

01. Datové formáty

Bit

- Základní a nejmenší jednotka informace; 2 hodnoty (0,1)

Byte

- „Bajt“
- 8 bitů
- Rozsah je od 00000000-11111111 → lze uložit maximálně 256 čísel

Datový formát

- Způsob, jakým jsou (zpravidla v nějakém souboru) organizována data
- Veškeré obsažené informace a způsob reprezentace dat a jejich následná interpretace

Znakové sady

ASCII

- Základní znaková sada (American Standard Code of Information Iterchange)
- Znaky jsou uložené pomocí bytu (256 znaků)
- V ASCII jsou uložené písmena, číslice, tisknutelné i netisknutelné znaky
- Původní velikost ASCII byla **128** znaků, poté rozšířena na **256**

UNICODE

- Tabulka znaků všech existujících abeced
- Obsahuje více než 110 000 znaků
- Založena na ASCII
- Využívá se ve Windows, Linuxu, HTML...
- Jednoduchý; univerzální; jednoznačný

UTF-8

- Způsob kódování řetězců znaků Unicode/UCS do sekvencí bajtů
- Navržen pro zpětnou kompatibilitu s ASCII

Windows 1250 (CP-1250)

- Navržen pro střední Evropu Microsoftem
- Využívá latinku

Soubor

- Pojmenovaná sada dat, uložená na datovém médiu
- Obsahem mohou být různá data.
- Podle toho, jak má být obsah souboru interpretován lze soubory rozdělit na **textové a binární**

Vlastnosti souborů:

- Jméno
- Typ
- Lokace
- Velikost
- Ochrana
- Datum
- Uživatelská identifikace

Atributy:

- R – Read – pro čtení
- S – System – systémový soubor
- H – Hidden - skrytý
- A – Archive

Komprese dat

Používá se za účelem zmenšení objemu dat

Ztrátová

Tam, kde je možné ztrátu některých informací tolerovat. (mp3, jpg...)

Bezeztrátová

Tam, kde ztráta jednoho znaku by mohla znamenat nenávratné poškození dat. (wmv, bmp, flac...)

True-Type

- Standart pro popis vektorových počítačových písem
- Vyvinutý koncem 80. let společností **Apple**
- Využívá i MS Windows a Gnu/Linux.
- Příponu TTF

Open-Type

- Standard pro popis vektorových počítačových písem
- Vyvinutý společností **Microsoft**
- Nástupce standardu True-Type
- Využití v MS Windows, Mac OS X a Linux
- Přípona OTF lze použít i starší TTF

02. Rastrová grafika

Existují 2 způsoby ukládání obrázků = Vektorově a Rastrově.

Rastrová / bitmapová grafika

- Celý obrázek je popsán pomocí jednotlivých barevných bodů (**pixelů**)
- Body jsou uspořádány do **mřížky**
- Každý bod má určen svou přesnou polohu a barvu (např. **RGB**)
- Tento způsob popisu obrázků používá také např. televize nebo digitální fotoaparát
- Kvalitu záznamu obrázku ovlivňuje především rozlišení a barevná hloubka

DPI

- Dots per inch, počet obrazových bodů na délku jednoho palce (2.24 cm)
- Popisuje potřebné rozlišení pro výstupní zařízení, které se použije pro zobrazení

Barevná hloubka

- Určuje kolik barev je použito na jeden bod
- Podle počtu barev se dají rozdělit obrázky na **monochromatické** (černá a bílá), **stupně šedi** a **barevné**

Výhody

- Optické uchování snímku
- Velmi jednoduché pořízení snímku
- Jednoduché zobrazení a programová podpora
- Možnost používání grafických efektů

Nevýhody

- Při zvětšení je patrný rastr
- Zvětšování a zmenšování obrázku vede ke zhoršení obrazové kvality

Rastrové soubory

Nekomprimované

BMP, TIFF...

Komprimované

Bezeztrátové – GIF, PNG...

Ztrátové – JPEG...

Barevné modely

Používá základní barvy a mísení těchto základních barev do výsledné barvy.

- RGB – Red, Green, Blue
- CMYK – Cyan, Magenta, Yellow, Key (nejčastěji černá)
- HSL – Hue, Saturation, Lightness
- HSV – Hue, Saturation, Value
- HSB – Hue, Saturation, Brightness

Histogram

Graf, který říká, jaké je rozložení jasů v obraze.

- Podexponování – příliš stínů
- Dobrá expozice
- Přeexpozivání – přesvícené

Formáty

- **Animated Portable Network Graphics (.apng)**
 - Rozšiřující formát PNG s podporou animací
- **Windows Bitmap (.bmp)**
 - Nepoužívá žádnou kompresi
 - Velmi velká velikost souboru
 - Velikost se dá snadno vypočítat (šířka v pixelech * výška v pixelech * bitů na pixel / 8).
- **Graphics Interchange Format (.gif)**
 - Bezeztrátovou komprese
 - Hlavní využití našel ve webové grafice a na Internetu
 - Použití pro loga
 - Obsahuje nízký počet barev
- **Joint Photographic Experts Group (.jpg, .jpeg)**
 - Ztrátovou komprese
 - Používá se především pro ukládání fotografií na Web
 - Není vhodný pro zobrazení textu, ikon a perokresby
- **Multiple-image Network Graphics (.mng)**
 - Grafický formát pro animované obrázky, který byl vyvinut jako doplněk PNG
 - Internetové prohlížeče tento formát často nepodporují.
- **Portable Network Graphics (.png)**
 - Bezeztrátovou komprese
 - Byl vyvinut jako náhrada a zdokonalení formátu GIF
 - Největší využití na Internetu
 - Nevýhodou je nedostupnost animace, což vyřešily jiné formáty jako doplňky
- **Tagged Image File Format (.tiff)**
 - Považován za neoficiální standard ukládání snímku pro tisk. Dále je používán pro ukládání faxů

Vektorizace

- Převod rastru na vektor
- Vytváří se digitální vektorová reprezentace vybraných prostorových prvků (např.: geologických jednotek, dokumentačních bodů, zlomů, vodních toků, komunikací, měst a obcí)
- Jednotlivé prvky mohou být vektorově reprezentovány pomocí: bodů, linií a ploch (polygonů)

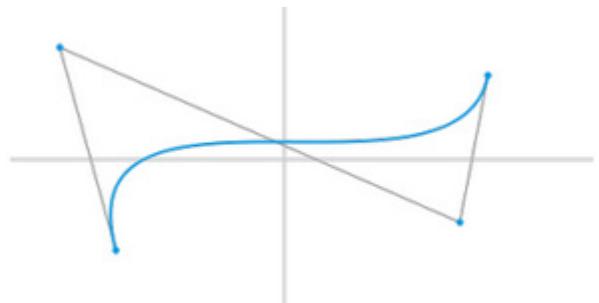
03. Vektorová grafika

Jeden z **2** základních způsobů reprezentace obrazových informací v počítačové grafice.

- Celý obrázek je zaznamenán matematicky pomocí geometrických útvarů (**body, přímky, mnohoúhelníky, kružnice nebo křivky**)
- Všechny tvary na obrázku jsou uloženy jako matematické vzorce jednotlivých křivek, ze kterých se skládají
- Ty jsou doplněny o další informace (barva, styl, tloušťka čáry...)

Beziérova křivka

- Pojmenovaná po inženýru Pierra Bézierovi
- Vyvinul metodu pro popsání libovolného úseku křivky pomocí čtyř bodů
- Křivka je definována dvěma kotevními body, které značí její začátek a konec, a dvěma kontrolními body, které udávají její tvar



Výhody

- Libovolného zmenšování a zvětšování obrázku bez ztráty kvality
- S každým objektem je možno pracovat odděleně
- Výsledná paměťová náročnost obrázku je u jednolitých barevných obrázků menší, než při použití rastrového zápisu

Nevýhody

- Složitější způsob pořízení obrázku oproti rastrové grafice
- U velmi složitých obrázků náročná na výkon procesoru a operační paměť
- Nehodí se na zápis složitých barevných ploch (např. fotografie)

Využití

- | | |
|---|--|
| <ul style="list-style-type: none"> • Fonty • Loga • Diagramy | <ul style="list-style-type: none"> • Plány • Výkresy |
|---|--|

Formáty

- Postscript (.ps) – Adobe; nezávislý na HW
- Portale Document Format (.pdf) – Adobe; vychází z postscriptu
- Scalable Vector Graphics – W3C; vychází z XML

Rasterizace

- Proces, při kterém se vektorově definovaná grafika konvertuje na rastrově definované obrazy
- Při zobrazení reálného modelu ve světových souřadnicích na výstupní zařízení je zapotřebí zajistit, co nejvěrnější podobnost reálného a zobrazovaného modelu
- Nejjednodušší prvek rastrové grafiky je **bod**
- Složitější objekty jsou jen skládankou z jednodušších objektů

04. Video

Pojem **video** společně označuje digitální a analogové způsoby ukládání obrazových záznamů.

Kvalita

- Kvalita videa je závislá na metodě zachycování a ukládání obrazu
- Nejdůležitějším kritériem je formát uložení
 - Různé formáty mají různý poměr kvalita/objem

Hlavní faktory ovlivňující kvalitu videa jsou:

Frame Rate

- Frames per Second (**FPS**), počet snímků za sekundu
- Technologie zobrazení začínaly na **6** či **8** snímkách
- V dnešní době se nejčastěji vyskytuje **24FPS** (23,976), **25FPS**, **30FPS** (29,97) a snahy o zvýšení plynulosti v některých nových filmech za pomocí **45FPS** a **48FPS** (Hobbit)

Prokládání

- Video může být **prokládané (interlaced)** nebo **progresivní (progressive)**
- Prokládání bylo zavedeno pro dosažení lepší vizuální kvality v limitech pásmo
- Každý snímek je rozdělen na dva půlsnímky trvající polovinu doby celého snímku
 - První obsahuje liché, druhý pak jen sudé řádky
- Progresivní video půlsnímky neobsahuje

Rozlišení

- Analogové formáty udávají rozlišení v řádcích a Digitální formáty v pixelech
- Rozlišení pro 3D video se udává ve **voxelech**
 - Množství obrázkových prvků reprezentující hodnotu v trojrozměrném prostoru

Poměr stran

- Popisuje poměr vodorovné a svislé strany
- Nejčastěji používané poměry jsou 4:3 (starší televize) a 16:9 (snad všechno)

Datový tok

- Množství digitálních dat přenesené za určitou časovou jednotku
- Nejčastěji v **Megabitech** za sekundu (**Mbit/s**)

VBR – Variable bit rate

- Způsob maximalizace kvality videa a zároveň co nejnižší množství přenesených dat
- Ve scénách s rychlými pohyby je datový tok daleko vyšší, než ve scénách bez pohybu
- Není-li třeba pro popsání obrazu tolik bitů, nepřenesou se, naopak je-li jich potřeba více, přenáší se jich více

CBR – Constant bit rate

- Po celou dobu nahrávání obrazu bude datový tok **konstantní**
- Lze snadno určit výslednou velikost

Formáty obrazu

- 720x576 – DVD formát
- 1024x720 – HD DVD formát 4:3
- 1280x720 – 720p; HD 16:9
- 1920x1080 – 1080p; FULL HD 16:9
- 3840x2160 – 2160p; 4K ULTRA HD 16:9

Kódování

Video kodek

Kóduje a dekóduje video do/z určitého formátu. Zpravidla za účelem zmenšení objemu dat.

