, že je uložíte a občerstvíte linkovanou stránku, může se stát, že se neprojeví změna.
  - <=> starší prohlížeče si totiž drží v cache paměti minulou verzi stylopisu

- **Import**

Namísto tagu <link rel="stylesheet" href="pokus.css"> lze použít zápis

```
<style type="text/css">
@import url('pokus.css');
</style>
```

- v prohlížeči Internet Explorer nečeká prohlížeč s vykreslováním stránky na načtení celého stylu, jako v případě zápisu přes <link>
- při načítání jakoby "poskakuje"
- je rychlejší (a uživateli se alespoň něco objeví velmi brzo)

**Tagy a **

- někdy je ale potřeba zformátovat kus textu, který není vymezen žádným konkrétním tagem.
- vloží se nový tag (proč ne?).
- párový tag - zahrnuje-li formátovaná oblast více odstavců
- - v rámci jednoho odstavce

```
<body>
... <!--normální odstavce -->
<div style="color: maroon">
... <!-- mnoho různých odstavců, všechny budou hnědé -->
</div>
...<!-- a už je to zase normál -->
```

Vypadá to takhle. A druhý příklad:

```
<p>Normální text a <span style="font-style: italic">text kurzívou</span> a zase normální text.</p>
```

Normální text a *text kurzívou* a zase normální text.

TŘÍDY, IDENTIFIKÁTORY A SLOŽENÉ DEKLARACE

- příkladem vlastního stylu může být podtitul. (Nepatří do nadpisu a přece by měl být formátován odlišně než normální text.)
- dá se formátovat přímo - aby byl ve všech dokumentech stejný - dobré nadefinovat jej jako styl - HTML nemá pro podtitul žádný tag <podtitul> => musíme si pomoci jinak
- vytvořím *třídu* s názvem podtitul, ve stylopisu mu nadefinuji vlastnosti (třeba tučnost, zarovnání na střed) a u daného textu jenom řeknu, že patří do třídy podtitul
  - <style>
  - **.**podtitul { text-align: center; font-weight: bold; text-decoration: overline}

- /* zarovnání na střed, tučné písmo a nadtržení*/
- </style>

- v těle dokumentu to vypadá takhle:

Text podtitulu

- v prohlížeči potom takhle:

Text podtitulu

- text uvnitř "zaklasovaného" elementu se bude formátovat podle definice ve stylopisu.
- tečka na začátku deklarace ve stylopisu - vyjadřuje, že deklarace se nebude týkat html tagu, ale třídy.
- Atribut class (třída) se může použít u libovolného elementu (tagu). Symbolicky:
  - <tag class="jméno_třídy">
- element se stejnou class se v dokumentu může vyskytovat mnohokrát
- takto je možné vytvořit si mnoho vlastních tříd - stylů

IDENTIFIKÁTOR

- pro jednoznačný popis nějakého elementu - univerzální atribut **ID** (zejména pro potřeby skriptů).
- ve stylopisu lze přiřadit nějaká deklarace
- na rozdíl od třídy nezačíná tečkou, ale dvojkřížkem **#**
- **!** V těle dokumentu by se element s jedním identifikátorem měl vyskytovat jenom 1.

=> předchozí příklad s použitím identifikátoru namísto třídy
#podtitul { text-align: center; font-weight: bold; text-decoration: overline}
a v těle by se odstavci přiřadila identifikace atributem id:
Text podtitulu
identifikátor id se z hlediska CSS chová stejně jako třída class.
rozdíly jsou právě jen ve skriptech a v parsování dokumentu.

SLOŽENÉ DEKLARACE

Hromadná deklarace

- Stylopisy umožňují nadeklarovat vlastnosti pro více elementů najednou.
- H1**,** H2**,** H3 {color: green}
- obarví všechny nadpisy první, druhé i třetí úrovně na zeleno.
- pokud zadávám mnoho stejných vlastností pro mnoho elementů.
- **!** čárka mezi selektory - Kdyby tam nebyla, šlo by o něco jiného, totiž o kontextový selektor.

Kontextová deklarace

- **H3 A** {font-style: italic}
- deklarace udělá kurzívou všechny odkazy **uvnitř** nadpisů třetí úrovně (elementy A uvnitř elementu H3).
- <h3>Obyčejný text nadpisu s <a>*odkazem kurzívou*</h3>
- Obyčejný odstavec s <a>obyčejným odkazem
- Odkazy v obyčejných odstavcích to nijak neovlivní, stejně tak obyčejný text trojkového nadpisu.
- **!** Zápisy selektorů kontextového zápisu jsou odděleny pouze mezerou.
- výborně použitelnou vlastnost CSS

Skládání stylů

- na jeden element může navrstvit mnoho různých deklarací (proto styly kaskádové), někdy i protichůdných.
- převládne zpravidla poslední deklarace.

