Softwarové technologie

[Logický program 3](#_Toc164440828)

[Predikáty 3](#_Toc164440829)

[Syntaxe Prologu 4](#_Toc164440830)

[Databáze 7](#_Toc164440831)

[Databázový systém (DBS) 7](#_Toc164440832)

[Historický vývoj a modely dat 7](#_Toc164440833)

[Systémy pro správu souborů 7](#_Toc164440834)

[Hierarchické DBS 8](#_Toc164440835)

[Síťové DBS 8](#_Toc164440836)

[Relační DBS 8](#_Toc164440837)

[Objektově relační DBS 8](#_Toc164440838)

[Objektové DBS 9](#_Toc164440839)

[Relační algebra 9](#_Toc164440840)

[Základní operátory relační algebry 9](#_Toc164440841)

[SQL 10](#_Toc164440842)

[Základní příkazy 10](#_Toc164440843)

[Parametry příkazu SELECT 10](#_Toc164440844)

[Agregační funkce 10](#_Toc164440845)

[Pohledy 10](#_Toc164440846)

[Uložená procedura 10](#_Toc164440847)

[Trigger 11](#_Toc164440848)

[Transakce 11](#_Toc164440849)

[Privilegia 11](#_Toc164440850)

[Konceptuální modelování 12](#_Toc164440851)

[Základní pojmy 12](#_Toc164440852)

[Konceptuální modelování 12](#_Toc164440853)

[ER model 12](#_Toc164440854)

[Relační model 13](#_Toc164440855)

[Transformace ER modelu do relačního modelu 13](#_Toc164440856)

[Měřítka, zda je DBMS relační 14](#_Toc164440857)

[Databázová relace 14](#_Toc164440858)

[Normální formy relací 15](#_Toc164440859)

[Znalosti 16](#_Toc164440860)

# Logický program

* Jazyk pro programování symbolických výpočtů
* Založen na predikátové logice
  + "x(má\_rád(x,pivo) Þmá\_rád(Honza, x))
  + Prolog: má\_rád(Honza, X) :- má\_rád(X, pivo)
* Název odvozen z PROgramování v LOGice
* Jeho úspěch pro vznik nové disciplíny matematické informatiky => logického programování
  + Specifické oblasti použití
  + Umělá inteligence
  + Znalostní inženýrství
  + Databázové a expertní systémy
  + Podpora specializovaných činností, např. projektování (CAD), výuce (CAI) aj.
* Porovnání s konvenčními jazyky
  + Jde o neprocedurální jazyk
  + Lze se více soustředit na popis vlastností relací – tedy co se má vypočítat
  + Bez nutnosti řešit, jak se to má udělat
  + Deklarativní
    - Při psaní programu deklarujeme fakta a pravidla pro popis vlastností a vztahů
  + Konverzační
    - Uživatel klade dotazy
    - Prolog odpovídá
  + Interaktivní
    - Pokud uživatel dotazy neklade, Prolog nepracuje a čeká
* V Prologu nejsou příkazy pro řízení běhu výpočtu ani příkazy pro řízení toku dat
* Chybějí prostředky pro programování cyklů, větvení, …
* Nepoužívá se přiřazovací příkaz
* Rozdílná úloha proměnných
  + Proměnná v Prologu označuje po dobu výpočtu objekt, který vyhovuje určitým podmínkám
  + Jeho vymezení se při výpočtu upřesňuje
* Průběh výpočtu v Prologu řízen jeho interpretem na základě znění programu
* Programátor může chod výpočtu ovlivnit řídícími příkazy v mnohem menší míře než u jiných jazyků
* Původně navržen jako specializovaný na symbolické výpočty, moderní implementace směřují k obecnějšímu použití

## Predikáty

* Vyjadřují fakta, která mají význam vztahů mezi objekty
* Píše se stejně jako funktor
* 2 základní režimy
  + Konzultační režim
    - Slouží k samotnému vytváření programu
  + Dotazovací režim
    - Kladou se otázky pomocí predikátů
  + Otázka vždy začíná otazníkem
  + Další výsledek se prolog pokusí najít po zadání středníku, stisknutím samotného enteru prolog ukončí dotazování

## Syntaxe Prologu

* Prolog se skládá z termů
* Jako termy se označují konstanty, proměnné, struktury
* Konstanty
  + Integer
    - Celá čísla
    - Rozsah záleží na implementaci
    - Většina moderních implementací i s reálnými
  + Atomy
    - Sekvence znaků začínající malým písmenem
    - Nebo řetězec uzavřený v apostrofech
    - Speciální atomy: ? – :- ! .
* Proměnné
  + Sekvence znaků začínající velkými písmeny nebo podtržítkem
  + Samotné podtržítko = anonymní proměnná
    - Její hodnota není podstatná
    - Nebude dále využívána
* Struktury
  + Objekt skládající se z jiných objektů
  + Komponenty lze spojovat pomocí funktoru
* Term je zapsán jako sekvence znaků
  + Malá a velká písmena
  + Číslice
  + Speciální znaky
* Vestavěné predikáty
  + Procedury nabízené danou implementací Prologu
  + Velké množství predikátů nejen pro výpis na obrazovku a také získávání dat od uživatele, ale také na klasifikaci a konverzi termů, groupování a ladící příkazy
  + Jsou považovány za termy
* Konstanty a proměnné se označují jako atomické termy
  + Z atomických termů se dají vytvářet složitější výrazy = struktury
* Programování v Prologu spočívá
  + V deklarování faktů o objektech a relacích mezi nimi
    - Fakta
      * Nepodmíněné výrazy, vyjadřují skutečnost o relaci
      * Jde o vztahy, které mají jméno, za kterým následuje v kulatých závorkách výčet objektů, kterých se vztah týká
      * Každý fakt je ukončen tečkou
      * matka(Matka, Dite).
  + Definování pravidel
    - Podmíněné výrazy
    - Dvě části, oddělené :-
    - V levé části je hlava pravidla, v pravé tělo pravidla
      * sourozenci(X,Y) :- matka(M,X), matka(M,Y)
  + Kladení a zodpovídání dotazů
    - Začíná dvojicí znaků ?-, končí tečkou
    - Dotaz, který neobsahuje proměnné se nazývá základní otázka
    - Výsledkem je odpověď
      * Výpis hodnot proměnných, které jsou součástí otázky yes/no
    - No může znamenat
      * Zápornou odpověď
      * Neexistující odpověď
* Seznamy
  + Prakticky jedinou předdefinovanou datovou strukturou
  + Tvořen posloupností prvků
    - Oddělených čárkami
    - A uzavřených do hranatých závorek
  + Vytváření
    - Funktor .
  + Seznam se skládá z hlavy a těla
    - Výjimkou je prázdný seznam
  + Syntaktická pomůcka
    - Svislítkem lze oddělit hlavu od těla
    - [ H | T ]
  + Operace se seznamy
    - Append – spojování
    - Member – zjistí, zda je prvek členem
    - Length – délka seznamu
    - Delete – odstraní prvek ze seznamu
  + Seznam je rekurzivně definovaná struktura
    - I procedury s ním pracující jsou rekurzivní
* Databáze
  + Můžeme s výhodou použít tam, kde potřebujeme aktivně přistupovat k bázi znalostí
  + Největší předností je možnost měnit program během jeho běhu
* Řízení databáze
  + Assert, asserta, assertz – přidá do DB
  + Retract, retractall – odstraní z DB
  + Listing – výpis klauzulí v DB
  + Clause – hledá klauzule
* Řízení průchodu programem
  + ! (řez)
    - Představuje klíčové rozhodnutí
    - Překročí-li jej, už se nevrací
    - Červený
      * Pokud jej z programu smažu, bude program pracovat chybně
    - Zelený
      * Pokud jej smažu, může dojít ke snížení efektivity, ale funkce se nezmění
  + Fail
    - Nikdy nesplněný predikát
    - Nelze se přes něj dostat
  + Repeat
    - Při průchodu programem blokuje návrat
    - Nekonečný cyklus
      * Repeat
      * Write(‘Pracuj’), nl,
      * Fail.

