Udite 4

jarek.kligl7

June 2024

1 Databázové modely

Data
Data = **informace**

- Jak uložit?
  struktura, velika **absrtakce**
- Jak spravovat? (čtení, zápis, více než jeden přístup, oprávnění, analýza)
- Vlastní implementace vs. existující řešení
  v pripade komplexnejsich dat exist reseni, bezpecnost atd
  dan za to je zase ta univerzalnost

Typ dat
int, string ...
urcuje pripustne operace (divat se na to skrz abstrakce)
treba cislo - ma moznost prumer, retezec ma delku

Struktura dat
jakou maji podobu (jednotlive polozky ktere chceme uchovavat) Zakaznik

- jmeno
- vek
- telefon

Database
Kolekce dat nejake konkretni struktury

"Perzistentní" uložení
ulozena na pevno, nejlepe v podobe ktera zabranuje poskozeni dat

Formální model - matemtaticku popis typu dat, struktury dat, ...
umoznuje o datech dokazovat zaveri, muzeme odvozovat ruzne operace
vs. implementace - konkretni software (ktery je podle formalniho modelu)

MariaDB -relacni model dat
ElastickSearch - dokumentova database

Database Management System (DBMS)

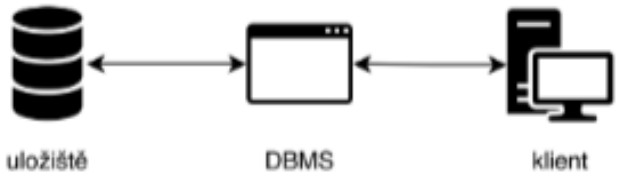

Figure 1: Enter Caption

DataBase Management System (DBMS)
Systém báze řízení dat, - offico Czechia preklad
systém pro správu dat
implementuje konkrétní model dat

velice komplikovane programy, nekdy v rozsahu OS
Služby:

- Ukládání dat, transakční zpracování a zabránění vzniku chyb (žurnálování) v podobe transakci (stejne jak u OS)
- Integritní omezení (konzistence dat)
  omezuji strukturu dat abych mel jenom validni data
  treba to omezi vlozeni zaporneho veku
- Víceuživatelský přístup (neblokující)
  v jeden okamzik muze k datum pristupovat k datum vic jak jeden uzivatel
- Bezpečnost dat (oprávnění, šifrování)
  Role - admin , user, atd..., maji ruzny pristup k ruznym datum treba user asi nebude mit pristup k telefonu zakazniku

Databázové schéma = struktura dat

Různé role: vývojář, administrátor, uživatel
Vyvojari- programatori ktreri spravuji program
spravce (admin) - jazyky ktere ten clovek ma

Jazyk pro definici dat (DDL) - urcuje strukturu dat, jazyk pro modifikaci dat (DML) - pro zmeny dat, nebo modifikace struktury dat, dotazovací jazyk (QL) - pro ziskani dat

Příklady: MariaDB - mySQL database, PostgreSQL, SQLite - samostojice aplikace, MongoDB - dokumentova db, Microsoft SQL Server, Oracle
Lisi se v Cene a primarnim ucelem a tim modelem dat a filozofii

Modely dat

- Souborový
  - Historicky první databáze
  - **Záznamy uloženy v souborech a adresářích**
    Mam slozku obchod mam tam podslozku objednavky , zakaznici
  - Velmi jednoduché = snadno implementovatelné
  - **Absence abstrakce** a mnohdy DBMS
    Treba vrat mi prumerny vek vsech zakazniku by bylo hodne komplikoavne - bych musel otevrit vsecky soubory preparsovat data atd...
- Síťový
  - Dnes již málo používaný
  - Záznamy provázány odkazy
  - Grafová struktura
    uzlu jsou zaznamy a jsou propojeny odkazy
  - mam treba seznam zakazniku a mam je propojene odkazem
  - Znovu objevení v podobě grafových databází (zmíníme později)
- Hierarchický
  - Varianta/zjednodušení síťového modelu
  - **Stromová struktura**
  - **Vztah potomek-rodič**
  - 1:N, rodič může mít více potomků, potomek má právě jednoho rodiče
  - vyhoda zjednoduseni struktury
  - nevyhoda nejdou zachitit vsechny typy tech vztahu
  - slo by v tom udelat sisttem DNS - ale neni tak
- Relační
  - Relace
    zakaznik = $< jmeno, vek, telefon >$ (n-tice)
  - Data uložena v tabulkách
  - Nejpoužívanější (nad 80 proceny klidne)

  - **Obvykle ve spojení s objektovým modelem** = objektově-relační databáze
    podrtzeni vyhod , vyhodne pro programatora

- Objektový
  - Data chápeme jako objekty
  - proste entita ktera ma vlastnosti (jmeno, vek, telefon)
  - da se velice snado spojit s tim co dela programator
    hodne lidi umi objektove programovat
- Další modely

Popularni deleni:
souvisi to s dominantim procentem relacnich db na trhu

- SQL = Relacni databaze
- NoSQL = nerelacni databaze (Ostatni)

CSV format
Comma separeted values
Data **uloženy v souborech** (obvykle přípona `.csv`)
uplne nejzakladnejsi format

