

P114

Datové modelování I

(základní kurs)

P114

Úvodní přednáška

1

Obsah

- představení, očekávání (*)
- cíle předmětu
- literatura
- požadavky ZK
- Motivace
- Souvislost s řízením projektů zavádění IS
- Přehled předmětu

Cíle předmětu

- vyložit základní pojmy DM a naučit metody tvorby těchto modelů tak, aby posluchači byli schopni rozpoznat kvalitní datový model, samostatně popsat danou oblast datovým modelem a použít DM pro porozumění neznámé oblasti
- přesvědčit, že DM je základem dobrého SW inženýrství

Studijní materiály

- přednášky + podklady k přednáškám
- Z.Staniček: Datové modelování metodou HIT, minikurs, Sborník konference DATASEM '96, vyd. CS-COMPEX a.s., 1996
- Školící materiály fy SHINE Consulting s.r.o. - případové studie datových modelů, SHINE studio 1999
- M. Duží: Konceptuální modelování - datový model HIT, Slezská universita v Opavě, FPF Ústav informatiky, 2000, (skripta)
- / Materna, Pala, Zlatuška: Logická analýza přirozeného jazyka, Academia, Praha, 1989
- / LBMS Systems Engineering, verze CZ 2.0, vyd. LBMS, 1995
- / Pokorný, Halaška: Databázové systémy, vyd. ČVUT Praha, 1998

Požadavky ke ZK

- Odevzdání semestrální práce (konkrétní DM) nahrazující cvičení
- 50% práce
(tvorba DM)
- 50% test
(cokoli z odpřednášeného, ale nic jiného)

Motivace k předmětu

- **Jde o projekty zavádění IS**
- Gartner group: 12 - 15 - 30 - 43
- diletantismus dodavatelů („lékaři bez ZK z anatomie“)
- kultura komunikací: umět popsat co děláme a co pro to potřebujeme
- nástroj pro porozumění novým/cizím oblastem ...
- problém „zadání úkolu“ - CO se vlastně má udělat ?

Řízení projektů zavádění IS

- Plány: **CO** - JAK - SKYM - **KDY** - **ZAKOLIK**
- **CO** se má udělat = specifikace provedení
 - jaké informace bude IS poskytovat = vymezení třídy dotazů nad daným IS zodpověditelných
 - jak bude informace poskytovat = pomocí jakých funkcí
 - za jakých okolností bude IS informace poskytovat = v rámci jakých procesů se to bude dít
 - komu bude jaké informace poskytovat = organizace přístupu k informacím

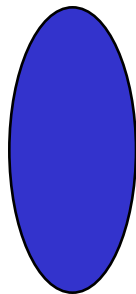
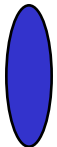
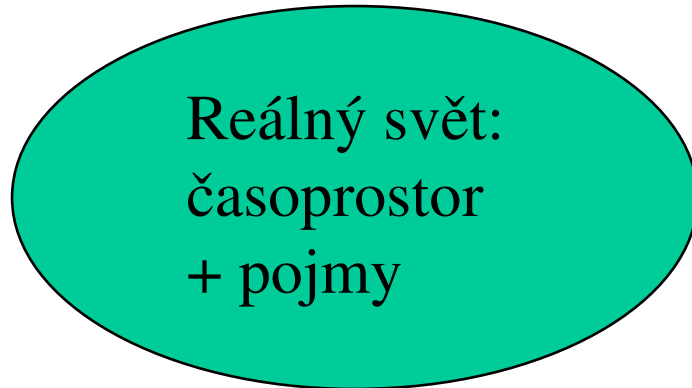
jaké informace bude IS poskytovat

- jak zadat třídu dotazů, které mají být daným IS zodpověditelné
- vytvořit seznam všech takových dotazů?
- určit „bázi“ prostoru dotazů, které mají být zodpověditelné
- z této „báze“ se pak seznam všech zodpověditelných dotazů dá vygenerovat
- „báze“ prostoru dotazů = Datový Model

Informační schopnost IS

- je dána prostorem dotazů zodpověditelných daným IS
- prostor je určen „bází“, tj. datovým modelem (DM) (? datovým schématem)
- IS zobrazuje reálný svět proto, abychom se dokázali domluvit o jeho struktuře a o jeho chování - abychom mohli komunikovat poznání jeho stavu

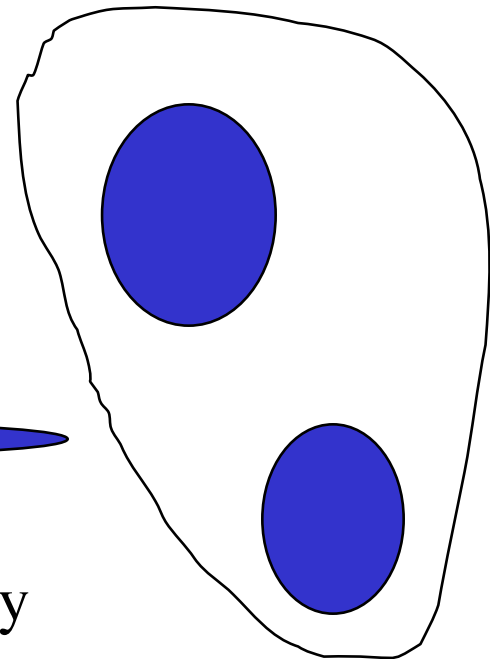
Reálný svět, ideální světy, kultura



Ideální světy

P114_1

Kultura



CO poznáváme, PROČ poznáváme

- stav reálného světa (PROCESY)
- tj. dynamické systémy
- chceme porozumět, dokázat předpovědět
- základní potřeba: komunikovat
 - s druhými = domlouvat se
 - se sebou = přemýšlení

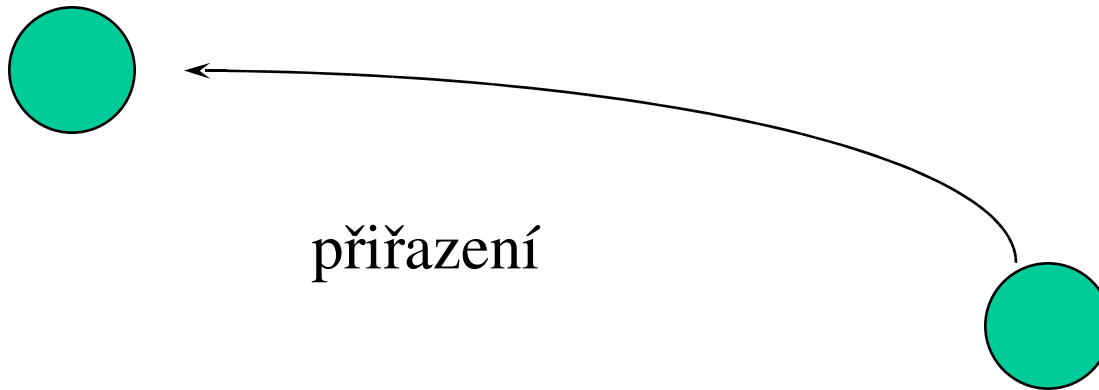
JAK poznáváme

- vždy pomocí modelů
- model dynamického systému
 - jiný dynamický systém (v reálném světě)
 - pomocí pojmů a představ (v ideálním světě)
- pojem: konstrukce (dohoda na významu)
- pojmový (konceptuální) model
- model v ideálním světě (ideální model)
- IDM = konceptuální (datový) model

Co se děje v našich hlavách?

- každý tam máme svůj svět představ, svůj ideální svět
- dynamické systémy reálného světa mapujeme statickými pojmovými modely
- vytváříme základní modely pozorováním
- odvozujeme sekundární modely přemýšlením

Elementární konstrukt



Přiřazení - zobrazení - funkce:

je to předpis (procedura), který říká jaký výsledek (výstup) je přiřazen k danému vstupu

O čem jsou ideální světy

- procesy (vnímáme z reálného světa)
- věci (vyabstrahujeme se na čas)
- události (navnímané koincidence procesů)
- kontejnery (sbalení do vyššího celku za účelem pořádání)
- objekty ideálního modelu

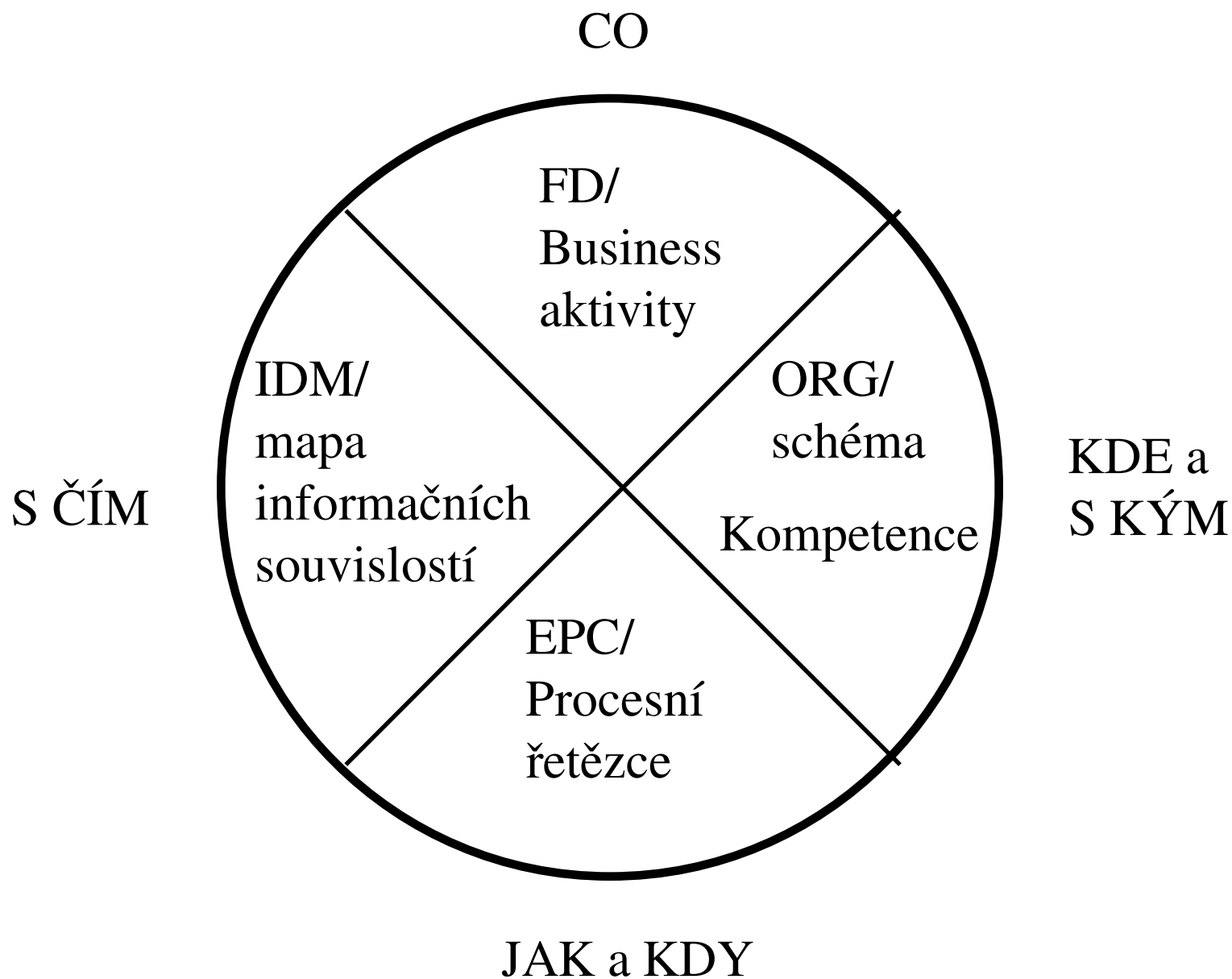
O čem je komunikace?

- o objektech reálného světa
(časoprostorového resp. světa pojmů)
- o souvislostech těchto objektů
(tj. o strukturách čili složitějších objektech)
- o chování těchto struktur

Jaké máme ke komunikaci nástroje?

- přirozený jazyk
- grafiku (obrázky)
- jiné ...

- Datové modely
obrázky (schémata) + přirozený jazyk
- BPM (Business Process Model)



BPM a DM - shrnutí

- BPM:
 - pojmové hřiště (IDM)
 - schopnosti, dovednosti („funkce“ - FD)
 - události a procesy (EPC)
 - organizace (ORG)
- pojmový model
 - objekty a souvislosti
 - proč říkáme „datové modelování“
- DM jako technika pro konstrukci BPM
 - všechno to jsou „objekty a souvislosti“

Datové Modelování, data a metadata

- DM a jeho vztah k datům a metadatům ?
- interpretace dat: přiřazení sémantiky
 - programem
 - pomocí metadat
- co víme, když známe data? ()
- co víme když známe metadata? (+)
- DM je nástroj pro vytvoření takových metadat, že data jimi interpretovaná poskytují odpovědi na požadované dotazy

Přehled předmětu

1

- 1 Úvod
- 2 Klasické metody modelování
- 3 Funkcionální přístup - základní intuice
- 4 Logické základy (TIL)
- 5 Konstrukce
- 6 Sémantika a její role

Přehled předmětu

2

- 7 Pojmy metody HIT
- 8 Postup tvorby modelu
- 9 Definovatelnost
- 10 Rozložitelnost
- 11 Doladění a transformace do ERD
- 12 Příklady datových modelů

P114

Klasické metody modelování

RDM, ERAM

2

Témata

- modelování v RDM
- univerzální relace
- dekompozice, normalizace
- syntéza relací
- omezení DM v RDM
- modelování v ERAM
- notace, postup
- špatně a správně utvořené modely

Cílem je implementace:

- Co máme na počítači?
 - soubory / tabulky
 - záznamy / řádky
 - položky / sloupce,
 - klíče
 - co se jak implementuje
(příklad tabulek Rozvrhu)

ROZVRH

UCITEL

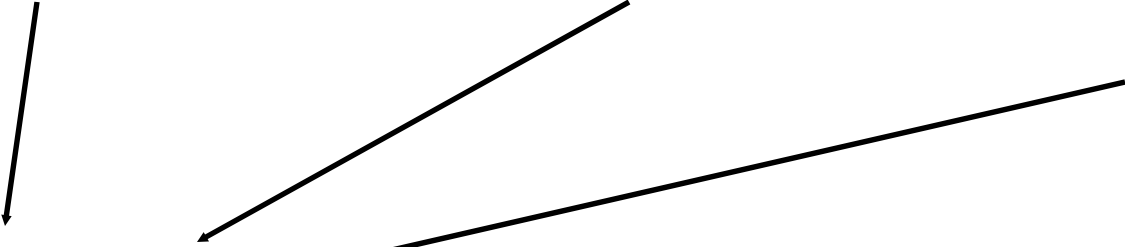
#UC		

PREDMET

#PR		

MISTNOST

#MI		



#UC	#PR	#MI	CAS

- popisy objektů
- zachycení souvislostí
- použití klíčů

Modelování v RDM

- definice relace, schématu relace
- klíče, primární, alternativní, cizí, referenční integrita
- relační schéma (databáze)
- funkční závislosti, Armstrongova pravidla
- objekty a vztahy jako relace

OPAKOVÁNÍ:

definice relace, schématu relace

- R je subset $D_1 \times \dots \times D_n$
- $R(A_1:D_1, \dots, A_n:D_n), D_i = \text{dom}(A_i)$

schéma relace

$\mathbf{A} = \{A_1:D_1, \dots, A_n:D_n\}$ množina atributů relace R ,

$R(\mathbf{A})$ jiný zápis schématu relace R nad množinou atributů \mathbf{A}

- schéma relace = záhlaví tabulky

n -tice = řádek tabulky

- rozdíly:

tabulka má vždy nějaké pořadí řádků a sloupců

tabulka může mít duplicitní řádky

OPAKOVÁNÍ: projekce

- projekce n -tice na podmnožinu atributů:
 B je subset A , u je n -tice z R : $u[B]$ je projekce
(k -tice s komponentami z B)
- projekce relace na podmnožinu atributů:
je to projekce všech n -tic z R na podmnožinu atributů B :
 $R[B]$

OPAKOVÁNÍ: klíče, primární, alternativní, cizí, referenční integrita

- klíč K relace $R(\mathbf{A})$: $K \subseteq \mathbf{A}$, u, v jsou z R různé: $u[K] \neq v[K]$, if K' je subset \mathbf{A} a má tutéž vlastnost jako K , pak K' obsahuje K
- kandidáti na prim. klíč, primární klíč - jeden zvolený, ostatní kandidáti: alternativní klíče
- jednoduché a složené klíče
- cizí klíč $C_K :=$ skupina atributů, která je primárním klíčem K jiné relace
- referenční integrita: $R_2[C_K]$ je subset $R_1[K]$

OPAKOVÁNÍ: relační schéma (databáze)

- **RSD** := (**R**,**I**), kde **R** = { R_1, \dots, R_m }, **I** je množina IO (logických podmínek, které musí data v DB splňovat)
- lokální IO: omezují data v jednom schématu relace
- globální IO: dávají vazby mezi daty různých schémat relací
- přípustná relační databáze R se schématem (**R**,**I**)
- stav databáze R

OPAKOVÁNÍ:

funkční závislosti, Armstrongova pravidla

- funkční závislost je vztah mezi daty v „tabulkách“
- funkční závislost je druhem IO
- B, C jsou subsety A : $B \rightarrow C$ jestliže pro libovolné n -tice $u, v \in R$ platí if $u[B] = v[B]$ then $u[C] = v[C]$
- X, Y, Z jsou subsety A . Potom:
 - if $Y \subseteq X$ then $X \rightarrow Y$ (triviální závislost) (AP1)
 - if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$ (AP2)
 - if $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$ (AP3)
 - if $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$ (AP4)

OPAKOVÁNÍ: normální formy

- 1NF: domény obsahují pouze atomické prvky (nikoli znovu relace)
- 2NF: 1NF + neexistují parciální fční závislosti na klíči
- 3NF: 2NF + neexistují transitivní funkční závislosti (C tranzitivně závisí na X: $X \rightarrow Y \rightarrow C$ a $C \notin X, C \notin Y$, a $Y \not\rightarrow X$)
- BCNF pro každou netriviální závislost $X \rightarrow Y$ platí X obsahuje klíč schématu relace R

OPAKOVÁNÍ:

dekompozice, normalizace, syntéza

- (pragmatické) důvody pro zavedení xNF: aktualizací anomálie
- normalizace pomocí dekompozice relačních schémat (použití AP4)
- konstrukce relačního schématu syntézou (použitím funkčních závislostí - AP3)

Univerzální relace

- modelování dekompozicí univerzální relace
- předpoklad schématu univerzální relace:
jednoznačnost jmen atributů
- předpoklad jednoznačnosti vztahů
 - protipříklad:
Ved_diplomky(Učitel, Student)
Učí (Učitel, Předmět, Student)
nelze získat z jednoho schématu univerzální relace

Omezení DM v RDM

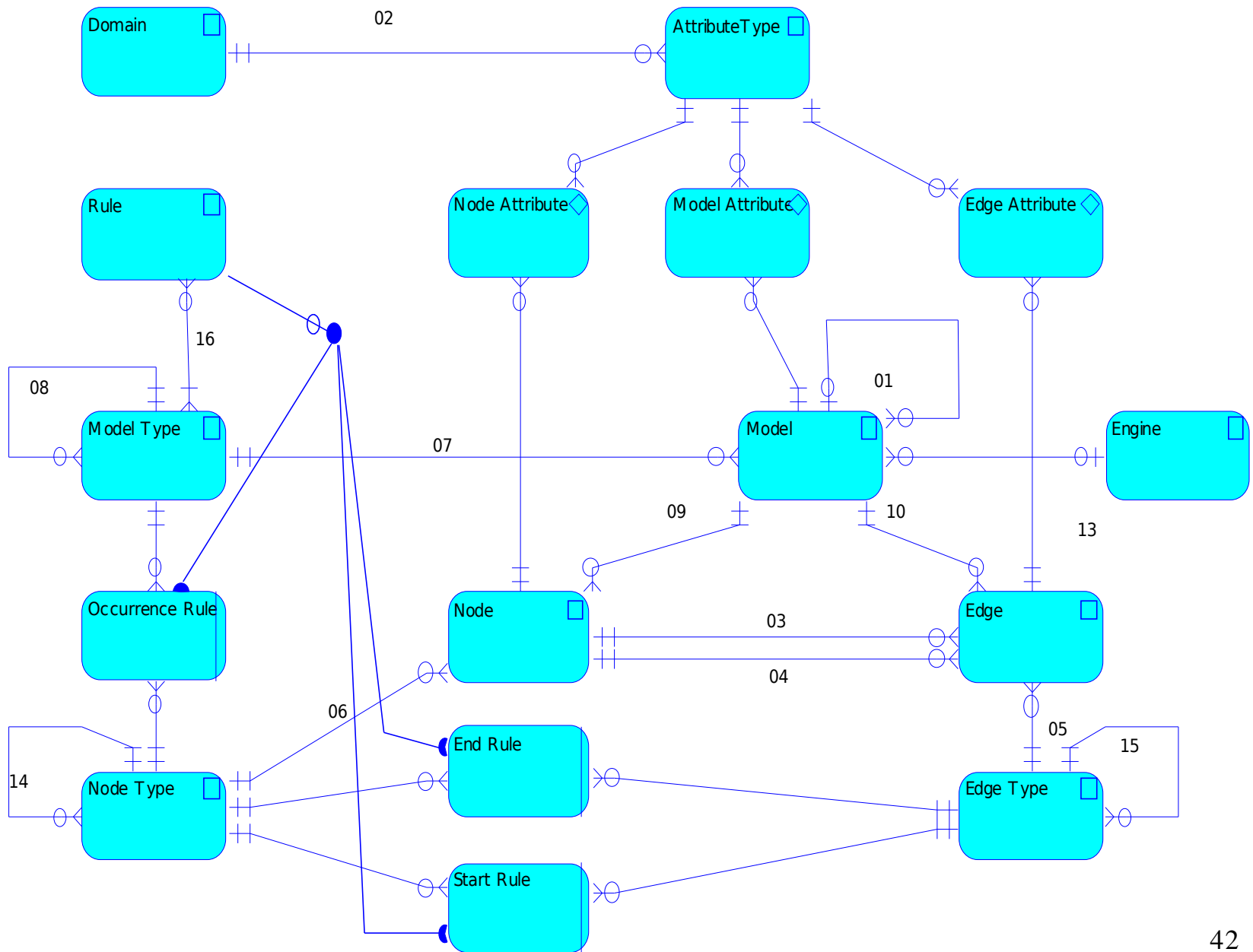
- praktická tvorba datového modelu v RDM
 - kombinace dekompozice a syntézy
- absence sémantiky ve formálním aparátu
 - hra symbolů jejichž interpretace leží „za hranicemi systému“
- nedostatečnost PL1 pro analýzu přirozeného jazyka
 - potřebujeme v jednom systému pracovat s objekty různých řádů
- přílišná formalizace snižující využitelnost intuice
 - o čem se vyjadřujeme, versus
 - o čem skutečně přemýšlíme
- nereálné předpoklady a jejich obcházení „krokem stranou“
 - předpoklad jednoznačnosti vztahů

Modelování v ERAM

- objekt -- kontejnerem je „entita“, „typ entity“
- vztah -- kontejnerem je „typ vztahu“
- atribut (typu entity nebo typu vztahu) -- je funkce přiřazující hodnoty popisných typů
- IO --soulad schématu s modelovanou realitou
- diagram typů entit a vztahů ERD
- kardinalita vztahu, členství ve vztahu
- ISA vztah

Notace (nástroje)

- entita
 - silná, popisná, vazební
 - klíče (primární, alternativní, cizí, nejednoznačné)
- vazba
 - maximální kardinalita, minimální kardinalita
 - role MASTER, DETAIL
 - pojmenování vazeb
- podtypy entity
 - definice podtypu, dědění
 - skupiny disjunktních podtypů s úplným nebo částečným pokrytím



Postup (kroky)

- identifikuj entity
- urči vazby (mřížka entit)
- vytvoř model: doplň diagram + kardinality
- odstraň redundance přístupu a duplicitní vazby
- modeluj podtypy entit a uprav vazby
 - totalita atributů, parcialita vazeb, rozdělení do kategorií
 - stav entity, rekursivní vazby, vazba typu kusovník
- Příklad: Rozvrh

„Nesprávnost“ modelu, příklady

- diagram bez sémantiky
- vztahy jsou v algoritmech a ne ve vazbách
 - Pojišťovací systém (rozvoje-schopnost „-“)
 - SELECT SE (procesy) (důsledek pro slévání submodelů)
- absence nadtypu a tím velká složitost vazeb
 - Údržba v REAS (velká složitost vazeb)
 - Pojišťovací systém (nekonzistentnost informací o partnerech)
- konečný počet podtypů
 - a algoritmů jejich zpracování (problém přidání podtypu, změny klasifikace)

„Správný“ model, příklady

- co je to „správný model“
 - pozor na „reálný svět“ vs „svět představ“
 - adekvátnost požadavkům (i nevyjádřeným)
 - úloha analytika - „datového modeláře“
 - text může být blábol ze špatně utvořených vět
 - model může být špatně konstruován z nedobře definovaných konstruktů
- „správný“ model:
 - IS Bílý Motýl
 - transakční systém EXPANDIA Banky

P114

Funkcionální přístup

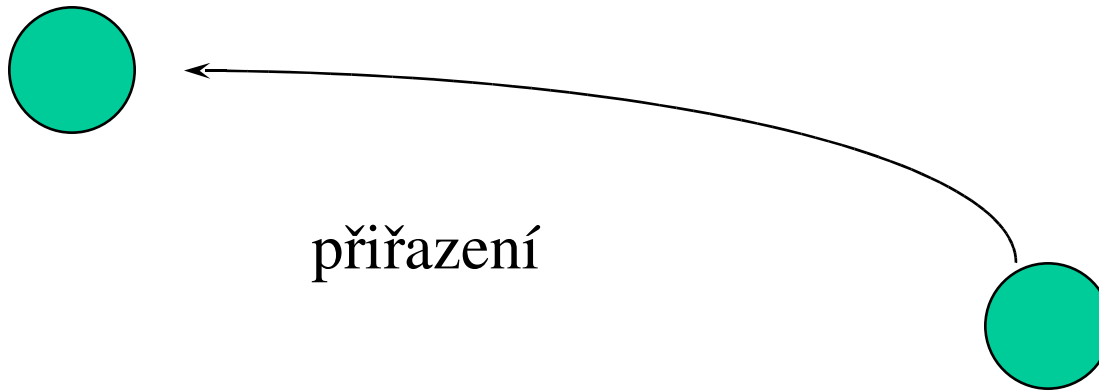
Základní intuice

3

Témata

- funkce jako „výpočetní“ procedura
- funkce nezávislé na stavu světa: deskripce, analytické funkce
- funkce závislé na stavu světa: entity, datové funkce
- funkce přiřazující entitám deskripce nebo entity
- objekty: extenze, intenze, funkce ...
- objekty - výrazy - pojmy
- pojmy jako identifikační procedury
- základní princip komunikace

Elementární konstrukt



Přiřazení - zobrazení - funkce:

je to předpis = procedura, který říká jaký výsledek (výstup) je přiřazen k danému vstupu

funkce (filozoficky)

- bazální pojem: pojem funkce
- skoro vše, o čem přemýšlíme jsou funkce (až na P,N, časové okamžiky, individua)
- způsob, jak přemýšlíme: něco něčemu přiřazujeme, tj. základním konstruktem přemýšlení je funkce
- jsme zvyklí jednotliviny, v něčem si podobné, „sypat“ do samostatných kontejnerů a přemýšlet a vyjadřovat se pomocí těchto kontejnerů = typů

elementární pojmy (množinově)

Uspořádaná dvojice prvků a, b :	$\langle a, b \rangle$	$\{\{a\}, \{a, b\}\}$
Kartézský součin množin A, B :	$A \times B$	$\{\langle a, b \rangle \mid a \in A, b \in B\}$
Korespondence množin A, B :	$f, fa \rightarrow b$	$f \subseteq A \times B$
Definiční obor f	Df	$Df = \{a \in A \mid \exists b \in B: fa \rightarrow b\}$
Obor hodnot f	Rf	$Rf = \{b \in B \mid \exists a \in A: fa \rightarrow b\}$

funkce (množinově)

- Parciální funkce z množiny A do množiny B :
 $f, f(a) = b$
 $\forall a, b_1, b_2: ((f(a) = b_1 \wedge f(a) = b_2) \Rightarrow b_1 = b_2)$
- Totální funkce z množiny A do množiny B :
 $f: A \rightarrow B$
 $Df = A$
- Parciální funkce v užším smyslu:
 $f: A \rightarrow B$
 $Df \subsetneq A \wedge Df \neq A$

funkce více proměnných

- Kartézský součin množin M_1, M_2, \dots, M_n :

$$\prod_{(i=1..n)} M_i =$$

$$= M_1 \times M_2 \times \dots \times M_n =$$

$$= \{ \langle m_1, m_2, \dots, m_n \rangle \mid m_1 \in M_1, \dots, m_n \in M_n \}$$

- Funkce „více proměnných“:

$$f: \prod_{(i=1..n)} M_i \rightarrow M$$

$$\text{Df} \subseteq \prod_{(i=1..n)} M_i, \text{Rf} \subseteq M$$

funkce, ...

- n-ární funkce: složitost funkce je $n+1$
- funkce nedefinovaná na n -tici
- totožnost funkcí : princip extenzionality
 $F_1: M_1 \times \dots \times M_n \rightarrow M$, $F_2: M_1 \times \dots \times M_n \rightarrow M$,
 $X \in M_1 \times \dots \times M_n$, $Y \in M$ -- libovolné :
 $F_1(X) = Y$ *právě když* $F_2(X) = Y$, píšeme $F_1 = F_2$
- důsledek:
je-li $F_1 = F_2$ a Z libovolný prvek z $M_1 \times \dots \times M_n$, pak F_1 je na Z nedefinováno právě tehdy, když F_2 je na Z nedefinováno

funkce jako procedura

- procesní pohled
- (parciální) funkce je „výpočetní“ resp. „vyhodnocovací“ pravidlo/procedura, které poskytne buď *nic* nebo *výsledek* z množiny M , jestliže jí na vstupu zadáme hodnoty parametrů z $M_1 \times \dots \times M_n$
- *deklar* $x_1, \dots, x_n / M_1, \dots, M_n$ *deklar* y / M (*tělo proc*)
- x_i ... formální parametry, y ... výsledek

funkce nezávislé na stavu světa (analytické funkce)

- tělo procedury je skutečný výpočet (algoritmus), který za všech okolností, tj. nezávisle na stavu světa, vypočítá při nahrazení formálních parametrů danými hodnotami výsledek
- Příklady: (každý ?) program na počítači, algoritmus, matematické funkce, logické funkce

deskripce

- deskripce jsou popisy něčeho pomocí nějakých hodnot
- množina hodnot, které používáme pro popis určitého typu, se nazývá deskriptivní sorta
- deskriptivní sorta „D“ jako funkce:
 $\text{Bool} = \{P, N\}$, $\text{Hodn} = \text{množina všech možných hodnot}$
 $D : \text{Hodn} \rightarrow \text{Bool}$
(charakteristická funkce množiny D)
- Příklady: dny v týdnu, měsíce v roce, prvočísla, ...

funkce závislé na stavu světa (datové či empirické funkce)

- tělo procedury vyhodnocuje výsledek náhrady formálních parametrů zadanými hodnotami různě podle toho, jaký je stav světa
- Prakticky: tělo procedury obsahuje tabulku $(x_1:M_1, \dots, x_n:M_n, y:M)$, jejíž konkrétní naplnění daty reprezentuje jistý stav světa,
- vyhodnocení = vyhledání řádku se zadanými hodnotami
- Příklady: „Plat daného Zaměstnance“, „Množství daného druhu Zboží dodané daným Dodavatelem danému Odběrateli“

Příklady:

- „Plat daného Zaměstnance“
- *deklar x/ZAM deklar y/PLAT (tělo procedury)*

ZAMESTNANEC	PLAT
Novák	9300
Horák	7800
Bílá	11000

Příklady:

- „Množství daného Zboží dodané daným Dodavatelem danému Odběrateli“
- *deklar* z/ZBOZI, d/DOD, o/ODB *deklar* m/MNOZSTVI
(*tělo procedury*)

ZBOZI	DOD	ODB	MNOZSTVI
...			
hrnecky	Keramo	MU	450
...			

Entity

- ZAMESTNANEC, DODAVATEL, ZBOZI
- je entita množina všech jednotlivin, které jsou její prvky ?
- právě teď: jeden zaměstnanec přibyl, dodavatel zkrachoval, jedno zboží se přestalo vyrábět a jiné-nové se objevilo
- Změnily se uvedené entity ?
- Entity jako funkce:
Entita : StavSvěta \rightarrow (Jednotliviny \rightarrow Bool)

Příklady

- Entita ZAMESTNANEC je funkce, která každému stavu světa přiřazuje množinu všech individuí (jednotlivin), které jsme ochotni pokládat za zaměstnance
- Entita DODAVATEL je funkce ...
- Entita ZBOZI je funkce ...

Funkce přiřazující entitám deskripce

- entita ZAM
 $\text{ZAM} : \text{StavySveta} \rightarrow (\text{Jednotliviny} \rightarrow \text{Bool})$
- deskripce PLAT
 $\text{PLAT} : \text{Hodn} \rightarrow \text{Bool}$
- $\text{PlatZam} : \text{ZAM} \rightarrow \text{PLAT}$
přiřazuje zaměstnanci jeho plat v závislosti na stavu světa
- tzv. popisné (deskriptivní) atributy
- Příklady: CisloVyrobru , AdresaDodavat , ...

Funkce přiřazující entitám entity

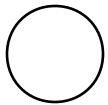
- entita ZBOZI
 $\text{ZBOZI} : \text{StavySveta} \rightarrow (\text{Jednotliviny} \rightarrow \text{Bool})$
- entita DOD
 $\text{DOD} : \text{StavySveta} \rightarrow (\text{Jednotliviny} \rightarrow \text{Bool})$
- $\text{DodZbozi} : \text{StavySveta} \rightarrow (\text{ZBOZI} \rightarrow \text{DOD})$
přiřazuje danému zboží jeho dodavatele v závislosti na stavu světa
- tzv. vztahové atributy
- Příklady („ $\text{StavySveta} \rightarrow$ “ vynecháváme):

Příklady

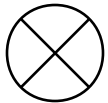
- $\text{OdbDodZbozi} : \text{DOD} \times \text{ZBOZI} \rightarrow \text{ODB}$
- přiřazuje dvojici (dodavatel, zboží) toho odběratele, kterému daný dodavatel dané zboží dodává
- (odpovídá to realitě ???)
- $\text{OdbDodZbozi} : \text{DOD} \times \text{ZBOZI} \rightarrow (\text{ODB} \rightarrow \text{Bool})$
- přiřazuje dvojici dodavatel zboží tu množinu odběratelů, kterým daný dodavatel dané zboží dodává
- (samozřejmě v závislosti na stavu světa)

Diagramy

- funkce reprezentující deskripci



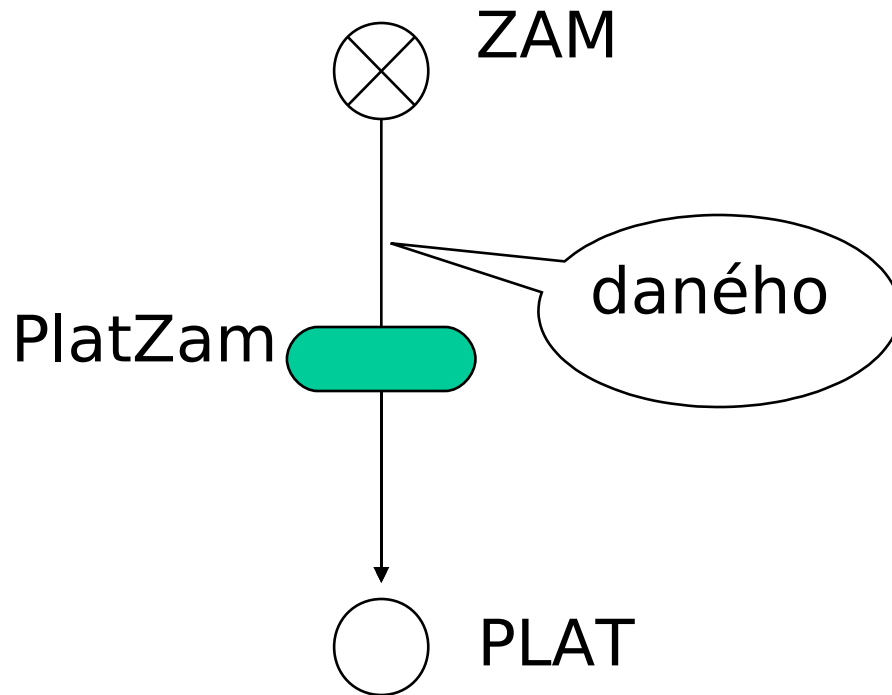
- funkce reprezentující entitu



- funkce reprezentující popisný resp. vztahový atribut (HIT-atribut, závislý na stavu světa)

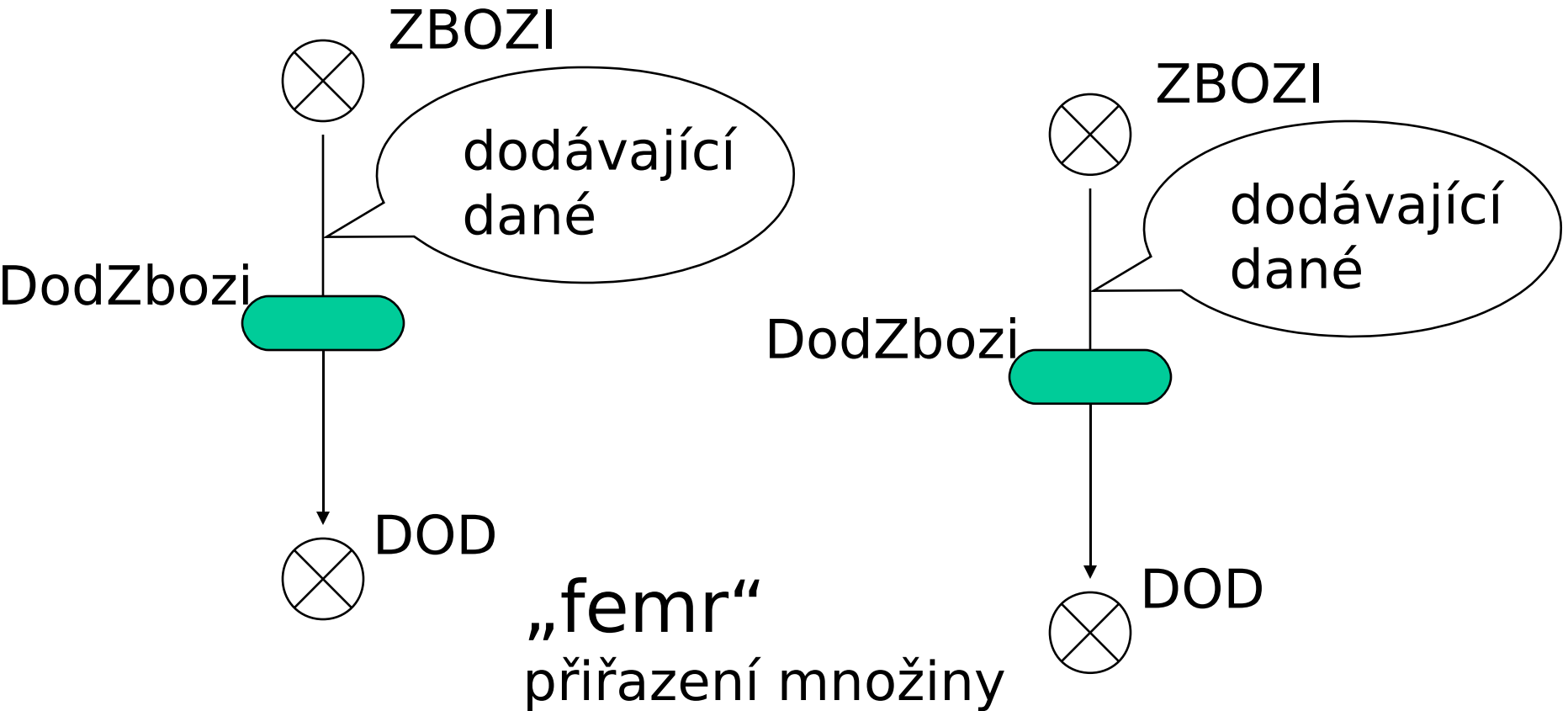


Popisný atribut

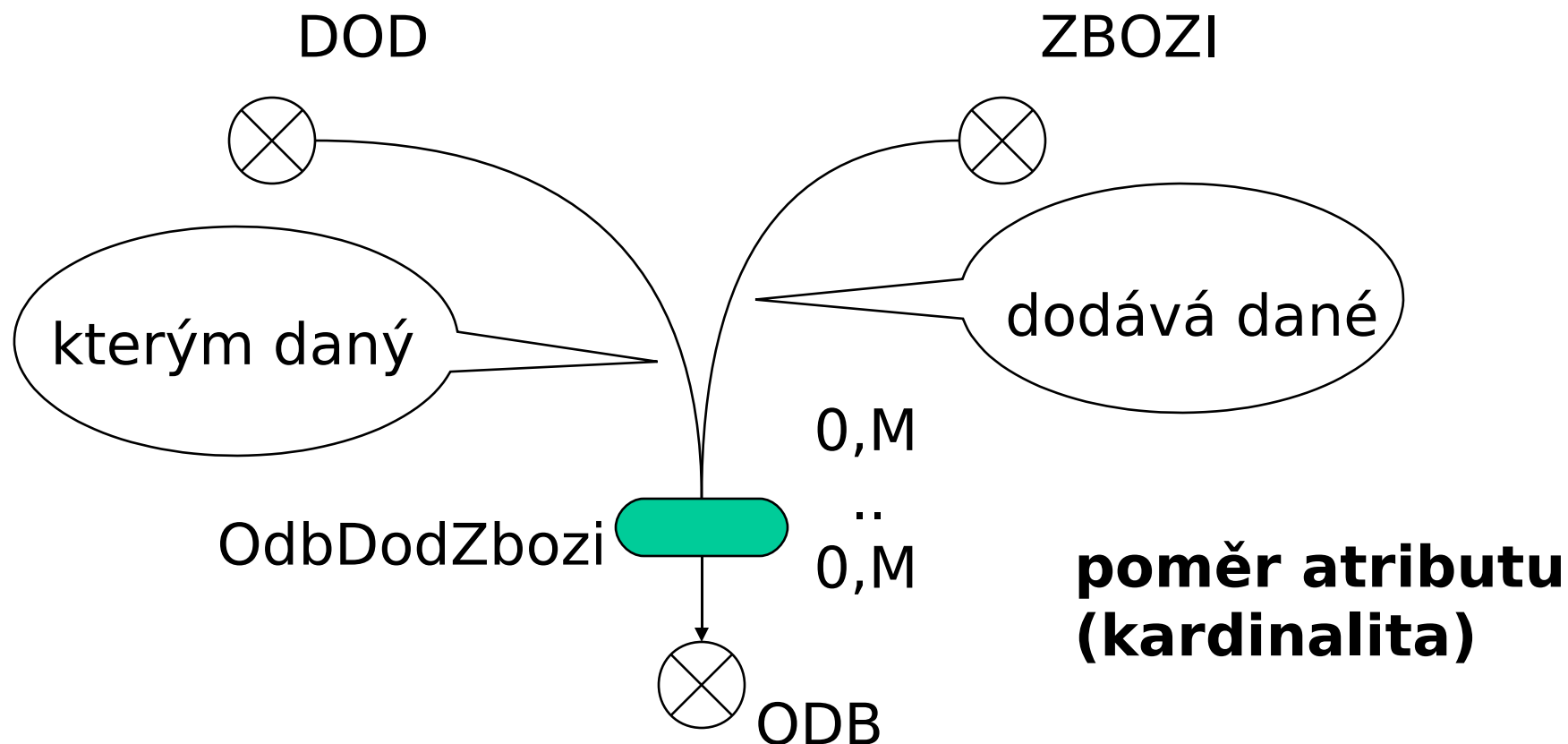


- typ hodnoty funkce
- typ argumentů
- typ samotné funkce
- role argumentů
- sémantika přiřazení

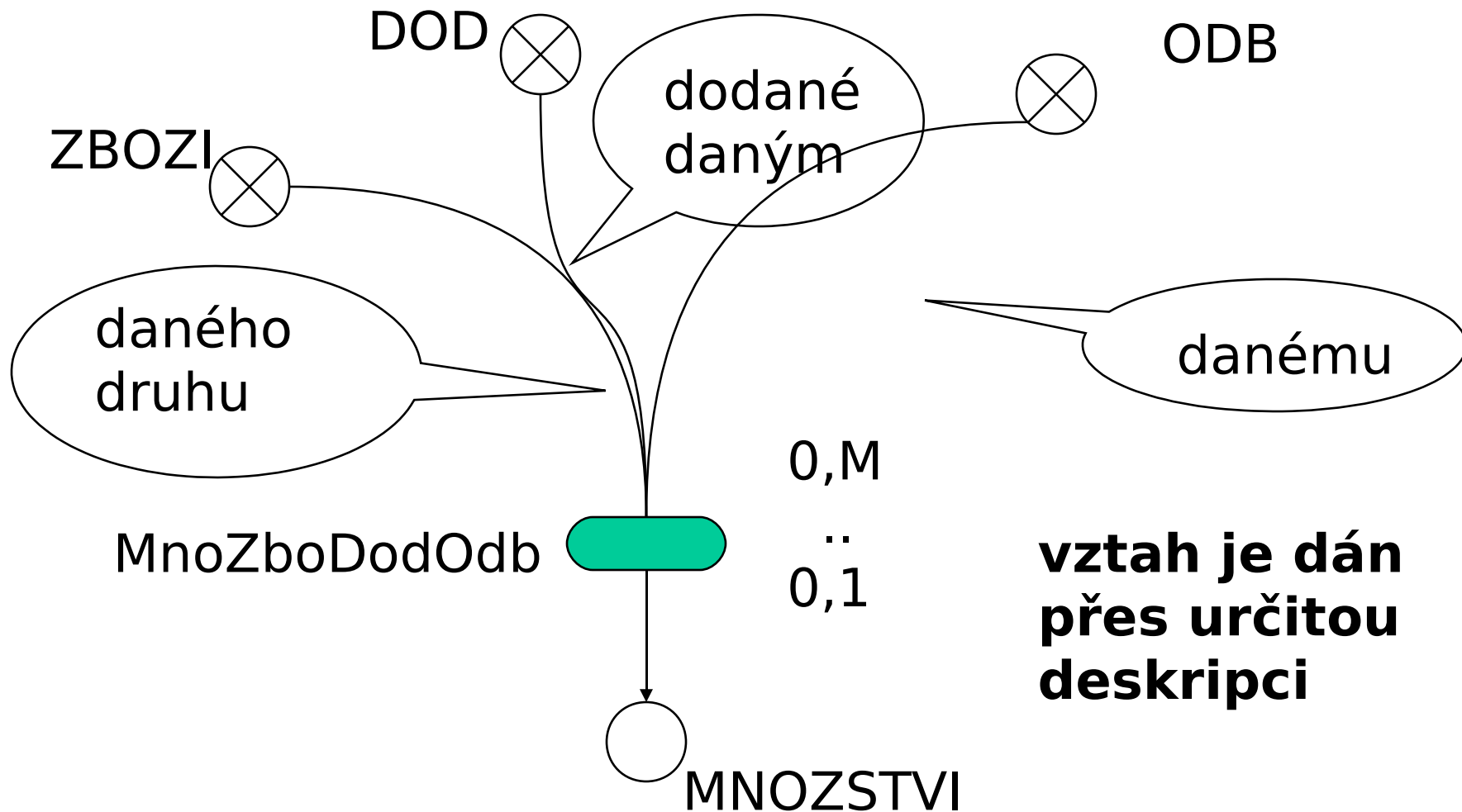
Vztahový atribut (složitosti 2)



Vztahový atribut (složitosti 3)



Vztahový atribut (složitosti 4)



Lineární zápis atributu

- $\text{PlatZam} =$
plat (PLAT) daného zaměstnance (ZAM)/1,1:0,M
- lépe vystihuje realitu:
 $\text{PlatZam} =$
plat (PLAT) daného zaměstnance (ZAM)/0,1:0,M
- $\text{DodZbozi} =$
dodavatelé (#DOD) dodávající dané zboží
(#ZBOZI)/0,M:1,M
- nebo může být výhodnější: /0,M:0,M

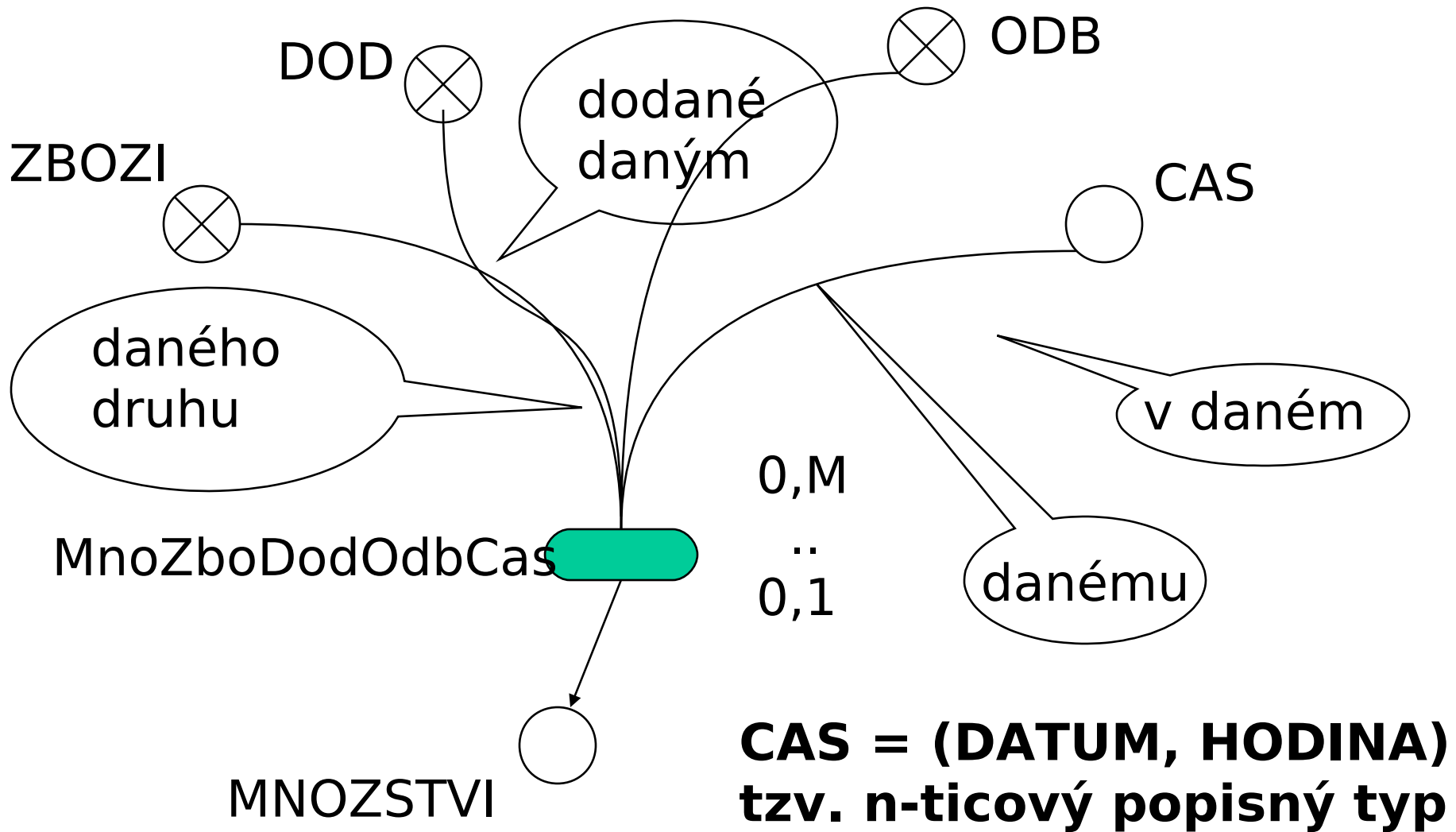
Lineární zápis atributu - pokračování

- OdbDodZbozi =
odběratelé (#ODB) kterým daný
dodavatel (#DOD) dodává dané zboží
(#ZBOZI)/0,M:0,M
- MnoZboDodOdb =
množství (MNOZSTVI) daného druhu
zboží (#ZBOZI) dodané daným
dodavatelem (#DOD) danému
odběrateli (#ODB) / 0,1:0,M
- (?) je takové množství skutečně jediné ...

