# Cvičení 1

## Data, jejich zápis transformace a interpretace

Na jednoduchém příkladu dat, která každodenně používáme, si ukážeme základní principy příjmu dat a jejich transformace a Interpretace. Proč vlastně data transformujeme? Je to za účelem snadnějšího zapamatování informace a jednoduššího způsobu přenosu.

## Cíle cvičení

* Pochopit účely transformace zápisu a interpretace dat
* Vědět, co je to číslo
* Jak zapsat číslo
* Jak čísla interpretovat
* Jak fungují základní operace s čísly
* Jak se pracuje s časem

## Vysvětlení



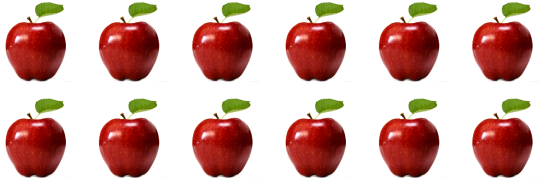
Data popisují **jevy**, něco, co se někde v nějaký okamžik stalo, nebo jaký byl aktuální stav něčeho (počasí, stav peněz na účtu, …). Pokud si chceme o tom, co jsme někde viděli a zažili s někým popovídat, musíme to umět pojmenovat. Každý máme nějakou slovní zásobu. Umíme pojmenovávat věci, se kterými se pravidelně setkáváme, a naučili jsme se je i počítat.

Jsme tedy určitý jev popsat větou v nějakém jazyce. V našem případě: „**Když jsem jel kolem, tak jsem na louce viděl jedenáct ovcí“**. Nad vznikem slovní zásoby přemýšlet asi nebudeme, to by bylo složité. Nicméně nad tím, co je to číslo, a jak čísla vznikají se trochu zastavíme.

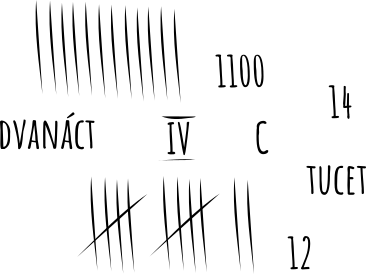
**Co je to číslo?**

Budeme-li se bavit o přirozených číslech, tak jejich **hodnota** vyjadřuje množství nebo pořadí něčeho. Pokud chceme **informaci** o počtu nebo pořadí předat, musíme hodnoty buď nějak pojmenovat (slovem) nebo nějak zapsat. Zápis čísla, tak jak ho známe a slovo pojmenovávající určitou hodnotu, jsou dvě formy, které uchovávají stejnou informaci. Pro nás je to i příklad transformace dat, kterou provádíme v podstatě neustále.

**Příklad**



**Co vidíte na předchozím obrázku? Která z následujících možností zápisu je správná, nebo špatná?**

****

V tomto případě jsou všechny zápisy správné. Jak je to možné? Je to způsobeno tím, že každý z uvedených zápisu je zapsán podle jiných pravidel. S největší pravděpodobností jste se už v životě setkali s pojem **číselná soustava**. Číselné soustavy určují, jak hodnoty zapisovat a jak čísla číst.

**Číselné soustavy**

Číselné soustavy představuji soubor pravidel, jak zapsat a přečíst číselnou hodnotu. Definují konečnou množinu znaků (číslic). Z číslic pak pomocí jejich kombinací vytváříme čísla, které reprezentují určitou hodnotu.

Číselné soustavy můžeme rozdělit do dvou základních skupin:

* Poziční číselné soustavy – hodnota číslice je závislá na její pozici
* Nepoziční číselné soustavy – hodnota číslice je nezávislá na její pozici.

Příkladem nepoziční soustavy jsou třeba římské číslice, kdy třeba u čísla XX (dvacet) každý znak představuje stejnou hodnotu. Pokud požijeme desítkovou číselnou soustavu, která se řadí mezi poziční číselné soustavy, tak číslo 11 (jedenáct) se skládá ze dvou stejných číslic. V tomto případě ale má každá jednička jiný význam. Jedno to je jedna a jednou deset.

Poziční číselné soustavy jsou definovány svým základem. Základ je číslo představující počet číslic, které pro zápis čísel používáme. Hraje také důležitou roli v tom, jakou hodnotu číslice představuje. Jednotlivé číslice nám říkají, kolikrát se v hodnotě vyskytuje mocnina základu. To nám ukazuje následující obrázek.

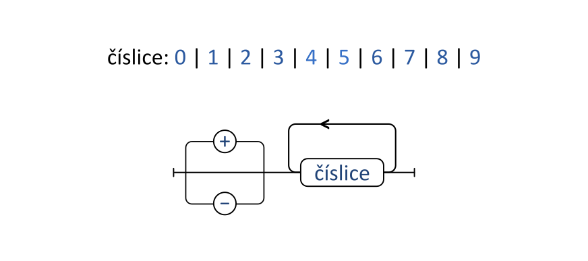


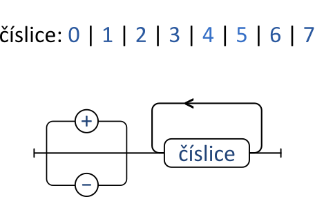
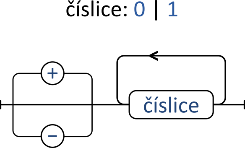
V tento okamžik bude možné dobré se pozastavit na dvěma pojmy **Syntaxí** a **sémantikou**. Syntaxe představuje pravidla, jak něco zapsat. Sémantika potom je význam daného zápisu. S těmito pojmy se budeme potkávat po celý semestr.

Pokud pracujeme s desítkovou soustavou, pravděpodobně nám vše připadá přirozené a automatické, v podstatě při jejím užití skoro nepřemýšlíme. Jiné to ale bude, když použijeme soustavu jinou. Ačkoli vše funguje naprosto stejně (rozdílem je pouze počet používaných číslic), tak na jednou se automatika vytrácí, a musíme začít přemýšlet (😊 to je také jedním z cílů tohoto předmětu)



Stejně zapsaná čísla, avšak v jiné číselné soustavě, má jinou hodnotu. Tím, jak zapsat a interpretovat hodnoty v jiných soustavách se budeme zabývat v příkladech, budeme tedy hledat jejich sémantiku.

**Syntaxe** zápisu čísel je potom jednoduchá. Máme přípustné znaky a ty píšeme za sebe.



Jediné, čím se syntaxe zápisu čísel v různých soustavách liší, je množina znaků, které můžeme použít. Desítková soustava používá 10 číslic, dvojková 2, osmičková 8, šestnáctková 16. Deset číslic používáme běžně, to znamená, že u čísel zapsaných v soustavách se základem menším nebo rovným desíti víme, co můžeme jako číslice použít. V případe, že potřebujeme číslicí zapsat hodnotu větší než 9, tak musíme použít jeden znak (v případě hodnoty dvanáct nemůžeme nepsat 12, protože to jsou dvě číslice). Proto v těch to případech používáme velká písmena anglické abecedy. Výhodou jejich použití oproti jiným znakům je, že známe jejich pořadí.

**ČAS**

Do problematiky číselných soustav zahrnujeme i čas. Čas hraje v oblasti ekonomie i ekonomiky velice podstatnou roli. U času nás zajímají dvě hlediska:

* Označení okamžiku (kdy se něco stalo, kdy něco začalo, kdy něco skončilo, …)
* Délka uplynulé doby (rozdíl dvou okamžiků).

Z ekonomického pohledu nás samozřejmě oba dva tyto aspekty. Například za uplynulý čas se platí (odpracovaná doba za jeden den, počet dnů na pracovišti, délka pracovní cesty,…), a okamžiky (konkrétní hodina dne, nebo den týdne) nám mohou třeba určovat cenu, nebo dostupnost nějakého ekonomického statku.

