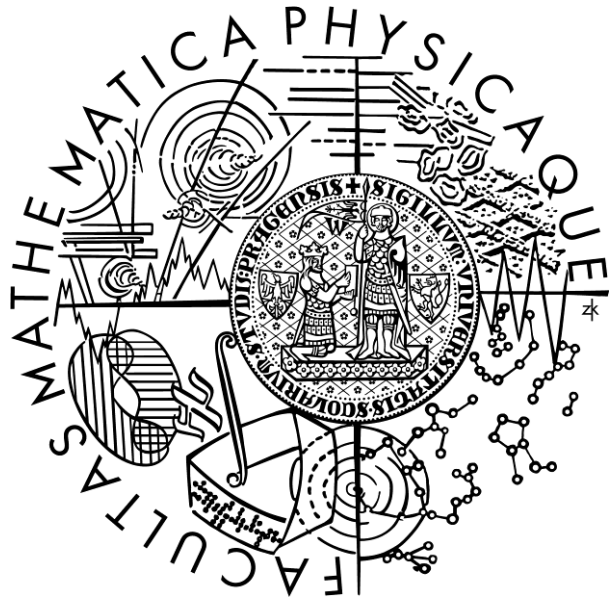
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Milan Plachý

Fuzzy databáze založená na E-R schématu

Katedra softwarového inženýrství

Vedoucí diplomové práce: prof. RNDr. Jaroslav Pokorný, CSc.

Studijní program: Informatika
Studijní obor: Softwarové systémy

Praha 2012

Rád bych poděkoval všem, kteří mne podporovali při psaní této práce. Zejména bych chtěl poděkovat svému vedoucímu prof. Jaroslavu Pokornému za poskytnutou literaturu, pomoc se strukturalizací práce a její formální stránkou.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 26.7.2012

Název práce: Fuzzy databáze založená na E-R schématu

Autor: Milan Plachý

Katedra/Ústav: Katedra softwarového inženýrství

Vedoucí diplomové práce: prof. RNDr. Jaroslav Pokorný, CSc., KSI

Abstrakt: Tento text je určen především zájemcům o oblast fuzzy logiky a její aplikaci v rámci relačních databází. Věnuje se návrhu fuzzyfikované relační databáze a samotné implementaci některé z možností. Text je rozdělen na dvě části: teoretické aspekty fuzzyfikace a samotnou implementační část. Rozšíření databáze je zvoleno na základě fuzzy E-R modelu, aby bylo možné co nejlépe specifikovat požadavky reálného světa. Dále popisuje stávající řešení v rámci různých úrovní fuzzyfikace (fuzzy množiny, FSQL). Součástí práce je i návrh a implementace jednoduchého programu pro dotazování nad fuzzyfikovanou databází. Práce by dále měla sloužit jako návod při návrhu a vytvoření konkrétního řešení.

Klíčová slova: fuzzy logika, relační databáze, fuzzy E-R, E-R schéma

Title: Fuzzy database based on an E-R schema

Author: Milan Plachý

Department: KSI

Supervisor: prof. RNDr. Jaroslav Pokorný, CSc., KSI

Abstract: This text is especialy intended to those who are interested into fuzzy logic and its application in relational databases. It is mainly focused on concept of fuzzyfied relational database and implementation of such database. This text consists of two parts: theoretical aspects of fuzzyfication and implementation part. Selected extension is based on fuzzy E-R model so the requirements of the real world can be better met. This paper also describes existing solutions on different level of fuzzyfication. Part of the work is design and implementation of a simple software for querying over fuzzyfied relational database. This work shoud also serve as a guide for design and implementation of fuzzy database.

Keywords: fuzzy logic, relational database, fuzzy E-R, E-R schema

Obsah

1. Úvod

Fuzzyfikace je oblast, která se obecně zabývá řešením nepřesnosti informací. Může se dokonce jednat o hodnoty chybějící. V rámci databází se fuzzyfikace zaměřuje na různé oblasti. Jedná se například o data uložená v databázi nebo o konkrétní dotazy do databáze. Hledání řešení tohoto problému vzhledem k databázím je proto z praktického hlediska velmi podstatné. Fuzzyfikaci si lze představit jako přiřazení míry přesnosti k údaji. Příkladem je seznam automobilů doplněný údaji o stáří, z něhož nejsme schopni stáří určit s naprostou přesností. Můžeme proto konkrétnímu automobilu přiřadit hodnotu, která udává přesnost našeho tvrzení o stáří automobilu. Příklady, jak se může fuzzyfikace projevovat jsou:

- **Nejistota**
  Automobil váží zhruba tunu (není zřejmé, zda váží méně, více nebo přesně jednu tunu).
- **Chybějící hodnota** (hodnota null, jiná hodnota, která nenese informaci).
  V tomto případě hodnota null neznamená, že hodnota neexistuje, ale že ji neznáme: Nevíme, kolik lidí bylo na festivalu.
- **Nekonzistence**
  Hodnoty si vzájemně odporují. Automobil váží tunu a zároveň dvě tuny.
- **Nepřesnost**
  Automobil neváží jednu tunu, automobil váží mezi jednou a šesti tunami, automobil váží jednu nebo dvě tuny.
- **Neurčitost**
  Automobil je starý.

Klasická dvouhodnotová logika (false/true) není schopná tyto případy dobře pokrýt, a proto je zavedena takzvaná fuzzy logika. Každému tvrzení je, na rozdíl od klasické logiky, přiřazena hodnota z intervalu [0,1].

1.1. Motivace

Protože velké množství databází je modelováno na základě ER (entity-relationship) modelu, je žádoucí tento model rozšířit o možnost používat data obsahující nepřesnost (dále jen fuzzy data). Takové rozšíření E-R modelu budeme označovat jako fuzzy E-R (fuzzy entity-relationship model). Ten se používá jako základ pro vytvoření fuzzyfikovaných relačních databází. Samotná úprava (rozšíření) databáze se označuje jako fuzzyfikace. Praktické použití nachází v dnešní době především v aplikacích, v nichž je možné hodnotit objekty. Příkladem jsou e-shopy a další databáze, ve kterých jsou uloženy informace o preferencích konkrétního uživatele formou údajů o nepřesnosti.

Fuzzyfikovat je možné na různých úrovních. Hlavní výhodou fuzzyfikace E-R modelu je možnost využití při implementaci libovolné relační databáze podle takového modelu (DB2, Oracle).

V dnešní době existují různé přístupy k fuzzyfikaci relačních databází (resp. E-R modelu). Relační databáze rozšířené pro podporu fuzzifikovaných dat budeme dále označovat jako *fuzzy relační databáze* (FR-DB). Velká část z nich však vzniká samostatně a neexistuje model, podle kterého by byla vytvořena.

1.2. Možnosti fuzzyfikace

Jednou ze základních otázek je, co vše budeme požadovat, aby bylo fuzzyfikováno. Samotný výběr záleží především na následném použití modelu. Některými z možností jsou:

- Povolení fuzzy hodnot u atributu (předpokládaná přesnost hodnoty atributu).
- Povolení neznámých hodnot u atributů (neznámé/nezadané hodnoty).
- Povolení fuzzy hodnot u entity (pravděpodobnost správnosti zařazení konkrétní instance).
- Možnost fuzzyfikace dotazu.
  Tato část umožňuje specifikovat fuzzy dotazy uživatelem, který může uvádět míru důležitosti pro jednotlivé části dotazu (především pro atributy). Na rozdíl od běžného dotazu je možné zadat další hodnoty specifikující nepřesnost zadaného údaje nebo jeho rozsah. Např.: Vyhledej všechna auta s velkým výkonem a stářím kolem pěti let.

1.3. Hodnocení výsledků

Jednou z největších výhod fuzzyfikovaných databází je, že výsledek dotazu má hodnocení, které mu přiřadila hodnotící funkce. Díky tomu můžeme výsledky dotazu seřadit sestupně od nejlepšího po nejhorší. Dále můžeme požadovat jen prvních *k* výsledků – na rozdíl od běžné relační databáze, kde jsou všechny výsledky rovnocenné a je nutné je vrátit na výstup všechny. S tímto tématem souvisí také algoritmy, které vybírají z ohodnocených atributů na základě fuzzy funkce *k* nejlepších výsledků bez nutnosti prohledávat všechny položky. Jedním z takových algoritmů je např. Top-k algoritmus. V dnešní době existuje velké množství implementací Top-k algoritmu především kvůli použití různých heuristik. Jednou z těchto implementací je software tokaf[1]. Obecně používané algoritmy pracují s posouvající se hranicí přesnosti pro různé atributy. Využití tato vlastnost nachází především v hledání nejlepších odpovídajících webových stránek, doporučení v e-shopech nebo preferenčních databázích.

1.4. Struktura práce

Následující kapitola je věnována základním informacím o fuzzyfikaci. Kapitola 2 popisuje E-R model a jeho grafickou reprezentaci E-R schéma. V kapitole 3 je definována fuzzy logika a příslušné potřebné funkce a výpočty. Kapitola 4 obsahuje popis některých přístupů k fuzzyfikaci E-R modelu, které se dají použít jako základ při vytváření vlastního fuzzyfikovaného E-R modelu. Ten je popsán v kapitole 5 včetně fuzzyfikací jednotlivých částí. Tento navržený přístup je převeden

[1] Příklad: http://www.ksi.mff.cuni.cz/sw/tokaf.html

v kapitole 7 do relační databáze, a je tak vytvořena její fuzzyfikace na základě E-R modelu. Další možné přístupy jsou zmíněny v kapitole 6.

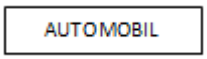
Druhá část práce se věnuje implementaci a testování jednoduchého programu pro dotazování nad fuzzy relační databází. Kapitola 8 popisuje návrh aplikace a její implementaci. V kapitole 9 jsou popsány provedené testy a interpretace jejich výsledků. Kapitola 10 shrnuje výsledky testování a celé práce.

2. E-R model a E-R schéma

Jednotlivé přístupy k fuzzyfikaci se často liší v použitém značení a terminologii. To se projevuje již na úrovni základního E-R modelu a dále pak v jeho rozšíření. Z tohoto důvodu je nutné definovat E-R model, který bude dále rozšířen.

E-R model slouží ke konceptuálnímu popisu výseku reálného světa a jeho následnému použití k vytvoření návrhu relační databáze. E-R schéma (diagram) je grafickým ztvárněním E-R modelu a slouží pro lepší představu o E-R modelu a pro jeho jednodušší vývoj. V E-R modelu budeme pro další použití rozlišovat následující komponenty:

- **Entita**
  Třída objektů shodného typu (AUTOMOBIL, UŽIVATEL). Konkrétní příklad entity nazýváme *instancí entity*. Instancí entity AUTOMOBIL je např. konkrétní vůz Škoda 120L. *Množinou instancí entity* budeme rozumět množinu všech instancí konkrétní entity. V E-R diagramu bude pro entitu použito následující značení:



Obrázek 1 - Entita

- **Vztah**
  Reprezentuje fakt, že některé entity (jejich instance) mají mezi sebou vazbu. Příkladem je vztah PATŘÍ(ČLOVĚK, AUTOMOBIL). Konkrétní příklad vztahu nazýváme *instancí vztahu*. Např.: PATŘÍ(Karel, Škoda 120L) určuje, že Karlovi patří konkrétní automobil Škoda 120L. Dále definujeme *množinu instancí vztahů* jako množinu $\{(e_1, e_2)|e_1 \in E_1, e_2 \in E_2\}$, kde E_1 a E_2 jsou entity ve vztahu R, e_1, e_2 jejich instance a existuje instance r vztahu R ve tvaru $r(e_1, e_2)$. V E-R diagramu bude pro vztah použito následující značení:

Obrázek 2 - Vztah

Pro vztah dále popisujeme kardinalitu entit zúčastněných ve vztahu. Běžně se používají kardinality (0,1), (1,1), (1,n) a (0,n), kde symbol n značí „více“. Interpretace těchto kardinalit je následující:

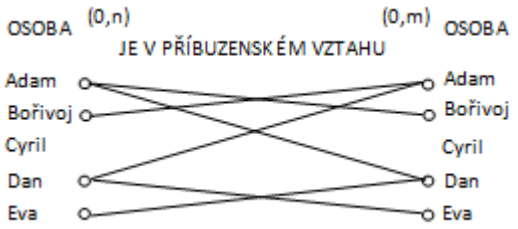
- **$R(E_1, E_2)$ s kardinalitou (0,1)**
  Pro všechny instance $e_1 \in E_1$ existuje nejvýše jedna instance r vztahu R ve tvaru $r(e_1, e_2)$, kde $e_2 \in E_2$.

- **R(E_1, E_2) s kardinalitou (1,1)**
  Pro všechny instance $e_1 \in E_1$ existuje právě jedna instance r vztahu R ve tvaru $r(e_1, e_2)$, kde $e_2 \in E_2$.
- **R(E_1, E_2) s kardinalitou (1,n)**
  Pro všechny instance $e_1 \in E_1$ existuje alespoň jedna instance r vztahu R ve tvaru $r(e_1, e_2)$, kde $e_2 \in E_2$.
- **R(E_1, E_2) s kardinalitou (0, n)**
  Pro všechny instance $e_1 \in E_1$ existuje libovolný počet instancí r vztahu R ve tvaru $r(e_1,e_2)$, kde $e_2 \in E_2$.

Názorné příklady použití:

Obrázek 3 - Kardinalita (1,n)

Jako příklad použití kardinality (1,n) slouží Obrázek 3. Zde je definováno, že každá osoba vlastní libovolný počet automobilů a automobil může být vlastněn nejvýše jednou osobou. Vztah s takovými kardinalitami se často označuje jako (1,n).



Obrázek 4 - Kardinalita (m,n)

Dále pak Obrázek 4 popisuje příbuzenské vztahy mezi osobami. Každá osoba může mít libovolný počet příbuzných. Vztah s těmito kardinalitami se často označuje jako (m,n). Tento příklad je zajímavý, protože vztah "Je v příbuzenském vztahu" není mezi dvěma různými entitami, ale vztahuje se pouze na jednu (OSOBA). Za povšimnutí stojí také „reflexivita" tohoto vztahu.

- **Atribut**
  Atribut A je parciální funkce nabývající hodnot z domény dom(A). Definičním oborem je množina instancí entity nebo množina instancí vztahu. Podle toho je možné rozlišit dva druhy atributu: atributy entity a atributy vztahu.

Obrázek 5 - Atributy

Atributy se dělí podle své komplexnosti a rozsahu:

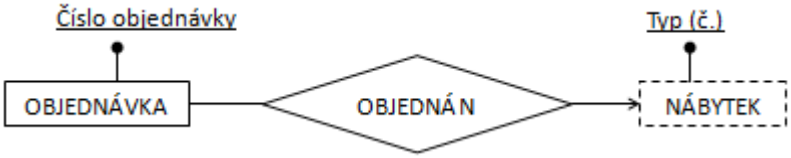
- **Jednoduché** (telefonní číslo, věk, …)
  $a \in Dom(A)$
- **Složené** (adresa, parametry motoru, …)
  $a \in Dom(A_1) \times ... \times Dom(A_n)$
- **Vícehodnotové** (seznam telefonních čísel, …)
  $a \in 2^{Dom(A)}$
- **Vícehodnotové složené** (adresy obchodů)
  $a \in 2^{Dom(A_1) \times ... \times Dom(A_n)}$

Jednoduchý atribut nebo množina jednoduchých atributů se nazývá *identifikační klíč*, pokud jeho hodnoty rozlišují jednotlivé instance entit (rodné číslo u osoby). Pokud takový klíč neexistuje, je možné vytvořit umělý.

a) jednoduchý **b) složený**

Obrázek 6 - Identifikační klíč

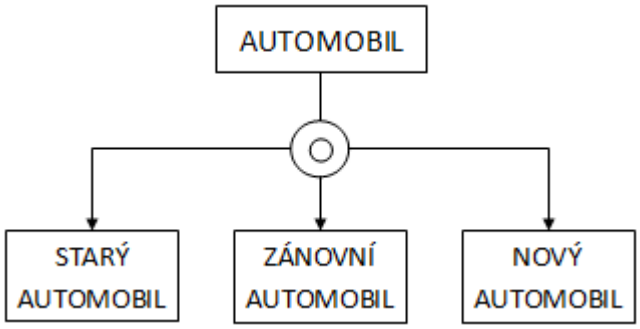
Na základě klíčových atributů se dělí entity na *slabé* a *silné*. Slabou entitou rozumíme entitu, jejíž atributy nejsou postačující pro jednoznačnou identifikaci jednotlivých instancí této entity (neexistuje identifikační klíč složen pouze z atributů této entity). Ve slabé entitě je pro jednoznačnou identifikaci nutné rozšířit vlastní klíčové atributy o další entity (identifikační vlastník) přes *identifikační vztah*. Tím vzniká jednoznačný identifikátor. Silná entita má identifikační klíč složen pouze z vlastních atributů. Obrázek 7 slouží jako názorný příklad využití slabých a silných entit. Entita NÁBYTEK je slabá, protože součástí identifikačního klíče je i číslo objednávky. Vztah OBJEDNÁN je identifikačním vztahem.



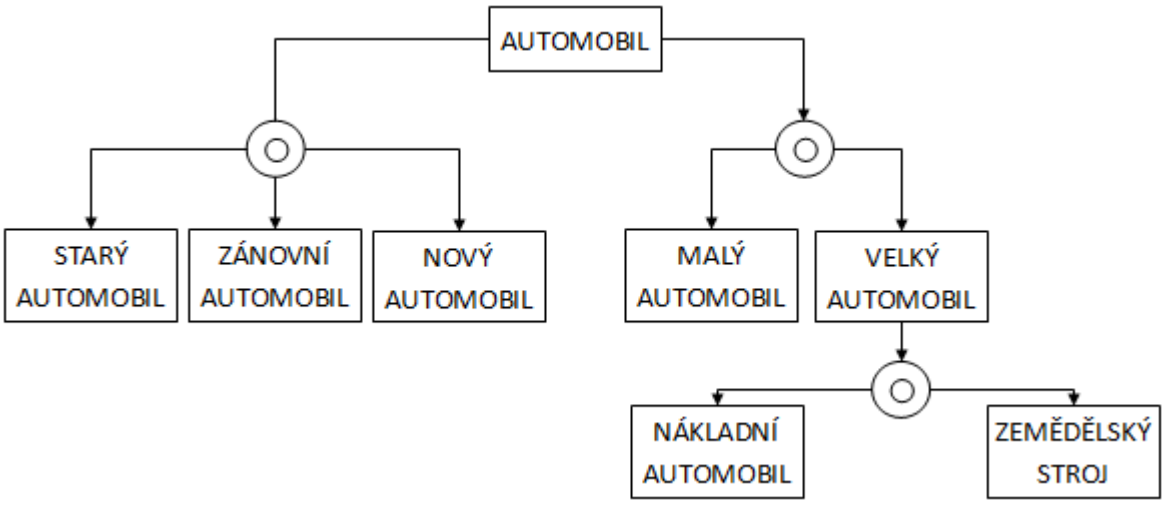
Obrázek 7 - Slabá entita

2.1. ISA-hierarchie

V E-R modelu je vhodné pro některé z entit zavést hierarchii. ISA-hierarchie přidává do modelu dědičnost podobně jako v objektově orientovaných programovacích jazycích. V rámci hierarchie dědí množina entit atributy nadřazené entity a rozšiřuje entitu, od které dědí i další atributy. Další možností je vícenásobná hierarchie, v níž vzniká více na sobě nezávislých množin odvozených od shodného entitního typu. Nadřazenou entitu budeme označovat jako *nadentitu* a entity od ní dědící jako *podentity*.