úvahy nad správnou formulací atributu

- je jediné, pakliže se omezíme na zadané datum:
- **MnoZboDodOdbDat** =
množství (MNOZSTVI) daného druhu
zboží (#ZBOZI) dodané daným
dodavatelem (#DOD) danému
odběrateli (#ODB) v daném dni
(DATUM) / 0,1:0,M
- ... možná mohou proběhnout dvě i více
dodávek v jednom dni ...
- potom atribut **MnoZboDodOdbDat** musíme
nahradit následovně: P114_3

Složitejší atribut



Co jsou *objekty* našeho zájmu?

- jednotlivé hodnoty nějakých deskripcí nebo
- jednotlivé výskyty nějakých entit - zkrátka **individa**
- resp. analytické funkce „*vypočítávající*“ *nezávisle na stavu světa* z jedněch individuí (argumentů) jiná individua (výsledky)
- toto vše jsou tzv. **extenze**
- deskripce jsou extenze (třídy nezávislé na stavu světa)
- ... a dále:

Co jsou *objekty* našeho zájmu? (2)

- jednotlivé entity
(StavySveta \rightarrow (Jednotliviny \rightarrow Bool))
- popisné atributy
(StavySveta \rightarrow (PlatZam))
- vztahové atributy
(StavySveta \rightarrow (MnoZboDodOdbCas))
- to vše jsou tzv. **intenze**
- a prakticky všechno to jsou funkce ...

Jak o tom všem mluvíme ?

- *výrazy* přirozeného jazyka (Cz, An, ...)
- *výrazy* umělých jazyků
 - programovací jazyky, specifikační jazyky
 - jazyk matematiky, logiky
 - diagramy
 - ...
- *výrazy* jazyka označují *objekty* našeho zájmu

Komunikace

- různými výrazy lze označit týž objekt
- pokud jsou tyto různé výrazy v rámci jednoho jazyka, hovoříme o **synonymech**
- jeden výraz může označovat více objektů
- pokud výraz uvažujeme v rámci jednoho jazyka, hovoříme o **homonymech**
- abychom se domluvili, potřebujeme identifikovat (jednoznačně) objekty, které máme na mysli \Rightarrow potřebujeme *pojmy*

Pojmy jako identifikační procedury

- Pojmy identifikují objekty, které máme na mysli
- Pojmy jsou jakési konstrukce, které nám umožňují zadat objekt, o kterém chceme něco vypovědět
- Pojmy jsou reprezentovány pomocí jazykových výrazů

Pojmy umožňují dorozumění

jazykový výraz:

množství daného druhu zboží dodané
daným dodavatelem danému odběrateli

- označuje objekty (= jednotlivé
tabulky) z obr. 14
- to mohou být rozličné objekty v závislosti na
stavu světa

- a reprezentuje pojem (=
konstrukci)

StavySveta \rightarrow MnoZboDodOdb (viz³⁴
obr. 24)

Základní princip komunikace

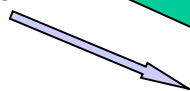
VÝRAZ

representuje

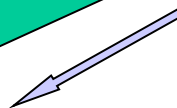


POJEM

označuje

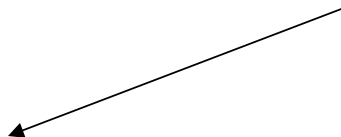


identifikuje



OBJEKT

Extenze

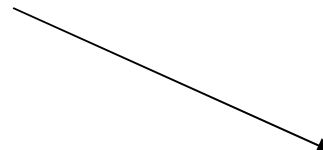


Intenze



Pojem

objekt vyššího řádu



.....

princip komunikace a DM

- zákazníci, uživatelé, informatici ... hovoří **výrazy** jazyka o **objektech** jež mají na mysli
- vzhledem k profesní různosti si špatně navzájem rozumí
- analytici („datoví modeláři“) nalézají *pojmy*, které jsou používanými **výrazy** reprezentovány a které identifikují jednoznačně předmětné *objekty*
- z těchto pojmů vytvářejí **konceptuální (datový) model**

princip komunikace a DM (2)

- konceptuální model identifikuje objekty = datové tabulky, které mohou být v umělém (databázovém dotazovacím) jazyce označeny jako výrazy z databáze
- v databázích jsou uloženy tyto výrazy
- pomocí výrazů z databáze opět komunikují uživatelé (zákazníci) při řešení svých „business“ problémů

P114

Logické základy

TIL

4

Témata

- parciální funkce, báze a typy
- objekty typu T
- TIL, Frege-Churchův trojúhelník označení
- epistémická báze
- universum diskursu
- možné světy
- extenze a intenze
- pojmy a intenze

parciální funkce ...

- n-ární funkce: složitost funkce je $n+1$
- funkce nedefinovaná na n -tici
- totožnost funkcí : princip extenzionality
 $F_1: M_1 \times \dots \times M_n \rightarrow M, F_2: M_1 \times \dots \times M_n \rightarrow M,$
 $X \in M_1 \times \dots \times M_n, Y \in M$ -- libovolné :
 $F_1(X) = Y$ *právě když* $F_2(X) = Y$, píšeme $F_1 = F_2$
- důsledek:
je-li $F_1 = F_2$ a Z libovolný prvek z $M_1 \times \dots \times M_n$, pak F_1 je na Z nedefinováno právě tehdy, když F_2 je na Z nedefinováno

funkce jako procedura

- procesní pohled
- (parciální) funkce je „výpočetní“ resp. „vyhodnocovací“ pravidlo/procedura, které poskytne buď *nic* nebo *výsledek* z množiny M , jestliže jí na vstupu zadáme hodnoty parametrů z $M_1 \times \dots \times M_n$
- *deklar* $x_1, \dots, x_n :: M_1, \dots, M_n$ *deklar* $y :: M$ (*tělo proc*)
- x_i ... formální parametry, y ... výsledek

Definice typů

- B_1, \dots, B_n jsou neprázdné, vzájemně disjunktní množiny. Kolekci takových množin nazveme **báze \mathbf{B}** .
 - (1) Každý prvek z \mathbf{B} (tj. každá z množin B_1, \dots, B_n) je **typ nad (bází) \mathbf{B}** .
 - (2) Nechť T_1, \dots, T_m jsou typy nad \mathbf{B} . Pak kartézský součin $T_1 \times \dots \times T_m$, značený (T_1, \dots, T_m) , je **typ nad \mathbf{B}** .
 - (3) Nechť T, T_1, \dots, T_m jsou typy nad \mathbf{B} . Pak množina všech možných parciálních funkcí z $T_1 \times \dots \times T_m$ do T , značená $(TT_1 \dots T_m)$, je **typ nad \mathbf{B}** .
 - (4) **Typ nad \mathbf{B}** je jenom to, co splňuje (1) až (3).
- Typy tvoří nad bází \mathbf{B} nekonečnou hierarchii

Relace a třídy jako typy

- Nechť $\text{Bool} = \{P, N\} = \{T, F\} = \{\text{Pravda}, \text{Nepravda}\} = \dots$ je množina pravdivostních hodnot.
- Nechť báze **B** obsahuje (mimo jiné) množinu Bool. Nad takovou bází jsou typem (pomocí charakteristické funkce):
 - relace
 - třídy (množiny)
 - kartézské součiny
- Typ, který je kartézským součinem, nazýváme **n-ticový typ** a značíme (T_1, \dots, T_m) --redundance (2) v Def typů? **NE**
- Všechny n-ticové typy předpokládáme v tzv. normálním tvaru: žádné T_i není n-ticový typ.

Pro totální funkce platí

- Schönfinkelova redukce: Typ $(TT_1T_2 \dots T_n)$ je přirozeně ekvivalentní s typem $(\dots(TT_n)T_{n-1} \dots)T_1)$
- Přirozená projekce: Typ $((T_1, \dots, T_n)T)$ je přirozeně ekvivalentní s typem $((T_1T), (T_2T), \dots, (T_nT))$
- Přirozená ekvivalence: totožnost podle principu extenzionality
-- výskyt funkce jednoho typu v libovolném výrazu je možno nahradit výskytem funkce druhého typu

T-objekt nad **B**:

- Nechť **T** je typ nad bází **B**. Každý prvek \underline{t} z množiny **T** nazýváme objektem typu **T** nad **B**, nebo-li T-objektem nad **B**.
- Fakt, že \underline{t} je T-objektem nad **B**, zapisujeme formou \underline{t}/T
- PŘÍKLADY: **B** = {#Zam, #Proj, Bool}
#Zam = {Kos, Sam, Mot}
#Proj = {Pojist, Maso}
#Proj-objekty ?, (Bool #Zam)-objekty ?,
(#Zam #Proj)-objekty ?
kolik kterých je?

označování

- $(TT_1T_2 \dots T_n) =_{df} ((T_1, \dots, T_n) \rightarrow T)$
- $(\#Zam \#Proj)$ -objekty $=_{df} (\#Proj \rightarrow \#Zam)$ -objekty
- $((Bool \#Zam) \#Proj)$ -objekty $=_{df}$
 $(\#Proj \rightarrow (\#Zam \rightarrow Bool))$ -objekty
- Schönfinkelova redukce: $Typ ((T_1, \dots, T_n) \rightarrow T)$ je přirozeně ekvivalentní s typem
 $(T_1 \rightarrow (T_2 \rightarrow \dots (T_n \rightarrow T) \dots))$
- Přirozená projekce: $Typ (T \rightarrow (T_1, \dots, T_n))$ je přirozeně ekvivalentní s
typem
 $((T \rightarrow T_1), (T \rightarrow T_2), \dots, (T \rightarrow T_n))$

Cíle v DM

- domluvit se
- vyjít z přirozeného jazyka, kterým se všichni (uživatelé, zákazníci, informatici ...) vyjadřují
- poznat adekvátní pojmy
- budovat všechny konstrukce nad rozumně zvolenou bází

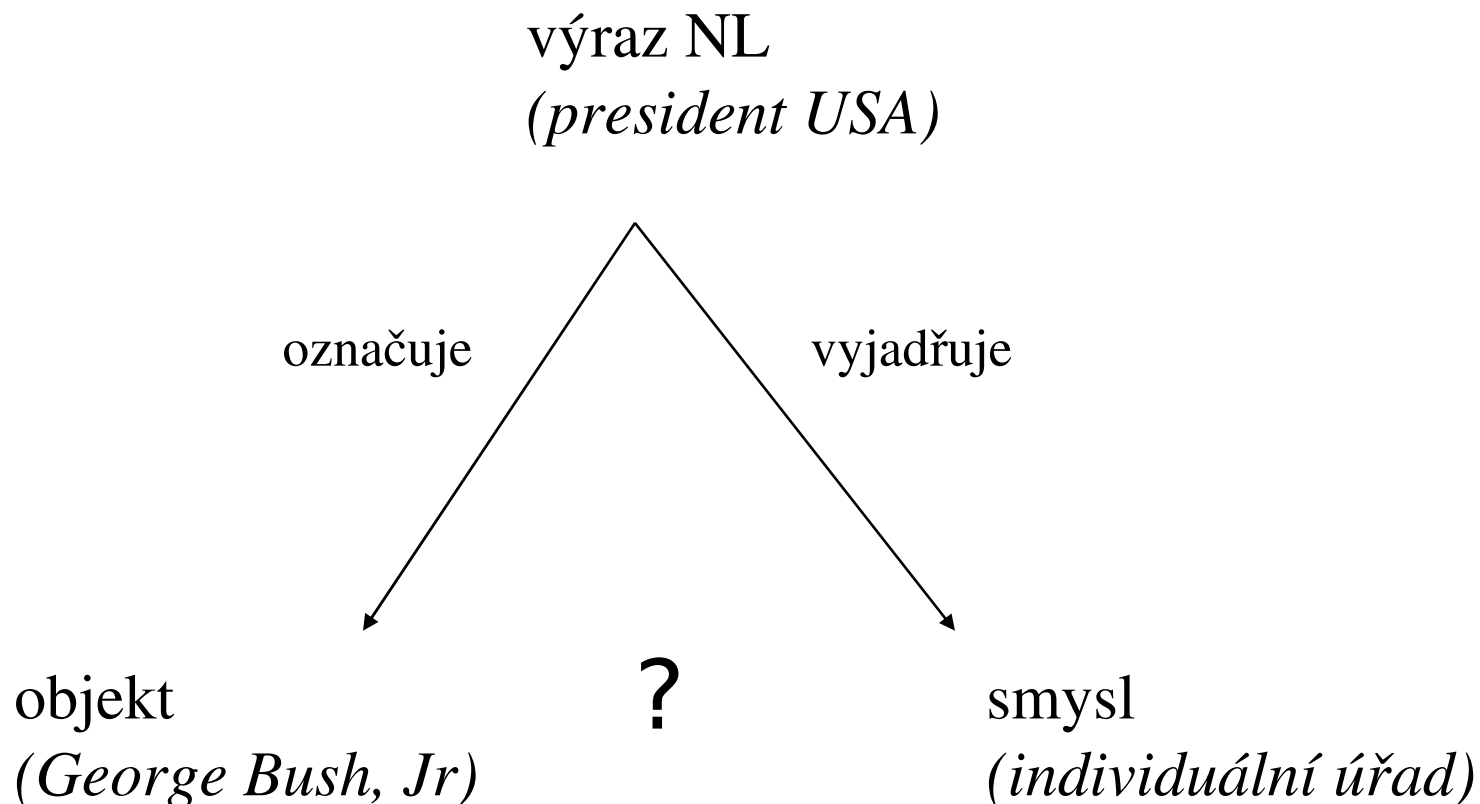
TIL = transparentní intenzionální logika

- logika založená na jisté modifikaci typovaného lambda kalkulu jako nástroje pro práci s funkcemi
- transparentní systém, formální aparát není předmětem studia ale jenom prostředkem ke studiu konstrukcí
- nepreferuje jistá tzv. logická slova (logické spojky, kvantifikátory,...)
- sémantika založená na „možných světech“
- „univerzum“ je chápáno jako společné všem možným světům

TIL - pokračování

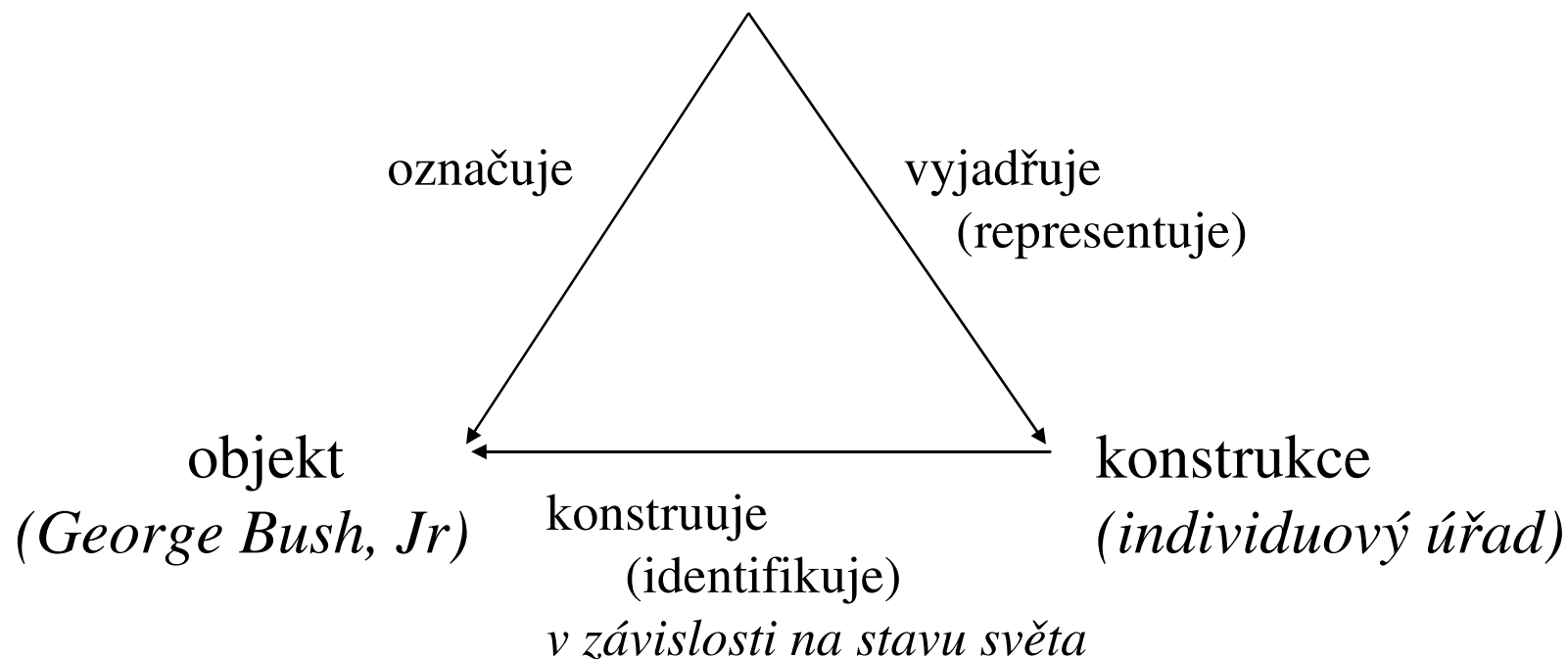
- nahrazuje Frege/Churchův trojúhelník označení:
výraz označuje *objekt* a
výraz vyjadřuje *smysl*
korigovaným trojúhelníkem označení:
výraz označuje *objekt*
výraz vyjadřuje *konstrukci*
konstrukce konstruuje *objekt*
- co je epistémickou (poznávací) bází ?

Frege-Church: Trojúhelník Označení



Korigovaný Trojúhelník Označení

výraz NL
(*president USA*)



epistémická báze

- epistémická báze: $\mathbf{B} = \{\text{Bool}, \text{Tim}, \text{Univ}, \text{Wrd},\}$
- Bool obsahuje prvky Pravda, Nepravda
- Tim je množina časových okamžiků, reálná čísla
- Univ je množina individuí apriorně daných
- Wrd je množina možných světů, logický prostor
- epistémické báze se mohou vzájemně lišit v množinách Univ a Wrd
- o objektech „bydlících“ v typech nad takovou bází je právě schopen vypovídat přirozený jazyk

Universum diskurzu

- individua (= jednotliviny)
- individua jsou jakékoli (ne? nutně fyzické)
objekty vzájemně odlišitelné (objekty časoprostorového světa)
- jsou to „nositelé vlastností“ a jsou odlišné od těchto vlastností
- jsou dána (vzhledem k jazyku jímž komunikujeme) a priori:
zkoumáme-li, jaké vlastnosti má daný objekt nemůžeme současně zjišťovat, o které individuum jde - vždy víme, které individuum vyšetřujeme
- zde odmítáme tzv. esencialismus: „individua mají některé empirické vlastnosti nutně - ty tvoří jejich esenci = podstatu“

Universum diskurzu -pokračování

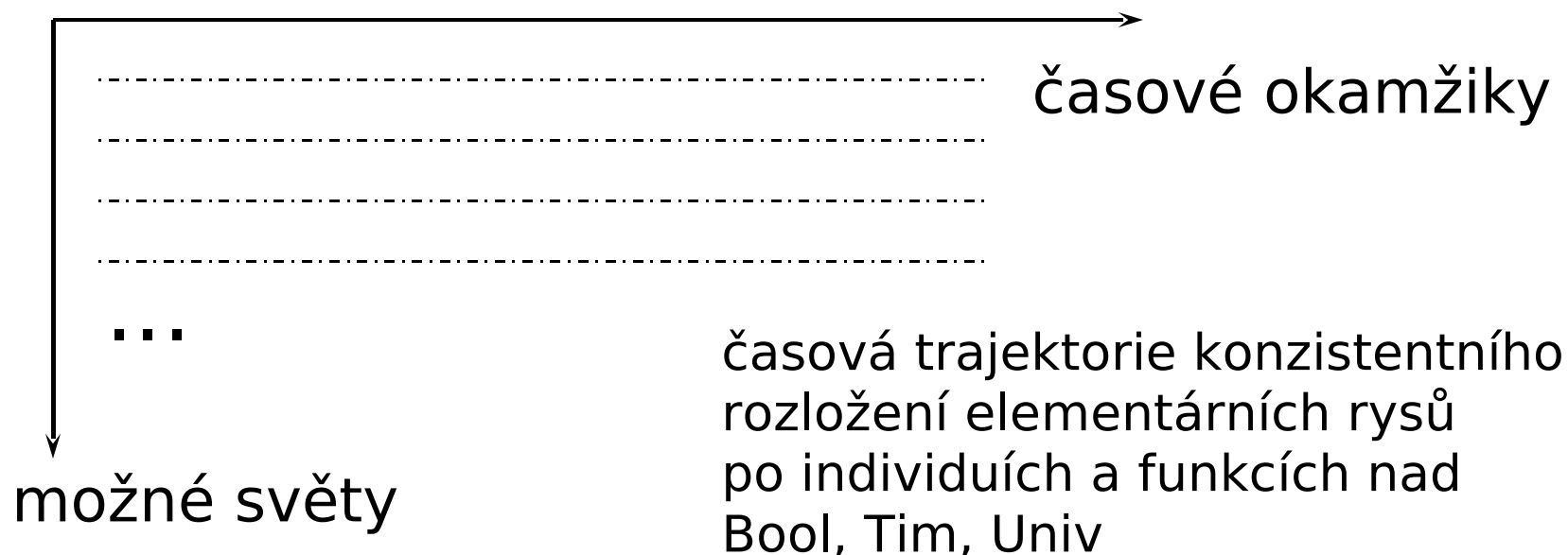
- podstatnou charakteristikou je vzájemná odlišitelnost
- ta je závislá na stavu našeho (vědeckého) poznání reálného světa (zejména na fyzikálním poznání)
- individuum se nemůže přeměnit v jiné individuum, ale
- individuum může přebírat v různých stavech světa různé vlastnosti, které (i velmi podstatně) mění jeho povahu (charakter) a tím i roli, kterou vůči nám jako pozorovatelům hraje
- (zde je základ ovlivnění DM filosofickou logikou)

Možné světy

- původně G. Leibnitz, autor TIL P. Tichý
- svět nechápeme jako souhrn věcí
- možný svět je souhrn všech faktů, které mohou platit a při tom si vzájemně neodporují
- množinu faktů, které neobsahují spor nazýváme konzistentní
- každá konzistentní množina faktů obsahuje fakta, která jsou (logicky) možná resp. myslitelná
- základní intuice vnímání jakéhokoli „světa“ je jeho vývoj v čase

Možné světy -pokračování

- každý možný svět je možno vidět jako časovou posloupnost maximálních množin logicky myslitelných faktů



Možné světy -pokračování

- soubor všech možných světů (Wrd) určuje to, co je logicky možné, proto jej nazýváme také „logický prostor“
- Wrd je dáno a priori (stejně jako ostatní prvky báze)
- časová posloupnost množin všech skutečných faktů se nazývá aktuální svět
- který z možných světů je aktuální, nelze „logicky vypočítat“, ale lze pouze empiricky zjišťovat některou výseč aktuálního světa
- vědět, který z možných světů je aktuální = vševědoucnost

Možné světy -pokračování

- tvrzení, že daný poznatek je (v daném okamžiku) pravdivý, znamená, že mezi možnými světy ve kterých tento poznatek platí, je aktuální svět
- poznatky sdělujeme oznamovacími větami, tzv. propozicemi
- každá propozice vyčleňuje ve Wrd podmnožinu těch možných světů, ve kterých je ona pravdivá
- v daném časovém okamžiku pravdivé propozice o stavu světa vyčleňují tedy ve Wrd podmnožinu možných světů, která zaručeně obsahuje aktuální svět

intenze, extenze

- T je typ nad epistémickou bází **EB**
 - (1) neexistuje T_1 nad **EB** tak, že
 $T = ((T_1 \text{Tim}) \text{Wrd})$, pak
T-objekty jsou **intenze 0-tého řádu (extenze)**
 - (2) nechť T intenze k-tého řádu. Potom
 $((T \text{Tim}) \text{Wrd})$ -objekty jsou **intenze (k+1)-ho řádu**
 - (3) intenze k-tého řádu pro $k > 0$ nazýváme
intenze

extenze, intenze - příklady

- třídy individuí: (BoolUniv)-objekty -- extenze
- vlastnosti individuí:
(((BoolUniv)Tim)Wrd)-objekty -- intenze
- propozice: ((BoolTim)Wrd)-objekty -- intenze
- individuové úřady: ((UnivTim)Wrd)-objekty
-- intenze
- veličiny: ((TimTim)Wrd)-objekty
(počet vlasů na hlavě) -- intenze
- třídy vlastností: (Bool(((BoolUniv)Tim)Wrd))-objekty
-- extenze!

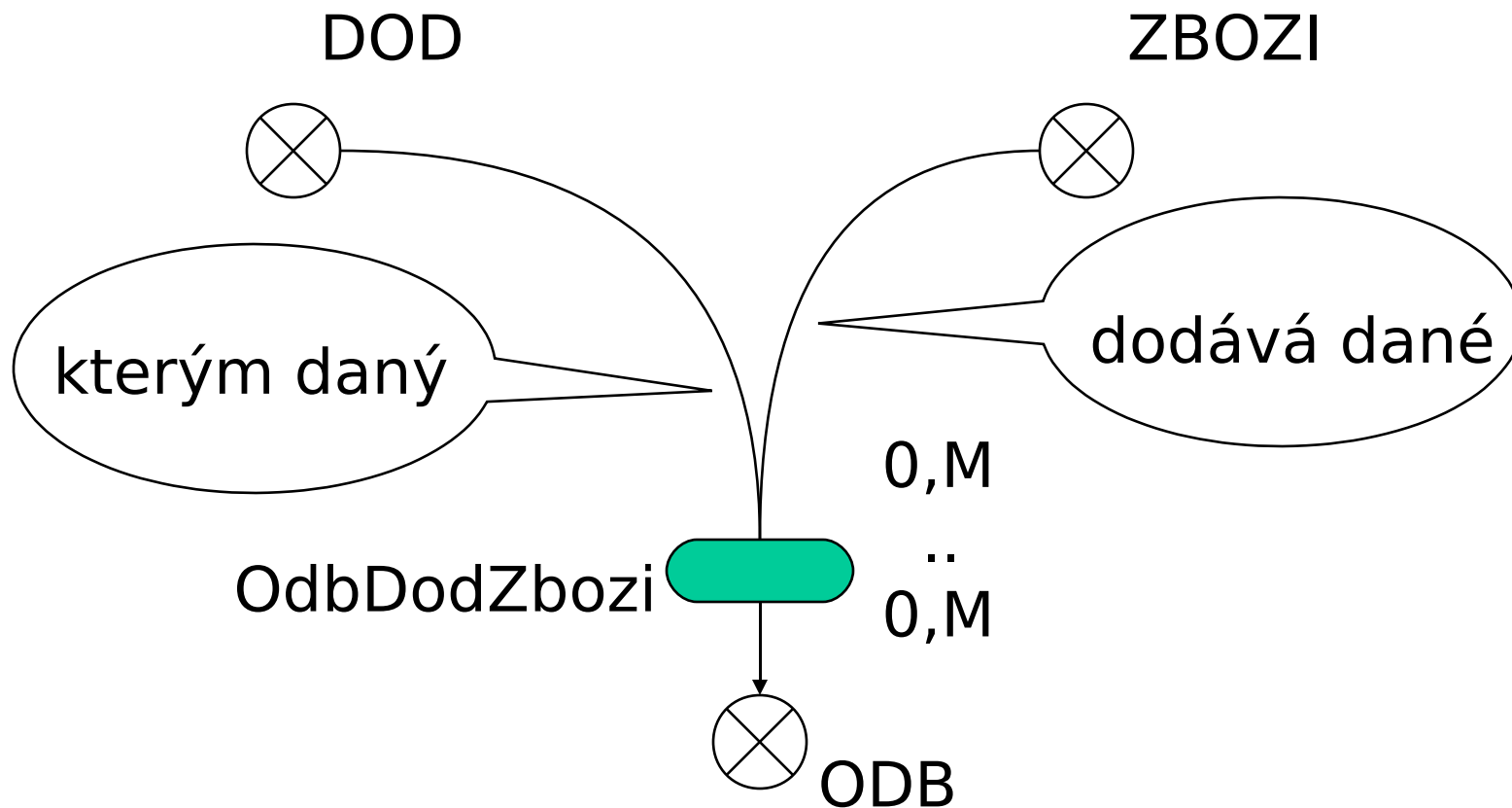
Důležité intenze - *vlastnosti individuí*:

- $((\text{BoolUniv})\text{Tim})\text{Wrd}$ -objekty
- $(\text{Wrd} \rightarrow (\text{Tim} \rightarrow (\text{Univ} \rightarrow \text{Bool})))$
- Příklady:
 - být kočkou: možnému světu přiřazuje historii vývoje populace koček
 - „být kočkou“ není dáno daným okamžikem, a v různých možných světech mohou být populace (extenze) koček různé
 - být Zaměstnancem
 - být Dodavatelem
 - být Zbožím
 - být Odběratelem

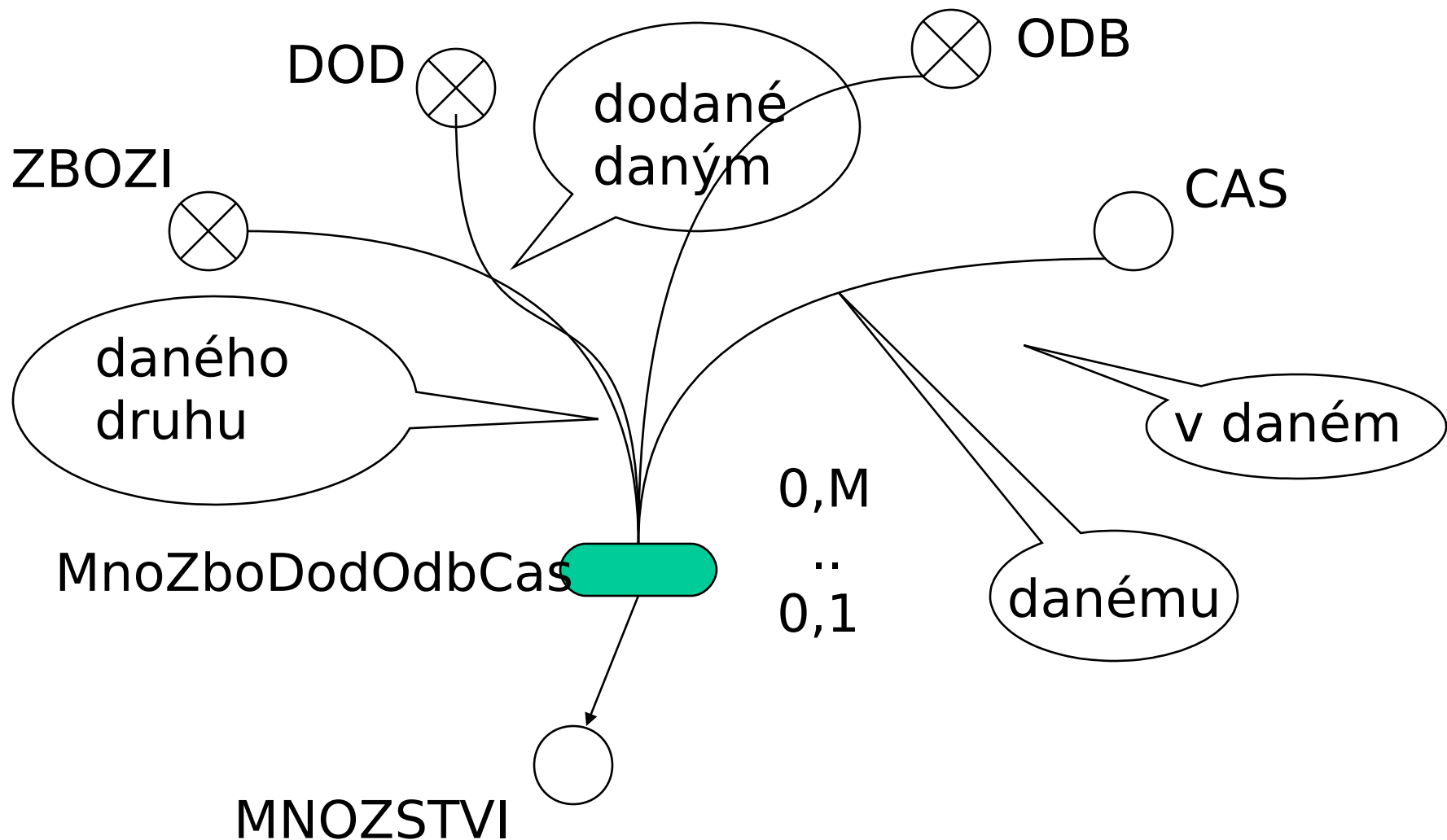
Důležité intenze - *propozice*:

- $((\text{BoolTim})\text{Wrd})$ -objekty
- $(\text{Wrd} \rightarrow (\text{Tim} \rightarrow \text{Bool}))$
- oznamovací věty jejichž pravdivost je závislá na stavu světa
- Příklady:
 - V Brně právě teď prší.
 - Kostecké uzeniny dodávají jemné párky do prodejen Tesco a Makro. (1)
 - Dodavatel (MPK) dodává výrobek (Selský salám) do prodejen odběratele (Delvita). (2)
- (1) a (2) jsou propozice „generované“ atributem OdbDodZbozi z minulé přednášky -- viz dále

HIT-atribut jako „generátor“ propozic

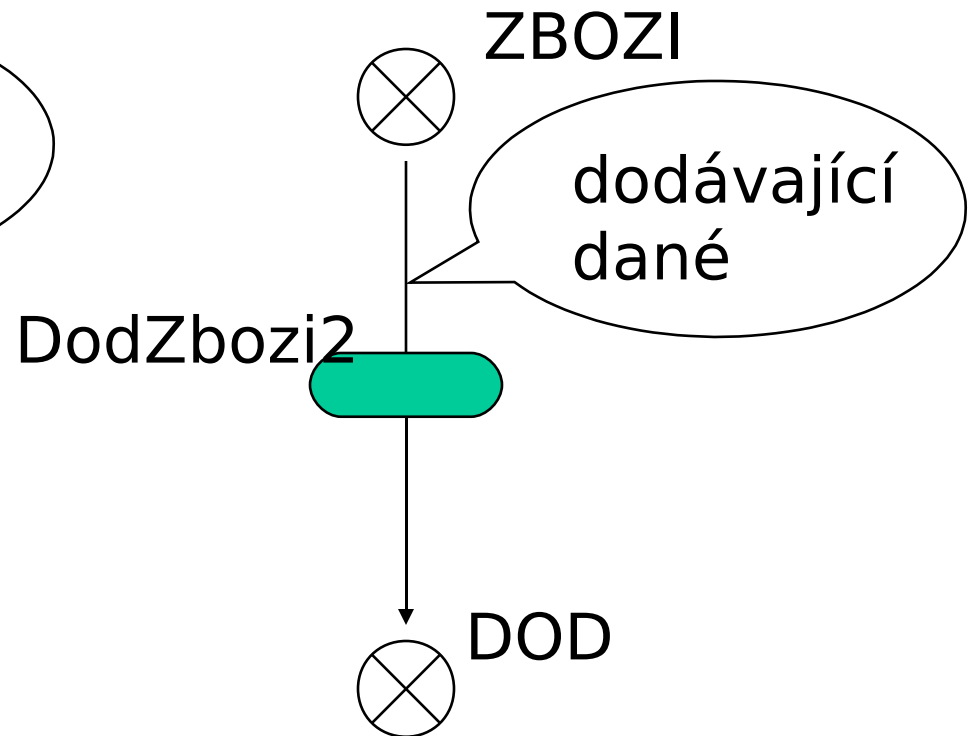
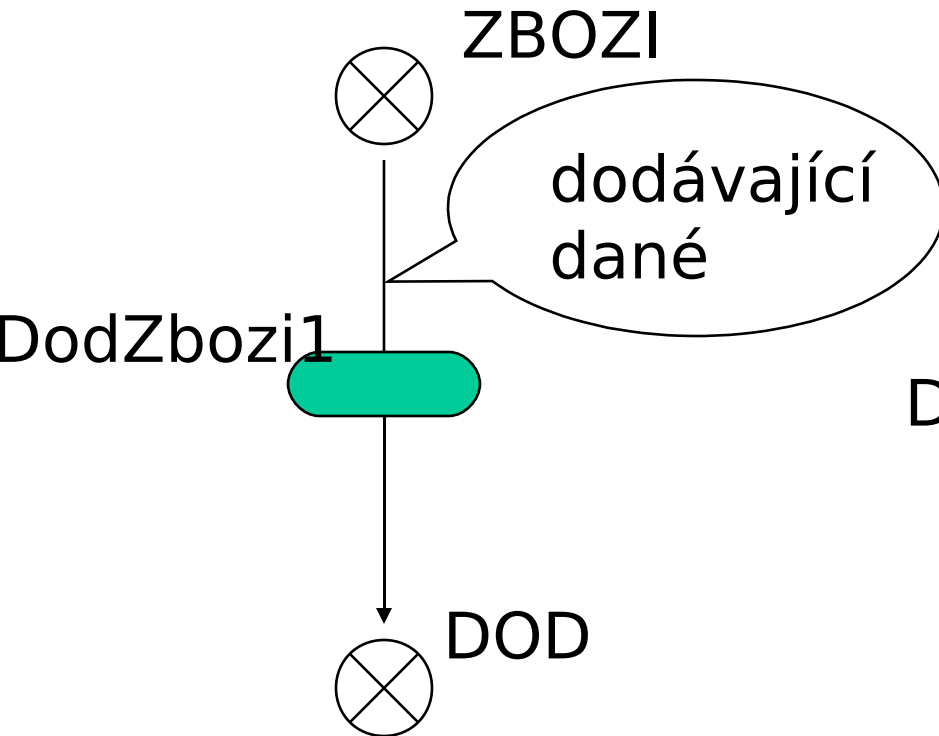


Jaké propozice vygeneruje atribut ?



Jaké propozice vygenerují atributy ?

jak se to bude lišit ?



další důležité intenze

- individuové úřady: $((\text{UnivTim})\text{Wrd})$ -objekty
 - president USA
 - Jitřenka, Večernice
 - nejvyšší hora na zeměkouli
 - děkan FI MU
- veličiny: $((\text{TimTim})\text{Wrd})$ -objekty
 - počet vlasů na hlavě
 - počet dodavatelů daného výrobku
 - hustota roztoku vstupujícího do výrobní operace

TIL s jednoduchou teorií typů

(historicky původní stanovisko -- 80-tá léta)

- Pojmové zachycování světa je v NL uskutečňováno pomocí jmen pojmů, jmen extenzí, syntaktických elementů a pragmatiky
- Pojmy: lze ztotožnit s intenzemi (výrazy jazyka označující intenze, jsou jména pojmů)
- Extenze v jazyce: jména extenzí v přirozeném jazyce jsou: spojky označující pravdivostní funkce, kvantifikátory, matematické výrazy, vlastní jména míněná jako „nálepky-jmenovky“ individuí, ...
- Pragmatika v jazyce: označení objektů v závislosti na situaci, v níž se děje promluva
- Syntaktické elementy: nemají samostatnou sémantiku (než, že,...)