U času spíše než základní základní fyzikální jednotku sekundu, používáme ty vyšší (minuty, hodiny, dny, týdny, měsíce, roky). Asi každý víme, že minuta má 60 sekund, hodina má 60 minut, den má 24 hodin (zpravidla – dvakrát do roka je to zatím jinak) týden má 7 dnů. S měsícem je to potom už trochu složitější :) a rok má 12 měsíců nebo 365 resp. 366 dnů.

S časem jako číselnou soustavou velice často počítáme, a nevidíme v tom žádnou problematickou záležitost (aspoň tedy doufám)

*Čas hraje důležitou roli v oblasti úročení peněz, počítání odpisů. Všechny tyto procesy jsou řízeny přesně definovanými algoritmy. K těm se ale dostaneme až v pozdějších fázích tohoto kurzu.*

## Úkoly a otázky

1. Jakou číselnou soustavu vlastně používá čas? Jedná se o poziční?
2. Jaké bude datum za 100 dní a 1000 minut od 1.1. 2024
3. Jakou hodnotu představuje hodnota (7423)8 zapsaná v osmičkové soustavě?
4. Jak by vypadala hodnota (1250)10 převedená do dvojkové soustavy a do šestnáctkové soustavy[[1]](#footnote-1)
5. Proč v šestnáctkové soustavě nemůžeme pro hodnotu deset použít zápis 10?
6. Jakou hodnotu bude mít číslo (BABA)16 při zápisu v desítkové soustavě?
7. Bude mít hodnota (10000)10 větší hodnotu než hodnota (10000)16? Zkuste vysvětlit proč.
8. Kolik vám bude přesně let a dní v okamžiku vašeho desetitisícího dne?
9. Kdyby vám rodiče, každý den, co jste na světě odložili 150 korun, kolik byste měli dnes našetřeno?
10. Kolik let, týdnů, dnů, hodin, a minut uběhne za 1 000 000 sekund.

*Existuje velké množství nástrojů, které by vám s těmito úlohami dokázali pomoci (Excel, kalkulačka, umělá inteligence). Tyto nástroje zkuste použít pouze pro kontrolu výsledků a výpočty se pokuste vyřešit pouze s tužkou a papírem. V tomto kurzu nám jde hlavně o pochopení toho, jak se s daty pracuje a jaké principy se dodržují. Správný výsledek je důležitý, ale nám jde hlavně o proces a jeho popis a pochopení.*

# Cvičení 2

## Podniková data, ENTITY, INSTANCE, ATRIBUTY, modelování

## Cíle cvičení

* Podpořit analytické myšlení
* Naučit se vidět data v souvislostech
* Pochopit principy vazeb mezi daty
* Naučit se abstrahovat obecné modely z konkrétních situací
* Naučit se přemýšlet nad potřebou a strukturou ukládaných dat
* Naučit se graficky znázornit jednoduchý model podnikových dat (vybraného úseku podnikových dat)
* Přemýšlet nad daty v kontextu potřeby uchovávat historická data za účelem analýz a následného porovnání.
* Vytvořit si znalostní základ pro práci s relačními databázemi.

## Úvod

*Na minulém cvičení jsme se zabývali daty, která představovala jednu konkrétní hodnotu (čas nebo číselná hodnota). Toto cvičení nebude tak ani o datech jako hodnotách, ale spíše o tom, v jakém kontextu data používáme. Zaměříme se na porozumění různých reálných situací, které nesou potřebu vzniku datového záznamu, třeba za účelem výpočtu celkových příjmů, výdajů, mezd, odpracované doby, atp.*

*Budeme se snažit analyzovat naše „datové a informační“ potřeby.*

## Vysvětlení

Představme si situaci, že jsme pracovník personálního oddělení nějaké větší firmy a potřebujeme pracovat s informacemi o tom, jaké pracovníky zaměstnáváme (nebo jsme historicky zaměstnávali), na jakých pracovištích tito pracovníci jsou, nebo byli zařazeni, s jak velkým úvazkem na konkrétním pracovišti v daném časovém období pracovali, a třeba který z pracovníků v určitém časovém úseku za přidělené pracoviště zodpovídal jako vedoucí.

Předchozí odstavec je popis nějaké situaci (reality), ze které vychází určité požadavky na ukládání dat a informací. Dříve než začneme nějaké záznamy někam zapisovat, musíme si vše dopředu dobře rozmyslet a systém samotného ukládání dat nastavit tak, aby byl schopen splňovat zadané požadavky, ale taky aby v budoucnu byl i rozšiřitelný.

Pokud chceme nějakou sledovanou realitu dobře a přehledně popsat, poslouží nám k tomu nástroj obecně označovaný jako **Model**. S tímto pojmem se setkáváme v různých oblastech našeho života.

Mají slova jako model a modelka používané v oblasti odívání stejný význam, jako ten, který budeme používat my, a třeba jej používají modeláři při stavbě modelových železnic, nebo malých letadélek? To si vysvětlíme v následující podsekci.

### Datový model

Co vůbec znamená slovo **model**? Jako model můžeme chápat zmenšeninu reálné věci. Modely obecně slouží za účelem zjednodušení a popisu reality. Ta může být hmotná (můžeme si představit třeba model letadla) a nebo logická, ta bude zajímat nás.

Jaké mohou být účely modelů:

* Porozumění
* analýza reality
* simulace
* zjednodušení
* abstrakce
* vizualizace

Tak že to znamená, že taková modelka v nových šatech je vlastně nástrojem takové simulace a viuzalizace, jak konkrétní oblečení vypadá oblečené na někom, komu se podobáme, nebo bychom se chtěli podobat, a nám si představit představit, jak bychom třeba v tom samém oblečení mohli vypadat my.

Z tohoto příkladu vede jedno velmi důležité ponaučení: **aby věci v realitě pracovali správně musíme vždy vybrat model, který co nejvíce odpovídá naší reálné situaci a aktuálním potřebám**. Toto je nutné zobecnit na jakoukoli situaci, kdy se chystáme něco koupit, něco zprovoznit, nebo obecně změnit.

**Modely nám tedy slouží pro lepší vytvoření představy systému**. Na modelu si můžeme vyzkoušet, jak věci budou v reálu vypadat a fungovat, můžeme na modelu odchytit možné následné chyby. Modely mohou být jak hmotné (třeba právě model letadla), tak myšlenkové – logické, mohou existovat jen v naší mysli, anebo tužkou namalované na papíru, či za pomocí nějakého softwarového nástroje.

Pokud budeme popisovat nějaký systém, bylo by dobré si vysvětlit, co tento pojem znamená.

**Systém je množina prvků a vazeb mezi nimi**. Má nějaké vnitřní prostředí, vnější prostředí a má také nějaké chování.

Za systém můžeme považovat třeba naše tělo, ale tak složitými systémy se zabývat nebudeme. Zatím se nebudeme zabývat ani chováním. Zajímat nás bude jen datová a informační stránka věci.

### Jak vytvořit datový model?

To, co vidíme, co popisujeme, je většinou něco **konkrétního**. Sledujeme **věci**, nebo **události**.

Stojíme-li na křižovatce a kolem nás projede nějaké vozidlo, tak se jedná o jeden konkrétní vůz s jedinečnou SPZ. Za chvíli za projede jiný vůz, bude asi jiného typu jiné značky, jiné barvy, budou v něm sedět jiní lidé, nebo poveze jiný náklad.

Jedno konkrétní auto, je takzvanou **INSTANCÍ**, pokud však budeme chtít vytvářet model této situace, tak si musíme uvědomit, že dopředu nevíme, jaká auta kolem nás budou jezdit. Musíme tedy naše myšlenky vést v nějaké obecnější rovině. Sledovaným objektem bude tedy AUTO. Auto bude takzvaná **ENTITA**.

