

Obsah

1	ÚVOD	3
2	RELAČNÝ MODEL	5
2.1	ZÁKLADNÉ POJMY	5
2.1.1	Statický pohľad	5
2.1.2	Dynamický pohľad – Aktualizačné operácie	9
2.2	INTEGRITNÉ OHRANIČENIA	10
2.3	NEVÝHODY RELAČNÉHO MODELU	10
2.3.1	Vertikálna homogenita	10
2.3.2	Horizontálna homogenita	10
2.3.3	Mená polí sú iba symboly	11
2.3.4	„Čierno–biele“ videnie sveta	11
2.3.5	Nepopísateľnosť ‘relácií’	11
2.3.6	Rôznorodosť významu informácie (ADT)	11
2.4	INTERPRETÁCIA DOTAZOV	12
2.5	REPREZENTAČNÝ SYSTÉM	13
2.6	RELAČNÝ MODEL A LOGIKA	14
2.7	PREDPOKLAD UZAVRETÉHO VS. OTVORENÉHO SVETA	16
2.8	TERMINOLOGICKÉ POZNÁMKY	17
3	NULLOVÉ HODNOTY	19
3.1	SÉMANTIKA NULLOV	19
3.2	JE „INAPPLICABLE“ NULL OPODSTATNENÝ?	23
3.3	ATOMICKÝ NULL – CODDOVE TABULKY	23
3.3.1	Trojhodnotová logika	24
3.3.2	Null-substitučný princíp	25
3.3.3	Relačná algebra	27
3.3.4	Problémy Coddových tabuliek	31
3.4	INDEXOVANÉ (ZNAČKOVANÉ) NULLY	34
3.4.1	V–tabulky	34
3.4.2	C–tabulky	35
3.4.3	Reiterov prístup	38
3.5	BISKUPOVE NULLY	41
3.6	„NO INFORMATION“ NULLY	44
3.7	UPDATE OPERÁCIE	53
4	ČIASTOČNÁ INFORMÁCIA	55
4.1	PODMNOŽINOVÁ KONŠTRUKCIA	55
4.2	ČÍSELNÉ DOMÉNY: INTERVALOVÉ HODNOTY	55
4.2.1	Grantové intervalové hodnoty	56
4.2.2	Množinové intervalové hodnoty	59
4.3	REŤAZCOVÉ DOMÉNY: REGULÁRNE VÝRAZY	60
4.4	DISJUNKTÍVNA INFORMÁCIA	62
4.4.1	I–tabulky	62
4.4.2	M–tabulky	63
4.4.3	MPV–tabulky	64
4.4.4	Pv–tabulky	65
4.4.5	Príklad: interpretácia dotazov	67
5	LOGICKÉ ASPEKTY NEÚPLNEJ INFORMÁCIE	71
5.1	POHĽADY (VIEWS)	71
5.2	KONCEPTUÁLNE MODELOVANIE	73
5.3	NÁVRH DATABÁZ S ČIASTOČNOU INFORMÁCIOU	73
5.3.1	Ako reprezentovať čiastočné hodnoty	74

5.3.2	<i>Ako zaobchádzať s neúplnou informáciou</i>	74
5.3.3	<i>Ako sa postaviť k heterogénnemu prostrediu</i>	75
5.4	NEÚPLNÁ INFORMÁCIA A FUNKČNÉ ZÁVISLOSTI	75
5.4.1	<i>Coddove tabuľky</i>	75
5.4.2	<i>V–tabuľky</i>	76
5.4.3	<i>Funkčné závislosti vyjadrené reprezentáciou</i>	77
5.4.4	<i>Neúplná informácia a normálne formy</i>	78
5.5	DEKOMPOZÍCIA TABULIEK ODSTRANJÚCA NULLY	79
5.5.1	<i>Jeden atribút s nullmi</i>	79
5.5.2	<i>Viacero atribútov s nullmi</i>	85
6	FYZICKÉ ASPEKTY NEÚPLNEJ INFORMÁCIE	94
7	DODATOK	95
7.1	PREVOD VŠEOBECNÝCH RELÁCIÍ NA KLASICKÉ RELÁCIE	95
7.2	ZNAČENIE	95
7.3	CHARAKTERISTIKA NAVRHNUTEJ DEKOMPOZÍCIE	96
7.4	NULLOVÉ HODNOTY V PRAXI	98
7.4.1	<i>Null v štandarde SQL92</i>	98
7.4.2	<i>Null v SQL Oracle7 Server</i>	100
7.4.3	<i>Null v Microsoft Access 97</i>	100
8	ZÁVER	102
9	LITERATÚRA A POUŽITÉ PRAMENE	103