Obrázek 8 - Jednoduchá hierarchie



Obrázek 9 - Vícenásobná hierarchie

3. Fuzzy logika a fuzzy funkce

Klasický E-R model používá základní dvouhodnotovou logiku (automobil je modrý / není modrý). Z tohoto důvodu je i funkce určující pravdivost výsledku nějakého dotazu jednoduchá. Booleovská logika není schopna dobře pokrýt případy popsané v kapitole 1. Proto je použita fuzzy logika a z ní poté vycházejí odpovídající fuzzy funkce. Ve fuzzy logice může nabývat hodnota pravdivosti výroku hodnot z intervalu [0,1] (na rozdíl od {0,1} v booleovské logice).

3.1. Fuzzy funkce

Fuzzy funkce slouží k určení pravdivosti elementárního výroku. Jedná se o zobrazení F: E → [0,1], kde E je elementárním výrokem. Např. výroku "Jsem starý." přiřadí hodnotu 0,5.

3.2. Ohodnocení celého výrazu

Kvůli použití fuzzy funkce se změní i definice a použití logických spojek ve složitých výrazech. Logické spojky jsou ve fuzzy logice nahrazeny odpovídajícími funkcemi. Např. disjunkce je nahrazena funkcí: $f^*_\vee : [0,1]^2 \rightarrow [0,1]$. Nyní je potřeba nové funkce pro jednotlivé logické spojky definovat. Pokud se díváme na logické spojky booleovského modelu jako na jednoduché funkce přiřazující výrazu hodnotu 1 nebo 0, budou odpovídající funkce ve fuzzy logice jejich rozšířením na celý interval [0,1]. Je žádoucí, aby se jak funkce z booleovské logiky, tak jejich rozšíření ve fuzzy logice, chovaly stejně v krajních hodnotách, kde jsou pro oba typy funkcí hodnoty definovány. Tyto hodnoty pro základní logické spojky jsou popsány v následující tabulce, kde *x* a *y* značí pravdivostní hodnoty jednotlivých výroků a logické spojky jsou nahrazeny jednoduchými funkcemi $f_\vee$ (disjunkce), $f_\wedge$ (konjunkce), $f_\neg$ (negace).

x	y	$F_\vee(x,y)$	$F_\wedge(x,y)$	$f_\neg(x)$
1	1	1	1	0
0	1	1	0	1
1	0	1	0	0
0	0	0	0	1

Tabulka 1 - Booleovská logika

Tvrzení v rámci booleovské i fuzzy logiky zavedeme induktivně:

1) Elementární výrok je tvrzení.
2) Nechť A, B jsou tvrzení, potom i (A ∨ B), (A ∧ B), ¬A jsou tvrzení.

Funkci příslušnosti (f) budeme používat místo běžného přiřazení pravdivosti tvrzení v booleovské logice. Tato funkce je zobrazení: tvrzení (booleovský výraz) → [0,1] (kde 1 znamená formuli pravdivou a 0 nepravdivou). Z tohoto zobrazení a definice fuzzy logiky vycházejí i definice jednotlivých „podfunkcí", ze kterých se výsledná funkce f skládá.

- $f^*_\vee:[0,1]^2 \rightarrow [0,1]$ pro disjunkci

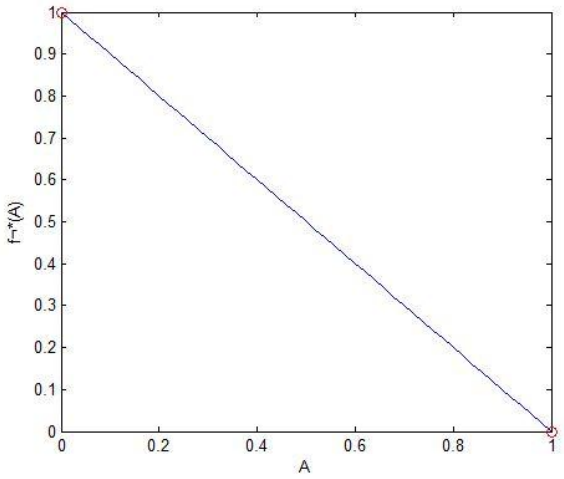
- $f^*_{\wedge}:[0,1]^2 \rightarrow [0,1]$ pro konjunkci
- $f^*_{\neg}:[0,1] \rightarrow [0,1]$ pro negaci

Celou funkci příslušnosti definujeme induktivně za pomoci předchozích funkcí.

f(A) = F(A) ... pro elementární výrok s fuzzy funkcí F
$f(A \vee B) = f^*_{\vee}(f(A),f(B))$ ... disjunkci dvou tvrzení
$f(A \wedge B) = f^*_{\wedge}(f(A),f(B))$ ... konjunkci dvou tvrzení

Funkce příslušnosti je tedy rozšířením booleovské logiky tak, aby bylo možné používat nepřesnost. Nyní dodefinujeme odpovídající funkce rozšiřující logické spojky. Výběr odpovídajících hodnot není pevně dán, nicméně existuje několik přístupů, které se chovají „rozumně". Následující část popisuje používaná rozšíření jednotlivých logických spojek.

- **Negace**
  Pro funkci reprezentující negaci se běžně používá předpis: $\mathbf{f^*_{\neg}(a) = 1 - a}$, pro $a \in [0,1]$. Graf funkce znázorňuje Obrázek 10. Zde, i v grafech popisujících další logické spojky, znázorňují červené body shodné pro fuzzy i booleovskou funkci.



Obrázek 10 - Graf negace

Přestože tato definice funkce negace je intuitivní, existují i jiné přístupy, které definují negaci například jako: $f^*_{\neg}(a) = 0$ pro a = 1, $f^*_{\neg}(a) = 1$ jinak.

Pro konjunkci a disjunkci budeme porovnávat Gödelovu a Lukasiewiczovu spojku, což jsou některé z možností, jak fuzzy funkci pro disjunkci a konjunkci definovat. Další možností rozšíření operátorů je produktová spojka, která se v rámci jednotlivých operátorů pohybuje mezi Gödelovou a Lukasiewiczovou spojkou a často popisuje situaci lépe než předchozí dvě spojky, které reprezentují krajní hodnoty.

- **Disjunkce**
  Na rozdíl od negace jsou disjunkce a konjunkce binární operátory. Tím vzniká velké množství hodnot a tedy i obrovský prostor pro různé návrhy. Pro

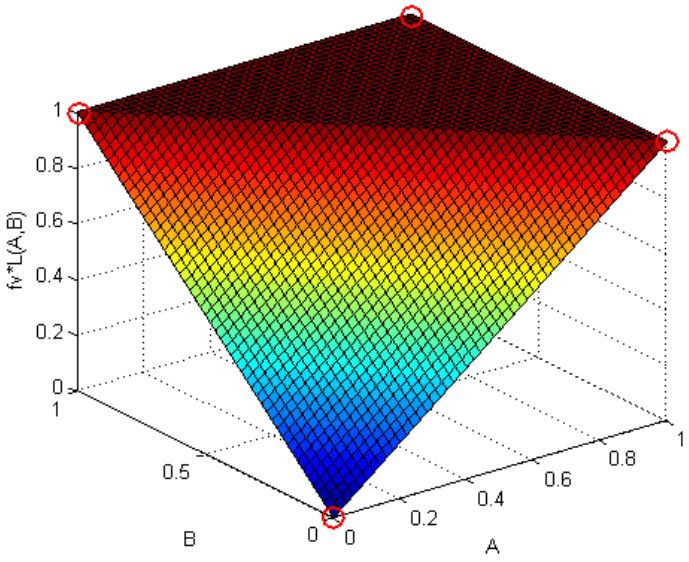
Lukasiewiczovu, Gödelovu a produktovou spojku je definice disjunkce následující:

$f^*_{\vee}L(a,b) = \min(1,a+b)$

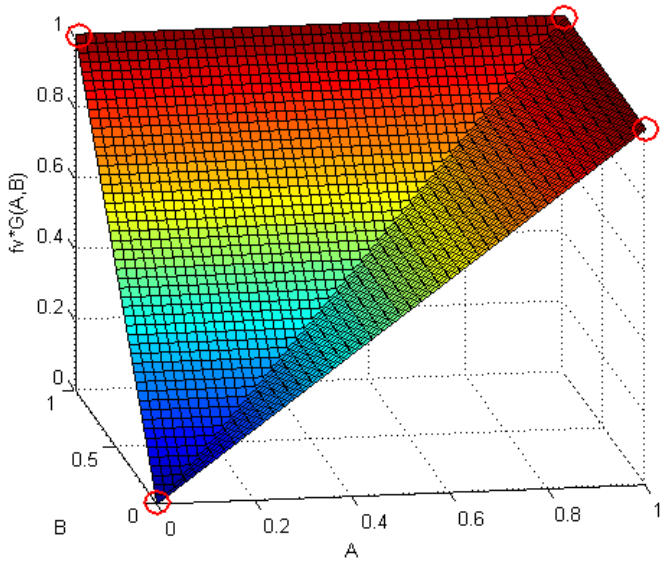
$f^*_{\vee}G(a,b) = \max(a,b)$

$f^*_{\vee}P(a,b) = (a+b)-(a*b)$

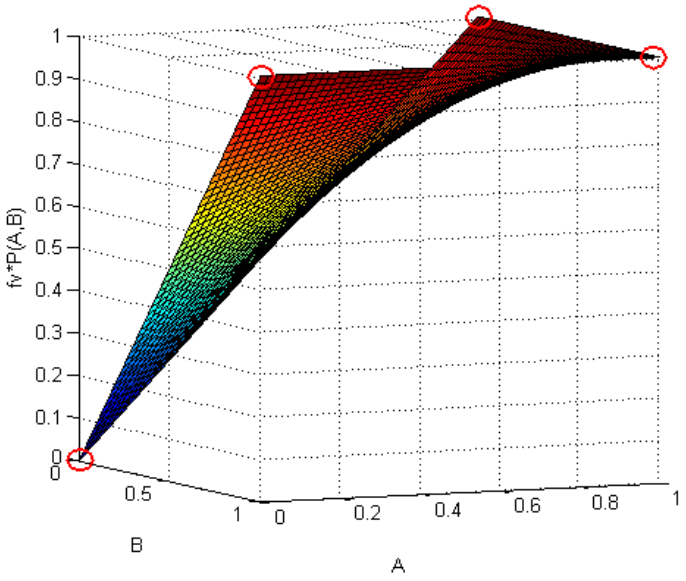
Pro lepší názornost jsou funkce a funkční hodnoty znázorněny na následujících obrázcích.



Obrázek 11 - Lukasiewiczova spojka

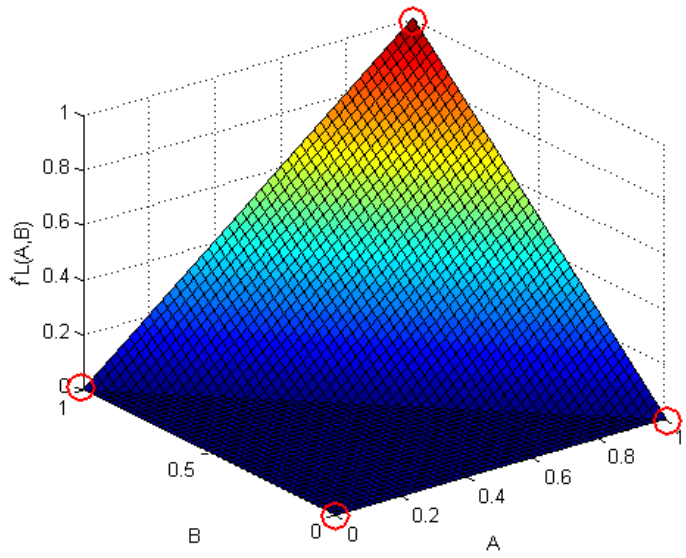


Obrázek 12 - Gödelova spojka

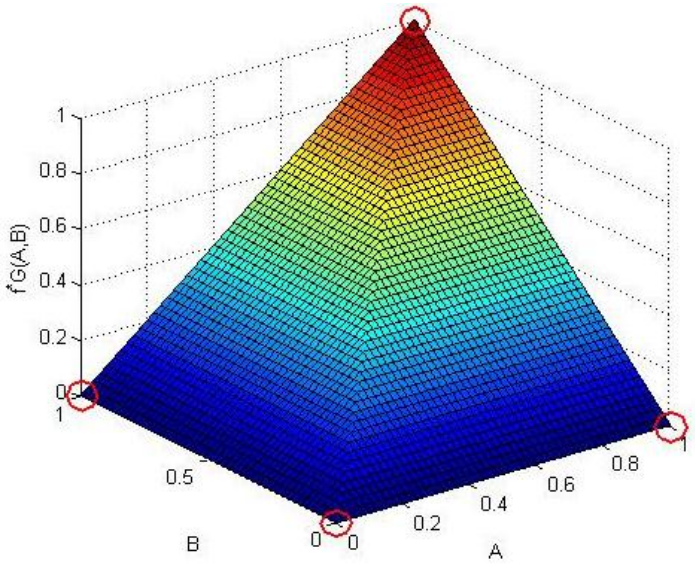


Obrázek 13 - Produktová spojka

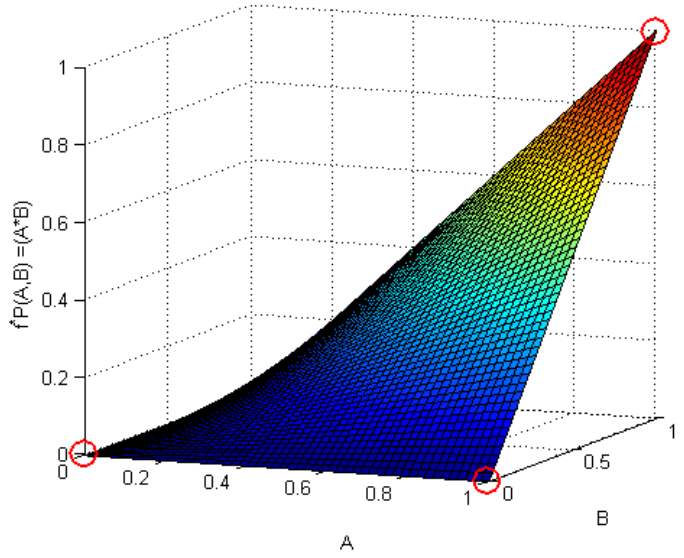
- **Konjunkce**
  Pro Lukasiewiczovu, Gödelovu a produktovou spojku je definice konjunkce následující:
  $f^*_{\wedge}L(a,b) = \max(0,a+b-1)$
  $f^*_{\wedge}G(a,b) = \min(a,b)$
  $f^*_{\wedge}P(a,b) = (a*b)$
  Pro lepší názornost jsou funkce a funkční hodnoty znázorněny na následujících obrázcích:



Obrázek 14 - Gödelova spojka



Obrázek 15 - Lukasiewiczova spojka



Obrázek 16 - Produktová spojka

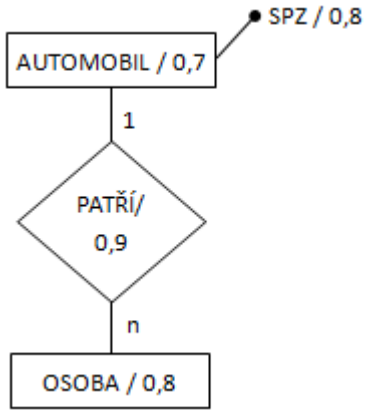
4. Existující přístupy k fuzzy E-R

V dnešní době existují různé přístupy k fuzzyfikaci ER modelů. Některé z nich jsou popsány například v [1]. Cílem této kapitoly je krátce objasnit některé z nich, případně zdůraznit některé z nedostatků a výhod. Většina prací se zabývá čistě teoretickou stránkou přístupu a nenabízí konkrétní implementace řešení.

4.1. Zvieli a Chen

Jeden z prvních přístupů k fuzzyfikaci. Zaveden a popsán v [2]. Tento přístup používá tři úrovně fuzzy informace.

- **Stupeň příslušnosti**
  Tento přístup předpokládá, že mohou být fuzzyfikovány entity, vztahy a atributy (Obrázek 17). Hodnoty za lomítkem popisují *stupeň příslušnosti* (membership degree) v rámci celého schématu. V tomto případě je to 0.8 pro SPZ automobilu (atribut), 0.7 pro automobil (element) a 0.9 pro vztah mezi automobilem a jeho vlastníkem (vztah). Po vytvoření návrhu je nutné rozhodnout, zda bude daná entita/vztah/atribut patřit do implementace databáze na základě tohoto modelu, nebo ne.



Obrázek 17 - První úroveň přístupu

- **Úroveň neurčitosti**
  Vztahuje se k výskytům fuzzy vztahů a fuzzy entit. Příkladem je entita STARÝ AUTOMOBIL, kde instance této entity budou mít jiný stupeň příslušnosti, než instance odvozené od entity AUTOMOBIL.
- **Úroveň atributu**
  Tato úroveň se zabývá fuzzyfikací speciálních atributů a vztahů. Například stav automobilu může být dobrý, špatný,... Tyto hodnoty není vždy možné vyjádřit pomocí konkrétních hodnot, může se totiž jednat o subjektivní pohled, který se mění v závislosti na uživateli. Toto je jeden z problému fuzzy hodnot, neboť ve velkém množství případů záleží na pohledu uživatele, který se může velkou měrou lišit od pohledu, jenž byl použit při zadávání dat.

4.2. Chen a Kerre

Tento přístup je podrobně popsán v [3], [4]. Zabývá se E-R modelem s ISA-hierarchií, v rámci které dělí entity do nadentit a podentit. Dále popisuje jejich vztahy a vlastnosti (generalizace, specializace, členění, sdílení). Pro zobrazení se používá značení hierarchií (Obrázek 8, Obrázek 9).

Pokud budeme brát v potaz dvě entity E_1 a E_2, kde E_2 je podentitou entity E_1, a dále funkce $E_1(e)$ a $E_2(e)$ reprezentující příslušnost, kde e je instancí E_2, potom platí, že $E_1(e) \leq E_2(e)$.