***Entita představuje obecné pojmenování jevů (věcí, událostí), které jsou stejného druhu.***  Každá entita je definována svým pojmenováním a výčtem **ATRIBUTŮ** (sledovaných vlastností). Sada hodnot atributů (data) se vztahují k jedné konkrétní **instanci**.

**ATRIBUTY** jsou námi zvolené vlastnosti entity, a představují typy dat, která k jednotlivým instancím ukládáme.

**INSTANCE** je konkrétní zástupce entity, sada konkrétních hodnot atributů, které se k ní (ke konkrétní instanci) vztahují

Teď na chvíli přerušme sledování aut na křižovatce (za chvíli v tom budeme pokračovat) a vraťme se k příkladu s personálním oddělením. Jak to bude s entitami a atributy v tomto případě?

Potřebujeme evidovat zaměstnance. Obecný **Zaměstnanec** pro nás tedy bude entitou, a asi budeme schopni vyjmenovat i nějaké atributy, které nás budou zajímat:

* jméno
* příjmení
* pohlaví
* datum narození
* telefon

A close-up of a card

Description automatically generated

*Předchozí obrázek ukazuje, jak se entita zapisuje v Entitně relačních diagramech.*

Instancemi entity Zaměstnanec jsou jednotliví zaměstnanci. Například:

* Jan Novák, muž, narozen 15. 5. 1990, 609123465
* Petr Svoboda, muž, narozen 23. 9. 1985, 903453653
* Marie Nováková, žena, narozena 10. 12. 1992, 103567778

Entitu zaměstnanec si tedy můžeme představit jako šablonu (formulář) pro údaje, které ke konkrétní osobě si potřebujeme pamatovat. V některých situacích může nastat, že některé atributy nemusí pro konkrétní Instanci entity existovat. Třeba může existovat zaměstnanec, který nemá telefon, a tak nemůžeme evidovat jeho telefonní číslo. Proto u některých atributů evidujeme skutečnost, že jsou **povinné** a **nepovinné**.

Další entitou našeho systému personálního oddělení bude Pracoviště. Kdybychom uvažovali nad tím, jaké atributy by Pracoviště mělo mít, napadne nás třeba název a vedoucí.

A black and white text

Description automatically generated with medium confidence

Instancemi entity Pracoviště by potom mohli být záznamy:

* Oddělení marketingu, Jan Novák
* Oddělení prodeje, Marie Nováková

Co nám ale informace o jménu a příjmení vedoucího přináší? Sice víme, jak se jmenuje vedoucí pracovník, ale další informace o něm nemáme. Vedoucí pracovník je zaměstnancem a měli bychom ho evidovat mezi zaměstnanci. Abychom byli schopni dohledat informace o vedoucím pracovníkovi, musíme entity Pracoviště a Zaměstnanec nějak propojit. Musíme vytvořit nějakou **vazbu**, resp. vazby.

A diagram of a computer

Description automatically generated

Předchozí schéma popisující koncept propojení entit je možné přečíst:

* Pracoviště musí být řízeno jedním zaměstnancem
* Zaměstnanec může řídit pracoviště.

Toto je příklad vazby s kardinalitou (násobnosti) **jedna ku jedné 1:1**, resp. 0:1 (Zaměstnanec nemusí mít řídí, pracoviště, ale pracoviště musí být řízeno právě jedním zaměstnancem)

Pojďme se na problematiku vazeb mezi entitami podívat trochu podrobněji.

Mezi entitami jsou **vazby** (**relace**). U nich se můžeme bavit o kardinalitě a povinnosti. Kardinalita znamená násobnost vztahu instance jedné entity k instancím entity druhé. Rozlišujeme 3 základní:

A black lines with a white background

Description automatically generated with medium confidence

* 1:1 (Instance první entity je ve vztahu pouze s jednou instancí druhé entity, instance druhé entity je ve vztahu pouze s jednou instancí první entity)

*Příkladem této vazby můžou třeba být: Osoba – Občanský průkaz, Auto – SPZ. Vazby 1:1 jsou užitečné pro modelování situací, kde je potřeba zajistit, aby každá entita byla unikátně spojena s jinou entitou. Tato unikátnost se však většino vztahuje k jednomu okamžiku. (Pokud dojde k prodeji Auta, nový majitel může požádat o novou SPZ, tz. Že z historického hlediska mělo auto více SPZ).*

*V našem případě to znamená, že v jeden okamžik může řídit pracoviště pouze jeden zaměstnanec a pracovníkovi je dovoleno řídit pouze jedno pracoviště.*

A diagram of a computer

Description automatically generated

* 1:N (Instance první entity je ve vztahu s jednou nebo více instancemi entity druhé, instance druhé entity je ve vztahu pouze s jednou instancí první entity)

*Příkladem tohoto typu vazby může být: Zákazník – Objednávka (Jeden zákazník může vytvořit více objednávek, ale jedna objednávka patří vždy k jednomu zákazníkovi), Město – Dům (V jednom městě se může nacházet více domů, ale jeden dům je vždy právě v jednom městě)*

*Kdyby jednomu pracovníkovy bylo povoleno řídit více pracovišť, tak bychom vztah mezi entitami zaměstnanec a pracoviště zaznačili následovně:*

A diagram of a computer

Description automatically generated

* M:N (Instance první entity může být ve vztahu s více instancemi entity druhé, instance entity druhé může být ve vztahu s více instancemi instance prvé)

*Příkladem v tomto případě mohou být vztahy entit: Učitel – Student (učitel učí více studentů, studen mívá více učitelů), Zaměstnanec – Projekt (Zaměstnanec se může podílet na více projektech, Jeden projekt může řešit více zaměstnanců).*

*V našem přím přikladu by tento typ vazby znamenal, že pracovník může pracovat na více pracovištích a na jednom pracovišti samozřejmě více pracovníků.*

*A black arrow pointing to a line

Description automatically generated*

U tohoto typu vazby, při vytváření modelu se setkáme s tím, že ji budeme převádět na dvě vazby 1:N.

Když už tedy známe všechny druhy vazeb, můžeme se na náš personální systém podívat již celkově. N následujícím modelu (konecptuálním modelu) vidíme kromě entit zaměstnanec a pracoviště ještě dvě nové: **Vedoucí pracovník** a **Úvazek**.

A diagram of a company

Description automatically generated

Tyto entity nám do našeho modelu vnáší časové hledisko. Díky atributům začátek a konec ukládáme informaci o vazbě mezi Zaměstnancem a Pracovištěm. Budeme tedy schopni dohledávat informace o tom, kdy, kdo, jaké pracoviště řídil. Nebude se tedy jedna pouze o aktuální vazbu, kde bychom v případě změny vedoucího pracovníka pouze přepsali (zapomenuli původní vazbu) atribut vedoucí.

Abychom nezůstali jen u jednoho příkladu dokončeme i ten s auty na křižovatce.

Při sledování aut na „naší“ křižovatce bychom se měli zamyslet, co je účelem našeho sledování. Jestli jsou to samotná auta, nebo něco jiného. V našem případě sledovanou entitou není AUTO, ale událost PRŮJEZD\_VOZIDLA. Jaký je v tom rozdíl? Je dost značný. Auto je hmotná věc, Průjezd\_vozidla je cosi nehmotného, je to popis okamžiku. Proč to rozlišovat? Jedno auto (to stejné, se svojí sadou atributů) může jednou křižovatkou projet několikrát. Zatímco okamžik nastane vždy pouze jednou. Kdybychom chtěli tuto situaci namodelovat detailně, tak bychom mohli přidat ještě entitu TYP\_VOZIDLA (instancemi této entity potom budou: osobní vůz, nákladní vůz, motorka, autobus, dodávka,…).

