

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

redukovaná hierarchie typů pro DM (zopakování)

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



Vazební a popisné atributy

- Necht' 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)

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

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

Necht' $P_i / (((\text{Bool}T_i)\text{Tim})\text{Wrd})$ 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, nezávislá na stavu světa.

Nadtypy, podtypy

- Necht' T, S jsou uzlové typy. Necht' T_w označuje populaci sorty T ve stavu světa w . Necht' 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:
- $\#Zamestnanec_w \subseteq \#Osoba_w$
 $\#Student_w \subset \#Osoba_w$
 $\#Zbozi_w \subseteq \#Produkt_w$
 $\#Vyrobek_w \subset \#Produkt_w$
 $\#Produkt_w \subset \#Artikl_w$
 $\#Sluzba_w \subseteq \#Artikl_w$
...
- 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

- Necht' 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) tutěž 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:
 $\exists A_i^R/(W \rightarrow (P_i \rightarrow Z)) (\forall x \in P_i ([A_i^R_w x] = [A_w x]))$
pro $i = 1, 2, \dots$
 $\exists A_i^E/(W \rightarrow (N_i \rightarrow Z)) (\forall x \in T ([A_i^E_w x] = [A_w x]))$
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

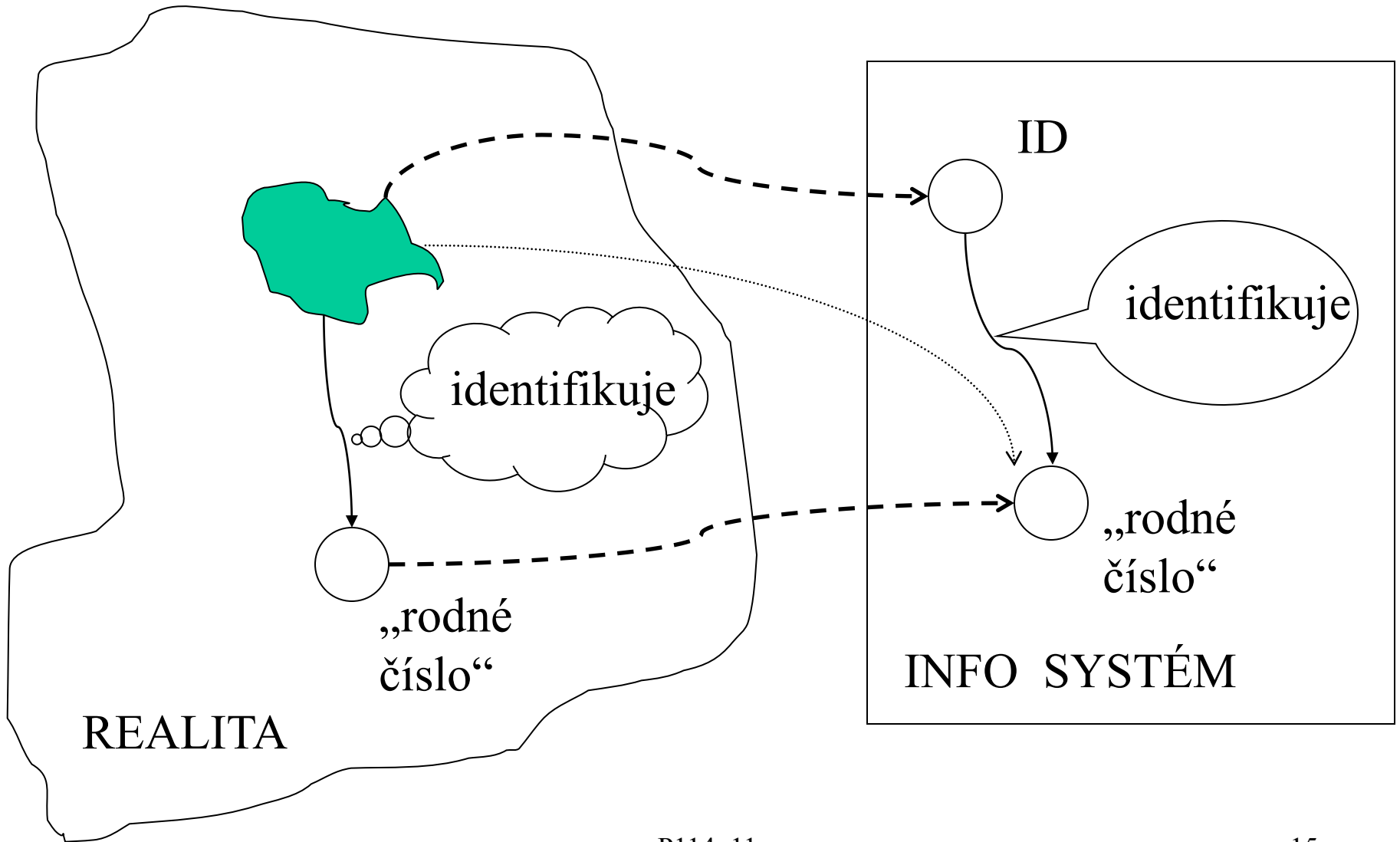
Identifikace

- 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 $M = (\mathbf{BZT}, \mathbf{K}, \mathbf{C})$
- Necht' $E \in \mathbf{BZT}$ 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 jednou 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

- Necht' 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

- Necht' E je E -typ, T_i je uzlový typ, $i=1,\dots,n$. Necht' 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

Necht' $A = \lambda w \lambda x \iota y (\Lambda_{(i=1..n)}([A_i_w x] = y_{(i)}))$.