# Databáze

* Báze dat
* Všechna potřebná data dané organizace

## Databázový systém (DBS)

* Specializovaný SW pro efektivní práci s daty
* DBS umožňuje shromažďovat různé informace
* Tyto informace ukládá a udržuje v platném stavu na centrálním místě
* Tvořen několika části
  + DBS = SŘBD + DBA + DB
  + SŘBD
    - Systém řízení báze dat
    - Program, jehož úkolem je pracovat s uloženými informacemi
    - Organizuje je a udržuje v platném stavu
  + DBA
    - Databázová aplikace
    - Program pro manipulaci s daty prostřednictvím SŘBD
  + DB
    - Databáze, množina informací
* Jednotlivé části DBS se mohou provozovat na jednom PC nebo na různých
* Historicky byly často SŘBD a DBA spojeny do jednoho programu
* V dnešní době nejpoužívanější způsoby zpracování dat databázové systémy klient/server
  + SŘBD a DBA od sebe odděleny
* Hlavní funkce
  + Definice DB
  + Efektivní manipulace DB
  + Souběžný přístup
  + Ochrana dat
  + Zotavení z chyb

## Historický vývoj a modely dat

* Databázový model je popisem fungování DB
* Hlavním úkolem je popsat, jakým způsobem jsou data zpřístupňována uživatelům
* Uživateli se data jeví jiným způsobem, než jak jsou ve skutečnosti uložena
* Využívá se abstrakce

### Systémy pro správu souborů

* Nejjednodušší databázový model
* Jako jediný přímo popisuje způsob uložení dat na disku
* Všechny položky ukládány sekvenčně do jednoho souboru
* Pokud je třeba najít informaci, nutno prohledat celý soubor od začátku
* Výhodou je jednoduchost
* Nevýhody
  + Mezi uloženými údaji nejsou vztahy
  + Problémy s integritou – žádná kontrola správnosti ukládaných dat
  + Pomalé vyhledávání informace
  + Obtížné změny struktury

### Hierarchické DBS

* Organizuje data do stromové struktury
* Vždy jeden kořenový uzel, jehož vlastník je SŘBD
* Z kořene vedou ukazatele k uzlům 1. úrovně v nichž začíná vlastní DB
* Každý uzel první úrovně může mít několik synovských uzlů
* Jednotlivé položky jsou umístěny na různých větvích
* Na fyzické struktuře na disku nezáleží, tento úkol řeší SŘBD
* Z diagramu je obtížné určit, z jakých položek se skládá záznam
* Velmi obtížná změna struktury DB
  + V tomto případě je nutné vytvořit novou strukturu a překopírovat do ní údaje
* Umožňuje definovat vztahy 1:N
* Efektivnější vyhledávání informace
  + Prohledávají se příslušné části stromu
* Neumožňuje v vztahy M:N

### Síťové DBS

* Popisuje DB, v nichž existují vztahy M:N
* Je založena na ukazatelích, které vyjadřují vztahy mezi jednotlivými položkami
* Vzájemné vztahy mezi množinami mohou být velmi komplikované
* Obdobné nevýhody se změnou struktury jako u Hierarchické DBS

### Relační DBS

* Z roku 1969, kdy E. F. Codd popsal model založený na matematickém pojmu relace
* Nevyužívají se rodičovské vztahy mezi položkami
* Data jsou organizována do uspořádaných n-tic
* Jednotlivé n-tice představují řádky tabulky
* Každý záznam je řádkem tabulky a každá položka sloupcem
* Při definování schématu je využíván proces normalizace, při němž dochází k dekompozici dat do podmnožin pro jednotlivé tabulky
* Výhody
  + Realizuje vztahy M:N
  + Snadná změna schématu
  + Změna struktury spočívá v přidání/odebrání sloupce
  + Zachování integrity
  + Pokud je SŘBD skutečně relační musí znemožnit přístup k datům jinými kanály než prostřednictvím sebe sama
  + Programy pracují s daty, aniž by znaly jejich umístění v DB
  + K veškeré manipulaci s daty využití služeb SŘBD

### Objektově relační DBS

* Pokus integrovat objekty a relace do jednoho systému
* Tři možné přístupy
  + Objektově relační datové manažery manipulující objekty do relačních tabulek
  + Relační obálky tvořené knihovnami, které se přilinkují k relační DB. Relační obálka detekuje změny objektů a promítá je pomocí SQL do relační DB a změny v relační DB promítá do objektů
  + Objektově relační DB ukládající data pomocí relací i jako objekty