TIL s jtt: korekce

- základní intuice (viz úvod): pojmy jsou identifikační procedury - způsob zadání objektů
- tedy **pojmy** nemůžeme **ztotožnit** přímo s intenzemi, ale s **konstrukcemi intenzí !!!**
- POZN.: takové pojetí „pojmu“ nezahrnuje matematické pojmy
- v tomto pojetí stojí konstrukce mimo budovanou teorii a jsou pouze nástrojem pro práci s objekty
- tento přístup neumožňuje vybudovat konzistentní teorii pojmu (viz DM2)

sémantika logických a matematických objektů, sémantika výpovědí o světě

- ... základní myšlenkový konstrukt je **funkce** (jako přiřazení)
- chceme-li porozumět (složitějším) objektům, ptáme se co je to za **funkce**
- znát sémantiku nějakého **jazykového výrazu** *označujícího* jistý **objekt**, znamená rozumět tomu, jakou **funkci** tento výraz *označuje*, a kterou z jejích *identifikací* (t.j. kterou její **konstrukci**) *reprezentuje*

objekty výrokového počtu

- Uvažme typ Bool nad **EB**
- pravdivostní funkce
unární (BoolBool)-objekty
 - negace $[\neg A]$ $\neg A$
- binární (BoolBoolBool)-objekty
 - konjunkce $[\wedge (A, B)]$ $A \wedge B$
 - disjunkce $[\vee (A, B)]$ $A \vee B$
 - implikace $[\Rightarrow (A, B)]$ $A \Rightarrow B$
- vše o čem hovoří výrokový počet, lze definovat nad bází **B** = {Bool}

objekty predikátového kalkulu

- Nechť T je typ nad **EB**
- obecný T -kvantifikátor $(\text{Bool}(\text{Bool}T))$ -objekt
funkce která množině T -objektů přiřadí P , je-li totožná s množinou T , jinak N
 $\forall x::T (C(x))$
- existenční T -kvantifikátor
 $(\text{Bool}(\text{Bool}T))$ -objekt
funkce která množině T -objektů přiřadí P , je-li neprázdná, jinak N
 $\exists x::T (C(x))$
- vše o čem hovoří predikátový kalkul, lze vybudovat nad
 $\mathbf{B} = \{\text{Bool}, \text{Univ}\}$

Některé matematické objekty

- T-singularizátor $(T(\text{BoolT}))$ -objekt
funkce, která jednoprvkové množině, jež je podmnožinou T , přiřadí onen jeden prvek a na všech ostatních podmnožinách T je nedefinovaná
 $\lambda x::T (C(x))$
- identita (BoolTT) -objekt
funkce, která dvojici T -objektů přiřadí P , právě když jsou oba jeden a týž objekt
 $[\equiv_T x y]$

sémantika

- vše o čem hovoří matematická logika i celá matematika, lze vybudovat pohodlně nad $\mathbf{B} = \{\text{Bool}, \text{Univ}, \text{Tim}\}$...
- ... ale sémantika sdělení používaných v přirozeném jazyce při popisu reálného světa chybí
- sémantiku lze zahrnout do podpůrné teorie právě nad epistémickou bází
 $\mathbf{EB} = \{\text{Bool}, \text{Univ}, \text{Tim}, \text{Wrd}\}$
- (sémantika možných světů)
- existují i jiné přístupy k budování sémantiky ...

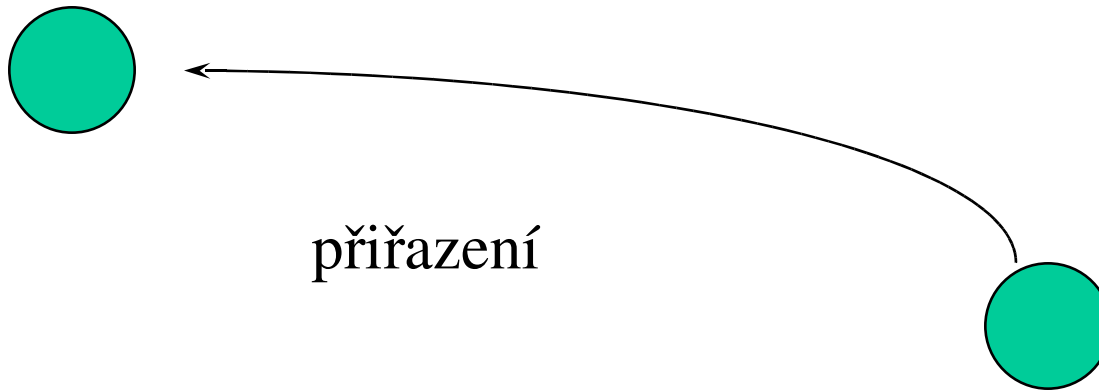
P114
Konstrukce
užití λ - kalkulu

5

Témata

- TIL s jednoduchou teorií typů
- atomické konstrukce
- konstrukce aplikace
- konstrukce abstrakce
- konstrukce n-tice a projekce
- ekvivalence konstrukcí
- uzavřené a otevřené konstrukce

Elementární konstrukt



Přiřazení - zobrazení - funkce:

je to předpis = procedura, který říká jaký výsledek (výstup) je přiřazen k danému vstupu

funkce (filozoficky)

- bazální pojem: pojem funkce
- skoro vše, o čem přemýšlíme jsou funkce (až na P,N, časové okamžiky, individua)
- způsob, jak přemýšlíme: něco něčemu přiřazujeme, tj. základním konstruktem přemýšlení je funkce
- jsme zvyklí jednotliviny, v něčem si podobné, „sypat“ do samostatných kontejnerů a přemýšlet a vyjadřovat se pomocí těchto kontejnerů = typů

funkce jako procedura

- procesní pohled
- (parciální) funkce je „výpočetní“ resp. „vyhodnocovací“ pravidlo/procedura, které poskytne buď *nic* nebo *výsledek* z množiny M , jestliže jí na vstupu zadáme hodnoty parametrů z $M_1 \times \dots \times M_n$
- *deklar* $x_1, \dots, x_n :: M_1, \dots, M_n$ *deklar* $y :: M$ (*tělo proc*)
- x_i ... formální parametry, y ... výsledek

jak zadat konkrétní funkci ?

- Máme danu bázi \mathbf{B} . Objekty nad \mathbf{B} jsou
 - prvky množin tvořících \mathbf{B} (bázových množin)
 - všechny funkce definované nad \mathbf{B}
- Vezměme funkci $\underline{f} / (BA)$ a prvek \underline{a} z A . Nechť $\underline{f}(\underline{a}) = \underline{b}$, \underline{b} je prvkem B . \underline{b} je hodnotou \underline{f} na \underline{a} .
- Funkci \underline{f} - objekt typu (BA) - a objekt \underline{a} typu A lze použít jako způsob určení jiného objektu - objektu \underline{b}/B .
- Jak sestrojít (zadat) konkrétní funkci? Sestrojít typ (množinu všech funkcí) umíme ...
- Jak obecně zadávat objekty, abychom s nimi mohli pracovat?

klíčový „pojem“: konstrukce

- návod k určitým intelektuálním krokům, pomocí kterých porozumíme významu jazykových výrazů, tj.
 - porozumíme tomu, co ony reprezentují
 - identifikujeme přesně to, co ony (někdy ne zcela jednoznačně) označují
- srovnej s KTO v přednášce č. 4

konstrukce v TIL s jtt (1)

- Termínem **T-konstrukce** rozumíme specifický způsob určení (zadání) nějakého T-objektu.
- Je-li objekt zadán nějakou konstrukcí, řekneme, že konstrukce konstruuje tento objekt.
- Konstrukce souvisí s „procedurou identifikace“ něčeho v rámci všeho ostatního
- POZOR: konstrukce objektu je zcela něco jiného než objekt sám.
- Nechť $\underline{f}(\underline{a}) = \underline{b}$. Z objektu \underline{b} nelze žádným způsobem poznat, že byl zadán právě konstrukcí $\underline{f}(\underline{a})$.

konstrukce v TIL s jtt (2)

- TIL s jtt = TIL s jednoduchou teorií typů (původní varianta TIL, která byla v 80-tých letech použita jako základ metody HIT datového modelování)
- v TIL s jtt sama konstrukce není objektem zájmu, ale pouze prostředek k identifikaci objektů nad EB
- v tomto (jednodušším) pojetí v konstrukcích přímo vystupují objekty
- později (DM2) ukážeme korekci tohoto přístupu použitím TIL s rtt (= TIL s rozvětvenou teorií typů) zavedením zásadního triku - tzv. *triviální konstrukce* nebo-li běžněji *konstrukce trivializace*
- odkaz na TIL s jtt budeme v dalším vynechávat

konstrukce (3)

- Nechť $\underline{f}(\underline{a}) = \underline{b}$. Říkáme, že objekty \underline{f} a \underline{a} se **vyskytují** v konstrukci objektu \underline{b} . Konstrukce jsou složeniny objektů.
- V konstrukci se některé objekty mohou vyskytnout vícekrát.
- V konstrukcích vystupují přímo **objekty**, nikoli neinterpretované symboly, kterým by objekty byly přiřazovány nějakou interpretací. V tom spočívá **transparentní pojetí**.
- Jedním z cílů je reprezentace informací v počítačích: proto se pohybujeme v dualismu transparentní - formální.

konstrukce, valuace (4)

- **Nevlastní konstrukce:** neexistuje objekt, který by byl konstruován touto konstrukcí.
- Nechť f je nedefinovaná na a . Potom konstrukce $f(a)$ nezadáva žádný objekt, a říkáme, že tato konstrukce je nevlastní.
- Představa konstrukce jako procedury na počítači:
dosazením konkrétních argumentů za formální parametry (**valuací**) procedura
 - buď něco „spočítá“ = konstruuje nějaký objekt
 - nebo abortuje = je při tomto dosazení nevlastní

proměnné, valuace (5)

- Nechť pro každý typ T nad \mathbf{B} máme přiřazenu spočetnou množinu V^T tak, že V^T je disjunktní s každým typem nad \mathbf{B} (speciálně tedy V^T je disjunktní s množinou T), a pro každé dva různé typy T_1, T_2 platí V^{T_1} je disjunktní s V^{T_2} . Prvky množiny V^T nazýváme **T-proměnnými**. **Proměnná** je T-proměnná pro nějaký typ T .
- Totální funkce v z množiny všech proměnných do množiny všech objektů se nazývá **valuace**, jestliže pro jakoukoli T-proměnnou x platí, že vx je T-objekt.
- valuace znamená dosazení hodnot za všechny proměnné při zachování „typové kázně“

konstrukce atomické (6)

- Nechť X je nějaký T -objekt. Tento objekt je zadán sám sebou. Říkáme, že X je **T -konstrukce** a nazýváme ji **T -atom**. Nechť v je libovolná valuace. T -konstrukce X **v -konstruuje** objekt X .
- Nechť X je nějaká T -proměnná. Jako taková může konstruovat libovolný T -objekt. Opět řekneme, že X je **T -konstrukce** a nazveme ji **T -atom**. Nechť v je libovolná valuace. T -konstrukce X **v -konstruuje** objekt vX .
- T -atom nemůže nikdy být nevlastní.

konstrukce aplikace (7)

- Nechť T, T_1, \dots, T_n jsou jakékoli typy nad \mathbf{B} . Nechť A je $(TT_1\dots T_n)$ -konstrukce, B_1 je T_1 -konstrukce, ..., B_n je T_n -konstrukce.
 - A je konstrukce, která zadává objekty, jež jsou funkce $(T \leftarrow (T_1, \dots, T_n))$
 - B_i jsou konstrukce, jež zadávají objekty typu T_i , o kterých dále nic nevíme
- $[AB_1\dots B_n]$ je **T-konstrukce** zvaná **aplikace** A na B_1, \dots, B_n .
 - zadání funkce lze aplikovat na zadání jiných objektů
 - objekt (funkci) nemohu aplikovat na jiné objekty !!!
(srov. s čím pracujeme např. v integrálním počtu)

konstrukce aplikace (8)

- Nechť v je libovolná valuace. Nechť A v -konstruuje (objekt) funkci $\underline{A}/(TT_1...T_n)$ a nechť pro každé i , B_i v -konstruuje objekt \underline{B}_i/T_i . Mohou nastat dva případy:
 - funkce \underline{A} je na n -tici $(\underline{B}_1, \dots, \underline{B}_n)$ nedefinována: potom řekneme, že konstrukce $[AB_1...B_n]$ je **v -nevlastní**
 - funkce \underline{A} na n -tici $(\underline{B}_1, \dots, \underline{B}_n)$ je definována: potom řekneme, že konstrukce $[AB_1...B_n]$ **v -konstruuje T -objekt**, který je hodnotou funkce \underline{A} na n -tici $(\underline{B}_1, \dots, \underline{B}_n)$

konstrukce aplikace (9)

- V ostatních případech, když buď
 - neexistuje objekt v -konstruovaný A nebo
 - pro nějaké i neexistuje objekt v -konstruovaný B_i
 - (a stačí alespoň něco z toho)
- nelze konstrukci $[AB_1...B_n]$ přiřadit žádný v -konstruovaný objekt a říkáme, že je tato konstrukce v -nevlastní
- PŘÍKLADY
 - z matematiky
 - ze života

konstrukce aplikace (9a) --příklady

- sčítání v oboru reálných čísel $:: ((\text{Tim}, \text{Tim}) \rightarrow \text{Tim})$
[+ 5 2] konstruuje Tim-objekt 7
[+ 5 x] v-konstruuje Tim-objekt 5+x
- dělení v oboru reálných čísel $:: ((\text{Tim}, \text{Tim}) \rightarrow \text{Tim})$
[: 6 2] konstruuje Tim-objekt 3
[: 6 0] je v-nevlastní pro všechna v, nekonstruuje nic
[+ 5 [: 6 0]] je rovněž v-nevlastní pro všechna v
- relace > v oboru reálných čísel $:: ((\text{Tim}, \text{Tim}) \rightarrow \text{Bool})$
[> 6 2] konstruuje Bool-objekt Pravda
[> 2 6] konstruuje Bool-objekt Nepravda
[> 6 [: 6 0]] je v-nevlastní pro všechna v
[> x 2] v-konstruuje Bool-objekt Pravda pro ty valuace v, které přiřadí x číslo větší než 2 a Nepravda pro všechny ostatní valuace v

konstrukce aplikace (9b) --příklady

- $ZAM' / (Wrd \rightarrow (Tim \rightarrow (Univ \rightarrow Bool)))$
[[[ZAM' w] t] Novák] konstruuje v daném světě w a čase t Pravda, jestliže individuum s „nálepkou“ Novák je zaměstnancem, a konstruuje Nepravda, jestliže není;
[[ZAM' w] t] konstruuje v daném světě w a čase t množinu individuí, která mají tu čest právě v tomto w a t býti zaměstnanci
- $PlatZam / (Wrd \rightarrow (Tim \rightarrow (ZAM \rightarrow PLAT)))$
[[PlatZam w] t] konstruuje tabulku (dvousloupcovou) platů zaměstnanců, která platí v daném světě w a čase t;
[[[PlatZam w] t] Novák] konstruuje tu hodnotu z množiny PLAT, která je přiřazena předchozí tabulkou Novákovi;
[[[[PlatZam w] t] Novák] 8000] konstruuje P nebo N podle toho, zda Novák má či nemá plat 8000.

konstrukce aplikace (9c) --příklady

- $\text{RektorMU} / (\text{Wrd} \rightarrow (\text{Tim} \rightarrow \text{Univ}))$
[[RektorMU w] t] konstruuje Univ-objekt označený nálepkou, která je jménem rektora MU v daném w a daném čase t;
v čase $t=1912$ a v aktuálním světě je v-nevlastní
v čase $t=2001$ konstruovala v aktuálním světě Jiřího Zlatušku
- $\text{RektorZDSBotanicka} / (\text{Wrd} \rightarrow (\text{Tim} \rightarrow \text{Univ}))$
[[RektorZDSBotanicka w] t] je v-nevlastní pro všechny v
(v žádném možném světě není žádné individuum rektorem ZDŠ - to je naše domluva na významu slova rektor a na významu slova ZDŠ)
- $\text{RektorZDS} / (\text{Wrd} \rightarrow (\text{Tim} \rightarrow (\text{Univ} \rightarrow \text{Bool})))$
[[RektorZDS w] t] v-konstruuje { } pro všechny valuace v

konstrukce abstrakce (10)

- Nechť T, T_1, \dots, T_n jsou jakékoli typy nad \mathbf{B} . Nechť A je T -konstrukce. Nechť x_1, \dots, x_n jsou navzájem různé proměnné, x_i je T_i -proměnná, značíme $x_i :: T_i$.
- Definujeme, že „ $\lambda x_1 \dots x_n (A)$ “ je $(TT_1 \dots T_n)$ -konstrukce, zvaná (lambda-) abstrakce A vzhledem k proměnným x_1, \dots, x_n .
- Všimněme si, že to připomíná zápis:
procedure $X_1 \dots X_n$ (**tělo procedury**),
který z „holého“ algoritmu nějakého výpočtu vyrábí (opakovatelně použitelnou a jako celek použitelnou) funkci
- lambda-abstrakce vyrábí (objekty) funkce z n -tic (T_1, \dots, T_n) do T .

konstrukce abstrakce (11)

- Jaké objekty lambda-abstrakce konstruuje?
- Příprava na odpověď:
- Nechť v je libovolná valuace. Označme $v(x_1 := X_1, \dots, x_n := X_n)$, kde X_i jsou T_i -objekty, valuaci která se od v liší pouze tím, že proměnným x_i ($i=1, \dots, n$) přiřazuje objekty po řadě X_i/T_i .
Všem ostatním proměnným přiřazuje to co valuace v .
- Nechme X_i probíhat všechny objekty z T_i , tj. postupně je „dosazujeme“ do $v(x_1 := X_1, \dots, x_n := X_n)$

konstrukce abstrakce (11a)

- Definujme funkci $F / (TT_1 \dots T_n)$ takto:
- $F(X_1, \dots, X_n)$ je nedefinováno, jestliže A je $v(x_1 := X_1, \dots, x_n := X_n)$ -nevlastní
- $F(X_1, \dots, X_n)$ je T -objekt $v(x_1 := X_1, \dots, x_n := X_n)$ -konstruovaný A , jestliže A není $v(x_1 := X_1, \dots, x_n := X_n)$ -nevlastní.
- Potom říkáme, že $\lambda x_1 \dots x_n (A)$ v -konstruuje F

konstrukce abstrakce (12)

- lamda-abstrakce nemůže být nikdy nevlastní, může maximálně dávat všude nedefinovanou funkci
- to že $\lambda x_1 \dots x_n (A)$ je $(TT_1 \dots T_n)$ -konstrukce se zapisuje $\lambda x_1 \dots x_n (A) :: (TT_1 \dots T_n)$
- říkáme, že je to x_1, \dots, x_n - uzavřená otevřená konstrukce A
- **volná proměnná:** lze jí valuací v udělit libovolnou hodnotu
- **uzavřená konstrukce:** neobsahuje volné proměnné
- nechť A obsahuje nejvýše proměnné x_1, \dots, x_n .
Potom $\lambda x_1 \dots x_n (A)$ je uzavřená konstrukce

konstrukce abstrakce - příklady

- otevřená konstrukce: $x+5$
k ní uzavřená konstrukce: $\lambda x (x+5)$
- v je valuace, která x přiřadí 3 a y přiřadí 4:
 - $\lambda y [> x y]$ v -konstruuje (Bool Tim)-objekt -- třídu čísel menších než 3
 - $\lambda x [> x y]$ v -konstruuje třídu čísel větších než 4
 - $\lambda xy [> x y]$ konstruuje (nezávisle na v) relaci „>“ (Bool Tim Tim)-objekt
 - $\lambda x \lambda y [> x y]$ konstruuje funkci, která každému číslu m (dosazenému za x) přiřadí třídu čísel menších než m
 - $\lambda y \lambda x [> x y]$ konstruuje funkci, která každému číslu m (dosazenému za y) přiřadí třídu čísel větších než m

konstrukce abstrakce - příklady

- Nechť $x::\text{ZAM}$, $y::\text{PLAT}$, $w::\text{Wrd}$, $t::\text{Tim}$
 $\text{PlatZam}::(((\text{PLAT ZAM})\text{Tim})\text{Wrd})$
- $\forall \lambda w \lambda t \lambda x \lambda y[[[[\text{PlatZam } w]t]x]y]$ konstruuje co?
- poněvadž datové funkce nad **EB** jsou (vzhledem k w, t) totální, lze uplatnit Shönfinkelovu redukci, a ekvivalentně psát:
 $\lambda wt \lambda x \lambda y[[[\text{PlatZam } (w, t)]x]y]$
- tato konstrukce ale konstruuje funkce typu
 $((((\text{Bool PLAT}) \text{ZAM})(\text{Tim}, \text{Wrd}))$
- jak vyjádřit, že množina $\lambda y[[[\text{PlatZam } (w, t)]x]y]$ je pro dané (w, t) a dané x vždy jednoprvková ?
- funkce „singularizátor“ -- viz dále

konstrukce „n-tice“

(tuple construction)

- A_i , $i = 1, \dots, n$, jsou (ne nutně různé) T_i -konstrukce
- Pak (A_1, \dots, A_n) je konstrukce konstruující objekty typu (T_1, \dots, T_n)
- Nechť v je libovolná valuace:
 - konstrukce (A_1, \dots, A_n) je v -nevlastní, jestliže některé A_i je v -nevlastní
 - jsou-li všechny A_i v -vlastní, nechť \underline{A}_i jsou jimi v -konstruované objekty: potom (A_1, \dots, A_n) v -konstruuje objekt $(\underline{A}_1, \dots, \underline{A}_n)$

konstrukce projekce

- Nechť B je n -ticová konstrukce konstruující objekty typu (T_1, \dots, T_n) , kde T_i již nejsou n -ticové typy.
- Potom $B_{(i)}$, $i = 1, \dots, n$, jsou T_i -konstrukce (projekce na i -tou složku)
- Nechť v je libovolná valuace:
 - je-li B v -nevlastní, pak každé $B_{(i)}$ je v -nevlastní
 - v opačném případě B v -konstruuje objekt $\underline{B}/(T_1, \dots, T_n)$ a $B_{(i)}$ v -konstruuje objekt $\underline{B}_{(i)}/T_i$ -- tzv. i -tou složku objektu \underline{B} .

podkonstrukce

- Nechť A je konstrukce.
- (1) A je podkonstrukce A
- (2) Je-li A tvaru $[BB_1...B_n]$,
pak $B, B_1, ..., B_n$ jsou podkonstrukce A .
- (3) Je-li A tvaru $\lambda x_1...x_n B$,
pak B je podkonstrukce A .
- (4) Jeli A tvaru $(B_1, ..., B_n)$,
pak $B_1, ..., B_n$ jsou podkonstrukce A .
- (5) Je-li C podkonstrukce B a B je podkonstrukce A ,
pak C je podkonstrukce A .
- Nic jiného, než je řečeno v (1) až (5) není podkonstrukce

vázané výskyty a volné proměnné

- Mějme konstrukci $\lambda x_1 \dots x_n (A)$ (1)
- všechny výskyty proměnných x_1, \dots, x_n v (1) jsou vázané výskyty těchto proměnných v konstrukci (1)
- Nechť C je podkonstrukcí B . Výskyty proměnných, které jsou vázanými výskyty v C , jsou vázanými výskyty v B . To platí pro všechna taková B , že C je podkonstrukcí B .
- Výskyty proměnných, které nejsou vázanými výskyty v dané konstrukci, nazýváme **volnými výskyty**.
- Proměnné, které mají alespoň jeden volný výskyt v A , nazýváme **volné proměnné** konstrukce A .

ekvivalence konstrukcí

- konstrukce C_1 je ekvivalentní s konstrukcí C_2 , jestliže pro libovolnou valuaci v a libovolný objekt \underline{A} platí C_1 v -konstruuje \underline{A} právě tehdy, když C_2 v -konstruuje \underline{A} .
- Srov. princip extenzionality
- ekvivalentní transformace konstrukcí /MaPaZla - str.78/
pravidlo *alfa*-redukce:
Jestliže konstrukce C obsahuje vázané i volné výskyty proměnné x , pak všechny vázané výskyty x můžeme přejmenovat na libovolné jméno y takové, které se nevyskytuje v konstrukci C . Výsledkem je konstrukce ekvivalentní s C .
- transparentní formulace pravidla *beta*-redukce z lambda kalkulu:

pravidlo *beta*-redukce:

- konstrukce $A::T$ nechť obsahuje volné proměnné

$$x_1::T_1, \dots, x_n::T_n$$

$$\forall \lambda x_1 \dots x_n (A) :: (TT_1 \dots T_n)$$

- definujeme operaci $A(x_1, \dots, x_n := B_1, \dots, B_n)$:

- ta v konstrukci A nahradí každý výskyt proměnné x_i konstrukcí $B_i::T_i$
- a je přípustná pouze tehdy, když nahrazení nevede k tomu, že některá z volných proměnných konstrukce B_i se po nahrazení stane vázanou proměnnou v B_i

pravidlo *beta*-redukce - pokračování

- Potom pro každou valuaci v :
konstrukce

$A(x_1, \dots, x_n := B_1, \dots, B_n)$

a konstrukce

$[(\lambda x_1 \dots x_n (A))B_1 \dots B_n]$

v -konstruuují týž T-objekt nebo jsou obě
 v -nevlastní

otevřené a uzavřené konstrukce

- Nechť A je konstrukce, která obsahuje volné proměnné. Potom A se nazývá *otevřená konstrukce*.
- Nechť A je konstrukce, která neobsahuje žádnou volnou proměnnou. Potom A se nazývá *uzavřená konstrukce*.
- To co konstruuji otevřené konstrukce, je vždy závislé na valuaci. Uzavřené konstrukce konstruuji objekty nezávisle na valuaci.
- Uzavřené konstrukce jsou vhodné pro modelování těch objektů, které chceme vnímat jako celé funkce, tj. kde pracujeme s funkcí jako takovou, nikoli s jejími jednotlivými hodnotami v závislosti na konkrétních argumentech
- Otevřené konstrukce jsou vhodné pro modelování toho, co je přirozené vnímat jako závislé na konkrétní valuaci: databázový stroj (obecně stroj = „engine“) který umožňuje manipulovat s daty

PŘÍKLADY

- logických operátorů
- matematických funkcí
- konstrukce množin
- funkcí ze života
- jak je to s nevlastností ekvivalentních konstrukcí ?

kvantifikátory, singularizátor

- $X::T$, B je Bool- konstrukce
- $\forall \Lambda^T::((T \rightarrow \text{Bool}) \rightarrow \text{Bool})$ je tzv. obecný kvantifikátor:
 $[\Lambda^T \lambda x B] = \text{Pravda}$, když $\lambda x B$ konstruuje právě T , jinak,
tj. když $\lambda x B$ konstruuje $T' \subset T$, je $[\Lambda^T \lambda x B] = \text{Nepravda}$
- $\forall^T::((T \rightarrow \text{Bool}) \rightarrow \text{Bool})$ je tzv. existenční kvantifikátor:
 $[\forall^T \lambda x B] = \text{Pravda}$, když $\lambda x B$ konstruuje neprázdnou
podmnožinu T , jinak Nepravda
- $I^T::((T \rightarrow \text{Bool}) \rightarrow T)$ je tzv. singularizátor:
 $[I^T \lambda x B] = t$, když $\lambda x B$ konstruuje jednoprvkovou
podmnožinu $\{t\} \subseteq T$, jinak je nedefinován
- místo $[\Lambda^T \lambda x B]$ píšeme $\forall x (B)$
místo $[\forall^T \lambda x B]$ píšeme $\exists x (B)$
místo $[I^T \lambda x B]$ píšeme $\iota x (B)$

reálný svět

- Mějme atribut (viz přednáška č. 3)
OdbDodZbozi =
odběratelé (#ODB) kterým daný dodavatel
(#DOD) dodává dané zboží (#ZBOZI)
:: (...((DOD, ZBOZI) → (ODB → Bool)))
- Jak vypadá jeho konstrukce:
x::DOD, z::ZBOZI, y::ODB, w,t jako
obvykle:

$$\lambda wt (\lambda xz (\lambda y ([OdbDodZbozi_{wt} (x,z)] y))))$$

jak je to s nevlastností ekvivalentních konstrukcí ?

- Nechť C_1 je ekvivalentní s C_2 . Nechť v je libovolná valuace.
- C_1 je v -nevlastní právě tehdy, když C_2 je v -nevlastní.
- Provedte důkaz.

otázky

- co je to matematika ?
- je matematika více o objektech nebo o konstrukcích
- co nám více pomůže při domlouvání? objekty nebo konstrukce?
- jsou v reálném světě objekty? jsou v něm konstrukce?
- jak je to s objekty a konstrukcemi v ideálních světech: jeden IdSvet versus dva a více IdSvetu, vytváření společné kultury
- kultura lidí vs. kultura lidí + matematických strojů

P114

Sémantika a její role

zaostření pozornosti

6

Témata

- Sémantika
- informace
- logické vyplývání
- informační schopnost
- zaostření pozornosti
- sortalizace
- báze sort
- definice jednoduchých typů

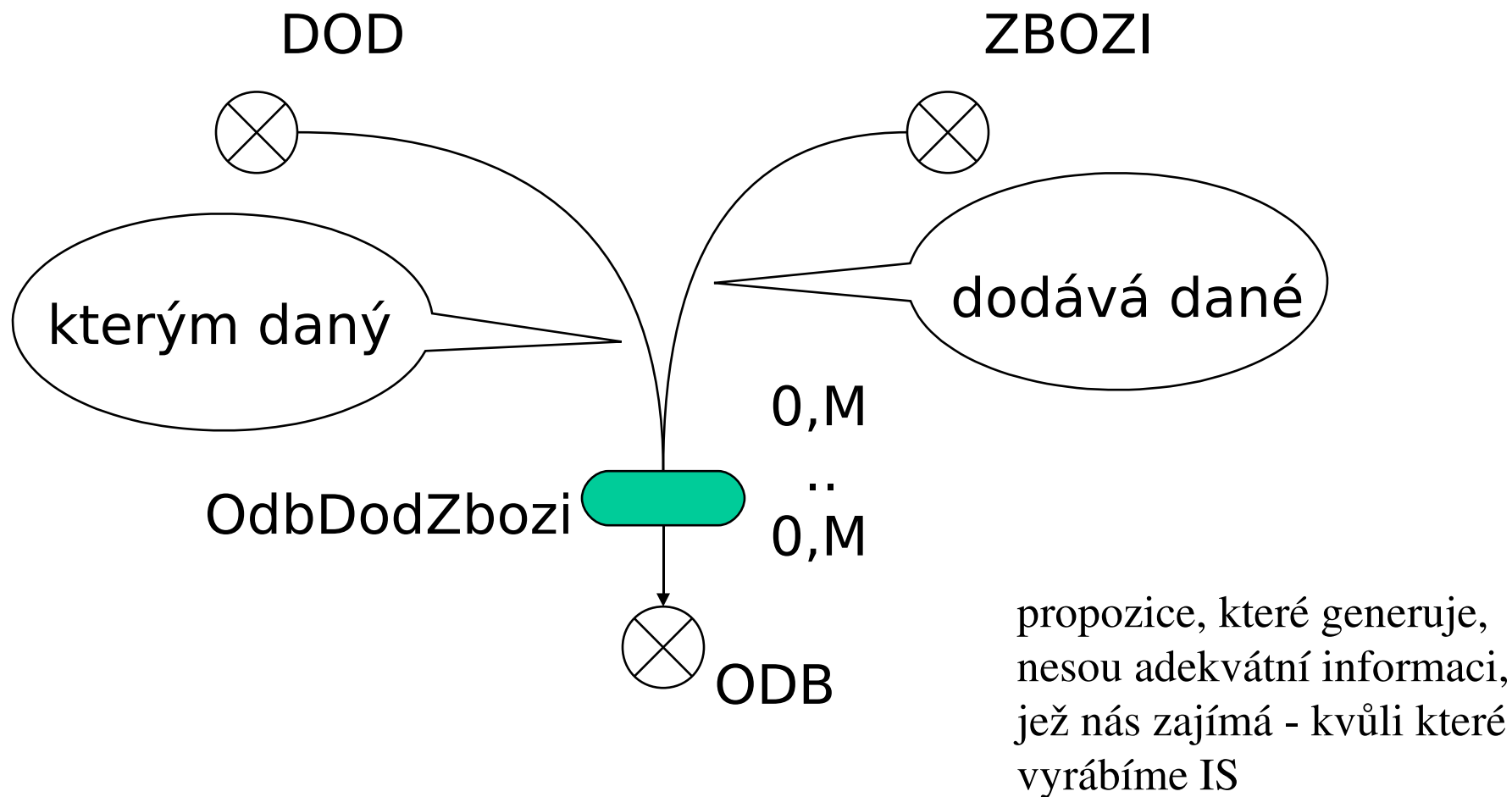
sémantika

- vše o čem hovoří matematická logika i celá matematika, lze vybudovat pohodlně nad $\mathbf{B} = \{\text{Bool}, \text{Univ}, \text{Tim}\}$...
- ... ale sémantika sdělení, používaných v přirozeném jazyce při popisu reálného světa, chybí
- sémantiku lze zahrnout do teorie podporující naše komunikace právě nad epistémickou bází
 $\mathbf{EB} = \{\text{Bool}, \text{Univ}, \text{Tim}, \text{Wrd}\}$
- je to tzv. sémantika možných světů
- ta je základem pro náš přístup k DM ...

sémantika a DM

- **běžný názor:** sémantiku nelze rozumně zachytit, proto se snažíme použít takové syntaktické prostředky, abychom na základě nich mohli alespoň vytušit sémantiku skrytou za našimi modely
autoři UML, specialisti na DB, DWH, ...
- proti tomu stavíme Datové Modelování metodou HIT a *základní tezi*:
„DM bez sémantiky je jako láska bez soulože ...“

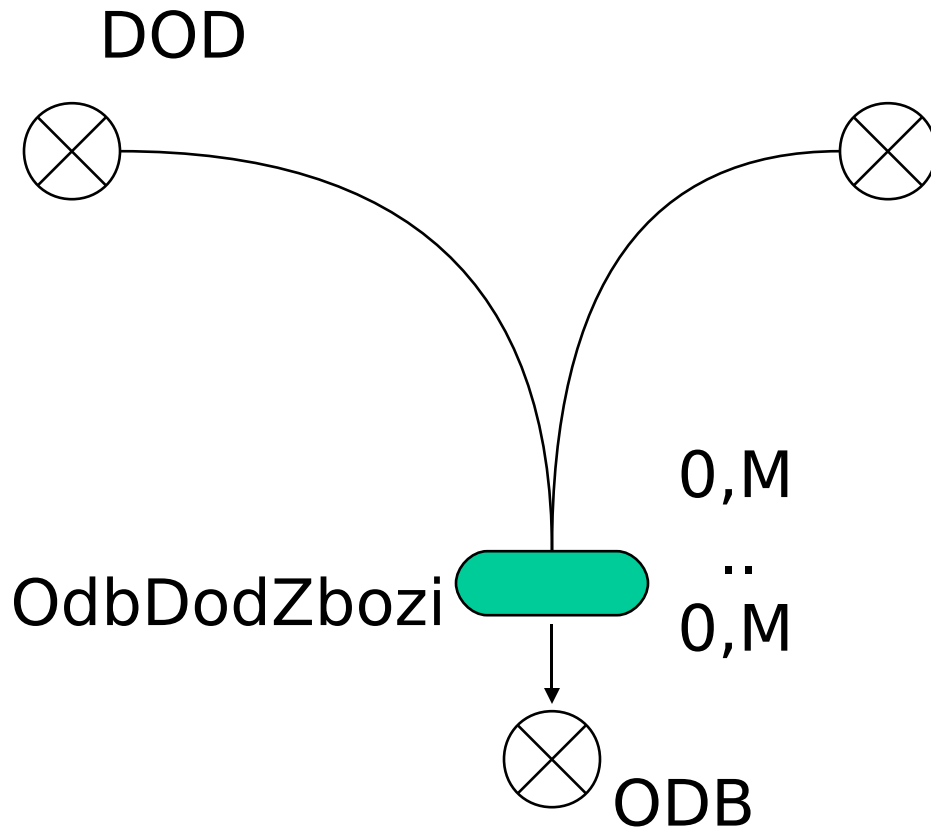
HIT-atribut: záznam sémantiky



Proč vyrábíme IS:

- chceme odpovědi na otázky typu:
 - kdo dodává jaké zboží do kterého obchodního řetězce?
 - komu co dodává daný dodavatel?
 - komu daný dodavatel dodává dané zboží?
 - ...
- Následující (1) a (2) jsou příklady propozic „generovaných“ atributem OdbDodZbozi:
 - (1) Kostelecké uzeniny dodávají jemné párky do prodejen Tesco a Makro.
 - (2) Dodavatel (MPK) dodává výrobek (Selský salám) do prodejen odběratele (Delvita).
- ... jsou to odpovědi na uvedené otázky

diagram bez sémantiky:



generuje propozice tvaru:
odběratelé (libovolným způsobem,
resp. z libovolného důvodu)
přiřazení ke dvojici dodavatel
a zboží

to není dostatečný důvod
pro tvorbu IS !!!

sémantika a informace

Novák	3600
Horák	4200
Máloberu	2800

Plat ?

alimenty ?

Vitální kapacita
plic ?

data, propozice, sémantika

- propozice explikují význam dat:
- zaměstnanec Novák platí alimenty 3 600 Kč
- zaměstnanec Horák má vitální kapacitu plic 4 200
- Tyto propozice dávají informaci, neboť snižují stupeň neurčitosti poznání reálného světa
- R. Carnap: informace obsažená v dané propozici je měřitelná počtem možných světů, které jsou touto propozicí vyloučeny (ve kterých nabývá pravdivostní hodnotu N)

Co je to informace

- základní přístupy

- Carnapovská informace: množství informace v propozici obsažené je dáno počtem možných světů pravdivostí dané propozice vyloučených
- Shanonovská informace: informace je míra snížení entropie (neurčitosti)
- čím více možných světů vyloučíme, tím více snížíme neurčitost

Carnapovská informace

- Nechť P je množina propozic a $R_1 \subseteq \text{Tim}$ je časový interval, q je nějaká propozice:

$$\text{Wrd}(P, R_1) = \{w \in \text{Wrd} \mid [[qw]t] = \text{Pravda} \wedge q \in P \wedge t \in R_1\}$$

$\text{Wrd}(P, R_1)$ se nazývá **přípustný logický prostor** množiny propozic P vzhledem k časovému intervalu R_1 .

- Čím je $\text{Wrd}(P, R_1)$ menší, tím více informace je v Carnapově smyslu podáno.
- $\text{Wrd}(P, R_1) = \text{Wrd}$... P nedává žádnou informaci

Carnapovská informace - pokračování

- Je-li $R_1 = \text{Tim}$, hovoříme o **přípustném logickém prostoru vzhledem k množině propozic P** a značíme $\text{Wrd}(P)$.
- Jestliže $\text{Wrd}(P) \neq \text{Wrd}$, pak $\text{Wrd}(P)$ je vlastní podmnožina Wrd , a říkáme, že množina propozic P dává (poskytuje) informaci.
- Jestliže $\text{Wrd}(P, R_1) \neq \text{Wrd}$, pak $\text{Wrd}(P, R_1)$ je vlastní podmnožina Wrd , a říkáme, že množina propozic P dává (poskytuje) informaci v časovém intervalu R_1 .

logické vyplývání

- rozdíl Implikace (\Rightarrow) vs. Logické vyplývání (\vdash)
- implikace (pravdivostní funkce) (BoolBoolBool)-objekt nezávislý na stavu světa
- Propozice B logicky vyplývá z propozice A, značíme $A \vdash B$, tehdy, když pro $\forall w \in W_{rd}, \forall t \in T_{im}$ platí $[[Aw]t] = Pravda \implies [[Bw]t] = Pravda$
- Jestliže $[[Aw]t] = Nepravda$ nebo je nedefinováno, pak o pravdivostní hodnotě propozice B nelze nic říci.

uspořádání podle množství podávané informace

- (1) Nechť P, Q jsou množiny propozic takové, že pro každé $q \in Q$ platí,
bud' $q \in P$ nebo
 $\exists p \in P$ tak, že $p \supset q$.
Potom pro přípustné logické prostory množin propozic P a Q platí $\text{Wrd}(P, R_1) \subseteq \text{Wrd}(Q, R_1)$ pro libovolné $R_1 \subseteq \text{Tim}$. (Důkaz plyne z definice logického vyplývání).
- (2) Říkáme, že množina propozic Q **dává nejvýše tolik informace, jako množina propozic P** , nebo že Q dává méně nebo stejně informace jako P .
- (3) Značíme $Q \angle_i P$. Relace \angle_i je částečné quasi-uspořádání na množině množin propozic. (Proveďte důkaz !)

Funkce Cn - důsledek

- P je množina propozic

$((\text{Wrd}, \text{Tim}) \rightarrow \text{Bool}) \rightarrow \text{Bool}$ - objekt

označme $((\text{Wrd}, \text{Tim}) \rightarrow \text{Bool}) = \text{Pr}$, pak $P/(\text{Pr} \rightarrow \text{Bool})$

- $\text{Cn} / (\text{Pr} \rightarrow \text{Bool}) \rightarrow (\text{Pr} \rightarrow \text{Bool})$

definovaná

$$\text{Cn} = \lambda p [\cup \lambda q (q \angle_i p)]$$

kde $p, q :: (\text{Pr} \rightarrow \text{Bool})$.

- Cn se nazývá funkce „důsledek“ a dává na každé množině propozic P **množinu všech možných jejich logických důsledků**

Tvrzení o C_n

- **Tvrzení 1:**

Operace (funkce) C_n je idempotentní:

$$[C_n[C_n P]] = [C_n P]$$

- Provedte důkaz!

- **Tvrzení 2:**

$Q \angle_i P$ právě když $[C_n Q] \subseteq [C_n P]$

- Provedte důkaz!

- **Důsledek:**

Množinu tříd propozic

$\{[C_n P] \mid P \text{ je libovolná množina propozic}\}$

lze částečně uspořádat.

- Provedte důkaz!

... více o C_n

- P_i třída propozic, $i = 1, \dots, n$
- $P_i \subseteq [C_n P_i]$
- „obrácená trojúhelníková nerovnost“:
$$\bigcup_{i=1}^n [C_n P_i] \subseteq [C_n (\bigcup_{i=1}^n P_i)]$$

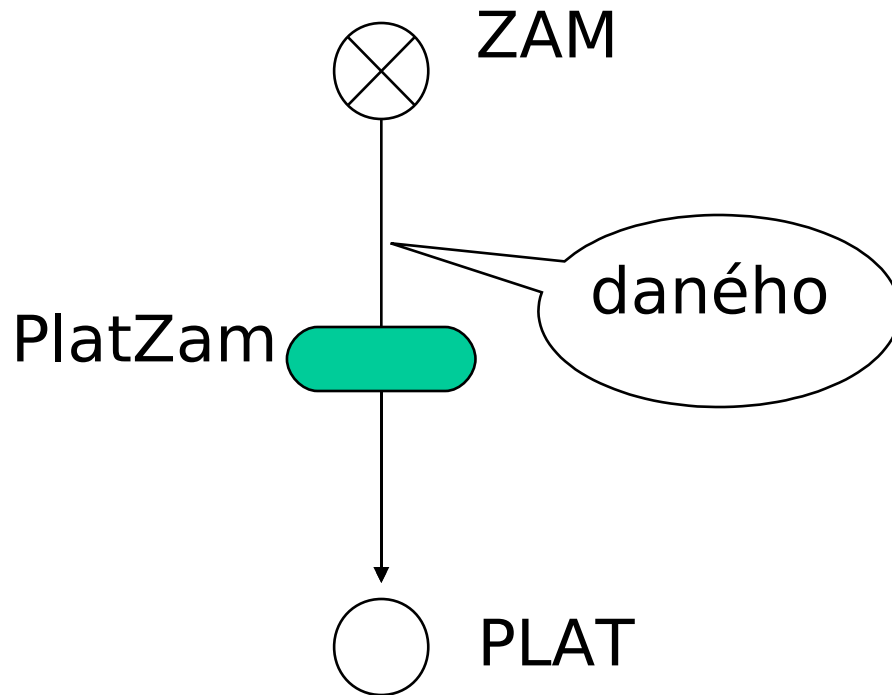
Provedte důkaz!