- Pevná struktura (hlavička) - 1. radek (nepovina)
  kazdy zaznam musi respektovat tu strukturu
- Záznamy uloženy na řádcích
- Hodnoty odděleny oddělovačem (obvykle čárka, ale i jiný znak)
  oddelovac se nesmi objevovat v polozkach
  prazda nohnota - se dela ze se nic tam nenapise
- **Žádná abstrakce**, omezené dotazování
- Univerzální formát, velká podpora

Klic-hodnota
Data ve formátu:

- **Klíč → Hodnota**
- Klíč = omezený datový typ (int, string)
- Hodnota = libovolná (i strukturovaná) (treba pole nebo dalsi struktura klic-hodnota)

Operace: ulož, přečti, smaž

- Omezené (zejména dotazování), ale velmi populární
- uloz (na konktetni klic)
- precti (precti hodnotu podle klice)
- smaz (podle klice)

Použití:

- Uchování nastavení (pouzit vychozi nastaveni ano, ne)
- Kešování
  duvod rychlost
- Menší databáze (we webovem prohlizci pomoci JS)
  mobilni aplikace

Návrh klíče:

- Jedinečný identifikátor (číselný, textový)
- Hash
  Haschovaci tabulky treba
- Časové razítko (time-stamp)
  Jedinecny identifikator

Zavedení struktury → namespace
struktura nad klicema - zakaznik:prijmeni:1

Semi-strukturovana data
Poměrně vágní pojem **data** mají strukturu, ale **ne tabulkovou**: aby nebyla moc velka ta tabulka - to zahlavi

- Samo-popisující
- **Struktura dat je součástí dat samotných**

Příklady:

- **XML**
- JSON
- YAML (intendace)
- BSON (binarni json)

XML Extensible Markup Language (XML)

- Textový formát
- Původně zamýšlen pro službu WWW
  popis strukturu webovych stranek (puvodne)
  strata benevolence jazyka HTML
  z duvodu programatorskeho pohodli se nechytlo
- Neujal se
- Ukladani semi-strukturovanych dat
- Univerzální, platformově nezávislý a značně rozšířený formát
  vetsina jazyku nabyzy nejakou funkcionalitu jak zpracovavat XML data
- nazvy xml elemntu si muzeme vymyslet sami
- hodne striktni
- neuzavreny element automaticky data neplatna
- mnohem obecnejsi nez XML
  XML umoznuje vytvorit jazyk pro popis konkretnich dat

Příklad:

```
<?xml version="1.0" encoding="UTF-8"?>
<poznamka>
  <od>Dave Barry</od>
  <nadpis>Database</nadpis>
  <telo>the information you lose when your memory crashes</telo>
</poznamka>
```

Podobnost s jazykem HTML:

- XML je více striktní
- XML je obecnější

HTML je teoreticky konkretni prikald jazyka HTML
Atributy v XML
kritizovana zalezitost, lepsi v prodobe dalsiho elementu
sintakticky cukr
ale komplikovanejsi zpracovavani
atributy na metadata
Omezení struktury dat → **XML schéma**
rika ze treba zpraba musi obsahovat elemnt od
urcuje poradi elementu atd
Validita - struktura odpovida schematu (zachovani integrity dat)
Podpora (knihovny) v programovacích jazycích

Priklad XML schema

```
<?xml version="1.0"?>

;;Povine a od te doby to je XML schema
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="poznamka">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="od" type="xs:string"/>
        <xs:element name="nadpis" type="xs:string"/>
        <xs:element name="telo" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Xml je komplikovana technologie

transformace XML → jazyk XSLT
transformacni jazyk
mam data v jazyce XML chci je zobrazit na webpage, pouziji jazyk XSLT ktery mi to prevede transoformaci do HTML Pomoci sablony $< P > [OD] < /p >$
dotazování → jazyk XPath
pozoruhodne slozity dotazovani, dokaze vybrat konkretni casti dokumentu
Treva vrat mi vsechny nadpisy, vrat mi vsechny jmena ktere maji poznamku
velmi podobne adresarove ceste
//poznamka/od[od zacnina na pismeno p] ...

parsování
precteni souboru (v pripade csv je to cteni po radku tady je to komplexnejsi)

- DOM (Document Object Model)
  stromova struktura, musi se nacist cely do pameti aby se s tim dalo pracovat
- SAX (Simple API for XML)
  pokud ctu data daneho typu tak neco udelej, ctu to postupne a v moment co neco chci tak to dostanu atd...
  staci jen malinkata cast v pameti
  mene pameti ale potrebuje vice rezie - je to pomalejsi

Vybrat podle velikosti dat

JSON
JavaScript Object Notation

Odlehčený formát (kontrast s XML)
V XML je verbozita (musime napat hodne)
popisuje jenom strukturu dat
Syntaxe:

- Data ve tvaru
  `"jméno" : "hodnota"`
  Klic hodnota
- Data se oddělují čárkou
- Složené závorky pro objekty
- Hranaté závorky pro pole

Omezení struktury dat → JSON schéma
podobne jako u XML
degraduje tu odlehcenost nemusi se to pouzivat
zmensuje rychlost trohcu

Obvykle znama struktura dat treba v CSV nebo XML nebo v JSON schematu