### Objektové DBS

* Systém vybudovaný pomocí objektově orientovaných metod
* Každá komponenta zapouzdřuje data a funkce
* Komponenty dědí atributy a chování z jiných komponent a navzájem komunikují pomocí zpráv
* První pokusy vycházely z existence objektově orientovaných programovacích jazyků

## Relační algebra

### Základní operátory relační algebry

* Kartézský součin
* Sjednocení
* Průnik
* Rozdíl
* Projekce
* Selekce
* Spojení
  + Přirozené spojení
  + Levé polo-spojení
  + Pravé polo-spojení

#### Projekce

* Tabulky definované podmnožinou atributů představuje vypuštění některých sloupců neobsažených v požadované množině atributů
* Součástí projekce je i případné vypouštění duplicitních řádek ve výsledné tabulce
* Tabulku lze také rozšířit o sloupec vzniklý nějakou operací nad hodnotami uložených atributů

#### Selekce

* Tabulky definované podmnožinou definičního oboru relace představuje průnik relace novou podmnožinou

#### Spojení

* Vytvoření nové spojené tabulky na základě shodných hodnot atributů v obou tabulkách
* Podle způsobu porovnávání hodnot ve spojovaných sloupcích se rozlišuje na
  + Rovnost
  + Nerovnost – nemá praktický význam
  + Vnější
* Pokud výsledný operace obsahuje všechny sloupce z první i z druhé tabulky, vyjma sloupce, který byl využit pro spojení, pak toto spojení je označováno jako přirozené

## SQL

* Strukturovaný dotazovací jazyk, který byl určen jako standard pro komunikaci s relačními databázemi
* Nezávislý na datech
* Neprocedurální jazyk
  + Popíšeme, jaká data chceme najít, ne jak se to má provést
* 2 hlavní komponenty
  + DDL – definuje strukturu tabulek
  + DML – vyhledává a manipuluje s daty

### Základní příkazy

* SELECT
* INSERT
* UPDATE
* DELETE

### Parametry příkazu SELECT

* FROM
* WHERE
* GROUP BY
* HAVING
* ORDER BY
* DISTINCT

### Agregační funkce

* COUNT
* SUM
* AVG
* MIN
* MAX

### Pohledy

* Objekt, který uživateli poskytuje data ve stejné podobě jako tabulka
* Obsahuje předpis, jakým způsobem mají být data získána z tabulek a jiných pohledů
* Virtuální relace, který se vytvoří v okamžiku vyžádání
* Výhody
  + Soustřeďují potřebná data pro uživatele
  + Skrývají složitost podkladových dat
  + Zjednodušují správu uživatelských oprávnění
  + Definují uspořádání dat pro export do jiných aplikací

### Uložená procedura

* Databázový objekt, který neobsahuje data, ale část programu, který se nad daty v databázi má vykonat

### Trigger

* Definuje činnosti, které se mají provést v případě definované události nad tabulkou
* Například při smazání nebo vložení dat

### Transakce

* Skupina, příkazů, která se navenek tváři jako jeden příkaz
* 3 příkazy
  + BEGIN
  + ROLLBACK
  + COMMIT

### Privilegia

* Akce, které uživatel může realizovat nad tabulkou nebo pohledem
* Mohou být omezena na vyjmenované databáze, tabulky nebo sloupce
* Vlastník databáze uděluje ostatním potřebná privilegia pomocí GRANT
* Odebírá pak pomocí REVOKE

# Konceptuální modelování

## Základní pojmy

* Podle ANSI/SPARC architektury lze rozdělit na tři úrovně návrhu
  + Externí hladina
    - Odpovídá pohledu jednotlivých skupin uživatelů
  + Konceptuální hladina
    - Celkový logický pohled na data
    - Její vytvoření = první krok databázového návrhu
  + Interní hladina
    - Popis dat na nižší úrovni
    - Interface pro operační systém

### Konceptuální modelování

* Nejčastěji se používá ER model a různé formy jeho rozšíření
* Návrh struktury dat da konceptuální úrovni
* Analýza požadavků a jejich zobrazení pomocí grafických prostředků
* Základní prvky
  + Entity
    - Objekty reálného světa
    - Schopny samostatné existence
    - Odlišný od ostatních objektů
  + Vztahy
  + Atributy
    - Vlastnost entity
    - Doména atributu – přípustné hodnoty
    - Jednoduchý atribut – dále nedělitelný
    - Složený atribut – dělitelný, př. Adresa
* Jeden z možných postupů je entitně relační modelování – ER
  + Různé notace
    - UML Class Diagram

## ER model

* Nástroj na popis konceptuálního schématu reality bez ohledu na to, jak bude daný model implementován v konkrétním DBMS
* Pracuje s
  + Typem entit
    - Množina objektů stejného typu
    - Objekt reálného světa, který je schopen nezávislé existence
  + Atributy
    - Vlastnosti entit
  + Vztahy
    - Propojení mezi entitami
  + Kardinalita vztahů
    - Vyjadřuje kolik entit jednoho typu může být ve vztahu s kolika entitami z druhého typu entit
  + IS-A hierarchie
    - *A is a B*, když typ entit B je zobecněním typu entit A, každý typ entit A je speciálním případem B
  + Klíči
    - Kandidátní klíč
      * Minimální množina atributů pro jednoznačnou identifikaci
    - Primární klíč
      * Kandidátní klíč, který byl vybrán k té identifikaci
* Rozšířený ER model = EER
  + Zavádí supertřídu / podtřídu
  + Existuje dědičnost vztahů
  + Pro grafické znázornění lze použít digram tříd z UML
  + Kromě prostých vztahů také kompozice a generalizace

## Relační model

* Základní struktura relace
  + Definována jako podmnožina kartézského součinu N množin
* Cílem zavedení byla
  + Disciplína v manipulaci s daty
  + Zabezpečení nezávislosti dat
  + Vyšší produktivita programování
* Databáze
  + Kolekce DB relací
    - Jsou reprezentované tabulkami
      * Všechny informace v tabulkách
  + Oproti matematické definici relace je databázová relace
    - Vybavena pomocnou tabulkou
    - Jménem relace, jmény atributů a definicí domén
    - Hodnoty jsou dále nedělitelné
  + Vlastnosti databázové tabulky
    - Jednoznačné jméno tabulky a jednotlivých sloupců
    - Všechny hodnoty v jednom sloupci jsou stejného typu
    - Na pořadí sloupců a řádků nezáleží
    - Nemá duplicitní řádky
    - Každá tabulka má primární klíč