1 Úvod

„Ten, kto vie, nehovorí.
Kto hovorí, ten nevie.“
Lao-c’¹

Relačný model od svojho vzniku takmer pred 30 rokmi zaznamenal masívny vzrast najmä v komerčnej oblasti. Hoci v poslednom čase sa znalostné systémy a objektovo-orientované (či objektovo-relačné) databázové systémy stávajú čoraz viac populárne, a to aj vďaka kritike relačného modelu, je nepopierateľné, že relačné databázy sú najvhodnejšie na prácu s veľkým množstvom neštruktúrovaných dát v mnohých oblastiach (ekonomické softvéry, ako sú jednoduché a podvojné účtovníctvo, evidencia majetku, mzdy, personalistika a mnoho iných). Preto relačnému modelu treba stále venovať pozornosť.

Jedným z chýlostivých problémov databázovej teórie (a relačného modelu zvlášť) je už desaťročia zaobchádzanie s neúplnou informáciou. Hoci bolo navrhnutých viacero prístupov, z ktorých mnohé sú popísané v tejto práci, definitívne riešenie tohoto problému zatiaľ neexistuje. Táto práca sa zaoberá aspektmi teórie neúplnej informácie v relačných databázach; nepojednáva o iných dátových modeloch (hniezdený, objektový). Ak sa aj zmieňujeme o iných modeloch, vždy len okrajovo, na porovnanie s relačným modelom. V jednej časti sa stručne venujeme nevýhodám relačného modelu postaveného na záznamoch. Okrajovo je pozornosť venovaná aj niektorým databázovým prostrediam, rozšíreným a používaným.

V tejto práci sa venujeme nasledovným aspektom neúplnej informácie:

- *logickému aspektu* (kapitoly 3, 4, 5), ktorý zahŕňa
 - sformulovanie všeobecných modelov pre neúplnú informáciu;
 - vyhodnocovanie dotazov a zovšeobecnenie relačnej algebry s neúplnou informáciou (relačné operátory a výrazy zachovávajúce požadovanú sémantiku neúplnej informácie);
 - funkčné závislosti a návrh databázovej schémy s nullovými hodnotami resp. neúplnou hodnotou (časti 5.4 a 5.3);
- *fyzickému aspektu*, ktorý zahŕňa porovnanie časovej a priestorovej náročnosti modelov s neúplnou informáciou a bez nej (kapitola 6).

Neúplná informácia zahŕňa dve kategórie:

- *nepresná informácia*², ktorá zahŕňa nekompletnosť hodnoty atribútu,
- *neistá informácia*³, ktorá vypovedá o stupni pravdivosti hodnoty atribútu.

My sa budeme venovať prvej kategórii neúplnej informácie, ktorú samotnú budeme nazývať *neúplná informácia*⁴.

¹ Lao-c’: *Tao-Te-Ťing*, 56 (vyd. CAD PRESS, Bratislava 1994)

² angl. *imprecise information*

³ angl. *uncertain information*

⁴ angl. *incomplete information*

V rámci modelov pre nepresnú informáciu sa študovali a študujú nasledovné modely:

- *nullové hodnoty* (kapitola 3),
- *čiasťové hodnoty* a spomedzi nich špeciálne *disjunktívna informácia* (kapitola 4),
- *aproximatívna informácia*,
- *maybe informácia*.

V prvých dvoch prípadoch predpokladáme, že informácia uchovávaná v databáze, akokoľvek čiastočná či neúplná, je vždy správna, pravdivá, istá (angl. *sure*). Neuvažujeme teda prípady, že v databáze je uložená informácia nepravdivá. Použitie nullov dáva v podstate dva extrémne prípady pre hodnotu atribútu:

- (a) Všetko je známe o hodnote atribútu (úplná informácia).
- (b) Nič nie je známe o hodnote atribútu.

Disjunktívna informácia ide poza toto „čierno-biele“ videnie sveta: v databáze zahŕňame viacero možností pre realitu.