```
{
    "firstName": "Oliver",
    "lastName": "Queen",
    "alterEgo": [
        { "name": "The Hood" },
        { "name": "The Arrow" },
        { "name": "Green Arrow" }
    ]
}
```

Grafove databaze
Uzly = záznamy, **hrany** = relace.
oproti sitovym tak hrany nejsou dokazy ale muzou nest mnohem vetsi informacni hodnotu

Jde tam pouzit DFS a dalsi grafove algorithmy - takto se realizuji operace

najit spolecne pratele - vezme 2 uzly a provede graf algo
u tabulkovych dat by se musely projit vsecky data
Velmi flexibilní
ideální v situacích, kdy data mají povahu sítě
například: **sociální sítě**, sémantický web, doprava.

Dokumentove database
halvni duvod pro NoSQL datrabaze
Data = sada dokumentů
Ideální v situacích, kdy záznamy nemají pevnou strukturu
vztahy mezi dokumenty na základě identifikátoru **(primární klíč)**

ne vsehno se da napasovat do Semi-strukturovanych dat nebo tabulkovych
Například: webové stránky. - kazda stranka ma strukturu naprosto odlisnou
vemze se WebPage popise se v Json Formatu a ulozi se do dokumentove database
a ulozim si vztahy mezi tema dokumentama treba odkazi

Treba webovy vyhledavac

Oranizace ontologijich (mam data nejakleho typu a ty jsou podmnozinou jineho typu)
Priklad MongoDB , elastick Search

2 Relační databáze

Revoluce
utlacili predchozi modely a stali se dominantnimy **Relace** - usporadana N-tice,
to tvori jednotlive polozky v databasi
vice n-tic pod sebou = tabulka **Benefity:**

- Eliminace duplicit
  Duplicitni data se tezko dodrzuji
- Zajištění konzistence dat
  velka vyhoda je ze nima stoji nejaka algebra

Data uložena v tabulkách
Sloupec = vlastnosti (atributy) záznamu
Doména atributu = přípustné hodnoty
Různá omezení atributu (typ, jedinečnost, ...)

Schéma databáze
Řádek = záznam v databázi (relace)
Řádky nesmí porušovat databázové schéma (lze chápat jako výhodu i nevýhodu)

Normalizace databáze → snížení redundancy a zvýšení integrity dat

Jmeno	telefon	datum	cena
Tomas Kuchar	6969	11 zari	5000 00
Iva Brunclikova	7231	7 dubna	20
Adela Pulcova	0000	1 rijna	33

Příklad
Radek = N-tice
Ma sloupce - predstavuji vlasnosti - jmena atributu Mnozina vsechn pristupnych hodnot = domena
Tvorito relaci nad kartezkym soucinem vsemi moznima hondotama tech sloupcu

integritini omezeni + jmena atributu = schema databaze (relacni schema) integritni omezeni , typ, neprazdnost atd...

Primary-key = jedinecny idetifikator (v tomto pripade to muze datum+castka) ale co kdzy si Hait objedna jidlo ve stejny den za stejnou castku jako Brunclik???? muzeme si zavest ID - Coz by byl primarni KEY

Redundance v tech datech - duplicity , musim projit vsechny data abych to vsude upravil
Nomralizace - normalni forma - zajistena Integrita a snizena redundance

- Mnoho modelu normalizace
- muzu si zavest tabulku 2. [ID, Jmeno, Tel]
- tabulka 3. [ID, objednavka, datum, cena, id-zakaznika]
- pouziti Jineho ID v tabulce se oznacuje Cizi-klic a tim je propojim
- dalsi tabulka s [id, produkt]
- treba dalsi tabulka s [ID-obejdnavky, ID-produktu] - **propojeni dalsi pomoci Cizich klice a Ciziho KLICE**

Nutne prevedeni do neredundanti podoby - pro lepsi praci
Primarni klic a Cizi klic - myslenka

Integritni omezeni - nesmim v zadnem pripade to porusit
nepovoli ty data ta databaze vlozit

2.1 Jazyk SQL

Dotazovací jazyk
Vysledek ve formne relace

Výběr dat

```
SELECT jmeno_sloupce [, jmeno_sloupce, jmeno_sloupce, ...]
FROM tabulka WHERE podmínky
```

- * pro výběr všech sloupců
- WHERE část je nepovinná
- tab1.jemno jde taky ve from muze byt tabulek vice a tim jasne rikam z jake tabulky

Agregace dat
nad jednim konkretnim sloupcem

```
funkce(jmeno_sloupce)
```

- například: min, max, sum, avg

```
SELECT sum(cena) from tab1
```

Setřídění, limitace, seskupení (píše se za `WHERE` podmínku)
Order by - je setrizeni podle nejakeho atrbutu, Group by - je seskupeni podle nejakeho atributu limit n - je limitace kolik jich je maximalne vraceno

```
ORDER BY jmeno_sloupce
LIMIT n
GROUP BY jmeno_sloupce

SELECT jemno, sum(cena) FROM tab1 GROUP BY jmeno --
Vysledeke bude
```

Spojení dat (vnitřní)

```
SELECT tab_1.sloupec_1, tab_2.sloupec_2
FROM tabulka_1 JOIN tabulka_2
ON tab_1.sloupec_1 = tab_2.sloupec_2 WHERE podmínky
```

- Mam tabulku jedna , mam tabulku 2 , vnitrni spojeni vybere prunik na zaklade stejne hodonty atributu

```
SELECT tab2.jemno , tab3.cena FROM tab2 JOIN tab3 ON tab2.id == tab3.id-z
```

Vysledek tabnulka s hlavickou: [JMENO — CENA] , Kde tab2.id = tab3.id-z

Prave , leve a uplne
Propojeni na zaklade primarniho klice a ciziho klice

- Vnitřní, levé, pravé, plné, kartézské spojení

Vytvoření tabulky