### Transformace ER modelu do relačního modelu

* Zjednodušeně
  + ER model – s obrázky
  + Relační model – textově
* Typ entity = tabulka
  + Její sloupce odpovídají atributům
* Asociace
  + Tabulka, která obsahuje primární klíče účastníků vztahu
  + U 1:1 a 1:N není nutná nová tabulka
  + U 1:N stačí zahrnout cizí klíč na stranu N
  + 1:1 sloučit tabulky do jedné
* Kompozice
  + Do nové tabulky zahrnout klíč rodičovské tabulky

### Měřítka, zda je DBMS relační

#### Základní pravidla

* RDBMS musí být schopný manipulovat s daty pomocí operací relační algebry
* Má-li RDBMS jazyk nižší úrovně, tato nižší úroveň nemůže porušit pravidla integrity vyjádřené na vyšší úrovni jazyka
* Přístup do DB řídí RDBMS tak, aby integrita nemohla být porušena bez vědomi admina

#### Pravidla týkající se struktury dat

* RDBMS by měl podporovat
  + Relace, domény, primární klíče, cizí klíče
* Všechny informace jsou reprezentovány explicitně na logické úrovni
* Je-li pohled teoreticky upravitelný, měl by být i fakticky upravitelný

#### Pravidla týkající se integrity dat

* Jsou podporovány hodnoty NULL na reprezentaci chybějící informace
* Integritní omezení musí být definována v DDL jazyku a musí být uložitelné v systémovém katalogu
  + Centralizovaná kontrola integrity

#### Pravidla týkající se modifikace dat

* Ideální DBMS by měl podporovat 18 možností na manipulaci s daty
* Garantovaný přístup ke každé atomické hodnotě na základě jména tabulky, hodnoty primárního klíče a názvu sloupce
* Obsáhlý jazyk na manipulaci s daty, který by měl umožňovat
  + Definici dat
  + Definici pohledů
  + Modifikace dat
  + Integritní omezení
  + Autorizaci
* Transakce

#### Pravidla o nezávislosti dat

* Nezávislost dat od aplikace, která data používá
* Uživatelé i programátoři jsou izolováni od organizace dat na nižší úrovni

## Databázová relace

* Reprezentována tabulkou, která má následující vlastnosti:
  + Každá tabulka má jednoznačné jméno
  + Každý sloupec v tabulce má jednoznačné jméno
  + Všechny hodnoty daného sloupce musí být stejného typu
  + Nezáleží na pořadí sloupců
  + Nezáleží na pořadí řádků
  + Tabulka nemůže mít duplicitní řádky
  + Hodnoty jsou atomické
  + Každé tabulky musí mít primární klíče
* DB musí splňovat následující integritní omezení
  + Integritu entit
    - V databázové relaci hodnota primárního klíče nesmí být NULL
  + Referenční integritu
    - Obsahuje-li databázová relace cizí klíč, tak každá jeho hodnota musí být obsažena v rodičovské tabulce, anebo musí mít hodnotu NULL

## Normální formy relací

* K odstranění opakujících se skupin a nadbytečných informací slouží normalizace
* Sledují se 2 cíle
  + Zabránit ztrátě užitečné informace při zrušení řádku tabulky
  + Vyhnout se redundanci
* Normalizace probíhá ve 3 stupních
  + Odstranění nadbytečnosti
  + Mít zjednodušenou kontrolu integrity
  + Odolné vůči anomáliím při údržbě dat
* Proces normalizace tabulek
  + Formální technika pro návrh relačních DB tabulek
  + Postupná dekompozice původních tabulek na základě analýzy závislosti atributů
* 1 NF
  + Všechny hodnoty jsou atomické
* 2 NF
  + Splňuje požadavky 1. NF
  + Každý neklíčový atribut je plně závislý na primárním klíčí
* 3 NF
  + Splňuje požadavky 2. NF
  + Všechny neklíčové atributy jsou vzájemně nezávislé
* Funkční závislost
  + Je daná relace R(A,B), kde A, B mohou být složené atributy
  + B je funkčně závislý na A, když pro hodnotu A je jednoznačně daná hodnota B
  + B je plně funkčně závislý na A, je-li funkčně závislý na A a není funkčně závislý na žádné podmnožině A

# Znalosti

## Schéma pro reprezentaci znalostí

### Deklarativní reprezentace

* Hlavní oblasti aplikace jsou produkční systémy
* Důraz je kladen na vyjádřitelnost
* Produkční pravidlo reprezentuje část poznatku ve formě – jestliže platí, tak platí
* Báze faktů
  + Obsahuje informace o aktuální situaci ve stavovém prostoru řešení problému
  + Informace vychází z dat, které byly odvozeny výpočtem, měřením, nebo nebyly zadány uživatelem
* Báze pravidel
  + Obsahuje pravidla, která byla zadána znalostním inženýrem ve spolupráci s expertem
  + Pravidlo funguje na principu – předpoklad => důsledek
* Inferenční mechanismus
  + Jde o soubor programů zajišťující vlastní výpočty. Jeho úkolem je napodobit pochody b mysli experta
  + Jde o soubor programů zajišťující vlastní výpočty. Jeho úkolem je napodobit pochody b mysli experta
* Způsoby řízení
  + Zpětný režim (zpětné řetězení) - Postup od cíle (cílů) směrem k počátečnímu stavu.
  + Přímý režim (dopředné řetězení) - Aplikace produkčních pravidel od počátečního stavu k některému ze stavů cílových

### Asociativní reprezentace poznatků

* Je kladen důraz na to, aby se poznatky dostaly vzájemně do souvislostí
* Generují se tzv. asociativní (sémantické) sítě, které umožňují graficky reprezentovat relace mezi jednotlivými objekty reality
* Tvorba relací
  + Z vět v češtině lze pomocí predikátové logiky vyjádřit poznatky takto:
  + věta: Rostliny a živočichové jsou živé organismy.
  + JE(x, živočich) => JE(x, živý organismus)
  + JE(x, rostlina) => JE(x, živý organismus)