Oblasť aproximatívnej informácie skúma, aký vplyv má obohatenie množín hodnôt o vnútornú štruktúru (definovanú algebraicky) na čiastočnú informáciu.⁵

V modeli pre *maybe* informáciu uchováваме aj informáciu, ktorá prípadne (po upresnení) nemusí byť pravdivá (odtiaľ pochádza jej názov: angl. *maybe*). Tomuto druhu neúplnej informácie sa budeme v práci venovať okrajovo, a to z toho dôvodu, že hoci nejde o neúplnú informáciu v pravom zmysle slova, je prinajmenšom súčasťou výsledkov dotazov (*maybe* informáciou je aj totálna tupľa s príznakom, pričom môže a nemusí byť zahrnutá vo výslednej interpretácii – *možno je zahrnutá, možno nie*).

Čo sa týka modelov pre neistú informáciu, študujú sa tieto dva modely:

- *fuzzy informácia* (tzv. *possibility approach*),
- *pravdepodobnostný prístup* (*probability approach*).

Oblasť *fuzzy databáz* zahŕňa o hodnote atribútu fuzzy výpoveď (napr. pre atribút vek: „okolo štyridsiatky“, „takmer 23“, „mladý“).⁶ Pravdepodobnostné databázy sú také databázy, v ktorých pri informácii je uvedená pravdepodobnosť, s akou sa táto informácia v interpretácii (realite) môže vyskytnúť.⁷

Súčasťou práce je aj návrh jednoduchšej dekompozície relácií s cieľom vysporiadať sa s troma základnými typmi nullov (časť 5.5).

Práca bola po jazykovej a formálnej stránke vytvorená sledujúc štandardy a odporúčania z diel [Katuščák, 1998] a [Eco, 1997].

⁵ Porov. napr. [Libkin, 1994].

⁶ Porov. napr. [Takahashi, 1993].

⁷ Porov. napr. [Barbará et al., 1992].

2 Relačný model

2.1 Základné pojmy

Používame koncept relačného modelu, ako je popísaný v štandardných učebniciach databázových systémov, napr. [Ullman, 1988], [Date, 1995]. V konkrétnych prístupoch, ktoré relačný model rozširujú, sú jednotlivé odlišnosti definované priamo pri podrobnom popise prístupu. Hovoríme o *statickom* resp. *dynamickom* pohľade na databázu.

- *Statický pohľad* charakterizuje jednotlivé „stavy“ databázy, teda aj relácií. Patria k nemu syntax a sémantika relácií, tuplí a tiež relačná algebra/kalkul resp. dotazovací jazyk ako nástroj na manipuláciu s jednotlivými časťami databázy a/alebo získavanie informácií z nej.
- *Dynamický pohľad* charakterizuje „prechody medzi stavmi“, teda zmeny, ktoré nastávajú v jednotlivých reláciách i databáze ako celku. Jedná sa o formálny popis *aktualizačných operácií* vyjadrených pomocou relačnej algebry/kalkulu, ktoré slúžia na zmenu (aktualizáciu) údajov databázy.⁸

2.1.1 Statický pohľad

2.1.1.1 Relácie

Základom relačného modelu je pojem *relácia*. Je to ľubovoľná podmnožina kartézského súčinu jednej alebo viacerých množín. Často tieto množiny nazývame *domény*, hoci nemáme na mysli, ak to nie je výslovne uvedené, že majú oproti štandardne matematicky definovanej množine špeciálne vlastnosti. Relácie označujeme veľkými písmenami, napr. R , S , domény samotné zväčša neznačíme (iba vzhľadom k atribútu).

Jednotlivý prvok relácie sa nazýva *tupľa*. Špeciálne, ak máme reláciu $R \subseteq D_1 \times \dots \times D_n$, nazýva sa tupľa *n-tica*. Tuple označujeme gréckymi písmenami, napr. μ , ν . Tupľa je z definície usporiadanou postupnosťou (*n-ticou*).⁹

Jednotlivé komponenty kartézského súčinu sa nazývajú *atribúty* a často sa aj špeciálne označujú, pretože dva rôzne atribúty môžu nadobúdať hodnoty z rovnakej domény. Doména priradená atribútu A sa označuje $D(A)$.