```
CREATE TABLE zakaznik (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    jmeno TEXT,
    telefon INTEGER NOT NULL UNIQUE
);
```

NULL, UNIQUE - jsou omezeni integrity dat
UNIQUE omezuje duplicity
AUTOINCREMENT - samopopisujici

Vložení řádku do tabulky

```
INSERT INTO zakaznik (
    id, jmeno, telefon
) VALUES (
    '11', 'Tomáš', '123'
);
```

Smazání řádku z tabulky

```
DELETE FROM zakaznik WHERE id = 11;
```

Muzeme se jit na primarni klic
Where podmince muze byt logicke operatory a mnoho vice (plati i u SELECT)

Smazání tabulky

```
DROP TABLE zakaznik;
```

Vcetne dat uvnitr tabulky

Integritni omezeni - ON DELETE CASCADE , pokud smazu tohle tak se smazou vsecky zaznamy ktere s tim souvisi

2.2 Big Data

“buzzword”- popularni slovo ktere kazdy nevi co znamena
Data mimo možnosti běžného zpracování - to co se nevejde do bezneho pocitace
Různé definice
různé nároky: objem, rychlost, různorodost, ...
treba mam data ktere chci spracovavat rychleji nez je mozne
ruznorodost: co zaznam to jina struktura
Speciální prostředky: např. různé platformy nebo **distribuované databáze**

3 Reprezentace obrazu v počítači

Pixel

Bezrozměrná jednotka - hw hledisk ten pixel rozmer ma , ale lisi se napric vsemi moznimi zarizenimi
Rozlišení = počet pixelů
ppi = počet pixelů na palec, dpi = počet bodů na palec
hustota pixelu na jednotku
Kolik se do jednoho palce vejde pixelu (tim rika i velikost jednotliveho pixelu)
zpohledu monitor ma nizke rozliseni
pokud chi neno tisknout tak musim vzit vetsi rozliseni (kazda tiskarna zvladne minimalne 300ppi)

Spoustu grafickych programu umoznuje vybrat rozliceni pomoci dpi
chci tisknout obrazek ve velikosti 300dpi pak musim vybrat 300dpi

Cim jemnejsi je mrizka tim je mene viditelne jednotlive pixely

Referencni pixel, nezavysli pixel - skupina hw pixelu (je to to same jenom s jednim prisel GOOGLE a s jednim APPLE)

treba 1 referencni pixel = 4 hw pixely , hodne casto u mobilnich telefonu treba kdyz chci zobrazit 1px caru tak by nebyla videt

Retina dysplaye maji pomer 1 referencni : 4 hw

1920px

- Monitor 24 palců (šířka 20 palců), rozlišení 1920px

$$\frac{1920}{20} = 96 \text{ ppi}$$

- Tisk 8,3 palců (formát A4),

$$\frac{1920}{8.3} \approx 231 \text{ ppi}$$

- Tisk 11,7 palců (formát A3),

$$\frac{1920}{11.7} \approx 164 \text{ ppi}$$

Barevny model

Reprezentace barev v počítači
Barevné modely popisují barvu pomocí různých částí (složek).

Dulezite:

- Aditivní (sčítání složek, více → světlejší)
  RGB ...
- Subtraktivní (odčítání složek, více → tmavší)
  CMY(K) ...

RGB

Dysplay je defaultne cerny

cim vic tam davam barevnych slozek tim ziskavam barvu svetlejsi
Uplne cerne je tezke dosahnout (OLED dysplaye maji opravdu cernou)

Aditivní model
Displeje
Barva = červená (R), zelená (G) a modrá složka (B)
to rozdeleni vychazi z lomu svetla

Složka reprezentována (obvykle) jako 8-bitové číslo
Každá složka 0–1, (v případě 8-bit. **reprezentace 0–255**)
Jde tim popsat 256^3 barev (cca 16 milionu (stejne jako u IP adress s maskou 255.0.0.0))
RGBA = RGB + průhlednost
A - ALPHA chanel
Zápis: hexadecimální, `rgb()` (web)
treba rgb(100,100,0)

CMY(K)

Subtraktivní
Papir je totiz defaulntne bili
Tiskárny
Barva = tyrkysová (C), fialová (M), žlutá (Y), černá (K)
Složka v %
cim vic jednotlivich slozek nanesu na ten bod tim ta barva je tmavsi
K-key (Cerna) - z duvodu ekonomicnosti
Prakticky: problém s přechodem od RGB k CMYK
(je to heurestika) - lisi se to s tiskarny od tiskarny
tiskarna musi byt dobre zkalibrovana, a musi byt nastaveny dobry barevny profil