### Procedurální reprezentace poznatků

* Vyplývá z použitelnosti
* Postupy používání poznatků budou zachycené v podobě procedur
* Procedura je volaná cílem – zadám fakta a požaduji nová fakta (vede k deklarativnímu programování)
* Definuje se: „Co musím udělat, abych s danými předpoklady zjistil, zda daný objekt 'něco' je.
* Nedeklaruje se, co se má vypočítat, ale jak toho dosáhnout
* Má-li procedura vstupy k dispozici, pak rozhodne. Nemá-li je, pak se ze vstupů formulují podcíle a systém pokračuje v řešení

### Rámcová reprezentace poznatků

* Rámec je speciální údajovou strukturou na reprezentaci stereotypních situací
* Dříve, než přijdeme do styku s nějakou konkrétní situací, máme už s ní spojeny nějaké očekávání (defaults) a vědomosti
* Rámec si můžeme představit jako formulář, tedy soubor rubrik, které jsou vyplněny položkami rozličných typů
* Údaje v rubrikách mohou být opět názvy jiných rámců, čím se jednotlivé rámce mohou spojovat do hierarchických sítí

## Životní cyklus znalostní aplikace

* Disciplinovaný postup, pořadí, segmentace na proveditelné aktivity nebo fáze
* Dobrá dokumentace projektu (možné změny, modifikace)
* Koordinace projektu, aby byl dokončen včas
* Průběžná kontrola v každé fázi cyklu

|  |  |  |
| --- | --- | --- |
| Fáze | Klíčové otázky | Výstup |
| Vyhodnotit existující infrastrukturu | Jaký je problém?  Je tvorba systému opodstatněná?  Je vytvoření systému proveditelné? | Vyjádření cílů Výkonnostní kriteria Strategický plán |
| Vytvořit tým pro znalostní management | Kdo bude členem týmu?  Jak bude tým fungovat? | Standardizovaná procedura pro vytvoření systému |
| Zachycení znalostí | Jaké a čí znalosti budou zachyceny?  Jak se bude při zachycování znalostí postupovat? | Získané znalosti |
| Tvorba ZA | Jak budou znalosti reprezentovány? | Návrh ZA  HW/SW implementační detaily  Plán testů  Bezpečnostní, kontrolní a operační procedury |
| Verifikace a validace ZA | Jak spolehlivý je systém? | Kontrola funkčnosti systému experty Návody, tutoriály |
| Nasazení ZA | Co je právě v organizaci řešeno?  Jak jednoduché je systém používat? | Uživatelsky přívětivý systém  Výukový program |
| Řízení změn a motivačních programů | Poskytuje systém požadovaná řešení? | Spokojení uživatelé |
| Post-evaluace | Měl by systém být modifikován? | Spolehlivý a aktualizovaný systém |

## Tvorba znalostní aplikace

1. Tvorba znalostního týmu
   * Minimální složení
     + Doménový expert (primární, sekundární)
     + Znalostní inženýr
   * Další možné osoby
     + Konzultant
     + Propagátor projektu („champion“)
     + Experti pro doplňující tematické okruhy
     + Uživatelé (beta testing)
2. Zachycení znalostí
   * Analýza textů – cíl: získat základní povědomí o problémové doméně, šetřit čas experta, neobtěžovat jej vysvětlováním základů
   * Konzultace s expertem/experty – Cíl: Získat znalosti experta v použitelné – strojově zpracovatelné – podobě
   * Tvorba základního popisu znalostní domény – Cíl: Vytvoření základního konceptuálního popisu znalostní domény. Jedná se o přípravný krok při tvorbě ZA.
3. Tvorba ZA
   * Tvorba znalostní aplikace základní úrovně – Cíl: Vytvoření funkčního prototypu ontologie, ujasnění vzájemných souvislostí mezi třídami
   * Vytvoření plnohodnotné ontologie – Cíl: Vytvoření ucelené, logicky správné ontologie, která rozsahem odpovídá zaměření projektu
4. Verifikace a validace ZA
   * Testování znalostní aplikace – Cíl: Ověřit funkčnost aplikace a její využitelnost v praxi
5. Nasazení ZA v organizaci – Cíl: Zajistit využívání ZA uživateli a ověřit si další správnost fungování ZA v organizaci.

## Expert a znalostní inženýr

### Expert

* Specialista v určité problémové oblasti
* Efektivní v řešení problémů spadajících do jeho oboru
* Nezbytný při vývoji každé rozsáhlejší znalostní aplikace
  + Získávání znalostí
  + Verifikace znalostní aplikace
  + Ručí za správnost (odvozování, struktura apod.)

### Znalostní inženýr

* Úkolem znalostního je tvorba znalostního systému
* Má-li znalostní inženýr navrhnout znalostní systém musí
  + Získat znalosti od odborníka(ů) z předmětné oblasti,
  + Zaznamenat znalosti ve vhodném tvaru, tj. zkonstruovat model znalostí,
  + Vložit znalosti do počítačového programu, který znalostní systém reprezentuje,
  + Ověřit programový systém, tj. ujistit se, že skutečně obsahuje vložené znalosti a reaguje stejným způsobem, jako by reagoval člověk a že reakce je adekvátní vloženým znalostem

## Získávání znalostí a jejich uchovávání

### Získávání znalostí

* Interview – nejčastěji používané
* Pozorování (on-site observation) - znalostní inženýr pozoruje experta při práci
* Brainstorming – v rámci diskuse jsou kreativním myšlením vytvořeny řešení problému, všechny jsou posuzovány rovnocenně
* Protokolární analýza – Experti jsou dotazováni na postup řešení – kroky jsou zapsány
* Konsenzuální rozhodování – jasná shoda na nejlepším řešení problému, obvykle po brainstormingu
* Repertory grid – Doménový expert klasifikuje a kategorizuje problémovou doménu, vytváří vlastní model. Cílem je najít skryté principy, které experti při uvažování aplikují
* Technika číselných skupin (Nominal Group Technique NGT) – ohodnocení řešení se sečte, nejlépe hodnocené je vybráno jako konečné
* Metoda Delphi – Skupinové rozhodnutí by mělo konvergovat k optimálnímu nebo správnému řešení
* Blackboarding – Skupina expertů spolupracuje na řešení problému, který je uveden na tabuli. Tabule slouží jako jejich pracovní prostor. Obsahuje popis problému a vstupní data. Každý expert může k řešení přispět vyřešením části problému na tabuli. Na tabuli jsou postupně přidávána dílčí řešení, až je problém vyřešen celý