V tabuľkovej reprezentácii relácie (teda keď každej relácii zodpovedá *tabuľka*, table) sa atribúty nazývajú *stĺpce* (columns) a jednotlivé tuple sa nazývajú *riadky* (rows). Zástancom stotožnenia relácií s tabuľkami je databázista C. J. Date (pozri napr. [Date, 1995]). Tabuľka 1 prehľadne zhŕňa tieto ekvivalentné pojmy.

⁸ Pre úplnosť poznamenajme, že na aktualizácie operácie je možné pozeráť sa tiež z hľadiska *dokazovateľnej teoretickej*, keď sa jednotlivá update operácia charakterizuje podmienkami, ktoré má výsledná databáza spĺňať. Jedná sa teda o nekonštruktívny pohľad na aktualizácie operácie. Z hľadiska teórie programovania ide o *axiomatickú sémantiku*. V tejto práci dáme prednosť *operačnej* resp. *denotačnej sémantike* aktualizčných operácií, porov. [Manna, 1981].

⁹ Od konkrétneho modelu závisí, či tupľa je *množina* alebo *n-tica*, teda či nezáleží alebo záleží na poradí atribútov (pokiaľ spolu s hodnotou je uchovávaný aj atribút). My budeme uvažovať o usporiadaných tupliach.

Tabuľka 1: Dvojaký spôsob názvov v relačnom modeli

reprezentácia			
množinová	relácia	tupľa (n -tica)	atribút
tabuľková	tabuľka	riadok	(pomenovaný) stĺpec

Keďže relácia bola definovaná ako podmnožina kartézskeho súčinu, jedna relácia nesmie obsahovať dve rovnaké tuple, slovami tabuľkovej reprezentácie, v konkrétnej tabuľke sa nesmú vyskytovať dva rovnaké riadky. Táto zákonitosť sa nazýva *neduplikačné pravidlo* (*nonduplication rule*).

Relácia, ktorá je uchovávaná v databáze vo forme tabuľky ako „nezávislá“ časť databázy a navyše obsahuje nejaký primárny kľúč, sa nazýva *bázová relácia*. *Odvozená (nebázová) relácia* je relácia, ktorej všetky hodnoty atribútov závisia od (a/alebo sú napríklad počítané z) iných relácií; sú to aj relácie, ktoré boli dočasne vytvorené nejakým dotazom.

Nech W je množina atribútov, nad ktorou je definovaná tupľa μ a $M \subseteq W$.¹⁰ Hodnota atribútu A tuple μ sa označuje $\mu[A]$. *Regulárna hodnota* atribútu A je každá hodnota, ktorá patrí do domény pre daný atribút, avšak nie je nullová/neúplná.¹¹ Hovoríme, že tupľa v je *reštrikciou* tuple μ na množine M , ak $\forall A \in M: \mu[A] = v[A]$. Tupľu v zapisujeme $\mu[M]$ a hovoríme tiež, že v je *M -komponent* tuple μ (tiež zápis $\pi_M(\mu)$). Hovoríme, že tupľa μ je *M -totálna*, ak v každom atribúte množiny M obsahuje regulárnu hodnotu. Hovoríme, že μ je *totálna* alebo že obsahuje *úplnú informáciu*¹², ak je W -totálna, teda keď v žiadnom atribúte nemá neúplnú/čiastočnú hodnotu. Hovoríme, že tupľa je *nullová*, ak v každom atribúte je jej hodnotou null.

Dve tuple sú *kompatibilné*, ak sú definované nad rovnakou množinou atribútov. Dve tuple sú *nekompatibilné*, ak nie sú kompatibilné.

Nech μ je tupľa definovaná nad $M = \{M_1, \dots, M_m\}$, v nad $N = \{N_1, \dots, N_n\}$. *Spojenie tuplí* μ a v je tupľa τ definovaná nad množinou atribútov $M \cup N$, pre ktorú platí

$$\tau[M] = \mu, \tau[N] = v,$$

a označujeme ju $\mu \bowtie v$. V tom prípade hovoríme, že tuple μ a v *mačujú*¹³ na množine atribútov $M \cap N$. *Zreťazenie tuplí* μ a v je tupľa τ definovaná nad množinou atribútov $\langle M_1, \dots, M_m, N_1, \dots, N_n \rangle$, ktorej prvých m komponentov tvorí tupľu μ a ostatných n komponentov tvorí tupľu v , inými slovami, taktiež platí $\tau[M] = \mu, \tau[N] = v$. Zreťazenie označujeme $\mu \times v$.¹⁴

¹⁰ Tupľa μ je definovaná nad množinou atribútov W , ak $\mu \in R$ a W je množina atribútov relácie R .