RGB to CMY je jednoznacne vyjadreni

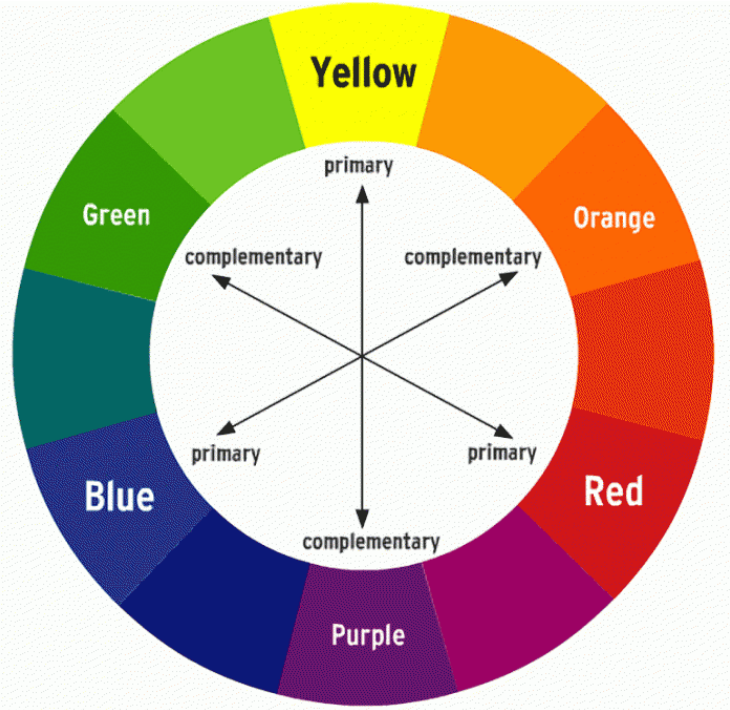

Figure 2: Enter Caption

HSV, HSL

zname s gympu
To co je popsane RGB modelem jde popsat timto modelem
blize vnimani barev pomoci lidskeho oka
Popularni mezi grafiky (ANOOO)
Taky je to fajn option v GD editoru

HSV barva = odstín (H), sytost (S), jas (V)

- nejvíce odpovídá lidskému oku

HSL barva = odstín (H), sytost (S), světlost (L)
Odstín = 0–360 (poloha na barevném kole), ostatní %
Populární v grafických nástrojích

Reprezentace obrazu v PC

Obraz → intuitivni chapani
obrazova fce z = f(x,y)
podobne LBM-mapam u krupky
obraz je funkce ktera je spojita
to znamena ze je tam nekonecno bodu
coz nelze uchovavat nekonecno boud

prevod obrazu do diskretni reprezentace (diskreditace)- omezeni na konecnou mnozinu

- vzorkovani = diskretizace x a y
  nevezmu vsechny body ktere tam jsou a vyberu si jenom vzorky (nejake konkretni body)
- kvantovani = diskretizace z
  rozdelim body do kvantilu a nasledne je priradim do urciteho kvantilu

rozliseni obrazu

pocet barev

Ulozeni obrazu

muzu to ulozit v raw ale to je moc velke
obrazek velikosti 100x100 se ulozi jako
256x100x100x3 Velikost
Hodi se kdyz si treba chceme ulozit fotku tak jak je nebo tak ale takto to nemuzeme dat na web

Typy obrázků

- Bitmapové
- Vektorové

Komprese

- Bezztrátová
- Ztrátová

Ulozeni obrazu: Priklady formatu

- JPEG (JPG)-histroricke duvody z Dos kde byly pro koncovku jenom 3 znaky
  ztrátová komprese, míru lze nastavit, ideální pro fotografie, nebezpečí kumulativní ztráty kvality, příliš vysoká komprese → čtverečkované oblasti, nepodporuje průhlednost
  Hodi se to pro fotografie, efektivni ulozeni, odstranime body ktere vnimame jako nepodstatny
  Ne pro fotky urcene na upravovani - pri kazdem znovu ulozeni je tam dalsi komprese
  Az jako produkcni obrazek

  progresivní JPEG - Postupne zaostrovani , prvni se na webu zobrazi rychle nejaka paforma a pak se to postupne zobrazi v lepsi kvalite (pro web)

- PNG
  určeno pro webovou grafiku (náhrada za GIF), různé typy barevných palet (8, 24 a další), bezztrátová komprese, progresivní PNG, vhodné pro obrázky obsahující velké plochy stejné barvy
  Pro souvysle plochy
  Neefektivni pro ukladani fotografii

- WebP
  určeno pro webovou grafiku, ztrátová i bezztrátová komprese, animace, progresivní WebP

- AVIF
  ztrátová komprese, slabší podpora, nepodporuje progresivní

- TIFF
  bezztrátová komprese, původně určeno pro tisk, nosny standart pro bezztratovou kompresi, mnoho fotoaparatu dava moznost to ulozit do tiff misto raw

Ukázka: `https://squoosh.app/`

Metadata
dotacne informace
datum porizeni , misto porizeni ...
Exif format
strukurovane data klic hondota

Vektorova grafika
Obraz složen ze základních objektů (bod, přímka, polygon, kružnice, křivka)

Matematicke objekty
Přesný popis
pojem rozliseni je u nich druhorady
Libovolné zvětšení a zmenšení
Obvykle úspornější než bitmapová grafika, ale náročnější práce (vytváření, editace)
Může obsahovat bitmapovou grafiku
Formáty: SVG, EPS - pro tisk, PDF

SVG

Značkovací jazyk
Původně určený pro webovou grafiku → lze vnořit do HTML, modifikovat pomocí CSS a JS
V ui nebo na loga treba
lze skrz svg delat i hry
Rozšířený a populární formát

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" width="300" height="300">
    <rect width="100%" height="100%" fill="#016BAB"/>
    <circle cx="150" cy="150" r="90" fill="white"/>
    <text x="93" y="160" font-size="28" fill="#016BAB">inf.upol.cz</text>
</svg>
```