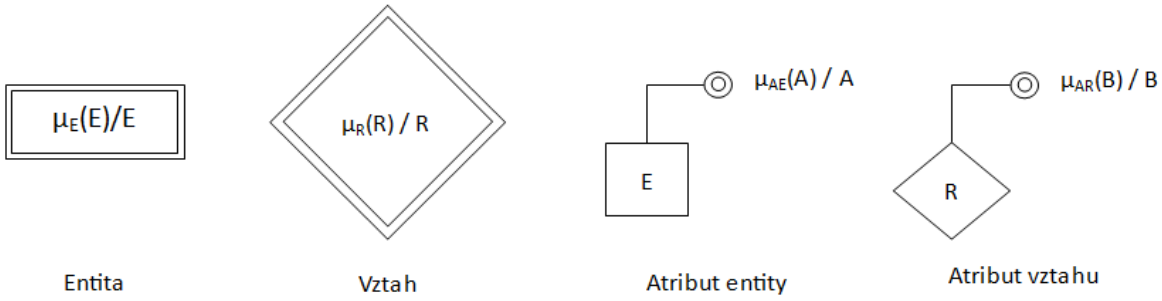
Koncept dále určuje tři základní omezení:

- Dědičnost: Podentita dědí všechny instance, ve kterých se účastnila jako nadentita.
- Přesnost: Pro každou instanci entity e entity E je definována příslušnost jako α_i a platí, že $\alpha_i > 0$.
- Kardinality vztahů: Základními kardinalitami vztahů jsou 1:1, 1:n, m:n

V modelu je použita funkce μ pro určení stupně příslušnosti entit vztahů a atributů. Atributy jsou rozděleny na typ A (atributy entity) a typ B (atributy vztahů). Pravidla a značení pro funkci μ jsou následující:

- $\mu_E(E) \in [0,1]$ pro nějakou entitu E
- $\mu_R(R) \in [0,1]$ pro vztah R mezi dvěma entitami
- $\mu_{AE}(A) \in [0,1]$ kde A je atribut entity
- $\mu_{AR}(B) \in [0,1]$ kde B je atribut vztahu

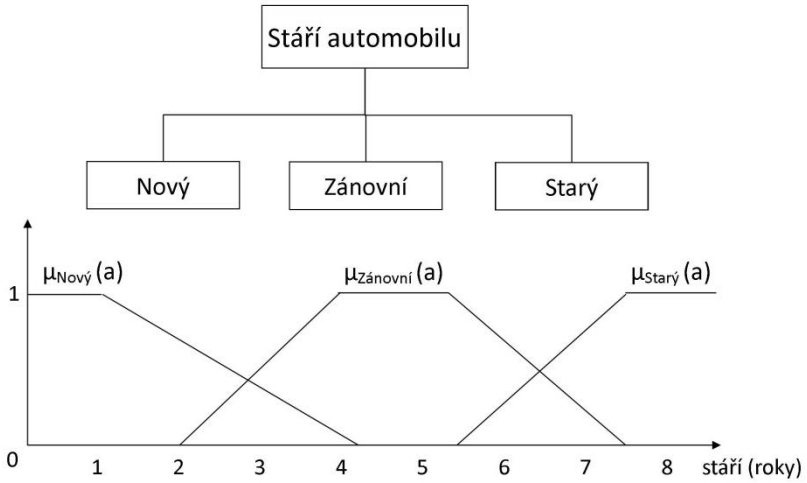
Pro názornou představu slouží Obrázek 18. Zdvojení rámečku u entity/vztahu/atributu značí, že se jedná o fuzzyfikované rozšíření.



Obrázek 18 - Značení typů

V tomto modelu se nahlíží na problematiku i z hlediska průniků a sjednocení podentit. Můžeme například považovat za nové automobily i některé ze zánovních, můžeme chtít získat informace o nových a/nebo starých automobilech atd. Tento pohled se týká nejen atributů, ale i entit. Příkladem je Obrázek 8, kde jsou rozlišeny odvozené entity atributem stáří (starý, nový, zánovní). Tyto atributy vytváří označení podentit a lichoběžníkovou funkci pro konkrétní entitu nebo atribut.

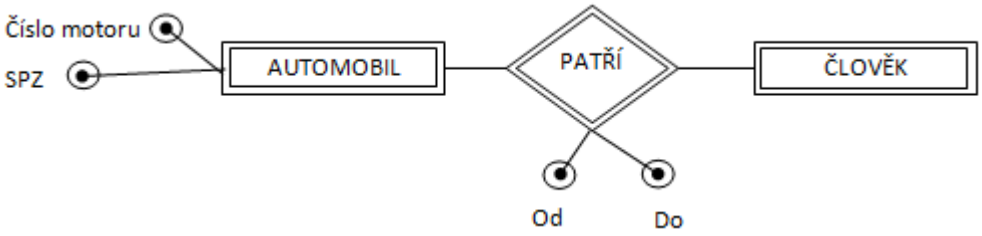
Jako příklad je upraven Obrázek 19, ke kterému je přidán graf. Rozhodujícím atributem je v tomto případě stáří automobilu. Obecně však nemusí rozdělení podentit být dáno atributem.



Obrázek 19 - Lichoběžníkový graf

5. Návrh fuzzy E-R

Tato kapitola se věnuje vlastnímu návrhu fuzzyfikovaného E-R modelu, který vychází ze zavedené fuzzy logiky a existujících přístupů. V kapitole 7 je tento návrh použit jako základ pro vytvoření fuzzyfikované relační databáze. Fuzzy E-R zachovává u nefuzzyfikovaných entit, vztahů a atributů značení booleovského modelu. Pro vyznačení fuzzyfikovaných komponent v diagramu použijeme zdvojení okrajové linky. Další možností je navrhnout model tak, aby byla fuzzyfikována veškerá data. To je umožněno především díky rozumnému chování fuzzy funkce a ohodnocující funkce, které se v okrajových hodnotách chovají shodně jako běžné vyhodnocování v booleovském modelu. Tento přístup ale není vhodný při tvoření relační databáze, kde poté vzniká velké množství pomocných atributů, čímž jsou zvýšeny nároky na dotazování.



Obrázek 20 - Fuzzyfikace

V tomto příkladu jsou fuzzyfikovány veškeré atributy, vztahy a entity.

Jak je již naznačeno, fuzzyfikace bude probíhat na třech úrovních. Jsou to: fuzzyfikace entity (sekce 5.1), fuzzyfikace vztahu (sekce 5.2) a fuzzyfikace atributu (sekce 5.3). Speciální částí je ISA-hierarchie, která částečně spadá pod fuzzyfikaci entity a její transformace je popsána v sekci 2.1.

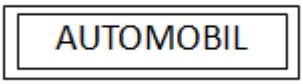
5.1. Fuzzifikace entity

Mějme entitu E, potom *fuzzyfikovanou entitu* E* odvozenou od E definujeme jako dvojici $E^* = (E,\mu_E)$, kde μ_E: $E \rightarrow [0,1]$ je funkce, která každé instanci entity $e \in E$ přiřadí stupeň příslušnosti. Stupeň příslušnosti je možné představit si jako pravděpodobnost, že daná instance je legitimním reprezentantem entity. Příkladem je entita STARÝ AUTOMOBIL, kde stupeň příslušnosti je pravděpodobnost, že se jedná skutečně o starý automobil. Fuzzyfikace entity má dále význam především v rámci ISA-hierarchie. Krajní hodnoty stupně příslušnosti odpovídají booleovskému modelu:

- $\mu_E(e) = 0$ (pro nějakou instanci entity $e \in E$)
  V tomto případě instance entity e není reprezentantem entity E. Například úroveň entity u jízdního kola je 0 vůči entitě AUTOMOBIL. V případě booleovského modelu taková instance entity neexistuje.
- $\mu_E(e) = 1$ (pro nějakou instanci entity $e \in E$)
  V tomto případě je e s jistotou reprezentantem entity. Například automobil Škoda

120L má stupeň příslušnosti 1 vůči entitě AUTOMOBIL. V případě booleovského modelu je e instancí entity E.

Je nutné zvážit zavedení podmínky $\mu_E(e) > 0$ a to především z důvodu menšího množství dat následně uložených v databázi a lepší reprezentaci booleovského modelu. Tuto podmínku bude požadovat i navrhovaný model. Je dále možné vyžadovat i striktnější opatření, jako například $\mu_E(e) > 0{,}5$. Ty jsou však příliš omezující a tento model je nebude splňovat.



Obrázek 21 - Fuzzyfikovaná entita

5.2. Fuzzyfikace vztahu

Mějme vztah R, potom *fuzzyfikovaný vztah* R* definujeme jako dvojici $R^* = (R,\mu_R)$, kde $\mu_R: R \rightarrow [0,1]$ je funkce, která přiřadí každé instanci vztahu míru přesnosti vztahu. Příkladem je vztah VLASTNÍ(OSOBA,AUTOMOBIL), kde míra přesnosti udává jistotu, s jakou jsme schopni potvrdit, že konkrétní instance splňuje tento vztah. Podobně jako u stupně příslušnosti jsou zajímavé krajní hodnoty této funkce.

- **$\mu_R(r) = 0$** (pro nějakou instanci vztahu $r \in R$)
  V tomto případě vztah reprezentovaný instancí vztahu r není platný. Například mějme vztah VLASTNÍ(OSOBA,AUTOMOBIL) a následně osobu Karla, který nevlastní žádný automobil. Potom instance vztahu VLASTNÍ(Karel,Škoda 120L) má míru přesnosti rovnu nule.
- **$\mu_R(r) = 1$** (pro nějakou instanci vztahu $r \in R$)
  V tomto případě naopak můžeme s jistotou říct, že daná instance vztahu je pravdivá. Například pokud Karel vlastní automobil Škoda 120L, tak míra přesnosti instance vztahu VLASTNÍ(Karel,Škoda 120L) bude rovna jedné.

Opět je nutné zvážit zavedení podmínky $\mu_R(r) > 0$, která snižuje zátěž výsledné databáze, ale zároveň snižuje funkčnost celého modelu. Navrhovaný model bude požadovat tuto podmínku. Opět je možné požadovat striktnější verze, a snižovat tak zátěž databáze na úkor funkčnosti modelu.

Obrázek 22 - Fuzzyfikovaný vztah

5.3. Fuzzyfikace atributu

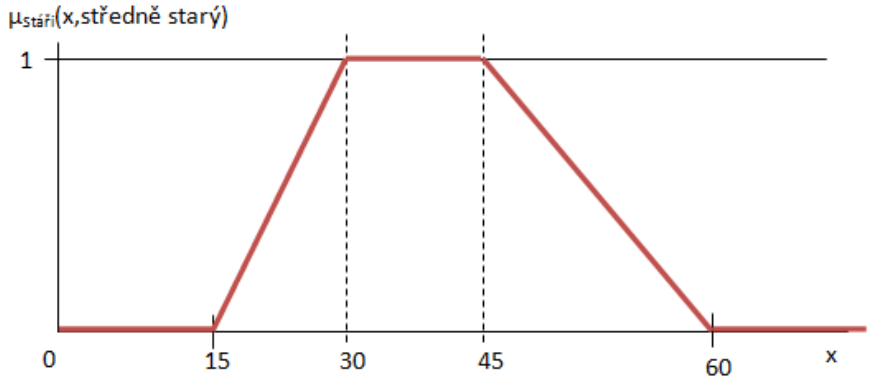
V rámci fuzzyfikace atributu využijeme fuzzy funkci. Tu rozšíříme tak, aby byla aplikovatelná na celou doménu atributu. Mějme tedy atribut A, potom fuzzyfikovaným atributem A* je dvojice A* = (A, μ_A), kde μ_A: Dom(A) × F-Dom(A) (fuzzy doména atributu A) → [0,1] je fuzzy funkce určují pravdivost tvrzení specifikovaného fuzzy hodnotou aplikovaného na hodnotu atributu. Příklady fuzzy hodnot jsou: starý, mladý, zhruba 60, o hodně starší než 20. Dále je nutné rozšířit doménu atributu o fuzzy hodnoty z F-Dom(A). Pro názornější představu slouží následující příklad.

Mějme element Automobil a jeho atribut stáří udávané v letech.

- $\mu_{Stáří}$(10,starý) = 0,9
- $\mu_{Stáří}$(1,starý) = 0
- $\mu_{Stáří}$(zánovní,starý) = 0,4

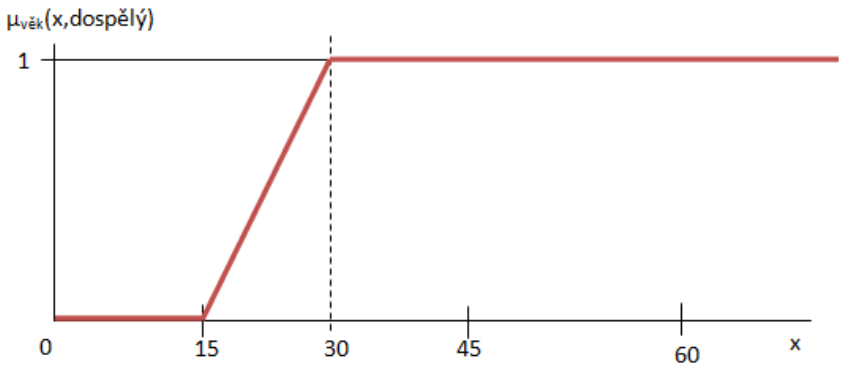
V tomto modelu budeme uvažovat několik typů fuzzy funkcí definovaných jednoduchým předpisem. Jednoduchost funkcí je klíčová z hlediska výpočetní složitosti. Proto je vhodné, aby byla funkce po částech lineární a aby bylo možné ji zapsat pouze pomocí zlomových bodů. Použité funkce jsou:

- **Rozsah**
  Funkce pro rozsah jsou definovány čtyřmi zlomovými body. Jedná se o rozsahy, které pokrývají pouze část domény atributu a jinak nabývají hodnoty 0. Příkladem je funkce popisující, že automobil je středně starý.



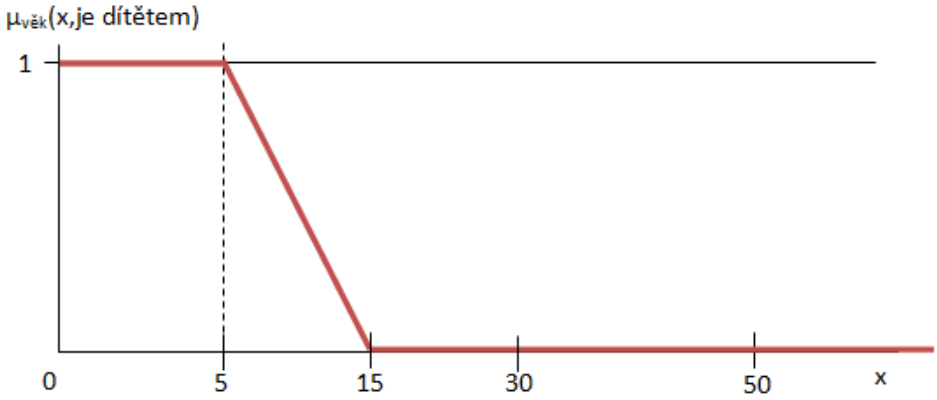
Obrázek 23 - Rozsah

- **Rozsah$^+$**
  Funkcemi pro kladný rozsah budeme rozumět funkce, které ve svém průběhu přechází z hodnoty 0 na hodnotu 1, kde již setrvávají. Zlomové body jsou pouze dva. Příkladem je funkce popisující dospělost jedince.



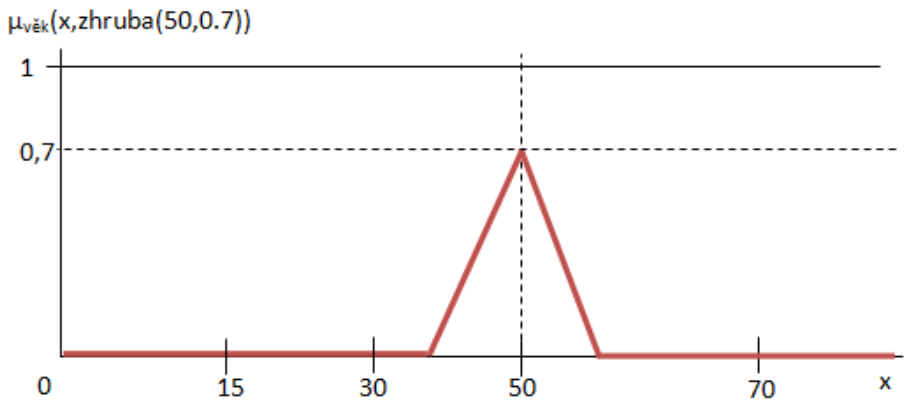
Obrázek 24 - Rozsah^{+}

- **Rozsah^{-}**
  Funkce pro záporný rozsah je oproti funkci pro kladný rozsah zrcadlově převrácená. Příkladem je funkce popisující, že jedinec je dítětem.



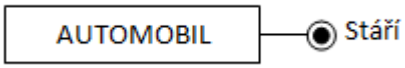
Obrázek 25 - Rozsah^{-}

- **Přibližnost**
  Funkce přibližnosti je pouze speciálním případem rozsahové funkce, kde se dva střední zlomové body spojují do jednoho. Do této funkce je dále vhodné přidat parametr informující o přesnosti středního bodu a jeho poloze. Například funkce popisující, že jedinec je starý zhruba x let s hodnotou 50 (je starý zhruba 50 let) a přesností 0,7, může vypadat takto:



Obrázek 26 – Přibližnost

Tyto skupiny funkcí byly zvoleny proto, že jimi lze popsat velká část reálných případů. Je možné je dále rozšířit o další zástupce nebo přidat k již použitým funkcím nové parametry. V této oblasti proti sobě stojí míra přiblížení se reálnému světu nebo konkrétnímu problému s výkonností a použitelností systému vytvořeném na základě modelu.



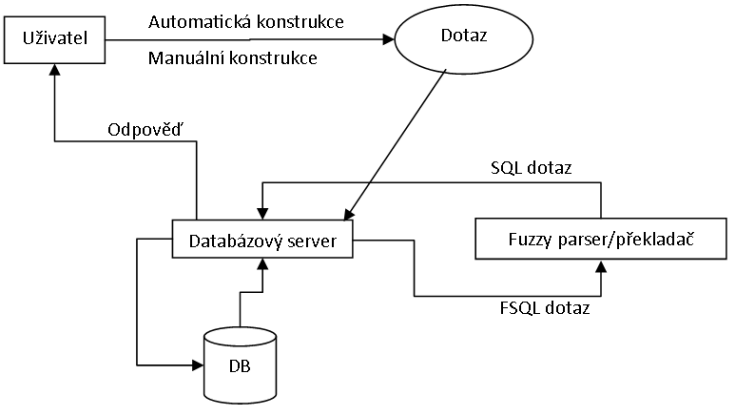
Obrázek 27 - Fuzzy atribut Stáří

V případě, že jsou fuzzyfikovány všechny atributy, které reprezentují primární klíče, jejich část nebo primární klíč neexistuje. Je nutné doplnit umělý primární klíč (Id záznamu).

6. FR-DB a dotazovací jazyky

Jak je již popsáno v předchozích kapitolách, E-R model slouží především k návrhu databáze a jako celek má význam i z hlediska vizualizace problému. Tato kapitola se zabývá FR-DB a jazyky pro dotazování nad nimi. Dále pak konkrétními příklady existujících databází a jazyků.

V případě, že je již fuzzyfikovaná databáze naimplementována, předpokládá se existence dotazovacího jazyka, který se bude používat. V ideálním případě bez toho, aby musel uživatel (případně administrátor) mít nějaké rozšířené znalosti z oblasti fuzzy databází. Jednou z možností je vytvořit aplikační rozhraní nad celou databází (například webové rozhraní), které požadavky uživatele převede na dotazy v běžně používaném jazyku (SQL,...). Toto řešení je však často těžkopádné z důvodu velkého množství výpočtů a logiky, která musí být vytvořena v aplikační vrstvě. Další možností je vytvořit nadstavbu použitého dotazovacího jazyka a rozšířit tím jeho funkčnost. V tomto případě musí databázový server obsahovat část, pomocí které bude převádět dotazy z nadstavbového jazyka do klasického (Obrázek 28).



Obrázek 28 – Postup při dotazování do FR-DB

Základními požadavky pro takový jazyk jsou:

- Kontrola počtu odpovědí
- Možnost zadávat nebooleovské části dotazů
- Aplikovatelnost na stávající řešení

Hlavní otázkou je, jak reprezentovat nepřesné dotazy v databázi s fakty. Pokud bychom použili pouze přidaná fakta (VYSOKÝ odpovídá výška > 180), tak dochází k chybám v interpretaci a není možné regulovat velikost výstupu. V této kapitole jsou popsány různé přístupy k samotné implementaci FR-DB a dotazování nad ní.