### Uchovávání znalostí

* Znalosti by měly být odděleny od lidí. Díky tomu mají všichni k informacím přístup a firma nepřichází o znalosti v případě odchodu zaměstnance
* Za tímto účelem jsou vytvářen znalostní systémy, nebo jsou alespoň znalosti uchovávány na jednom místě s různými úrovněmi přístupu

# Ontologické inženýrství

## Formální ontologie

* Popis sdílení dat nějaké domény
* Reprezentuje množinu tříd v doméně a vztahy mezi nimi
* Reprezentuje znalost nebo její část
* Etymologie
  + Z řeckého ontos (bytí, to, co jest) a logos (slovo, řeč, význam) \* Původně podoblast filosofie, která se zabývá bytím a podmínkami existence věcí okolo nás \* Oblast zájmu - co existuje nebo by mohlo existovat v našem světě, jsoucno a jeho podstata.
* V kontextu informatiky (Sowa):
  + „Předmětem ontologie je studium kategorií věcí, které existují nebo mohou existovat v určité doméně. Výsledek tohoto studia, nazývaný ontologie, je katalog věcí, jejichž existenci předpokládáme v dané doméně D, z perspektivy osoby používající jazyk L, aby mluvila o D. “
  + (Wikipedie) "Je to formální a deklarativní reprezentace, která obsahuje definici pojmů a tezaurus (=definici vztahů mezi jednotlivými pojmy). Ontologie je slovníkem, který slouží k uchovávání a předávání znalostí týkající se určité problematiky."
* Gruber:
  + „Ontologie je explicitní specifikace konceptualizace.“ (explicitní = znalost je jednoduše dostupná, vyjádřitelná; konceptualizace = systém pojmů modelující určitou část světa)
* Řeší, jak znalosti \* Získávat \* Sdílet \* Uchovávat (reprezentovat, modelovat) \* Zpracovávat a používat \* Předávat \* Vytvářet

### Komunikace člověk – člověk

* Stačí jednoznačná, neformální ontologie
* Komunikace mezi znalostním inženýrem a expertem
* Usnadňuje učení, vzájemné pochopení
* Podpora sběru požadavků

### Komunikace člověk – počítač

* Znovupoužitelnost
  + Formální model je možno používat opakovaně
    - Důležité entity, atributy, procesy a vztahy jsou již namodelované
  + Lze vytvářet různé aplikace, uživatelsky přizpůsobené obsahem i pojetím
    - Nad stejnou ontologií
* Vyhledávání
  + Základ indexu pro úložiště informací
* Spolehlivost
  + Možnost automatické kontroly konzistence
  + Formální modely umožňují strojové zpracování, vyhodnocení, kontrolu

### Komunikace počítač – počítač

* Podpora komunikace pomocí jednotného slovníku
* Zajišťuje jednoznačné reference na pojem/objekt
* Ontologie je součástí formátu výměny dat

## Základní prvky ontologie

* Jedinec
  + Základní stavební objekt datového modelu ontologie
* Třída
  + Množina jedinců určitého typu
  + Podmnožinou je podtřída
  + Může obsahovat jedince i podtřídy
* Atribut
  + Popisuje vlastnost, charakteristiku či parametry jedince
  + Každý atribut obsahuje přinejmenším název a hodnotu
* Vazba
  + Jednosměrné, nebo obousměrné spojení dvou jedinců
  + Typicky atribut jehož hodnota je jiný objekt v ontologii
  + Nejdůležitější typ vazeb je zařazení
  + Taxonomie vytváří neúplné uspořádání
  + Meronymy relationship (part-of) vztahy ukazují ja je vytvořen kompletní celek pomocí jednotlivých objektů

## Typy ontologií

### Dle expresivity jazyka a formálnosti

#### Informační

* Kresby, náčrtky, diagramy, schémata
* Pro organizaci myšlenek, podpora v rozhodování a učení
* Myšlenková mapa – poskytnutí informací o určitém předmětu zájmu, který je v centru
* Pojmová mapa – není centrální prvek, více vztahů mezi koncepty, pro celkový pohled na doménu

#### Lingvistické/terminologické

* Glosáře, slovníky, taxonomie, tezaury
* Reprezentace a definice slovníku – seznamu slov používaného daným jazykem
* Glosář
  + Vysvětluje ne zcela jasné pojmy
  + Využití přirozeného jazyka
* Taxonomie
  + Kolekce pojmů organizovaných do hierarchie
  + Nejčastější aplikace – strukturované slovníky pro klasifikaci zdrojů
* Tezaurus
  + Nabízí uživateli seznam synonym, antonym pro vyjádření významu

#### Softwarové

* Poskytují konceptuální schémata využitelné při analýzách a návrzích softwarových systémů
* Pohled na strukturu systému, nikoliv jeho funkcionalitu
* Např UML/AML

#### Formální

* Reprezentovány formálními jazyky s přesně definovanou sémantikou
* Nejvhodnější pro strojové zpracování znalostí
  + Mají vysokou úroveň formalizace
* Rámcové
  + Rámec jako struktura dat, která obsahuje informace o jedinci
  + Lze je strukturovat do podoby sítě
  + Inspirace při objektově orientovaném modelování
  + Syntéza deklarativního a procedurálního přístupu k reprezentaci dat
* Sémantické sítě
  + Založené na reprezentaci pomocí gramů
  + Uzel = koncept
  + Hrana = relace mezi uzly
  + Chybí jim matematická formalizace
  + Aplikace
    - Zpracování přirozeného jazyka
    - Znalostní a expertní systémy
    - Neformální způsob reprezentace znalostí
* Konceptuální grafy
  + Logický formalismus vycházející ze sémantických sítí
  + Strukturovaný do podoby orientovaného grafu
  + Struktura vhodná pro člověka, ale i pro strojové zpracování