- **Bezeztrátové** kodeky (HuffYUV, Lagarith)
- **Ztrátové** kodeky (DivX, Xvid, Quick Time)

Televizní normy

Souhrn standardů kódování signálu pro televizní vysílání.

- PAL
 - Phase Alternating Line
 - Evropa, Austrálie, část jižní Ameriky
- NTSC
 - National Television System(s) Committee
 - Amerika, Japonsko, Jižní Korea
- SECAM
 - Séquentiel couleur à mémoire
 - Postoupení barevné informace do paměti
 - První evropský systém barevné televize
 - Používá barevný model velmi podobný YUfu

Střih videa

- Linerární
 - Přímé stříhání a slepování filmové pásky
- Nelinerání
 - Moderní přístup
 - PC úpravy ve střihovém formátu → render

Formáty

- AVI
 - Microsoft
 - Nejstarší, Nejrozšířenější
- MP4
- MKV
 - Ruský; Matroska
 - Kontejner
 - Umožňuje nést více audio, video stop, titulků, kapitol...
- OGG
- WMV
 - Windows
 - Komprimovaný

Média

- VHS
 - Firma JVC
 - Nahrávání na pásku
 - 576x240
 - 240 minut
- Laserdisc
 - Philips, Pioneer
 - Oboustrané
- CD
 - Philips, Sony
 - MPEG-1
 - 1150 kb/s
 - 80 minut
 - 700MiB
- DVD
 - Philips, Sony, Toshiba
 - MPEG-2
 - PAL – odlišný od televizního standardu
 - 720x576
 - 4:3, 16:9
- HD DVD
 - Toshiba
 - VC-1, H.264, MPEG-2
 - 1920x1080
 - 15GiB, 30GiB
- Blu-ray disc
 - LG, Samsung, Sony
 - H.262/MPEG-2, H.264/MPEG-4 AVC, VC-1
 - 1920x1080
 - 25GiB, 50GiB

05. Zvuk

Mechanické vlnění vzduchu (přesněji: látkového prostředí) v rozsahu od **10² Pa – 10⁵ Pa**. Frekvence tohoto vlnění každý člověk vnímá individuálně (obvykle v intervalu **16 Hz – 20 kHz**).

Tón

Hudební. Zvuk se stálou frekvencí.

Hluk

Nehudební. Zvuk, který má rušivý charakter.

Vlastnosti zvuku

Výška

- Výšku udává Frekvence, ↑frekvence = ↑výška
- S výškou souvisí slyšitelné a neslyšitelné frekvence.
- Slyšitelné = 16 (20) Hz – 20 kHz. Neslyšitelné = **infrazvuk (<16 Hz)**, **ultrazvuk (>20 kHz)**
- Při hodnocení zvuku se používá „relativní výška tónu“ ($\frac{\text{frekvence tónu}}{\text{frekvence referenčního tónu}}$)
- Hudební **akustika** (vědní obor; zabývá se zvukem) udává jako referenční tón 440 Hz
- V praxi se spíše používají jiné referenční napětí
- Při prvotním nastavování převážně **1 kHz**

Barva

- Existují zvuky o stejném tónu, které se od sebe liší přítomností „**vyšších harmonických frekvencí**“ → Obsahují stejnou základní frekvenci a zároveň její celočíselné násobky (2f, 3f, 4f)
- Liché násobky základní frekvence zvuk **zostřuje** a sudé ho **zjemňuje** → určují výslednou barvu

Hlasitost

- Veličina závislá na velikosti akustického tlaku ($L_p = 20 \log \frac{P}{P_0}$ dB; $P_0 = 20\mu\text{Pa}$ – Práh slyšení)
- Aby bylo možné přiblížit hlasitost bez závislosti na frekvenci, vznikly **4 křivky**, které slouží jako normy (A,B,C a D)
 - Křivka A udává, že frekvence 1 kHz odpovídá 0db (250 Hz = -10 db)

Intenzita

- Zvuková energie dopadající na plochu za čas; akustický výkon na plochu ($I = \frac{E}{S*t}$)
- Hladina intenzity = udává intenzitu zvuku v dB ($L = 10 \log \frac{I}{I_0}$)

Zvuk z pohledu techniky

- Při digitalizaci se využívá „**Shannonův-Nyquistův-Kotělníkovův teorém**“. → Udává, že „*Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byla vzorkovací frekvence vyšší než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu.*“

Úpravy zvuku

- Frekvenční
 - Změna barvy
- Amplitudové
 - Změna hlasitosti
- Modulační
 - Přidání dalšího signálu
- Kvalitativní
 - Komprese
 - Resampling
- Efektové

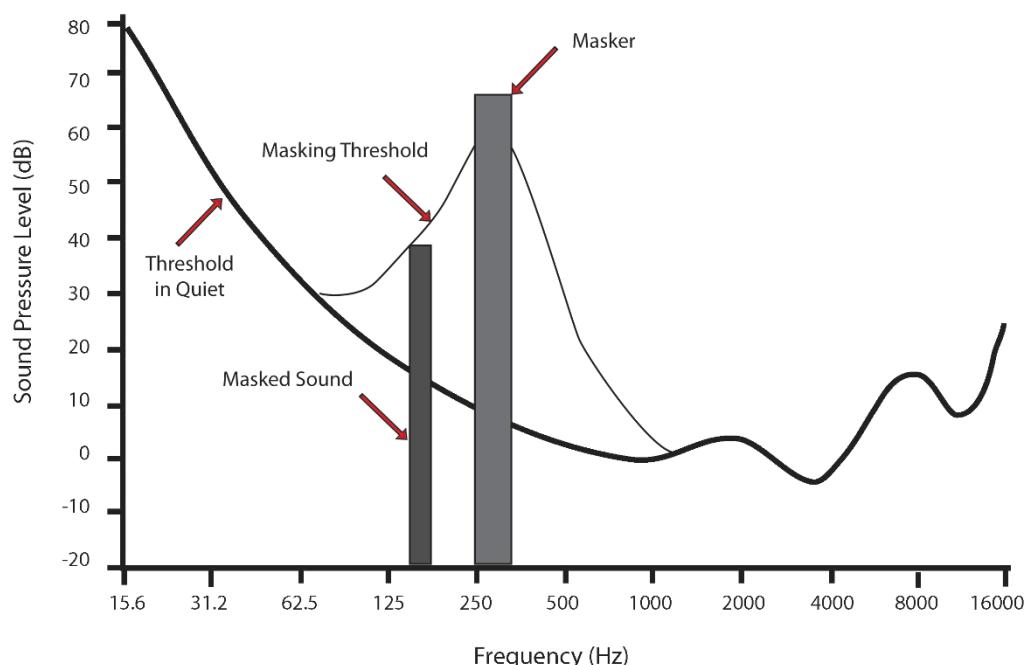
Stopy

- Mono
- Stereo
- Quadro
- 2:1
- 5:1
- 7:1

Formáty

MP3

- Komprimovaný, ztrátový
- Odstraňuje redundantní zvukové signály na základě psychoakustiky
 - Ze vstupního signálu se odeberou informace, jež člověk neslyší, nebo si je neuvědomuje
 - Časové a frekvenční maskování



AAC

Následník MP3 při vyšších bitratech. Existuje ve spoustě profilů (FAAC...)

FLAC

Beztrátový

06. Textový procesor

- Aplikace určená ke zpracování, úpravě nebo tisku textu

Textový editor X procesor

- Editor umožňuje uchovat pouze plain text
- Procesor umožňuje měnit vzhled dokumentu (formát, font...)

Karty

- Domů (písmo, odstavec, styly)
- Vložení (stránky, ilustrace, tabulky, odkazy, záhlaví a zápatí, text, symboly)
- Rozložení stránky (motivy, vzhled, pozadí, odstavce, uspořádání)
- Reference (obsah, poznámky, citace, titulky, rejstřík)
- Korespondence (hromadná korespondence)
- Revize (kontrola pravopisu, sledování změn, komentář, uzamčení)
- Zobrazení (lupa, okna, zobrazení dokumentů)
- Vývojář (kód, ovládací prvky)

Formátování textu

- Font
- Velikost písma
- Varva písma
- Řez písma
- Efekty, Styly

Formátování odstavce

- Zarovnání
- Řádkování
- Stínování
- Ohraničení
- Odsazení
- Mezery
- Tabulátory

Reference

- Vytvoření obsahu
- Poznámky pod čarou
- Citace
- Bibliografie

Vložení

- Tabulky
- Ilustrace
- Multimédia
- Odkazy
- Komentář
- Záhlaví a zápatí
- Text
- Symboly

Rozložení stránky

- Vzhled stránky
- Odstavec
- Uspořádat

Odkazem může být:

- Hypertextový odkaz
- Křížový odkaz
- Záložka

07. Tabulkový kalkulátor – vzorce, funkce

Tabulkový kalkulátor

- Program sloužící k matematickým operacím s číselnými údaji
- Soubor = sešit
- List – maximálně 256 listů v sešitu
- Řádky se označují číslicemi (lze přenastavit na R1, R2...)
- Sloupce se označují písmenami (lze přenastavit na C1, C2...)
- Buňka = průsečík sloupce a řádku

Formát buněk

- Číslo
- Zarovnání
- Písmo
- Ohraničení
- Výplň
- Vlastní

Buňky

- První Buňka má adresu A1 (R1C1)
- Poslední adresa je XFD1048576
- Pravý dolní „úchyt“ – vyplnění řady

Vzorce

- Zapisují se do buňky stejně jako klasický text
- Zápis musí začínat znakem =
- Výsledky se zobrazují klasicky v buňce a vzorce v poli vzorců

Funkce

- Datum a čas (DATUM, DNES, DENTYDEN...)
- Logické (A, KDYŽ, NEBO...)
- Matematické (ZAOKROUHLIT, SUMA...)
- Text (ČÁST, HODNOTA.NA.TEXT...)
- Vyhledávání (VYHLEDAT...)
- Databáze; Finanční; Informační; Kompatibilita; Statické

Vnořené funkce

Používají funkci jako jeden z argumentů jiné funkce.