- ... abychom věděli co to znamená, když říkáme „... to je přece logické, když ...“

Informační schopnost IS

- **CO** se má udělat = specifikace provedení
 - jaké informace bude IS poskytovat = vymezení třídy dotazů nad daným IS zodpověditelných
- jak zadat třídu dotazů, které mají být daným IS zodpověditelné
- vytvořit seznam všech takových dotazů? **NE!**
- určit „bázi“ prostoru dotazů, které mají být zodpověditelné
- **Najít konstrukce, které generují propozice dávající právě požadované odpovědi**

konstrukce generující množiny propozic



- typ hodnoty funkce
- typ argumentů
- typ samotné funkce
- role argumentů
- sémantika přiřazení

- $W::Wrd, t::Tim, z::ZAM, x::PLAT$

$$\forall \lambda w \lambda t \lambda z \lambda x ([[[PlatZam(w)]t]z]=x)$$

(x je ale jediné - nutno použít singularizátor)

$$\forall \lambda w \lambda t \lambda z \iota x ([[[PlatZam(w)]t]z]=x) \quad (k)$$

- Konstrukce (k) generuje všechny možné propozice tvaru :

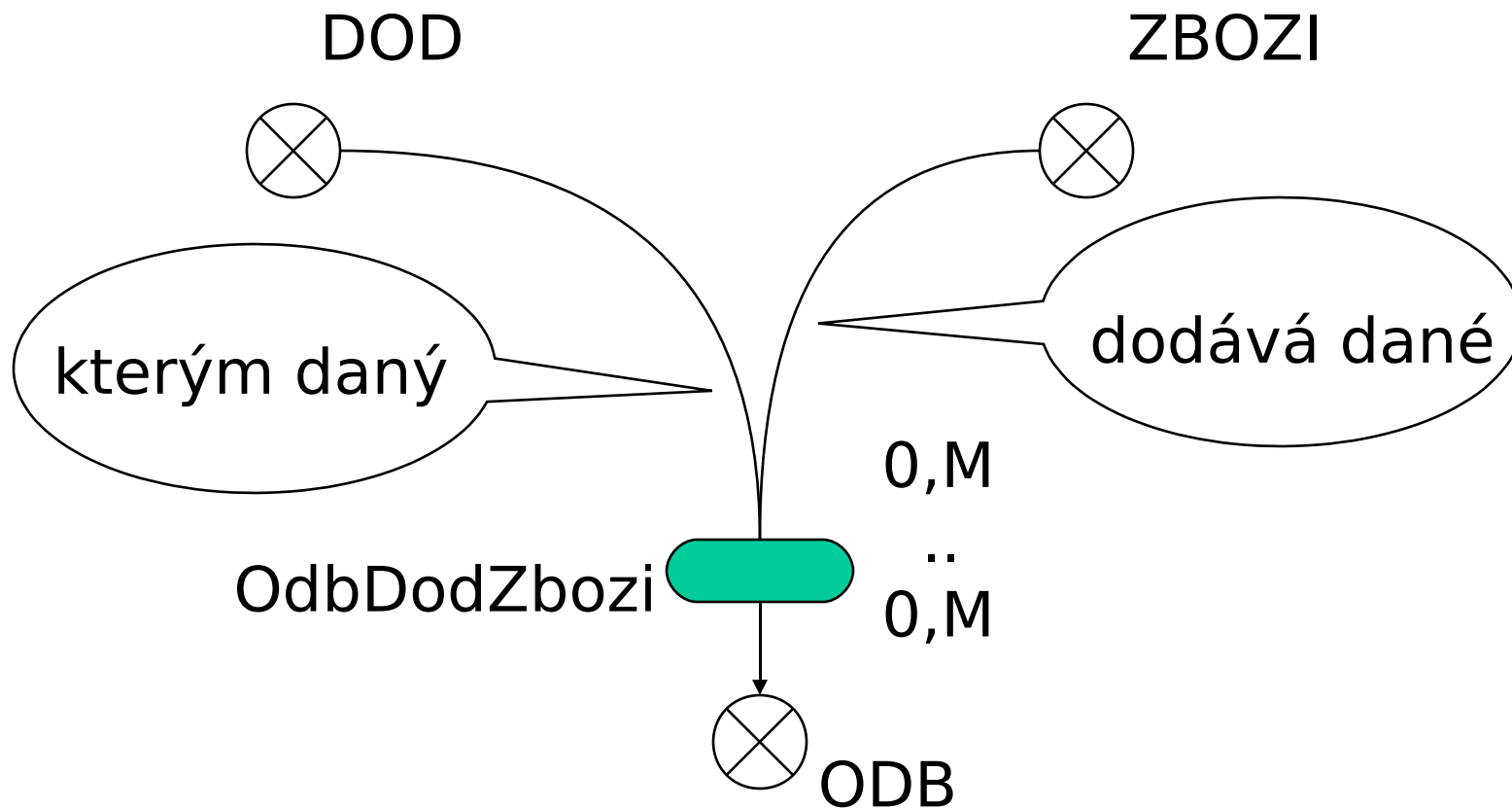
„Zaměstnanec Novák má plat 15 000,-Kč“

„Zaměstnanec Mach má plat 17 500,-Kč“

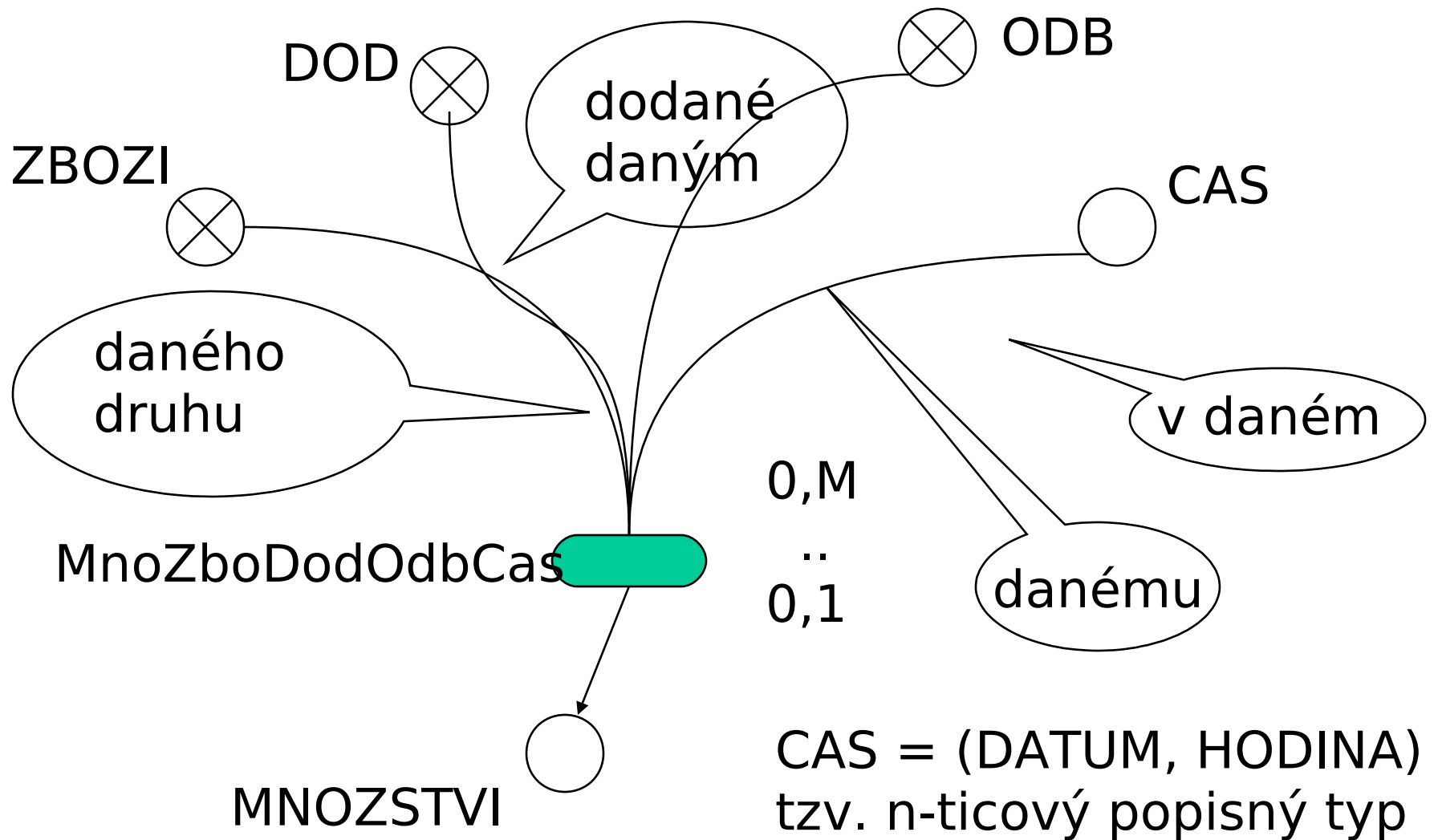
....

„Existuje zaměstnanec s platem nad 16 000,-Kč.“

? jaké propozice generuje atribut:



? jaké propozice generuje atribut:



Bázové propozice generované atributem

- A je atribut,
A / ((Wrd, Tim) \rightarrow (T \rightarrow S)) nebo
A / ((Wrd, Tim) \rightarrow (T \rightarrow (S \rightarrow Bool)))
T, S jsou entity nebo deskripce, resp. jejich n-tice
-- viz přednáška 3
- Množina bázových propozic generovaná atributem A ve stavu světa W / (Wrd, Tim) je definována takto:
$$BP(A)^W = \lambda p (\exists x (\exists y ([A \ W] \ x] = y \wedge$$
$$\wedge p = (\lambda w [[A \ w] \ x] = y))))),$$

kde $p::Pr$, $w::(Wrd, Tim)$, $x::T$, $y::S$ nebo $y::(S \rightarrow Bool)$

Informační kapacita atributu

- Informační kapacitou atributu A ve stavu světa W nazýváme množinu všech propozic generovaných atributem A ve W , tj. množinu $P(A)^W$ všech logických důsledků báзовých propozic generovaných atributem A ve W :

$$P(A)^W = [Cn \ B P(A)^W]$$

- Informační kapacita množiny atributů $\{A_1, \dots, A_n\}$ ve W je dána množinou všech logických důsledků propozic generovaných atributy A_i :

$$P(A_1, \dots, A_n)^W = [Cn \ \cup_{i=1}^n P(A_i)^W]$$

Důsledky

- Informační kapacity (množin) atributů lze částečně uspořádat. -- proveďte důkaz!
- ... jsme schopni porovnávat informační schopnosti databází a IS

... zpět k propozicím generovaným PlatZam:

- Zaměstnanec Novák má plat 15 000,-Kč./ ((BoolTim)Wrd)
(propozice)
- Zaměstnanec / (((BoolUniv)Tim)Wrd)
(vlastnost individuí)
- Novák / Univ
(jméno individua jako nálepka)
- má plat / (((TimUniv)Tim)Wrd)
(fce z individuí do reálných čísel v závislosti na možném světě a čase)
- Propozice říká
 - 1) že Novák je zaměstnanec a
 - 2) že má plat a
 - 3) že ten plat je 15 000,-
- **... stavět konstrukce přímo nad epistémickou bází je pragmaticky neúnosné !** P114_6

„Zaostření“ pozornosti

- proces modelování (zájmové části) světa se podobá zaostřování při fotografování: před našim objektivem je vše, my ale snímkem zobrazíme jasně pouze to, na co jsme zaostřili
- To znamená:
 - nad **EB** vybereme určité typy a na ty zaostříme pozornost
 - modely pak obsahují pouze konstrukce těchto „zaostřených“ typů
- Zaostřené typy jsou: funkce tzv. jednoduchých typů (viz přednáška 3) a funkce definující tzv. sorty (viz dále)

příklady sort

Sorta ZAMESTNANEC:

je třída (extenze) individuí, kterou v určitém časovém intervalu R a v aktuálním světě w_a generuje konstrukce $\lambda w \lambda t ([[\text{Být_zaměstnancem}(w)] t])$, kde $\text{Být_zaměstnancem} :: (((\text{BoolUniv})\text{Tim})\text{Wrd})$

Sorta PLAT:

je třída (extenze) čísel konstruovaná konstrukcí $\lambda r ([\text{Plat}(r)] = \text{Pravda})$, kde $\text{Plat} :: (\text{BoolTim})$

definice deskriptivní sorty

Deskriptivní sortou (popisnou sortou) je každá algoritmicky vyčíslitelná množina.

- Je to tedy rekursivní množina.
- Je to množina reprezentovatelná na počítači.

Je vždy dána nějakou definicí, která umožňuje rozhodnout, zda daný řetěz znaků resp. dané číslo patří do této sorty či nikoliv. Toto rozhodování je nezávislé na stavu světa (na Wrd a Tim).

Tím jsme mezi objekty našeho zkoumání zařadili i prvky jazyka, kterým se vyjadřujeme - zejména názvy konkrétních objektů jako „nálepky“.

definice entitní sorty

Nechť $R \subseteq \text{Tim}$ je rozumné časové okolí (bylo-je-bude) přítomnosti. Nechť $r \in R$ je časový okamžik a w_a je aktuální svět.

Nechť T_1, \dots, T_m jsou ne nutně různé typy nad **EB**.

Nechť $P_i / (((\text{Bool}T_i)\text{Tim})\text{Wrd})$ jsou konkrétní vlastnosti přisouditelné T_i -objektům.

Označme $C(P_i, r, w_a)$ třídu T_i -objektů generovanou vlastností P_i v daném časovém okamžiku r a aktuálním světě w_a .

Potom:

$\bigcup_{r \in R} C(P_i, r, w_a)$ je **entitní sorta** definovaná vlastností P_i .

$\bigcup_{i=1..m} \bigcup_{r \in R} C(P_i, r, w_a)$ je **entitní sorta** definovaná disjunkcí vlastností P_i , $i=1, \dots, m$.

$\bigcap_{i=1..m} \bigcup_{r \in R} C(P_i, r, w_a)$ je **entitní sorta** definovaná konjunkcí vlastností P_i , $i=1, \dots, m$.

sortalizace

- výběr vhodných entitních a deskriptivních sort pro popis zájmového výseku reality
 - definice entitních sort
 - popis hodnot deskriptivních sort
 - sortalizace je to co jiní autoři nazývají klasifikací, tj. výběr základních tříd, nad nimiž budeme operovat : jak při modelování, tak při práci s databázovým systémem
-
- Uvědomme si, že entitní sorty jsou extenze!
 - ...stejně jako prvky každé báze

Jednoduché typy, HIT-atributy

- HIT-atributy (funkční závislosti) jsou objekty tzv. **jednoduchých typů**
 - (a) $((((S_1, \dots, S_m)(T_1, \dots, T_n))\text{Tim})\text{Wrd})$
 - (b) $((((\text{Bool}(S_1, \dots, S_m))(T_1, \dots, T_n))\text{Tim})\text{Wrd})$pokud platí, že alespoň jedna ze sort S_i nebo T_i je entitní sorta.
- číslo $m+n$ se nazývá složitost HIT-atributu
- **HIT-atributy** konstruujeme nad tzv. bází sort **BS**. Sorty viz definice entitních a deskriptivních sort. **BS** je určena tak, že pragmaticky odpovídá našemu konkrétnímu zájmu.

vztah sort k modelování

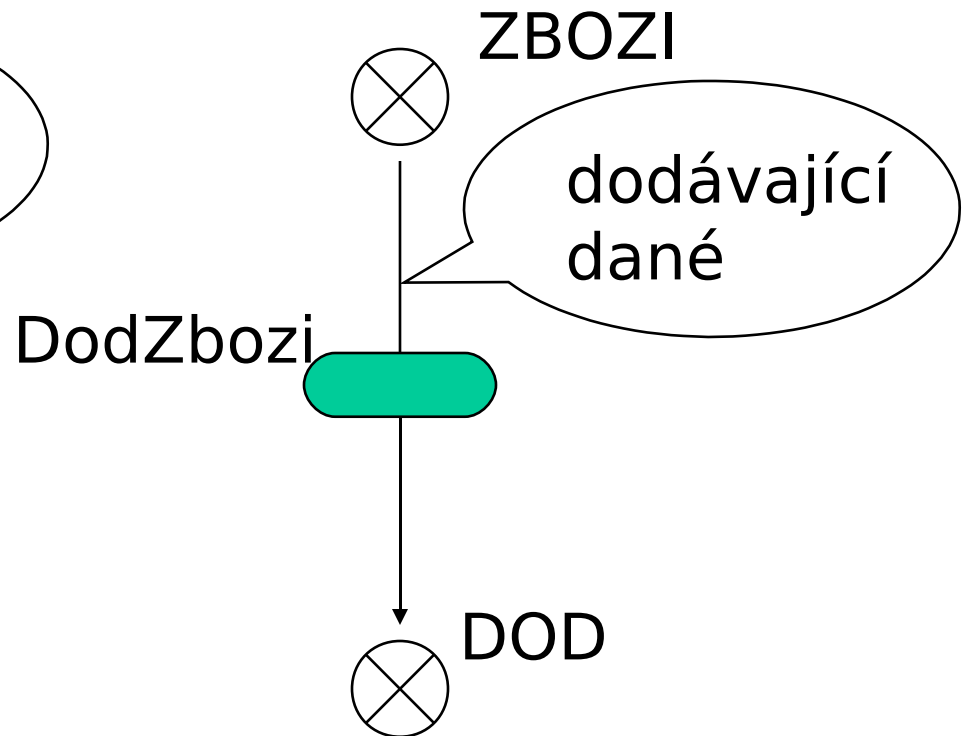
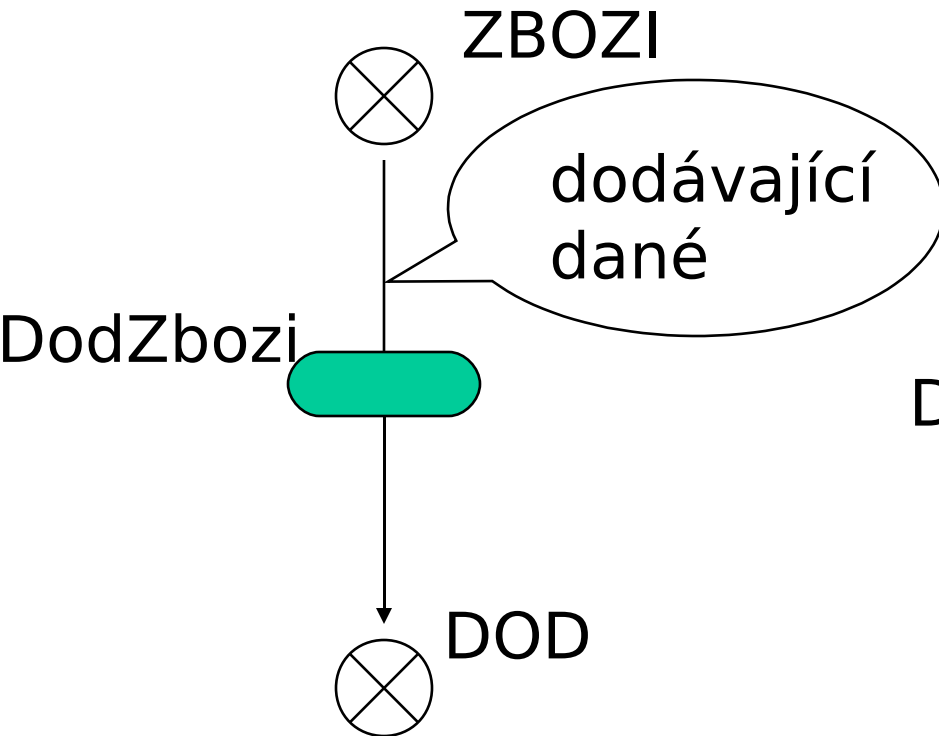
- **entitní a deskriptivní sorty** jsou objekty zájmu, o kterých hovoří uživatelé IS, a ze kterých bereme hodnoty funkcí a jejich argumentů při popisu výseku reálného světa
- jejich podmnožiny jsou definiční obory a obory hodnot funkcí uložených formou tabulek **v databázích**
- **v modelech** nás zajímají **konstrukce** těch **typů** (tj. funkcí nad **EB**), které „v rozumném časovém okolí přítomnosti“ a v aktuálním světě poskytují tyto sorty (pro případ entitních sort), resp. poskytují tyto sorty nezávisle na stavu světa (pro případ deskriptivních sort)

Příklady k procvičení

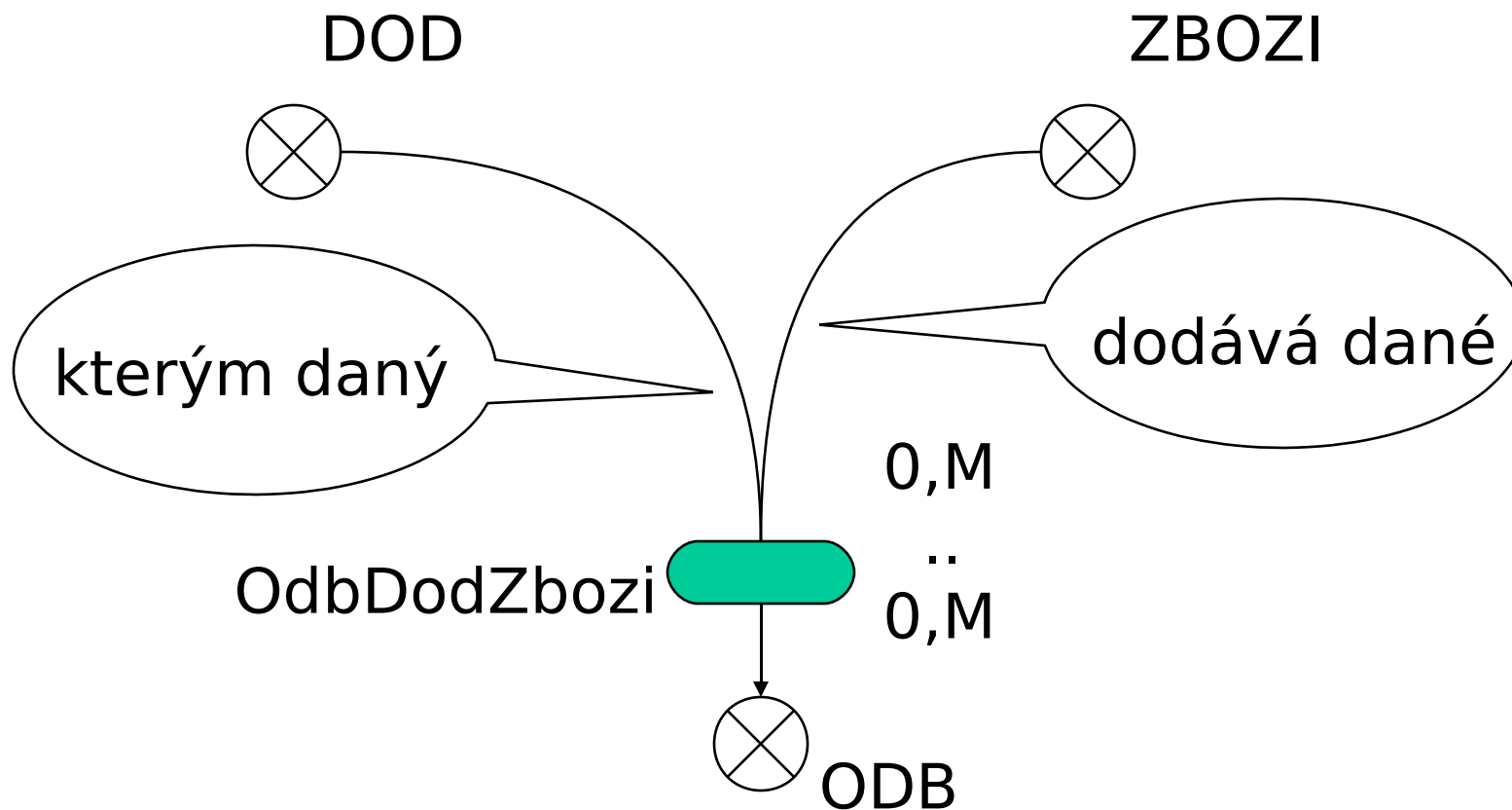
Porovnejte informační kapacitu:

které báze propozice generují?

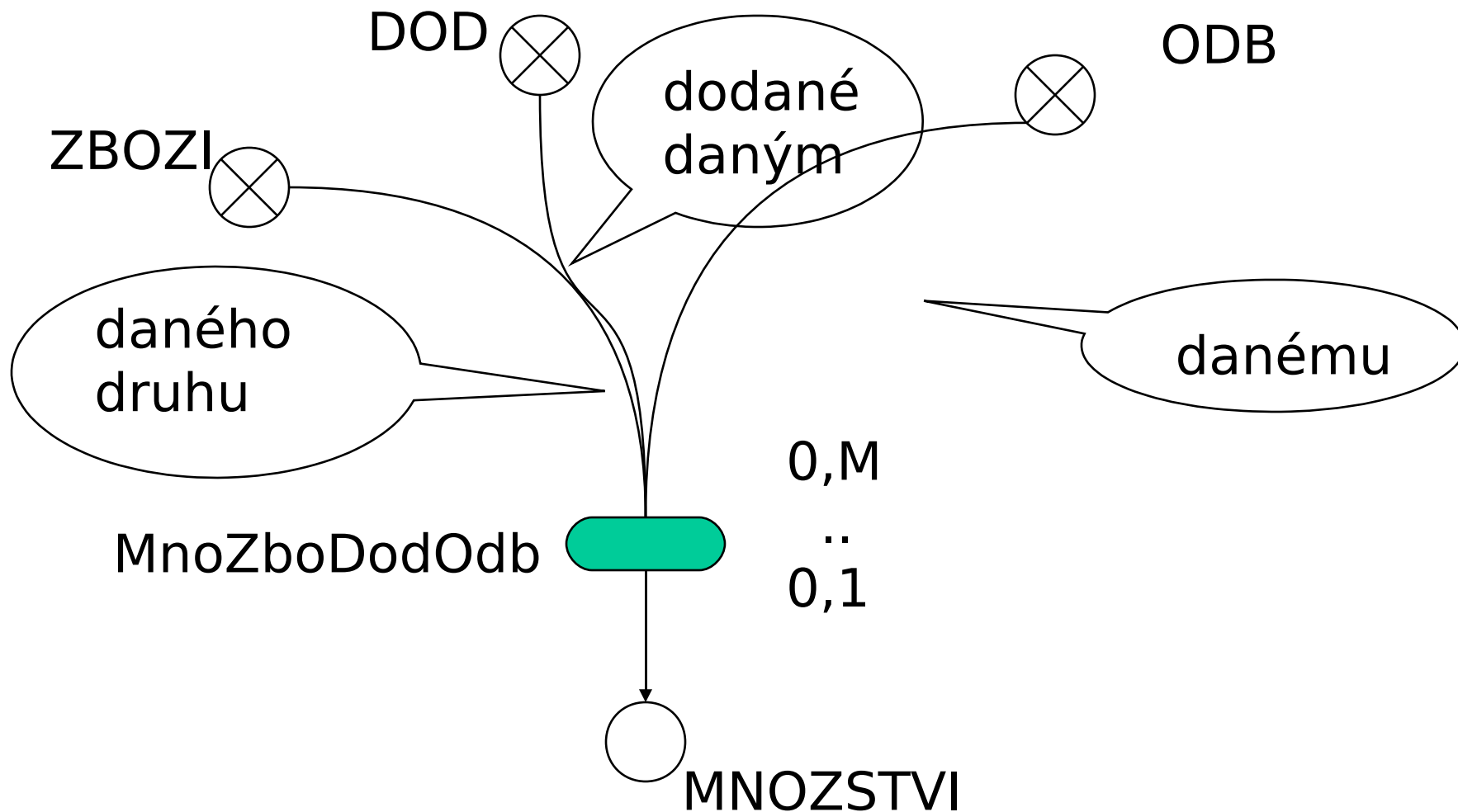
které propozice generují?



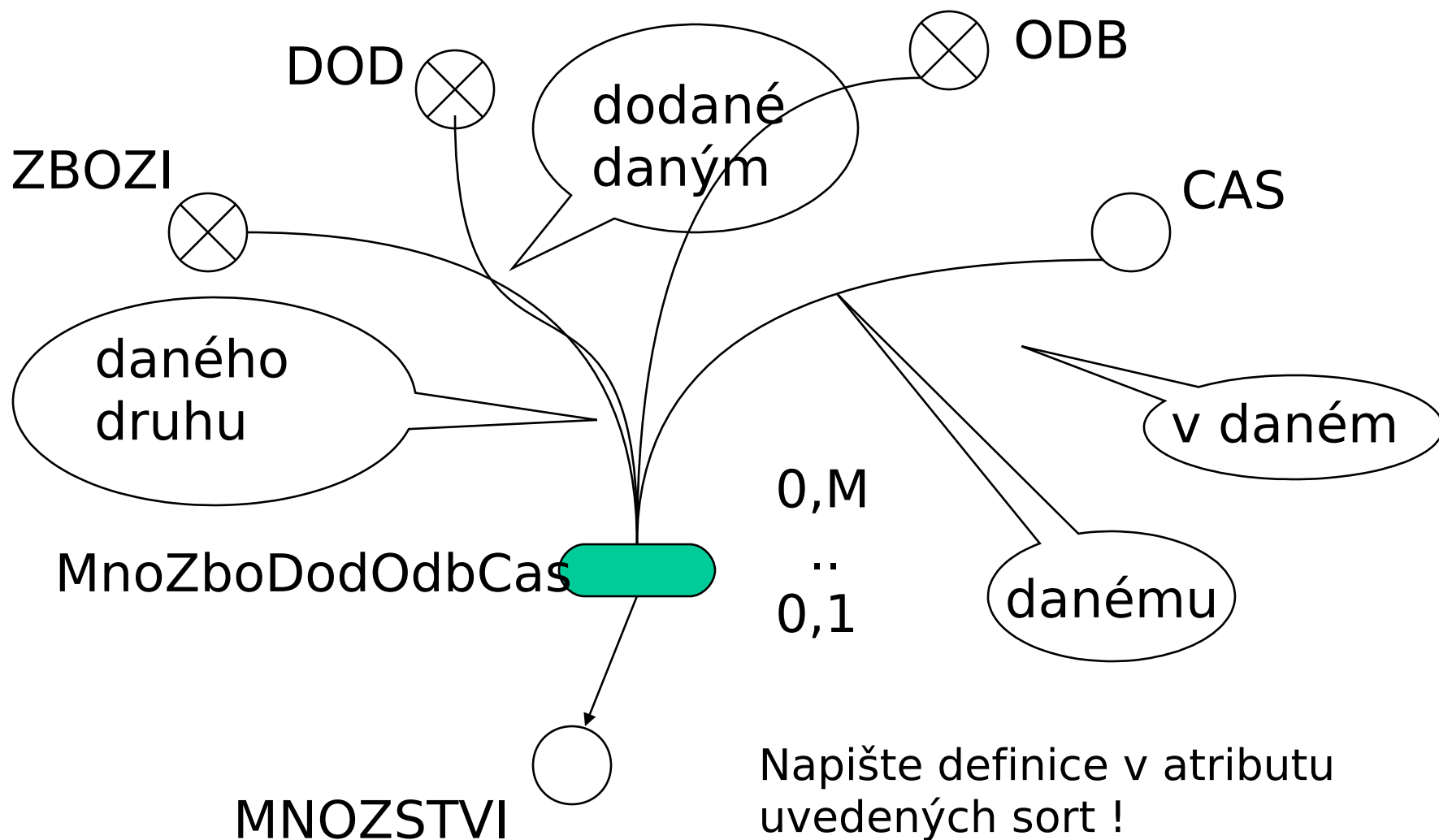
které bázové propozice generuje?
které propozice generuje?



které bázové propozice generuje?
které propozice generuje?



které bázové propozice generuje?
 které propozice generuje?



P114

Pojmy metody HIT

sortalizace, základní a jednoduché
typy

7

Témata

- zaostření pozornosti a báze základních typů
- definice bazových typů
- HIT-atributy
- sémantika HIT-atributů
- integritní omezení
- poměr HIT-atributu
- podstata modelování

„Zaostření“ pozornosti

- proces modelování (zájmové části) světa se podobá zaostřování při fotografování: před našim objektivem je vše, my ale snímkem zobrazíme jasně pouze to, na co jsme zaostřili
- To znamená:
 - nad **EB** vybereme určité typy a na ty zaostříme pozornost
 - modely pak obsahují pouze konstrukce těchto „zaostřených“ typů
- Zaostřené typy jsou: funkce tzv. jednoduchých typů (viz přednáška 3) a funkce definující tzv. sorty (viz dále)

vztah sort k modelování

- **entitní a deskriptivní sorty** jsou objekty zájmu, o kterých hovoří uživatelé IS, a ze kterých bereme hodnoty funkcí a jejich argumentů při popisu výseku reálného světa
- jejich podmnožiny jsou definiční obory a obory hodnot funkcí uložených formou tabulek **v databázích**
- **v modelech** nás zajímají **konstrukce** těch **objektů** (tj. funkcí nad **EB**), které „v rozumném časovém okolí přítomnosti“ a v aktuálním světě poskytují tyto sorty (*pro případ entitních sort*), resp. poskytují tyto sorty nezávisle na stavu světa (*pro případ deskriptivních sort*)

... zopakování

- **EB** = {Bool, Univ, Tim, Wrd}
- Typ nad **EB**:
 - Bool, Univ, Tim, Wrd
 - jsou-li T_1, \dots, T_n typy nad **EB**, pak (T_1, \dots, T_n) je typ nad **EB**
 - jeli ještě T typ nad **EB**, pak $((T_1, \dots, T_n) \rightarrow T)$ je typ nad **EB**
- Nad **EB** máme celou hierarchii typů
- jejich prvky jsou vesměs funkce
- tyto prvky umíme identifikovat pomocí konstrukcí
 - atomických (přesněji atomických + „trivializace“)
 - aplikace
 - n-tice, projekce
 - abstrakce
- pro obecné manipulace máme modifikovaný typovaný lambda-kalkul

Princip zaostření pozornosti (1)

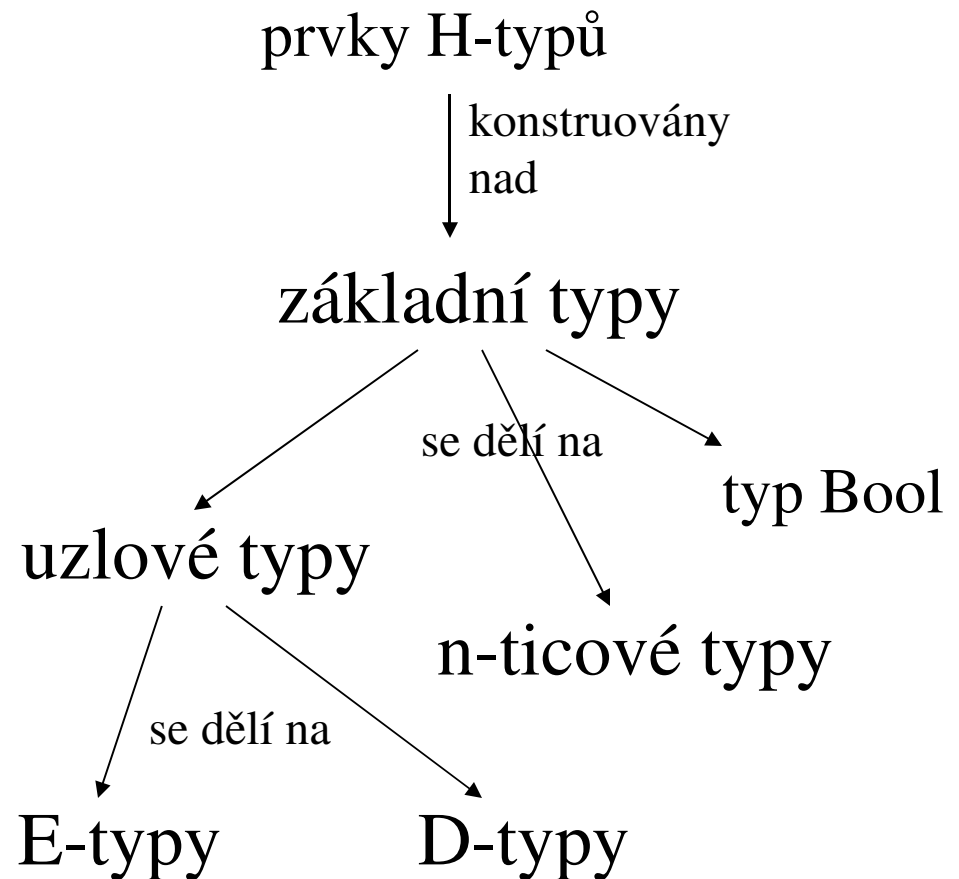
- Nad **EB** *definujeme* tzv. **uzlové typy**, kterými jsou entitní nebo deskriptivní sorty
- *POZOR!!! Ve slovním spojení **uzlové typy** má nyní slovo **typ** jiný význam než v definici typů nad bází B resp. nad epistémickou bází.*
- Přidáme bazový typ Bool a n-ticové typy, čímž získáme množinu **základních typů**
- *Provádíme přesun od epistémické báze k „bázi základních typů“, která je tvořena něčím jiným než původními typy a dokonce ani nemá vlastnost báze v původním smyslu !!!*

Princip zaostření pozornosti (2)

- Nad základními typy *konstruujeme* HIT- atributy, jako **prvky** tzv. jednoduchých typů
- TEDY: v hierarchii typů nad **EB** provedeme **zaostřením pozornosti** jakousi redukci, a jednotlivé prvky (objekty) konstruujeme jenom nad základními typy;
konstrukcí prvků základních typů se nezabýváme ! Nahrazujeme ji definicemi !
- Podrobněji viz DM2

modifikovaná hierarchie typů pro DM

- E-typy, D-typy
- uzlové typy
- n-ticové typy
- základní typy
- prvky H-typů



sortalizace

- Určení základních typů pro danou zájmovou oblast, kterou chceme modelovat, se nazývá **sortalizace**.
- Název souvisí s bází sort BS - viz minulá přednáška. Sortalizace je to, co jiní autoři nazývají klasifikací, či výběrem základních tříd a pod.
- Sortalizaci provádíme procesem „zaostření pozornosti“

Entitní typy (E-typy)

- E-typem nazýváme každou entitní sortu (extenzi) definovanou „v rozumném časovém okolí přítomnosti“ a v aktuálním světě pomocí vlastností nad **EB**, tj. pomocí intenzí
- Samotný E-typ je extenze !
- typicky, ale ne vždy, bývá E-typ podmnožinou Univ (viz dále)
- E-typy jsou **nedisjunktní** a typově-teoreticky **polymorfní**
- ZBOZI, DODAVATEL, ODBERATEL, ZAMESTNANEC, ...

E-typy (pokračování)

- Při praktickém použití jsou E-typy často třídy individuí, které vnímáme závisle na stavu světa (na wt)
- Ale mohou to být třídy libovolně složitých typů T nad **EB** (např. DODÁVKA, VÝPUJČKA, ALGORITMUS, DRUH VÝROBKU, ...)
- Abychom se dorozuměli (veškeré modelování děláme pro zlepšení komunikace !), **musíme každý E-typ definovat**
- Pro definici E-typu používáme vždy takového pojmového systému, který pokládáme za srozumitelný těm, kterým definici předkládáme (o tom je třeba učinit dohodu na tzv. formovacím semináři)
- Praktický problém je ve vnímání množiny Univ: říkáme sice, že Univ je nám dána a priori, ale jak se postupně „učíme“ rozumět určitému prostředí, tak se naše vnímání množiny Univ obohacuje o nové prvky, které jsme v předchozím stádiu učení ještě nevnímali.

definice E-typů

- Prvkem typu (#jmeno E-typu) je každý (takový objekt), pro který platí ...
- Příklady (pragmaticky zjednodušené):
 - Objektem typu (#Artikl) je každý produkt nebo služba nebo právo, který může být předmětem nákupu či prodeje a to včetně produktů, služeb nebo práv dosud neexistujících, ale potenciálně vytvořitelných pro účely rozvojových aktivit obchodní společnosti.
 - Objektem typu (#Dokument) je každý záznam nebo zpráva, jehož/jejíž zaznamenání má pro organizaci smysl.
 - Objektem typu (#Business Partner) je každé takové individuum, které je, bylo nebo může být účastno obchodních aktivit naší společnosti a které je zajímavé z pohledu rozvojových aktivit naší společnosti.

Deskriptivní typy (D-typy)

- D-typem je každá analytická funkce, která poskytuje deskriptivní sortu
- obvykle charakteristická funkce ($T \rightarrow \text{Bool}$), kde $T = \text{Tim}$ nebo $T = \text{Univ}$, při čemž do Univ zahrnujeme všechny možné řetězce znaků reprezentovatelné na počítači
- D-typy jsou vždy extenze
- DATUM, PLAT, RODNE CISLO, TEL-CISLO, JMENO, ...
- každý D-typ použitý v modelu musí být definován

definice D-typu

- Prvkem typu (jmeno D-typu) je každý řetězec znaků (každé číslo), který splňuje podmínky ...
- Příklady:
 - Prvkem typu (Datum) je každý řetězec číslic 8 znaků dlouhý, který má tvar RRRRMMDD, kde RRRR je číslo roku, MM je číslo dvouciferné měsíce a DD je dvouciferné číslo dne v měsíci.
 - Prvkem typu (Jmeno) je každý maximálně 45 znaků dlouhý řetězec písmen a znaků „-“, „.“, „ „, který začíná písmenem, a ve kterém v každé dvojici sousedních znaků je alespoň jedno písmeno.
 - Prvkem typu (Mnozstvi) je každé přirozené číslo nebo nula.
 - ...

n-ticové typy

- n-ticový typ je každý typ tvaru (D_1, \dots, D_m) , kde každé D_i je D-typ (pragmaticky zúžená definice)
- D-typy v n-ticovém typu nemusí být nutně různé
- n-ticové typy jsou extenze
- POZN.: někteří autoři připouštějí i n-ticové typy, ve kterých se jako komponenty vyskytují i E-typy. To je pro modelování zbytečné (vždy lze obejít vhodnou konstrukcí prvku H-typu) a v praxi spíše nevýhodné. Avšak pro formulace a důkazy obecných tvrzení je potřebné n-ticový typ brát v tomto širším smyslu.
- n-ticový typ má svoje jméno (jako každý jiný základní typ)
- OBDOBÍ(OD, DO), ADRESA(PSC, MESTO, ULICE, CISLO_DOMU)

definice n-ticového typu

- $T = (D_1, D_2, \dots, D_m)$,
kde
 D_1, D_2, \dots, D_m
již byly definovány jako D-typy
- $Adresa = (PSC, Mesto, Ulice, Cislo_domu)$,
(**Mesto** znamená jméno města, **Ulice** znamená jméno ulice)

H-typy

- T_1, \dots, T_n nechť jsou uzlové (ne nutně různé) typy, z nichž alespoň jeden je E-typem.
- S nechť je libovolný základní typ
- H-typem je každý typ formy
 $(Wrd \rightarrow (Tim \rightarrow ((T_1, \dots, T_n) \rightarrow S)))$
nebo
 $(Wrd \rightarrow (Tim \rightarrow ((T_1, \dots, T_n) \rightarrow (S \rightarrow Bool))))$
- ve druhém případě S nesmí být Bool

HIT-atributy (v TIL s jtt)

- Objekt A nechť je nějakého H-typu

tj.

$A / (\text{Wrd} \rightarrow (\text{Tim} \rightarrow ((T_1, \dots, T_n) \rightarrow S)))$

resp.