Pismo

Součást OS, lze přidávat

Bitmapové písmo
zastarala vec (podleha to strate kvality)

Outline písmo

- založeno na křivkách
- Type1, TrueType
- třeba přizpůsobit pixelové mřížce (tak to nakonec zobrazi pc)
- velke ovlivnuti citelnosti pisma (muze se stat necitelne)
- komplikovanejsi pisma se nemusi zobrazovat na nejakych dysplajech

Rodiny písem

- patkové písma (serif), např. Times, Georgia
  takove zobacky (dobre pro citelnost textu) patka se oznacuej jako serif
- bezpatkové písma (sans-serif), např. Arial, Verdana
  displaye atd... , snadneji se to prizpusobuje displayi

- stejná velikost písmen (monospace), např. Courier
- kurzíva, např. Comic Sans
  skloneny font
- dekorativní (fantasy), např. Impact

Font znatelně ovlivňuje prožitek ze čtení
Typografie

4 Bezpecnost

Obecné požadavky (na systém)

- Data jsou přístupná pouze uživatelům, kteří na to mají nárok. (Opravneni)
- Data mohou měnit pouze uživatelé, kteří na to mají nárok. (Opravneni)
- Ověření identity uživatele.
  tehnhle ten clovek je opravdu tenhle ten clovek a muze tedal to a to

Necheme aby ty informace videli vsichni
je treba nezadouci aby si studenti mohly zadavat znamky
nebo aby mohly jejich informace cist vsichni (OUTRATA MOMENT XDDD borec se na to vyjebal actually mam pocit ze je to dokonce trestne)

System bez bezpecnosti = spatny sytem
Lidi jsou zmrdi tohle je ponauceni
Hlavne pri vytvareni software

Důvody porušení bezpečnosti

Bezpecnost z pohledu OS
Procesy a soubory mají vlastníka
Oprávnění v OS = **řízení přístupu k procesům a souborům**
Oprávnění (kdo, co, s čím) - treba uzivatel **jan picus** muze **smazat** soubor **windows33**
Různé implementace
Autentifikace uzivatele (login screen)

- Access Control List (ACL) = seznam uživatelů a jejich práv (Windows)
- Role-Based Access Control (RBAC)
  Matice uzivatelu x opravneni (v matici je 1 ma to opravneni v matici je 0 nema)
- Přístupová práva pro vlastníka, skupinu a ostatní (UNIX)
  Upraveny Acces Control List
  Jde to treba **delat u kazdeho souboru**

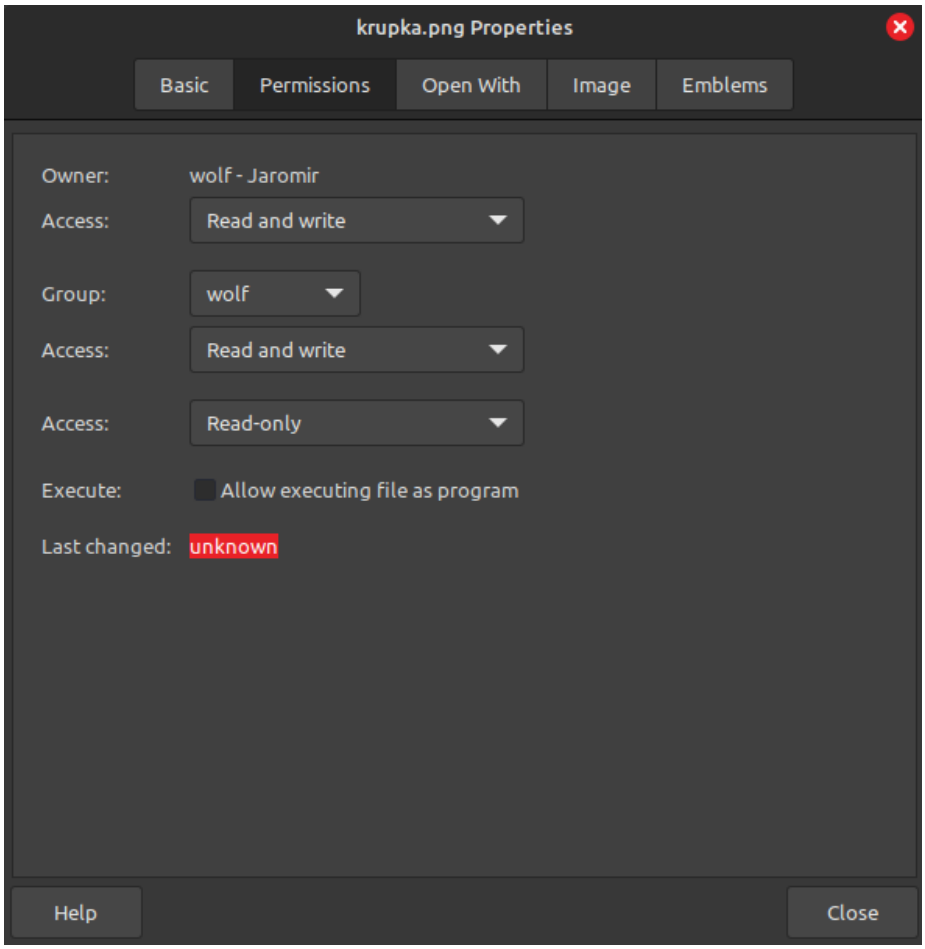

Figure 3: Enter Caption

Ukazka Pristupova prava pro vlastnika, skupinu a ostatni
Ano mam linux tak to sem muzu dat vetsina lidi to podle me nevidela(nevideli linux)