Grafy

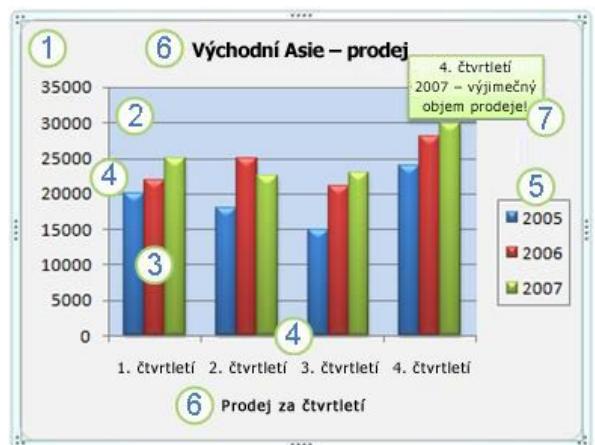
- Sloupcový
- Spojnicový
- Výsečový
- Pruhový
- Plošný
- Bodový

Prvky grafu

- 1. Oblast grafu
- 2. Zobrazovaná oblast
- 3. Datové body v Datová řadě, které jsou graficky znázorněny v grafu
- 4. Vodorovná (kategorie) a svislá (hodnota) Osa, podél které jsou data graficky znázorněna v grafu
- 5. Legenda grafu
- 6. Název grafu a osy, který je možné použít v grafu
- 7. Popisek dat, který lze použít k identifikaci detailů datového bodu v datové řadě

Vytvoření grafu

- Označení dat → Vložení → Graf (vybrat typ grafu)
- Dále lze přidat obrázky, popisky...



08. Tabulkový kalkulátor - filtry, souhrny, kontingenční tabulky

Import dat

- Data, která se mají analyzovat v aplikaci Excel, není třeba znova zadávat, stačí je importovat
- Import dat z databází a souborů
- Import dat pomocí aplikace Microsoft Query
- Import dat z webu
- Import dat pomocí VBA

Řazení dat

- Nedílná součást analýzy dat
- Seřazení textu – od A do Z a naopak
- Seřazení čísel – od nejnižšího po nejvyšší a naopak
- Seřazení podle data – od nejnovějšího po nejstarší a naopak
- Vlastní řazení – barvy buněk, barva písma, ikony...

Filtrování

- Rychlý a snadný způsob vyhledávání a práce s podmnožinou dat
- Po filtrování jsou zobrazeny jenom řádky, které splňují zadaná kritéria
- Zbylé řádky jsou skryty

Ověření dat

- Funkce, kterou lze použít k definici omezení dat, která mohou být nebo by měla být zadána do buňky

Souhrny

- Rozsáhlejší tabulku lze zpřehlednit a doplnit ji o součty sloupců, které jsou požadovány (cena, prodané kusy)
- Tyto součty budou provedeny na základě seskupení shodných dat v dalších sloupcích (roční období, lokalita)
- Místo souhrnu lze využít i jiné možnosti (Kontingenční tabulku nebo VBA)

Makra

- Posloupnost akcí, funkcí nebo příkazů, které usnadňují určitou činnost

Kontingenční tabulka

	A	B	C
1	Sport	Čtvrtletí	Prodej
2	Golf	Čtv3	15 000 Kč
3	Golf	Čtv4	20 000 Kč
4	Tenis	Čtv3	6 000 Kč
5	Tenis	Čtv4	15 000 Kč
6	Tenis	Čtv3	40 700 Kč
7	Tenis	Čtv4	50 000 Kč
8	Golf	Čtv3	64 300 Kč

1. Zdrojová data, v tomto případě z listu
2. Zdrojové hodnoty pro souhrn Golf za Čtv3 v sestavě kontingenční tabulky
3. Celá sestava kontingenční tabulky
4. Souhrn zdrojových hodnot v buňkách C2 a C8 ze zdrojových dat

	E	F	G
3	Součet z Prodej	Čtvrtletí ▾	
Sport	4	Čtv3	Čtv4
Golf	79 300 Kč	20 000 Kč	
Tenis	46 700 Kč	65 000 Kč	
Celk. součet	126 000 Kč	85 000 Kč	

- Základní nástroj pro práci a vyhodnocování dat
- Vizualizace vzájemného vztahu dvou nebo více statistických znaků
- Analýza dat, souhrny, třídění, výpočty...
- Snadné pochopení prezentovaných dat

Vytvoření kontingenční tabulky

- Kliknutí do tabulky → Vložení → Kontingenční tabulka

09. Prezentační software

- Počítačový program, který umožňuje vytvořit prezentaci (sérii stránek s přehledně zobrazenými informacemi)

Microsoft Office PowerPoint

- Nejpoužívanější
- Přípony .ppt, .pptx

Přechody

- Mezi jednotlivými snímky

Animace

- Mezi jednotlivými objekty daného snímku

Pravidla pro prezentéra

- Vhodné oblečení
- Nedávat ruce do kapes
- Artikulovat, klidně, hlasitě mluvit
- Neotáčet se zády
- Udržovat pozornost
- Komunikace s publikem
- Věnovat se pouze prezentaci

Pravidla pro prezentace

- Psát v bodech
- Nedávat zbytečné animace, přechody
- Vyhýbat se zbytečným přechodům, moc obrázků, kříklavé barvy...)

10. Databázový procesor - teorie, pojmy

Databáze

- Uspořádaná množina dat (informací) uložena na paměťovém mediu
- Součástí databáze jsou softwarové prostředky pro manipulaci s daty + přístup k datům

Rozdělení

- Systémy sálových počítačů (Mainframe)
- dBase
 - Souborově orientované databáze s indexekvenční metodou přístupu
 - Každá tabulka má samostatný **.dbf** soubor
 - Software: dBase, FoxPro, Paradox, Access
- Relační databázové systémy (lepší datová integrita, bezpečnost...)
- Objektově orientované databáze (specializované uplatnění, data se ukládají jako objekt s vlastnostmi)

Databázový procesor

- Nástroj, který slouží pro práci s velkým množstvím dat; MS Access, Firebird, Oracle
- V databázi se data upravují, ukládají, získávají
- Obsahuje jednotlivé akce – moduly:
 - Tabulka
 - Dotazy
 - Formuláře
 - Sestavy

SŘBD; DBSŘ; DBMS

- Systém řízení báze dat; Databázový systém řízení; Database management system
- Softwarové vybavení, které zajišťuje práci s databází (tvoří rozhraní mezi aplikačními programy a uloženými daty)
- **Databázová aplikace** je program, který umožňuje vybírat, prohlížet a aktualizovat informace uložené prostřednictvím SŘBD
- SŘBD musí být schopen efektivně pracovat s velkým množstvím dat a také musí být schopný řídit data (vkládat, modifikovat, mazat) a definovat strukturu těchto dat

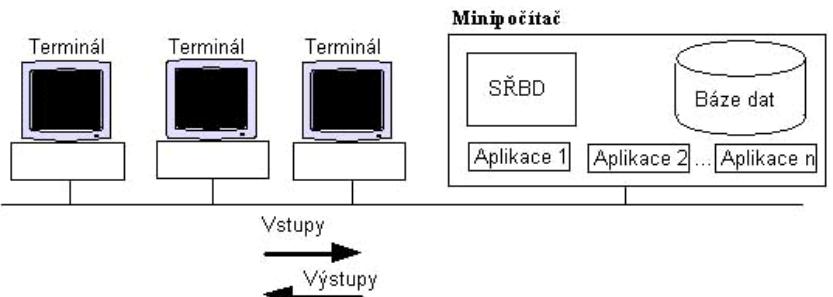
Služby

- Definice dat (definování a uchovávání datové entity)
- Údržba dat (každému členu entity vyhrazuje záznam skládající se z položek)
- Manipulace s daty (služby umožňující vkládání, aktualizaci, rušení a třídění dat)
- Zobrazování dat (poskytuje metody prezentace dat uživateli)
- Integrita dat (metody pro zajištění správnosti dat nepovolením vložení duplicitního řádku s unikátním klíčem)

Architektury DB

Centrální

Tato architektura je typická pro terminálovou síť, kdy se po síti přenáší vstupní údaje z terminálu na centrální počítač do příslušné aplikace, výstupy z této aplikace se přenáší na terminál. Protože aplikacní program i vlastní zpracování probíhá na centrálním počítači, který může zpracovávat více úloh, mají odezvy na dotazy určité zpoždění.

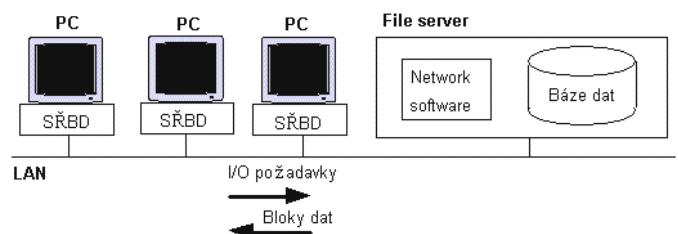


- Data i SŘBD jsou v centrálním počítači

File-Server

Tato metoda souvisí zejména s rozšířením osobních počítačů a sítí LAN.

- SŘBD a databázové aplikace jsou na jednotlivých počítačích
- Data jsou na File-Serveru



Komunikace uživatele se systémem:

- Uživatel zadá dotaz
- SŘBD přijme dotaz, zasílá požadavky na data file-serveru
- File-server posílá bloky dat na lokální počítač, kde jsou data zpracovávána podle zadaného dotazu (vyhledávání, setřídění...)
- Výsledek dotazu se zobrazí uživateli

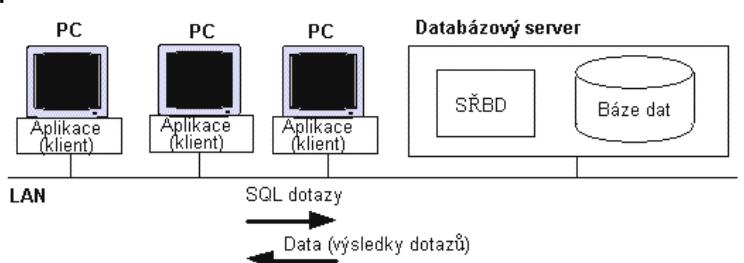
Klient-Server

V podstatě je založena na lokální síti (LAN), personálních počítačích a databázovém serveru. Na počítačích běží program pro komunikaci se serverem.