$A / (\text{Wrd} \rightarrow (\text{Tim} \rightarrow ((T_1, \dots, T_n) \rightarrow (S \rightarrow \text{Bool}))))$

- potom objekt A nazýváme HIT-atribut

- Každý HIT-atribut je intenze

- HIT-atribut A je konstruován konstrukcí

$\lambda_{\text{wt}} \lambda_{x_1 \dots x_n} \iota y ([A_{\text{wt}}(x_1, \dots, x_n)] = y)$ v prvním případě

a

$\lambda_{\text{wt}} \lambda_{x_1 \dots x_n} \lambda y [[A_{\text{wt}}(x_1, \dots, x_n)] y]$ ve druhém případě

HIT-atributy - pokračování

- A/T , kde T je H-typ

tj.

$$A/ (Wrd \rightarrow (Tim \rightarrow ((T_1, \dots, T_n) \rightarrow S)))$$

resp.

$$A/ (Wrd \rightarrow (Tim \rightarrow ((T_1, \dots, T_n) \rightarrow (S \rightarrow Bool))))$$

- číslo $n+1$ nazýváme složitost atributu
- Říkáme, že HIT-atributy jsou konstruovány nad bází základních typů **BZT**.
- obecně píšeme, že HIT-atribut je konstruován konstrukcí
 $\lambda_{wt} \lambda_{x_1 \dots x_n} \rho y ([A_{wt}(x_1, \dots, x_n)] * y)$
kde ρ stojí namísto λ nebo ι ,
a $*$ stojí namísto aplikace nebo identity

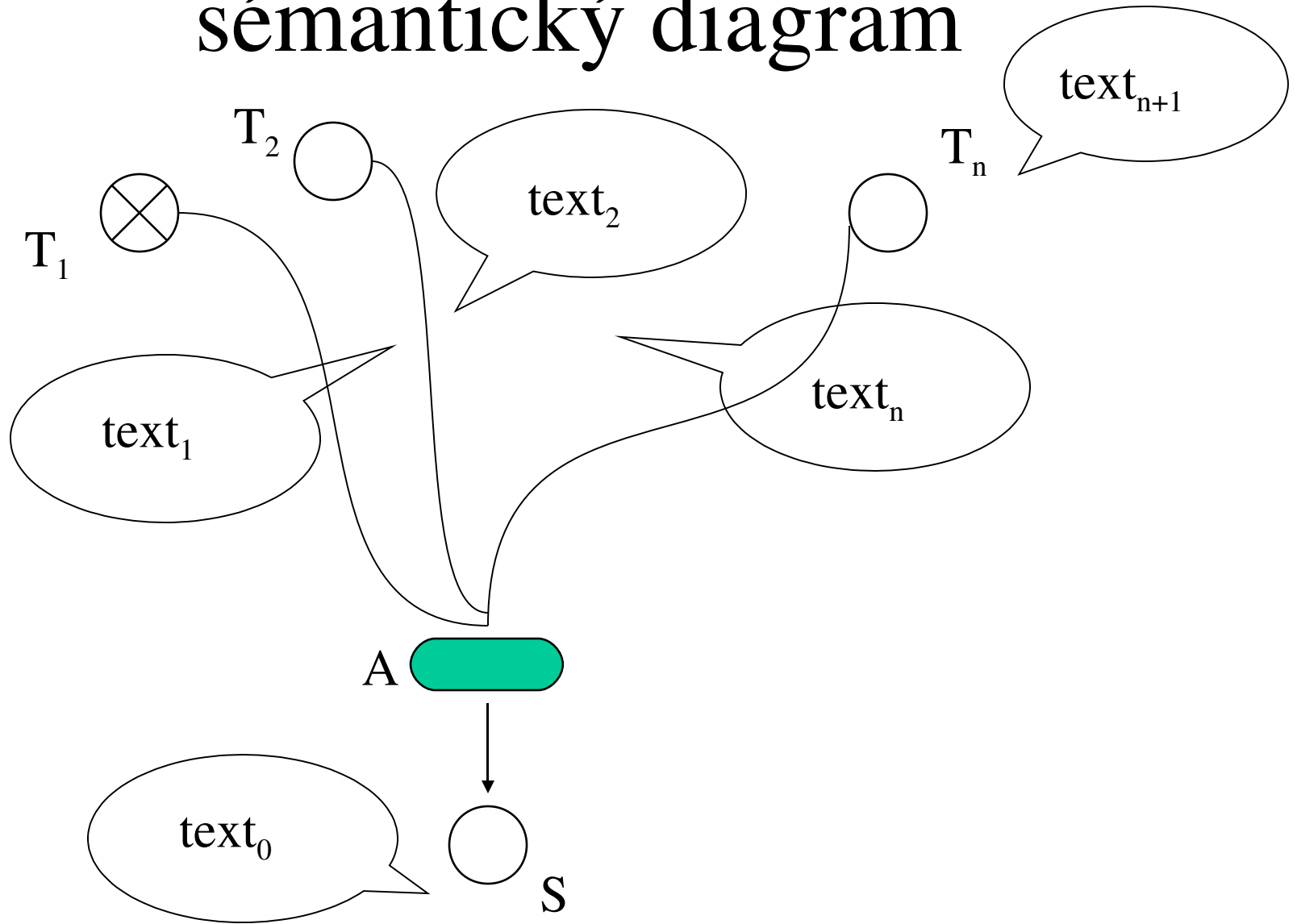
zápis sémantiky

- povahu matematické funkce vyjádříme výrazem umělého jazyka
- znalost interpretace symbolů tohoto umělého (matematického) jazyka nám umožňuje rozumět sémantice matematické funkce
- **povahu funkce** popisující svět, tj. povahu HIT-atributu vyjádříme **výrazem přirozeného jazyka**
- znalost komunikace v přirozeném jazyce nám umožňuje rozumět sémantice těchto funkcí

zápis sémantiky (2)

- $A / (\text{Wrd} \rightarrow (\text{Tim} \rightarrow ((T_1, \dots, T_n) \rightarrow S)))$
- $A = \text{text}_0(S) \text{text}_1(T_1) \text{text}_2(T_2) \dots \text{text}_n(T_n) \text{text}_{n+1}$
- kde pouze text_0 a text_{n+1} mohou být vynechány **a celý výraz (čtený jako část věty přirozeného jazyka) nám sděluje funkci, která dává hodnoty z S na argumentech z n-tice (T_1, \dots, T_n)**
- tomuto jednoznačně odpovídá (sémantický) diagram

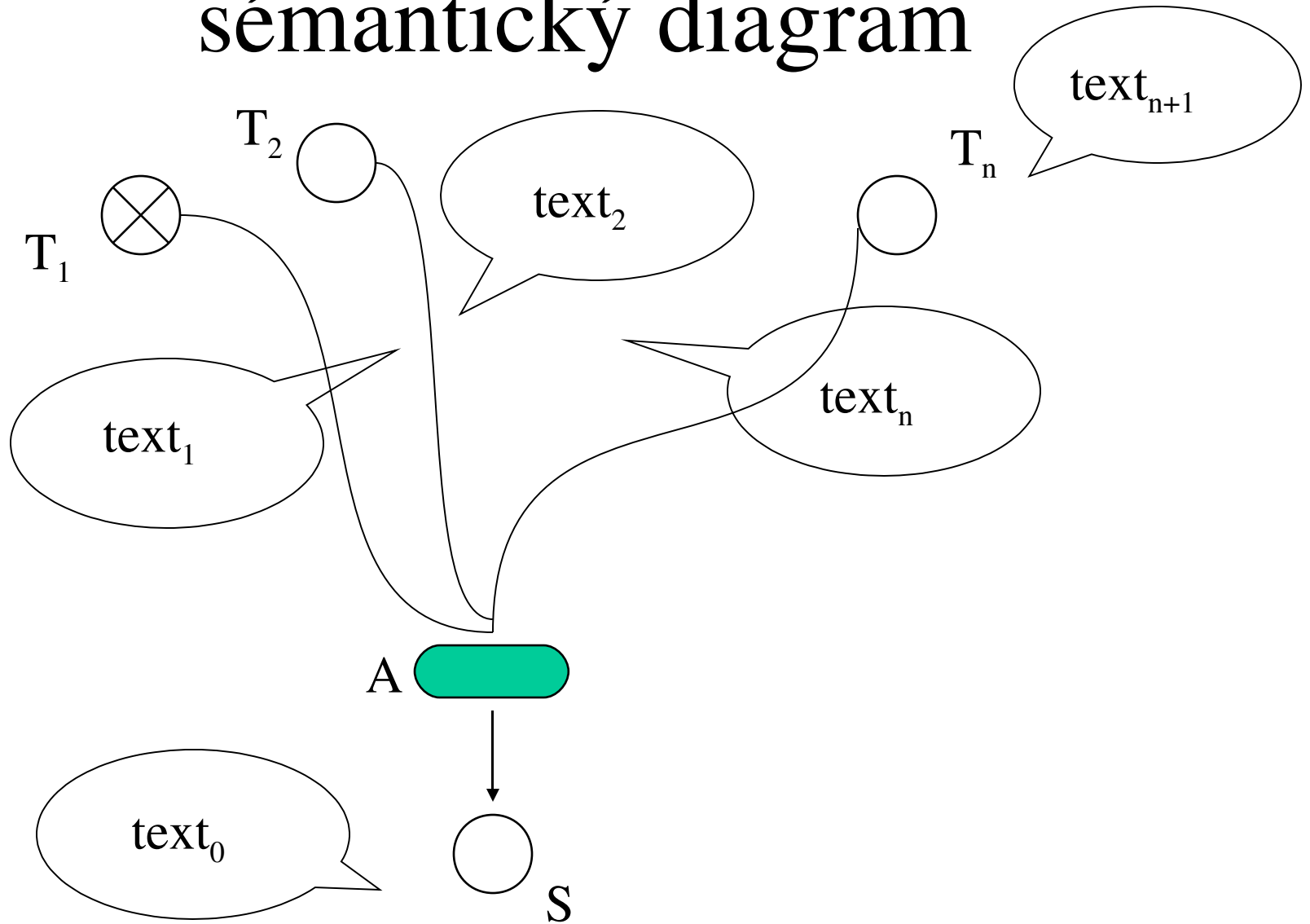
sémantický diagram



zápis sémantiky (3)

- $A / (\text{Wrd} \rightarrow (\text{Tim} \rightarrow ((T_1, \dots, T_n) \rightarrow (S \rightarrow \text{Bool}))))$
- $A = \text{text}_0(S)\text{-s text}_1(T_1) \text{ text}_2(T_2) \dots \text{text}_n(T_n) \text{ text}_{n+1}$
- kde pouze text_0 a text_{n+1} mohou být vynechány a **celý výraz (čtený jako část věty přirozeného jazyka) nám sděluje funkci, která dává hodnoty z 2^S na argumentech z n -tice (T_1, \dots, T_n)**
(přípona -s odlišuje od prvního případu a nabádá ke čtení v množném čísle)
- tomuto jednoznačně odpovídá (sémantický) diagram

sémantický diagram



Poznámky:

- text_0 (je-li uveden) vyjadřuje roli oboru hodnot funkce
- text_i ($1 \leq i \leq n$) vyjadřuje roli i -tého argumentu funkce
- text_{n+1} je nepovinný dodatek pro lepší porozumění
- *proč je špatně, napsat pouze oznamovací větu (propozici) namísto funkci vyjadřujícího výrazu !*
- *Problém jednoznačné interpretace výrazů NL, který je přirozeně nejednoznačný*

Integritní omezení

- vycházejí z našeho porozumění výrazům přirozeného jazyka a z naší znalosti popisované reality
- vyjadřují to, že ne všechny funkční hodnoty datových funkcí (E-typů nebo HIT-atributů) jsou přípustné
- integritní omezení je dáno pravdivostí určité propozice, tj. vždy přináší informaci
- nazvěme databázovým stavem každé možné naplnění databázových tabulek
- integritní omezení definují přípustná naplnění databázových tabulek, tzv. databázový prostor
- speciálním případem je tzv. **poměr HIT-atributu**
- ponecháváme stranou tzv. analytická integritní omezení ...

Poměr HIT-atributu

- Poměrem HIT-atributu A rozumíme zápis

$$p,m:q,n$$

kde: $p, q = 0 \vee 1$; $m, n = 1 \vee M$

- $p = 0$ znamená, že A je parciální v užším smyslu
- $p = 1$ znamená, že A je totální funkce
- $m = 1$ znamená $A/(\dots \rightarrow S)..$ jednoznačná fce
- $m = M$ znamená $A/(\dots \rightarrow (S \rightarrow \text{Bool}))..$ mnohoznačná fce
- q, n má stejný význam pro funkci „obrácenou“ A^{-1}

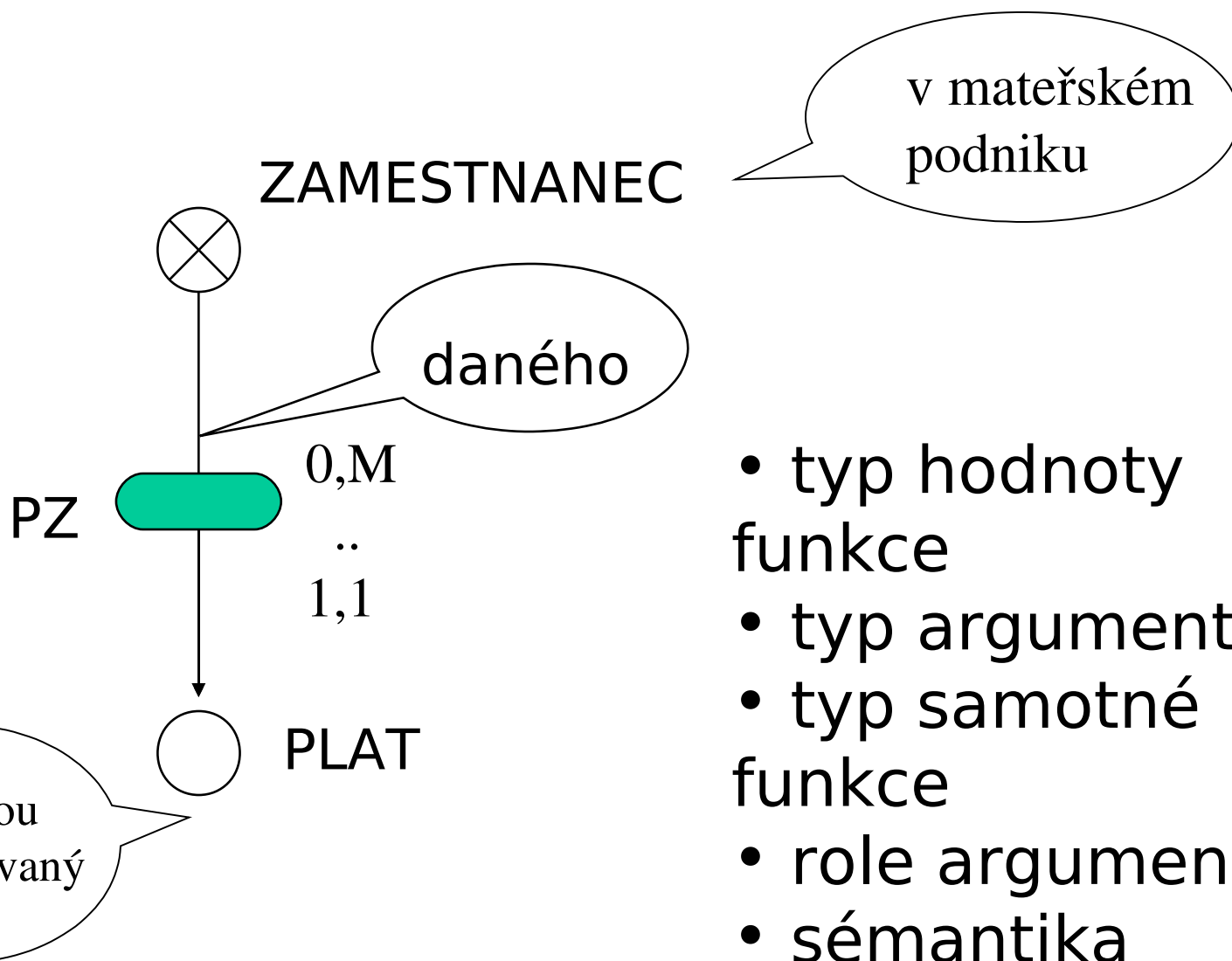
„obrácená“ funkce

- $A / ((T_1, \dots, T_n) \rightarrow S)$,
pak „obrácená“ funkce je
 $A^{-1} / (S \rightarrow (T_1, \dots, T_n))$ nebo
 $A^{-1} / (S \rightarrow ((T_1, \dots, T_n) \rightarrow \text{Bool}))$
- $A / ((T_1, \dots, T_n) \rightarrow (S \rightarrow \text{Bool}))$,
pak „obrácená“ funkce je
 $A^{-1} / (S \rightarrow (T_1, \dots, T_n))$ nebo
 $A^{-1} / (S \rightarrow ((T_1, \dots, T_n) \rightarrow \text{Bool}))$
- Pozor: obrácená funkce není totéž co funkce inverzní v matematice

Příklady HIT-atributů

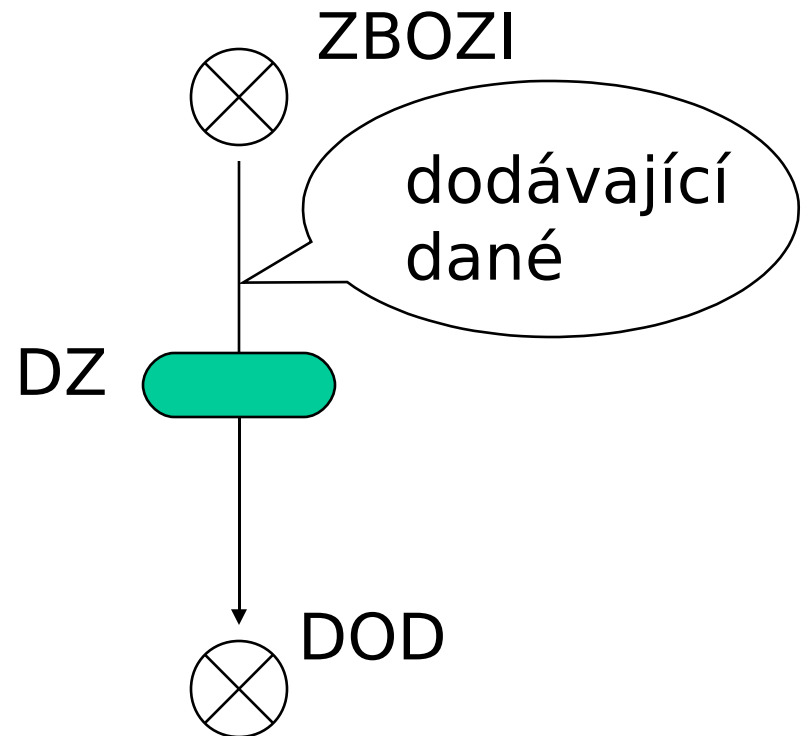
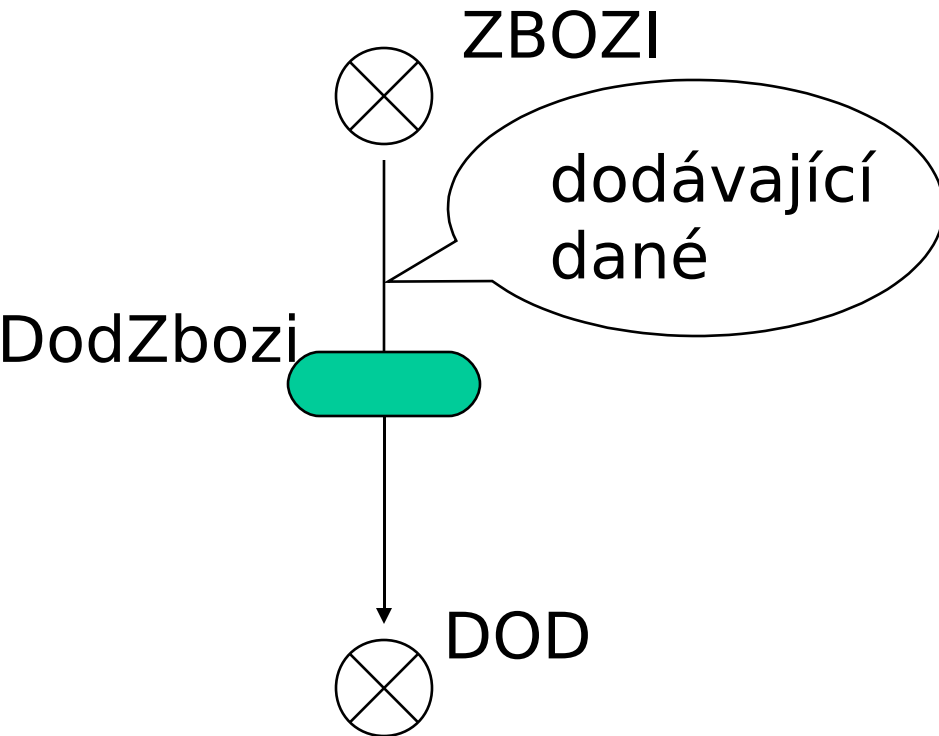
- PZ: smlouvou deklarovaný plat (Plat) daného zaměstnance (#Zamestnanec) v mateřském podniku /1,1:0,M
- DZ: dodavatelé (#Dodavatel)-s dodávající dané zboží (#Zbozi)
- ODZ: smluvní odběratelé (#Odberatel)-s kterým daný dodavatel (#Dodavatel) dodává dané zboží (#Zbozi) /0,M:0,M
- A32: sledovaná množství (Mnozstvi)-s daného druhu zboží (#Zbozi) dodaná daným dodavatelem (#Dodavatel) danému odběrateli (#Odberatel) /0,M:0,M
- A33: množství (Mnozstvi) daného druhu zboží (#Zbozi) dodané daným dodavatelem (#Dodavatel) danému odběrateli (#Odberatel) v daném čase (Cas) /0,1:0,M

Příklady:

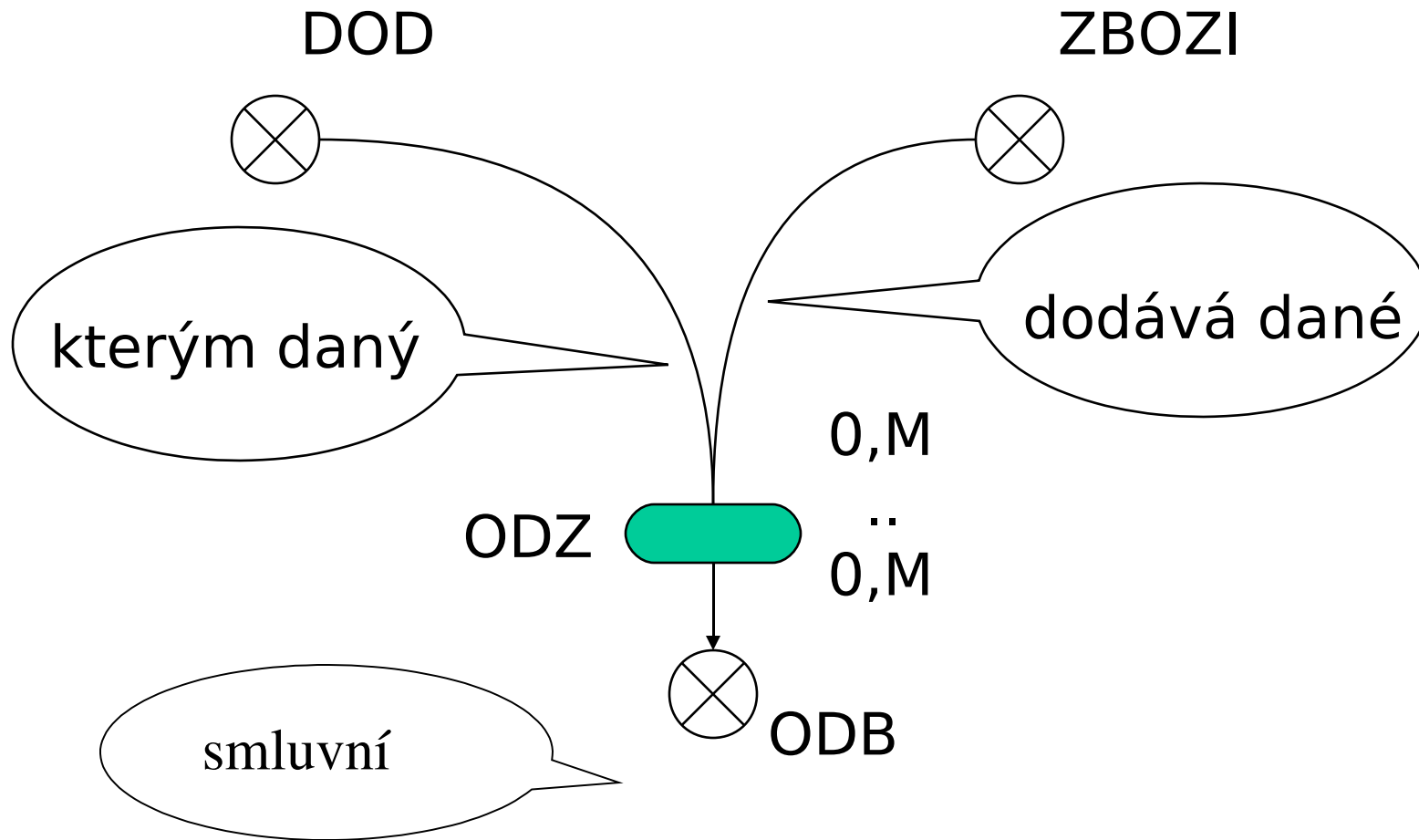


- typ hodnoty funkce
- typ argumentů
- typ samotné funkce
- role argumentů
- sémantika přiřazení

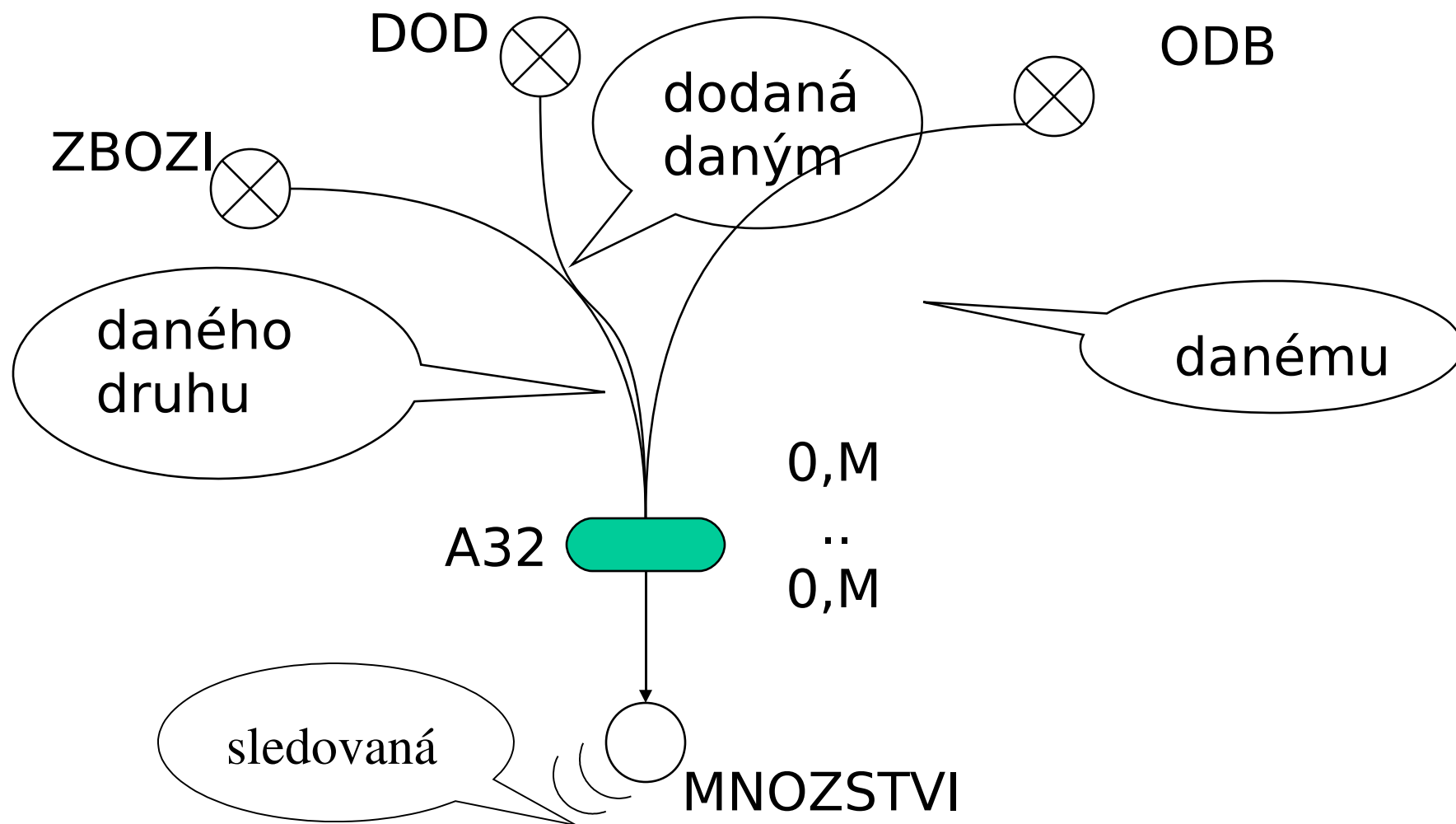
Příklady: co lépe vystihuje realitu



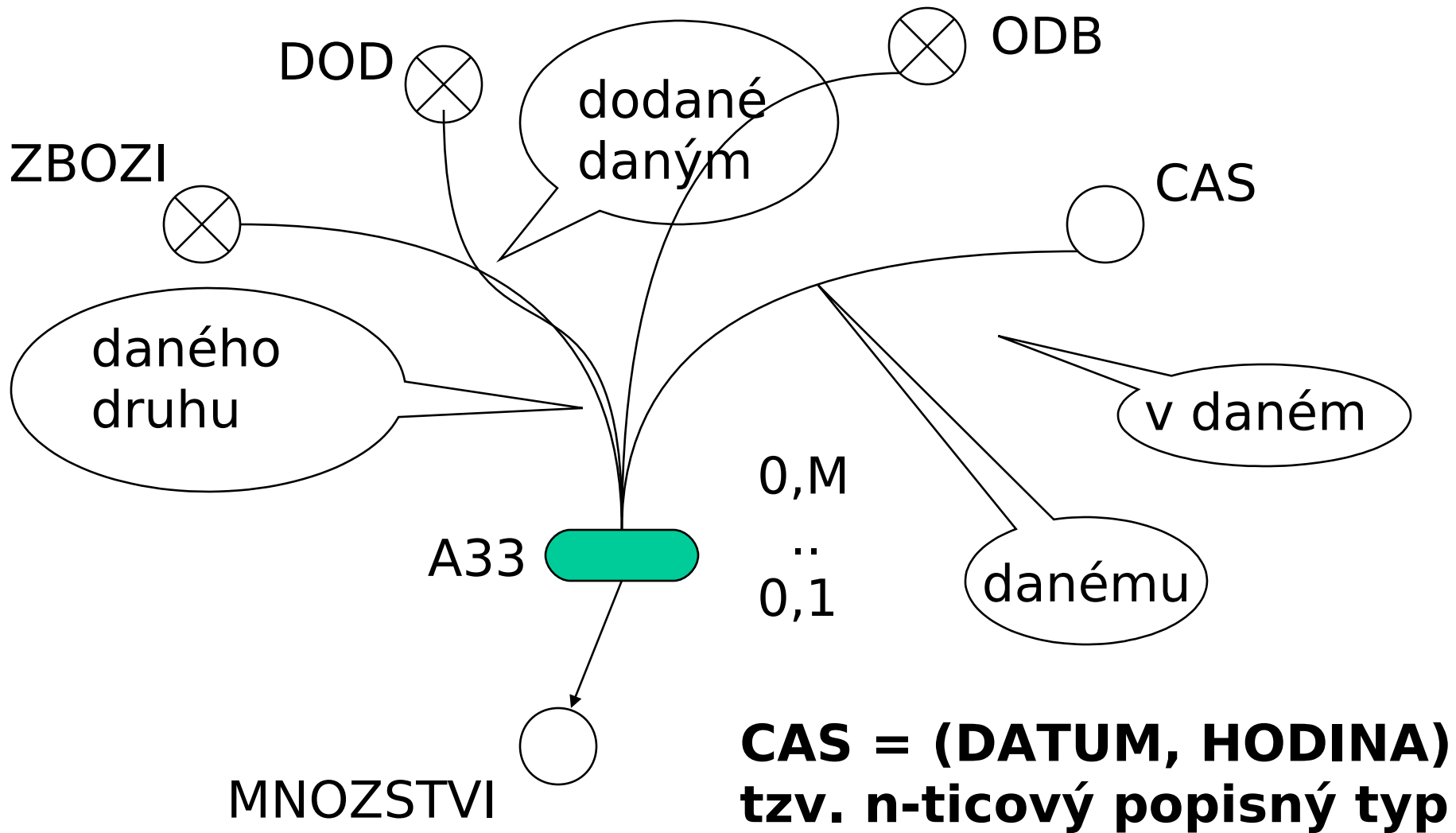
Příklady



Příklad: atribut (složitosti 4)



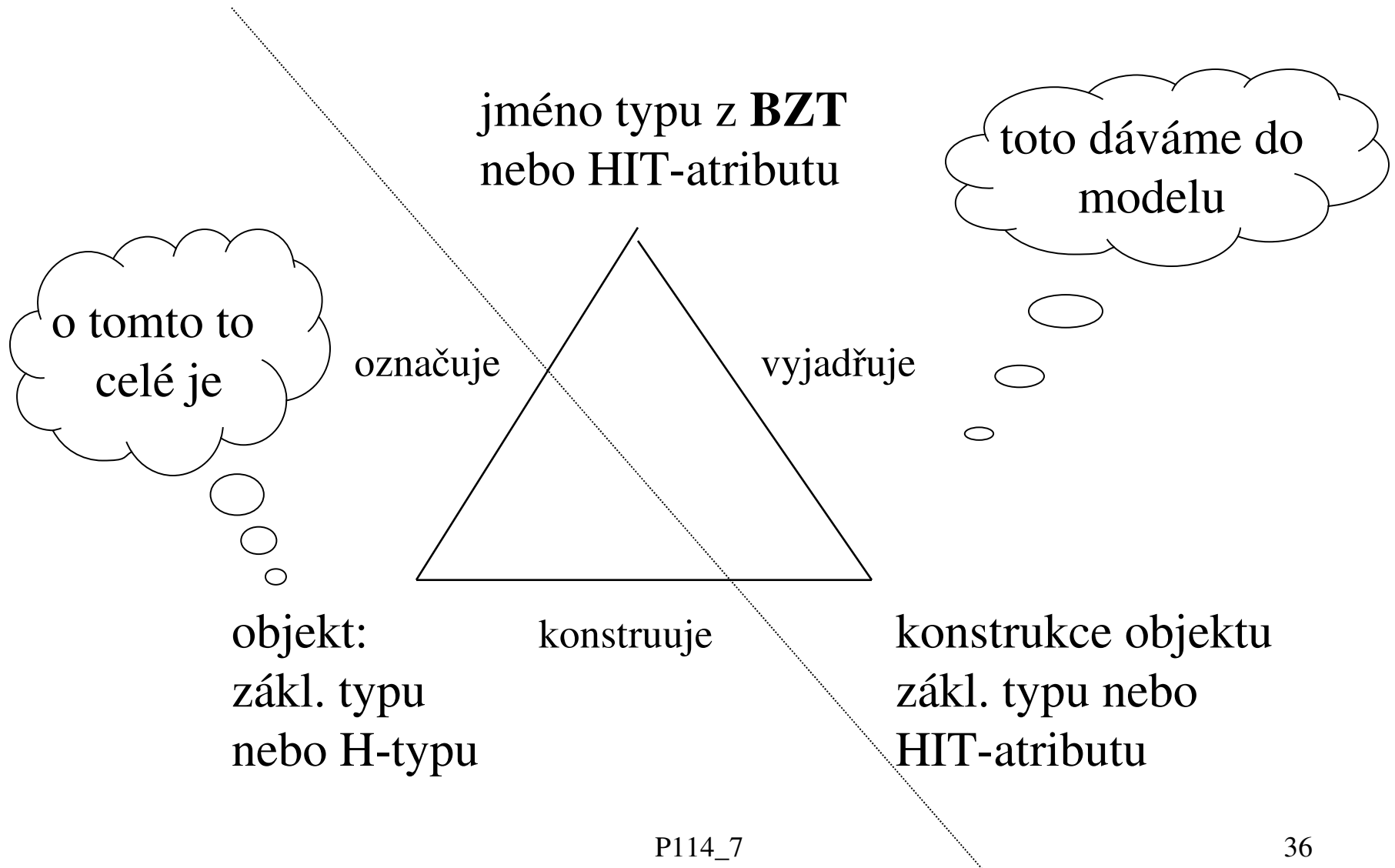
Příklad s chybkou:



Podstata modelování

- Složitost hierarchie typů nad **EB** a jejich konstrukcí je skryta v definicích uzlových typů patřících do **BZT**.
- Veškeré konstrukce, se kterými při práci s HIT-atributy pracujeme, jsou konstrukcemi jednoduchých typů (a) nebo (b) - viz přednáška 6.
- Toto odpovídá praxi implementace na počítačích: prvky **BS** (viz přednáška 6) reprezentujeme jako tabulky / soubory; konstrukce jednoduchých typů jako jednoznačné nebo mnohoznačné vztahy

KTO v HIT metodě



... a kvůli tomuto to celé děláme:

- aplikací prvků H-typů na *wt* dostaneme
...
- extenze HIT-atributů, které vyjadřují vztahy mezi konkrétními prvky sort (entitních či deskriptivních) z **BZT**
- ... čili to, co máme ve formě databázových tabulek uloženo v databázích

?

stav světa, datový model a databázový stav

- jaký je vztah datového modelu a aktuálního stavu světa?
- Databázový stav (DbSt): každá možná **populace sort**?
- ... nebo **naplnění kontejnerů** na prvky jednotlivých sort v dané databázi
- V databázi jsou ke každé sortě vždy jen konečné množiny
- Jak tedy definovat DbSt?
- Jaký je vztah DbSt a datového modelu?
- Jak se má DbSt k aktuálnímu světu?

?

aktualizace databáze a poznatelnost stavu světa

- Jak je možno popsat operaci aktualizace databáze?
- Podobnost s valuací?
- ... nebo je to stejné jako operace dotazu, jenom přehodíme co je vstupem a co je výstupem ...
- Je poznatelný aktuální svět?
- Zvýší se poznatelnost stavu světa zavedením informačního systému se „správně“ zkonstruovanou databází?
- V čem spočívá „přidaná hodnota“ databází ?

P114

Postup tvorby modelu

praktická doporučení

8

Témata

- Celkový postup tvorby datového modelu
- Komponenty datového modelu
- Jak vytvořit seznam E-typů
- Jak psát definice základních typů
- Jak objevovat HIT-atributy
- Dialog modeláře s „expertem“
- Jak zapisovat konstrukci HIT-atributu se sémantikou
- určení a zápis poměru HIT-atributu
- vybalancování definic E-typů a sémantiky HIT-atributů
- Hledání nejvýstižnějšího vyjádření

Postup DM

- Vymezení zájmové oblasti
- Dialog s expertem
- Studium podkladů
- Tvorba seznamu E-typů
- Definice každého E-typu
- Prvotní návrh datového modelu v diagramech + textech
- Formovací seminář
- Revize datového modelu
- Transformace do tvaru ERD
- Zápis ve zvoleném CASE

Vymezení zájmové oblasti

- ... zlatokop si vždy vykolíkuje svůj claim
- Určení komponent, které budeme řešit
- určení hranic těchto komponent
co do nich patří, a co už ne

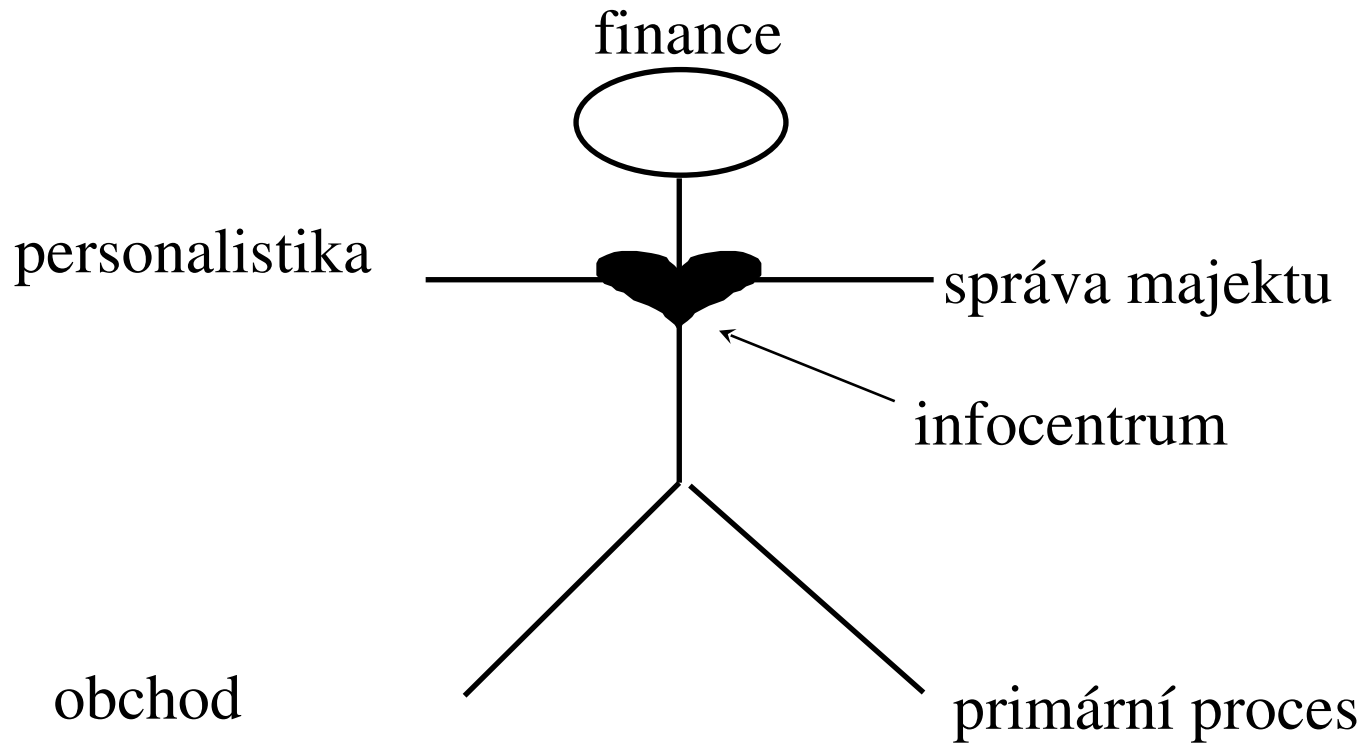
Z čeho se DM organizace skládá

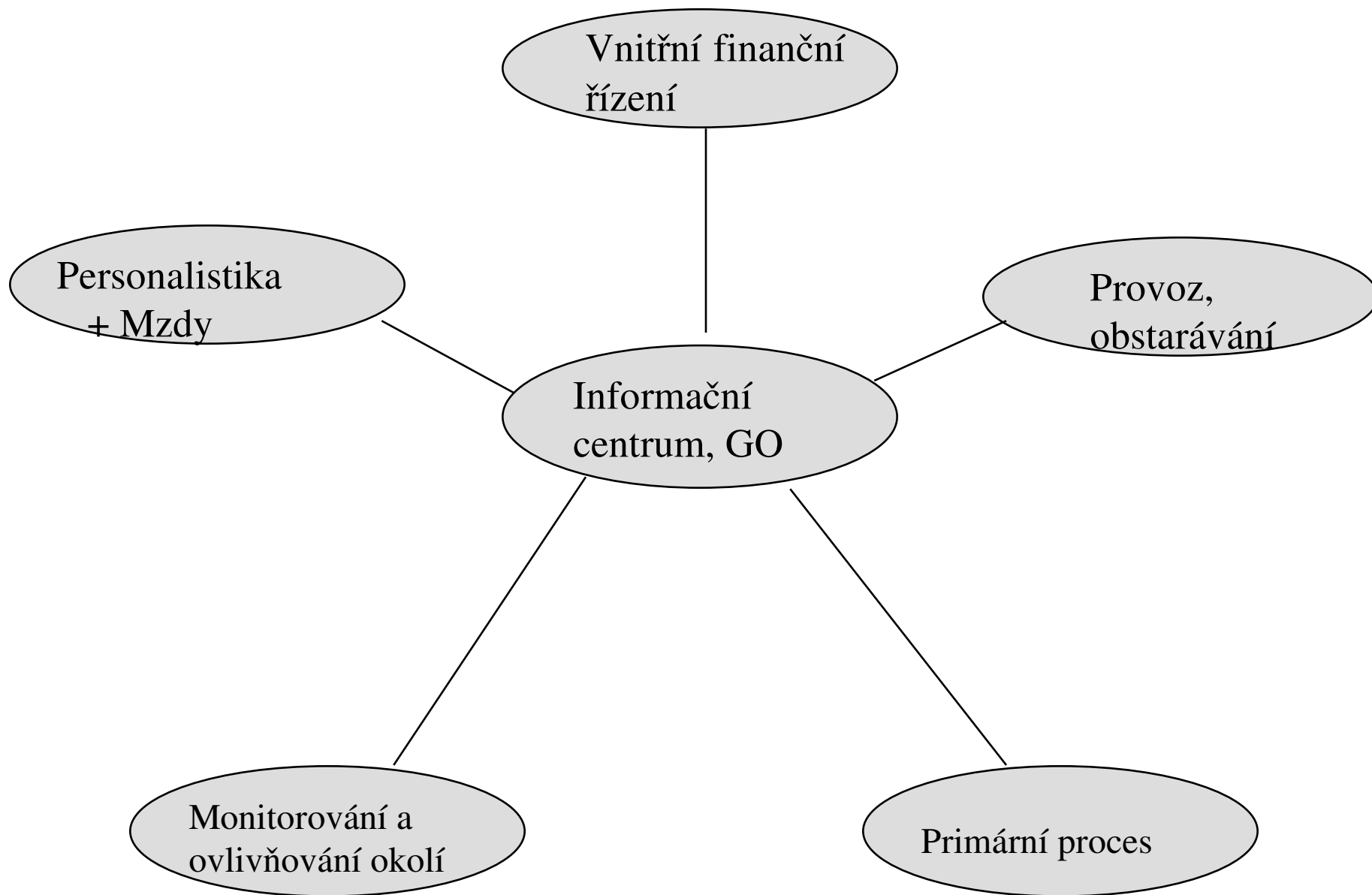
- z komponent odpovídajících funkčním (fyziologickým) okruhům „organizmu“
- funkční okruh \longrightarrow komponenta / 1:M
- proč více komponent a ne jedna „všeobjímající“ (dynamika rozvoje v dimenzích funkčních okruhů)
- proč ne podle morfologie?
- fyziologický panák a model komponent

Komponenta:

- minimum vazeb (souvislostí) překračuje hranici do jiných komponent
- maximum vazeb (souvislostí) uvnitř komponenty
- pokrývá vždy ucelenou business oblast
- její zavedení jako části IS znamená přínos pro fungování organizace
- její vymezení není a ani nemůže být exaktní

Fyziologie





Příklady

- komponenta Rozvrh
- komponenta Studium
- komponenta Mzdy
- komponenta Výroba
- komponenta Obchod
- komponenta Sklady

Jak vytvořit seznam E-typů

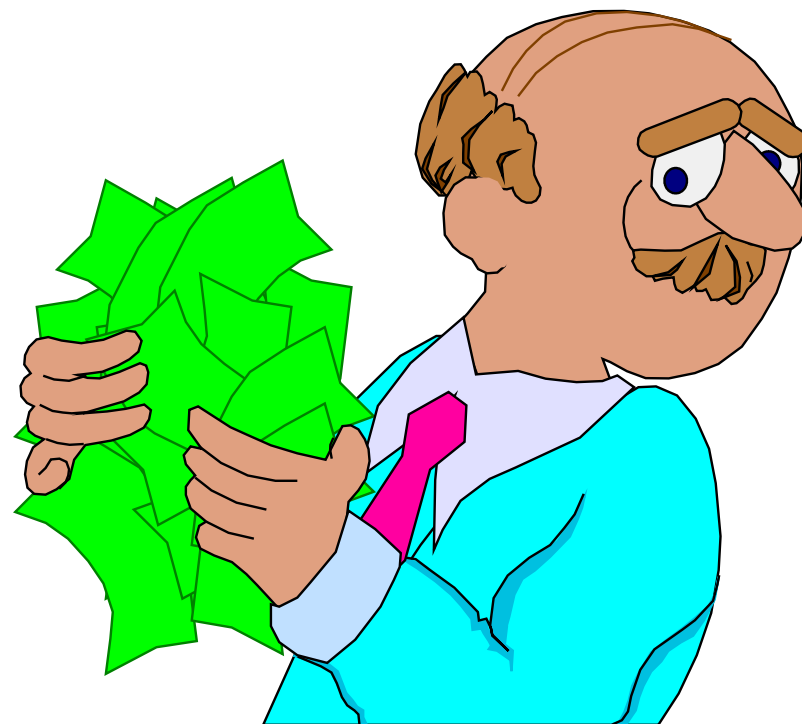
- technika „brainstorming“
 - odložená kritika
 - týmové myšlenkové asociace
 - čištění seznamu
 - *použití při práci se skupinou „expertů“*
- technika „podstatných jmen“
 - vyprávění starého fabrického inženýra
 - *použití při práci s jedním „expertem“*

technika brainstorming:

- vysvětlit co je to „objekt“
(jednoznačná odlišitelnost od všeho ostatního, má smysl evidovat alespoň dvě různé charakteristiky)
- vysvětlit brainstorming
(odložit kritiku a autokritiku, vykřikovat nápady; využití týmových myšlenkových asociací)
- provést brainstorming (cca 5 minut, dokud vykřikují nápady)
zapisovat na flip-chart
„tapetovat“ místnost popsanými plakáty
- provést „čištění“
viz dále

technika „podstatných jmen“

- nechat vyprávět „experta“ a NEPOSLOUCHAT !
- pouze zapisovat podstatná jména
- provést „čištění“



Čištění seznamu E-typů

- podle kritérií „co je objekt“
 - vyškrtat všechny „ne-objekty“
 - vyloučit „objekty“ nezajímavé z hlediska businessu organizace
 - vše v dialogu se skupinou nebo s jedním expertem
- ... následuje dialog s expertem za účelem odhalení funkčních závislostí (konstrukce HIT-atributů)

Jak psát definice základních typů

- Objektem typu (#jmeno E-typu) je každý takový ... (každé takové individuum), pro který/é platí ...
- Příklady:
 - Objektem typu (#Artikl) je každý produkt nebo služba nebo právo, který může být předmětem nákupu či prodeje a to včetně produktů, služeb nebo práv dosud neexistujících, ale potenciálně vytvořitelných pro účely rozvojových aktivit obchodní společnosti.
 - Objektem typu (#Dokument) je každý záznam nebo zpráva, jehož/jejíž zaznamenání má pro organizaci smysl.
 - Objektem typu (#Business Partner) je každé takové individuum, které je, bylo nebo může být účastno obchodních aktivit naší společnosti a které je zajímavé z pohledu rozvojových aktivit naší společnosti.
- ***Tento způsob definice je závazný, povinný !***