A diagram of a computer

Description automatically generated

Konceptuální diagramy, které jsme doposud používali, znázorňují myšlenkový koncept. Určitou představu toho, jak by věci měli fungovat. Neřeší tedy zatím fyzické zpracování, jsou na něm nezávislé. Tyto diagramy pomáhají uživatelům a vývojářům lépe porozumět struktuře a logice systému, který modelují. Jejich základní stavební prvky jsou Entity, Atributy a vazby.

### Jak poznat rozdíl mezi Entitou a atributem?

Proč třeba u předchozího příkladu není typ vozidla pouze atributem entity Auto? Pokud si uvědomíme, že některé hodnoty atributů se nám **budou často opakovat**, tak se hned nabízí otázka, jestli by se nemělo jednat o samostatnou entitu. Důvodů může být několik. Jak se budeme bavit o fyzické realizaci databáze, tak z jedním hlavních důvodů bude třeba úspora místa v paměti. A to z toho důvodu, že si u každého auta nebudeme muset pamatovat (všechny informace týkající se typu vozidla). Dalším důvodem je právě potřeba ukládání si vice informací k dané hodnotě. Tz. že kdyby se v případě aut jednalo o služební vozy, tak nás bude zajímat třeba typ řidičského průkazu, abychom třeba mohli zjistit, jestli jeden konkrétní zaměstnanec může řídit jeden konkrétní vůz

## A diagram of a company Description automatically generated

Zaměstnanec může mít více řidičských oprávnění, jedno řidičské oprávnění může mít více zaměstnanců. Konkrétní řidičské oprávnění může příslušet více typům vozidel a určitý typ vozidla můžete uplatnit různá oprávnění (S oprávněním B můžete řídit osobní vozidlo do 8 pasažér a váhy 3,5 tuny, „malou“ motorku, nebo třeba dodávku do 3,5 tuny. Malou motorku pak můžete řídit i s oprávněním AM)

## Úkoly cvičení 2

1. Určete kardinalitu vazeb následujících entit a vaši odpověď zdůvodněte:
   * Auto × SPZ
   * Student × Studijní\_program
   * Manžel × Manželka
   * Matka × Dítě
   * Produkt × Faktura
   * Faktura × Zákazník
   * Osoba × Emailový\_účet
2. Na základě následujícího příběhu navrhněte konceptuální model:

Pokuste se sami zamyslet nad tím, jak byste evidovali toky financí ve vaší rodině. Každý člen domácnosti může mít několik účtů. Mezi účty mohou probíhat transakce. Příkladem může být, že tatínek pošle dětem kapesné. Transakce však nemusí probíhat jen mezi účty v našem systému, ale můžeme obdržet peníze z venku (například výplata). Samozřejmě můžeme naše peníze i utrácet. Důležité je u každé transakce sledovat její účel, abychom byli schopni sledovat různé statistiky. Například, kolik se utratilo, za dopravu, kolik se utratilo za jídlo, zábavu, atd. Každý účet má svého vlastníka, vlastník jej však může svěřit k užívání někomu jinému. Může také stanovit dohlížitele nad účtem, který neprovádí operace, ale může si je zobrazit.

Určete entity, atributy a vazby.

# Cvičení 3

## Cíl cvičení

Cílem tohoto cvičení je:

* Převést logický model na fyzický
* vytvořit databázi
* správně napsat požadavek
* Vložit data do tabulek
* Vypsat obsah tabulek
* Upravit data v tabulce
* Vymazat data z tabulky

### Fyzický datový model

Fyzický datový model je oproti konceptuálnímu konkrétní a detailní co se týče reprezentace dat. Již se zaměřuje na způsob, jakým jsou data uložena v databázovém systému. Fyzický datový model se soustředí na technické aspekty, jako jsou:

* **Struktura tabulek**: Definice tabulek, jejich sloupců a datových typů.
* **Realizace vazeb – Primární a cizí klíče**: Určení, které sloupce slouží pro identifikaci záznamů a cizí klíče pro definování vztahů mezi tabulkami.
* **Integritní omezení**: Definice pravidel, která zajišťují konzistenci dat, jako jsou unikátní omezení nebo pravidla pro referenční integritu.
* Indexy: Specifikace indexů pro optimalizaci výkonu dotazů.

V konceptuálním modelu jsme používali Entity, Atributy, a Vazby. U fyzického to budou Tabulky, sloupce a klíče.

|  |  |
| --- | --- |
| Entita | Tabulka |
| Atribut | Sloupec |
| Vazby | Klíče |
| Instance | řádek |

*Je nutné upozornit na to, že my se zde budeme zabývat relačním datovým modelem. Který právě pro ukládání relací využívá tabulky a principy, které si zde budeme vysvětlovat. Nejedná se o jediný typ databázových modelů, jsou i další (objektový, hierarchický, NoSQL,…) těmi se však v tomto kurzu zbývat nebudeme.*

### Entita vs. Tabulka

A close-up of a computer screen

Description automatically generated

Entita je obecné označení třídy objektů (jevů) stejného typu definující seznam zkoumaných atributů. Tabulka je záznamová konstrukce dat, rozdělených do řádků a sloupců, kde řádky jednotlivé řádky představují konkrétní instance Entity a sloupce (jejich přesně definované pořadí) představují hodnoty atributů.

Samotná fyzická realizace tabulky může být následovná:

+--------------+-------+---------+---------+--------------+----------+

|id\_zamestnance|  jmeno| prijmeni|  pohlavi|datum\_narozeni|   telefon|

+--------------+-------+---------+---------+--------------+----------+

|             1|    Jan|    Novák|      muž|    1990-05-15| 609123465|

|             2|   Petr|  Svoboda|      muž|    1985-09-23| 903453653|

|             3|  Marie| Nováková|     žena|    1978-12-10| 103567778|

|             4|  Ferda| Mravenec|      muž|    1988-10-10| 103567888|

|             5| Monika|     Nová|      muž|    1988-10-10| 103567888|

|           ...|    ...|      ...|      ...|           ...|       ...|

+--------------+-------+---------+---------+--------------+----------+

### Datový typ

Datové typy určují, jaké hodnoty mohou jednotlivé atributy nabývat, a jaké operace je možné s nimi provádět.

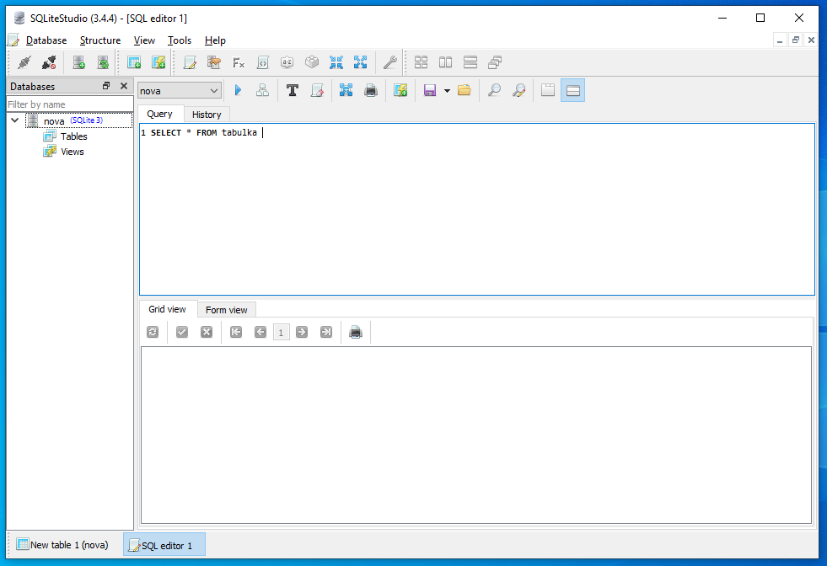
*Datové typy, které můžeme používat jsou závislé na zvoleném databázovém systémů. My budeme používat lehkou databázi typu SQLite, ta se snaží práci s ní držet na jednoduché úrovni, tak i nabídka datových typů není tak podrobná. My si podstatě vystačíme jenom s texty a čísly. Komplexnější databázové systémy nabízejí specifičtější datové typy a také možnost tvorby vlastních datových typů (třeba výčtů hodnot).*