- Redukuje množství přenesených dat (v porovnání s File-Serverem)

Komunikace:

- Uživatel zadává dotaz (buď přímo v SQL, nebo musí být do tohoto jazyka přeložen)
- Dotaz je odeslán na server
- Server vykoná dotaz
- Výsledek dotazu je poslan zpět na vysílací počítač, kde je zobrazen



Distribuované

Množina databází, která je uložena na několika počítačích. Uživateli se však jeví jako jedna velká databáze.

- **Funkční**

- Vertikální členění

- **Objektové**

- Horizontální členění

Návrh DB

- Určení účelu DB
- Vyhledání a uspořádání požadovaných informací
- Rozdělení informací do tabulek
- Převod jednotlivých informací do sloupců
- Zadání primárních klíčů
- Vytvoření relací mezi tabulkami
- Úprava návrhu
- Použití normalizačních pravidel
 - **Sémantické modelování** – analyzuje požadavky a zobrazuje tyto požadavky určitými grafickými prostředky
 - **Entitně** – relační modelování (E/R diagram)

E/R Diagram

- **Entita**

- Subjekt, o němž se bude v databázi uchovávat informace

- **Relace**

- Propojení tabulek

- **Tabulky**

- Slouží k uložení dat

- **Sloupce, atributy**

- Popisuje určitou část dat, kterou má každý záznam
- Sloupec představuje část tabulky
- Atribut se vztahuje k reálné entitě

- **Domény**

- Popisují typ dat, obor hodnot

- **Řádky, záznamy, n-tice**

- Každý řádek v tabulce představuje záznam o jedné entitě

Klíče

Databázová konstrukce, sloužící ke zrychlení vyhledávacích a dotazovacích procesů v databázi, definování unikátní hodnoty sloupce tabulky

- **Primární klíč**
 - Svou hodnotou jednoznačně identifikuje každý záznam
- **Unikátní klíč**
 - Nemusí být jediný
- **Cizí klíč**
 - Odkaz mezi tabulkami

Kardinalita

Vyjadřuje, kolik entit jednoho typu může být ve vztahu s kolika entitami z druhého typu entit

- **1:1**
 - Používá se, pokud záznamu odpovídá právě jeden záznam v jiné tabulce
- **1:N**
 - Přiřazuje jednomu záznamu více záznamů z jiné tabulky
- **N:M**
 - Umožňuje několika záznamům z jedné tabulky přiřadit několik záznamů z tabulky druhé
 - V praxi se spíše používá 1:N a M:1 pomocí jedné propojovací tabulky

Relační Algebra

- Základní prostředek pro manipulaci s daty
- Teoretický základ dotazovacích jazyků (SQL, LINQ, DMX, MDX, Datalog)
- Je dána operátory, které se aplikují na relace a výsledkem jsou opět další relace
- $R(\{A_1, A_2, \dots, A_n\}); S(\{B_1, B_2, \dots, B_m\})$

Základní operace

- **Sjednocení (Union) R**
 - Vytvoření relace obsahující všechny řádky (prvky) obou relací, ale shodné řádky se neopakují
 - Relace S, R musí být kompatibilní
 - Mají stejný počet atributů, v některých případech musí mít atributy stejný název a datový typ
- **Průnik (Intersection)**
 - Vytvoření relace obsahující společné řádky obou relací, ale společné řádky se neopakují
 - Relace S, R musí být kompatibilní

- **Rozdíl** (Difference)
 - Vytvoření relace obsahující jen ty řádky první relace, které nejsou obsaženy v druhé relaci
 - Relace S, R musí být kompatibilní
- **Kartézský součin** (Cartesian product)
 - Vytváří relaci obsahující všechny řádky první relační tabulky zřetězené postupně se všemi řádky druhé relační tabulky

Speciální operace

- **Projekce** (Projection)
 - Projekce $R[C]$ na relaci se schématem $R(\{A_1, A_2, \dots, A_n\})$ na množinu C, kde C je menší, nebo rovno množině $\{A_1, A_2, \dots, A_n\}$
 - Odstraní se i duplicitní řádky
- **Selekce, Restrikce** (Selection, Restriction)
 - Je relace se schématem $R(\{A_1, A_2, \dots, A_n\})$ podle logické podmínky φ (φ je jednoznačně true/false)
 - Projekce a selekce jsou operace s jednou relací – **unární** operace
- **Spojení** (Join)
 - Slouží pro spojení množin na základě společných prvků zvoleného atributu
 - Natural join
 - Podmínka je určována automaticky, ne často se používá
 - Inner join
 - Kartézský součin
 - Full outer join
 - Stejně jako inner join
 - Left outer join
 - Výsledek uzná, pokud existuje levá část vazby a pravá neexistuje
 - Do hodnot sloupců z připojované části se vloží NULL
 - Right outer join
 - Pokud bude existovat pravá, připojovaná část a nebude k ní levá část, bude stejně ve výpisu zahrnut

Relační kalkul

- Formální neprocedurální jazyk
- N-ticově a doménově orientovaný

Zápis:

- | | |
|--|---|
| <ul style="list-style-type: none"> • Termy <ul style="list-style-type: none"> ○ Proměnné ○ Jejich komponenty ○ Konstanty | <ul style="list-style-type: none"> • Predikáty <ul style="list-style-type: none"> ○ > ○ < ○ >= ○ <= ○ <> ○ = |
|--|---|

- **Atomické formule**

- Konjunkce – &
- Disjunkce - ∨
- Negace - ¬
- Implikace - ⇒
- Ekvivalence - ⇔

- **Kvantifikátory**

- Univerzální (\forall) – „pro každý“
- Existenční (\exists) - „existuje“

11. Databázový procesor - dotazy, formuláře, sestavy

Ms Access

- Nástroj k vytváření databázových aplikací, databází
- Skládá se z:
 - Tabulky
 - Dotazy
 - Formuláře
 - Sestavy
 - Makra
 - Moduly

Dotazy

- Na základě zadaných argumentů (podmínek) lze pracovat s daty
 - Filtrace
 - Agregace
 - Vytvářet souhrny

Typy dotazů

- Výběrový
 - Nejběžnější
 - Zobrazuje data z tabulky/tabulek
- Parametrický
 - Po spuštění zobrazí dialogové okno pro zadání argumentů
- Křížový
 - Usnadňuje analýzu dat
 - Seskupuje data a umožňuje použití agregačních funkcí
- Akční
 - Odstranit; Vytvářecí; Aktualizační; Přidávací; Sjednocovací; Předávací; Definiční

Sestavy

- Oddělený a přehledný vzhled záznamů z tabulky nebo dotazu
- Největší využití je na vizitkách, štítcích nebo přehledech

Formuláře

- Prostředník mezi tabulkou a uživatelem
- Snadnější forma zadávání dat pro uživatele
- Labely; text fieldy; buttony; checkboxy; listy

Makra

- Pro vytvoření akce bez potřeby znát VBA
- Makro se postupně tvoří pomocí slovy definovaných modulů, jsou zde hlavně ty nejčastější, aby se nemuseli pokaždé psát v VBA znova
- Nejčastější úkoly pro makra jsou HledatZáznam, OknoSeZprávou, ZavřítDatabázi

12. Databázový procesor – VBA

- Visual Basic for Application
- 1993; Microsoft
- Programovací jazyk zaměřený na události a objekty
- Neobjektový

Moduly

- **Standardní modul**
 - Deklarace globálních proměnných, konstant a procedur
 - Ukládá se jako objekt (modul)
- **Modul třídy**
 - Deklarace globálních proměnných, konstant a procedur
 - Ukládá se jako objekt (modul)
- **Modul procedury**
 - Pro deklaraci lokálních proměnných, konstant a procedur
 - Je přidružen k formulářům a sestavám Accessu

Datové typy

Datový typ	Uloženo v Byte	Rozsah
Byte	1	0 až 255
Boolean	2	True, False
Integer	2	-32 768 až 32 767
Long (long integer)	4	-2 147 483 648 až 2 147 483 647
Single (jednoduchá přesnost, plovoucí desetinná čárka)	4	-3,402823 * 10 ³⁸ až 3,402823 * 10 ³⁸
Double (dvojitá přesnost, plovoucí desetinná čárka)	8	-1,79769313486231 * 10 ³⁰⁸ až 1,79769313486232 * 10 ³⁰⁸
Currency	8	
Decimal	14	
Date	8	1. 1. 100 až 31. 12. 9999
Object	4	
String,(proměnné délky)	10 + počet znaků řetězce	0 až 2 * 10 ⁹
String (pevné délky)	počet znaků řetězce	1 až 65 400
Variant (čísla)	16	Stejně jako Double
Variant (znaky)	22 + počet znaků řetězce	Stejně jako String
User defined (uživatelem definovaný)	počet podle prvků	Podle datových typů prvků

Operátory

Matematické

- +
- -
- *
- /
- %
- ^

Logické

- AND
- OR
- NOT
- XOR

Porovnávací

- ==
- <>
- <
- >
- <=
- >=

Cykly

- For
- For Each
- While Wend
- Do While
- Do Until

Podmínky

- If
- If else
- If elseif
- Select Case

Funkce

Funkce

- Blok kódu, který je znovu využitelný a může být volaný kdekoli v programu
- Většinou vrací hodnotu
- Volání:
 - MsgBox (così) Název funkce, argumenty v závorkách
 - Call MsgBox

Procedury

- Nevrací hodnotu (odlišné od funkce)
- Volání:
 - MsgBox così – Název funkce, argumenty bez závorek
 - MsgBox (così) – Volá se jako funkce; Využije se návratový typ
 - Call MsgBox (così) – Call název_funkce, argumenty v závorce

Modifikátory přístupu

- Public
 - Ve všech modulech a procedurách lze zavolat
- Private
 - Dostupné pouze pro modul

13. Internet a elektronická komunikace

Internet

- Celosvětový systém navzájem propojených sítí
- Slouží ke vzájemné komunikaci jednotlivých počítačů
- Cílem je přenos dat
- TCP/IP protokol

ARPANET (Advanced Research Project Agency)

- Projekt amerického ministerstva obrany z 60. let
- Měl prozkoumat možnosti výměny dat mezi vzdálenými „super“ počítači
- V 70. letech vzniká první e-mailový program a vznikají protokoly TCP/IP
- Síť je rozdělena pro výzkum a armádu
- Dostává se do komerční oblasti a přesahuje hranice USA
- V roce 1987 vzniká pojem Internet
- V roce 1991 vzniká WWW

World Wide Web

- Světová rozsáhlá síť
- Autorem Webu je **Tim Berners-Lee**
- Aplikace pro protokol HTTP
- Pracuje na principu **Client-Server**

Internetové služby

- Dostupnost množství informací
- Umožňuje komunikaci (elektronická pošta)
- Sociální sítě
- Bankovnictví, obchody...