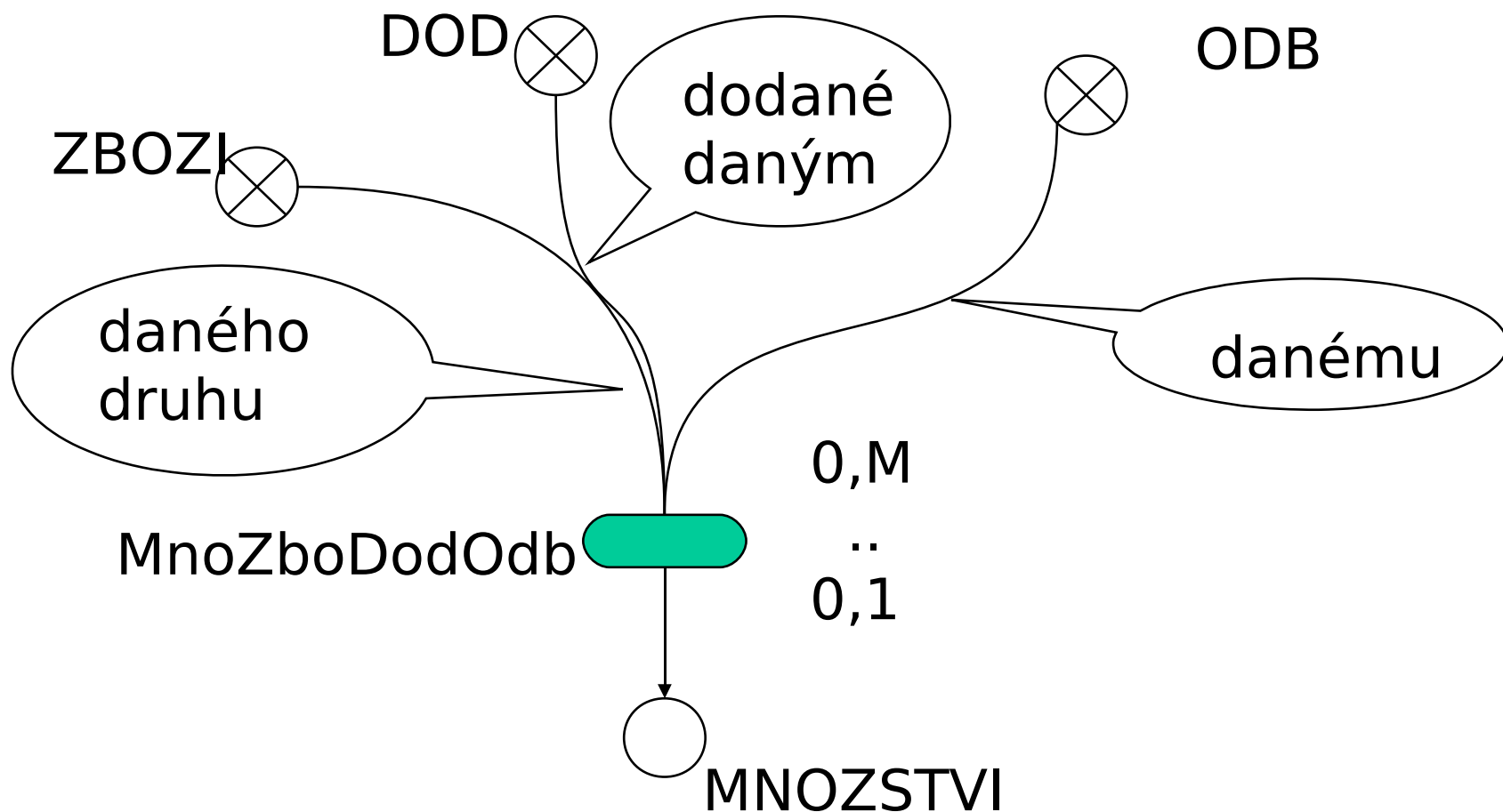
Jak psát definice základních typů - rigorózně

- Prvkem typu (#jmeno E-typu) je každý takový objekt, pro který platí ...
- Příklady:
 - Prvkem typu (#Artikl) je každý takový objekt, který má vlastnost být produktem nebo službou nebo právem, a který může být předmětem nákupu či prodeje a to včetně produktů, služeb nebo práv dosud neexistujících, ale potenciálně vytvořitelných pro účely rozvojových aktivit obchodní společnosti.
 - Prvkem typu (#Business Partner) je každý takový objekt, který má jednoznačnou právní identitu a který je, byl nebo může být účasten obchodních aktivit naší společnosti a který je zajímavý z pohledu rozvojových aktivit naší společnosti.
- Všimněte si rozdílu v pragmatickém přístupu k tomu, co je (#Business Partner) oproti předchozí definici

Jak objevovat HIT-atributy

- inspirace v řeči experta nebo v textech podkladových materiálů - tj. v přirozeném používání NL
- rozpoznat ty výrazy NL, které mají funkční charakter, tj. označují přiřazení něčeho něčemu, čili vyjadřují konstrukci funkce
- „... potřebujeme datum naskladnění každého zboží ...“
(Datum) naskladnění daného (#Zbozi)
- „... nemáme přehled kdo komu dodává které výrobky ...“
(#Dodavatel)-s kteří dodávají daný (#Vyrobek) danému (#Odberatel)
- je vhodné si pomoci grafickým vyjádřením:

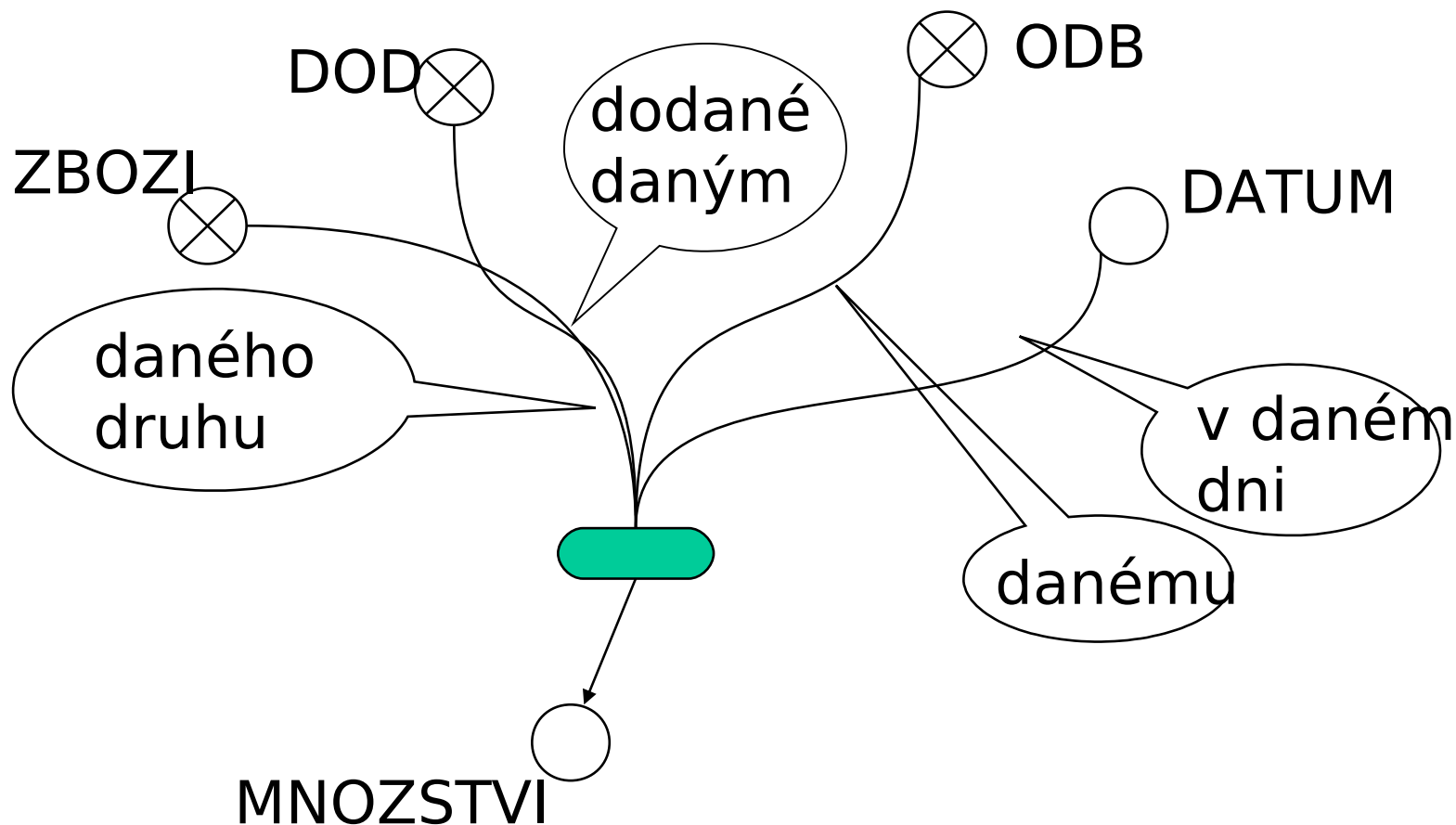
„... my ale, kromě toho, potřebujeme vědět, kdo kolik čeho komu skutečně dodal ...“



Diskuse s „expertem“

- M: je jasné, že atribut na obrázku popisuje něco jiného, než atribut předchozí. Není ale zboží a výrobek totéž?
- E: ano
- M: ponecháme tedy jenom „zboží“ a přeformulujeme již zapsané atributy:
(#Dodavatel)-s kteří dodávají dané (#Zbozi) danému (#Odberatel)
To množství zboží vás zajímá kumulativně? za jaké období? stačí rok? nebo měsíc ?
- E: nee, my potřebujeme zapsat dodávku každý den ...
- M: bude to tedy vyhovovat takto:

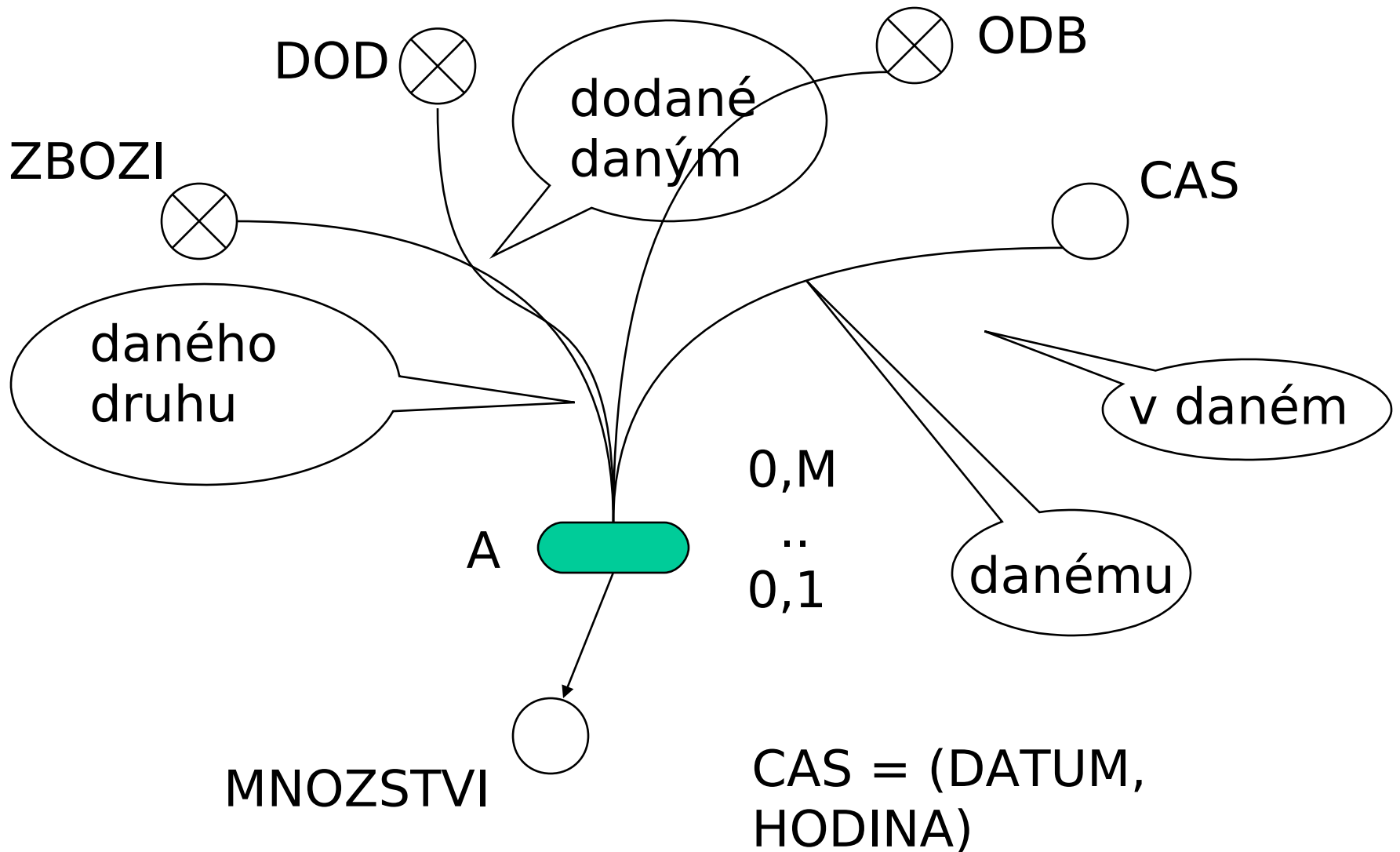
množství (MNOZSTVI) daného druhu zboží
(#ZBOZI) dodané daným dodavatelem (#DOD)
danému odběrateli (#ODB) v daném dni
(DATUM)



diskuse - pokračování

- M: a v jednom dni může přijít pouze jedna dodávka?
- E: ano ... Vlastně někdy jich přijde víc...
- M: potom to musíme nakreslit takto:

dodávka; (připíšeme i poměr):



Jak zapisovat konstrukci HIT- atributu se sémantikou

- při dialogu s expertem je srozumitelnější kreslit diagramy
- mezi zkušenými stačí psát lineární zápis HIT-
atributu
- NIKDY ! však schémátka bez sémantiky
- Viz zápis sémantiky v přednášce č. 7

Určení a zápis poměru HIT-atributu

- E: my ale ke každému zboží, chceme i výrobce ...
- M: dobře, zapíšeme:
(#Vyrobcce) daného (#Zbozi) /**1,1:1,M**
... každé zboží má právě jednoho výrobce, a každý výrobce, kterého evidujeme, musí nějaké zboží vyrábět, někteří však vyrábějí více ...
- E: my ale u některého zboží nevíme, kdo ho vyrábí ...
- M: dobře, pak musíme psát:
(#Vyrobcce) daného (#Zbozi) /**0,1:1,M**
... a skutečně všechno vaše zboží je vždy od jediného výrobce?
- E: ne, máme na některá zboží více dodavatelů

pokračování

Určení a zápis poměru - pokračování

- M: chtěl jste říci výrobců ..., pak musíme zapsat:
(#Vyrobcce)-s daného (#Zbozi) /0,M:1,M (1)
... a vy neevidujete jiné výrobce, než ty, od kterých již něco odebíráte?
- E: Ne.
- M: co když se dozvíte o novém výrobcí pro vaše zboží?
Nechcete si ho zaznamenat?
- E: my to neděláme ...
- M: ale nebylo by to výhodné ?
- E: no, možná...
- M: napíšeme raději
(#Vyrobcce)-s daného (#Zbozi) /0,M:0,M (2)

Vybalancování

definic E-typů a poměru HIT-atributů

- definice E-typu #Vyrobcce se liší podle toho, zavedeme-li do modelu (1) nebo (2) (viz předchozí slide):
- v případě (1):
Objektem typu (#Vyrobcce) je každý subjekt účastný trhu, který vyrábí některý z těch druhů zboží, které my nakupujeme od dodavatelů.
- v případě (2):
Objektem typu (#Vyrobcce) je každý subjekt účastný trhu, který vyrábí nějaký druh zboží, a který má smysl evidovat pro účely našeho podnikání

Hledání nejvýstižnějšího vyjádření

- viz předchozí popis dialogu s expertem
- Formovací seminář:
soustředěná práce týmu expertů z podniku pod vedením zkušeného moderátora - konzultanta
 - opravy definic E-typů + opravy sémantiky a poměrů HIT-atributů, vzájemné vyvažování
 - doplňování E-typů a HIT-atributů, které chyběly
- Revize datového modelu
 - formální správnost definic E-typů a zápisu sémantiky HIT-atributů
 - křížové kontroly:
sémantika HIT-atributu \leftrightarrow definice E-typů \leftrightarrow poměr HIT-atributu
- Upřesňující dialogy s „experty“
- *To jsou nástroje hledání nejvýstižnějšího vyjádření*

P114

Definovatelnost

Manipulace s HIT-atributy

9

Témata

- Identita a semiidentita HIT-atributů
- Definovatelnost
- Informační ekvivalence
- Informační schopnost
- Rotace, triviální odvození

HIT-atributy

- každá taková funkce **A** je dána konstrukcí

$$\lambda wt \lambda x_1 \dots x_n \rho y \ (C)$$

kde $w :: \text{Wrd}$, $t :: \text{Tim}$, $x_i :: T_i$, $y :: S$, T_i , S jsou základní typy splňující podmínky definice HIT-atributu, nebo

$$\lambda wt \lambda x \rho y \ (C)$$

kde $x :: (T_1, \dots, T_n)$

C je otevřená Bool-konstrukce neobsahující jiné volné proměnné než w , t , x_i , y

A / ($\text{Wrd} \rightarrow (\text{Tim} \rightarrow ((T_1, \dots, T_n) \rightarrow S))$) nebo

A / ($\text{Wrd} \rightarrow (\text{Tim} \rightarrow ((T_1, \dots, T_n) \rightarrow (S \rightarrow \text{Bool}))))$,

Základní předpoklad o HIT-atributech:

- **extenze HIT-atributů jsou konečné tabulky.**
- Předpoklad je oprávněný, poněvadž při modelování bereme v úvahu vždy
 - konečný diskrétní časový interval „života“ IS
 - konečnou populaci sort (E-typů a/nebo D-typů)

základní předpoklad o analytických funkcích

- Analytické funkce které budeme dále uvažovat jsou vždy surjekce a jsou to totální funkce.
- ZOPAKOVÁNÍ:
 $f / (T \rightarrow \mathfrak{R})$, kde $\mathfrak{R} = S$ nebo $\mathfrak{R} = (S \rightarrow \text{Bool})$
 f je injekce (zobrazení prosté), když $\forall x, y \in T ([f\ x] = [f\ y] \Rightarrow x = y)$
 f je surjekce (zobrazení na), když $\forall y \in \mathfrak{R} (\exists x \in T ([f\ x] = y))$
 f je bijekce, když je injekce a surjekce
- často používané budou logické funkce, tj. funkce jejichž oborem hodnot je Bool
- základní předpoklad vylučuje možnost, že by mezi logickými funkcemi, které budeme používat, byly tautologie nebo kontradikce

Identita HIT-atributů

- konstrukce C_1 je ekvivalentní s konstrukcí C_2 , jestliže pro libovolnou valuaci v a libovolný objekt \underline{A} platí C_1 v -konstruuje \underline{A} právě tehdy, když C_2 v -konstruuje \underline{A} .
- Nechť A_1, A_2 jsou HIT-atributy pořadě konstruované konstrukcemi C_1 a C_2 .
- Jestliže C_1 je ekvivalentní s C_2 , pak HIT-atributy A_1 a A_2 jsou identické a píšeme $A_1 = A_2$.
- To, že konstrukce C konstruuje HIT-atribut A budeme pro jednoduchost rovněž zapisovat $A = C$

Problém s parcialitou HIT-atributů

- Budeme konstruovat atributy pomocí jiných atributů
- atribut může být na některých argumentech nedefinován: konstrukce jej používající pak je pro příslušnou valuaci v-nevlastní
- jiná konstrukce, která konstruuje prakticky totéž, však na těchže argumentech vrací $\{ \}$.
- Tento rozdíl nemá vliv na množinu z atributu generovaných propozic
- Proto se zavádí tzv. semiidentita

Semiidentita

Nechť $A = \lambda_{wt} \lambda_x \rho_y (C)$, $B = \lambda_{wt} \lambda_x \rho_y (C_1)$

Atributy A , B nazveme semiidentické, jestliže

- buďto jsou identické
- nebo se liší následujícím způsobem:

$$\forall wtx ([A_{wt}x] = [B_{wt}x] \vee ([A_{wt}x] = \{ \} \wedge [B_{wt}x] = \{\perp\}))$$

kde $\{ \}$ je prázdná množina, to v případě, že C je na x

Nepravda, a

$\{\perp\}$ je třída taková, že o žádném prvku daného typu nemůžeme říci, že patří do této třídy proto, poněvadž C_1 je na x nedefinováno.

V dalším bude $A = B$ značit i semiidentitu.

Definovatelnost

- Atribut A je definovatelný nad množinou atributů $\{B_1, \dots, B_n\}$ právě když

$$\exists f (\forall w t ([A \ w t] = [f ([B_1 \ w t], \dots, [B_n \ w t])]))$$

kde f je analytická funkce splňující základní předpoklad, w je možný svět, t je časový okamžik.

Píšeme: $A \leftarrow (B_1, \dots, B_n)$

- Množina atributů M je definovatelná nad množinou atributů N , je-li každý prvek z M definovatelný nad nějakou podmnožinou N .

Píšeme: $M \leftarrow N$

- Atribut A je definovatelný nad atributem B , když je definovatelný nad množinou $\{B\}$.

Píšeme: $A \leftarrow B$

Informační ekvivalence

- Nechť $M = \{A_1, \dots, A_n\}$, $N = \{B_1, \dots, B_m\}$ jsou množiny atributů a nechť $M \leftarrow N$ a zároveň $N \leftarrow M$. Potom říkáme, že M a N jsou informačně ekvivalentní. Píšeme: $M \approx N$
- Intuitivně je zřejmé, a lze dokázat, že dvě informačně ekvivalentní množiny atributů generují v každém stavu světa stejnou třídu propozic.
- Jiná intuice: z n -tice tabulek daných množinou atributů M dostaneme tutéž informaci jako z m -tice tabulek daných množinou atributů N
- VĚTA: Relace \approx na množině atributů je ekvivalence.

Důsledek

- Nechť atribut A je definovatelný nad množinou atributů $\{B_1, \dots, B_n\}$.

Potom $\{A, B_1, \dots, B_n\} \approx \{B_1, \dots, B_n\}$.

- Intuitivně: přidáním definovatelného („odvoditelného“) atributu nic nového nezískáme, nebo odebráním definovatelného („odvoditelného“) atributu nic z původního neztratíme
- Co je to, co nezískáme ani neztratíme ?

quasi-uspořádání

- Relace definovatelnosti mezi množinami atributů je:
 - reflexivní (evidentně $M \leftarrow M$, za f stačí vzít identitu)
 - tranzitivní (viz důkaz předchozí věty)
- Avšak není antisymetrická:
tj. z $M \leftarrow N$ a $N \leftarrow M$ neplyne $M = N$
viz předchozí důsledek
- Tedy není to relace částečného uspořádání, ale pouze quasi-uspořádání
- Z Algebry:: lze definovat částečné uspořádání na množině ekvivalenčních tříd

Informační schopnost

- Nechť **BZT** je nějaká báze základních typů. Nechť **ATR** je množina všech možných HIT-atributů nad **BZT**.
- Faktorová množina \mathbf{ATR} / \approx (tj. množina všech tříd ekvivalence nad **ATR**) je částečně uspořádaná relací indukovanou definovatelností. Značíme \angle
- Každá třída ekvivalence z \mathbf{ATR} / \approx definuje určitou **informační schopnost**. Relace \angle je částečným uspořádáním informačních schopností.

Tvrzení o informační schopnosti

- Nechť $C_M, C_N \in \mathbf{ATR} / \approx$, $M \in C_M$, $N \in C_N$
- $C_M \angle C_N$ právě když $M \leftarrow N$
- **Tvrzení 1:**
všechny prvky třídy C_M mají stejnou informační schopnost
- **Tvrzení 2:**
každý prvek C_M má menší informační schopnost než každý prvek třídy C_N
- **Tvrzení 3:**
Jestliže pro C_M a C_N neplatí ani $C_M \angle C_N$, ani $C_N \angle C_M$, pak informační schopnost libovolného prvku z C_M je nesrovnatelná s informační schopností libovolného prvku z C_N .

dohoda na označení, příklad

- $A = \text{počet (PrirozCislo) druhů výrobků vyráběných daným výrobcem (\#Vyrobce) / 0,1:0,M}$
- $A/(\text{Wrd} \rightarrow (\text{Tim} \rightarrow (\#Vyrobce \rightarrow \text{PrirozCislo})))$
to lze psát (Shönfinkelova redukce):
 $A/((\text{Wrd}, \text{Tim}) \rightarrow (\#Vyrobce \rightarrow \text{PrirozCislo}))$
toto platí pro všechny HIT-atributy
- Označíme $(\text{Wrd}, \text{Tim}) = W \dots$ stav světa, potom:
 $A/(W \rightarrow (\#Vyrobce \rightarrow \text{PrirozCislo}))$
a konstrukce A má tvar:
 $A = \lambda_w \lambda_x \iota_y ([Aw] x = y)$, kde
 $w::W, x::\#Vyrobce, y::\text{PrirozCislo}$
- Aplikace $[Aw]$ je velmi častá; proto zkracujeme: A_w
- A_w je extenze atributu A ve stavu světa w (je to tabulka)

Příklad

- Atribut A je definovatelný nad atributem B = druhy výrobků (#Vyrobek)-s vyráběné daným výrobcem (#Vyrobce) / 0,M:0,M
- Odvození: $(x::\#Vyrobce, z::\#Vyrobek, y::PrirozCislo)$
 $B = \lambda w \lambda x \lambda z ([B_w x] z])$

hledáme f tak, aby pro všechna w
 $A_w = [f B_w]$:

$b :: (\#Vyrobce \rightarrow (\#Vyrobek \rightarrow Bool))$,
 $a :: (\#Vyrobce \rightarrow PrirozCislo)$ jsou proměnné, pak
 $f = \lambda b \lambda a (\forall x ([ax] = [Card [bx]]))$,
kde Card je funkce kardinality množiny

Skutečně (beta-pravidlo):

$$[f B_w] = \lambda a (\forall x ([ax] = [Card [B_w x]])) = A_w$$

méně formálně

- **f** je algoritmus, který nezávisle na stavu světa vypočítává z tabulky **b** (mimoходом nenormalizované) tabulku **a**.
- V praxi se netrápíme hledáním konstrukce funkce **f** ve tvaru lambda-termu, ale stačí, když se přesvědčíme o existenci algoritmu, který - **ale v každém stavu světa** - vypočítává z jedné tabulky tu druhou
- lambda-termíny potřebujeme pro důkazy obecných tvrzení, které aplikujeme v praxi při vytváření konkrétních datových modelů

Příklad

- $A_1 = \text{plat (Pl) daného (\#Zamest)} / 0,1:0,M$
 $A_2 = (\text{\#Zamest})\text{-s daného (\#Podnik)} / 0,M:1,M$
- $B = \text{průměrný plat (Pl) v daném (\#Podnik)} / 0,1:0,M$
- B je definovatelný nad $\{A_1, A_2\}$
toto intuitivně cítíme;
(mohl by poměr B být $1,1:0,M$??)
- **Jak se hledá konstrukce příslušné odvozovací fce f ? Tj. její algoritmus ?**
- fce f počítá v každém $w \in W$ tabulku atributu B z tabulek atributů A_1 a A_2

Příklad - pokračování

- tedy při označení $z :: \#Zamest$, $p :: \#Podnik$
- for all $p:Podnik$ do
 - write p (* první sloupec tabulky B *)
 - $Sum_1 = 0$; $Sum_2 = 0$
 - for all $z:Zamest$ do
 - if $[[A_2p]z]$ then $Sum_1 = Sum_1 + [A_1z]$
 - $Sum_2 = Sum_2 + 1$
 - endfor
 - write Sum_1/Sum_2 (* druhý sloupec tabulky B *)
- endfor

Rotace atributu

- Nechť atribut A je dán konstrukcí

$$A = \lambda_w \lambda_{x_1 \dots x_{n-1}} \rho_{x_n} ([A_w(x_1, \dots, x_{n-1})] * x_n)$$

kde $x_i :: T_i$, T_i jsou uzlové typy.

Nechť (i_1, \dots, i_n) je libovolná permutace indexů $(1, \dots, n)$.

Potom atribut $\text{rot}A$ daný konstrukcí

$$\text{rot}A = \lambda_w \lambda_{x_{i_1} \dots x_{i_{n-1}}} \rho_{x_{i_n}} ([A_w(x_1, \dots, x_{n-1})] * x_n)$$

se nazývá rotací atributu A .

- Jestliže ρ v $\text{rot}A$ zastupuje λ , říkáme že je to λ -rotace, resp. plurální rotace. Jestliže ρ v $\text{rot}A$ zastupuje ι , říkáme že je to ι -rotace, resp. singulární rotace.
- Rotace atributu A se nazývá **přípustná**, jestliže $A \leftarrow \text{rot}A$.
- Nebude-li řečeno jinak, označuje $\text{rot}A$ v dalším přípustnou rotaci.

Lemma 1

- Nechť A je dán v plurální rotaci, tj.

$$A = \lambda_w \lambda_{x_1} \dots \lambda_{x_{n-1}} \lambda_{x_n} [[A_w(x_1, \dots, x_{n-1})] x_n]$$

Potom každá jiná jeho plurální rotace je přípustná.

Přípustné singulární rotace

- $A = (\#Podnik)\text{-s ve kterých pracuje daný } (\#Zamest)$
- $rot_1 A = (\#Zamest)\text{-s daného } (\#Podnik)$
 $rot_2 A = (\#Zamest) \text{ daného } (\#Podnik)$
- Zřejmě $rot_1 A$ je přípustná a $rot_2 A$ není přípustná:
 $rot_2 A / (W \rightarrow (\#Podnik \rightarrow \#Zamest))$
tj. v každém stavu světa je definovaná pouze na podnicích,
které mají nejvýše jednoho zaměstnance
tedy:
z $rot_2 A$ nelze odvodit zpět atribut A , poněvadž
neoprávněným použitím singularizátoru jsme ztratili
informaci
- **POZN.: obrácená funkce** (viz přednáška 7) je speciální
případ rotace atributu

Věta 1

(o informační ekvivalenci a rotacích)

- Každá přípustná rotace atributu A je informačně ekvivalentní s A . Každá nepřípustná rotace A není informačně ekvivalentní s A .

Pravidlo singularity

- Z daného atributu nelze žádným formálním způsobem odvodit, které jeho rotace jsou přípustné singulární rotace.
- To že nějaká singulární rotace je přípustná je netriviální empirický fakt.
- Studium přípustnosti singulárních rotací určujeme poměr atributu.
- **PRAVIDLO:**
Při datovém modelování je třeba s každým HIT-atributem prozkoumat všechny jeho rotace, a zjistit zda má nějaké přípustné singulární rotace.
Pokud ano, do výsledného modelu zapisujeme vždy tyto přípustné singulární rotace, nikoli jiné - plurální - rotace.

Triviální odvození

- Řekneme, že atribut A vznikl z atributu B triviálním odvozením, jestliže $A \leftarrow B$, tj. pro všechna w

$$A_w = [f B_w]$$

a funkce f obsahuje nejvýše identitu, přípustně použitý singularizátor a/nebo existenční kvantifikátor.

- Triviální odvození jsou nejčastěji používána jako nástroj odvození odpovědi na dotazy nad databází. Pro **formulování dotazů** je vhodnější využívat všech funkcí jednoduchých typů (přednáška 6) a neomezovat se pouze na H-typy. Pro vlastní modelování je toto omezení (pouze na H-typy) výhodnější.

obecná rotace atributu

- Vychází z obecnějšího pojetí atributu jako funkce jednoduchého typu:

$$A = \lambda_w \lambda_{x_1 \dots x_k} \rho_{x_{k+1} \dots x_n} [A_w(x_1, \dots, x_k)] * (x_{k+1}, \dots, x_n)$$

- Nechť (i_1, \dots, i_n) je libovolná permutace indexů $(1, \dots, n)$.

Potom atribut $\text{rot}A$ daný konstrukcí

$$\text{rot}A =$$

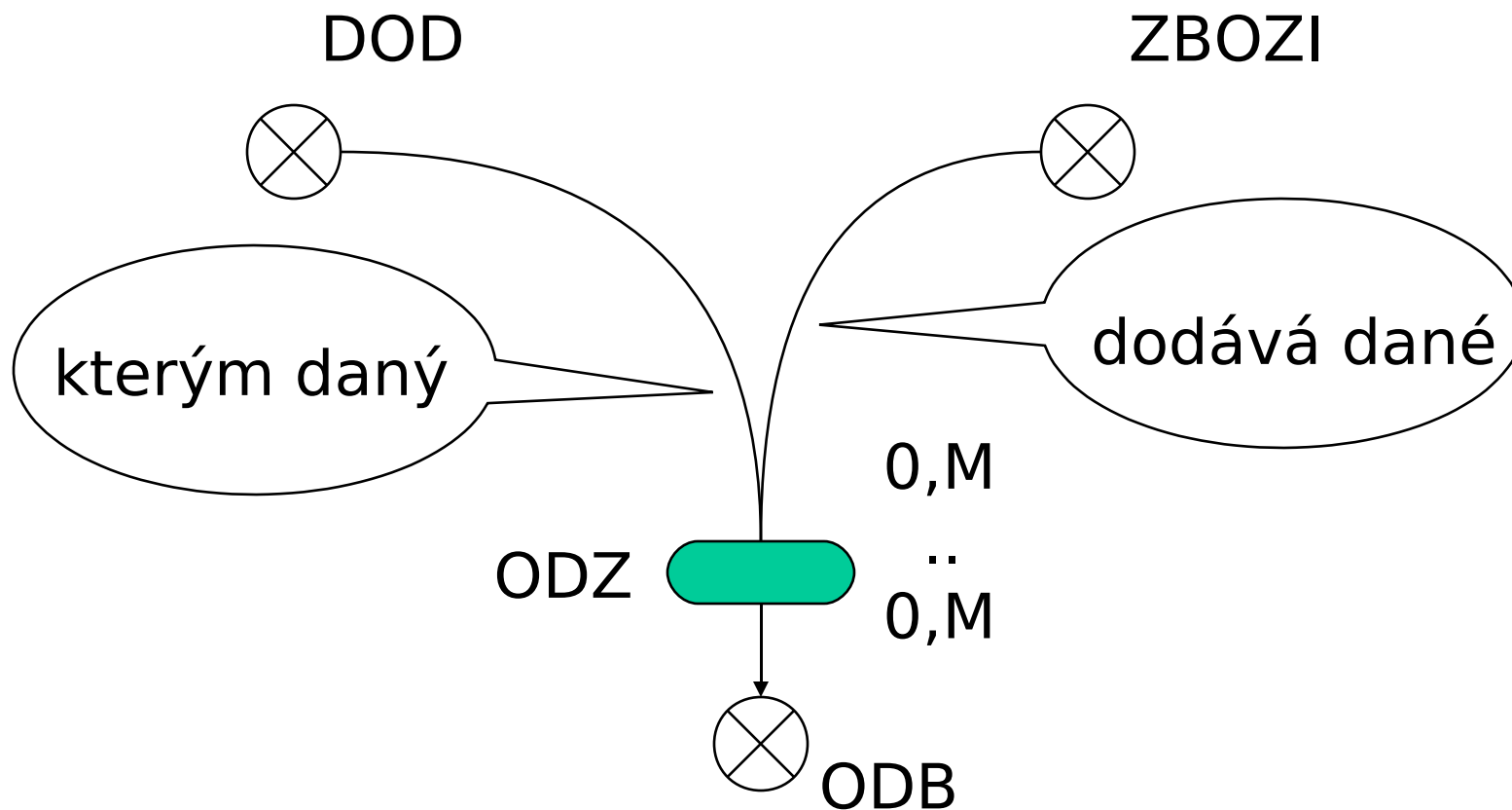
$$= \lambda_w \lambda_{x_{i_1} \dots x_{i_k}} \rho_{x_{i_{k+1}} \dots x_{i_n}} [A_w(x_1, \dots, x_k)] * (x_{k+1}, \dots, x_n)$$

se nazývá rotací atributu A .

Plurální a singulární rotace jako v původní definici.

- Používá se v důkazech a odvozování, nikoli jako modelovací konstrukt.

Příklad:



příklad - pokračování

- ROTACE:

$ZDO = (\#Zbozi)\text{-s dodávaná daným } (\#Dodavatel) \text{ danému } (\#Odberatel) / 0, M:0, M$

$DZO = (\#Dodavatel)\text{-s kteří dodávají dané } (\#Zbozi) \text{ danému } (\#Odberatel) / 0, M:0, M$

podle lemmatu víme, že jsou přípustné. Empirickým výzkumem se přesvědčíme, že žádná singulární rotace není přípustná.

- TRIVIÁLNĚ ODVOZENÉ ATRIBUTY:

$ZD = (\#Zbozi)\text{-s dodávaná daným } (\#Dodavatel) / 0, M:0, M$

$OZ = (\#Odberatel)\text{-s daného } (\#Zbozi) / 0, M:0, M$

$OD = (\#Odberatel)\text{-s daného } (\#Dodavatel) / 0, M:0, M$

- Příklad algoritmu odvození:

$OD \leftarrow ODZ$

- for all x:DOD do
 for all y:ODB do
 if $\exists z:ZBOZI ([[ODZ_w(x, z)] y])$
 then write y
 else endif
 endfor
endfor

příkazem write vytváříme tabulku odběratelů příslušných k danému dodavateli

příklad obecné rotace

- Dvojice zboží + dodavatel (#Zbozi, #Dodavatel) přiřazená k danému odběrateli (#Odberatel) taková, že onen dodavatel dodává ono zboží tomu dodavateli / 0,M:0,M
- nakreslete si schéma tohoto atributu
- Z lineárního zápisu je vidět, že obecné rotace nejsou vhodné jako modelovací konstrukty. Avšak pro úvahy o správnosti a adekvátnosti vystižení reality a pro důkazy určitých zákonitostí (viz následující přednáška) jsou potřebné.

P114

Rozložitelnost

Manipulace s HIT-atributy

10

Témata

- Podatributy
- Rozklad atributů
- Jak poznat rozložitelnost
- Věta o rozkladu
- Jádro datového modelu

Zopakování

- Atribut A je definovatelný nad množinou atributů $\{B_1, \dots, B_n\}$ právě když

$$\exists f \forall w t ([A \text{ wt}] = [f ([B_1 \text{ wt}], \dots, [B_n \text{ wt}])])$$

kde f je analytická funkce splňující základní předpoklad, w je možný svět, t je časový okamžik.

Píšeme: $A \leftarrow (B_1, \dots, B_n)$

- Množina atributů M je definovatelná nad množinou atributů N , je-li každý prvek z M definovatelný nad nějakou podmnožinou N .

Píšeme: $M \leftarrow N$

- Atribut A je definovatelný nad atributem B , když je definovatelný nad množinou $\{B\}$.

Píšeme: $A \leftarrow B$

Informační ekvivalence (zopakování)

- Nechť $M = \{A_1, \dots, A_n\}$, $N = \{B_1, \dots, B_m\}$ jsou množiny atributů a nechť $M \leftarrow N$ a zároveň $N \leftarrow M$. Potom říkáme, že M a N jsou informačně ekvivalentní. Píšeme: $M \approx N$
- Intuitivně je zřejmé, a lze dokázat, že dvě informačně ekvivalentní množiny atributů generují v každém stavu světa stejnou třídu propozic.
- Jiná intuice: z n -tice tabulek daných množinou atributů M dostaneme tutéž informaci jako z m -tice tabulek daných množinou atributů N
- VĚTA: Relace \approx na množině atributů je ekvivalence.

Triviální odvození (zopakování)

- Řekneme, že atribut A vznikl z atributu B triviálním odvozením, jestliže $A \leftarrow B$, tj. pro všechna w

$$A_w = [f B_w]$$

a funkce f obsahuje nejvýše identitu, přípustně použitý singularizátor a/nebo existenční kvantifikátor.

- Triviální odvození jsou nejčastěji používána jako nástroj odvození odpovědi na dotazy nad databází. Pro formulování dotazů a transformací atributů je vhodnější využívat všech funkcí jednoduchých typů (přednáška 6) a neomezovat se pouze na H-typy.

Podatribut

- Atribut A_1 vzniklý triviálním odvozením z atributu A budeme nazývat **podatributem** A .
Každý podatribut, který není totožný s některou přípustnou rotací A , nazýváme **vlastní podatribut**.
- Zejména A je podatributem A .
- Složitost vlastního podatributu A_1 je vždy menší než složitost A .
- Všechny rotace a odvozené atributy v posledním příkladu přednášky 9 jsou podatributy.

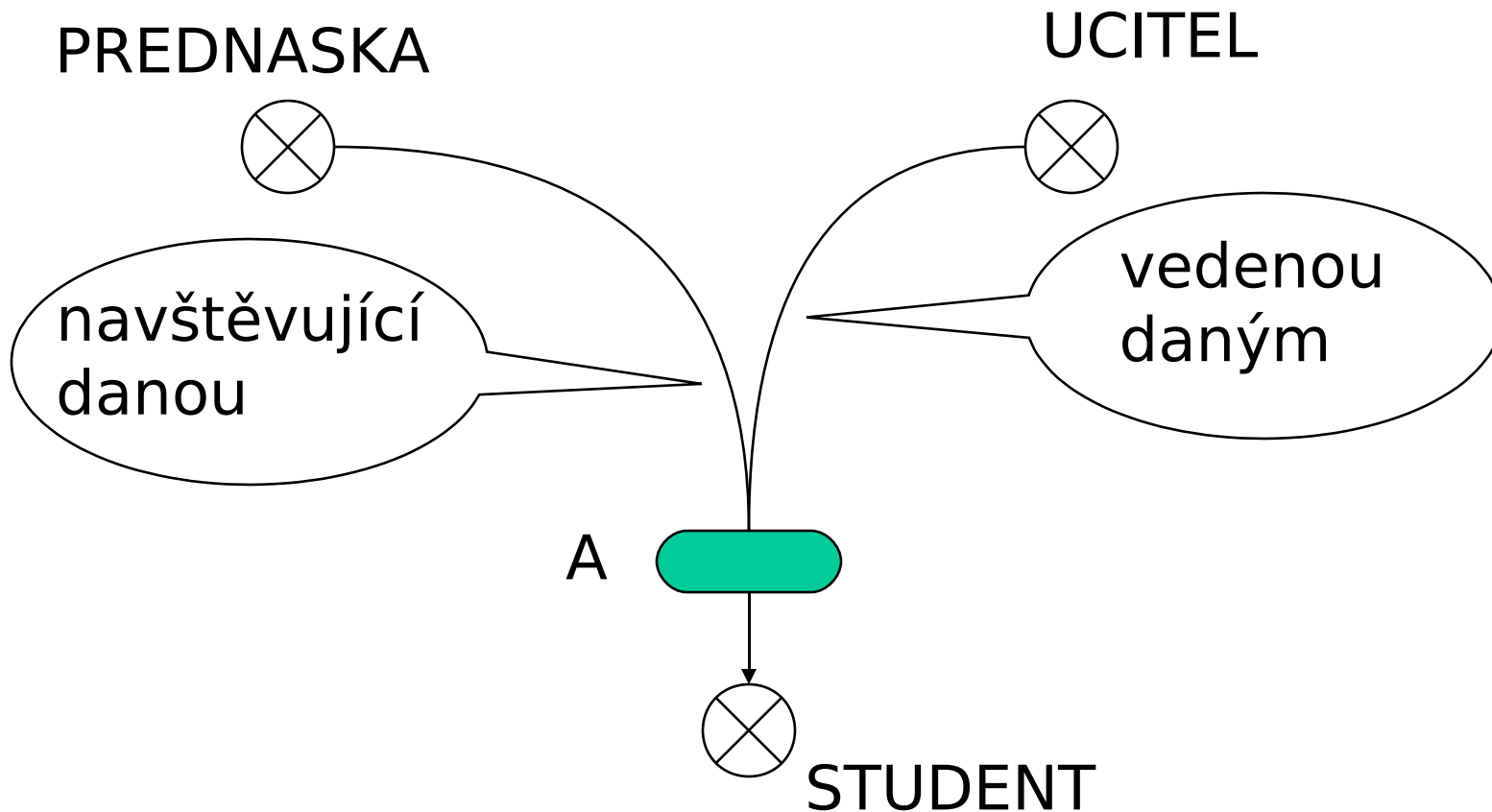
Rozklad atributu

- Atribut A je **rozložitelný** na atributy B_1, \dots, B_n , píšeme $A \diamond (B_1, \dots, B_n)$, když
 - 1) B_1, \dots, B_n jsou vlastní podatributy A , a
 - 2) $A \leftarrow (B_1, \dots, B_n)$
- **DŮSLEDEK 1:**

Jestliže $A \diamond (B_1, \dots, B_n)$, potom $A \approx \{B_1, \dots, B_n\}$.
DŮKAZ: plyne z definic.
- **DŮSLEDEK 2:**

Jestliže $A \diamond (B_1, \dots, B_n)$, potom každá jeho přípustná rotace $\text{rot}A$ je rovněž rozložitelná na B_1, \dots, B_n .
DŮKAZ: plyne z definice rotace, definice rozložitelnosti a věty 1 (viz přednáška 9).

Příklad: $A/(W \rightarrow ((\text{Prednaska}, \text{Ucitel}) \rightarrow \text{Student}))$



příklad - pokračování

- uvažme atribut:

$B_1 = (\#Ucitel)\text{-s kteří vedou danou } (\#Prednaska)$

opravdu je jich víc?