Pseudoelementy

- ve specifikaci CSS se vyskytují pseudoelementy
  - p:first-line {color: blue} /*první řádek
  - p:first-letter {font-size: 200%} /*první písmeno
- měly by způsobit, že odstavce budou mít první řádek modrý a první písmeno dvakrát větší.

Více tříd pro jeden prvek

- když je třeba aby prvek přebral formátování dvou tříd, pak je prostě stačí uvést obě v atributu class a oddělit je mezerou.

Styl může například vypadat takto:
<style>
.zlutepozadi {background-color: yellow;}
.vlevo {float: left; width: 150px;}
</style>
Prvek dostane obě třídy:

obsah prvku

- obsah prvku bude odplavaný doleva a bude mít žluté pozadí.
- vezme prostě vlastnosti obou tříd.
- Ve většině případů je samozřejmě lepší spojit deklaraci do jedné třídy, ale vyskytují se stránky, kde ostatní prvky chcete oclassovat odděleně.

24. WEBOVÉ PROHLÍŽEČE

Webový prohlížeč (též **browser** [brauzr]) je počítačový program, který slouží k prohlížení World Wide Webu (WWW). Program umožňuje komunikaci s HTTP serverem a zpracování přijatého kódu (HTML, XHTML, XML apod.), který podle daných standardů zformátuje a zobrazí webovou stránku. Textové prohlížeče zobrazují stránky jako text, obvykle velmi jednoduše formátovaný. Grafické prohlížeče umožňují složitější formátování stránky včetně zobrazení obrázků. Pro zobrazení některých zvláštních součástí stránky, jako jsou Flash animace nebo Java applety, je třeba prohlížeč doplnit o specializované zásuvné moduly. Mezi nejznámější webové prohlížeče patří grafické (seřazeny podle počtu uživatelů) Windows Internet Explorer, Mozilla Firefox, Safari, Google Chrome, Opera a textové Links a Lynx.
V čem je problém?
Začínající Internetoví autoři nejsou psychicky připraveni na smutnou pravdu:

- **Stejná stránka se v různých prohlížečích může zobrazit odlišně.**

Jako autor stránek na webu nevím, jaký prohlížeč bude můj čtenář používat. Můžu to jenom tušit. Proto nemá cenu specializovat se na jeden typ prohlížeče a stránky ladit jen pro něj. Obvykle je potřeba, aby stránky vypadaly stejně ve všech hlavních prohlížečích.

Svoje stránky byste vždy měli vyzkoušet alespoň na těchto prohlížečích:

- Internet Explorer 7 nebo 8
- Firefox
- Opera 9

Znamená to, že si Firefox a Operu musíte nainstalovat. Není to tak těžké (napište mi, pokud potřebujete návod). Explorer je na každých Windows už od začátku. Ostatní prohlížeče mít nemusíte, protože se chovají velmi podobně jako výše zmíněné.
Tolik ve stručnosti, níže proberu nejčastější prohlížeče a jejich specifika podrobněji. Ale napřed krátký historický úvod.
Proč se prohlížeče liší
Internet se zpočátku vyvíjel neuvěřitelně rychle. Podobně dynamicky se přetvářel jazyk HTML, tak aby umožňoval zařazovat do stránek nové a nové věci.
Starší prohlížeče neumožňovaly stejné zobrazení stránek jako prohlížeče moderní, protože v době vzniku starších prohlížečů se prostě nevědělo, jak se budou stránky psát za pár let. Něco samozřejmě zůstalo stejné, ale staré prohlížeče neumějí zobrazovat nové styly, skripty a jiné vychytávky.
Kromě oficiální verze jazyka HTML existovaly různé hybridní formy HTML a rozšíření HTML. Výrobci prohlížečů (zejména Microsoft) se snažili do svých prohlížečů zabudovat podporu nestandardních věcí, které byly teprve v návrhu nebo které si sami vymysleli. Čili **interpretace jazyka HTML je závislá na prohlížeči**, který používá čtenář (klient).
V poslední době se to trochu uklidnilo. Microsoft do Internet Exploreru 6 (který je nyní na podzm 2007 stále dominantní) zapracoval podporu některých důležitých standardů. Konkurenční prohlížeče Mozilla a Opera jsou na tom ohledně standardů ještě lépe.