Internetová Doména

- Označení jednoznačného jména počítače nebo počítačové sítě
- Doména se skládá z několika úrovní

Připojení k internetu

- **Telefonní linka** – modem
- **ISDN** – digitální telefonní linka
- **Kabelová televize** – síťová karta a kabelový modem
- Bezdrátové připojení lokálních sítí
- **Wi-Fi** – nejznámější standard pro bezdrátové sítě

Webový prohlížeč

- Program sloužící k prohlížení WWW
- Komunikuje pomocí HTTP protokolu s webovým serverem a přijatá data zformátuje a zobrazí na obrazovce počítače
- Textové - Links, Lynx
- Grafické - Google Chrome, Internet Explorer, Mozilla Firefox, Safari

Internetový vyhledávač a katalog

Vyhledávač

- Slouží pro vyhledávání uživatelem požadovaných informací a dat na Internetu
- Pracují ve čtyřech krocích
 - Procházení webových stránek
 - Vytvoření databáze výskytu slov
 - Indexování
 - Poskytování odpovědí na dotazy

Katalog

- Seznam odkazů na webové stránky, které jsou setříděny do stromu kategorií a podkategorií.

Elektronická pošta (Email)

- E-mailové služby se poprvé rozšířily mezi veřejnost, až díky vzniku jedné z prvních volných e-mailových služeb Hotmail v roce 1996
- Termín e-mail se používá jak pro internetový systém elektronické pošty založený na protokolu SMTP, tak i pro intranetové systémy

Historie

- Starší než internet
- Elektronická pošta vznikla v roce 1965
- E-mail se rychle rozšířil a stal se síťovým e-mailem
- Tehdy nebyly všechny počítače nebo sítě navzájem síťově propojené, e-mailové adresy musely obsahovat „cestu“ pro zprávu

Komunikační protokoly

- **SMTP (Simple Mail Transfer Protocol)**
- **POP3**
 - Pošta je při tomto protokolu stažena na lokální disk
- **IMAP**
 - E-maily jsou uloženy na serveru a je k nim možno přistupovat odkudkoliv
 - Stahuje pouze hlavičky e-mailů na rozdíl od POP3, kde se stahují celé zprávy

14. HTML

HyperText Markup Language je značkovací jazyk pro tvorbu webových stránek a aplikací. Byl vytvořen pro World Wide Web. V roce 1989 Tim Berners-Lee a Robert Caillau vytvářeli propojení informačních systémů pro CERN. V té době se užíval převážně PostScripta SGML, ale z práce Tima Berners-Lea v roce 1990 vznikla nová služba, která změnila pohled na Internet, byl to právě World Wide Web. Tim Berners-Lee v tomto roce navrhl HTML, HTTP protokol a spustil první webový server na světě. Tim je také tvůrcem prvního prohlížeče na světě WorldWideWeb, který byl pouze textový. První prohlížeč s grafickým uživatelským prostředím se jmenuje Mosaic.

HTML dokument je tvořen **tagy** a jejich atributy. Jsou 2 druhy tagů **Párové** a **Nepárové**.

```
<p align = „center“>< /p>  
<br />
```

Tagy by se neměly křížit!

Vývoj HTML

- 1991 – 1993 → verze 0.9 – 1.2; bez podpory grafiky
- 1995 → verze 2.0; interaktivní formuláře a podpora grafiky
- 1997 → verze 3.2; tabulky, stylové prvky, W3C validátor
- 1997 → verze 4.0; podpora rámců (<frame>) další prvky formuláře; měla být finální verze HTML, budoucí náhradník = XHTML
- 2014 → verze 5.0; aktuální verze, přidány nové tagy, modernizace a příprava pro multimediální obsah
 - Nový tag pro multimédia <audio>, <video>
 - Nový tag pro doctype <!DOCTYPE html>
 - Přidány tagy pro hlavičku a patičku <header>, <footer> ...

Validace

Zpětná kontrola HTML kódu a detekce syntaktických chyb. Oficiální validátor je od firmy W3C.

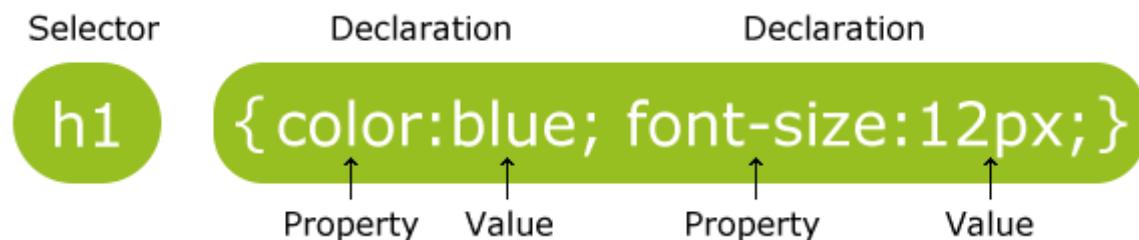
Při tvorbě webových stránek, či HTML5 aplikací se nepoužívá pouze samotné HTML, ale mnoho dalších jazyků. Nejhlavnější z nich jsou: JavaScript, CSS, XML, AJAX. Na straně serveru: PHP, ASP...

15. CSS

Cascading Style Sheets. Kaskádové styly. Jazyk pro popis zobrazení jednotlivých elementů v dokumentu HTML (XHTML). Vznikly v roce **1997** jako oddělení grafické části HTML od samotného HTML souboru. Aktuální verzí CSS je verze CSS3, která přinesla novinky ohledně HTML5 a je zcela kompatibilní se starými CSS specifikacemi. CSS spravuje W3C (stejná firma, která spravuje i HTML).

Kaskádový styl je stylopis, množina pravidel, která se používá jako grafická nástavba klasického HTML. HTML neumožnuje tak rozmanitě nastavovat vzhled stránky a prvků na stránce.

Možnosti zápisu



Inline CSS

```
<h1 style="color: red">Tento nadpis bude červený.</h1>
```

Stylopis

Do hlavičky se napíše stylopis mezi tagy <style></style>.

```
<head>
  <title>Test</title>
  <style>
    p {color: red}
  </style>
</head>
```

Externí CSS soubor

Soubor s příponou .css

Do hlavičky se přidá:

```
<link rel="stylesheet" type="text/css" href=".//test.css" />
```

CSS soubor bude vypadat následovně:

```
p { color: red }
h1 { tag: hodnota}
```

Selektory

CSS selektory se používají k výběru (hledání) HTML elementu na základě jejich jména, ID, třídy (class)...

Element selector

Vybírá element na základě jejich jména.

Pokud mají být všechny elementy stejně nastylované stačí pouze:

```
p {
    text-align: center;
    color: red;
}
```

ID selector

Využívá ID atribut HTML elementu ke specifikování určitého elementu.

ID by mělo být unikátní (pro celou stránku), aby přesně identifikoval element.

ID by nemělo začínat číslem.

Identifikátor lze přiřadit pouze k jednomu elementu.

HTML:

```
<h1 id="para1"> TEST </h1>
```

CSS:

```
#para1 {
    text-align: center;
    color: red;
}
```

Class selector

“Třídní Selektor” vybírá elementy podle specifického **class** atributu.

HTML:

```
<h1 class="center"> TEST </h1>
```

CSS:

```
.center {
    text-align: center;
    color: red;
}
```

Popřípadě specifikace/oddělení jednotlivých elementů.

```
p.center {
    text-align: center;
    color: red;
}
```

Sjednocování selektorů

Pokud existuje více elementů se stejnými styly, lze je sjednotit.

```
h1 {
    text-align: center;
    color: red;
}

h2 {
    text-align: center;
    color: red;
}

p {
    text-align: center;
    color: red;
}
```

Místo dlouhého a zbytečného zápisu lze použít následující zkrácený zápis:

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

Pseudotřídy

Speciální události.

U odkazů lze nastavit, jak bude vypadat, když se přes něj přejede myší...

```
a:link {color: #FFE400; text-decoration: none;}
a:visited {color: #FFE400; text-decoration: none;}
a:active {color: #FF9000; text-decoration: none;}
a:hover {color: #FF9000; text-decoration: none;}
```

Pozicování

Pomocí CSS lze nastylit, jak budou jednotlivé elementy umístěny.

Existují 2 základní druhy pozicování:

Relativní

Určuje pouze, o kolik se má objekt posunout oproti své normální pozici

Absolutní

Umístí objekt do stránky bez ohledu na okolní text

Fixní

Speciální pozice, která udává, že objekt zůstává na stejném místě při rolování.

16. Práce se soubory v programovacích jazycích

Soubor

Pojmenovaná sada dat uložených na paměťovém médiu, se kterou lze pracovat jako s jedním celkem.

V programování slouží k uložení dat z programu nebo načtení dat do programu. Data ze souboru a do souboru putují přes streamy (proudy).

K souborům lze přistupovat dvěma způsoby a to **textově** nebo **binárně**. Rozdíl je v tom, jak se data čtou (zapisují).

Práce se soubory v jazyce JAVA (viz. [PRM 17](#))

Pro práci se soubory v jazyce JAVA slouží třída **File**. Zároveň slouží jako manažer souborů. Na začátku programu je nutné importovat knihovnu **java.io.File**.

Popřípadě **java.io.FileWriter**, **java.io.FileReader**, a jiné knihovny, které jsou potřeba. Popřípadě rovnou importovat celou input/output knihovnu **java.io***.

Aby bylo možné pracovat se soubory (textovými či binárními), musí se vytvořit **Streamy** (proudy). Stream je proud dat z programu na nějaké místo na disku, konkrétně k souboru, s nímž pracuje.

Stream může být vstupní pro čtení, nebo výstupní pro zápis, existuje i varianta, že lze číst a zapisovat do souboru pomocí jednoho proudu.

Streamy lze dál dělit dle toho, jaká data v něm proudí. Bajtové, znakové, datové, standardní, objektové a proudy vyrovnávací paměti.

Po dokončení práce se soubory, by se měly uzavřít všechny proudy, které byly otevřené. Hlavně výstupní proud. Pokud se neuzavře výstupní proud, data se neuloží.

Třídy pro práci se soubory

Základní třídy pro práci s textovými soubory jako jsou ***.txt**, ***.java**, ***.sql** a mnoho dalších jsou **java.io.FileReader** pro čtení a **java.io.FileWriter** pro zápis.