Necht' 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

- Necht' E je nadtypem E_1, \dots, E_k . Necht'
 $\forall w (\forall ij (E_i \cap E_j = \{ \}))$ a
 $\cup_{(i=1..k)} E_i = E$, (\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

- Necht' 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.

Připustné 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 připustná, jestliže $\mathbf{B} \approx \mathbf{A}$. Také říkáme, že když transformace je připustná, 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 připustné transformace konceptuálního modelu

Věta 3 (o intenzionálních relacích)

- Necht' A je libovolný HIT-atribut,
 $w::W$, $x_i::T_i$, $y::T$, T_i , T příslušné základní typy, jak vyžaduje definice,
 $A = \lambda w \lambda x_1 \dots x_n \square y ([A_w(x_1, \dots, x_n)] * y)$.
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)

- Necht' A je libovolný nerozložitelný HIT-atribut složitosti větší než 2, $w::W$, $x_i::T_i$, T_i , příslušné základní typy, jak vyžaduje definice,

$$A = \lambda w \lambda x_1 \dots x_{n-1} \lambda x_n ([A_w(x_1, \dots, x_{n-1})] * x_n).$$

- 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 $[A w]$

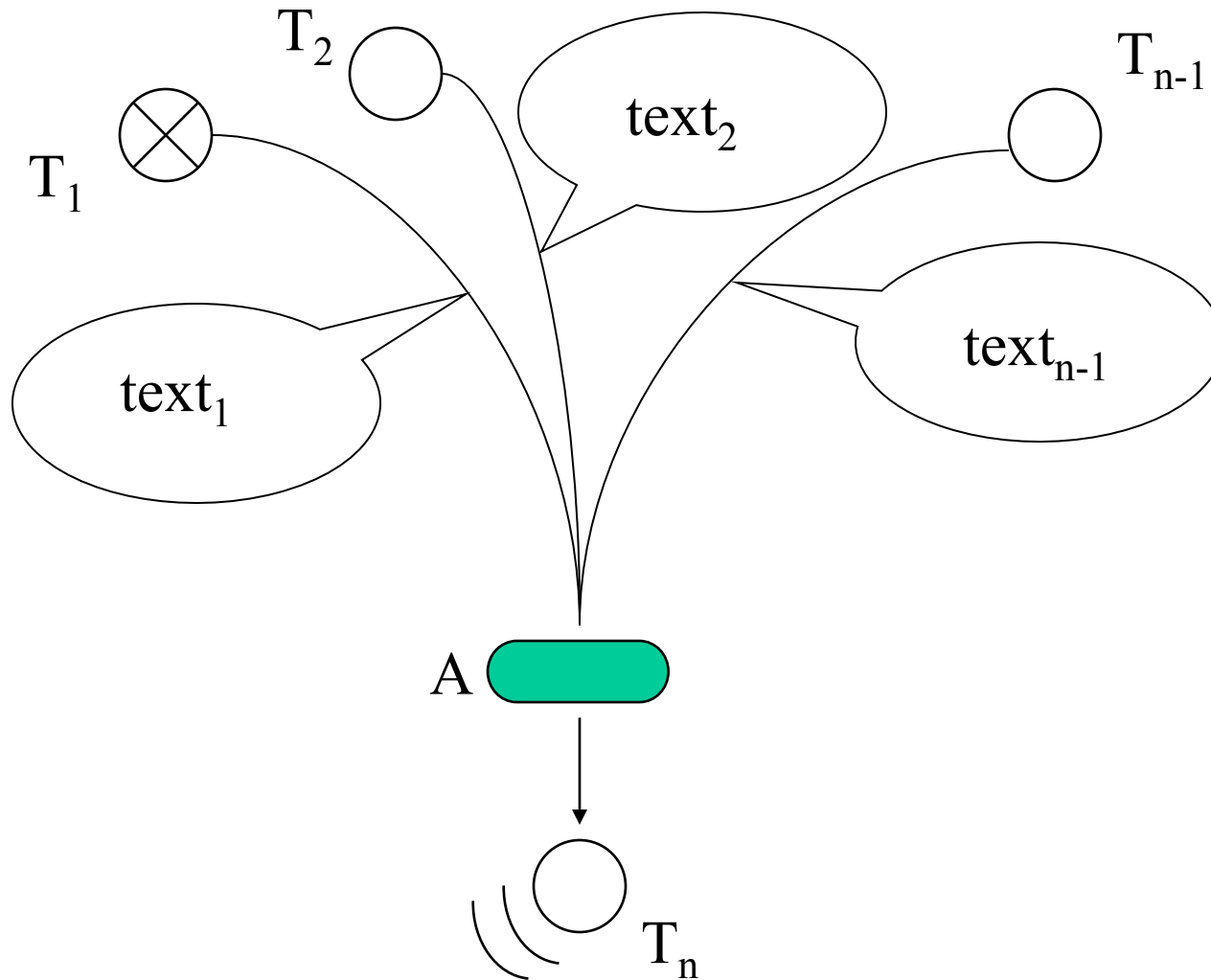
Binarizační princip (2)