¹¹ Adjektívum „regulárny“ používame s výnimkou časti 4.3 výlučne v uvedenom zmysle, ako nie-neúplný.

¹² angl. tiež *definite tuple*

¹³ angl. *match*

¹⁴ Hoci požadujeme od výslednej tuple τ v oboch prípadoch rovnakú vlastnosť, sú definované nad potenciálne rozličnými množinami atribútov. Nech $\mu = \langle a, b \rangle$ je definovaná nad $\langle A, B \rangle$ a $v = \langle b, c \rangle$ nad $\langle B, C \rangle$. Potom

$$\mu \bowtie v = \langle a, b, c \rangle \text{ nad } \langle A, B, C \rangle,$$

$$\mu \times v = \langle a, b, b, c \rangle \text{ nad } \langle A, B, B, C \rangle.$$

2.1.1.2 Relačná algebra

Používame štandardne definovanú relačnú algebru, ako ju uvádzajú napr. [Ullman, 1988] a [Date, 1995]. Operácie zjednotenia, prieniku, rozdielu, podielu, kartézského súčinu, projekcie, selekcie, F -joinu a prirodzeného joinu označujeme \cup , \cap , \times , \setminus , \div , π , σ , \bowtie_F , \bowtie . Stručne zhrnieme význam relačných operácií:

- *zjednotenie* (U), *prienik* (I), *rozdiel* (*diferencia*, D): množinovo-teoretický význam, t.j.
 $R \cup S =_{df} \{\mu \mid \mu \in R \vee \mu \in S\}$,
 $R \cap S =_{df} \{\mu \mid \mu \in R \wedge \mu \in S\}$,
 $R \setminus S =_{df} \{\mu \mid \mu \in R \wedge \neg(\mu \in S)\}$,
- *projekcia* (P): $\pi_X(R) =_{df} \{\mu[X] \mid \mu \in R\}$,
- *selekcia* (S): $\sigma_F(R) =_{df} \{\mu \in R \mid F(\mu) = true\}$, pričom formula F obsahuje konjunkciu, disjunkciu, test rovnosti atribútu s konštantou a/alebo iným atribútom,
- *pozitívna selekcia* (S^+): selekcia v prípade, že F je pozitívna formula (neobsahuje negáciu),
- *kartézsky súčin* (X): $R \times S =_{df} \{\mu \times \nu \mid \mu \in R \wedge \nu \in S\}$,
- *(F -)join*: $R \bowtie_F S =_{df} \sigma_F(R \times S)$,
- *prirodzený join* (J): $R \bowtie S =_{df} \{\mu \bowtie \nu \mid \mu \in R \wedge \nu \in S\}$,
- *podiel*: nech R je definovaná nad množinou atribútov $M \cup N$ a $S \neq \emptyset$ je definovaná nad N . Podiel je potom definovaný nasledovne: $R \div S =_{df} \pi_M(R) \setminus \pi_M((\pi_M(R) \times S) \setminus R)$.

2.1.1.3 Funkčné závislosti

Nech X, Y sú množiny atribútov (t.j. $X, Y \subseteq W$). Hovoríme, že X *určuje* Y (označenie $X \rightarrow Y$) v relácii R , ak pre ľubovoľné tuple z R zhodujúce sa v hodnotách atribútov množiny X platí, že sa zhodujú aj vo všetkých atribútoch množiny Y , formálne

$$\forall \mu, \nu \in R: \pi_X(\mu) = \pi_X(\nu) \Rightarrow \pi_Y(\mu) = \pi_Y(\nu),$$

a ďalej hovoríme, že v takom prípade nad reláciou R *platí funkčná závislosť*¹⁵ $X \rightarrow Y$. Môžeme tiež napísať, že platnosť funkčnej závislosti $X \rightarrow Y$ v relácii R je daná nasledovne:

$$\text{platnosť}(X \rightarrow Y, R) =_{df} \begin{cases} true, & \text{ak } \forall \mu, \nu \in R: \mu[X] = \nu[X] \Rightarrow \mu[Y] = \nu[Y], \\ false & \text{inak.} \end{cases}$$

Nech F je množina funkčných závislostí. Platnosť tejto množiny nad reláciou R je daná paralelným aplikovaním jednotlivých platností, teda

$$\text{Platnosť}(F, R) =_{df} \bigwedge \{\text{platnosť}(X \rightarrow Y, R) \mid X \rightarrow Y \in F\}.$$

F stále značí množinu funkčných závislostí. Nasledovné pravidlá, ktoré slúžia na odvodenie ďalších funkčných závislostí v F , *Armstrongove axiómy*, sú zdravé a úplné.