Obou třídám v konstruktoru lze předat buď objekt třídy File, nebo řetězec s cestou k souboru.

Při vytváření instance třídy FileReader může nastat výjimka **java.io.FileNotFoundException** pokud soubor neexistuje. Pokud soubor neexistuje a vytváří se instanci třídy FileWriter, tak se soubor vytvoří, ale může nastat **java.io.IOException** pokud soubor nelze vytvořit.

Práce se soubory v jazyce PHP

V jazyce PHP k vytvoření souboru slouží metoda fopen:

```
$soubor = fopen ("soubor.txt", "w+");
```

Místo w+ lze použít další hodnoty:

- **a** – otevře soubor pro přidání, pokud soubor neexistuje, PHP jej vytvoří
- **a+** - otevře soubor pro přidávání a čtení, pokud soubor neexistuje, PHP jej vytvoří
- **r** – otevře soubor pouze pro čtení.
- **r+** - otevře soubor pro čtení a zápis
- **w** – otevře soubor pro zápis, původní data budou ztracena. Pokud soubor neexistuje, PHP jej vytvoří
- **w+** - otevře soubor pro zápis a čtení, původní data budou ztracena, pokud soubor neexistuje, PHP jej vytvoří
- **x** – vyhodí error, pokud soubor existuje. Nemusí se existence souboru ověřovat pomocí **file_exists()**;
- **c** – zamyká soubor. Nemusí se použít flock, aby se zamkl soubor a další request nemohl zapisovat do stejného souboru.

Zavření souboru se provádí stejnou funkcí jako vytvoření. Po dokončení práce se souborem je rozumné jej zavřít.

```
fclose(id_souboru);
fclose($soubor); //fopen vrací ID souboru
```

Práce se soubory v jazyce VBA

V jazyce VBA slouží pro práci se soubory funkce:

```
Open nazev_souboru For rezim As #cislo
```

Režimy pro práci se soubory:

- Input
 - Jednosměrný vstup ze souboru do programu (čtení dat ze souboru)
- Output
 - Jednosměrný výstup z programu do souboru (zápis dat do souboru přepsáním, pokud již existuje soubor se stejným jménem)
- Append
 - Jednosměrný výstup z programu do souboru (zápis konec - přidávání na konec)
- Random
 - Náhodný přístup, čtení i zápis
- Binary
 - Binární režim pro čtení i zápis

Příkaz CLOSE

Close #cislo

- Uzavírá (ukončuje přístup) otevřený soubor s uvedeným číslem
- Bez uvedení čísla uzavírá všechny otevřené soubory v programu
- Dokončuje poslední zápisy z vyrovnávací paměti – vyprázdní buffer
- Přesunuje celý soubor z operační paměti na trvalé paměťové médium

Funkce EOF

Eof (cislo)

- EOF = End Of File (konec souboru)
- Funkce vrací logickou hodnotu
 - True – zda bylo dosaženo konce souboru
 - False – zda nebylo dosaženo konce souboru

Příkaz Input

Input #cislo, seznam

- Seznam identifikátorů jednoduchých proměnných (nikoli polí)
- Jednotlivé identifikátory jsou odděleny čárkou
- Data ve čteném souboru musí svým charakterem a polohou odpovídat typu jednotlivých proměnných v seznamu

Příkaz LINE INPUT

Line Input #cislo, promenna

- Přečte ze souboru jeden řádek zakončený znakem s kódem 13 (Carriage Return – Konec řádku, Návrat vozíku)
- Čteny jsou všechny znaky v daném řádku (včetně mezer, čísel, uvozovek a jiných znaků)
- Znaky konce řádku – Chr(13) a Chr(10) – jsou z řetězce vypuštěny
- Další příkaz přečte následující řádek

Příkaz GET

Get #cislo, zaznam, promenna

- Obdobně jako LINE INPUT přečte jeden řádek (záznam), ale lze specifikovat libovolný záznam (řádek v databázi)
- Cislo – manipulační číslo otevřeného souboru
- Zaznam – pořadové číslo záznamu (řádku), který se má přečíst

Příkaz PRINT

Print #cislo, seznam

- Zapisuje data do otevřeného souboru
- Oddělovače v seznamu
 - ; – znak se zapíše na další pozici
 - , – znak se napiše do další zóny (jedna tisková zóna je 14 znaků)
- Každý příkaz PRINT bude psát na nový řádek
- Oddělovač použitý na konci seznamu způsobí, že další příkaz PRINT bude zapisovat na stejný řádek (potlačení přechodu na další řádek)

Příkaz WRITE

Write #cislo, seznam

- Výrazy jsou v seznamu odděleny čárkou
- Každý příkaz WRITE píše na nový řádek
- Řetězcové hodnoty jsou ve výstupním souboru ohrazeny uvozovkami
- Jednotlivé hodnoty jsou ve výstupním souboru odděleny čárkou

Příkaz PUT

PUT #cislo, zaznam, promenna

- Výstup do souboru

Příkazy GET a PUT jsou vhodné pro jednoduchou obsluhu databází, neboť dovolují manipulovat se záznamy v libovolném pořadí na rozdíl od sekvenčního přístupu.

Práce s adresáři

ChDir cesta 'nastaví pracovní cestu

ChDrive disk 'změna aktuální jednoty

MkDir cesta 'vytvoří adresář

17. Datová struktura pole

Skupina proměnných obvykle stejného typu, které jsou v paměti alokovány za sebou. K těmto proměnným lze přistupovat pomocí jejich indexů (pořadí v této posloupnosti, počítáno od 0). Z tohoto způsobu alokace je zřejmé, že pole má fixní délku, kterou nelze nijak změnit (protože paměť v oblasti za polem může být obsazena jiným objektem).

Dělení Polí

- **Statické**

- Velikost je jistá před překladem, interpretací programu, obvykle indexované od 0

- **Dynamické**

- Velikost závisí na průběhu programu, a lze měnit i po jeho vytvoření
 - Pošle se požadavek na správce paměti a ten buď přidělí paměť, nebo oznámí chybu
 - V Javě nelze měnit velikost pole za běhu programu

- **Indexované**

- Přístup k prvku pomocí čísla, většinou od 0
 - Java má přístup k prvkům pouze pomocí indexu

- **Asociativní**

- Přístup pomocí řetězce
 - V PHP lze přistupovat pomocí řetězce

- **Homogenní**

- V poli může být jen jeden datový typ

- **Různorodé**

- V poli můžou být různé datové typy
 - V Javě pole nemůže být různorodé
 - V PHP pole může být různorodé

Vícerozměrná pole

Vícerozměrná pole v Javě jsou ve skutečnosti jen pole plné polí.

Při deklaraci vícerozměrných polí se postupuje obdobně jako u pole jednorozměrného, jediným rozdílem je počet závorek, který se uvádí.

Dvojrozměrné pole je časté, trojrozměrné je neobvyklé a více jak trojrozměrné je obvykle chyba návrhu aplikace (jenom málokdo si umí taková data představit).

```
Typ [ ] [ ] jméno = new typ [pocet_prvku] [pocet_prvku] ;
```

JAVA

V okamžiku, kdy se pole vytvoří, tak jsou jeho hodnoty přednastaveny do výchozího stavu. V případě integerů 0, u booleanu false, a v případě referencí ukazatel do prázdná null.

```
typ[] jmeno;
```

Vytvoří se proměnná pole, ale zatím se nealokuje místo v paměti.

Místo v paměti se alokuje konstruktorem.

```
jmeno = new typ[velikost_pole];
```

Inicializaci a deklaraci pole lze zapsat obdobně jako u ostatních typů.

```
typ[] jmeno = {hodnota, hodnota2, hodnota3, hodnota4};
```

```
typ[] jmeno = new typ[velikost_pole];
```

PHP

V PHP existuje spoustu syntaxí pro vytvoření pole.

```
$pole = array("test", 2, 2.16);
$pole1 = array(0 => "test", 1 => 2, 2 => 2.16);
$pole2 = array("pozice" => "test", "pozice2" => 2,
"pozice3" => 3.14);
$pole3 = ["test", 2, 2.16];
$pole4 = [0 => "test", 1 => 2, 2 => 2.16];
$pole5 = ["pozice" => "test", "pozice2" => 2,
"pozice3" => 3.14];
```

Jak je vidět na příkladech, pole v PHP je **asociativní a různorodé**.

V PHP lze vytvořit i vícerozměrné pole.

```
$a1 = array("a" => 0, "b" => 1);
$a2 = array("aa" => 00, "bb" => 11);
$together = array($a1, $a2);
```

VBA

Dim nazev(velikost) As Datovy_typ

Metody nad třídou Array:

Třída Array obsahuje pomocné metody pro práci s poli.

- **Sort**
 - Metoda pro třídění pole
- **Reverse**
 - Obrácení posloupnosti
- **IndexOf, LastIndexOf**
 - Tyto metody vrátí index prvního nebo posledního nalezeného prvku
- **Copy**
 - Zkopíruje část pole do jiného pole
 - Prvním parametrem je zdrojové pole, druhým cílové a třetím počet znaků, který se má zkopírovat

Metody nad polem:

Třída Array není jedinou možností, jak s polem manipulovat. Přímo na samotné instanci pole (konkrétní proměnné) lze volat také spoustu metod.

- **Length**
 - Vrátí délku pole
- **Min, max, average, sum**
 - Vracejí minimum, maximum, průměr, součet všech prvků
 - Bez parametrů
- **Concat, intersect, union**
 - Vrátí na výstupu nové pole
 - Jako parametr mají druhé pole
 - Spojení; Průnik
 - Sjednocení; prvky, které byly v obou, jsou zde pouze jednou
- **First, last**
 - Vrátí první, poslední prvek
- **Take, skip**
 - Berou jako parametr počet prvků
 - Take vrátí pole s daným počtem prvků zkopírovaných od začátku
 - Skip naopak vrátí pole bez těchto prvních prvků
- **Contains**
 - Vrací True/False podle toho, zda se prvek v parametru metody, nachází v poli
- **Reverse**
 - Vrátí nově vytvořené pole s otočenými prvky
- **Distinct**
 - Zajistí, aby byl v poli každý prvek jen jednou
 - Vrátí nově vytvořené pole

18. Vyhledávání

Způsob zjištění, zda je hledaný prvek přítomen v prohledávaných datech. Vyhledávání probíhá v organizované datové struktuře (pole, spojový seznam, arraylist, soubor...). Vyhledávací algoritmy typicky vrací logickou hodnotu (**true / false**), která indikuje, zda byl hledaný prvek nalezen. Nejčastěji prohledáváme pole.