Utoky

Hrubá síla (brute force) - snazim neco zlomit vysokym vypocetnim vykonem , zkouset vsechny kombinace (heslo ma 8 znaku a zkousim vsechny hesla co maji 8 znaku)
Trojský kůň, backdoors - pripojeni ve FTP serveru, doslo k poruseni authentifikaze a identifikace, Treba si to muze vytvorit sam programator
Spoofing, phishing - udelam treba stranku upol portall a budu tak rybarit nepozorne lidi, podvrhovani cehokoliv
Denial of Service (DoS), Distributed Denial of Service (DDoS) - zabraneni pristupnosti sluzby, pomoci masivniho vykonu (co dopoctu), neustale kontaktuji treba webovy server a nestiha reagovat na pozadavky, ani ti chtene a treba to spadne pak
Je to tezke poznat obcas
Dos - jeden na jednoho
DDos - mnoho na jednoho
Malware (virus, rootkit - navazan na os , spyware - monitoruje data na pocitaci, adware - podstrcena reklama, ransomware - zasifruje data na pocitace a chce vypalne)
Postranní kanál - monitorovani dodatecnych informaci
Sociální inženýrství - vzdycky nelepsi pouzit, apeluje na hloupost bezneho uzivatele (Jsem vas spravce site) (Spravce site upoi)
Cross-site scripting (XSS), SQL injection - moznost vlozeni codu do jineho codu (treba do formulare)

Bezpecnost je jako cibule - ma vrstvy (ne ze smrdi) (trneckovy slova)

Kryptograficke hashovaci fce

Kryptografické hashovací funkce
Libovolný vstup, **výstup pevné délky**
Základní vlastnosti:

- Hash $f(x)$ je snadno spočitatelný pro libovolné x.
  rozumna casova doba
- Drobná změna x způsobí velkou změnu $f(x)$.
- Z hodnoty $f(x)$ je **obtížné** určit x.
- Je těžké nalézt dvě různé x_1 a x_2 takové, že $f(x_1) = f(x_2)$.
  kolize - jak v algu

Příklad: MD5, SHA-1 (obě nejsou bezpečné), **SHA-256 a vyssi**
KMI/UDITE
→ 2844012995373d81dbcd20d0bac97b34
LMI/UDITE
→ 10da7dc30e9c8b709545f813dcc55bfe

Ukazka toho ze to hazi pri male zmene uplne jiny vystup

Heslo

Silné heslo → Určené počtem pokusů potřebných pro uhádnutí hesla
`https://www.purecloudsolutions.co.uk/how-long-will-it-take-to-hack-your-password/`

Obtížně zapamatovatelné heslo = bezpečnostní riziko
lide si ho nekam zapisou atd, nebo pouzivaji vsude stejne
Idealni heslo je dlouhe heslo
Uniklé heslo (k účtu) lze ověřit ve veřejných databázích

Unikle hesla jsou davana do slovniku
`https://haveibeenpwned.com/`
Pozor na podvodné weby

Ukládání hesel
Nutne ulozeni pro authentifikaci
Nikdy v textové podobě!!!!
Obvykle tzv. sůl, tedy kryptografický hash $f(heslo + sul)$, sůl je pro každé heslo jiná
Sul je navic nejaka nezmana - treba prvni 3 pismena z emailu
Clovek se prihlasi a dojde jenom k porovnani hashu

Sifrovani

Zabránění čtení neoprávněnou osobou
Symetrické šifrování

- Symetrický klíč (a nejaka funkce)
- Problém výměny klíče
  Klic je jenom 1 a musim ho nejak vymenit , jsou jako cesty pomoci nejake vycenfazove authentifikace
- treba: Transpozicni sifry (Posun pismen kde klic je konkretni cislo o kolik to mame posunout) Moznost snadneho prolomeni (treba frekvencni alalyza) nebo brute-force

Asymetrické šifrování

- Asymetrický klíč

- Dva druhy klíčů: soukromý a veřejný, navzájem propojeny:

$$f_s(f_v(m)) = m = f_v(f_s(m))$$

- **Z veřejného nelze odvodit soukromý**
- Generování klíčů: např. RSA, eliptické křivky
- pro komunikace 2 je potreba 4 klicu
- propojeny matematikou kdyz neco zasifruji soukromim klicem a pak pouziju verejny klic zvnikne zprava a naopak (viz ty rovnice nahore)

Šifrování vs. utajení (např. steganografie)
sifrovani - i kdyz vidim komunikaci je mi to k hovnu
Utajeni - snaha tu informaci nevidem, jedine zabezpeceni je nevedomost
Treba steganografie (ulozeni do obrazku) nebo vyholis hlavu otrokovy a pak mu narostou vlasy
Komunikace by mela byt sifrovana a utajena zaroven

Integrita zpravy a elektronicky podpis

Integrita = zpráva nebyla změněna

- "Alice posle bobovy text miluji te a Trudy to zmeni na nemiluji te a mame hruzu"
- Kryptografické hashovací funkce
- Posílá se: zpráva, hash zprávy a ověřovací klíč (zabránění podvržení zprávy)
- bob by mohl rict ze k tehle te zprave neodpovida ten hash a zjitime ze to bylo pozmenene
- poslu hash zpravy a nejakeho utajeneho overovaciho klice
- nevyhoda vymena overovaneho klice (sym sifrovani)
- Běžně se používá pro **ověření integrity datových souborů**