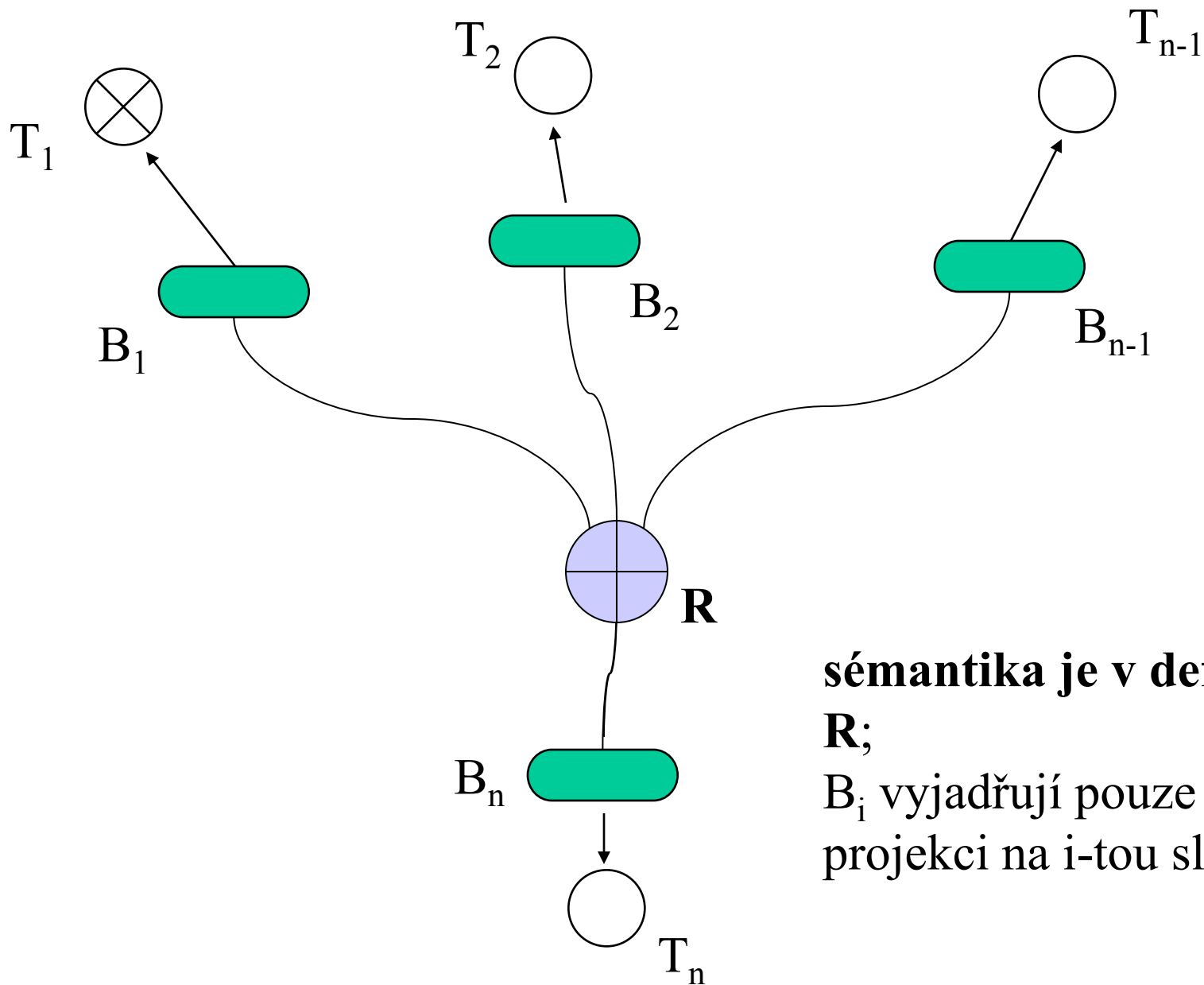
- Definujeme pro $i = 1..n$ projekce $B_i / (W \rightarrow (R \rightarrow T_i))$ takto:

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

- 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.