Mezi parametry patří **složitost**. S ní související hardwarové nároky, nároky na vstupní data (struktura, datové typy, nutnost seřazení prvků).

Možnosti algoritmu jsou dány jazykem, ve kterém je programován. To ovlivňuje například datové struktury a typy, které mohou být prohledávány (neobjektové jazyky neumožní použití spojových seznamů, striktně typované jazyky neumožňují použití nehomogenní datové struktury).

Vyhledávací algoritmy

Brute Force (Naivní vyhledávání)

Algoritmus prochází celé pole, od prvního prvku až po poslední. U každého prvku kontroluje, zda není shodný s hledaným. I když najde shodu, prochází celé pole do konce.

Výhody

- Jednoduchost
- Aplikovatelný pro všechny datové typy a struktury
- Není třeba upravovat pole
- Jednoduchou úpravou lze získat další funkce

Nevýhody

- Prohledává celé pole

Složitost = N

```
public static boolean bruteForce(int[] array, int search) {
    int count = 0;
    boolean result = false;
    for(int i = 0; i<array.length; i++) {
        if(array[i] == search) {
            result = true;
            count++; //počítání výsledků
        }
    }
    System.out.println(count);
    return result;
}
```

Vyhledávání bez zarážky

Algoritmus prochází pole, dokud nenajde hledaný prvek. Když najde shodu, pole dál neprochází.

Výhody

- Jednoduchost
- Aplikovatelný pro všechny datové typy a struktury
- Není třeba upravovat pole

Nevýhody

- Dva testy v každém kroku (zda nebyl nalezen prvek a zda není konec pole)

Složitost = 2^*N

```
public static boolean withoutStop(int[] array, int search) {  
    boolean result = false;  
    int i = 0;  
    while((i < array.length) && (result == false)) {  
        if(array[i] == search) {  
            result = true;  
        }  
        i++;  
    }  
    return result;  
}
```

Vyhledávání se zarážkou

Aby byla vyloučena nutnost dvou testů bez rizika, že se „sáhne“ mimo pole, přidá se na konec pole další prvek, který bude stejný jako hledaný.

Pokud se přidává další prvek jako **zarážka** je nutné pole překopírovat do nového a většího.

Poté stačí jediný test pro každý krok, protože je zaručené, že prvek bude vždy nalezen a vyhledávání skončí.

Nakonec je nutné zjistit, zda byl hledaný prvek v původním poli. Pokud byl hledaný prvek nalezen na posledním indexu, znamená to, že v původním poli nebyl.

Výhody

- Jednoduchost
- Aplikovatelný pro všechny datové typy a struktury

Nevýhody

- Nutná úprava pole

Složitost = N+1

```
public static boolean withStop(int[] array, int search) {
    int newArray[] = new int[array.length+1];
    System.arraycopy(array, 0, newArray, 0, array.length);
    newArray[array.length] = search;
    boolean result = false;
    int i = 0;
    while (result == false) {
        if (newArray[i] == search) {
            result = true;
        }
        i++;
    }
    if (i == newArray.length) {
        return false;
    } else {
        return true;
    }
}
```

Binární vyhledávání

Nejdříve se otestuje prvek uprostřed. Je-li nalezena shoda, vyhledávání skončí.

Pokud ne, porovná se velikost a podle ní se rozhodne, ve které polovině pole se bude dále vyhledávat.

Z vybrané poloviny se vybere prostřední prvek a vyhledávání pokračuje až do doby, kdy je prvek nalezen, nebo než už pole dále nejde dělit.

Zde se vytvoří dva indexy do pole, **levý**, který ukazuje na první prvek pole, a **pravý**, který ukazuje na poslední.

Z těchto se poté počítá prostřední prvek a tento prvek se porovná s hledaným prvkem.

Pokud se nerovnají, pak se buď levý index posune doprava, nebo pravý doleva. Toto se opakuje, dokud se neprekříží.

Výhody

- Efektivnost

Nevýhody

- Nutná úprava pole (seřazení)
- Pouze pro číselné datové typy

Složitost = $\log_2(N)$

```
public static boolean binary(int[] array, int search) {
    int left = 0;
    int right = array.length - 1;
    do{
        int middle = (left + right) / 2;
        if(array[middle] == search) {
            return true;
        }else if(array[middle] > search) {
            right = middle - 1;
        }else{
            left = middle + 1;
        }
    }while(left <= right);
    return false;
}
```

19. Řadící algoritmy I (Bubble, Select, Insert)

Řadící algoritmy zajišťují seřazení daného souboru dat do specifikovaného pořadí. Nejčastěji se řadí podle velikosti čísel nebo abecedně.

Bubble Sort

V poli se porovnávají 2 sousední prvky.

Pokud je číslo vlevo větší (menší) než vpravo, prohodí se.

Po průchodu celým polem je seřazený minimálně 1 prvek. Pole se o seřazený prvek zkrátí a pokračuje znova, dokud pole není seřazeno.

Počet porovnání = $(n^2 - n) / 2$

Počet přesunů = přibližně n^2

Složitost = n^2

```
public static void bubble(int[] array) {
    for (int i = 0; i < array.length-1; i++) {
        for (int j = 0; j < array.length-1-i; j++) {
            //if(array[j] < array[j+1]) { směr řazení
            if(array[j] > array[j+1]) {
                int tmp = array[j];
                array[j] = array[j+1];
                array[j+1] = tmp;
            }
        }
    }
}
```

Select Sort

Najdeme maximum (minimum) v poli.

Tento prvek se prohodí s prvkem na posledním (prvním) místě.

Postup se opakuje a po každém průchodu bude vynechán poslední (první) prvek.

Počet porovnání = $(n^2 - n) / 2$

Počet přesunů = přibližně $n-1$

Složitost = n^2

```
public static void select(int[] array){  
    for (int i = 0; i < array.length-1; i++) {  
        int max = pole.length-i-1;  
        for (int j = 0; j < array.length-i; j++) {  
            //if(array[j] < array[max]) { směr řazení  
            if(array[j] > array[max]) {  
                max = j;  
            }  
        }  
        int tmp = array[array.length-1-i];  
        array[array.length-1-i] = array[max];  
        array[max] = tmp;  
    }  
}
```

Insert Sort

Postupně prochází prvky a každý nesetříděný se zařadí na správné místo.

K tomu je nutné mít pole již částečně setříděné. To se provede tak, že se první prvek považuje za setříděný a začne se od druhého.

Počet porovnání = $(n^2 - n) / 2$

Počet přesunů = přibližně $(n^2 - n - 2) / 2$

Složitost = přibližně n^2

```
public static void insert(int[] array) {
    for (int i = 0; i < array.length; i++) {
        int tmp = array[i];
        int j = i - 1;
        //while ((j >= 0) && (array[j] < tmp)) {směř řazení
        while ((j >= 0) && (array[j] > tmp)) {
            array[j+1] = array[j];
            j--;
        }
        array[j+1] = tmp;
    }
}
```

20. Řadící algoritmy II (Quick - sort)

Jeden z nejrychlejších běžných algoritmů řazení porovnávání prvků. Vymyslel jej Sir Charles Antony Richard Hoare v roce 1962.

Quick Sort

Rozdělí řazené posloupnosti čísel na dvě přibližně stejné části.

V jedné části jsou čísla větší a ve druhé menší, než nějaká zvolená hodnota (pivot).

Pokud je tato hodnota zvolena dobře, jsou obě části přibližně stejně velké.

Pokud budou obě části samostatně seřazeny, je seřazené i celé pole.

Obě části se pak rekurzivně řadí stejným postupem (Lze zapsat i iteračně).

Složitost = n^2 (nejhorší možný případ)

```
public static void quick(int[] array, int left, int right) {
    int i = left;
    int j = right;
    int pivot = array[(left+right)/2];
    do{
        //while(array[i] > pivot) i++;
        //while(array[j] < pivot) j--;
        while(array[i] < pivot) i++;
        while(array[j] > pivot) j--;
        if(i <= j) {
            int tmp = array[i];
            array[i] = array[j];
            array[j] = tmp;
            i++;
            j--;
        }
    }while(i<j);
    if((j-left) > 0){
        quick(array, left, j);
    }
    if((right-i) > 0){
        quick(array, i, right);
    }
}
```

Lomuto Quick Sort

Nico Lomuto v roce 1984 přišel s vylepšením Quick Sortu. Principem je postupné umisťování prvků menších jak pivot doleva. Levá půlka pole se zvětšuje. Výhodou je méně porovnání, ale nevýhodou je více přesunů než Hoareho metoda.

Složitost závisí na zvolení vhodného pivota, může degradovat na N^2 .

21. MySQL – DDL

DDL

Příkazy DDL (**Data Definition Language**) slouží pro definici dat.

Umožňují:

- Přidávat
- Upravovat
- Mazat logické struktury, které obsahují data (databáze, tabulky, indexy, pohledy,...)

CREATE

Slouží pro vytvoření databázových objektů. Všechny jeho možnosti syntaxe se mohou lišit podle typu databáze.

```
CREATE DATABASE `knihovna` DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci;

CREATE TABLE kniha(
    id INTEGER NOT NULL AUTO_INCREMENT,
    nazev VARCHAR(50) NOT NULL,
    autor VARCHAR(50) NOT NULL,
    nakladatelstvi VARCHAR(50) NOT NULL,
    rok_vydani YEAR(4) NULL,
    isbn VARCHAR(13) NOT NULL,
    PRIMARY KEY(id),
    UNIQUE KEY(isbn)
) ENGINE = InnoDB DEFAULT CHARSET utf8;
```

ALTER

Slouží ke změně databázových objektů.

Přidání sloupce

```
ALTER TABLE kniha ADD COLUMN k_sehnani SET ('ano', 'ne', 'mozna')
NOT NULL;
```

Změna nastavení sloupce

```
ALTER TABLE kniha CHANGE COLUMN k_sehnani je_na_sklade ENUM('ano',
'ne') NOT NULL;
```

Odstrannění sloupce

```
ALTER TABLE kniha DROP COLUMN je_na_sklade;
```

DROP

Slouží k odstranění databázových objektů.

```
DROP DATABASE 'knihovna';
```

```
DROP TABLE `kniha`;
```

Datový typ	Délka	Rozsah
INT, INTEGER	4	-2 147 483 648 až 2 147 483 647
FLOAT		-3.402823466E+38 až 3.402823466E+38
DOUBLE		-1.7976931348623157E+308 až 1.7976931348623157E+308
DATE	'0000-00-00'	"RRRR-MM-DD"; 1000-01-01 až 9999-12-31
DATETIME	'0000-00-00 00:00:00'	"RRRR-MM-DD HH:MM:SS"); 1000-01-01 00:00:00 až 9999-12-31 23:59:59
TIME	'00:00:00'	"HH:MM:SS"; "-838:59:59" až "838:59:59"
YEAR(n)	0000	
CHAR(n)		Pevná délka; 0-255
VARCHAR(n)		0-255
BLOB, TEXT		maximálně 65 535 znaků
LONGBLOB, LONGTEXT		maximálně 4 294 967 295 znaků
ENUM('item1','item2',...)		pole předem definovaných řetězců o maximálním počtu 65 535

22. MySQL – DML

DML

Příkazy DML(Data Manipulation Language) slouží pro získání dat z databáze a pro jejich úpravy.