JÁDRA

Trident (též známé jako **MSHTML**) je renderovací jádro používané v prohlížeči Windows Internet Explorer pro Microsoft Windows. Jeho první verze se objevila v říjnu 1997 v Internet Exploreru 4.0 a bylo postupně vylepšováno. Aktuální verze 5 je součástí Windows Internet Exploreru 7.0. Jádro je známé svou slabší podporou webových standardů, některých nestandardních rozšíření a slabší bezpečností (platí pro starší verze).
Díky své integraci do operačního systému Windows je renderovací jádro využíváno v řadě komponent systému a využívá ho i řada instalovaných aplikací. Příkladem může být Windows Explorer, nápověda systému Windows či kancelářský balík Microsoft Office.
Gecko je open source renderovací jádro používané produkty Mozilla pro vykreslování webových stránek. Je napsáno v programovacím jazyce C++ a licencován pod trojlicencí MPL/GPL/LGPL. Díky licenci a podpoře webových standardů je renderovací jádro používáno v řadě jiných prohlížečů jako například Flock, K-Meleon či Epiphany. Jádro bylo původně vytvořeno firmou Netscape Communications Corporation, ale nyní je vyvíjeno Mozilla Corporation.
Gecko díky svému bohatému API nenabízí pouze možnost renderování webových stránek, ale slouží též vykreslování grafického rozhraní (XUL), které využívá Firefox či Thunderbird. Jedná se o multiplatformní jádro, takže je k dispozici pro řadu platforem jako Microsoft Windows, Linux, Mac OS X a další.

Gecko je po renderovacím jádře Trident, které používá webový prohlížeč Windows Internet Explorer, druhé nejpoužívanější renderovací jádro. Aktuální verze jádra je 1.9.

Prohlížeče postavené na jádře Trident

- Windows Internet Explorer
- Maxthon (dříve MyIE, ve verzi 1.x umožňuje používat také jádro Gecko)
- Netscape Browser 8.0 (tento prohlížeč umožňuje používat jak jádro Gecko, tak jádro Trident)
- 32bit Web Browser
- AmiWeb
- Fast Browser
- NeoPlanet
- Smart Explorer
- AOL Explorer
- Avant Browser

… a další

Prohlížeče postavené na jádře Gecko

- Mozilla Firefox (s rozšířením IE Tab může používat i jádro Trident)
- Mozilla Suite
- SeaMonkey
- Epiphany
- Galeon
- Netscape Navigator (od verze 6.0)
- Camino
- K-Meleon
- Flock
- SkipStone
- My Internet Browser
- Maxthon (ve verzi 1.x)

… a další

Prohlížeče postavené na jádře WebKit

- Konqueror
- Safari
- OmniWeb
- Google Chrome
- Maxthon (verze 3, navíc umožňuje používat také jádro Trident)
- Arora
- Epiphany (od verze 2.28)

Ostatní grafické prohlížeče

- Amaya
- Arachne
- Dillo
- Links2
- Lobo
- Opera

Textové prohlížeče

- Elinks
- Links
- Lynx

SOUČASNÉ PROHLÍŽEČE

V roce 2008 lze potkat tyto prohlížeče:

Typ, verze	Rozšířenost	Vlastnosti
Internet Explorer 6 a 7	45 % všech uživatelů Windows, na Linuxu není	Dobrý prohlížeč, který je v současnosti de facto standardem. Verze 5, 5.5, 6, 7 a 8 se liší pouze v detailech.
Mozilla, FireFox a další klony Mozilly	45 % uživatelů Windows, spousta linuxáků	Velmi dobrý program označovaný obecně jako Mozilla. Vykreslovací jádro Gecko. Patří pod to i FireFox.
Opera 9	5 % uživatelů	Velmi dobrý prohlížeč se zajímavým ovládáním.
Google Chrome	5 % uživatelů	Prohlížeč vyvinutý Googlem, používá vykreslovací jádro Webkit. Velmi dobrý.
Konqueror	30 % linuxáků	Dobrý prohlížeč s podporou lecčeho
Safari	asi 90 % mac-uživatelů	Nejlepší prohlížeč platformy Mac, vykreslovací jádro Webkit.
Prohlížeče mobilních zařízení		Velmi různá podpora HTML, spíše špatná podpora CSS stylů.

Následují podrobnosti o důležitějších prohlížečích.

INTERNET EXPLORER (ZKRATKA IE NEBO TAKÉ MSIE)