Datové typy, které budeme používat:

* Integer – celé číslo, používá se však i pro uložení logických hodnot ve formě 0,1
* Real – reálná čísla
* Text – ukládání textových řetězců, může ukládat i data specifických formátů, třeba čas ‘rrrr-mm-dd hh:hh:ss.sss’
* Blob – ukládání binárních dat (obraz, zvuk) (s tím ale moc pracovat nebudeme)

### Vytvoření databáze v SQLiteStudiu

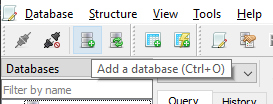
SQLiteStudio je program, který nabízí grafické rozhraní pro práci SQLite databázemi. Samozřejmě se nejedná o jediný nástroj tohoto typu. Je možné třeba používat DB Browser, DBeaver. Ty by mělo být dostupné i pro jiné platformy (Linux, MacOs)



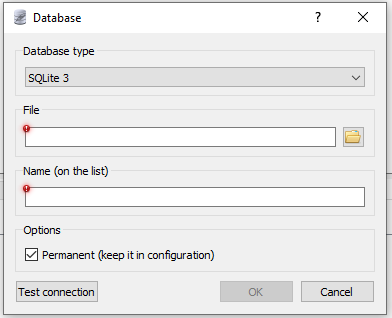
Jak pracovat s tímto program najdeme třeba na následujícím videu:

<https://www.youtube.com/watch?v=K7T0zUlWBY8>

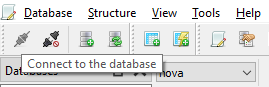
Databázi vytvoříme klinutím na tlačítko Add Database:



Následně pak nastavíme název souboru, ve kterém bude databáze uložena. Soubor by měl mít koncovku .db , .sqlite, nebo .sqlite3. Dále je pak možné nastavit název (identifikátor) databáze, pod kterým je v aplikaci zobrazena. Pomůže nám také, když potřebujeme pracovat s daty více databází, k identifikaci tabulek.

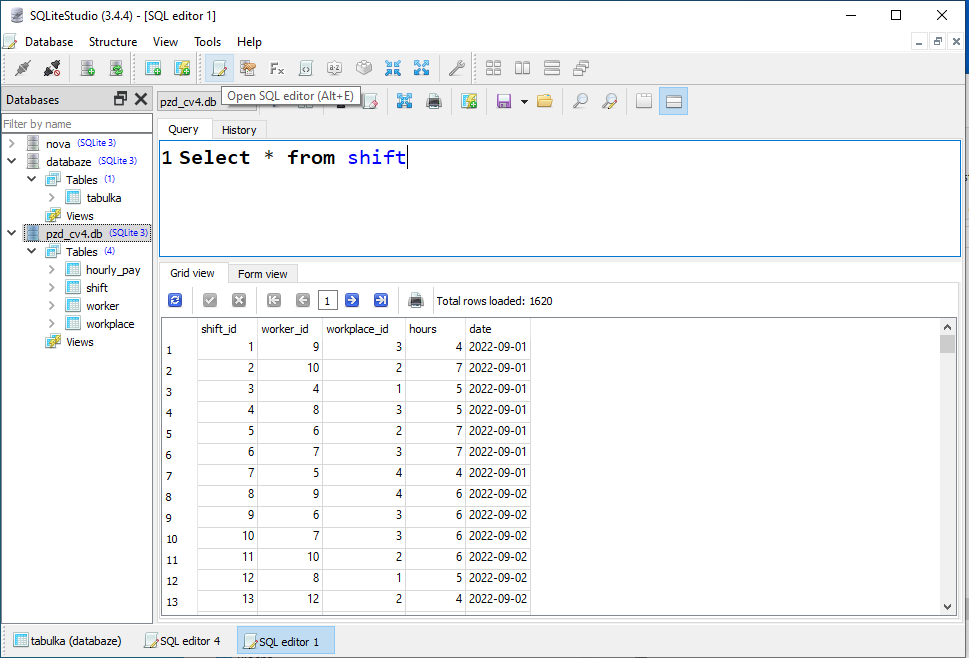


Pokud již databázi máme vytvořenou, tak stačí se k ní pouze připojit.



Toho docílíme kliknutím na první ikonku, nebo v nabídce databáze.

Psát SQL dotazy (příkazy) můžeme v panelu, který si zobrazíme kliknutím na ikonku papíru a tužky.



### Jazky SQL

Structured Query language je standardizovaný jazyk používaný pro správu a manipulaci s daty v relačních databázích. Umožňuje uživatelům provádět různé operace, jako je vytváření, čtení, aktualizace a mazání dat (známé jako CRUD operace) prostřednictvím příkazů, které jsou syntakticky podobné zjednodušené angličtině.

Existuje velké množství tutoriálů pro jazyk SQL. V některých případech můžete najít drobné odchylky, to může být způsobeno aplikací jazyka pro konkrétní typ databázového systému. Nicméně pro základy, které my v tomto předmětu budeme řešit jsou rozdíly nepodstatné.

* <https://www.w3schools.com/sql/>
* <https://www.sqlitetutorial.net/>

My se nebudeme zabývat všemi příkazy a možnostmi, které jazyk nabízí. Vystačíme si se základními příkazy.

Příkazy jazyka se dělí do 3 skupin:

* DDL – Data definition language – Jazyk pro definici databázových objektů (tabulek)
  + Create, Alter, Drop
* DML – Data modification language – Jazyk pro modifikaci obsahu databáze
  + Insert, Update, Delete
* DQL – Data Query languade – Jazyk pro dotazování nad databází
  + Select

## Úkoly cvičení 3

1. Model z předchozího cvičení převeďte do fyzické podoby.

A diagram of a company

Description automatically generated

1. V programu SQLiteStudio vytvořte databázi (všechny 4 tabulky, včetně správného nastavení primárních a cizích klíčů)
2. Pomocí umělé inteligence (ChatGpt, Perplexity, BingCopilot) si vygenerujte data pro tabulky zaměstnanci a pracoviště. Pokuste se prompt napsat tak, abyste odpověď získali v podobě již připravených insertů, které jen provedete.
3. Vypište obsahy obou tabulek.
4. Z tabulky zaměstnanci vypište pouze muže
5. Z tabulky zaměstnanci vypište pouze ženy mladší 30 let.
6. Do tabulky vedoucí vložte záznamy tak, aby v aktuálním čase mělo každé oddělení právě jednoho vedoucího. Pokuste se vložit i údaje o historických vedoucích.
7. Pomocí AI si vygenerujte data, která budou představovat úvazky zaměstnanců na pracovištích za posledních 5 let.
8. Do tabulky zaměstnanci, přidejte sloupec plat
9. Všem zaměstnancům, nastavte počáteční hodnotu platu na 35000
10. Zaměstnancům starším 35 let navyšte plat o 20 procent.
11. Do tabulky pracoviště, přidejte nové pracoviště.
12. Následně toto pracoviště z tabulky vymažte.

# Cvičení 4

## Jazyk SQL, dotazování

## Cíl cvičení

Cílem tohoto cvičení je:

* JAZYK SQL – synstaxe a semantika
* Dotazování

Pracovat s nástrojem SQLiteStudio již umíme z minula. Tento týden se budeme snažit zaměřit se na dotazy nad více tabulkami.

