Softwarové technologie

[Logický program 2](#_Toc164079136)

[Predikáty 2](#_Toc164079137)

[Syntaxe Prologu 3](#_Toc164079138)

[Databáze 6](#_Toc164079139)

# 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