Nechť realita je, že danou přednášku vede vždy jenom jeden učitel. Potom bude správně:

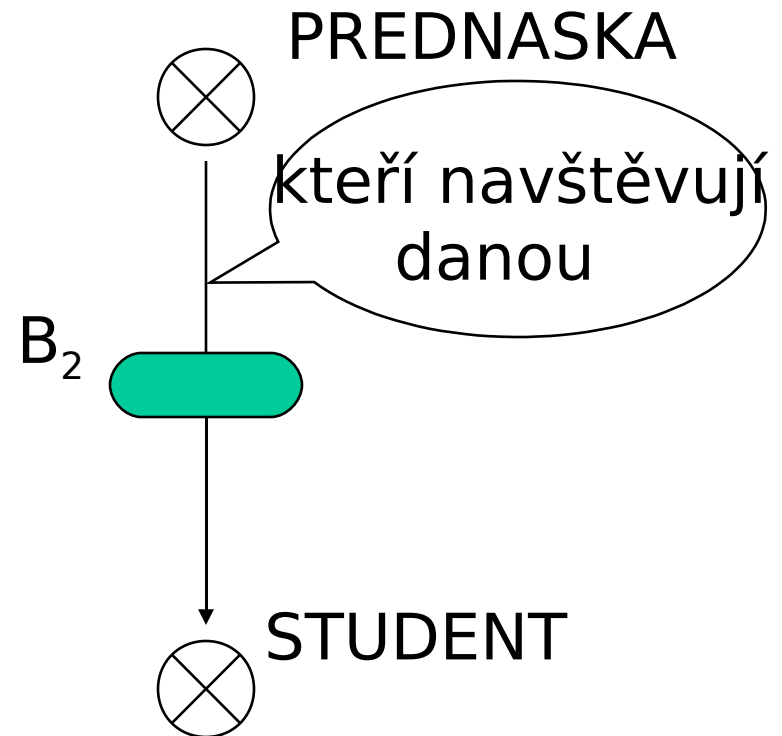
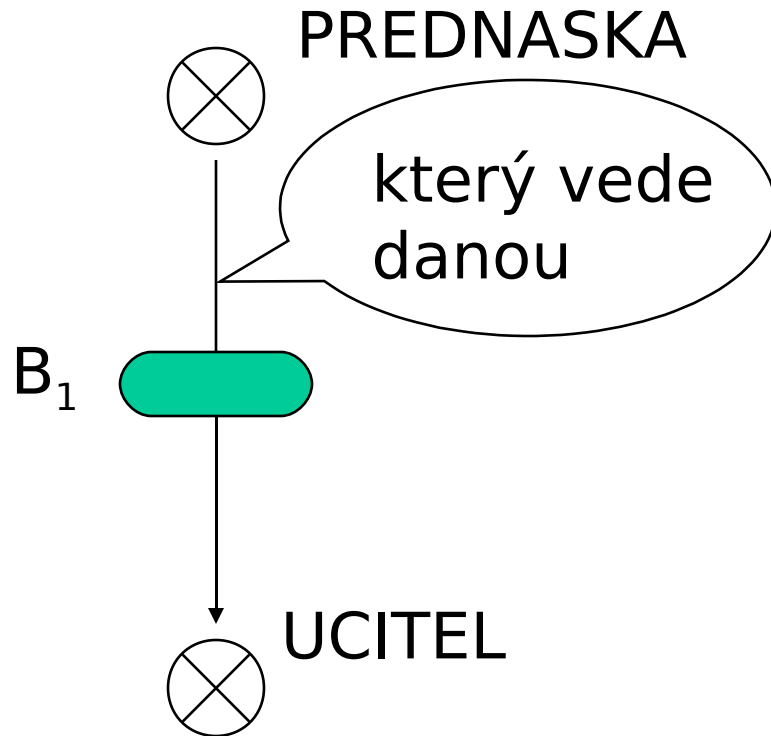
$B_1 = (\#Ucitel)$ který vede danou $(\#Prednaska)$

- $B_1 \leftarrow A$ triviálně (pomocí $=$, \exists a ι ve \mathbf{f})
- přidejme:
 $B_2 = (\#Student)\text{-s kteří navštěvují danou } (\#Prednaska)$
 $/0,M:0,M$
- $B_2 \leftarrow A$ triviálně (pomocí $=$ a \exists ve \mathbf{f})
- dokážeme napsáním algoritmu, že $A \leftarrow (B_1, B_2)$:

příklad - pokračování

- declare p:Prednaska, u:Ucitel
declare s:Student
for all p do
 for all u do
 for all s do
 if $[[B_{2w} p] s] \wedge [B_{1w} p] = u$
 then write s
 else endif
 endfor
 endfor
endfor
- ve f stačila identita a konjunkce ...

$$A \diamond (B_1, B_2), \quad A \approx \{B_1, B_2\}$$



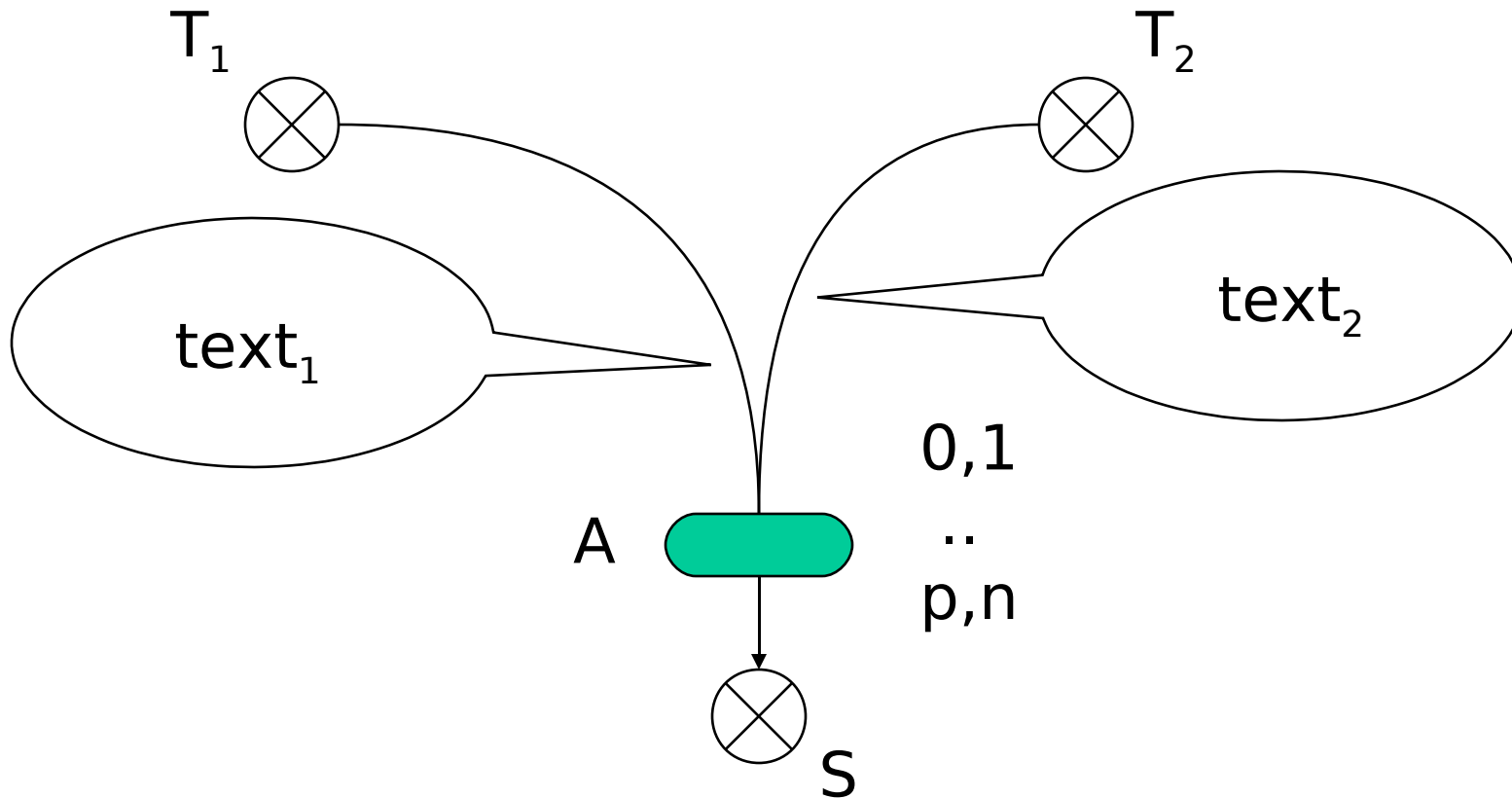
Poznámka o rozkladové konstrukci

- Nechť $A \diamond (B_1, \dots, B_n)$, a nechť $A_w = [f(B_{1w}, \dots, B_{nw})]$.
- Potom vždy f obsahuje konjunkce a identity
- V případě, že A je zadán v singulární rotaci, pak f obsahuje navíc singularizátor.

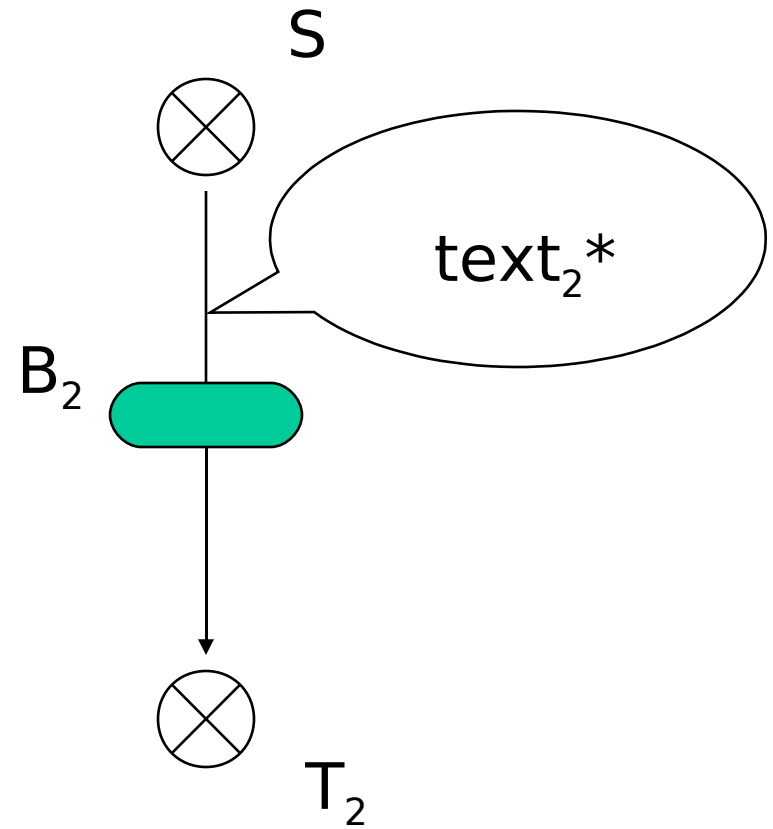
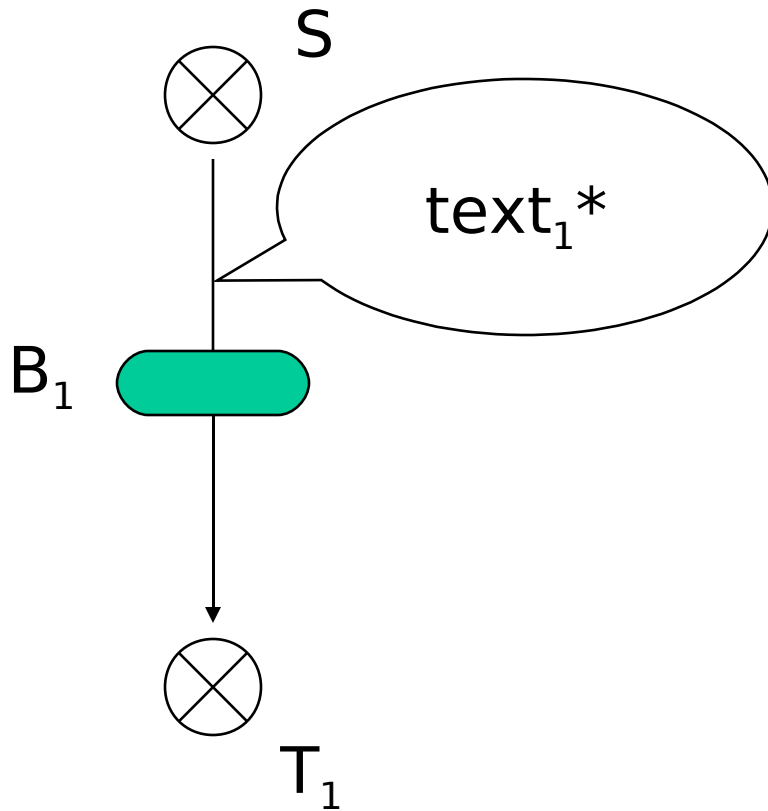
Jak poznat rozložitelnost

- (1) použitím definice:
v diskusi s expertem se často k atributu složitosti > 2 objeví jeho podatribut; pak hledáme další podatribut původního a pokoušíme se zkonstruovat (algoritmem) původní atribut (viz min. příklad)
- (2) podle horního poměru
je-li poměr atributu
 $A / (W \rightarrow ((T_1, \dots, T_n) \rightarrow S))$ resp.
 $A / (W \rightarrow ((T_1, \dots, T_n) \rightarrow (S \rightarrow \text{Bool})))$
tvaru **p,n:0,1**, tj. horní poměr obrácené funkce je 1,
pak A je rozložitelný. (viz násl. obrázek)
- (3) použitím věty o rozkladu (viz dále)

Příklad: $A/(W \rightarrow ((T_1, T_2) \rightarrow (S \rightarrow \text{Bool})))$



$$A \diamond (B_1, B_2)$$



Věta 2 (o rozkladu)

- Zobecnění obou příkladů
- Nechť A je atribut složitosti větší než 2. Nechť existuje jeho vlastní podatribut B_1 daný v přípustné singulární rotaci.

Potom $A \diamond (B_1, B_2)$, při čemž

(a) definiční obor funkce B_{2w} je stejný jako definiční obor funkce B_{1w}

(b) obor hodnot funkce B_{2w} je tvořen zbylými uzlovými typy atributu A , které nejsou v podatributu B_1 .

- Netriviálnost přesné formulace věty (DM2) je dána parcialitou uvažovaných datových funkcí: je třeba se vypořádat s tím, že jedna funkce je na daném argumentu nedefinována (píšeme \perp), a druhá - skoro identická - vrací na tomtéž argumentu $\{ \}$.

Jádro datového modelu

- Nechť \mathbf{A} je libovolná množina HIT-atributů nad \mathbf{BZT} . Jádrem množiny \mathbf{A} je taková množina atributů \mathbf{K} , pro kterou platí:
 - (1) $\mathbf{K} \approx \mathbf{A}$
 - (2) každý atribut v \mathbf{K} je nerozložitelný
 - (3) neexistuje atribut $A \in \mathbf{K}$ tak, že
$$A \leftarrow \mathbf{K} - \{A\}$$
- Tedy jádro je minimální množina elementárních atributů, poskytující danou informační schopnost

Konceptuální model

- Konceptuální datový model je tvořen
 - a) bází základních typů **BZT**
 - b) jádrem **K** množiny **A** všech relevantních (datovým modelářem navržených) atributů
 - c) množinou **C** integritních omezení (formulovaných datovým modelářem nad atributy z **K**)
- Použití: IDM, LDM, Object-Class Model, UML, ...

DM metodou HIT – minikurs

(jak pracovat s příručkou)

- Zopakování a přehled pojmů
- Příklady k jednotlivým jevům
- Metodická doporučení
 - Pojmenování
 - Zkoumání rozložitelnosti
 - Zkoumání odvoditelnosti
- Zadání semestrální práce

P114

Doladění a transformace do ERD

Nadtypy, podtypy, identifikace

Transformace

11

Témata

- vazební a popisné atributy
- nadtypy, podtypy
- identifikace
- ekvivalentní vyjádření v RDBMS, ...
- algoritmus transformace

redukováaná hierarchie typů pro DM (zopakování)

- E-typy, D-typy
- uzlové typy
- n-ticové typy
- základní typy
- prvky H-typů



Vazební a popisné atributy

- Nechť A je atribut složitosti 2, tj.
 - (a) $A / (W \rightarrow (T \rightarrow S))$
 - (b) $A / (W \rightarrow (T \rightarrow (S \rightarrow \text{Bool})))$kde
 T, S jsou uzlové typy, z nichž alespoň jeden je E-typ.
- Je-li jeden z typů (T nebo S) D-typem, nazýváme A popisným (deskriptivním) atributem.
- V opačném případě nazýváme A vazebním (vztahovým) atributem.
- Je-li A deskriptivní atribut typu (b), kde S je D-typ, říkáme že S je význačným popisným typem pro E-typ T

souvislost s databázovým schématem

- Množina všech deskriptivních atributů daného E-typu T je podkladem pro strukturu objektové třídy (entity) v databázovém schématu (vyjma atributů s význačnými popisnými typy)
- Vazební atributy jsou podkladem pro strukturu vazeb v databázovém schématu (a popisné atributy s význačným popisným typem také)
- Atributy složitosti větší než 2 jsou podkladem pro strukturu vazebních tříd (vazebních entit) v databázovém schématu (pro realizaci složitějších vazeb)

Pragmatický pohled na E-typy

- Když modelujeme realitu, vždy vycházíme z daného stavu představ o tom, které prvky tvoří jednotlivé entitní sorty. Tento stav představ je závislý na stavu světa $w \in W$, ve kterém se nacházíme.
- Stav představ o tom, které prvky tvoří danou entitní sortu, budeme nazývat **populace** entitní sorty. Populaci entitní sorty T ve stavu světa w budeme značit T_w .
- To nic nemění na definici entitní sorty pomocí rozumného časového okolí R přítomnosti a aktuálního světa w_a v definici z přednášky č. 6.
- Pragmatický pohled zaujímáme a populaci T_w zavádíme proto, že exaktní definice entitní sorty nám nedává použitelný návod k tomu, jak se sortami jako s množinami nějakých prvků pracovat.

definice entitní sorty (zopakování)

Nechť $R \subseteq \text{Tim}$ je rozumné časové okolí (bylo-je-bude) přítomnosti. Nechť $r \in R$ je časový okamžik a w_a je aktuální svět.

Nechť T_1, \dots, T_m jsou ne nutně různé typy nad **EB**.

Nechť $P_i / (((\text{Bool}T_i)\text{Tim})\text{Wrd})$ jsou konkrétní vlastnosti přisouditelné T_i -objektům.

Označme $C(P_i, r, w_a)$ třídu T_i -objektů generovanou vlastností P_i v daném časovém okamžiku r a aktuálním světě w_a .

Potom:

$\bigcup_{r \in R} C(P_i, r, w_a)$ je **entitní sorta** definovaná vlastností P_i .

$\bigcup_{i=1..m} \bigcup_{r \in R} C(P_i, r, w_a)$ je **entitní sorta** definovaná disjunkcí vlastností P_i , $i=1, \dots, m$.

$\bigcap_{i=1..m} \bigcup_{r \in R} C(P_i, r, w_a)$ je **entitní sorta** definovaná konjunkcí vlastností P_i , $i=1, \dots, m$.

Entitní sorta je tedy extenze, ^{P114, 115} nezávislá na stavu světa.

Nadtypy, podtypy

- Nechť T, S jsou uzlové typy. Nechť T_w označuje populaci sorty T ve stavu světa w . Nechť pro každý stav světa $w \in W$ platí

$$T_w \subseteq S_w$$

Potom T nazýváme **podtypem** S , resp. S nazýváme **nadtypem** T .

- Jestliže T a S jsou ve **vztahu nadtyp-podtyp**, pak buďto oba jsou E-typy, nebo oba jsou D-typy.
(Plyne přímo z definice)
- Vztah nadtyp-podtyp zapisujeme (v HITu) přímo do definic příslušných typů.
- Oproti zvyklostem zavádíme vztah nadtyp-podtyp nejenom pro E-typy, ale obecněji pro všechny uzlové typy.
- Z definice vyplývá, že vztah nadtyp-podtyp je dán prostě množinovou inkluzí. (Všechny sorty jsou extenze !!!)

Příklad:

$$\text{PLAT} \subseteq \text{PrirozCisla}$$

$$\#ZAMESTNANEC \subseteq \#OSOBA$$

Příklady, doporučení

- Pragmatické použití populace entitních sort:
- $\#Zamestnanec_w \subseteq \#Osoba_w$
 $\#Student_w \subseteq \#Osoba_w$
 $\#Zbozi_w \subseteq \#Produkt_w$
 $\#Vyroba_w \subseteq \#Produkt_w$
 $\#Produkt_w \subseteq \#Artikl_w$
 $\#Sluzba_w \subseteq \#Artikl_w$
...
- Pro rozhodování o tom, zda dva uzlové typy (a zejména E-typy) jsou ve vztahu nadtyp-podtyp, jsou nutné definice těchto typů.
- Definice E-typů a jejich vztahy nadtyp-podtyp je nutné **vyvažovat** se sémantikou a poměrem atributů, ve kterých vystupují.

Vyvažování - příklady , doporučení

- (IČO) daného (#Podnik) / 1,1:0,1
 $\#Podnik_w \subseteq \#Organizace_w$ pro $\forall w$
(IČO) dané (#Organizace) / 0,1:0,1
„rozšířením funkce na nadtyp se ona stane parciální“
- (Plat)-s dané (#Osoba) / 0,M:0,M
 $\#Zamestnanec_w \subseteq \#Osoba_w$ pro $\forall w$
(Plat) daného (#Zamestnanec) / 1,1:0,M
„restrikcí funkce na podtyp se ona stane totální, a dokonce definicí zaměstnance jako zaměstnance „našeho“ podniku (kde má každý zaměstnanec jeden plat) se stane ona funkce jednoznačnou“
- „Ztotálňování“ parciálních funkcí je vhodný prostředek k vyčleňování podtypů daného E-typu
- Definice E-typu často rozhoduje o horním poměru atributu (viz uvedený příklad)

Princip hierarchie nadtyp-podtyp

- Nechť je dána hierarchie nadtyp-podtyp
 $\dots \subseteq P_{2w} \subseteq P_{1w} \subseteq T_w \subseteq N_{1w} \subseteq N_{2w} \subseteq \dots$
- pro každý objekt x/T platí
 $\exists x_i \in N_i (x_{iw} = x_w)$ pro $i = 1, 2, \dots$
tj. x má obrazy ve všech nadtypech T takové, že pro každý stav světa w tyto obrazy a x konstruuji (triviálně) tutéž extenzi

Princip hierarchie nadtyp-podtyp

- pokračování

- pro každý atribut $A/(W \rightarrow (T \rightarrow Z))$, kde Z je základní typ nebo $Z=(Z_1 \rightarrow \text{Bool})$ pro Z_1 uzlový nebo n-ticový typ, platí:

$$\exists A_i^R/(W \rightarrow (P_i \rightarrow Z)) (\forall x \in P_i ([A_i^R x] = [A_w x]))$$
 pro $i = 1, 2, \dots$

$$\exists A_i^E/(W \rightarrow (N_i \rightarrow Z)) (\forall x \in T ([A_i^E x] = [A_w x]))$$
 pro $i = 1, 2, \dots$
 tj. je možné zužovat nebo rozšiřovat atributy přechodem k pod- nebo nad- typům
- totéž lze formulovat i pro integritní omezení

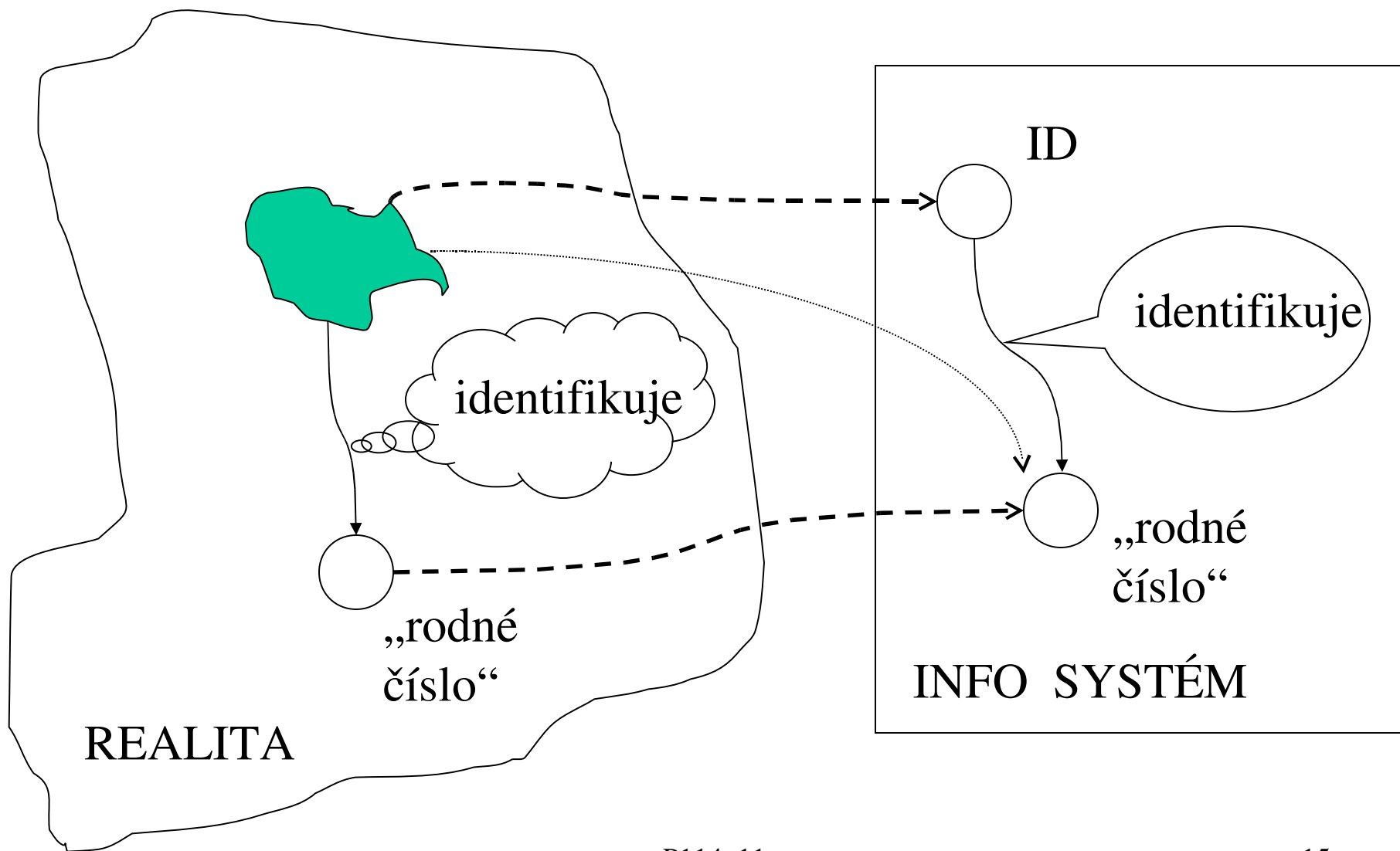
Identifikace

- Prvky uzlových typů je třeba v každém stavu světa
 - a) od sebe navzájem rozlišit
 - b) rozpoznat, že se jedná o jeden a týž objekt
- Prvky D-typů jsou identifikovány samy sebou - svojí hodnotou (jsou reprezentovatelné)
- Prvky E-typů musí být identifikovány pomocí hodnot některých D-typů (poněvadž samy nejsou reprezentovatelné)
- Identifikaci potřebujeme při práci s prvky entitních a deskriptivních sort (přednáška 6) v informačním systému. Avšak návrh identifikace je podstatnou součástí modelování.
- Identifikace musí „fungovat“ pro všechny stavy světa !

Princip identifikace

- Mějme konceptuální model $M = (\mathbf{BZT}, \mathbf{K}, \mathbf{C})$
- Nechť $E \in \mathbf{BZT}$ je E-typ. Potom v \mathbf{K} musí existovat takový atribut
 $A / (W \rightarrow (E \rightarrow D))$,
že platí:
 - 1) D je D-typ
 - 2) A je totální funkce
 - 3) $\text{rot}A / (W \rightarrow (D \rightarrow E))$ je přípustná singulární rotace
- A se nazývá identifikační atribut E-typu E a D se nazývá identifikací typu E (v databázi se pak většinou označuje ID).
- Identifikace, přiřazená jednomu konkrétnímu objektu sorty E , se nesmí žádnou aktualizací měnit.
- Identifikace je „nálepkou“ na příslušnou jednotlivinu z množiny Univ, pokud E je podmnožinou UNIV.

Princip izomorfního zobrazení světa



Kandidát identifikace

- Nechť E je E -typ, T je základní typ, $T \neq \text{Bool}$. Jestliže v konceptuálním modelu M existuje atribut $A / (W \rightarrow (E \rightarrow T))$, který je totální funkcí, pak A nazýváme kandidátem identifikačního atributu a T nazýváme kandidátem identifikace E -typu (entitní sorty) E
- Kandidátem identifikace je např. ono „rodné číslo“
- Kandidáti identifikace jsou dáni atributy z reálného světa, nikoli jako uměle zavedená ID.

Klíč

- Nechť E je E -typ, T_i je uzlový typ, $i=1,\dots,n$. Nechť v \mathbf{K} existují atributy

$$A_i / (W \rightarrow (E \rightarrow T_i)),$$

které jsou kandidáty identifikačních atributů.

$x::E$, $y::(T_1,\dots,T_n)$, $\Lambda_{(i=1..n)}$ značí konjunkci n členů

Nechť $A = \lambda w \lambda x \iota y (\Lambda_{(i=1..n)}([A_i]_w x] = y_{(i)}))$.

Nechť platí

(1) singulární rotace $\text{rot}A = \lambda w \lambda y \iota x ([A_w]x] = y)$

je přípustná.

(2) $\{T_1,\dots,T_n\}$ je minimální množina, pro kterou platí (1)

Potom n -tici (T_1,\dots,T_n) nazýváme klíčem typu E

- Klíče jsou „uživatelské“ identifikace pro E .

Pravidlo identifikace nadtypů a podtypů

- Nechť E je nadtypem E_1, \dots, E_k . Nechť
 $\forall w (\forall ij (E_i \cap E_j = \{ \}))$ a
 $\bigcup_{(i=1..k)} E_i = E$, (\cup značí množinové sjednocení)
- Jestliže pro každé E_i existuje v \mathbf{K} identifikační atribut $A_i / (W \rightarrow (E_i \rightarrow D_i))$, D_i je D-typ, pak pro E nemusí v \mathbf{K} existovat identifikační atribut. Říkáme, že E je identifikováno implicitně.
- Napište konstrukci, která by konstruovala identifikační atribut E pomocí A_i .
- Jestliže pro E existuje v \mathbf{K} identifikační atribut A , pak A je identifikačním atributem i pro E_1, \dots, E_k . Říkáme, že E_1, \dots, E_k jsou identifikovány implicitně.
- Podle jakého principu to platí?

... a incidenční atributy

- Nechť pro E a některé E_j (resp. pro všechna) existuje v \mathbf{K} identifikační atribut (v takovém případě říkáme, že E nebo E_j jsou identifikovány explicitně).
- Potom musí v \mathbf{K} existovat atributy $I_j / (W \rightarrow (E_j \rightarrow E))$ takové, že
 - a) I_j je totální funkce
 - b) $\text{rot}I_j / (W \rightarrow (E \rightarrow E_j))$ je přípustná singulární rotace
- dokažte, že $\text{rot}I_j$ je surjekce E na E_j
- Jinak bychom totiž nedokázali rozpoznat, že objekt $x \in E_j$ je totožný sám se sebou, tj. s objektem $x \in E$ objekt $x \in E$ má totiž jinou identifikaci než objekt $x \in E_j$
- Atributy I_j nazýváme incidenčními atributy.

Přípustné transformace

- Transformací množiny atributů **A** rozumíme takovou množinu atributů **B**, pro kterou platí:
$$B \in \mathbf{B} \Rightarrow (B \in \mathbf{A} \vee B \leftarrow \mathbf{A})$$
- Transformace se nazývá přípustná, jestliže $\mathbf{B} \approx \mathbf{A}$. Také říkáme, že když transformace je přípustná, tak zachovává informační schopnost.
- Transformací konceptuálního modelu $M = (\mathbf{BZT}, \mathbf{K}, \mathbf{C})$ rozumíme transformaci množiny **K** a odpovídající přeformulování všech integritních omezení z **C** pomocí transformovaných atributů.
- pro navrhování DB a IS jsou důležité přípustné transformace konceptuálního modelu

Věta 3 (o intenzionálních relacích)

- Nechť A je libovolný HIT-atribut,
 $w::W$, $x_i::T_i$, $y::T$, T_i , T příslušné základní typy, jak vyžaduje definice,
 $A = \lambda w \lambda x_1 \dots x_n \lambda y ([A_w(x_1, \dots, x_n)] * y)$.
Potom z A lze přípustnou transformací odvodit atribut
 $A_1 / (W \rightarrow ((T_1, \dots, T_n, T) \rightarrow \text{Bool}))$,
ve kterém n -ticový typ bereme v normálním tvaru (tj. neobsahující
vnořené n -ticové typy)
- Atribut A_1 nazýváme intenzionální relace (relation in
intension).
- **DŮSLEDEK: Ke každému konceptuálnímu modelu M
existuje schéma relací (v relačním modelu dat) se
stejnou informační schopností.**

Důsledky

- Každý konceptuální model vytvořený metodou HIT lze implementovat pomocí RDBMS se zachováním informační schopnosti.
- To co dokážeme implementovat v RDBMS, je možno vyjádřit rovněž v ERD, tj. také v běžně používaných prostředcích CASE (SDW, SELECT SE, Rational Rose, ...) a v běžně používaných standardech jako je UML (Unifying Modeling Language)
- *HIT metodu lze použít jako myšlenkový aparát a způsob zápisu výsledků při práci s kterýmkoli ze zmiňovaných nástrojů resp. standardů.*

Binarizační princip (1)

- Nechť A je libovolný nerozložitelný HIT-atribut složitosti větší než 2, $w::W$, $x_i::T_i$, T_i , příslušné základní typy, jak vyžaduje definice,
$$A = \lambda w \lambda x_1 \dots x_{n-1} \rho x_n ([A_w(x_1, \dots, x_{n-1})] * x_n).$$
- Zavedeme tzv. konkatenovaný typ:
$$R = \text{cn}(T_1, \dots, T_n)$$
- $R \subseteq (T_1, \dots, T_n)$ je podmnožina kartézského součinu tvořená právě těmi n -ticemi $\langle x_1, \dots, x_n \rangle$, které jsou dány tabulkou $[A \ w]$

Binarizační princip (2)

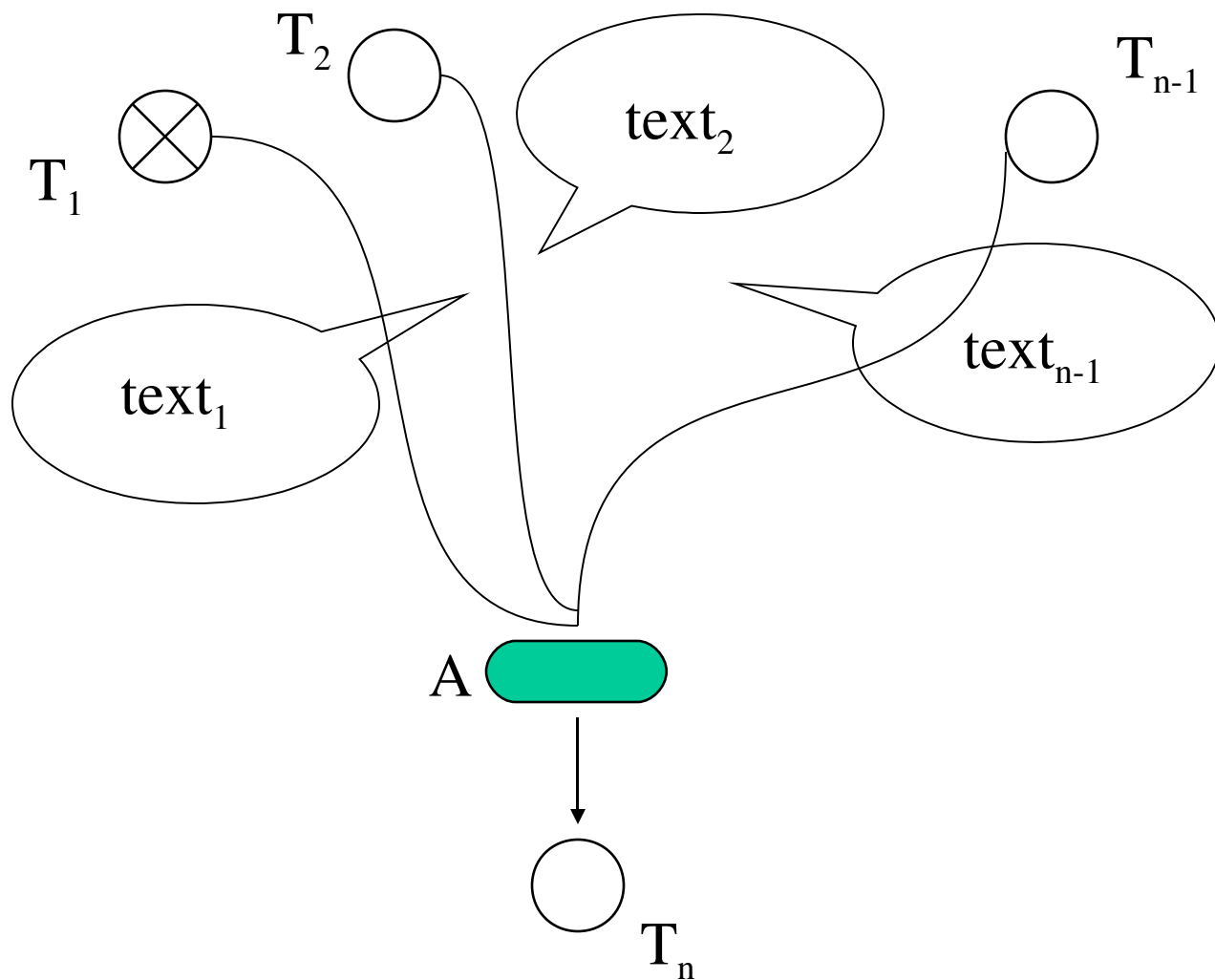
- Definujeme pro $i = 1..n$ projekce $B_i / (W \rightarrow (R \rightarrow T_i))$

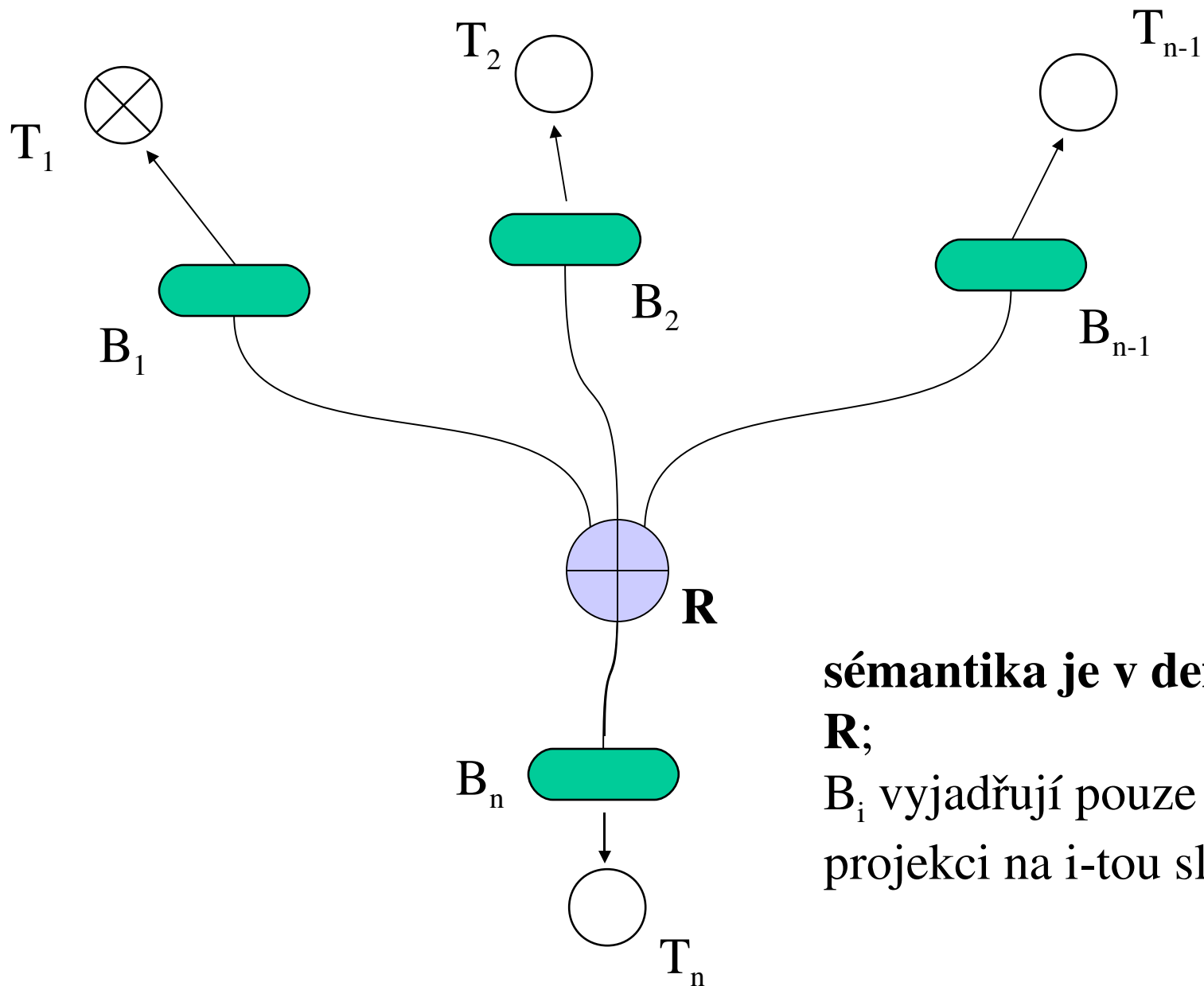
takto:

$$B_i = \lambda w \lambda r \iota x_i (([A_w(r_{(1)}, \dots, r_{(n-1)})] * r_{(n)}) \wedge r_{(i)} = x_i)$$

- Potom B_i jsou totální funkce na R , a $\{B_1, \dots, B_n\} \approx A$
- **DŮSLEDEK: všechno co dovedeme zapsat (HIT-) konceptuálním modelem, lze zaznamenat pomocí sítě uzlů a hran.**

atribut A - nerozložitelný





sémantika je v definici R ;
 B_i vyjadřují pouze
 projekci na i -tou složku

Algoritmus transformace

$M = (\mathbf{B}, \mathbf{K}, \mathbf{C})$, \mathbf{B} je báze tvořená uzlovými typy a Bool.

Následující transformace do ERAM zachovává (nesnižuje!!!) informační schopnost:

- (1) Pro každé $A \in \mathbf{K}$, jeli složitost A větší než 2, zavést konkatenovaný typ R a A nahradit konfigurací projekcí B_i
POZN.: tím zavedeny vztahové entity z ERAM
- (2) Všechna singulární omezení, platná pro A , přeformulovat jako tvrzení konzistence pomocí B_i
- (3) Každý E-typ, který je explicitně identifikovaný a každý konkatenovaný typ reprezentovat entitou (po řadě kernel a vztahovou) v ERAM

algoritmus - pokračování

- (4) Popisné atributy k E-typu resp. konkatenovanému typu z (3), které neobsahují význačný popisný typ, reprezentovat jako ERAM atributy příslušné kernel nebo vztahové entity
- (5) Popisné atributy, které obsahují implicitně identifikovaný typ, nahradit dle principu hierarchie nadtypů-podtypů příslušným rozšířením resp. zúžením v rámci záměny implicitně identifikovaného typu za explicitně identifikovaný. Dále pokračovat krokem (4).
- (6) Význačné popisné typy reprezentovat samostatnými entitami v ERAM (tzv. charakteristickými entitami)

algoritmus - pokračování

- (7) Vazební atributy, v nichž každý E-typ je explicitně identifikovaný, reprezentovat hranou v ERAM, jejíž typ (1-1, 1-M, M-M) je dán horním poměrem atributu
- (8) Je-li ve vazebním atributu implicitně identifikovaný E-typ, nahradit jej dle principu hierarchie nadtypů-podtypů příslušným rozšířením resp. zúžením v rámci záměny implicitně identifikovaného typu za explicitně identifikovaný. Dále pokračovat krokem (7).
- (9) Všechny definice E-typů z **B** zapsat jako definice kernel entit a všechny sémantiky atributů z **K** zapsat jako sémantiky hran resp. vztahových entit v ERAM.

KONEC

Praktické použití - poznámky

- Výsledkem je model v ERAM, který má všechny entity (brány jako relační tabulky) v BCNF, dokonce v 4NF a 5NF.
- Zkušený datový modelář provádí uvedený algoritmus (rovnou při návrhu datového modelu) v hlavě a přímo zapisuje výsledek v ERAM pomocí vhodného CASE.
- Pro vysvětlení složitých vazeb resp. pro jejich analýzu a rozpoznání pravého stavu věcí, se doporučuje zakreslit situaci pomocí sémantických diagramů („kytiček“), a nad nimi diskutovat a řešit problém.
- Krok (9) algoritmu je nepominutelný a závazný !!
- Příklady výsledků - viz přednáška 12.

P114

Příklady datových modelů

Úrovně datových modelů

IDM Organizace práce, ...