### Jazky SQL

Structured Query language je standardizovaný jazyk používaný pro správu a manipulaci s daty v relačních databázích. Umožňuje uživatelům provádět různé operace, jako je vytváření, čtení, aktualizace a mazání dat (známé jako CRUD operace) prostřednictvím příkazů, které jsou syntakticky podobné zjednodušené angličtině.

Existuje velké množství tutoriálů pro jazyk SQL. V některých případech můžete najít drobné odchylky, to může být způsobeno aplikací jazyka pro konkrétní typ databázového systému. Nicméně pro základy, které my v tomto předmětu budeme řešit jsou rozdíly nepodstatné.

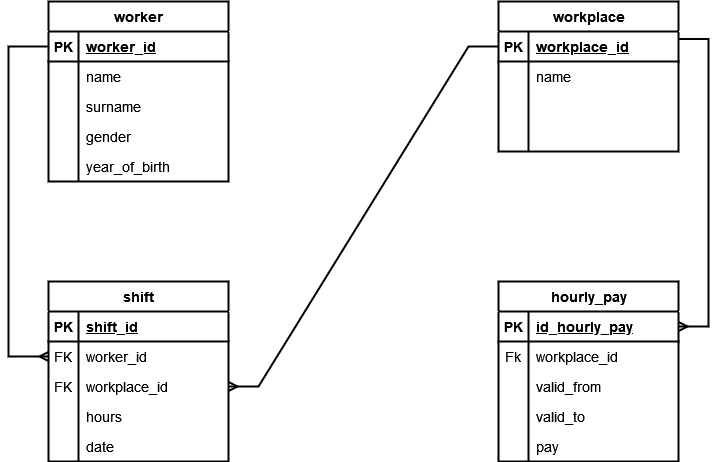
* <https://www.w3schools.com/sql/>
* <https://www.sqlitetutorial.net/>

My se nebudeme zabývat všemi příkazy a možnostmi, které jazyk nabízí. Vystačíme si se základními příkazy.

Příkazy jazyka se dělí do 3 skupin:

* DDL – Data definition language – Jazyk pro definici databázových objektů (tabulek)
  + Create, Alter, Drop
* DML – Data modification language – Jazyk pro modifikaci obsahu databáze
  + Insert, Update, Delete
* DQL – Data Query languade – Jazyk pro dotazování nad databází
  + Select

## Úkoly cvičení 4



Otevřete si přiložený databázový soubor a proveďte následující příkazy

1. Vypiště obsah celé tabulky worker (pracovníci)

SELECT \* from worker

1. Vypiště pouze jméno, příjmení a pohlaví všech pracovníků

SELECT name,surname from worker where gender="M"

1. Vypište přijmení věk všech mužů

SELECT name,surname,

2024-year\_of\_birth vek

from worker where gender="M"

1. Vypište jméno a příjmení všech žen, které se narodili po roce 1992

SELECT name,surname

from worker

where gender="F" and year\_of\_birth>1992

1. Vypište jména a příjmení všech pracovníků v pořadí od nejmladšího po nejstaršího.

SELECT name,surname, year\_of\_birth

from worker

order by year\_of\_birth

1. Do tabulky worker přidejte nového pracovníka (Richard, Newton, 1994, muž)

Insert into worker

(name, surname, gender,year\_of\_birth)

values ('Richard','Newton','M',1994)

1. Do tabulky shift (směna) přidejte nově přidanému pracovníkovi 6 hodinovou směnu na pokladně, která proběhla 25.2.2023.

Insert into shift

(worker\_id,workplace\_id,hours,date)

values (101,1,6,"2022-02-23")

1. Z tabulky shift vymažte všechny směny, které proběhly 24. 12. 2022

DELETE from shift where date="2022-12-24"

1. V tabulce hourly\_pay navyšte všechny mzdy o 10 procent

UPDATE hourly\_pay set pay=pay\*1.1

1. V tabulce hourly\_pay snižte všechny mzdy náležející k pracovišti sklad (storeroom) o 10 korun

UPDATE hourly\_pay set pay=pay-10

where workplace\_id=2

1. Vypište seznam (jmeno a prijmeni pracovnika, nazev pracoviste, datum) seřazený podle datumu vzestupně

SELECT worker.name ||" "||surname pracovnik,

date datum,

workplace.name pracoviste

from worker

join shift using(worker\_id)

join workplace using(workplace\_id)

order by date

1. Vypište seznam směn (nazev pracoviště, délka směny, datum) pro pracovníka s worker\_id rovným číslu 5.

SELECT worker.name ||" "||surname pracovnik,

hours,

date datum,

workplace.name pracoviste

from worker

join shift using(worker\_id)

join workplace using(workplace\_id)

where worker\_id=5

1. Vypište kolik zástupců mezi pracovníky mají jednotlivá pohlaví.

SELECT gender,count(\*) from worker

group by gender

1. Vypište průměrný věk mužu a žen

SELECT gender,avg(2024-year\_of\_birth)

from worker

group by gender

1. Vypište seznam směn (nazev pracoviště, délka směny, datum) pro pracovníka „Františka Podhorská“.

SELECT hours,date datum,

workplace.name pracoviste

from shift

join workplace using(workplace\_id)

where worker\_id=(SELECT worker\_id from worker

where name="Ferdinand" and

surname="Mravenec")

1. Vypiště seznam pracovníků (jméno a příjmení) s celkovým počtem odpracovaných hodin

SELECT worker.name ||" "||surname pracovnik,

sum(hours)

from worker

join shift using(worker\_id)

group by pracovník

1. Vypište seznam pracovníků s počem hodin odpracovaných na jednotlivých pracovištích  
   SELECT worker.name ||" "||surname pracovnik,

sum(hours),

workplace.name pracoviste

from worker

join shift using(worker\_id)

join workplace using(workplace\_id)

group by pracovnik, pracoviste

order by pracovnik

1. Vypiště seznam pracovníků (jméno a příjmení) s celkovým počtem odpracovaných směn

SELECT name,surname, pocet

from

(SELECT worker\_id,count(\*) pocet from shift

group by worker\_id) join

worker using(worker\_id)

1. Vypiště seznam pracovníků (jméno a příjmení) s jejich celkovou mzdou

SELECT name,surname, plat from

(SELECT worker\_id,sum(hours\*pay) plat from shift join hourly\_pay

on (shift.workplace\_id=hourly\_pay.workplace\_id

and date between valid\_from and valid\_to)

group by worker\_id) join

worker using (worker\_id)

1. Vypište seznam pracovišť (název) se součtem na nich vyplacených mezd.

SELECT name, plat from

(SELECT shift.workplace\_id,sum(hours\*pay) plat from shift join hourly\_pay

on (shift.workplace\_id=hourly\_pay.workplace\_id

and date between valid\_from and valid\_to)

group by shift.workplace\_id) join

workplace using (workplace\_id)

# Cvičení 5

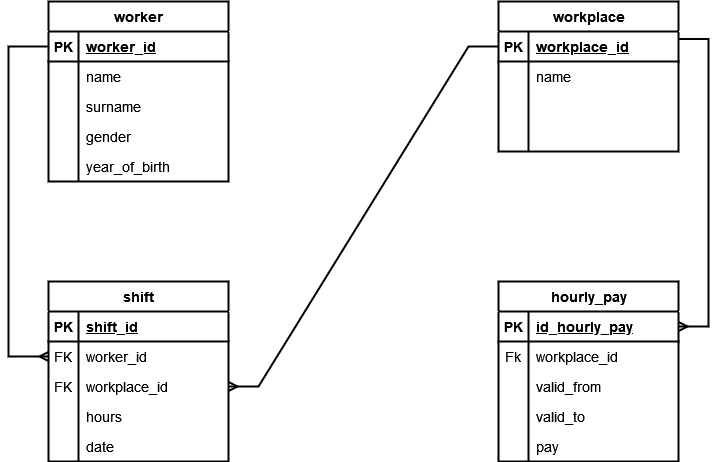
## Pokročilejší dotazy

## Cíl cvičení

Cílem tohoto cvičení je:

* Procvičovat dotazování pomoci jazyka sql

## Úkoly cvičení 5



Otevřete si přiložený databázový soubor a proveďte následující příkazy

1. Zjistěte, jaká byla průměrná hodinová mzda jednotlivých pracovníků?

SELECT name,surname, plat/hodiny prum from

(SELECT worker\_id,sum(hours\*pay) plat,

sum(hours) hodiny

from shift join hourly\_pay

on (shift.workplace\_id=hourly\_pay.workplace\_id

and date between valid\_from and valid\_to)

group by worker\_id) join

worker using (worker\_id)

1. Který pracovník odpracoval nejvíce hodin na pracovišti „Cashdesk“?

SELECT name,surname, pocet\_hodin from

(select worker\_id,sum(hours) pocet\_hodin

from shift

where workplace\_id=1

group by worker\_id

order by pocet\_hodin desc)

join worker using(worker\_id)

1. Vypište průměrný věk všech pracovníků, jejichž příjmení obsahuje písmeno „r“.

SELECT avg(2024-year\_of\_birth)

FROM worker

where name like "%r%"

1. Kolik hodin celkem odpracoval nejstarší pracovník?

SELECT sum(hours) from shift

where worker\_id=

(SELECT worker\_id from worker where

year\_of\_birth=(SELECT min(year\_of\_birth) from worker))

nebo:

SELECT worker\_id,sum(hours) from shift join

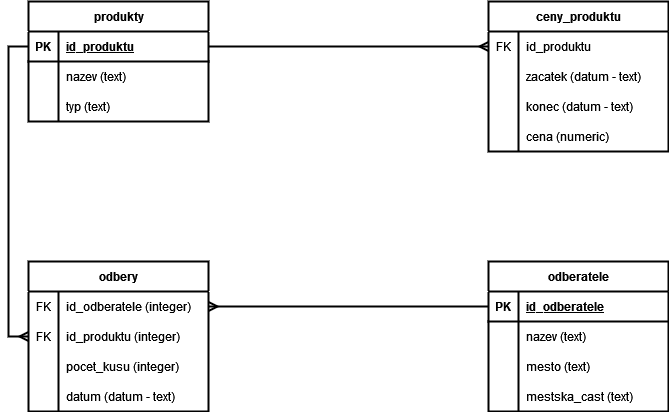
(SELECT worker\_id from worker where

year\_of\_birth=(SELECT min(year\_of\_birth) from worker))

using (worker\_id)

group by worker\_id

Pronásledující úkoly si stáhněte soubor prednaska.db (najdete jej na dokumentovém serveru). Datový model je následující. Pro vytvoření dotazů se pokuste použít umělou inteligenci. Avšak nechte si od ní dotazy vysvětlit, tak abyste jim rozuměli.



1. Kolik druhů produktů jednotlivých typů pekárna vyrábí?

select typ,count(typ) from produkty

group by typ

1. Kolik odběratelů se nachází v jednotlivých městech?

select mesto,count(\*) from odberatele

group by mesto

1. Ve kterém městě se nachází nejvíce odběratelů?

select mesto,count(\*) pocet from odberatele

group by mesto

order by pocet desc

1. Který odběratel odebral nejvíce rohlíků?

SELECT nazev,pocet from

(SELECT id\_odberatele,sum(pocet\_kusu) pocet

from odbery

where id\_produktu=(SELECT id\_produktu

from produkty where nazev="Rohlík")

group by id\_odberatele

order by pocet desc)

join odberatele using(id\_odberatele)

1. Kolik kusů jednotlivých produktů bylo v měsíci lednu celkem odebráno.

select nazev, pocet from produkty join

(select id\_produktu, sum(pocet\_kusu) pocet

from odbery where datum like "2024-01%"

group by id\_produktu)

using (id\_produktu)

# Cvičení 6

## Opakování a Úvod do algoritmizace

## Cíl cvičení

Cílem tohoto cvičení je:

* Zopakovat si předchozí látku
* Nakreslit svůj první algoritmus

## Opakování

* Transformace dat – číselné soustavy
  + Převod dat
  + Počítání v různých číselných soustavách
* Datové modelování
* Jazyk SQL
  + Syntaxe jazyka
  + Vytváření dotazů

## Úkoly

### Pro následující příběh nakreslete ER diagram

Provozujeme malou knihovnu a potřebujeme evidovat naše čtenáře, knihy, exempláře knih a výpůjčky. Každá kniha může patřit do více žánrů, což nám umožní lépe organizovat naše tituly a usnadnit čtenářům jejich výběr. Vaším úkolem je navrhnout Entity-Relationship Diagram (ERD), který zahrnuje 4 až 5 tabulek a správně určuje atributy pro každou entitu. U **čtenáře** potřebujeme ukládat informace o jménu, příjmení, datu narození, adrese a kontaktu. U **knih** pak zaznamenáváme autora, název, rok vydání a žánr. Pro **exempláře knih** bychom měli mít atributy jako ID exempláře, ID knihy (cizí klíč) a stav. U **výpůjček** potřebujeme evidovat datum výpůjčky, datum vrácení a ID čtenáře (cizí klíč). Vytvořte diagram, který jasně znázorňuje vztahy mezi těmito entitami, a nezapomeňte napsat krátký popis každého vztahu.

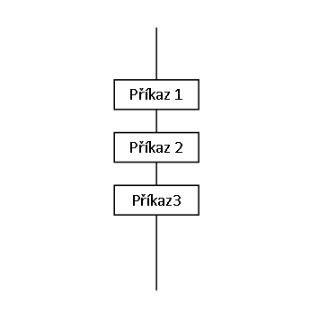
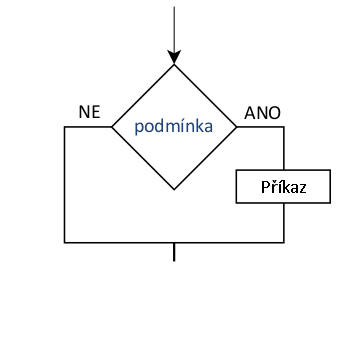
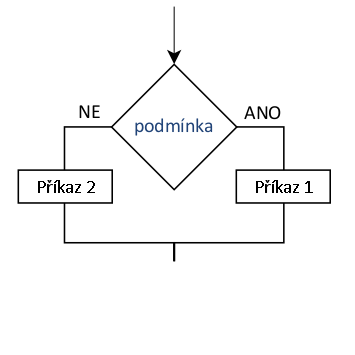
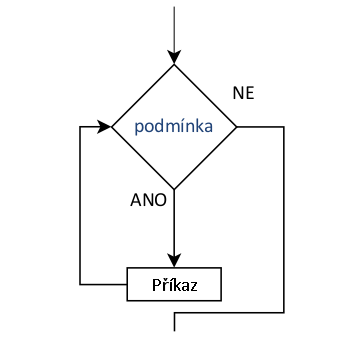
## Úvod do algoritmizace

Je nutné si uvědomit, že algoritmizace je aktivitou, kterou běžně ve svém životě provádí každý z nás. Je součástí běžné lidské komunikace. Jedná se o situace, kdy chceme po někom, aby pro nás něco udělal, nebo třeba vysvětlujeme, proč jsme něco sami udělali a jak jsme při nějaké činnosti postupovali.