6.1. ARES

ARES (popsán například v [5]) je jedním z pokusů, jak rozšířit SŘBD, aby byly podporovány některé fuzzy operátory. Je zajímavý především proto, že na rozdíl od

většiny ostatních přístupů není založen na teorii fuzzy množin. Používá operátor ≈, který reprezentuje přibližnou rovnost při operaci spojení nebo ho lze použít pro selekci v podmínkách. Jeho interpretace záleží na úrovni, kde se vyskytuje. Např. na úrovni podobnosti 0 bude podmínka *značka ≈ 'škoda'* znamenat: *značka = 'škoda'*. Na úrovni podobnosti 1 potom jako *značka ϵ {škoda, audi, mercedes,…}*. Možnosti regulace velikosti výstupu jsou dvě:

1. **Kvantitativní**
   Je vytvořena globální úroveň podobnosti záznamu jako součet podobností vzhledem ke všem podmínkám. Záznamy jsou poté setříděny vzestupně podle globální podobnosti a uživateli je vráceno prvních *n*, kde *n* je číslo zadané uživatelem.
2. **Kvalitativní**
   Možnost nastavení dolní hranice podobnosti pro operátor ≈. Záznamy, které při vyhodnocení libovolné podmínky nepřekračují zadanou hranici, nejsou zahrnuty v odpovědích.

Systém ARES pracuje pouze s konjunkcí podmínek a lze ho tedy používat jen velmi omezeně. Slouží jako modelový příklad, jak lze řešit například omezení velikosti odpovědi na zadaný dotaz.

6.2. Rozšíření na základě fuzzy množin

Hlavním směrem v oblasti rozšíření SŘBD je použití teorie fuzzy množin (rozšiřující klasické množiny). Relaci je možné si představit jako množinu, fuzzy relaci jako fuzzy množinu. Fuzzy relaci R_f si můžeme tedy představit jako podmnožinu kartézského součinu $D_1 \times ... \times D_n$ kde $D_1,...,D_n$ reprezentují domény jednotlivých atributů, a pro všechna i: $D_i \in [0,1]$. Tento atribut je označován jako členství (membership) a popisuje úroveň přesnosti jednotlivých instancí entit vůči entitě. Pokud by ve všech instancích byla tato hodnota 1, jednalo by se v tomto ohledu o normální relační databázi. Cílem tohoto rozšíření je pomocí transformací dotazu a dat získat odpovídající výsledek. Postup transformací je následující:

1. **Vstupní relace** (VR)
2. **Transformované relace** (TR)
   Přetransformované vstupní relace na základě fuzzy podmínek. Členství v TR představuje míru splnění fuzzy podmínky.
3. **Finální fuzzy relace** (FFR)
   Kombinace TR do jedné relace pomocí množinových operací.
4. **Běžná relace** (BR)
   Výstupní relace vytvořená z FFR pro reprezentaci výsledku.

Fuzzy podmínka je reprezentována stromem vytvořeným ze čtyř typů termů: Transformace, Modifikace, Interakce a Booleovské výrazy. Booleovské výrazy vychází z klasického relačního modelu, ostatní typy termů jsou popsány dále.

Transformace

Transformace slouží pro zobrazení podmnožiny kartézského součinu domén atributů do intervalu [0,1]. Pro značení se používá symbol μ. Jejich úkolem je popisovat základní podmínky. Dále se používají k porovnávání objektů.

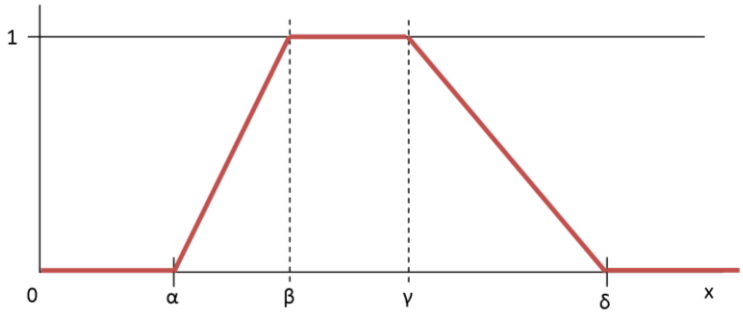
starý: Věk → [0,1], v →$\mu_{starý}$(v)

daleko mladší než: Věk × Věk → [0,1], (v_1,v_2) → $\mu_{daleko\ mladší\ než}$($h_1$,$h_2$)

Pro tato zobrazení lze definovat různé funkce, tak aby nejlépe popisovaly situaci a co nejvíce se přibližovaly reálnému chápání světa. Příkladem je například často používaná funkce S(x,α,β,γ,δ) a $\alpha < \beta < \gamma < \delta$. Tato funkce se označuje jako *trapezium* a její funkční hodnoty jsou:

0 pro $x \leq \alpha$ nebo $x \geq \delta$
$(x - \alpha)/(\beta - \alpha)$ pro $\alpha \leq x \leq \beta$
1 pro $\beta \leq x \leq \gamma$
$(x - \delta)/(\gamma - \delta)$ pro $\gamma \leq x \leq \delta$

Tuto funkci znázorňuje Obrázek 29.



Obrázek 29 - Trapezium

Modifikace

Modifikace se aplikuje na fuzzy množiny. Jedná se o zobrazení [0,1] → [0,1]. Pomocí modifikací jsou reprezentovány další hodnoty jako *velmi*, *kolem*, *negace*,... Klasická reprezentace pro *velmi* a *negaci* je následující:

velmi fuzzy podmínka(x) = $(\mu_{fuzzy\ podmínka}(x))^2$
not fuzzy podmínka(x) = $1 - \mu_{fuzzy\ podmínka}(x)$

Interakce

Interakce slouží k vyjádření spojení mezi fuzzy množinami. Příkladem jsou spojky AND a OR, které jsou reprezentovány jednou z možností vyjádření průniku a sjednocení fuzzy množin. Například:

$(f_{pa}$ AND $f_{pb})(x) = \min(\mu_{fpa}(x),\mu_{fpb}(x))$
$(f_{pa}$ OR $f_{pb})(x) = \max(\mu_{fpa}(x),\mu_{fpb}(x))$

Rozšíření jazyka SQL
Jazyk SQL musí být rozšířen, aby podporoval možnosti specifikované v předchozích sekcích. Klasická syntaxe jazyka SQL vypadá ve zjednodušené formě následovně:

`SELECT` <atributy> `FROM` <relace> `WHERE` <podmínka>

Základní funkce, u kterých požadujeme, aby rozšíření umožňovalo, jsou:

- Regulace velikosti výstupu (podobně jako v systému ARES kvalitativně/kvantitativně).
  Značení: n ∈ **N**...omezení počtu výstupů, t ∈ **R**...nastavení prahu pro jednotlivé účasti.
- Použití fuzzy podmínek.

`SELECT` n/t <atributy> `FROM` <relace> `WHERE` <fuzzy podmínka (kombinace s booleovskou podmínkou)>

Problémem jsou duplikované záznamy s různou hodnotou členství. Řešením je klíčové slovo `UNIQUE` ve třech možných variantách.

1. unique – výsledkem je pouze relace s nejvyšší hodnotou členství
2. unimin – výsledkem je pouze relace s nejnižší hodnotou členství
3. uniavg – výsledkem je jedna relace, jejíž hodnota členství odpovídá průměru všech validních relací

Množinové operace `UNION`, `INTERSECT` a `DIFFER` jsou rozšířeny na FFR, kde definujeme

$\mu_{AUB}(x) = \max(\mu_A(x),\mu_B(x))$
$\mu_{A\cap B}(x) = \min(\mu_A(x),\mu_B(x))$
$\mu_{A-B}(x) = \mu_{A\cap B'}(x) = \min(\mu_A(x),1 - \mu_B(x))$

Následující příklad ilustruje syntaxe:
Entity: AUTOMOBIL(Vyrobni_cislo,spolehlivost),
PRODEJ(Vyrobni_cislo,prodano_za)

Dotaz: Najdi 5 automobilů s vysokou spolehlivostí, takové, že byly prodány přibližně za 100 000 Kč.

`SELECT` 5 vyrobni_cislo `FROM` automobil `WHERE` spolehlive(spolehlivost)
`INTER`
`SELECT` vyrobni_cislo `FROM` prodej `WHERE` priblizne(prodano_za, 100000)

Dělení a porovnávání množin `GROUP BY`, `HAVING`

Škálování do podmnožin lze provádět na základě dvou možností:

1. Přes agregace (HAVING SUM > ...)
   SELECT n/t <atributy1'> FROM <relace> WHERE <booleovsská podmínka> GROUP BY <atributy1> HAVING <fuzzy podmínka + agregační pravidlo>
   atributy1' je podmnožinou množiny atributy1

   SELECT 10 vyrobni_cislo FROM automobil WHERE typ = 'osobní' GROUP BY vyrobce HAVING cca(AVG(rychlost),180)

2. Porovnání s jinou množinou
   Je potřeba zavést míru, která bude určovat podobnost dvou množin A, B definovaných v univerzu U.

$$d_{eq} = \frac{\sum_{x \in U} min(\mu_A(x),\ \mu_B(x))}{\sum_{x \in U} max(\mu_A(x),\ \mu_B(x))}$$

   SELECT <atributy1'> FROM <relace1> WHERE <podmínka1>
   GROUP BY <atributy1> HAVING SET(<atributy2>) IS/CONTAINS SELECT <atributy3> FROM <relace2> WHERE <podmínka2>

   Duplicitu lze opět řešit za použití unique, tím ale přicházíme o možné výsledky v případě, že některé ze záznamů mají v atributech2 nebo atributech3 shodné hodnoty. Proto je možné použít klíčové slovo WITH.

   SELECT n/t <atributy1'> FROM <relace1> GROUP BY <atributy1>
   HAVING SET (<atributy2> WITH <fuzzy podmínka + agregační pravidlo>) IS/CONTAINS SELECT <atributy3> FROM <relace2> WITH <fuzzy podmínka + agregační pravidlo>

6.3. Model GEFRED a FSQL

Předchozí sekce byla zaměřena na logickou stránku databáze, na vyhodnocování záznamů a jejich interpretaci. V této sekci bude popsán model GEFRED, který je rozšířením klasické relační databáze a jazyk FSQL hlavně z hlediska reprezentace dat (atributů, entit, …) tak, aby bylo možné použít fuzzy dotazovací jazyk (v tomto případě FSQL).

6.3.1. Model GEFRED (GEneralized model Fuzzy heart RElational Database)

GEFRED [6] se věnuje především reprezentaci atributů. Rozlišuje dvě třídy atributů. Atributy, jejichž hodnoty jsou fuzzy množiny, a atributy reprezentující přesnost, kde doména odpovídá intervalu [0,1] a jejich interpretace se může lišit aplikaci od aplikace (míra nepřesnosti, míra důležitosti). Atributy reprezentované jako fuzzy množiny se dále dělí na:

- **Klasické nefuzzyfikované atributy**
  Obsahují pouze přesná data a nepovolují nepřesnost. Od běžných atributů se liší pouze tak, že jsou označené pomocí štítku (labelu). Díky tomu lze nad nimi provádět dotazy zahrnující velký počet atributů. Jejich reprezentace v rámci entity je shodná jako v klasické relační databázi.
- **Klasické i fuzzy atributy**
  Jedná se o rozšíření předchozího typu. Reprezentují hodnoty jako: „je zhruba 80 Kg těžký“. K jejich reprezentaci slouží 5 přídavných atributů, kde atribut FT udává typy hodnot, jakých může atribut nabývat, a parametry F1–F4 udávají parametry pro funkci definovanou pro transformace (Tabulka 2).

Hodnota (typ)	FT	F1	F2	F3	F4
UNKNOWN	0	NULL	NULL	NULL	NULL
NULL	1	NULL	NULL	NULL	NULL
LABEL	2	FUZZY_ID	NULL	NULL	NULL
INTERVAL	3	n	NULL	NULL	m
TRAPEZE (α,β,γ,δ)	4	α	β	γ	δ

Tabulka 2 - Příklad hodnot přídavných atributů v systému GEFRED

- **Atributy obsahující diskrétní nesetříditelná data**
  Příkladem je atribut, jehož doménou jsou barvy. V rámci této domény, jejíž prvky jsou skaláry, je zavedena funkce popisující podobnost hodnot atributu (hnědá je blíže červené než bílé atd.). Reprezentace je podobná jako v předchozím případě a skládá se z atributu FT, který udává typy hodnot. Dále pak z atributů $(FP_1, F_1),..,(FP_n,F_n)$ (kde n je velikost domény), které popisují pravděpodobnostní rozdělení na doméně (Tabulka 3).

Hodnota	FT	FP1	F1	...	FPN	FN
NULL	0	NULL	NULL	...	NULL	NULL
UNKNOWN	1	NULL	NULL	...	NULL	NULL
SIMPLE	2	p	d	...	NULL	NULL
PRAVD. ROZDĚLENÍ	3	p1	d1	...	pn	dn

Tabulka 3 - Část hodnot přídavných atributů v systému GEFRED

- **Atributy obsahující nesetříditelná diskrétní data (bez podobnosti)**
  Tyto atributy jsou shodné jako předchozí, ale není pro ně definována funkce podobnosti. Jejich reprezentace je s předchozím typem shodná.

6.3.2. FSQL

Popis databáze, nad kterou se FSQL používá, není veřejně dostupný a existují pouze popisy použitého dotazovacího jazyka. Ty se často liší, protože vznikají různé implementace a různé verze tohoto jazyka[2].

[2] Příklad: http://www.lcc.uma.es/~ppgg/FSQL.html

Předpokládá se, že atributy jsou označeny a dá se poznat, zda se jedná o fuzzyfikovaný atribut (viz předchozí sekce). Štítky označené atributy jsou v jazyce označeny pomocí symbolu $ ($velikost). Jazyk dále obsahuje fuzzy komparátory (a jejich negace):

- FEQ (NFEQ) – fuzzy equal
- FGT (NFGT) – fuzzy greater then
- FLT (NFLT) – fuzzy less then
- FGEQ (NFGEQ) – fuzzy greater or equal
- FLEQ (NFLEQ) – fuzzy less or equal
- MGT (NMGT) – much greater then
- MLT (NMLT) – much less then

Pomocí těchto komparátorů lze porovnávat hodnotu fuzzy atributu s konstantou nebo dvou fuzzy atributů mezi sebou. Dále jazyk obsahuje specifikaci prahové hodnoty (dále prahu): THOLD X. Podmínka musí být vyhodnocena s hodnotou vyšší, než má THOLD, který jí omezuje. Pokud není THOLD v podmínce použit, je jeho implicitní hodnota 1. Pro specifikaci chybějících a neznámých hodnot jazyk obsahuje konstanty:

- UNKNOWN – atribut je možné použít, ale má neznámou hodnotu
- UNDEFINED – atribut nemá význam
- NULL – nevíme nic o hodnotě atributu

Pro příklad dotazu můžeme chtít získat z databáze všechna auta, která jsou menší než velká auta a mají výkon o hodně větší než 1500 koní. Prahy pro tyto podmínky jsou 0,8 pro velikost a 0,5 pro výkon.

```
SELECT * FROM Auta
        WHERE Velikost FLT $velka_auta THOLD 0,8 AND
                Vykon MGT 1500 THOLD 0,5
```

7. Návrh FR-DB

Cílem této kapitoly je navrhnout rozšíření relační databáze na základě modelu popsaného v kapitole 5. Tento přístup se od většiny ostatních liší především tím, že vznikl na základě fuzzy E-R. Jedná se především o promítnutí jednotlivých úrovní fuzzyfikace do databáze formou pomocných atributů a tabulek.

7.1. Globální rozšíření a základní značení

Pro zjednodušení popisu celého schématu bude databáze rozšířena o tabulku

Tables(Table_Id, Table_Name, Is_Fuzzy)

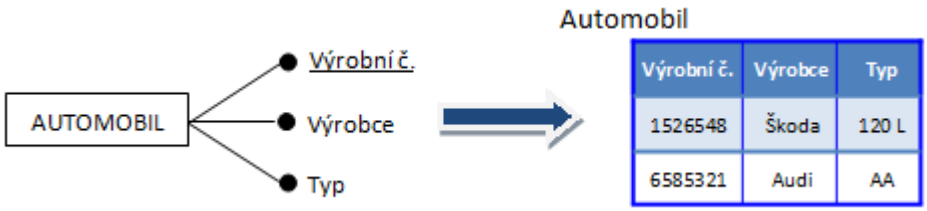
pro zjednodušení indexace a zrychlení procesu výpočtu. Tato tabulka bude obsahovat záznamy o všech tabulkách, které neslouží jako pomocné.

Dále je nutné všechny tabulky, které nejsou pomocné, rozšířit o primární klíč (Id), který slouží k další indexaci a nebude fuzzyfikován.

7.2. Převod entity

Nefuzzyfikovaná entita je reprezentována odpovídající tabulkou a jednotlivé instance jsou uloženy jako záznamy v tabulce (Obrázek 30). V případě fuzzyfikované entity je nutné tuto tabulku doplnit o atribut pro reprezentaci stupně příslušnosti jednotlivých záznamů ($\mu_E(e)$). Dále je nutné zajistit unikátnost jména atributu a vytvořit označení tabulky, aby bylo zřejmé, že se jedná o fuzzyfikovanou entitu (tabulku). Informace o tom, že je tabulka fuzzyfikována je uložena v tabulce *Tables* jako atribut *Is_Fuzzy* a atribut udávající stupeň příslušnosti bude mít název @. Příkladem je tabulka

Starý automobil(Id, @, Výrobní č., Výrobce, Typ) (Obrázek 31)



Výrobní č.	Výrobce	Typ
1526548	Škoda	120 L
6585321	Audi	AA

Obrázek 30 – Převod nefuzzyfikované entity

Id	@	Výrobní č.	Výrobce	Typ
1	0,7	1526548	Škoda	120 L
2	0,2	6585321	Audi	AA
3	1	5852365	Škoda	105

Obrázek 31 – Převod fuzzyfikované entity

Na ukázkových obrázcích není žádný z atributů fuzzyfikován. Fuzzyfikaci jednotlivých atributů je věnována sekce 7.4.

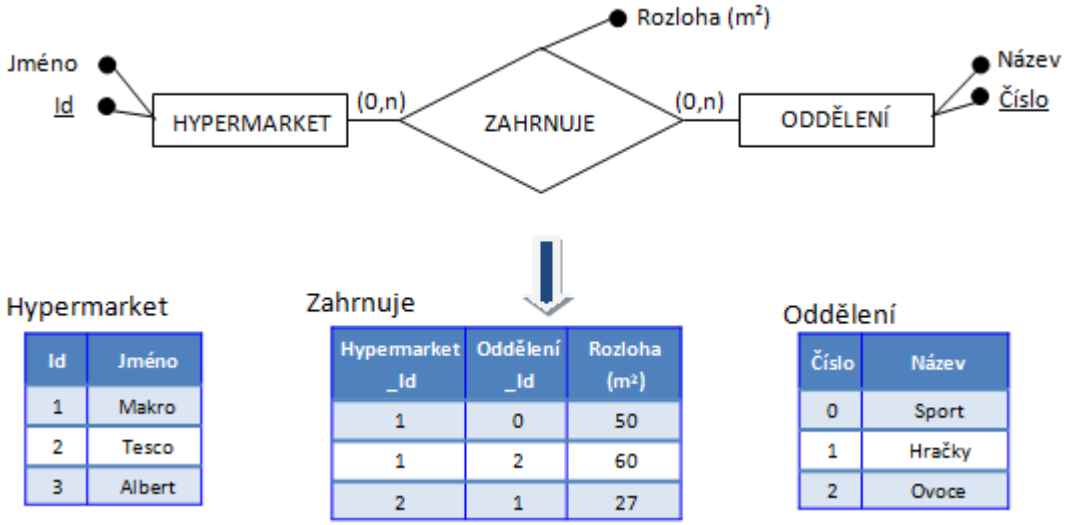
7.3. Převod vztahu

V rámci převádění vztahu je nutné rozlišovat dva případy na základě kardinality vztahu. Jedná se o vztahy typu (m,n), (1,n) a vztahy typu (1,1). Podobně jako v případě entit je potřeba do databáze přenést informaci o přesnosti (pravdivosti) jednotlivých instancí vztahů.