12

Témata

- IDM - LDM -PDM
- Definice business potřeb
- Výsledný produkt datového modelování
- Zápis v CASE a pod.
- Příklad IDM Organizace práce
- Příklad IDM Sklady

IDM - LDM -PDM

- IDM - ideální datový model: entity + vazby
terminologie LBMS
= konceptuální datový model
cíl je definovat problém, definovat potřebnou informační schopnost
- LDM - logický datový model: entity + vazby + popisy
cíl je detailní zadání požadované informační schopnosti pro konkrétní realizaci IS
- PDM - fyzický datový model: tabulky + klíče + indexy
cíl je fungující IS v provozu

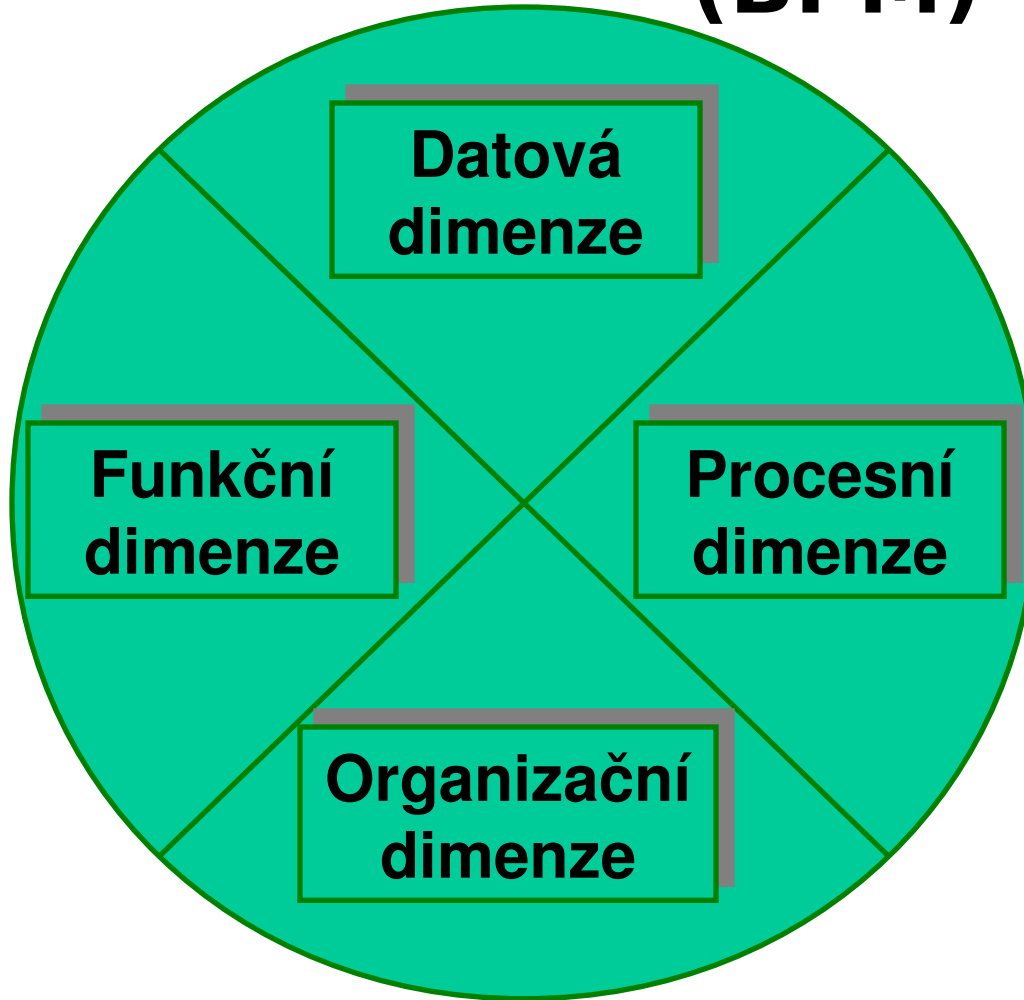
Začlenění do procesu realizace IS

- Kdy - ve které fázi - se co dělá
- Nástroje: CASE, BPMT, ...
 - myšlenkové postupy pro jejich naplňování
- Postupy a techniky SW inženýrství
 - klasická strukturovaná analýza
 - OO přístup
 - *vždy role CM (conceptual modeling)*
- návaznost předmětů:
 - Sochor: Analýza a návrh systémů
 - Král: Softwarové metody výstavby IS

Definice business potřeb

- Co se dělá při definici business potřeb
 - Funkce + Procesy + Data + Organizace
 - vzájemná provázanost
- IDM resp. konceptuální datový model
- třeba ve formě návrhu object-class diagramu v OO- přístupu a nástrojích

Business Process Model (BPM)



Dimenze

BPM

Datová - S ČÍM?

- ☑ typy dat ve firmě

Funkční - CO?

- ☑ schopnosti, dovednosti

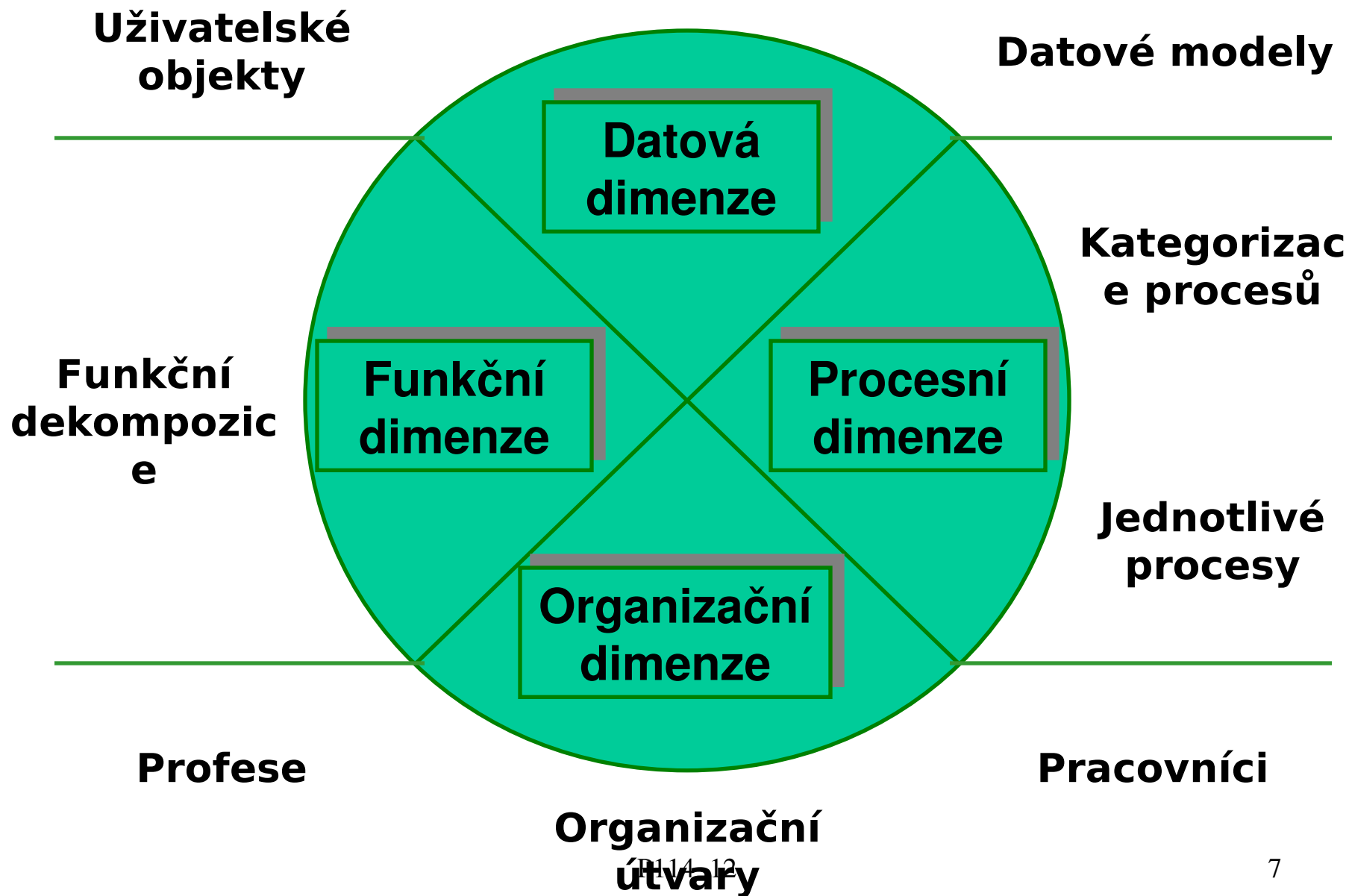
Organizační - KDO?

- ☑ organizační uspořádání

Procesní - JAK?

- ☑ firemní procesy

Modely BPM



Výsledný produkt Datového Modelování

- viz příklady dále
- diagram ERD-like
- textový popis !!!
 - definice entit
 - sémantika vazeb

Zápis v CASE a pod.

- každý rozumný CASE a podobný nástroj má
 - podporu tvorby různých diagramů
 - možnost psaní textů (poznámky, descriptions) k jednotlivým prvkům diagramu
 - asociace mezi diagramy
- Zvolit který diagram použijeme pro zápis datového modelu
- Zvolit grafické prvky, které budete používat, a zaznamenat jednoznačné definice jejich významu
- Zvolit prostředky pro zápis sémantiky
- Vytvořit z předešlého standard

Příklady datových modelů

IDM RVV /Organizace práce

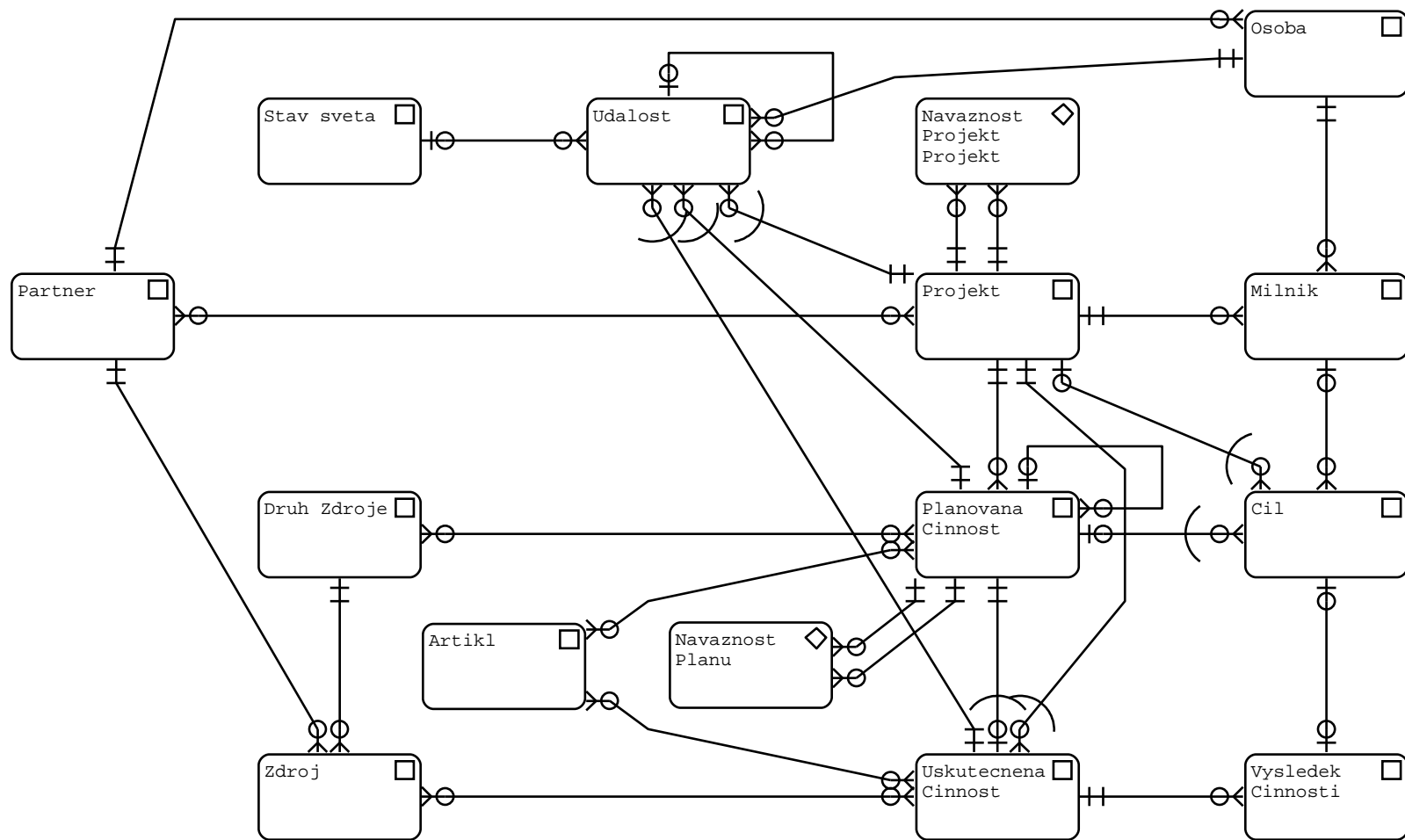
IDM Sklady

- použité prvky v diagramu
- zápis definic kernel- entit
- zápis sémantiky vazeb
- zápis sémantiky vazebních entit
- dvojí zápis kardinality vazeb (kontrola)

RVV /OrgPra

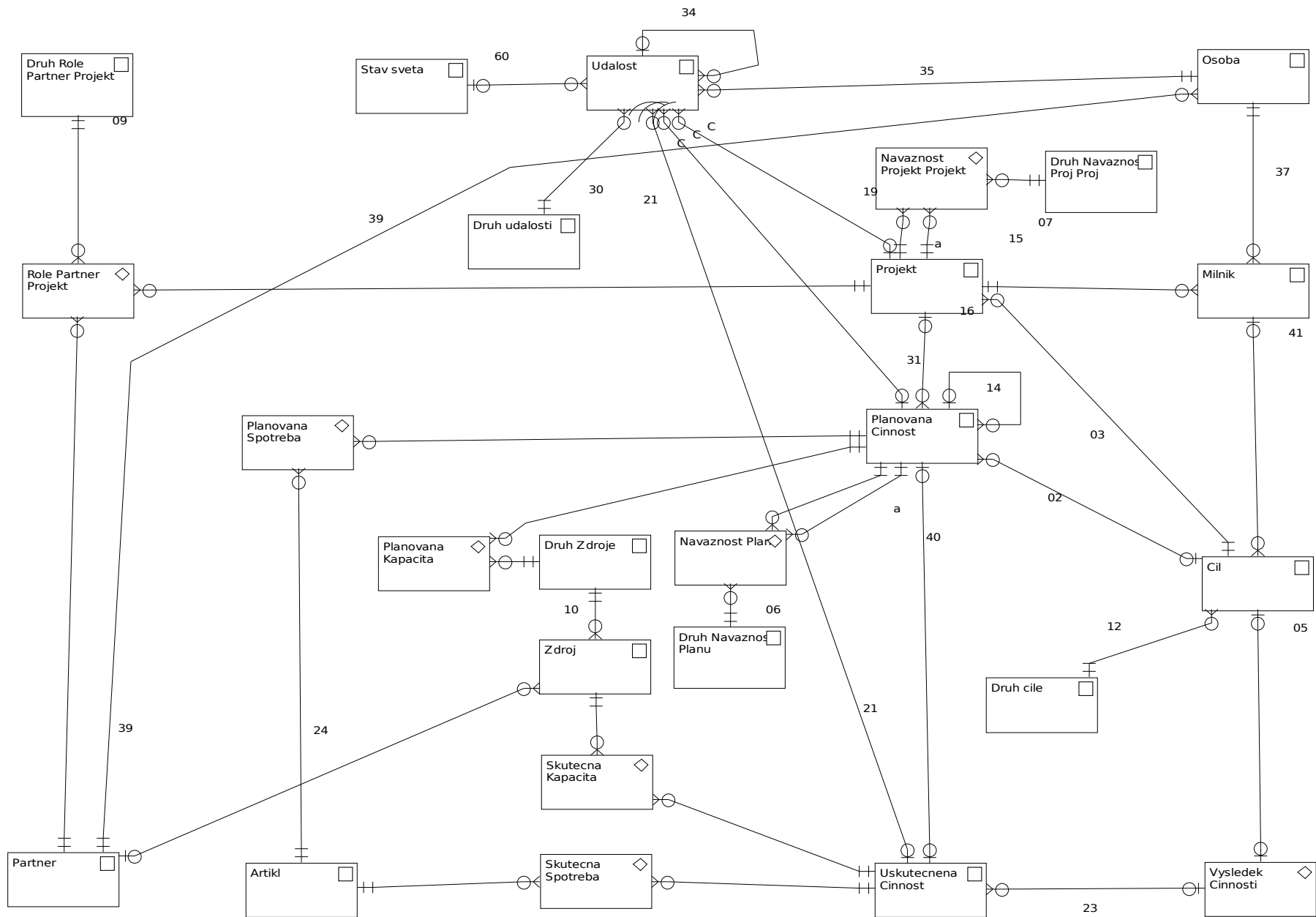
- model pro řízení soustavy projektů
- vzájemné ovlivňování projektů soustavy
- monitorování stavu procesů (dimenze „časový plán“ a „rozpočet“ projektového Troj-imperativu)
- přeplánování

Konceptuální model OrgPra



Cesta k LDM

- doplnění atributů
- doplnění klasifikací objektů
- detailní popis vztahů
- ... tvorba skutečného zadání pro realizaci

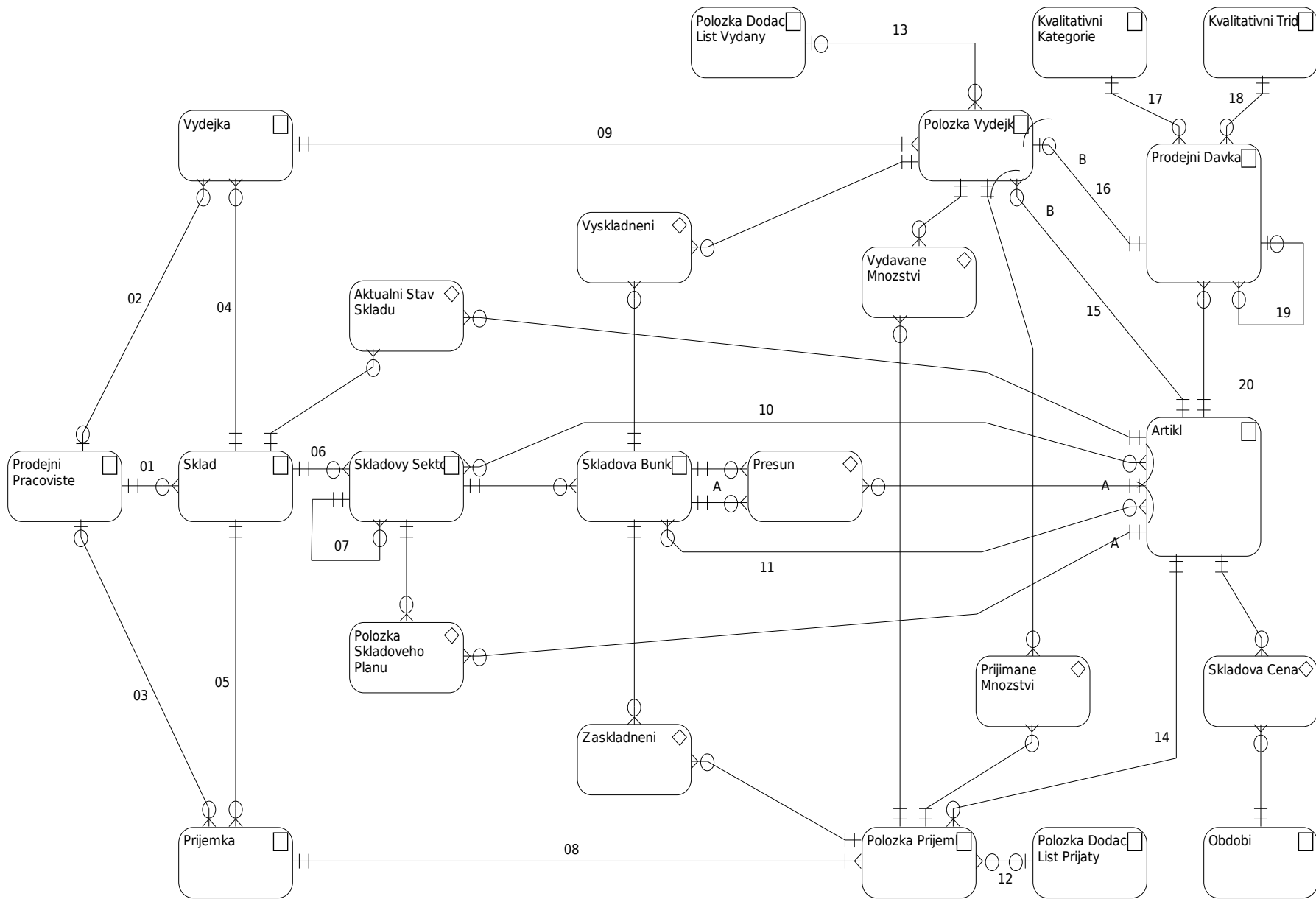


Obecné Sklady

- pohyb hmotných produktů a jejich uložení za účelem zefektivnění a zvýšení dosažitelnosti zboží na jeho cestě od výrobce k zákazníkovi
- o co jde viz schéma vyskladnění
- jak je to řešeno viz následující konceptuální model

O co ve Skladech jde

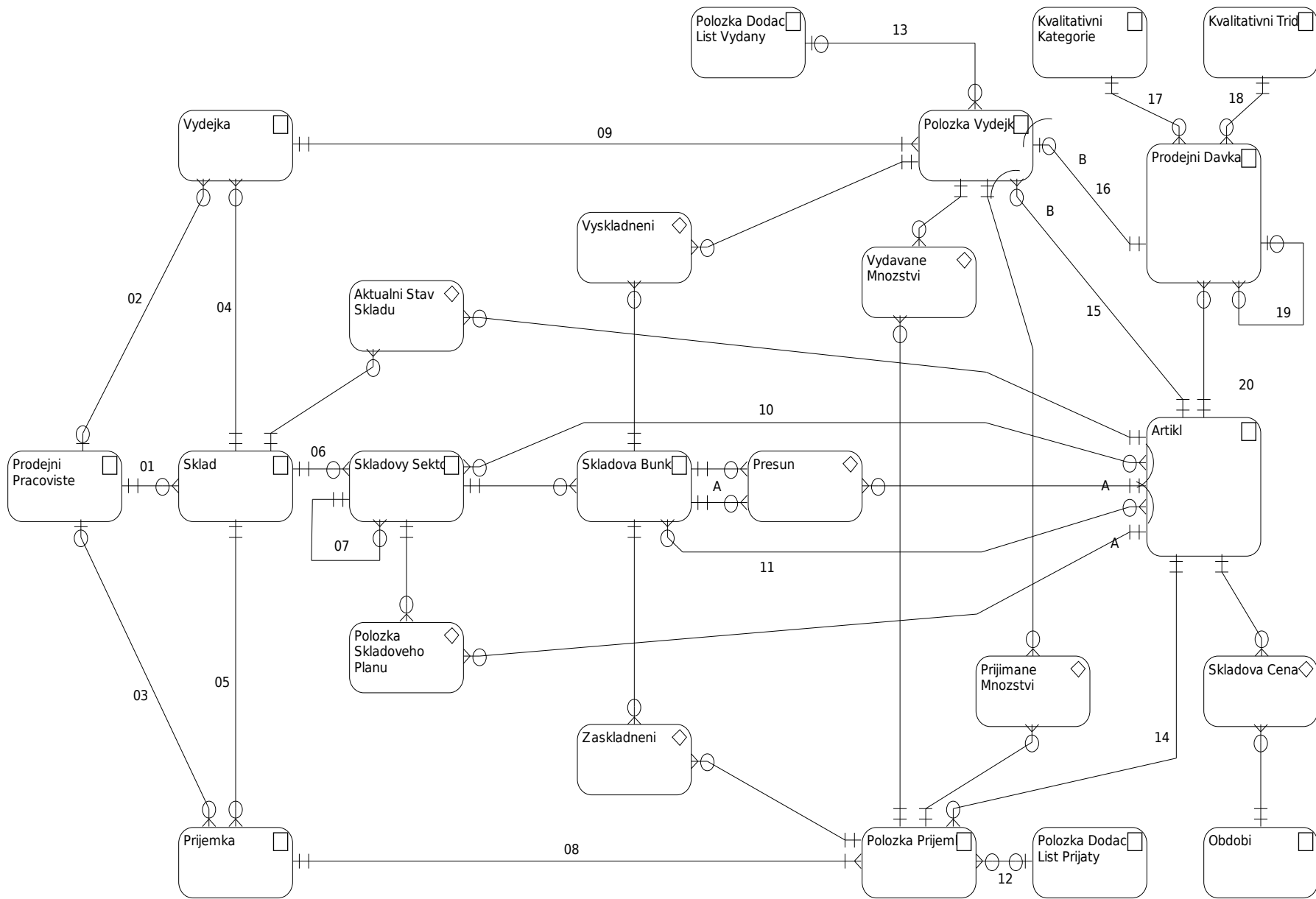
- Diagram procesu vyskladnění – viz obr.
- Obsluha zákazníka – vyskladňování
- Příprava skladu – naskladňování
- Optimalizace skladu – přesuny
- Závislost na povaze skladovaného produktu



IDM Sklady definuje informační schopnost komponenty informačního systému pro zabezpečení provozu skladů. Typickou vlastností této komponenty je, že je neúčetní, tj. o dokladech, které v rámci ní vznikají, se běžně neúčtuje.

V této komponentě je možné evidovat všechny pohyby na skladech, tj. příjmy, výdeje, přesuny, včetně přesunů zboží mezi jednotlivými prodejními pracovišti. Komponenta podporuje členění skladů na skladové sektory (např. chladicí boxy) a skladové buňky a plánování (#Polozka Skladoveho Planu), jakým způsobem bude na jednotlivé sektory zaskladňováno zboží. Dále podporuje evidenci vztahů mezi příjmy a výdeji, jejichž prostřednictvím je možno řešit problematiku vratných obalů a dosledování původu vydávaného zboží (např. pro účely reklamací resp. šaržování).

V této komponentě vznikají jednotlivé prodejní dávky (#Prodejni Davka), a to vytríděním vyskladněného zboží do různých kvalitativních tříd a kategorií. Lze zde také sledovat rozpad jedné dávky na více jiných dávek, který vzniká prodejem části nebo novým tříděním původní dávky.



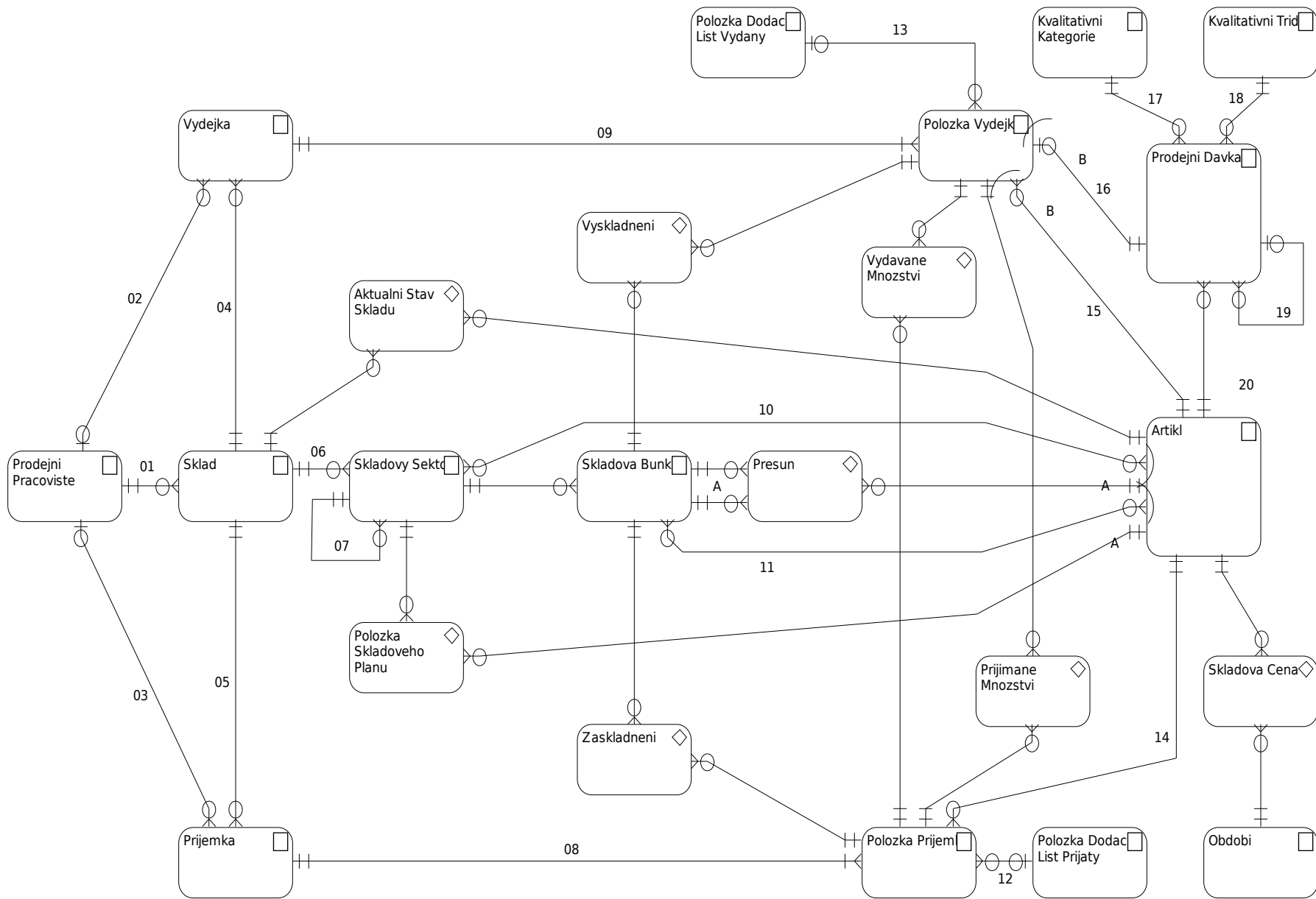
Objektem typu (#Artikl) je každý produkt nebo služba, která je nebo může být předmětem prodeje nebo nákupu v organizaci, včetně produktů a služeb konkurence (evidovaných např. pro potřeby marketingu) a produktů nebo služeb dosud neexistujících (např. pro potřeby plánování).

Objektem typu (#Sklad) je každé místo, na kterém jsou nebo můžou být fyzicky skladovány artikly a ke kterému se vztahuje hmotná odpovědnost.

Objektem typu (#Skladova Bunka) je každý nejmenší jednoznačně vymezený a identifikovatelný prostor ve skladu, ve kterém můžou být skladovány nějaké artikly.

Objektem typu (#Skladovy sektor) je každý jednoznačně vymezený skladový prostor, u kterého jsou specifikovány jeho zvláštní charakteristiky.

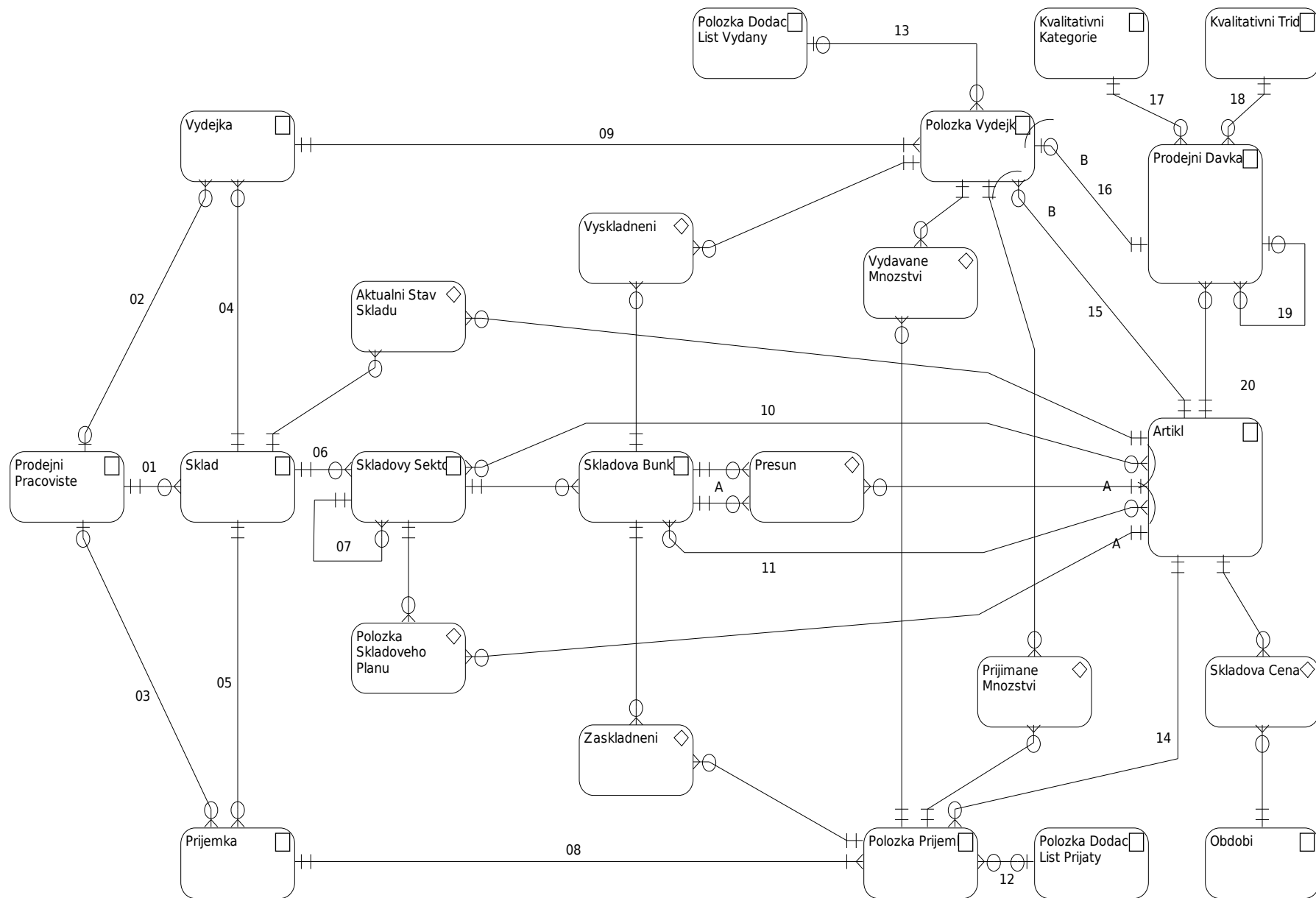
Pozn. Zvláštní charakteristiky jsou např. ochranná atmosféra, vlhkost, teplota či jiné klimatické podmínky, které mimo jiné určují jaké artikly lze v daném sektoru skladovat.



Objektem typu (#Presun) je každá reprezentace vazby mezi dvěma skladovými buňkami a artiklem se smyslem:
Přesunutě množství (Mnozstvi) daného artiklu (#Artikl) přesunutého na danou skladovou buňku (#Skladova Bunka) z dané skladové buňky (#Skladova Bunka) stejného skladu. / 0,1:0,M

Objektem typu (#Vyskladneni) je každá reprezentace vazby mezi položkou výdejky a skladovou buňkou se smyslem:
Vyskladněné množství (Mnozstvi), které bylo vyskladněno pro potřeby dané položky výdeje (#Polozka Vydejky) z dané skladové buňky (#Skladova Bunka). / 0,1:0,M

Objektem typu (#Zaskladneni) je každá reprezentace vazby mezi položkou příjemky a skladovou buňkou se smyslem:
Zaskladněné množství (Mnozstvi), které bylo zaskladněno z dané položky příjmu (#Polozka Prijemky) na danou skladovou buňku (#Skladova Bunka). / 0,1:0,M



Objektem typu (#Vydejka) je každý doklad dokumentující jeden výdej nějakých artiklů ze skladu.

Objektem typu (#Prijemka) je každý doklad dokumentující jeden příjem nějakých artiklů na sklad.

Objektem typu (#Polozka Vydejky) je každá položka (řádek) výdejky, která dokladuje výdej jedné dané prodejní dávky resp. daného množství jednoho daného artiklu.

Objektem typu (#Polozka Prijemky) je každá položka (řádek) příjemky jednoho daného artiklu, která specifikuje skutečné přijímané množství tohoto artiklu na sklad.

Objektem typu (#Prijimane Mnozstvi) je každá reprezentace vazby mezi položkou příjemky a položkou výdejky se smyslem:

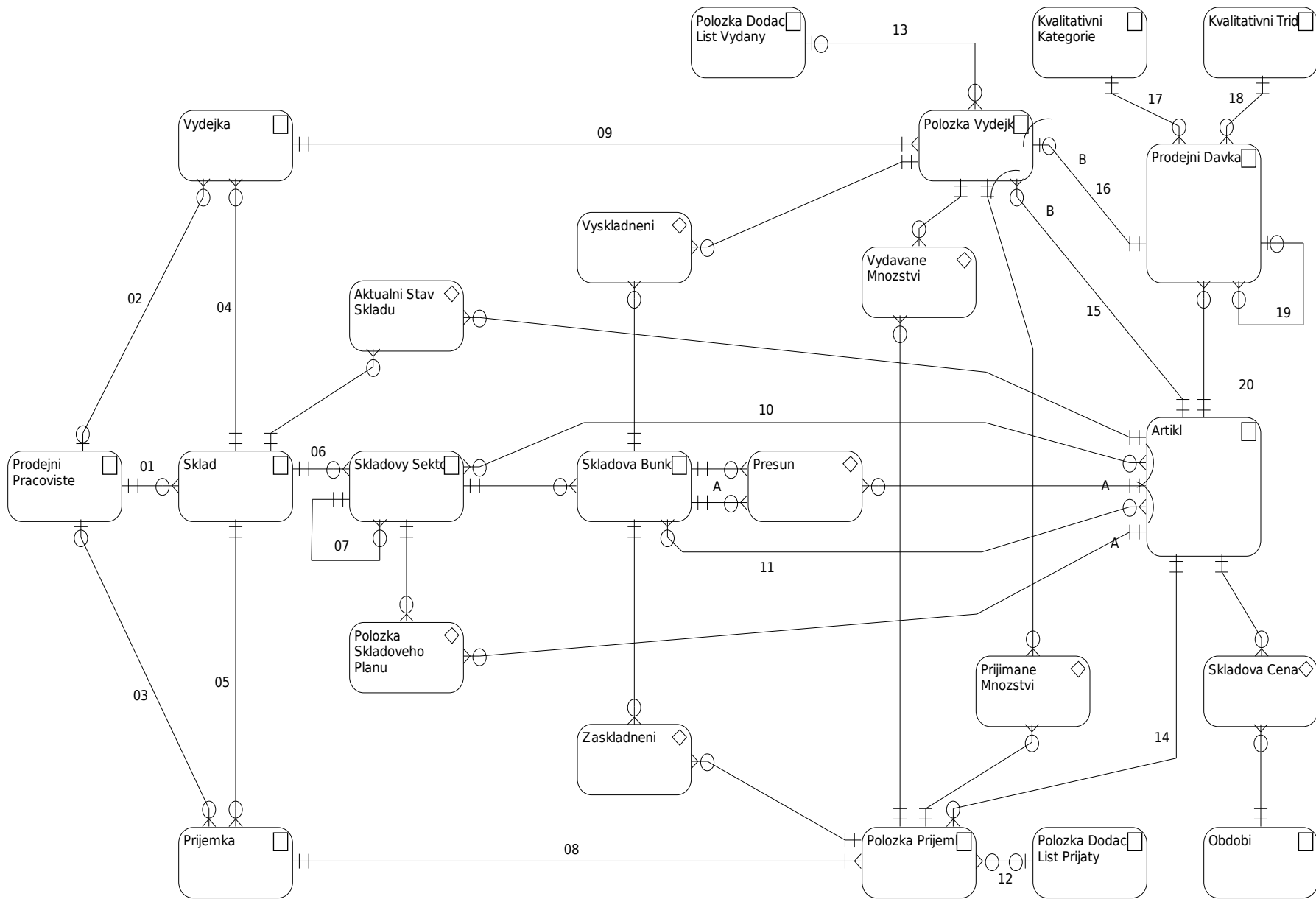
Přijímané množství (Mnozstvi), které bylo přijaté danou položkou příjmu (Polozka Prijemky) z dané položky výdeje (#Polozka Vydejky). / 0,1:0,M

Pozn. Používá se pro příjmy, které mají vztah k výdejům, např. příjem vydaných vratných obalů.

Objektem typu (#Vydavane Mnozstvi) je každá reprezentace vazby mezi položkou příjemky a položkou výdejky se smyslem:

Vydávané množství (Mnozstvi), které bylo vydané danou položkou výdeje (Polozka Vydejky) z dané položky příjmu (#Polozka Prijemky). / 0,1:0,M

Pozn. Používá se pro zpětné dosledování původu vydaného artiklu. Každý výdej musí být ve vztahu alespoň k jednomu příjmu.



Objektem typu (#Prodejni davka) je každá konkrétní dávka zboží určená k prodeji, u které jsou již určeny následující atributy:

- artikl
 - přesné množství
 - kvalitativní třída
 - kvalitativní kategorie
 - prodejní cena
 - datum vzniku dávky
 - pořadí dávky ve dni
 - prodejní pracoviště, na kterém vznikla
- a případně další atributy.

Vlivem manipulace s prodejní dávkou se může tato dávka případně rozpadnout na jiné, menší prodejní dávky.

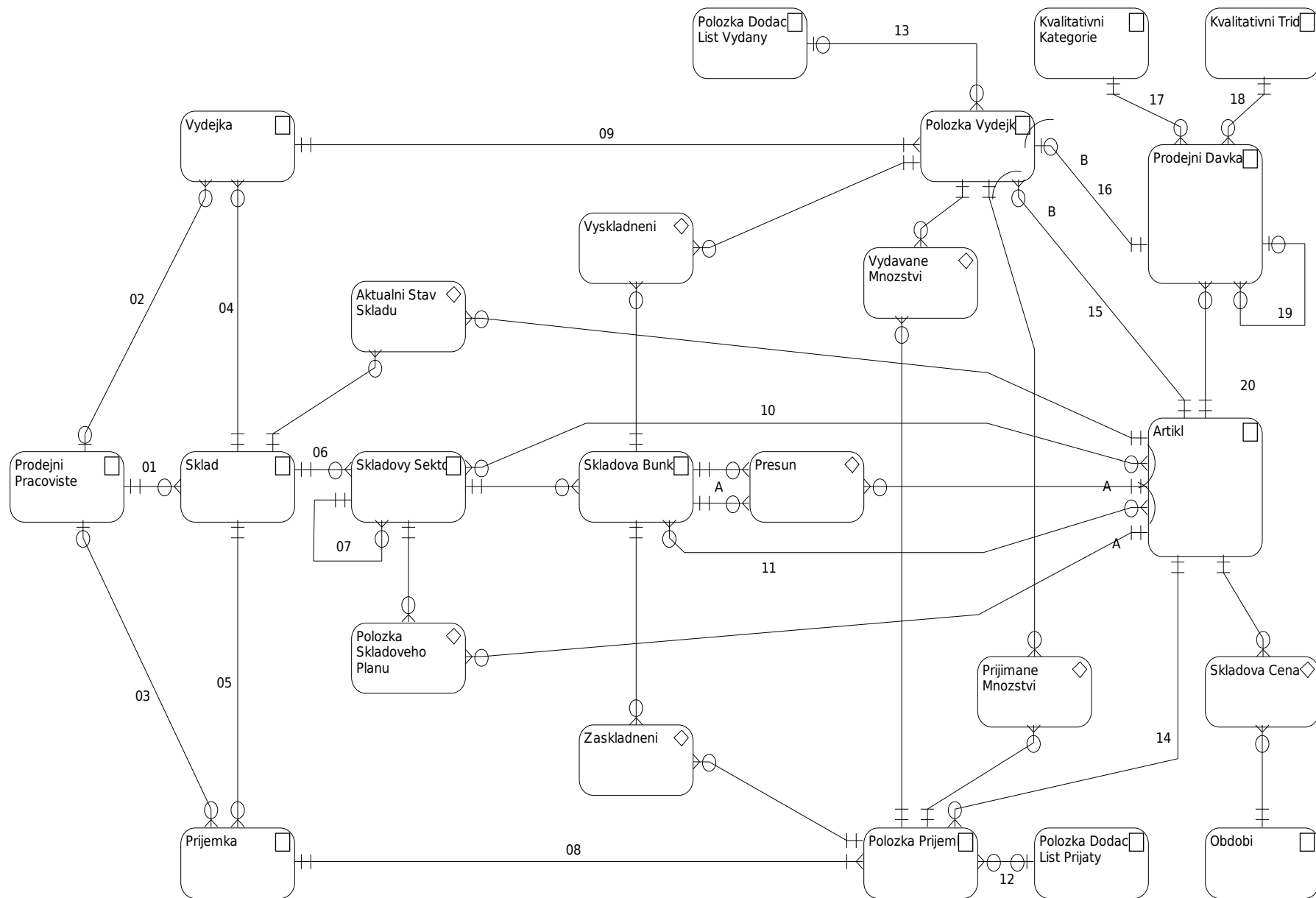
Pozn. Prodejní dávky vznikají vytríděním určitého množství skladovaného zboží.

Objektem typu (#Kvalitativni Kategorie) je každá kategorie určující úroveň kvality dané prodejní dávky artiklu. Kategorie udává úroveň kvality v jiném smyslu, než kvalitativní třída (entita #Kvalitativni Trida).

Pozn. Kvalitativní kategorie jsou např. podle velikosti zboží a pod.

Objektem typu (#Kvalitativni Trida) je každá třída resp. označení určující úroveň kvality dané prodejní dávky artiklu . Třída udává úroveň kvality v jiném smyslu, než kvalitativní kategorie (entita #Kvalitativni Kategorie).

Pozn. Kvalitativní třídy jsou např. I₁, II, výběr a pod.



- **Ref: 16**

Prodejní dávka (#Prodejni Davka), jejíž výdej ze skladu je dokladován danou položkou výdejky (#Polozka Vydejky). / 1,1:0,M

Pozn. Vazba se používá při výdeji vytríděného zboží ze skladu (tj. má určenu kvalitu), zpravidla pro prodej odběrateli.

- **Ref: 17**

Kvalitativní kategorie (#Kvalitativni Kategorie) dané prodejní dávky zboží (#Prodejni Davka). / 1,1:0,M

- **Ref: 18**

Kvalitativní třída (#Kvalitativni Trida) dané prodejní dávky zboží (#Prodejni Davka). / 1,1:0,M

- **Ref: 19**

Prodejní dávky (#Prodejni Davka), které vznikly z dané prodejní dávky (#Prodejni Davka). / 0,M:0,1

Pozn. Vazba zajišťuje rozpad prodejních dávek, který vzniká zejména roztríděním dané prodejní dávky do více jiných dávek (rozdílné kvality) nebo prodejem části určité prodejní dávky.

- **Ref: 20**

Prodejní dávky (#Prodejni Davka) daného prodejního artiklu (#Artikl). / 0,M:1,1