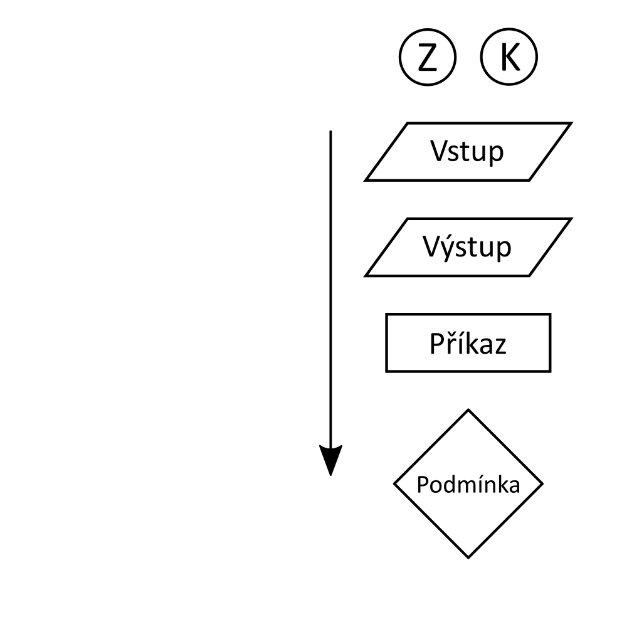
Algoritmizace je činnost. Algoritmus je přesný popis postupu činnosti. Algoritmizace má tedy svou analytickou a kreativní stránku. Abychom bylo schopni napsat dobrý algoritmus, musíme prvně dokonale porozumět řešenému problému. Znát jeho principy a všechny možné alternativy. Tz. Chceme-li umět napsat algoritmus, musíme algoritmizovaný proces dokonale znát.

Pokud jej známe, tak už jen záleží na naší schopnosti předávat jednoznačně předávat informace a jazyku (přirozeném, grafickém, umělém), který budeme používat.

My v následujících cvičeních se budeme snažit algoritmy zapisovat graficky. Budeme k tomu používat tzv. **jazyk vývojových diagramů**.



Sekvence neúplný podmíněný příkaz úplný podmíněný příkaz cyklus



## Úkoly

1. Nakreslete algoritmus pro zjištění vzdálenosti dvou libovolných čísel na číselné ose.
2. Nakreslete algoritmu pro nalezení všech možných řešení kvadratické rovnice.

# Cvičení 7

## Jednoduché algoritmy

## Cíle cvičení

* Podpořit analytické myšlení
* Naučit se přemýšlet nad činnostmi
* Zlepšit schopnost popisovat, jak se má něco udělat

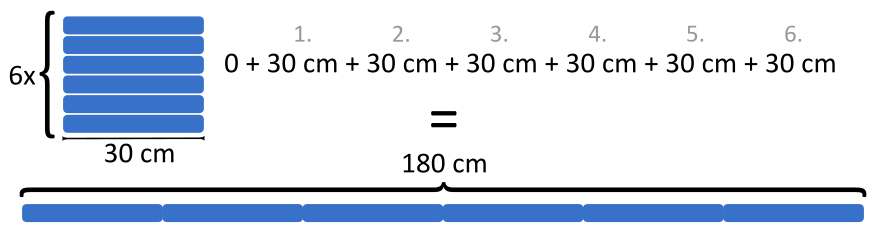
## Úvod

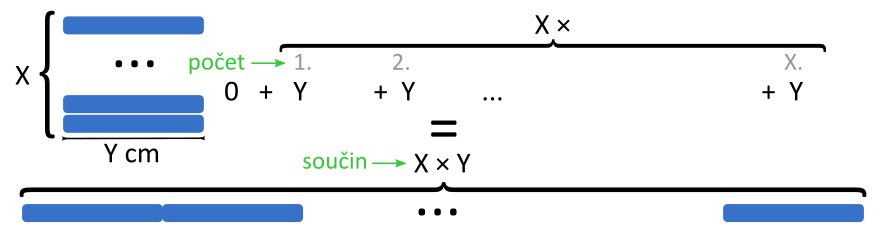
Každá činnost představuje posloupnost nějakých, kroků (operací), které musí být ve správném pořadí za stanovených podmínek provedeny. Všechno, co v životě děláme je nějaká činnost. Většinou tyto činnosti mají nějaký účel a důvod. V naprosté většině jsou spuštěny nějakým impulzem a po jejich dokončení očekáváme nějaký výsledek. Pokud víme, jak se něco dělá, měli bych být schopni tuto činnost popsat. A tento popis být schopni předat někomu nebo něčemu jinému, aby byla provedena.

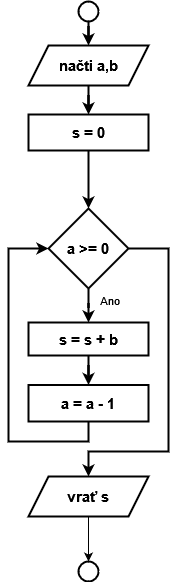
Dnes se budeme snažit zapisovat jednoduchých algoritmy činností, které byste měli znát. Zkuste tedy zavzpomínat, jak jste se na základní škole třeba učili počítat.

**Jak jste si třeba vysvětlovali, co je to součin?**

Je to opakované přidávání stejné hodnoty. Hodnotu přičítáme tolikrát, kolikrát hodnotu násobíme.







Tento příklad se zaměřuje pouze na kladná čísla. Jak by to vypadalo, kdyby měl algoritmus pracovat se všemi celými čísly?

## Úkoly

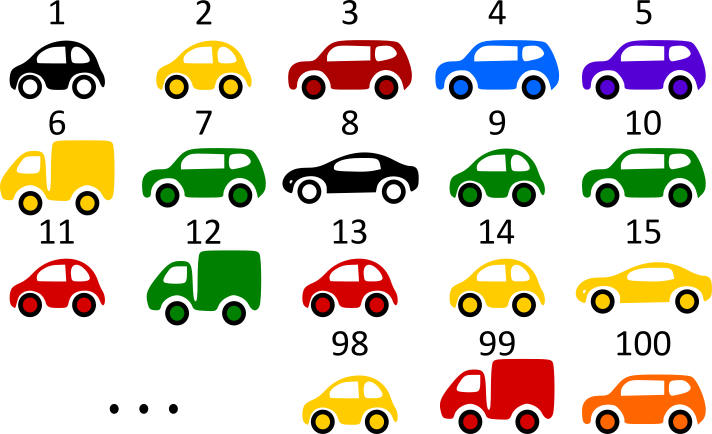
1. Nakreslete algoritmus pro nalezení zbytku po dělen dvou přirozených čísel
2. Nakreslete algoritmus pro nalezení součinu dvou celých čísel (kladných i záporných) bez užití operace krát
3. Nakreslete algoritmu pro nalezení největšího společného dělitele dvou přirozených čísel
4. Nakreslete algoritmus pro rozpoznání typu trojúhelníku na základě zadaných délek stran
5. Nakreslete algoritmus pro zjištění, jestli zadané číslo je nebo není prvočíslem.

# Cvičení 8

## Zpracování číselných řad a seznámení s jazykem Python

Algoritmizace a analýza řad dat představují klíčový aspekt v mnoha oblastech vědy a techniky ekonomiku a ekonomii nevyjímaje. Datové řady jsou posloupnosti hodnot, které jsou třeba chronologicky uspořádány a umožňují sledovat dynamiku jevů, jako jsou ekonomické ukazatele, meteorologické údaje nebo biologické procesy. Tyto řady poskytují cenné informace o trendech, sezónních vzorcích a cyklických fluktuacích, což je nezbytné pro predikci budoucího vývoje.

Můžeme si představit různé situace, například, že budeme sledovat provoz na křižovatce a sledovat její časové vytížení a charakter dopravních prostředků.



Sledujeme určité jevy, které v každý okamžik nastanou a provedeme

1. V šestnáctkové soustavě nám u hodnoty 9 dojdou číslovky. Pro číslice reprezentující hodnoty 10 až 15 použijme písmena anglické abecedy (A, B, C, D, E, F). [↑](#footnote-ref-1)