Protože nejrozšířenějším prohlížečem je Internet Explorer, hodně lidí stránky ladí pro něj a v ostatních prohlížečích jenom trochu kontrolují, jestli to funguje. Dá se to tak dělat.
Dobrá verze Internet Exploreru je číslo 6. Pětky se také dají skousnout, ale už se naštěstí moc nevyskytují. Liší se pouze v detailech (zejména interpretace blokových CSS vlastností, rozšíření CSS a JavaScriptu). Čtvrtá verze IE se dnes prakticky nevyskytuje (méně než promile v roce 2005). Sedmá verze byla vydána v říjnu 2006, osmá v dubnu 2009. Sedmička a osmička jsou v létě 2009 nejrozšířenější.
Do jednoho počítače se dá pro testovací účely nainstalovat více verzí Internet Exploreru. Dlouho se to ale nevědělo (Microsoft říkal, že se ty verze navzájem nesnesou). Ale existují balíčky umožňující instalaci verzí 5.0 a 5.5 vedle dříve nainstalovaného IE 6.0. Od roku 2009, kdy se tyto staré verze prohlížečů vyskytují málo, už nemá smysl takové simulace dělat.
Internet Explorer 6 byla docela dobrá verze. V jednu chvíli ho mělo asi 80 % lidí, na konci roku 2008 ho mělo asi 20% uživatelů. To znamená, že verze 6 reálně mezi uživateli nezmizí asi tak do konce roku 2011 a je otázka, jestli ho lze již ignorovat. Verzi 5.0 má asi 0,1 % a verzi 5.5 asi 0,1 % uživatelů, takže pětky už při výrobě stránek lze ignorovat.
Do starších počítačů si na zkoušku můžete nainstalovat také MSIE 7.0. Výhoda je, že se dá odinstalovat. Výhodou IE 8 je, že prý dovoluje simulovat velké množství starších verzí (osobně jsem to nezkoušel).

MOZILLA, FIREFOX, GECKO

Možná nejlepší dnešní prohlížeč (2009) je Firefox na jádru zvaném Gecko.
Gecko podporuje CSS ve verzi 2, DOM2 a Javascript lépe než Microsoft Explorer. Drží se standardů. Umí dobře zobrazovat XML. Blokové elementy vykresluje trochu jinak než Internet Explorer (tedy správně podle specifikace). Zdrojáky a instalačky pro všechny platformy lze najít na www.mozilla.org. Uživatelé si mohou svou Mozillu přizpůsobit pomocí jazyka XUL.
Existují různé distribuce Mozilly, nejpopulárnější se jmenuje **FireFox**. Firefox existuje i v češtině. (Projekt CZilla si klade za cíl lokalizovat do češtiny všechny produkty z Mozilla.org). Ke stažení je Firefox třeba na české stránce Firefoxu. Pokud někde potkáte Netscape 6 nebo 7, je to také klon Mozilly.
Mozilla se dá spustit jak na Windows, tak na Linuxu i na Macu. Vykreslovací jádro prohlížeče Mozilla se jmenuje Gecko (čte se to gecko, géčko nebo geko).

OPERA 9

Opera je velmi dobrý prohlížeč, stáhněte si na www.operasoftware.com. Verze 9 je skvělá. Má velmi dobré ovládání (gesta myší jsou fantastická), je rychlá. Stránky zobrazuje hodně podobně jako Explorer, ale umí i standardní vykreslování. Některé věci v Opeře fungují trochu hůře. Od verze 8.5 je Opera zdarma a bez reklamních bannerů.

GOOGLE CHROME

Google chrome se objevil na podzim 2008 celkem překvapivě. Vydala ho firma Google. Chrome je založen na vykreslovacím jádru Webkit, které používá také Safari. Na konci roku 2008 ho má asi 1,7 % uživatelů, ale jeho podíl rychle roste. Na stránkách Googlu si Chrome můžete stáhnout. Já mám Chrome rád.

NETSCAPE (NN)

Nejhorším prohlížečem minulosti je Netscape 4. Dnes jej používají pouze nepřátelští webmasteři a kritici, když chtějí dokázat, že máte špatně udělané stránky. Chybuje a padá. Lze na něj zcela zanevřít (psáno 2004) a nebrat jej v úvahu.
Co je Netscape 6, 7 a 8: Je to klon Mozilly, tedy něco jako Firefox. Netscape pátou verzi prohlížeče nechal vyvíjet nezávislou skupinou jako open-source. Tak vznikla Mozilla. Vývojovou verzi Mozilly přejal Netscape a udělal z ní prohlížeč Netscape 6. Jádrem je vykreslovací program Gecko. Netscape 6 a 4 jsou ke stažení na www.netscape.com, kde jsou též archivní starší verze (např. verze 3, která vůbec nepodporuje styly).
Když se dnes mezi českými webmastery mluví o Netscape, myslí se tím verze 4. Verzi 6 a 7 se říká Mozilla, protože to je Mozilla. V anglosaském světě v tom mají trochu větší zmatek a jako Netscape označují i šestou a sedmou verzi.

PROHLÍŽEČE PRO MAC

- Nejlepší prohlížeč na Macu je **Safari**, jde o nativní systémový prohlížeč.
- Existuje Internet Explorer 5 pro Mac, který ale s IE na Windows nemá moc společného.
- Existují portace Mozilly jako třeba FireFox.
- Camino je postaven na jádru Mozilla.