Podpis = ověření autora

- Využití asymetrického šifrování
- Podpis zprávy $m = f_s(m)$
- Podpis je platný pouze pro zprávu $m \rightarrow$ Integrita dat

- Výpočetní náročnost hashování → Podpis hashe zprávy
- Zajisti i integritu zpravy

- mam zpravu tu zaheshuju a tu podepisu svojim neverejnym klicem
- Certifikace veřejných klíčů
  → Ověření vlastníka klíče certifikační autoritou (podepsání veřejného klíče)
  Podrebujeme pisemny doklad a vysledkem je certifikovany verejny klic, muj klic nekdo vezme a podepiseho svym soukromim klicem
- v protokolu HTTPS (stahnu si verejny klic a ten pouzivam pro vymenu informacich s tim web serverem)

4.1 BlockChain

Řetězec bloků (data, hash dat a hash předchozích dat) = datová struktura distribuované (P2P) a bezpečné uchovávání dat

- Představeno v roce 1991 jako **distribuovaná** účetní kniha (DLT), 2009 Satoshi Nakamoto → Bitcoin protokol → hype

- persistentni uchovani dat

- data jsou neprepsatelne a distribuovane
  Notarsky denik dat ktera jsou nemena
- vytvoreni noveho bloku je vypocetne narocne
- V kostce:
  - Problém v P2P: shoda na blockchainu (každý uzel může mít jiný)
  - **Nejdelší řetězec je považován za validní**
  - Teoretický útok: získat 51 % výkonu → Nejdelší řetězec je vytvářen 1 entitou
    - jeho retezec by byl pak vzdycky nejvetsi
    - a mohl by si stim delat co chtel
  - Řešení: vytvoření nového bloku je výpočetně náročné (různé metody, např. proof-of-work)
  - Každé přidání bloku = potvrzení předchozích
  - Změna v existujícím bloku zneplatní předchozí bloky
  - je velmy nevyhodne pro utocnika pozmenit ty bloky (v moment co zmenim jeden musim prepocitat vsechny pred tim (ty odakazy) atd)
- Použití: velmi populární, kryptoměny , evidence lekarkych zazanmu, notarske-spisi (puvodne to bylo proto vymysleno)

Blockchain – Bitcoin

Blockchain = účetní kniha (kdo, komu, kolik), všechny transakce
Vtom blokchainu jsou uplne vsechny transakce ktere sni vzdycky byly
Bitcoin penezenka - sifrovani s verejnym klicem
Transakce : jeden verejny klic, druhy verejny klic

- Transakce, digitální podpis pro verifikaci
- Blok = několik transakcí
  velikost bloku je paramter, bezne 1mb cim vetsi tim bezpecnejsi a pomalejsi
- Přidání bloku (těžba Bitcoinů) - bitcoinmining
  Vezmu transakce a vytvorim znich blok a provedu nejake overeni ktere je popsano v Protocolu Kryptomeny
  - Umístění transakcí do bloku, přidání odkazu na předchozí blok, ověření pravidel protokolu
  - Verifikace bloku (uhádnutí vstupní hodnoty SHA-256, každé 2 týdny se mění obtížnost, čas vždy cca 10 minut), proof-of-work - musim neco matematicky spocitat
    jedina cesta je nahdona cesta
  - Nejrychlejší vyhrává (získá odměnu)
  - Odměna = poplatky + nové bitcoiny (cca každé 4 roky se změní na polovinu, 2140 dojdou)
    pravo byt pridan do blockchainu
- Vylepšení: smart contracts
  transakce pokud se provedou, za predpokladu nejakejch podminek
- Kritika:
  - mining-pools → centralizace, spotřeba elektrické energie, škálovatelnost
  - tezba kryptomen zpotreboval
  - centralizace - moznost provest 51 procentniho utoku

vyhoda kryptomen: stat do toho nevidi, je to soukroma transakce mezi 2 osobama

4.2 Verzovaci Systemy

Tohle nebylo na te prednasce!!!
Verzovaci software
Vyhody: spoluprace na vyvoji, evidnce zmen, ...
lokalni, centralizovane a decentralizovane(beznejsi)
napriklad

- Git
- Mercurial
- SVN
- Apache Subversion

Git
verzovaci system
v roce 2005 vytvoril Linus Torvald pro verzovani Linuxoveho jadra
Git (verzovaci system) $\neq$ GitHub (repozitar)
pomerne komplikovany sytem
uklada stav vsech souboru (na rozdfil od jinych, ktery ukladaji seznam zmen)
data jsou ulozena lokalne (repozitar) a nachazi se ve trech stavech

- modified (zmena neulozena data)
- staged (data pripravena k zapsani)
- commited (zapsana data)

zapsani zmen (vytvoreni verze) = commit, obvykle obsahuji popis
aktualizace lokalnich dat ze vzdalenych (pull)
aktualizace vzdalenych dat z lokalnich (push)

Vetveni
Odklon od hlavni vetve
Git realizuje jako ukazatel
aktualni vetev (HEAD ukazatel)
slouceni vetvi (merge), vytvoreni nove verze spojujici vetve
preskladani (rebase), slouceni vetvi do jedne (ztrata vetve)