INSERT

Přidá do tabulky nový záznam.

Vložení více záznamů na jednou

```
INSERT INTO kniha (id, nazev, autor, nakl, pocet, rok, isbn, precteno)  
VALUES  
    (NULL, 'Così', 'Kdosi', 'Kdesi', 500, 1978, '0-13-110163-3', 'ano'),  
    (NULL, 'Nekdo', 'Ona', 'Kdesi', 460, 1998, '0-13-659723-8', 'ano');
```

INSERT IGNORE INTO

Při pokusu o vložení řádku shodujícího se v primárním či unikátním klíči potlačí standardní chování databázového stroje (hovení chybové hlášky a skončení jako celku), místo toho bude tuto chybu ignorovat a pokračovat dál. Má význam zejména v případě vkládání více záznamů – po skončení příkazu budou vloženy jen ty nekolidující řádky, ty kolidující — duplicitní vloženy nebudou.

UPDATE

Upravuje data (záznamy) v relační databázi. Může být upraven jeden záznam, nebo i více záznamů najednou. Upravené záznamy musí odpovídat definované podmínce.

```
UPDATE kniha SET pocet_stran = 236 WHERE id = 1;
```

DELETE

Slouží k odstranění záznamů z tabulky.

WHERE je volitelný parametr, bez udání smaže všechny záznamy v tabulce, ale ne samotnou tabulku.

```
DELETE FROM kniha WHERE id = 1;
```

23. MySQL – SQL

SQL

Patří do skupiny příkazů pro manipulaci s daty (**DML**). Vybírá data z databáze a umožňuje výběr podmnožiny a řazení dat.

```
SELECT * FROM cosi;
```

Specifikace

WHERE

Slouží k filtrování dat. Uvede se podmínka, pokud je splněna, vyberou se data

```
SELECT * FROM cosi WHERE id=1;
```

LIKE

Oproti WHERE nabízí použití „patternů“, zástupných znaků.

JOIN

Pokud se vybírají data z více tabulek.

ORDER BY

Seřazení dat. Specifikace sloupce (podle, kterého se má řadit) a směr řazení ASC, DESC

```
SELECT nazev, zanr, rezie FROM cosi WHERE rezie = 'Menzel' ORDER BY nazev DESC;
```

GROUP BY

Seskupení dat. Nejběžnější použití je získání počtu záznamů odpovídající každé jednotlivé hodnotě jiného sloupce, časté je také získání součtu, aritmetického průměru či jiných statistických hodnot z vybíraných záznamů

```
SELECT zanr, COUNT(*) FROM filmy GROUP BY zanr;
```

DISTINCT

Nevypisuje duplicitní záznamy

Agregační funkce

COUNT – počet

SUM – součet

MIN – minimum

MAX – maximum

AVG – průměr

24. PHP – Formuláře a jejich zpracování

Formuláře

Důležitá součást stránek, slouží k přihlašování k některým stránkám (eshopy, bakaláři, diskuze na webu apod.), k odeslání údajů do databáze nebo k vyplnění anketních otázek. Samo o sobě formuláře nejsou součástí PHP, ale už HTML jazyka.

Formulář je definován párovým tagem `<form> . . . </form>`. Formuláře se používají k odesílání dat na server, který je zpracuje. Jedna HTML stránka může obsahovat žádný, jeden, nebo libovolně mnoho formulářů na jedné stránce. Důležité atributy tagu `<form>` jsou **ACTION** (skript, který zpracovává data, určuje, kam se budou data posílat, není-li uvedeno, odešlou se data též stránce) a **METHOD** - určuje jakým způsobem se budou data posílat.

Způsoby posílání dat

GET

Data se budou předávat jako součást URL.

POST

Data jsou součástí dotazu.

Vhodné pro posílání hesel a obsáhlých dat...

Vstupní pole

Pro zadávání hodnot od uživatele se používá nepárový tag `<input />`. Input jako takový má spoustu atributů pro různé typy vstupů. Předávají se mu určité důležité atributy:

- Type – Specifikuje, jak bude input vypadat.
- Name – Jméno, kterým se dané pole identifikuje
- Value – Výchozí hodnota, která je předvyplněná

Text

Pro zadávání textových vstupů od uživatele.

Password

Pro zadávání textových vstupů uživatele, které nebudou čitelné.

Checkbox

Slouží pro výběr. Výběr N hodnot z M. Pokud není checkbox zatrhnut, neodesílá se.

Radio

Slouží pro vybrání 1 možnosti z N možností; Checked

Select

Slouží pro vybrání 1 možnosti z N možností pomocí rolovací nabídky; Nepárový tag option (value) mezi tagama se udá, co má uživatel vidět.

Submit

Slouží k odeslání dat vyplněných ve formuláři. Atribut value → Nápis, který bude na tlačítku.

Odesílá se vždy, pokud má atribut name.

Superglobální proměnné

PHP používá zvláštní pole, ve kterých jsou uložena data.

- **\$GLOBALS** – všechny běžně používané proměnné v PHP (i všechny superglobální pole; \$GLOBALS včetně)
- **\$_SERVER** – různé systémové proměnné, údaje o serveru, uživateli...
- **\$_ENV** – proměnné poskytované skriptu z prostředí
- **\$_POST** – proměnné poskytované skriptu přes **HTTP POST**
- **\$_GET** – proměnné poskytované skriptu přes **HTTP GET**
- **\$_COOKIE** – proměnné poskytované skriptu přes **HTTP COOKIES**
- **\$_FILES** – proměnné poskytované skriptu přes HTTP POST upload souborů
- **\$_REQUEST** – proměnné poskytované skriptu přes libovolný vstupní mechanismus (POST, GET...)
- **\$_SESSION** – proměnné registrované pro aktuální session

25. PHP – Cookies a sessions

Sessions

Pojem představující permanentní síťové spojení mezi klientem a serverem, zahrnující výměnu paketů.

Jakmile PHP obdrží příkaz k započetí session, zjistí nejprve, zda již session neběží. Pokud ne vytvoří ji, pokud ano, připojí se k ní. PHP přidělí session identifikátor a vyhradí si někde místo pro ukládání tzv. session-proměnných. Od tohoto místa dále lze u libovolné proměnné zvolit, že bude součástí session, a server si pak její obsah pamatuje mezi stránkami. Session může být kdykoli ukončena. Když se to nepovede, zruší se zavřením prohlížeče.

Pro zapnutí práce se sessionama slouží příkaz `session_start()`; a pro zapnutí outputbufferingu `ob_start()`;

Pro přístup k sessionám se používá superglobální pole `$_SESSION['nazev']`;

Pro smazání sessiony lze použít `unset($_SESSION['cosi']);`, `session_unset()`;, `session_destroy()`;, `$_SESSION = array();`

Cookies

V protokolu HTTP se jako **cookies** označuje malé množství dat, která WWW server pošle prohlížeči, který je uloží na počítači uživatele. Při každé další návštěvě téhož serveru pak prohlížeč tato data posílá zpět serveru.

Cookies se odešlou s požadovanou stránkou a to v hlavičce. Ukládají se do souboru na disku (můžou být hashovány). Když se potom prohlížeč připojí na stejný server, bude automaticky posílat cookies.

Na začátku je nutno zadat `ob_start()`. Pak se pracuje stejně jako se sessions, ale nastavení cookie se dělá metodou `SetCookie (nazev_cookie, hodnota ($hodnota), [platnost (time () + 3600)])`. Na závěr aplikace je nutno zadat `ob_end_flush()`.

26. PHP – Propojení s DB

K databázi se připojuje, pokud je třeba pracovat s větším množstvím dat. Databáze je prakticky jediným rozumným řešením. Díky SQL je práce s daty podstatně rychlejší a poskytuje širokou škálu nástrojů pro jejich spravování. Pomocí PHP se poté jen odešle zformulovaný dotaz a získá se již zpracovaný výsledek.

Pokud se jedná o jednodušší projekt, ve kterém bude třeba připojovat se k DB jen v jediném souboru, je možné skript definující parametry připojení (a realizující připojení samotné) vložit přímo do něj. U větších projektů se však vyplatí pro tento účel vytvořit externí skript, který lze pak pokaždé snadno vložit.

Komunikace s DB pomocí `mysql_*()` je už zastaralou variantou a v novějších verzích PHP plánují vývojáři jejich podporu odstranit. Jejich funkce už v současnosti nahrazuje PDO_MySQL.

```
<?php
    /**Data source name
     * Typ DB
     * Adresa server
     * Port serveru
     * Jméno DB, ke které se připojuje
     */
    $dsn      = "mysql:host=127.0.0.1;port=3306;dbname=telefony";
    /**Uživatel, který může spravovat Db*/
    $user     = "root";
    /**Heslo uživatele*/
    $password = "";
    /**Vytvoření instance PDO
     * PDO::ERR_MODE_EXCEPTION → chyby se ohlašují jako výjimky
     * PDO::MYSQL_ATTR_INIT_COMMAND → Kódování
     */
    try{
        $db = new PDO($dsn, $user, $password,
            [ PDO::ATTR_ERRMODE          => PDO::ERRMODE_EXCEPTION,
              PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8',
              PDO::ATTR_PERSISTENT        => TRUE ] );
    } catch(PDOException $e) {
        echo 'Connection failed: ' . $e->getTraceAsString();
    }
?>
```

V souborech, kde je třeba se připojit k DB, tak se pouze vyžádá soubor s připojením.

Důležité funkce pro práci s DB

Prepare

Příprava SQL dotazu. Používají se Placeholdery, za které se poté dosadí přesná data.

Params

Pole parametrů, které se dosazují do dotazu.

Execute

Spuštění dotazu.

Fetch

PDO::FETCH_BOTH, PDO::FETCH_ASSOC...

Vrací pole, které představuje řádek, s parametry.