ALTERNATIVNÍ PROHLÍŽEČE

- Linuxový **Konqueror** je fajn, má vykreslovací jádro příbuzné k macovskému Safari. Je vyvíjen pro prostředí KDE.
- **Textové** prohlížeče **Links** a **Lynx** jsou velmi oblíbené mezi uživateli Unixů a Linuxů. Links je český a relativně nový, Lynx je starý nečeský.
- Některé programy tvářící se jako autonomní prohlížeče (třeba MyIE, Avant Browser) ve skutečnosti používají vykreslovací jádro z Internet Exploreru nebo z Mozilly (Gecko). Některé z nich jsou opravdu zajímavé, například Avant Browser má taby a gesta myší.

KOMPATIBILITA

Kompatibilita nebo řekněme podobnost prohlížečů popisuje rozdíly v jejich zobrazovacích schopnostech. Jednoduše řečeno, jedna stránka může vypadat v různých prohlížečích různě.

V tomto článku nebudu popisovat rozdíly mezi prohlížeči, je jich nekonečná řada, pouze se pokusím samotný problém trochu představit.

PROHLÍŽEČE

Existuje mnoho prohlížečů, některé jsou hojně užívané, jiné méně. Ale i ty nejužívanější (Internet Explorer, Mozilla, Opera) jsou v mnohém rozdílné. Navíc ke každému prohlížeči existuje mnoho verzí, např.

- Microsoft Internet Explorer 3, 4, 5, 6 (nejčastěji 5.5 a 6)
- Netscape (Mozzila) 2, 3, 4, 5, 6, 7 (nejčastěji 6, 7 - Mozzilla), Firefox
- Opera 5,6,7

Čtyřkové verze všech prohlížečů už snad vyhynuly, ale pořád se najdou.
Co to vše znamená? Vytvořit plošně zobrazitelnou stránku se neobejde bez menších problémů.

WEBOVÉ STANDARDY

Existují standardy pro HTML, XHTML, XML, CSS a vůbec všechny webové formáty, ty určuje W3.org.
Kdyby všechny prohlížeče dodržovali tyto standardy, většina problémů webdesignu by vymizela. Problém je, že např. Microsoft a jeho prohlížeč tyto standardy respektují neúplně a navíc ještě vytváří nové.

Jak to vyřešit

- Kašlat na ostatní a dělat si stránky pro svůj oblíbený prohlížeč
- Nepoužívat sporné prvky výsledkem bude chudší, ale funkční stránka
- A nebo pomocí skriptu zjistit verzi prohlížeče a podle toho zařídit stránky. Stránky budou velké, ale všude funkční

První metoda je nejjednodušší. Mně se zdá nejlepší ta druhá.

Závěr

Vytvořit funkční webové stránky není vůbec jednoduché a pokud to budete s webdesignem myslet vážně, budete potřebovat hned několik prohlížečů a nejlépe i více verzí, abyste mohli výsledek průběžně ověřovat. S přibývající zkušeností poznáte, co se smí a co ne.

25. OPTIMALIZACE WEBSITU

META TAGY

Meta-tagy jsou skryté tagy v hlavičce (head) každé webové stránky (samozřejmě záleží jen na Vás jestli je přidáte). Doporučujeme meta-tagy přidávat do každé vaší webové stránky a alespoň "description" (popis) přizpůsobovat zvlášť daným webovým stránkám...

<META> informace umísťované v HTML kódu mohou být důležité a užitečné. Jen vědět co s nimi....

[author] [classification] [copyright] [description] [distribution] [formatter] [generator] [keywords] [rating] [resource-type] [robots] [HTTP-EQUIV] [a jiné ?]

Začněme tím nejdůležitějším.

Umístění <META> je jediné možné - META patří mezi <HEAD> a </HEAD>. Pokud umístíte META kamkoliv jinam, je to totéž jako když nic takového neuděláte.

Použití META záleží hlavně na programech, které se rozhodnou je použít. Jedním z nejdůležitějších jsou prohledávací stroje. Druhé nejdůležitější je použití META pro určení kódové stránky.

author

<META name="author" content="označení autora">

Určení autora stránek. Použití závisí na způsobu využití některým z programů.

classification

<META name="classification" content="určení">

Objevuje se pouze u Netscape Gold a těžko říci k čemu slouží.

copyright

<META name="copyright" content="text určující (c) dané stránky">

Umožňuje určit (c) dané stránky, ten se také zpravidla objeví v indexovacím stroji (pokud tento umí zpracovat a využít tyto informace).

description

<META name="description" content="odstavec popisující místo/stránku">

Ke své stránce doplníte kompletní popis. Vyhledávací stroje použití tuto informace pro určení popisu místa.

distribution

<META name="distribution" content="global|local">

Umožňuje sdělit vyhledávacímu stroji zda stránka je globálním vstupním místem do WWW místa nebo zda jde o některou z lokálních stránek.

formatter

<META name="formatter" content="označení formátovače">

Určuje formátovací program či prostředek, který se postaral o naformátování výsledného HTML. Používá jej zejména Frontpage.