7.3.1. Vztahy s vysokou kardinalitou

Jedná se o vztahy, jejichž počet instancí vůči jedné instanci entity není omezen. Konkrétně jsou to vztahy typu (m,n). Z důvodu neznámého počtu instancí je vztah převeden na tabulku, kde jednotlivé záznamy obsahují odkazy na instance obou entit (jejich primární klíče) a dále pak další atributy náležející přímo vztahu (Obrázek 32). V případě fuzzyfikovaného vztahu je nutné do tabulky přidat informaci o přesnosti vztahu. Ta je podobně jako v případě fuzzyfikovaného elementu reprezentována jediným atributem označeným znakem @. Konkrétní příklad popisuje Obrázek 33, kde jsou vytvořeny tabulky:

Hypermarket(Id, Jméno)
Oddělení(Číslo, Název)
Zahrnuje(Id, Hypermarket_Id, Oddělení_Id, Rozloha)



Hypermarket

Id	Jméno
1	Makro
2	Tesco
3	Albert

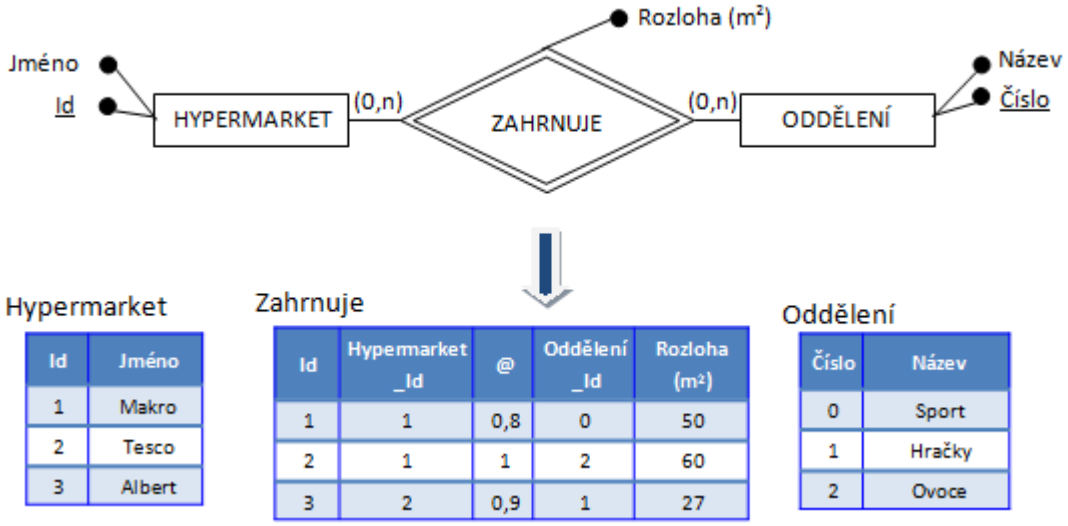
Zahrnuje

Hypermarket_Id	Oddělení_Id	Rozloha (m²)
1	0	50
1	2	60
2	1	27

Oddělení

Číslo	Název
0	Sport
1	Hračky
2	Ovoce

Obrázek 32 - Převod nefuzzyfikovaného vztahu



Hypermarket

Id	Jméno
1	Makro
2	Tesco
3	Albert

Zahrnuje

Id	Hypermarket_Id	@	Oddělení_Id	Rozloha (m²)
1	1	0,8	0	50
2	1	1	2	60
3	2	0,9	1	27

Oddělení

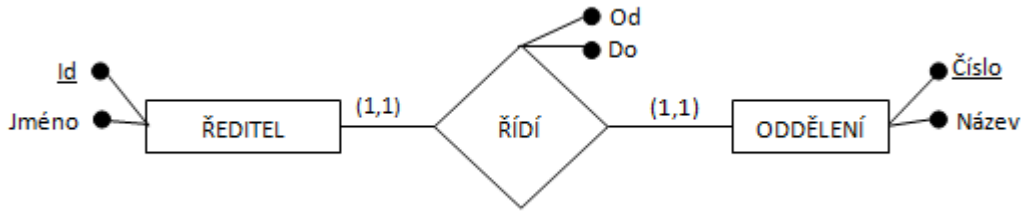
Číslo	Název
0	Sport
1	Hračky
2	Ovoce

Obrázek 33 - Převod fuzzyfikovaného vztahu

7.3.2. Vztahy s nízkou kardinalitou

Jedná se o vztahy typu (1,1) a (1,0). Instance vztahu je pro každou instanci entity nejvýše jedna. Proto není nutné vytvářet speciální tabulku pro reprezentaci vztahu a stačí přidat do tabulky reprezentující entitu (případně do obou v závislosti na použitelnosti) přidat atribut sloužící jako odkaz na primární klíč tabulky odpovídající druhé entitě. Dále je nutné doplnit všechny atributy vztahu (Obrázek 34). V případě, že je vztah fuzzyfikován, bude atribut odkazující se na primární klíč začínat znakem @. Dále bude do tabulky přidán speciální atribut *@Odkaz_na_primární_klíč_FuzzyConjunction*, který shodně jako při převodu vztahů s vysokou kardinalitou nese informaci o přesnosti vztahu. Konkrétní případ znázorňuje Obrázek 35, kde vzniknou následující tabulky:

Ředitel(Id, Jméno, @Oddělení_Číslo, @Oddělení_Číslo_FuzzyConjunction, Od, Do)
Oddělení(Id, Číslo, Název)



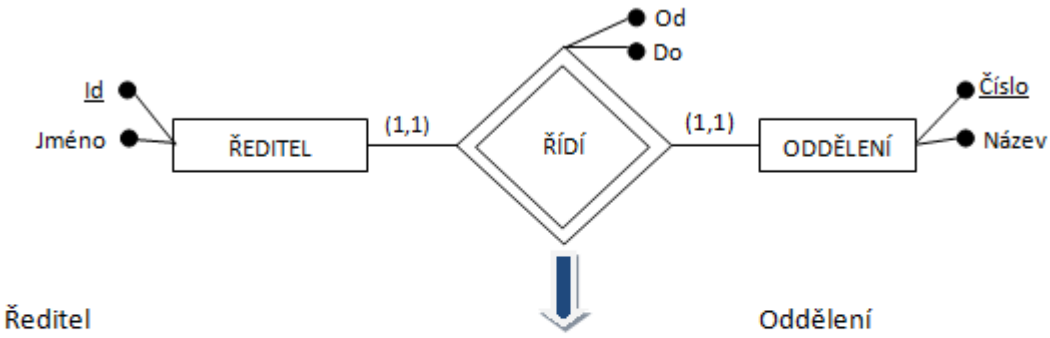
Ředitel

Id	Jméno	Oddělení_Číslo	Od	Do
0	Karel Novák	82	4.5.2002	6.10.2005
1	Jan Zahálka	115	6.9.1998	11.12.2010
2	Petra Zmoklá	337	9.3.1999	2.1.2008

Oddělení

Číslo	Název
115	Výroba
82	Marketing
337	PR

Obrázek 34 - Převod nefuzzyfikovaného vztahu



Ředitel

Id	Jméno	@Oddělení_Číslo	@Oddělení_Číslo_FuzzyConjunction	Od	Do
0	Karel Novák	82	0,5	2.3.2001	2.5.2011
1	Jan Zahálka	115	0,3	3.2.2000	11.8.2011
2	Petra Zmoklá	337	0,8	6.9.1998	13.2.2006

Oddělení

Id	Číslo	Název
1	115	Výroba
2	82	Marketing
3	337	PR

Obrázek 35 - Převod fuzzyfikovaného vztahu

Tento přístup se dá použít i pro kardinalitu (1,n). V případě, že se entita vyskytuje ve více vztazích, jsou přidány příslušné atributy za každý vztah, ve kterém se entita vyskytuje.

7.4. Převod atributu

Jak je popsáno v sekci 5.3, fuzzy funkci atributu (v navrhnutém rozšíření) lze popsat pomocí čtyř zlomových proměnných a případně dalších parametrů. Protože je nutné specifikovat, které atributy jsou fuzzyfikovány a jak se chovají jednotlivé fuzzy funkce, bude schéma rozšířeno o další tabulky. Jsou to:

- **Fuzzy_Attributes**(Attribute_Id, Attribute_Name, Table_Id, Is_Fuzzy, Is_Numerical)
  V této tabulce jsou zaznamenány všechny fuzzyfikované atributy. Jednotlivé záznamy obsahují odkaz do tabulky se jmény tabulek (Tables), aby bylo možné jednoznačně rozlišit atributy se shodným jménem v různých tabulkách.
- **Fuzzy_Functions**(Fuzzy_Function_Id, Fuzzy_Function_Name, Attribute_Id, Type, A, B, C, D)
  Tato tabulka popisuje jednotlivé fuzzy funkce příslušných atributů. Každý atribut může mít přiřazenu množinu fuzzy funkcí, ze kterých pak může případně zadávající vybírat. Hodnoty A, B, C, D slouží jako popis zlomových bodů. Atribut Type určuje, o jaký typ fuzzy funkce se jedná (rozsah, rozsah^{+}, rozsah^{-}, přibližnost).
- **Fuzzy_Parameters**(Id, Attribute_Id, Record_Id, A, B)
  Slouží pro informace o parametrech jednotlivých fuzzy funkcí u konkrétních záznamů. V případě tohoto návrhu slouží pouze pro funkci přibližnosti (sekce 5.3), takže by bylo možné parametr zakomponovat přímo do původní tabulky. Vybraná varianta však nabízí možnosti, jak funkčnost dále rozšířit o funkce s větším množstvím parametrů. Parametr Record_Id specifikuje konkrétní záznam a hodnoty A, B hodnotu parametru. Například pro funkci přibližnosti

udává parametr A hodnotu, kolem které se pohybujeme, a parametr B udává přibližnost.

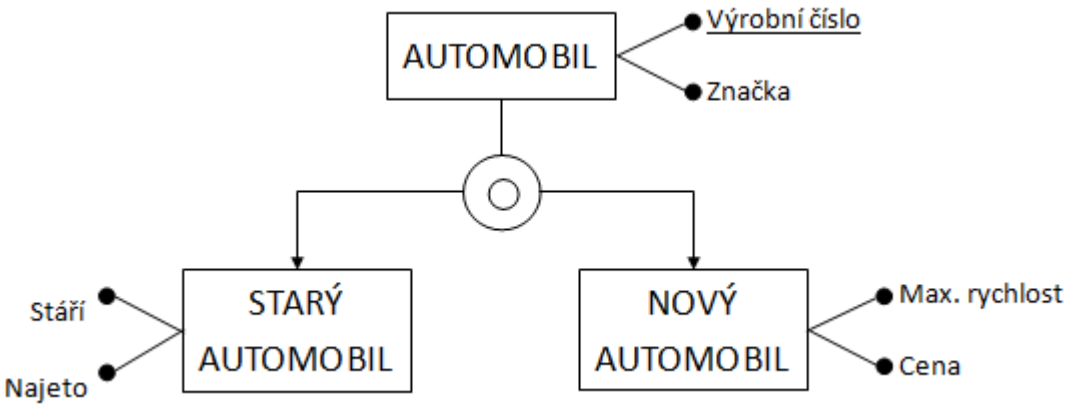
V rámci samotné tabulky je fuzzyfikovaný atribut vhodné označit. Pro jeho rozlišení bude použit počáteční symbol $. Dále pak je nutné dodat odkaz na funkci, kterou atribut používá. Tento odkaz je zastoupen atributem $jméno_atributu_FuzzyId (v případě, že se jedná o konkrétní hodnotu, je atribut prázdný). Pro lepší představu o tom, jak budou vypadat tabulky po zavedení fuzzy atributů, slouží Obrázek 36.

Obrázek 36 - Fuzzyfikace atributu Věk

V tomto případě je použita pouze jediná základní tabulka:
Lidé(Id, $Věk, $Věk_FuzzyId)
Pomocné tabulky pro jednodušší práci s databází a popis jednotlivých fuzzy funkcí jsou:
Tables(Table_Id, Table_Name)
Fuzzy_Attributes(Attribute_Id, Table_Id, Attribute_Name, Is_Fuzzy, Is_Numerical)
Fuzzy_Functions(Fuzzy_Function_Id, Fuzzy_Function_Name, Attribute_Id, Type, A, B, C, D)
Fuzzy_Parameters(Id, Attribute_Id, Record_Id, A, B)
Množství pomocných tabulek se již s přidáváním dalších fuzzyfikovaných atributů nebude zvyšovat.

7.5. Transformace ISA-hierarchie

Poslední částí návrhu je zvážení, jak uložit informace o ISA-hierachii. Jednou z možností je transformace na úrovni E-R modelu do fuzzy entit a fuzzy vztahů. To v tomto případě není nutné a ISA-hierarchie bude transformována přímo do tabulek. Pro ukázku transformace bude použit příklad, který popisuje Obrázek 37. V případě, že je hierarchie vícenásobná, bude se následující postup po řadě používat pro všechny účastníky hierarchie.



Obrázek 37 - ISA-hierarchie

7.5.1. Složení

Použitím tohoto přístupu vzniká z ISA-hierarchie jedna tabulka obsahující všechny atributy všech entit v ISA-hierarchii. Pokud by byla transformována hierarchie, kterou popisuje Obrázek 37, vznikne jediná tabulka:

Automobil(Id, Výrobní číslo, Značka, Stáří, Najeto, Max. rychlost, Cena)

Nevýhodou tohoto přístupu je vznik rozsáhlé tabulky, kde velká část atributů u jednotlivých záznamů nemusí být vyplněna. V našem příkladu nebudou mít staré automobily vyplněny atributy Max. rychlost a Cena. Podobně v případě normálních automobilů nebude využit ani jeden ze čtyř rozšiřujících atributů. Tento přístup není pro fuzzy reprezentaci příliš použitelný, protože není možné dobře reprezentovat stupeň příslušnosti a zároveň zajistit, aby bylo možné rozpoznat zpětně, že se jedná o ISA-hierarchii.

7.5.2. Úplné rozložení

Tento přístup je zcela opačný než předcházející. Z jednotlivých entit, které jsou součástí ISA-hierarchie, jsou vytvořeny tabulky, které na sobě nejsou závislé. V našem příkladu by vznikly tři tabulky:

Automobil(Id, Výrobní číslo, Značka)
Starý automobil(Id, Výrobní číslo, Značka, Stáří, Najeto)
Nový automobil(Id, Výrobní číslo, Značka, Max. rychlost, Cena)

V tomto přístupu jsou již využitelné vždy všechny atributy. Nevýhodou je samostatnost tabulek a ztráta vazby dané ISA-hierarchií. Zvláště kvůli ztrátě vazby tabulek mezi sebou byl tento přístup pro transformaci zamítnut.

7.5.3. Částečné rozložení

Tento přístup vytváří podobně jako předchozí přístup samostatné tabulky, kde jedna z tabulek slouží jako hlavní a ostatní se na ni odkazují s doplňujícími informacemi. Hlavní tabulka obsahuje jako záznamy všechny instance všech entit v hierarchii s atributy společnými všem entitám. Další tabulky se na ni odkazují a obsahují rozšiřující informace v závislosti na tom, jaké entity reprezentují. V našem případě vzniknou tři tabulky:

Automobil(Id, Výrobní číslo, Značka)
Starý automobil(Id, Automobil_Výrobní číslo, Stáří, Najeto)
Nový automobil(Id, Automobil_Výrobní číslo, Max. rychlost, Cena)

Toto schéma nejlépe popisuje vztah nadentit a podentit, a bylo proto zvoleno pro transformaci ISA-hierarchie. Aby bylo zajištěno jednoduché rozpoznání, bude název atributu reprezentujícího odkaz od podentity k nadentitě (v rámci reprezentace tabulkami) a název tabulky zakončen symbolem !. Jednotlivé entity lze fuzzyfikovat, jak bylo popsáno v sekci 7.2. Dále je možné zařadit jeden záznam do dvou tabulek reprezentujících podentity s různým stupněm příslušnosti. Pokud budeme v našem příkladu předpokládat, že chceme fuzzyfikovat obě podentity, budou poté vzniklé tabulky následující:

Automobil

Id	Výrobní číslo	Značka
1	1111	Škoda
2	2222	Audi
3	3333	Škoda
4	4444	Opel
5	5555	Honda
6	6666	Hyundai

Nový automobil!

Id	@	Automobil_Výrobní číslo!	Výkon	Cena
1	0,7	1111	110	12000
2	0,9	2222	200	100000
3	0,5	6666	150	92000

Starý automobil!

Id	@	Automobil_Výrobní číslo!	Stáří	Najeto
1	0,4	3333	5	15000
2	0,9	4444	7	15000
3	1	5555	9	40000
4	0,5	6666	3	5000

Obrázek 38 - Transformace ISA-hierarchie

7.6. Dotazování v rámci navržené FR-DB

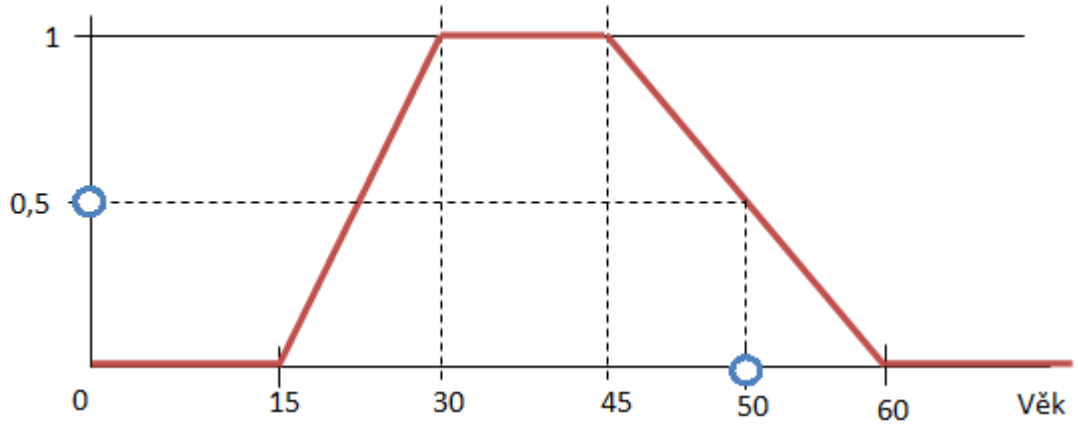
Pokud je vytvořena databáze, je nutné zvážit, jaké prostředky využít pro dotazování a jak samotné vyhodnocení dotazu provádět. Mezi základní prostředky patří:

- Použití stávajícího jazyka pro dotazování
  Toto řešení vyžaduje, aby uživatel znal schéma databáze, orientoval se v oblasti fuzzyfikace a znal samotný návrh. Dále pak veškerá výpočetní zátěž spadá na databázový server.
- Použití nadstavbového jazyka
  V případě použití nového dotazovacího jazyka, který se bude na základní jazyk překládat, se snižují požadavky na znalosti uživatele v oblasti fuzzyfikace a samotné databáze. Veškeré výpočty však provádí stále databázový server.
- Použití aplikačního rozhraní
  Na klientské části je vytvořena aplikace pro komunikaci s databází, která umožňuje uživateli provádět základní operace. Tato aplikace dále využívá stávající jazyk pro komunikaci s databází. Hlavní výhodou je možnost převedení části výpočetní zátěže na počítač klienta a nezatěžovat příliš databázový server. Nevýhodou je omezení funkčnosti aplikace. Toto řešení bylo vybráno pro další postup a testování.