1. (*reflexivita*) ak $Y \subseteq X$, tak $X \rightarrow Y \in F$ (nazýva sa *triviálna závislosť*),
2. (*rozšírenie*) ak $X \rightarrow Y \in F$ a $Z \subseteq W$, tak $XZ \rightarrow YZ \in F$,
3. (*tranzitivita*) ak $X \rightarrow Y \in F$ a $Y \rightarrow Z \in F$, tak $X \rightarrow Z \in F$.

¹⁵ angl. *functional dependency*, preto niekedy používame ustálenú skratku FD

Množina atribútov $M \subseteq W$ relácie R (nad W) sa nazýva *nadkľúč*, ak $M \rightarrow W$, inými slovami, keď žiadne dve tuple relácie R nemajú rovnaký M -komponent. Množina atribútov M sa nazýva *kľúč* (candidate key alebo len key), ak je nadkľúč, ale žiadna jej vlastná podmnožina nie je nadkľúč. Vlastnosť množiny M „byť kľúčom“ znamená:

- (1) žiadne dva riadky nemajú rovnaký M -komponent,
- (2) ak vyjmeme z množiny M ľubovoľný atribút, táto nová množina stratí vlastnosť z predošlého bodu.

V danej relácii môže existovať viac množín atribútov, ktoré majú vlastnosť kľúča. Avšak z praktických dôvodov¹⁶ sa jeden z kľúčov vyberie a nazýva sa *primárny kľúč* pre danú reláciu. Keď kľúč obsahuje viacero atribútov, nazýva sa *multiatribútový* (alebo tiež *zložený*, angl. *compound*).¹⁷

2.1.1.4 Dotazovací jazyk

Ako sme spomínali vyššie, k statickému pohľadu na databázu patrí aj *dotazovací jazyk*. Väčšina autorov sa nezaobera presnou syntaktickou štruktúrou dotazovacieho jazyka, pretože používajú matematický aparát zapisovania dotazov. Matematický (množinovo-teoretický) spôsob zápisu dotazov budeme používať aj v tejto práci. Zapisujeme dotazy taktiež v slovnej podobe, pričom na danom mieste vždy vysvetlíme, čo sa týmto slovným vyjadrením myslí. Pod dotazmi máme na mysli ako dotazy *zistovacie* (teda či daný prvok patrí do množiny, či daná množina spĺňa nejaký predikát) – odpovedá sa na ne odpoveďou *áno* alebo *nie*, tak aj dotazy *doplňovacie* (aké prvky spĺňajú nejakú podmienku) – odpoveďou je množina elementov.

Elementárny dotaz je dotaz, ktorý je možné zodpovedať jednoduchým nahliadnutím do tabuľky (v podstate otázka tvaru „je hodnota atribútu A tuple/tuplí a ?“, pre číselné domény nielen rovnosť, ale základné predikáty porovnania). *Konjunktívny (disjunktívny) dotaz* je taký dotaz, ktorý je konjunkciou (disjunkciou) elementárnych dotazov.

Pre úplnosť stručne popíšeme syntax dotazovacieho jazyka, ako ho navrhol Lipski, pričom vychádzame z jeho prác [Lipski, 1977], [Lipski, 1979] a [Lipski, 1981]. Lipskeho dotazovací jazyk sa skladá z termov a formúl. *Termy* popisujú podmnožiny množiny objektov (doplňovacie dotazy), zatiaľ čo *formuly* (zistovacie dotazy) vyjadrujú fakty, ktoré platia o databázovom systéme ako celku.