generator

<META name="generator" content="označení generátora">

Ve většině případů slouží k určení jména a čísla verze programu/skriptu/kouzelníka použitého pro vygenerování stránek

keywords

<META name="keywords" content="klíčová slova oddělená čárkami">

Klíčová slova mající vztah ke stránce/dokumentu. Vyhledávací stroje je používají často pro zatřídění dokumentu.

rating

<META name="rating" content="určení ratingu">

Určuje RSACi či PICS informace ochrany před nevhodným obsahem.

resource-type

<META name="resource-type" content="document">

Uváděná syntaxe je prozatím jediná možná. Tento META se vyplatí uvádět pro vyhledávací stroje indexující Web má tento META význam pokynu k tomu že dokument je "dokument". Pokud není META uvedeno, není jisté že prohledávací stroj se rozhodne indexovat.

robots

<META name="robots" content="ALL|INDEX|FOLLOW|NOFOLLOW|NOINDEX">

Umožňuje sdělit vyhledávacímu robotu co má dělat se stránkou. ALL = zpracovat všechno, INDEX = indexovat, NOINDEX = neindexovat, NOFOLLOW = nenásledovat linky.

Často se používá "NOINDEX,FOLLOW" kombinace.

HTTP-EQUIV

HTTP-EQUIV je zvláštní kategorie META informací protože slouží spíše serveru obsluhujícím HTTP server a dodávajícím stránky prohlížeči. V zásadě HTTP-EQUIV bude převedeno na příslušné pole hlavičky poslané po GET/HEAD požadavku.

HTTP hlavičky jsou definovány v RFC1945 a v RFC2068 pokud chcete bližší informace o významu jednotlivých variant.

<META HTTP-EQUIV="Expires" CONTENT="mon, 01 Dec 1997 01:00:00 GMT">

určuje "expiraci" neboli vypršení platnosti stránky. Objeví se jako Expires: Mon, 01 Dec 1997 01:00:00 GMT" v hlavičce stránky. Datum musí být v RFC850 formátu a pochopitelně v GMT podobě. Pokud se objeví content="0" je to totéž jako určení "okamžitě".

Expires neurčuje zda bude či nebude dokument uložen v lokálním cache, ale zda další "návštěva" stránky bude vyžádáním stránky z cache nebo ze skutečného WWW místa.

GEOURL

- webová služba GeoURL umožňuje přiřadit webovým stránkám souřadnice domova, školy, firmy nebo hlavního stanu.
  - ⇨ zanese web do mezinárodní databáze fyzických umístění webů a najděte své sousedy.
- nezbytnost je znalost zeměpisné šířky a délky fyzického umístění objektu jež je předmětem webu.
- Fyzickým umístěním uvažujme u organizace její sídlo, u osobních stránek pak pravděpodobně hlavní působiště jejich majitele.

Metadata – Dublin Core

do hlavičky úvodní stránky webu doplňte (<head> ... </head>) následující dva řádky:

<meta name="ICBM" content="XX.XXXXX, YY.YYYY">

<meta name="DC.title" content="Jmeno Webu">

XX.XXXXX >> zeměpisnou šířkou

YY.YYYYY >> zeměpisnou délkou

Jmeno webu >> jménem webu – bude se zobrazovat ve výsledcích hledání

POZOR: hodnoty mají být zadány v úhlových stupních jako desetinné číslo.

Řada služeb (Mapy.cz, …) vrací výsledky ve stupních (DEG), minutách (M) a sekundách(S): DEG° M′ S″.

=> **alpha[°] = DEG + S/60 + M/3600**

Zaindexování do databáze fyzicky umístěných webů.

Na serveru http://geourl.org/ping vyplňte úplnou adresu webu a potvrďte ji.

Server odpoví, že do deseti minut bude web v databázi.

Krom toho vypíše HTML kód, jehož přidáním do webu získáte ikonku pro snadné vyvolání výpisu webových sousedů.

zkuste vyvolat stránku http://geourl.org/near/?p=http://nasweb.cz.

Měl by se vrátit seznam webů seřazených od nejbližších k nejvzdálenějším v okruhu 500mil s pomyslným středem tvořeným vaším webem.

Můžete přepnout i na alternativní zobrazení, kde jsou výsledky rozděleny do světových stran.

U každého webu je pak informace o vzdálenosti a odkaz na mapu

JAK STRÁNKY DOSTAT NA SERVER

Pokud máte na disku vytvořené stránky -- soubory typu html -- stojíte před problémem, jak je dostat na server a zpřístupnit je tím světu. Podle mých zkušeností je to úkol těžký, takže to při počátečním neúspěchu nevzdávejte.