7.6.1. Vyhodnocení dotazu

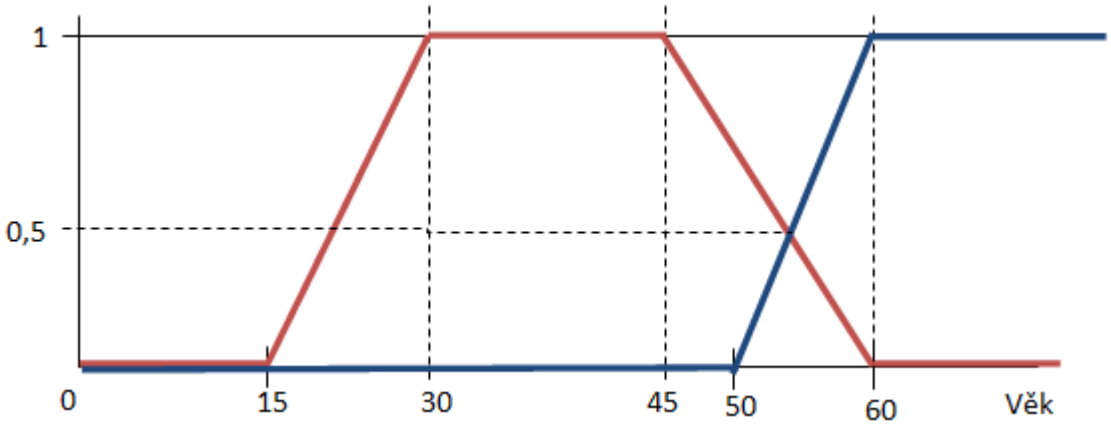
Vyhodnocení dotazu jako celku bude probíhat v několika na sebe navazujících krocích. Vzhledem k tomu, že uživatel bude používat aplikaci, jejíž ovládání by mělo být intuitivní, nebude samotné dotazování využívat plnou sílu dotazovacího jazyka a bude možné použít pouze základní dotazy. V našem případě pouze vyhodnocení podmínek nad jednotlivými tabulkami, spojování tabulek. Pro rozšíření funkčnosti je nutné zvážit, jak se změna promítne na uživatelské aplikaci a jak na samotném hodnocení. Jednotlivé kroky vyhodnocení jsou:

1. **Výpočet hodnot fuzzy funkcí pro jednotlivé části podmínek dotazu v rámci tabulky**
   Na základě stanovených fuzzy funkcí je možné jednoduchým výpočtem zjistit *míru splnění podmínky* v intervalu [0,1]. Na podmínku se můžeme dívat podobně jako na fuzzy data a spočítat funkční hodnotu průniku obou funkcí. Podle toho, zda je dotaz zadán běžným způsobem nebo také fuzzyfikovaně, je možné rozlišit čtyři případy:
   - Dotaz je zadán hodnotou, atribut aktuálního záznamu je zadán hodnotou
     Odpovídá vyhodnocení v rámci normální databáze. V případě, že je podmínka splněna, má míra splnění podmínky hodnotu 1, v opačném případě 0.
   - Dotaz je zadán hodnotou, atribut aktuálního záznamu je zadán fuzzy funkcí
     V tomto případě je míra splnění podmínky funkční hodnotou fuzzy funkce v bodě, který odpovídá hodnotě zadané záznamem. Pro lepší představu slouží Obrázek 39. Mějme tabulku reprezentující seznam osob. U aktuálního záznamu je věk popsán fuzzy funkcí: *středně starý* (na obrázku červeně). Dotaz zní: „Najdi všechny padesátileté osoby“. Potom míra splnění tohoto jednoduchého dotazu bude mít hodnotu 0,5. Shodně jako tento případ se chová i případ opačný, kde je dotaz zadán fuzzy funkcí a atribut hodnotou.



Obrázek 39 - Hodnota a fuzzy funkce

- Dotaz i atribut je zadán fuzzy funkcí
  Podobně jako v předchozím případu je hodnota míry splněnosti dána maximální funkční hodnotou průniku těchto funkcí. Například Obrázek 40 znázorňuje dotaz: "Najdi všechny staré (modrá křivka) lidi" a věk v aktuálním záznamu je popsán jako *středně starý* (červená křivka). Míra splněnosti je v tomto případě 0,5.



Obrázek 40 - fuzzy dotaz a fuzzy hodnota

Konkrétní případy, které je nutné zvážit jsou rozebrány v sekci 8.3. Výpočet míry splnění podmínek u jednotlivých záznamů záleží na tom, jak se autor rozhodne interpretovat jednotlivé fuzzy funkce a jejich průniky.

2. **Výpočet míry splnění podmínky v rámci celého záznamu**
   Pro tento výpočet je nutné zvolit si vhodnou funkci příslušnosti (sekce 3.2), která bude použita pro spojení jednotlivých hodnot udávajících míry splnění podmínek pro konkrétní tabulku. Pro náš pokus byla zvolena pro všechny logické spojky produktová spojka. Například pokud by dotaz zněl: "Najdi všechny staré osoby, které mají vysoký plat", potom středně starý Karel Zelinka s vysokým platem bude splňovat podmínku s mírou 0,5 (za předpokladu, že jednotlivé míry splnění jsou 0,5 a 1).

3. **Výpočet na úrovni fuzzyfikovaného záznamu**
   Pokud tabulka vznikla z fuzzyfikované entity nebo vztahu, je nutné aplikovat atribut udávající stupeň příslušnosti daného záznamu (v tabulce se jedná o hodnotu atributu se jménem @). Zde záleží čistě na individuální představě řešitele, který systém implementuje. V tomto řešení bylo zvoleno vynásobení míry splnění celého záznamu stupněm příslušnosti a získání tak nového stupně příslušnosti celého záznamu.

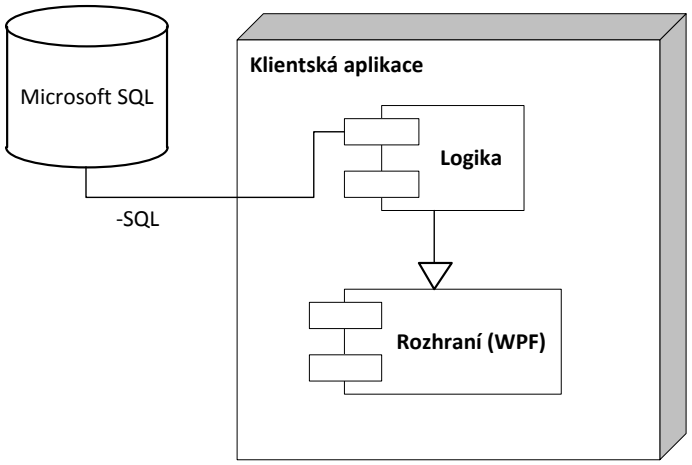
4. **Spojení tabulek**
   Poslední akcí, kterou je možné provést, je libovolné spojení tabulek (přirozené, křížové, …). V tomto případě nové záznamy vytvořené spojením dílčích záznamů získají stupeň příslušnosti vynásobením stupněm příslušnosti záznamů, ze kterých byly složeny.

8. Návrh programu pro dotazování nad FR-DB (SIFQ)

Tato kapitola je věnována návrhu a popisu fungování programu vytvořeného pro dotazování do FR-DB popsané v předchozí kapitole. Tento program slouží především pro testovací účely s cílem zjistit, zda je možné použít software jako rozhraní k fuzzyfikované relační databázi. Dále pak jak velká je zátěž na databázi a na klientskou část, kde je program nasazen. Pro komunikaci programu s databází je použit pouze jazyk SQL a není vytvořena nebo použita žádná jeho nadstavba. Vytvořený testovací software SIFQ (Simple Interface for Fuzzy Querying) je na přiloženém CD včetně programátorské a uživatelské dokumentace.

8.1. Použité nástroje pro vývoj

Celý program SIFQ je implementován v jazyce C# za použití Microsoft Visual Studia 2010. Pro vytvoření uživatelského rozhraní slouží technologie WPF (Windows Presentation Foundation). Jako databázový server, na kterém je testovací databáze nasazena, je použit MS SQL Server. Strukturu programu včetně použitých technologií popisuje Obrázek 41.



Obrázek 41 - Struktura programu

8.2. Fungování programu

Hlavním cílem programu SIFQ je snížení zátěže databázového serveru na minimum a naopak veškeré výpočty lokalizovat na stroj, kde je spuštěn program. Proto je funkcionalita programu rozdělena do následujících fází:

- Načtení informací o databázi a vytvoření konfliktních tabulek
- Zadání dotazu uživatelem
- Vytvoření SQL dotazu a získání odpovídajících záznamů z databáze
- Výpočet přesnosti jednotlivých záznamů
- Interpretace výsledků

8.2.1. Načtení informací o databázi a vytvoření konfliktních tabulek

Informace o složení databáze získá program z tabulek *Fuzzy_Tables*, *Fuzzy_Attributes* a *Fuzzy_Functions* popsaných v kapitole 7. Tyto informace dále

slouží pro vytvoření uživatelského rozhraní, kde může uživatel následně přiřazovat jednotlivým atributům jejich hodnoty, případně fuzzy funkce a jejich parametry a specifikovat tím podmínky. Dále je vytvořena tabulka konfliktů pro každý atribut, ve které jsou specifikovány konflikty fuzzy funkcí rozsah, rozsah^{+} a rozsah^{-} (mají průnik s hodnotou > 0). Předpokládejme například, že jsou zadány fuzzy funkce pro atribut věk: Mladý (rozsah^{-}) se zlomovými body 25 a 30, Středně starý (rozsah) se zlomovými body 20, 30, 40, 50 a Starý (rozsah^{+}) se zlomovými body 45 a 60. Tabulku konfliktů reprezentuje Tabulka 4.

	Mladý	Středně starý	Starý
Mladý	ANO	ANO	NE
Středně starý	ANO	ANO	ANO
Starý	NE	ANO	ANO

Tabulka 4 - konfliktní tabulka atributu Věk

Tato tabulka slouží při následném vytváření dotazu pro snížení počtu dotazů do databáze a bylo možné část dotazu generovanou na základě konfliktů těchto fuzzy funkcí vytvořit samostatně.

8.2.2. Zadání dotazu uživatelem

V této fázi uživatel vybírá hodnoty u jednotlivých atributů vybraných tabulek. Výsledný dotaz je konjunkcí zadaných podmínek. Spojení tabulek je možné provádět jako křížové, případně lze zadat parametry pro spojení. Pro více informací slouží uživatelská dokumentace na přiloženém CD a jako příloha této práce (přílohy - kapitola 1).

8.2.3. Vytvoření SQL dotazu a získání odpovídajících záznamů z databáze

Na základě podmínek specifikovaných uživatelem je vygenerován SQL dotaz, pomocí kterého jsou získána veškerá relevantní data. Protože je cílem, aby databázový server nemusel počítat veškeré funkční hodnoty pro jednotlivé funkce, je v rámci každého atributu pro jednotlivé typy funkcí vytvořena podmínka (ta je zabudována ve WHERE části dotazu), jenž specifikuje, které funkce a případně které jejich hodnoty mají s uživatelem zadanou podmínkou reálný průnik. Ostatní funkce a hodnoty by získaly v průběhu ohodnocení hodnotu 0, která by při použití produktové spojky vynulovala hodnotu přesnosti (příslušnosti) celého záznamu. SQL dotaz vygenerován při uživatelském dotazu: „Chci všechny mladé lidi, kteří měří zhruba 175 cm.“ by mohl vypadat následovně:

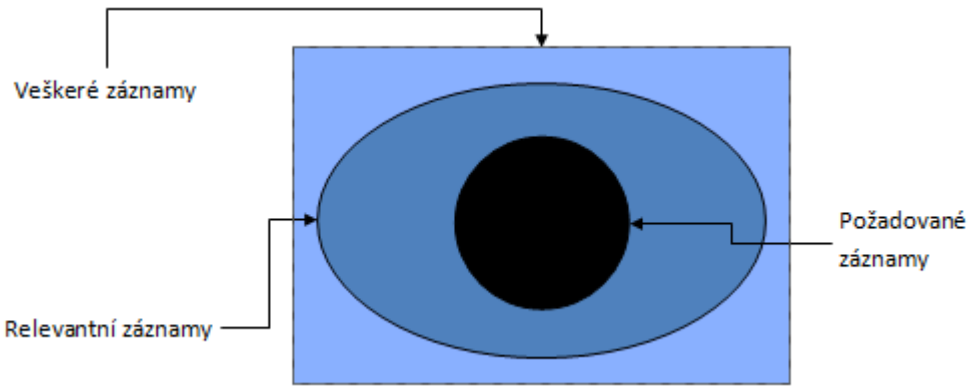
```
SELECT
Osoba.[$Vek] AS 'Osoba_Vek',
Osoba.[$Vek_FuzzyId] AS 'Osoba_Vek_FuzzyId',
Osoba_Vyska_Parameters.A AS 'Osoba_Vyska_Parameters_A',
Osoba_Vyska_Parameters.B AS 'Osoba_Vyska_Parameters_B',
Osoba.[$Vyska] AS 'Osoba_Vyska',
```

```
Osoba.[$Vyska_FuzzyId] AS 'Osoba_Vyska_FuzzyId',
Osoba.Jmeno AS 'Osoba_Jmeno',
Osoba.Prijmeni AS 'Osoba_Prijmeni'

FROM
Osoba LEFT OUTER JOIN Fuzzy_Parameters AS Osoba_Vyska_Parameters ON
(Osoba.Id = Osoba_Vyska_Parameters.Record_Id AND
Osoba_Vyska_Parameters.Attribute_Id = 2)

WHERE
(((Osoba.[$Vek_FuzzyId] = 1 OR Osoba.[$Vek_FuzzyId] = 2) OR (Osoba.[$Vek] <
30)) AND ((Osoba.[$Vyska_FuzzyId] = 5) OR (Osoba_Vyska_Parameters.A > 145
AND Osoba_Vyska_Parameters.A < 195 AND Osoba.[$Vyska_FuzzyId] = 4) OR
(Osoba.[$Vyska] > 160 AND Osoba.[$Vyska] < 185)))
```

V tomto případě se jedná o SQL dotaz vygenerovaný přímo programem SIFQ, a obsahuje proto další součásti, jako například přejmenování atributů pro zajištění unikátnosti jednotlivých jmen. Pro lepší představu o struktuře celé databáze slouží Obrázek 49, Obrázek 50 a Obrázek 51.

Při vytváření dotazu je nutné brát v potaz všechny kombinace, které mohou v rámci jednoho atributu vzniknout. Uživatel může zadat dotaz pomocí jedné z fuzzy funkcí nebo pomocí konkrétní hodnoty (>, < , =). Na druhé straně i data v databázi mohou být zadána stejným způsobem. Proto velmi roste velikost výsledného dotazu.

Tento přístup v sobě zahrnuje dvě nevýhody. První z nich: rozsah vygenerovaného SQL dotazu je zmíněn výše. Dalším problémem je, že pomocí takto vytvořeného dotazu jsou všechna relevantní data (záznamy s přesností ostře větší než 0) získána z databáze. Pokud by uživatel požadoval větší přesnost, tak by část záznamů získaných z databáze přestala být validní. Tyto záznamy program filtruje podle zadané přesnosti po získání záznamů. Situaci pro zpřehlednění znázorňuje Obrázek 42, kde jsou záznamy zakresleny množinově.



Obrázek 42 - Odpovídající záznamy

Tento problém lze částečně zmírnit za použití heuristik při vytváření dotazu. Tyto možnosti jsou popsány v sekci **Error! Reference source not found.**.

8.2.4. Výpočet přesnosti jednotlivých záznamů

Po přijetí záznamu dochází na počítači s programem k jeho vyhodnocení. To probíhá na základě typu dotazu a typu výsledku. Podobně jako v případě tvorby dotazu je nutné zvážit veškeré kombinace podmínky zadané uživatelem a výsledných dat. Výpočtu průniků jednotlivých funkcí a hodnot mezi sebou se věnuje sekce 8.3.

8.2.5. Interpretace výsledků

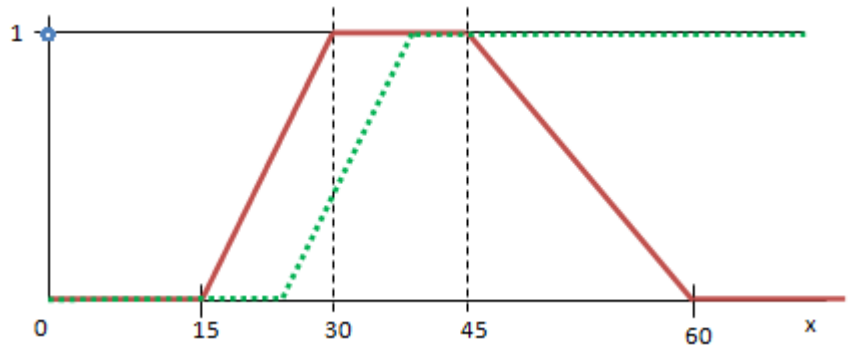
Získané výsledky je dále nutné interpretovat uživateli. Nejprve bylo zavedeno zobrazení přímo pomocí software jako samostatného okna s tabulkou. Toto řešení však není použitelné při velkém počtu výsledných záznamů (paměťová náročnost, doba zobrazování). Proto byla zavedena možnost zápisu do souboru, kde jsou problémy zobrazení výstupu programem eliminovány na úkor přehlednosti.

8.3. Řešení jednotlivých výpočtů

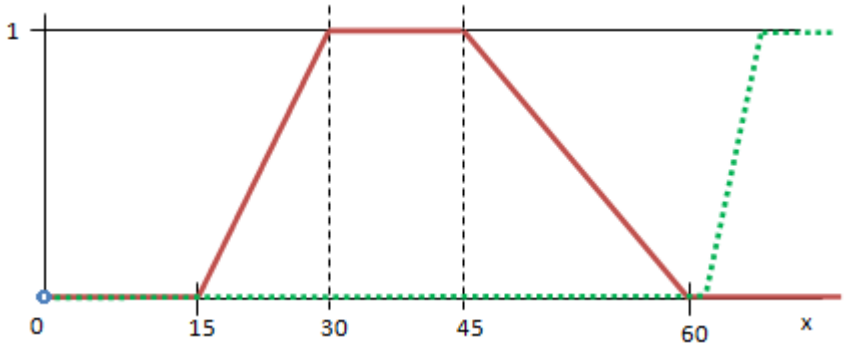
Díky výběru produktové spojky jako operátoru pro výpočet přesnosti jednotlivých záznamů, omezení programu na konjunktivní dotazy a vhodně zvolených funkcí spočívá problém vyhodnocení jednoho záznamu pouze ve výpočtu průniku fuzzy funkcí (případně reálných hodnot) mezi sebou. Je nutné navrhnout výpočty tak, aby odpovídaly běžnému chápání světa. Proto je velká řada z nich intuitivní. Při výpočtech je možné zaměnit fuzzy funkci dotazu (hodnotu) s potenciálním výsledkem a chovat se k nim jako k dvojici fuzzy funkcí. Nejedná se o funkce v matematickém slova smyslu, protože je možné, aby v jednom bodě nabývala funkce pomyslně více hodnot, aby bylo zajištěno zjištění průniku. Tato pomůcka je čistě vizuální a v rámci výpočtu nic nemění. Následující sekce popisují možnosti, které mohou v rámci průniku dvou fuzzy funkcí nastat a jakých hodnot nabývají.