### Dle specifičnosti

#### Generické

* Doménově nezávislé
* Zachycení obecných zákonitostí platných napříč problémovými oblastmi
* Využitelnost
  + Výchozí bod pro tvorbu nových ontologií
  + Reprezentace znalostí běžného uvažování
  + Definice návrhových vzorů pro běžně se vyskytující situace

#### Doménové

* Zaměřené na konkrétní věcnou oblast nebo její dílčí část
* UMLS, FMA, FOAF ontologie

## Reprezentace formálních ontologií

* Ontologie reprezentovány formálními, semiformálními nebo neformálními jazyky

### RDF

* Reprezentace metadat o webových zdrojích
* Dekomponuje znalosti do malých fragmentů – podmět, přísudek, předmět
* Pro jednoznačnou identifikaci věcí používá URI reference nebo konstantní hodnoty (literál)
* Pouze abstraktní přepis pro popis metadat/ontologie

### RDFS

* Nadstavba RDF pro tvorbu ontologií, která vznikla rozšířením syntaxe a sémantiky
* Třída je jakýkoliv zdroj, která má specifickou vlastnost (rdf:type), jejíž hodnotou je právě rdfs:Class
* Definuje pojem třídy
  + Definuje jak vyjádřit, že zdroj patří do jedné nebo několika tříd
* Oproti RDF poskytuje
  + Specifikace definičního oboru a oboru hodnot
  + Specifikace datových typů
  + Reprezentace tříd a jejich hierarchií
  + Reprezentace hierarchie vlastností
  + Tvorba komentářů
* Nelze vyjádřit
  + Ekvivalenci tříd
  + Inverzní vlastnosti
  + Funkcionalitu vlastnosti nebo její tranzitivitu
  + Omezení kardinality vlastností
  + Negaci tvrzení
  + Disjunktnost tříd, ale pouze vztah je podtřídou
  + Třídu použitím logických operátoru AND, OR, NOT na třídy

### OWL

* Jazyk pro tvorbu ontologií využitelných v prostředí sémantického webu
* Založený na RDF(S) a deskripční logice
* Typy vlastností
  + Inverzí – reprezentace dvousměrných vztahů
  + tranzitivní – pokud A je ve vztahu X k B a B je ve vztahu X k C pak A je ve vztahu X k C
  + symetrická – pokud A je ve vztahu X k B pak B je ve vztahu X k A (např. „sousedíS“)
  + reflexivní – pokud platí A je ve vztahu X k A
  + datotypová vlastnost – přiřazuje jedincům primitivní hodnoty (integer, float, string, boolean, …)
* Nové vlastnosti v OWL2
  + Ireflexivní – A není ve vztahu X k A, tj. Žádný jedinec nemá vztah sám k sobě
  + Asymetrická – Jestliže existuje vztah mezi jedincem A a jedincem B, pak nepřipouštíme vztah jedince B s jedincem A
* Vymezení třídy za účelem určení jejího významu v porovnání s jinými třídami
* Omezení se vztahují na hodnotu vlastnosti
  + someValuesFrom (some) – alespoň jedna hodnota vlastnosti musí být uvedeného typu, ale mohou existovat i další hodnoty
  + AllValuesFrom (only) – všechny hodnoty vlastnosti musí být uvedeného typu
  + hasValue
  + minCardinality, maxCardinality, cardinality…
* Nutné podmínky
  + Jestliže něco patří do třídy X, pak splňuje podmínky
* Nutné a dostačující podmínky
  + Jestliže něco… a zároveň…
* Anonymní třída
  + Bezejmenná, vymezená definováním logických výrazů
  + Třída není v hierarchii tříd explicitně vyjádřena
  + Členy jsou jedinci, kteří splňují určitou logickou definici
* Primitivní třída
* Definovaná třída
  + CWA – close world assumption – není v OWL
    - Nějaké tvrzení o světě může být pravdivé, nepravdivé nebo nerozhodnutelné
    - v uzavřeném světě nepředpokládáme, že budou zjišťovány nové skutečnosti, které by mohli změnit pohled na svět
  + OWA – open world assumption – je v OWL
    - V otevřeném světě předpokládáme, že časem bude možné přidat další informace k aktuálnímu zkoumané domény
    - Předpoklad otevřeného světa: „Jestliže tvrzení X (obžalovaný je nevinen) není pravdivé, pak tvrzení X může být jak nepravdivé (tudíž obžalovaný je vinen) tak nerozhodnutelné.“ \* „Jestliže se neprokáže nevina, pak nelze usuzovat, že obžalovaný je vinen.“

## Návrhové vzory

* Návrhový vzor je konstrukcí, která pomáhá s implementací často se opakujících netriviálních problémů

### Rozklad hodnot

* Vyjádření určité vlastnosti používané pro popisy nebo definice tříd
* Kvůli OWA musíme dále rozklad hodnot uzavřít a výslovně zadat, že výčet hodnot je úplný = axiom pokrytí

### Pseudojedinec

* Pokud chceme při modelování ontologie zůstat na úrovni tříd
* Platí
  + Pseudojedinec je třída, která nemá žádné podtřídy
  + Pseudojedinec je podtřídou té třídy, do které by měl nahrazovaný jedinec patřit
  + Do třídy pseudojedinec patří jediný, a to právě ten nahrazovaný jedinec
  + V ontologii výslovně reprezentujeme i to, že se jedná o třídu typu pseudojedinec
  + Pseudojedince použijeme ve všech výrazech, kde bychom měli použít nahrazovaného jedince
* Lze použít i v případech, kdy si nejsme jisti, zda daný jedinec již může jedincem být

### N-ární vztah

* Binární relace x N-ární relace
* Pokud je nutné reprezentovat další vlastnosti u určitého vztahu
* Primitivní třídy: Zvíře, Povaha
* Přídavné třídy: TypyPovahy
* Vztahy: Zvíře – máPovahu – Povaha