Kam kopírovat

Logika říká, že ony soubory stačí nějakým způsobem někam zkopírovat. Hlavní problém je kam, na jaký server. Podle toho, kolik máte peněz:

* můžete hostovat zdarma
* nebo si někde objednat placený hosting (většinou i s doménou druhého řádu)

Zadarmo

Jsou dva způsoby, jak publikovat na webu zadarmo.

1. Buďto využijete služeb veřejných serverů -- freehostingů. První byl Geocities (u nás MujWeb). Nyní takových serverů existují desítky. Mám odzkoušené servery www.sweb.cz, www.webpark.cz a www.volny.cz, www.webzdarma.cz a další. Někde se musíte smířit s tím, že vám tam možná budou skákat cizí reklamy.
2. Nebo máte nějakého známého nebo firmu, který má server a nechá vás na něm publikovat, případně využít serveru školy nebo zaměstnavatele (to bych třeba já nedělal, vzhledem k tomu, jak často školy a zaměstnavatele měním: pořád bych musel měnit adresu).

Nejčastěji se uplatňuje první varianta, publikování na freewebu (bezplatném hostingu). V takové situaci je nevýhoda, že máte ne zcela hezkou adresu a na stránkách mohou být reklamy poskytovatele.

Za peníze

Pokud můžete obětovat pár korun, doporučuji najít si nějaký placený hosting. Chcete-li mít doménu druhé úrovně, musíte tuto možnost zvolit vždy. Zpočátku vám bude stačit několik málo mega prostoru a statické stránky, to se dá sehnat za pár stovek ročně. Doména na rok stojí dalších pár stovek. Poskytovatelé hostingu vždy umožňují kopírování pomocí FTP (viz níže), což je dobře.

Jak založit webový prostor zdarma na bezplatném hostingu:

* Vlezte na hlavní stránku freehostingu. Na portálech je to odkaz typu "www zdarma. Seznam českých freehostingů.
* Najděte nějaký odkaz na registraci (někdy se to dělá dohromady s mailem)
* Zadejte údaje, zvolte uživatelské jméno a heslo. Některé hostingy (například sweb) vyžadují odeslání kontrolní esemesky.
* Zapamatujte si pečlivě všechny parametry (nejlépe je zapsat si to nebo vytisknout). V to patří:
  - uživatelské jméno (aneb login)
  - adresa stránek (http://něco, obvykle obsahuje uživatelské jméno)
  - heslo
  - adresa ftp serveru
  - jméno startovního souboru (většinou index.html, případně se to zkusí).
* Zkuste tam poslat stránky (návod níže) a podívat se na to.

Freehostingy nabízejí dva zásadně odlišné způsoby aktualizace (publikování) stránek:

1. aktualizace pomocí FTP,
2. aktualizace pomocí grafického webového rozhraní.

Na některých serverech nabízejí oboje, na jiných jenom jeden způsob. Já doporučuji používat spíše aktualizaci přes FTP, protože grafické rozhraní je pro správu většího počtu stránek těžkopádné. Oba způsoby popisuji níže.

FTP

Při používání FTP pro nahrávání souborů na server vždy potřebujete znát

* FTP adresu na svůj server (většinou ftp.domena.cz, např. ftp.volny.cz)
* login (uživatelské jméno)
* heslo
* a někdy navíc cestu (jméno složky)

To všechno by vám měl sdělit správce serveru nebo byste si to měli zjistit při registraci.

Protokol "File Transfer Protocol" (zkratka FTP) se používá na kopírování stránek na vzdálený server (nebo pro stahování, to je teď jedno). FTP příkazy se dají ťukat ručně (Start > Spustit > FTP), to ale není nic pro smrtelníky. Častěji se musí použít program, který se nazývá FTP klient.

VALIDACE WEBU

Ptáte se, co je to validace webu? Odpověď je celkem jednoduchá. Jde o opravování chyb v zdrojovém kódu WWW stránky. Pokud bude web obsahovat mnoho chyb je možné, že se v některém prohlížeči bude zobrazovat špatně. Ještě horší je, pokud se Vám web ukazuje v každém prohlížeči jinak. Proto je tu validace webu. Pomocí webových nástrojů opravíte všechny chyby v kódu a tím docílíte správného zobrazení Vaší WWW jak opravdu chcete.

Pro lepší pochopení si uvedeme **příklad chyby v HTML kódu:**

```
<div>
 <p>Validní příklad</p>
</div>
<div>
 <p>Nevalidní příklad</div>
</p>
```

Validní příklad

- Je napsán správně. Všechny tagy jsou ukončeny postupně.

Nevalidní příklad

- Je již špatně. Všechny tagy musí být ukončovány postupně.

V tomto příkladu máme otevřené DIV a v něm se nachází P, který není uzavřen. Je uzavřen až za tagem DIV a proto ho prohlížeč již nevidí = ukončovací značka P je na tomto místě k ničemu.