8.3.1. Průniky jednotlivých rozsahových funkcí

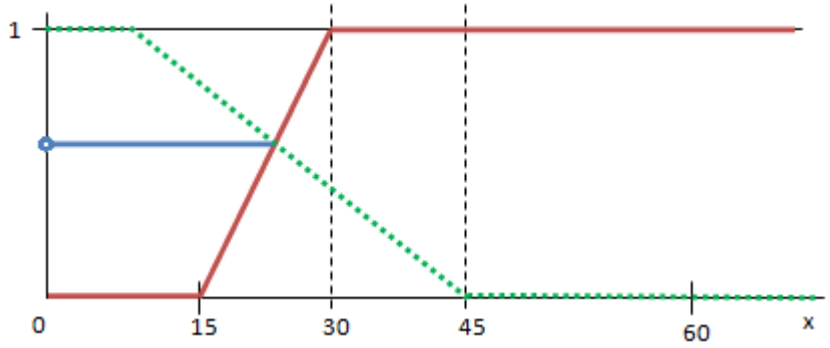
V případě, že jsou obě funkce rozsahové (rozsah, $rozsah^+$, $rozsah^-$), je výpočet průniku intuitivní. Každou takovou funkci lze rozdělit na tři části. Na oblast, kde nabývá funkce hodnotu 0, na oblast nabývající hodnotu 1 a na oblast, kde se hodnoty průběžně zvyšují (snižují). V případě, že se funkce protínají v oblasti, kde obě nabývají hodnoty 1, je i výsledná hodnota přesnosti 1 (Obrázek 43). Pokud se funkce protínají pouze v oblastech, kde obě nabývají hodnotu 0, je výsledná hodnota přesnosti 0 (Obrázek 44). V posledním případě stačí vypočíst hodnotu průniku na základě stanovených zlomových bodů obou funkcí (Obrázek 45). Na následujících obrázcích je vyznačena výsledná funkční hodnota na ose y pomocí kolečka.



Obrázek 43 - Přesnost 1



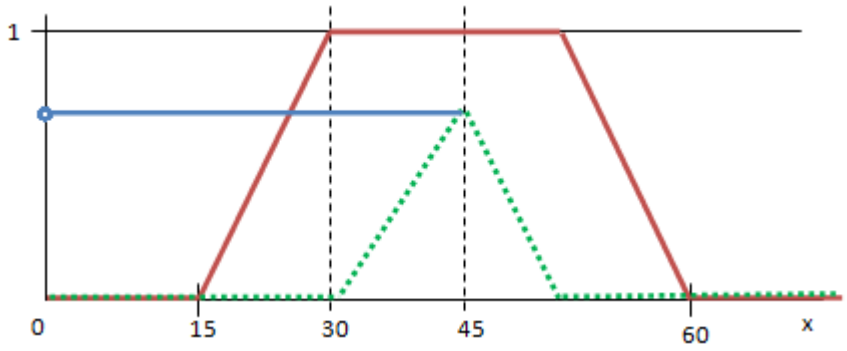
Obrázek 44 - Přesnost 0



Obrázek 45 - Průnik

8.3.2. Průniky s fuzzy funkcí přibližnosti

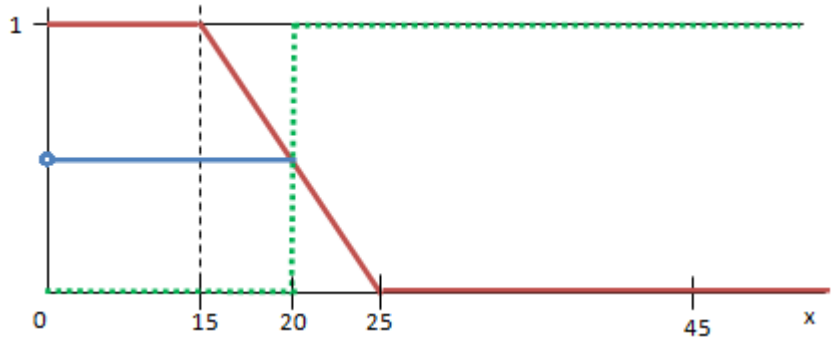
K fuzzy funkcím pro přesnost se lze chovat shodně jako k fuzzy funkcím rozsahu s tím rozdílem, že u přesnosti chybí složka, kde by nabírala funkce hodnoty jedna ve více než jednom bodě. Dále je nutné zvážit speciální situaci, kde je funkce přesnosti ve svém nejvyšším bodě „pod“ druhou funkcí. V tomto případě je výslednou hodnotou funkční hodnota nejvyššího bodu (zlomu) funkce přesnosti (Obrázek 46).



Obrázek 46 – Funkce podobnosti "pod" srovnávanou funkcí

8.3.3. Průniky s reálnými hodnotami

Jako reálné hodnoty rozumíme podmínky >, < (je větší, je menší) a dotazy na rovnost resp. zda je porovnávaný výsledek zadán konkrétní hodnotou. K podmínkám >, < se lze chovat jako ke speciálním funkcím rozsahu^{+} a rozsahu^{-}, kde jsou oba zlomové body shodné a nabývají hodnoty dotazu. Pro názornou představu slouží Obrázek 47, kde je znázorněn průnik fuzzy funkce mladý (rozsah^{-}, zlomové body: 15, 25) a dotazu, ve kterém je specifikováno, že stáří hledané osoby je větší než 20 let.



Obrázek 47 - Průnik s reálnými hodnotami

Pokud jsou obě hodnoty atributu zadány konkrétní hodnotou, nabývá přesnost v rámci tohoto atributu hodnoty 1 nebo 0 podle toho, zda se hodnoty shodují nebo ne. Pro průnik s fuzzy funkcí je odpovídající hodnota přesnosti funkční hodnotou fuzzy funkce v bodě konkrétní hodnoty.

8.4. Možné úpravy a rozšíření

Protože se jedná o testovací software, není ve všech případech zamezeno uživateli zadat nesmyslné nebo neplatné dotazy. Proto by bylo vhodné jako jedno z prvních rozšíření programu ošetření proti zadání neplatných dat. Dalšími rozšířeními jsou především:

- **Vytvoření rozhraní pro správu databáze**
  Uživatel (administrátor) by mohl po vytvoření jednoduchého rozhraní pro správu databáze (Vytváření/mazání tabulek, správa záznamů) být zcela odstíněn od fungování databáze. Je však nutné, aby se veškeré změny v databázi propagovaly do pomocných tabulek. Dále je nutné, aby uživatel

měl alespoň základní znalosti o tom, jak software funguje a jak se úpravy projeví v rámci databáze.

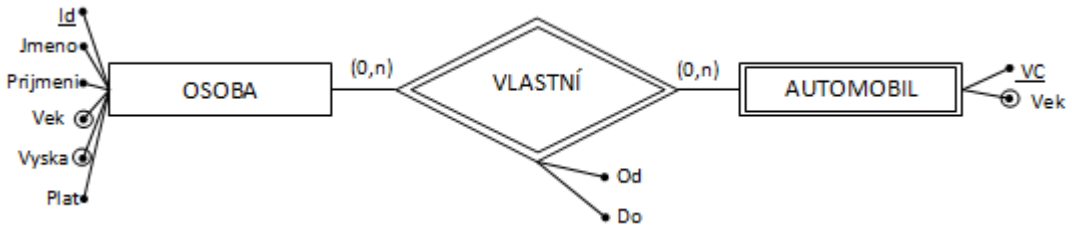
- **Použití heuristik pro vytváření dotazu**
  Tato část se týká dotazů, u nichž je přesnost omezena zespodu zadáním uživatele. Díky volbě produktové spojky jako operátoru a konjunktivních dotazů se jedná o vynásobení hodnot, kde některé z nich lze předpočítat (průniky funkcí rozsahu). Poté není nutné fuzzy funkce s funkční hodnotou průniku brát v potaz, protože dalším násobením se tato hodnota pouze sníží. Lze například rozšířit konfliktní tabulku tak, aby obsahovala místo informace, zda ke konfliktu dochází, i hodnotu průniku konfliktních funkcí.
- **Zvýšení funkčnosti nabízené uživateli**
  V základní verzi je program SIFQ značně omezen oproti jazyku SQL a je možné přidat další funkcionalitu například v podobě dalších běžných operátorů (`LIKE`, ≥,...). Je nutné zvážit, jaký tato rozšíření budou mít vliv na již zavedenou fuzzyfikaci a jak případně změnit pomocné tabulky.
- **Zavedení disjunkce a negace**
- **Zavedení více typů fuzzy funkcí**
  Zavedené funkce popisují velkou část často požadovaných případů z běžného života. Přesto jimi nelze vystihnout některé případy. Je nutné vzít v potaz, zda bude s nově přidanou fuzzy funkcí aplikace schopná stále předpočítávat SQL dotaz a zda nebude nutné zcela změnit systém fungování.

9. Testování provedená s programem SIFQ

Účelem tohoto testování je zjistit, zda výpočty potřebné ke zjištění přesnosti jednotlivých záznamů nezatěžují příliš zařízení, na kterém je program nasazen. Pro testování bude použita jednoduchá databáze, na které se budeme během testování dotazovat třemi různými dotazy, které se liší složitostí a počtem relevantních záznamů. Nebude měněna dolní hranice, kterou uživatel může zadat, neboť její změnou se délka výpočtu nemění. Testovanými hodnotami, které program SIFQ měří, jsou počet výsledků a doba zpracování všech relevantních záznamů včetně setřídění podle přesnosti.

9.1. Struktura databáze

Jak je již nastíněno v úvodu, databáze je velmi jednoduchá a je vytvořena podle E-R schéma, které znázorňuje Obrázek 48.



Obrázek 48 - Schéma testovací db

Tabulky výsledné databáze vypadají takto:

Automobil(Id, @, $Vek, $Vek_FuzzyId, VC)
Osoba(Id, Jmeno, Prijmeni, $Vek, $Vek_FuzzyId, $Vyska, $Vyska_FuzzyId, Plat)
Vlastni(Id, @, Od, Do, Osoba_Id, Automobil_Id)

Pro vygenerování databáze slouží SQL script *GenerateSFD.sql*, který je na přiloženém CD. Samotné naplnění tabulek náhodnými daty lze provést pomocí konzolového programu Generator, který je na přiloženém CD (ovládání je intuitivní). Nevýhodou použití tohoto programu je rychlost, s jakou jednotlivé tabulky plní.

Pomocné tabulky popisující atributy a funkce vystihují následující obrázky.

	Table_Id	Table_Name	Is_Fuzzy
1	1	Osoba	0
2	2	Automobil	1
3	3	Vlastni	1

Obrázek 49 - Tabulka Tables

	Attribute_Id	Table_Id	Attribute_Name	Is_Fuzzy	Is_Numerical	Is_Conjunction
1	1	1	Vek	1	1	0
2	2	1	Vyska	1	1	0
3	3	1	Jmeno	0	0	0
4	4	1	Prijmeni	0	0	0
5	5	1	Plat	0	1	0
6	6	2	Vek	1	1	0
7	7	2	VC	0	1	0
8	8	3	Od	0	0	0
9	9	3	Do	0	0	0
10	10	3	Osoba_Id	0	1	0
11	11	3	Automobil_Id	0	1	0
12	12	1	Id	0	1	0
13	13	2	Id	0	1	0
14	14	3	Id	0	1	0

Obrázek 50 - Tabulka Fuzzy_Attributes

	Fuzzy_Function_Id	Fuzzy_Function_Name	Attribute_Id	Type	A	B	C	D
1	1	mlady	1	3	20	30	NULL	NULL
2	2	stredne stary	1	1	25	40	50	60
3	3	stary	1	2	50	60	NULL	NULL
4	4	cca	2	4	10	15	NULL	NULL
5	5	vysoky	2	2	170	180	NULL	NULL
6	6	maly	2	3	140	150	NULL	NULL
7	7	stare	6	2	5	6	NULL	NULL
8	8	nove	6	3	2	3	NULL	NULL

Obrázek 51 - Tabulka Fuzzy_Functions

9.2. Testovací zařízení

Testování bude prováděno na notebooku středního výkonu (4GB RAM, core i3). Na tomto stroji bude spuštěn i databázový server s nasazenou databází, aby nedocházelo k časovým prodlevám kvůli přenosu po síti. Protože je cílem zjistit, jak se program chová při přidávání dalších záznamů, je malá odchylka tolerovatelná. Nejde o přesné hodnoty, ale o fenomén, který se při testování projeví.

9.3. Jednotlivé uživatelské dotazy

Pro testování jsou stanoveny tři dotazy z reálného světa, kde první dotaz odpovídá složitostí běžnému dotazu, druhý dotaz je velmi jednoduchý a třetí dotaz je naopak složitější. Dotazy jsou pro přehlednost označovány jako D1, D2 a D3. Jejich znění je následující:

D1: Najdi všechny středně staré, vysoké lidi, kteří vlastní starý automobil.

D2: Najdi všechny lidi, kteří jsou starší 25 let a jsou menší než 180 cm.

D3: Najdi všechny lidi vysoké cca 170 cm (jedná se o přesnost, kde druhý parametr je 0.9), dále jsou středně staří a plat mají větší než 15 000 Kč. Tyto osoby musí vlastnit nový vůz, jehož VČ (výrobní číslo) je větší než polovina počtu záznamů o automobilech v databázi (v DB je uloženo 500 automobilů, potom VČ > 250).

Dotazy D1, D2 a D3 jsou záměrně volené tak, že v případě D1 je k dotazování použito pouze fuzzy funkcí, v D2 pouze konkrétních hodnot a v D3 kombinace fuzzy funkcí i reálných hodnot. SQL dotazy, které generuje program SIFQ, jsou pro svůj rozsah uloženy v souboru *TestSQLQueries.txt* na přiloženém CD.

9.4. Provedené testy

Testy byly rozděleny do tří skupin na základě dotazu a počet záznamů byl inkrementálně přidáván do jednotlivých tabulek. Počáteční počet záznamů v každé z tabulek je 1000 a koncový 1 mil. Výsledky testování jsou interpretovány následujícími tabulkami:

D1 – #Záznamů	#Výsledků	Doba výpočtu (ms)
1000	164	60
10000	1691	390
100000	16699	1123
300000	15045	1482
500000	41781	2886
700000	82299	9267
1000000	167442	15850

Tabulka 5 - Testy na dotazu D1

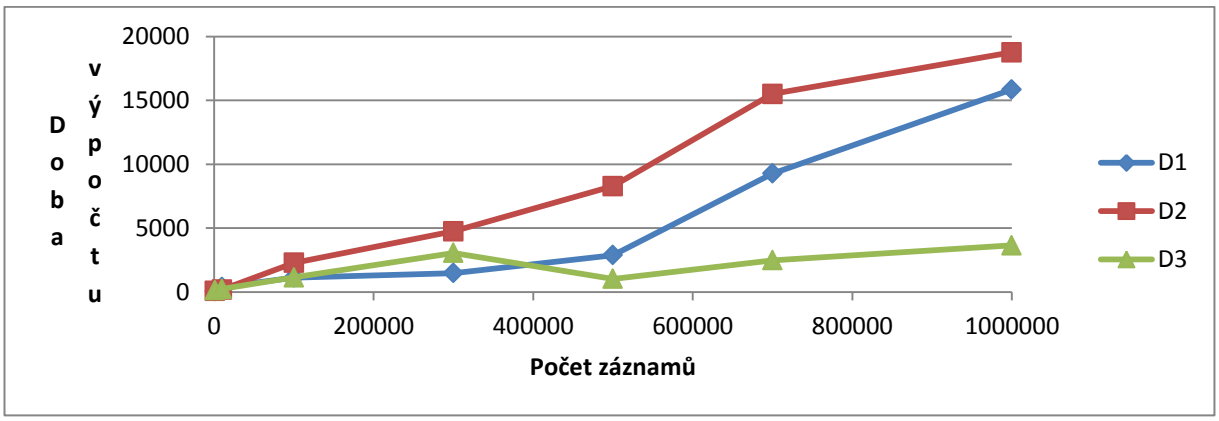
D2 – #Záznamů	#Výsledků	Doba výpočtu (ms)
1000	607	78
10000	6184	187
100000	61813	2277
300000	186005	4743
500000	310177	8284
700000	434122	15491
1000000	619962	18751

Tabulka 6 - Testy na dotazu D2

D3 – #Záznamů	#Výsledků	Doba výpočtu (ms)
1000	21	125
10000	285	234
100000	2814	1138
300000	817	3052
500000	6949	1027
700000	13760	2481
1000000	28036	3651

Tabulka 7 - Testy na dotazu D3

Pro lepší představu o výsledcích testování slouží Graf 1. Zde je každý z testů vyznačen jednou z lomených čar. Zobrazena je závislost počtu záznamů na době výpočtu přesnosti záznamů.



Graf 1 - Graf provedených testování

9.5. Závěry vyvozené z provedených testů

Jak je z grafu a naměřených hodnot patrné, zátěž na výpočet hodnot přesností všech záznamů je nízká a mění se pouze na základě složitosti dotazu a počtu odpovídajících záznamů. Tento výsledek byl předpokládán, protože pro každý atribut, podle kterého se uživatel dotazoval, bylo provedeno pouze malé množství aritmetických operací. Dále byla testována doba předzpracování a vytvoření konfliktní tabulky (není v tabulkách zaneseno). Ta se pohybovala pro všechny testy kolem hodnoty 120 ms a jedná se tedy pouze o konstantu, jejíž hodnota záleží na velikosti databáze, konkrétně pak na počtu použitých rozsahových funkcí. Dále by bylo vhodné na databázi s reálnými daty otestovat poměr počtu relevantních záznamů a záznamů, které uživatel skutečně považuje za směrodatné (omezení hodnoty přesnosti zespoda) a případně zlepšení této hodnoty za použití heuristik nastíněných v sekci **Error! Reference source not found.**.

10. Závěr

Fuzzy relační databáze jsou velmi užitečným rozšířením relačních databází a je možné tak zvýšit původní funkcionalitu o další možnosti z oblasti nepřesnosti. Fuzzyfikaci je možné provádět na různých úrovních. Vybraná fuzzyfikace na základě E-R modelu je vhodná, protože nejlépe odpovídá potřebám reálného světa a je možné na jejím základě jednoduše vytvořit odpovídající fuzzy relační databázi. V rámci samotného dotazování do databáze je možné využít různých přístupů. Každý z nich má své výhody a nevýhody. V této práci byl testován přístup za použití softwaru coby rozhraní mezi uživatelem a samotnou databází. Hlavní výhodou tohoto přístupu byla nízká zátěž databáze, protože veškeré výpočty byly zpracovány na straně uživatele. Nevýhodami jsou poté nízká funkcionalita a rozsah generovaných SQL dotazů.

Seznam použité literatury

[1] A. Urrutia, J. Galindo, L. Jimenéz a M. Piattini, „Data Modeling Dealing With Uncertainty in Fuzzy Logic“ *IFIP,* sv. 214, pp. 201-217, 2006.

[2] A. Zvieli a P. Chen, „Entity-Relationship Modeling and Fuzzy Databases“ *Proceedings of the Second International Conference on Data Engineering,* pp. 320-327, 1986.

[3] Chen a Kerre, „Extending ER/EER concepts towards fuzzy conceptual“ *Proceedings of the 1998 IEEE International Conference on Fuzzy Systems,* pp. 1320-1325.

[4] Kerre a Chen, „An overview of Fuzzy Data Models“ *Fuzzyness in Database Management Systems,* pp. 23-41, 1995.

[5] P. Bosc, M. Galibourg a G. Hammond, „Fuzzy Querying with SQL: Extensions and Implementation Aspects“ *IRISA/INRIA,* pp. 333-349, 1987.