### Část – Celek

* Snižuje složitost
* Místo mnoha objektů se zabýváme jedním
* Modelování
  + Stavebnicové díly
    - Reprezentujeme objekty, mezi kterými existuje vztah
  + Vztah je částí mezi stavebnicovými díly
    - Reprezentujeme skutečnost, že stavebnicový díl se skládá z jiných stavebnicových dílů, jako tranzitivní vztah
  + Objekty s lokalizací na stavebnicových dílech
    - Reprezentujeme objekty, které mimo jiné chceme lokalizovat na stavebnicové díly
  + Vztah "je lokalizován" mezi stavebnicovým dílem (SD) a objektem s lokalizací na stavebním díle
    - Reprezentujeme skutečnost, že jeden objekt (OSLO) je lokalizován na jiném objektu (SD) jako vztah hasLocation.
  + Propojení objektů s lokalizací (OSLO) se stavebnicovými díly (SD) aneb tvorba definovaných tříd
    - Definice podtříd objektů s lokalizací
  + Klasifikace klasifikátorem

## Normalizace ontologie

* Pro udržitelnost ontologie
* Ontologii vytváříme jako stromovou strukturu nikoliv jako graf
* Na nejvyšší úrovni pouze 3 třídy

### Primitivní třída

* Nevyžaduje další definici, jednoznačný název
* Pro modelování konceptu, který v přirozeném jazyce vyjadřujeme podstatným jménem
* Modelujeme jako neúplnou třídu, tj. třídu pouze popisujeme pomocí omezení (existenčních a/nebo univerzálních) v bloku NECESSARY

### Přídavná třída

* Slouží k upřesnění jiných pojmů. Např.: červená, mladý, náročné, dlouho, levně.
* Obvykle představuje koncept, který v přirozeném jazyce vyjadřujeme přídavným jménem (odpověď na otázku Jaký?) nebo příslovcem (odpověď na otázku Jak?)
* Přídavnou třídou modelujeme nějakou dimenzi dané třídy
* Modelujeme s použitím návrhového vzoru "Rozklad třídy na podtřídy"

### Definovaná třída

* K modelování definovatelného konceptu, tj. takový koncept, k jehož vymezení použijeme jiné koncepty
* Např.: stařec (člověk, který je starý), student (člověk, který studuje), matka (člověk ženského pohlaví, který má alespoň jednoho potomka)
* Definovanou třídu modelujeme pomocí omezení (existenčních, univerzálních) v bloku NECESSARY AND SUFFICIENT, případně dále popisujeme pomocí omezení v bloku NECESSARY

## Odvozování

* Odvozovací stroj = program, který je schopen z ontologie odvodit informace/znalosti, které v ní nejsou explicitně vyjádřeny
* Pomocí nich může uživateli pomoci navrhnout a udržet bezchybné ontologie, které budou smysluplné, korektní, minimálně redundantní a bohatě axiomatizované
* Mezi standartní činnosti odvozovacího stroje patří:
* Kontrola konzistentnosti ontologie
* Klasifikace tříd
* Klasifikace jedinců
* Kontrola konzistence ontologie: brání disjunktnosti tříd
* Klasifikátor tříd: je třída A podtřídou třídy B?
  + Klasifikátor (reasoner) slouží k provádění tzv. testu subsumpce = podle podmínky uvedených u definovaných a popsaných tříd testuje, jestli jedna třída nemůže být podtřídou jiné = vytváří odvozenou (inferred) hierarchii. Zviditelňuje skryté znalosti.
  + Klasifikátor umí též ontologii zkontrolovat, jestli si některé její části navzájem logicky neodporují (neporušují pravidla disjunkce, kardinality, špatný Df či Hf u vlastností...)

## Nástroje

* JENA
  + Zdarma šířená, open-source platforma pro vývoj aplikací SW využívajících jazyk Java
  + komponenty:
    - API pro RDF, OWL
    - Čtení/zápis RDF v notaci RDF/XML, N3, N-triples
    - Íložiště pro modely RDF
    - Engine pro dotazování se pomocí SPARQL
    - Pravidlový engine
  + Vývoj nejčastěji v prostředí Eclipse
* SESAME
  + Open-source Java framework pro ukládání a dotazování se na RDF
  + Komponenty:
    - RDF model: definuje rozhraní a možnosti implementace RDF struktur
    - API: metody pro práci s RDF daty (+ odvozování s RDFS)
    - HTTP server: Java servlety implementující protokol pro přístup k repozitáři Sesamu přes HTTP
* Ontopia Navigator Framework
  + Framework založený na platformě J2EE
  + Využitelnost: vývoj webových stránek zapsaných v JSP (JavaServer Pages) s využitím ontologie vytvořené dle standardu Topic Maps
* Protégé
  + Bezplatný open source
  + Poskytuje grafické uživatelské rozhraní pro definici ontologií
  + Lze z něj exportovat do všech hlavních jazyků
  + Klasifikátor Pellet

## Aplikace ontologií

* Znalostní management ve firmách
  + Pro efektivní fungování organizace je třeba, aby se informace a znalosti (jak interního, tak externího původu) neztrácely, a včas se dostávaly k těm pracovníkům, kteří je mohou využít
  + Lze zabezpečit konzistenci znalostní, jednak usnadnit jejich vyhledání.
* Elektronické obchodování typu B-to-C i B-to-B
  + Vyhledání požadovaného produktu zákazníkem či rychlé vyhledání potenciálního partnera, ale perspektivně také automatizace procesu sjednání obchodních podmínek.
* Zpracování přirozeného jazyka
  + Terminologické ontologie mohou napomáhat např. při překladu nebo automatické sumarizaci textů.
* Inteligentní integrace informací
  + Zastřešení datových schémat distribuovaných zdrojů (strukturovaných nebo semi-strukturovaných databází, případně „tabulárních“ webových stránek) na vysoké úrovni abstrakce
* Pojmové vyhledávání informací jako vylepšení stávajících internetových vyhledávačů
* Sémantické webové portály konstruované polo-automaticky na základě metadat od poskytovatelů informace
* Agentové systémy
* Wikidata
* Google Knowledge Graph
* MusicBrainz
* Inteligentní výukové systémy
* Společné rysy aplikací:
  + Využívání strukturovaných dat – jednotný způsob vyjadřování se o "věcech", formalizace tvrzení o "věcech" pro strojové zpracování, usnadnění chápání o "věcech" pro stroje-> tj. Zejména pro webové vyhledávače
  + Kombinace dat z více zdrojů - sdílení informací a znalostí, vyjádření souhlasu s obsahem zdroje

# Sémantický web