K čemu slouží Validace webu?

Validní kód je zcela přehlednější, příjemnější a dokonce se i rychleji načítá.

Proto se vyplatí Validovat web. Po opravení všech chyb je dobré zhlédnou svojí stránku v několika nejpoužívanějších prohlížečích. Doporučuji volbu: Mozilla Firefox, Internet Explorer a Opera. Pokud máte web správně napsáný měl by se zobrazovat ve všech prohlížečích stejně. A to je také nesmírná výhoda validace, proto je validování důležité!

Pokud se web v prohlížečích nezobrazuje správně a je validní, bude chyba na straně prohlížeče. Jediné co v této situaci můžeme udělat je chybu nahlásit
nebo doufat, že v nové verzi prohlížeče tato chyba již nenastane.
Důležité je, snažit se psát stránky rovnou validní. Po dokončení webu jej validátorem pouze zkontrolovat a opravit nalezené chyby. Nikdo není dokonalý a překlepnout se může každý.
Někteří lidé si myslí, že validace webu je zbytečná a naprosto k ničemu. To je hloupost! Validace webu je důležitá. Máte jistotu, že se stránka bude zobrazovat tak, jak má. Validace také urychluje načítání stránek, protože s nevalidním webem má prohlížeč více práce. Pro lidi co stále nevěří jsem udělal jednoduchý test. Svůj nejmenovaný web jsem měl nevalidní s množstvím chyb. Načítal se přibližně za 1.2 sekundy. Po kompletním opravení chyb se načítal rychleji! Zrychlení bylo přibližně o 0.4 sekundy. K tomu je dobré myslet na to, že čím rozsáhlejší web tím bude rozdíl ještě větší.
Závěrem si položme otázku. Má opravdu význam validovat web? Ano! Má. Pokud chcete, aby Váš web vypadal profesionálně a načítal se co nejrychleji. Až se na web podívá profesionál bude mu jasné, že právě tenhle web nedělalo žádné prase.
Neustále nevěříte? Potom se můžete alespoň utěšit, že správně napsaným webem dáte najevo, že nejste žádný idiot a víte, co děláte.

Jak validovat web?

Stačí navštívit některou stránku, které jsou vypsané níže a zkontrolovat svůj web. Existují také různé doplňky do prohlížečů, které Validátor integrují do vašeho prohlížeče. Například HTML Validátor pro Firefox, který vřele doporučuji.

Odkazy na validátory:

Oficiální validátor:
http://validator.w3.org/

Český validátor:
http://validator.w3.cz/

Ostatní validátory:
http://www.xhtml-css.com/
http://htmlhelp.com/tools/validator/
… a další.

SEO

SEO (Search Engine Optimization, optimalizace pro vyhledávače) je metodologie vytváření a upravování webových stránek takovým způsobem, aby byly ve výsledcích hledání v internetových vyhledávačích zobrazeny na nejlepších místech (tj. tam, kde je hledající hledají). Cílem je nalákat na vlastní stránky co nejvíce zákazníků (popř. návštěvníků).

Cílem SEO optimalizace je dostat webovou prezentaci na první pozice ve fulltextových vyhledávačích, neboť většina uživatelů při hledání věnuje pozornost jen několika prvním odkazům. Webová prezentace by tedy měla být důvěryhodná nebo-li kvalitní. Tohoto výsledku lze dosáhnout několika způsoby, a to především kvalitním obsahem a správným způsobem programování, čili **SEO optimalizací**. Není proto jediným kritériem funkčnost stránek, ale webové stránky musí splňovat jisté normy, validitu, přehlednost tak, aby roboti v nich našli co nejvíce klíčových slov. Námi vytvořené webové stránky *SEO optimalizaci* již zahrnují.

Proč jsou některé stránky na prvních pozicích ve vyhledávačích a jiné nejsou k nalezení vůbec?

Pořadí nalezených odkazů je silně ovlivněno tzv. *PageRankem*. PageRank je hodnota, která vyjadřuje jakousi důvěryhodnost stránky. Každá stránka má svůj **PageRank** a čím je jeho hodnota vyšší, tím je webová stránka důvěryhodnější, tudíž ji fulltextový vyhledávač umístí na přednější pozici. Hodnota PageRanku webových stránek velmi závisí na množství zpětných odkazů směřujících z jiných stránek na Vaší - proto jen SEO optimalizace mnohdy nestačí a je potřeba webové stránky zaregistrovat do katalogů. Nejlepší však je, hned na začátku, tvorbu webových stránek svěřit odborníkovi, než-li pod zamínkou ušetřit, ve skutečnosti později za SEO optimalizaci a registraci zaplatit mnohem více. Mít webové stránky, které nejsou k nalezení, je to samé, jako je nemít!