[6] A. G. Touzi a M. A. B. Hassine, „New Architecture of Fuzzy Database Management Systems“ *The International Arab Journal of Information Technology,* sv. 6, č. 3, pp. 213-221, 2009.

Přílohy

1. Program SIFQ – Uživatelská dokumentace

1.1. Úvod

Toto je uživatelská dokumentace programu SIFQ (Simple Interface for Fuzzy Querying), který slouží pro dotazování nad fuzzyfikovanou databází. Vzhledem k důrazu na jednoduchost se jedná o referenční příručku. Pro více informací o fungování slouží programátorská dokumentace a diplomová práce, přiložené na CD. Aplikace je implementována pomocí technologie WPF (Windows Presentation Foundation). Jedná se tedy o okenní aplikaci skládající se z hlavního okna a dalších pomocných oken.

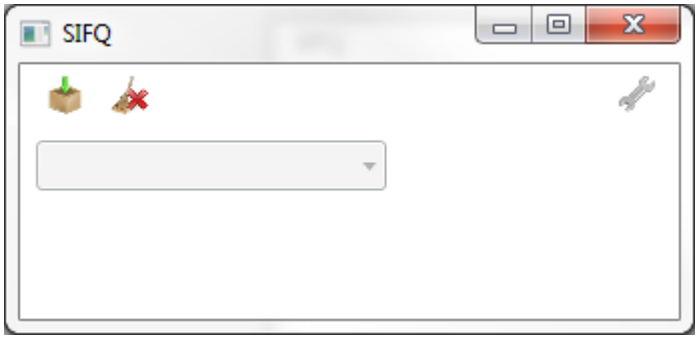
Aplikace je implementována v jazyku C#. Proto je nutné, aby na stroji, kde bude spuštěna, byl nainstalován framework .NET verze 4.0. Podporovaný operační systém je Microsoft Windows XP a vyšší. Program byl testován na OS Microsoft Windows 7. Minimální požadavky na hardware odpovídají požadavkům na běh OS.

Pro správné fungování aplikace je nutné mít přístup ke kompatibilní databázi (struktura databáze je podrobně popsána v diplomové práci). Pro nasazení databáze je použit Microsoft SQL Server 2005. Pro testovací účely je možné použít databázi SFD (Simple Fuzzy Database). Jednotlivé kroky pro její nasazení jsou:

- Vytvoření databáze pomocí skriptu *GenerateSFD.sql* na přiloženém CD na Microsoft SQL Serveru. Například za použití Microsoft SQL Management Studia (součást CD – Aplikace třetích stran).
- Vygenerování náhodných dat pomocí programu Generator. Je nutné správně nastavit připojení k vytvořené databázi (položka *změna současného nastavení*).
- Nastavení správného řetězce pro připojení k databázi v nastavení software (sekce 1.6).

1.2. Hlavní okno aplikace

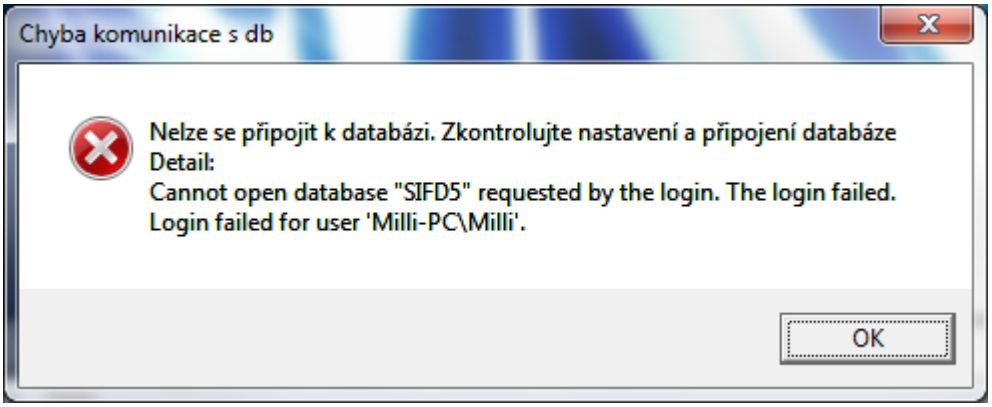
Hlavní okno aplikace (Obrázek 52) je zobrazeno po spuštění pomocí odpovídajícího souboru (běžně SIFQ.exe).



Obrázek 52 - Hlavní okno aplikace po startu

Ihned po startu jsou základními ovládacími prvky:

- Tlačítko 
  Načtení informací o databázi a provedení automatických kroků (vytvoření konfliktních tabulek,...). V případě, že načtení proběhne úspěšně, je možné pokračovat v dalších krocích (sekce 1.3). Pokud není možné se k databázi připojit nebo nastane jiný problém, je zobrazeno chybové hlášení (Obrázek 53).

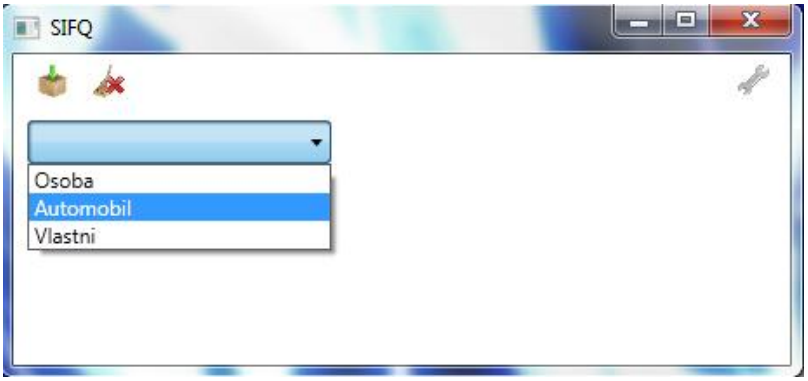


Obrázek 53 - Chyba komunikace s databází

- Tlačítko 
  Po stisku tohoto tlačítka je otevřen formulář pro nastavení parametrů aplikace. Podrobné informace o nastavení popisuje kapitola 1.6.
- Tlačítko 
  Slouží pro návrat aplikace do původního stavu (po startu). Jsou ztraceny veškeré změny, provedené uživatelem.

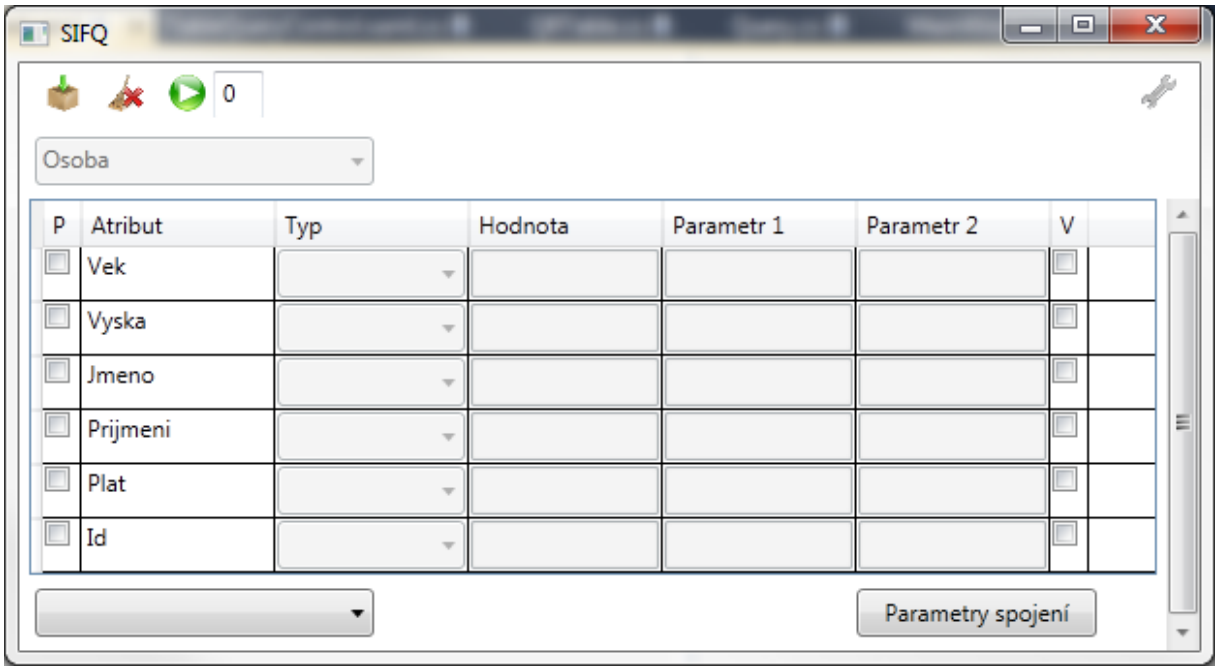
1.3. Výběr tabulek a atributů

Po úspěšném načtení informací o struktuře databáze je povolen ovládací prvek pro výběr tabulky (Obrázek 54). Pomocí tohoto ovládacího prvku je možné vybrat tabulku, na kterou se chce uživatel dotazovat.



Obrázek 54 - Výběr tabulky

Po výběru požadované tabulky v databázi je zobrazena soupiska atributů formou tabulky Obrázek 55).



Obrázek 55 - Tabulka Osoba

Jednotlivé řádky tabulky reprezentují veškeré atributy tabulky, která je vybrána. Položky pro vyplnění není možné po počátečním zobrazení měnit a jejich stav se mění na základě vybraného typu atributu. Součástí tabulky jsou:

- Sloupec *P*
  Po zaškrtnutí položky v tomto sloupci je odemčena možnost výběru typu dotazu nad atributem. V rámci dotazování to znamená, že uživatel chce tento atribut použít při tvorbě dotazů jako jeden z parametrů. Po zaškrtnutí je automaticky zaškrtnuto políčko ve sloupci *V*.
- Sloupec *Typ*
  Slouží k výběru typu fuzzy funkce a případně běžného dotazu na konkrétní hodnotu. Při zadání dotazu na přesnou hodnotu je odemčen pro změny sloupec Hodnota. Pokud je vybrána funkce přesnosti, jsou odemčeny sloupce Parametr 1 a Parametr 2.
- Sloupec *Hodnota*
  Odemčen pokud je dotaz v rámci aktuálního atributu zvolen jako běžný dotaz (>, <, =). Jeho vyplněná hodnota odpovídá druhé části podmínky. Pokud bychom chtěli získat osoby starší 20ti let, stačí ve sloupci *Typ* vybrat znak > a do sloupce *Hodnota* zadat číslo 20.
- Sloupce *Parametr 1* a *Parametr 2*
  Odemčeny pouze pro funkce přesnosti. Reprezentují parametry této funkce. *Parametr 1* odpovídá hledané hodnotě a *Parametr 2* její přesnosti. Hledáme-li například osobu s výškou 165 cm a nejsme si zcela jistí touto výškou, mohou být data vyplněna následovně: *Parametr 1* = 165, *Parametr 2* = 0.85. hodnoty druhého parametru musí být z intervalu (0,1).

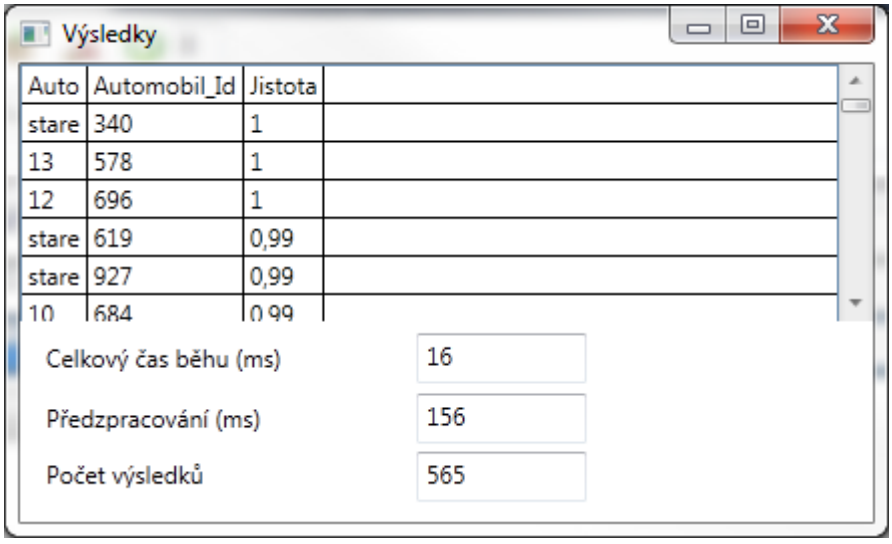
- Sloupec *V*
  Zaškrtnutí u atributu znamená, že mají být hodnoty atributu zobrazeny u výsledných záznamů.

Pod tabulkou jsou obsaženy další ovládací prvky:

- Ovládací prvek pro výběr další tabulky
  Slouží pro přidání další tabulky do výsledku. Pokud není specifikováno jinak, jedná se o křížové spojení.
- Tlačítko *Parametry spojení*
  Nastavení spojení s předchozími tabulkami. Jak spojení nastavit popisuje sekce 1.5.

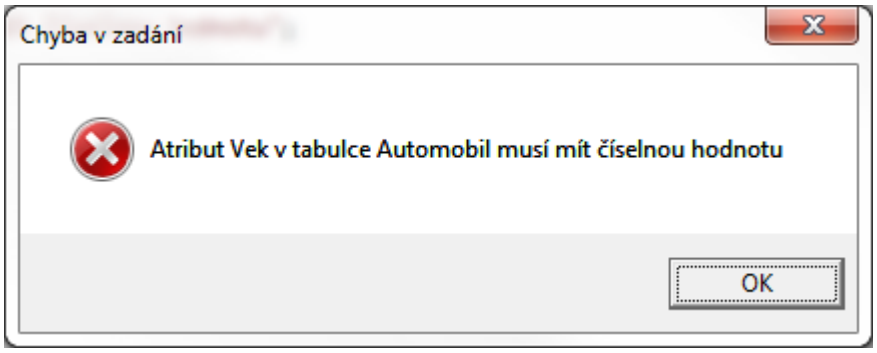
1.4. Spuštění výpočtu

Po nastavení parametrů pro dotazování je možné spustit výpočet tlačítkem . Toto tlačítko je zobrazeno až po výběru první tabulky. Dále je možné nastavit minimální přesnost výsledných záznamů v poli napravo od spouštěcího tlačítka (formát s desetinnou tečkou). V případě, že byl výpočet úspěšný, jsou v závislosti na nastavení buď zobrazeny výsledky v samostatném okně (Obrázek 56) nebo jsou vypsány do souboru *Output.txt*.



Obrázek 56 - Zobrazení výsledků

Pokud byl zadán dotaz uživatelem chybně, je zobrazeno chybové hlášení s detaily problému (Obrázek 57).



Obrázek 57 - Chybné zadání

1.5. Spojení tabulek

Tento formulář (Obrázek 58) je zobrazen v případě, že chce uživatel použít jiné spojení než křížové. Vybranou tabulku lze spojit pouze s již vybranými tabulkami.

Obrázek 58 - Spojení tabulky Vlastní s tabulkou Osoba

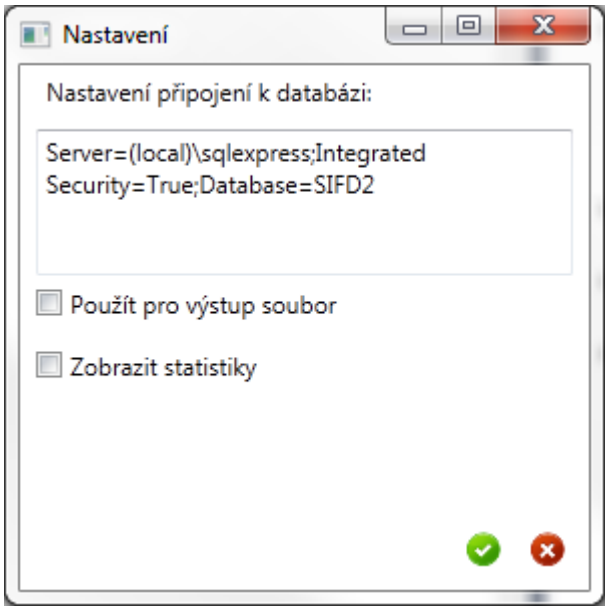
Hlavními ovládacími prvky jsou:

- Tlačítko 
  Přidání podmínky pro spojení. Podmínek může být více.
- Tlačítko 
  Potvrzení nastavených podmínek. Po stisku tohoto tlačítka jsou podmínky nastaveny a uloženy.
- Tlačítko 
  Zavření formuláře. Poslední změny nejsou uloženy.

V rámci jedné podmínky je nutné nastavit atribut, přes který je tabulka spojena a dále pak vybrat tabulku a její odpovídající atribut. Spojení je na rovnost. Na příkladu, který popisuje Obrázek 58 jsou spojeny tabulky *Osoba* a *Vlastní*. Atribut *Osoba_Id* musí být shodný s *Id* osoby.

1.6. Nastavení aplikace

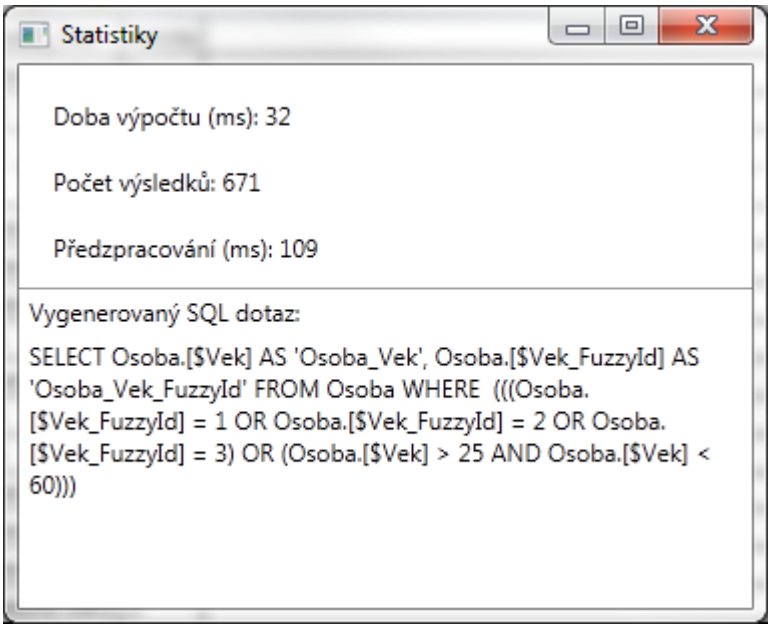
Nastavení aplikace je reprezentováno samostatným formulářem (Obrázek 59). Změny nastavení jsou uloženy a zůstávají nezměněny i po vypnutí aplikace.



Obrázek 59 - Nastavení aplikace

Jednotlivými ovládacími prvky jsou:

- Pole pro vyplnění připojení k databázi
  Nastavení řetězce použitého pro připojení k databázi.
- Pole *Použít pro výstup soubor*
  Uložení výstupu do souboru *Output.txt* místo běžného zobrazení v samostatném okně. Tento přístup je rychlejší než zobrazování a je doporučen použít především v případě, kdy je výsledků velké množství (více než 100000) a hrozí vyčerpání operační paměti.
- Pole *Zobrazit statistiky*
  Po výpočtu jsou zobrazeny informace o délce běhu jednotlivých částí výpočtu (Obrázek 60).



Obrázek 60 - Statistiky výpočtu

- Tlačítko 
  Potvrzení změn provedených uživatelem, jejich uložení a zavření okna.
- Tlačítko 
  Zavření okna, změny nejsou uloženy.