Termy sa skladajú z:

- (1) *deskriptorov*, čiže dvojíc $\langle i, A \rangle$, kde $i \in I$ (I je množina atribútov) a A je (obyčajne nie ľubovoľná) podmnožina D_i (nosič pre atribút i);
- (2) symbolov logických operácií (*false*: 0, *true*: 1, *not*: $-$, *or*: $+$, *and*: \cdot , *impl*: \rightarrow);
- (3) zátvoriek.¹⁸

¹⁶ keďže mnohé štruktúry fyzického uloženia dát vyžadujú jednoznačný a jediný kľúč (potreby implementácie)

¹⁷ Tie domény, nad ktorými sú v danej databáze definované jednoatribútové primárne kľúče, sa nazývajú *primárne domény*.

¹⁸ Formálnejšie: je to najmenšia množina generovaná uvedenými troma konštruktormi (prípadne sklasickou štvrtou klauzulou „nič iné je v množine termov“).

Formuly sa skladajú z:

- (1) logických konštánt *true*, *false* a všetkých atomických formúl (tvaru $s = t$, kde s, t sú ľubovoľné termy);
- (2) ak ϕ a ψ sú formuly, potom aj $\neg\phi$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \Rightarrow \psi)$ sú formuly.¹⁹

Ak formula neobsahuje negáciu, nazýva sa *pozitívna*. Ak je formula tvorená iba konjunkciou (disjunkciou) atomických formúl, nazýva sa *konjunktívna* (*disjunktívna*).

Príklad 1: Termy

Nech $i \in I$ znamená dĺžku a nech D_i je množina nezáporných reálnych čísel. Na reprezentáciu termu $\langle i, A \rangle + \langle i, B \rangle$, kde $A = \{x \mid x \leq 100\}$, $B = \{x \mid x \leq 50 \vee x < 200\}$, môžeme v nejakej rozumnej implementácii dotazovacieho jazyka zapísať výrazom

$$\langle \text{Dĺžka} \leq 100 \rangle + \langle \text{Dĺžka IN } [0, 50] + (200, \infty) \rangle,$$

pričom sme použili štandardné matematické označenie $[a, b]$ pre (obojsstranne) uzatvorený a (a, b) pre (z obidvoch strán) otvorený interval na množine reálnych čísel.

2.1.2 Dynamický pohľad – Aktualizačné operácie

Základné aktualizácie (update) operácie definujeme podľa [Abiteboul–Grahne, 1985].

Nech T je tabuľka a μ je tupľa, ktorá má rovnakú signatúru ako relácia zodpovedajúca tabuľke T a F je výraz (ktorý môže spĺňať množina tuplí). Aktualizačné operácie môžeme vyjadriť množinovo–teoreticky v relačnej algebre (výsledná tabuľka je označená T'):

- *insertion* (vlozenie) tuple μ do tabuľky T :

$$T' = T \cup \mu,$$
- *deletion* (vymazanie) tuple μ resp. tuplí, spĺňajúcich F z tabuľky T :

$$T' = T \setminus \mu \text{ resp. } T' = T \setminus \{v \mid F(v)\},$$
- *modifikácia* tuple/tuplí nachádzajúcej sa v tabuľke:²⁰

(a) nahradenie tuple $\mu \in T$ tupľou μ' :

$$T' = (T \setminus \mu) \cup \mu',$$

(b) nahradenie tuplí spĺňajúcich F tak, aby spĺňali F' , pričom F' je výraz neobsahujúci nerovnosti (ide vlastne o formálne zapísanú substitúciu)²¹:

$$T' = (T \setminus \{v \mid F(v)\}) \cup \{F'(v) \mid F(v)\}.$$

Autori tiež definujú ďalšie update operácie, súvisiace najmä s neúplnou/čiastočnou informáciou:

- *integration*: zosúladenie dvoch databáz (priemik ich reprezentácií),
- *subjection*: spresnenie čiastočnej informácie (obmedzenie podmienkou či zlepšenie informácie – ide o špeciálny druh modifikácie),
- *augmentation*: rozšírenie databázy o ďalšie možné stavy.

Ide o operácie odvodené z predošlých, preto sa im vo zvyšku práce budeme venovať okrajovo.

¹⁹ Formálnejšie: je to najmenšia množina generovaná uvedenými dvoma konštruktormi (prípadne s klasickou treťou klauzulou „nič iné je v množine formúl“). Používame jedine základné predikátové symboly $=, \neg, \wedge, \vee, \Rightarrow$.

²⁰ niekedy sa *modifikácia* samotná označuje pojmom *update*

²¹ záleží na navrhovanom modeli, či umožní nahrádzať aj neúplnými hodnotami, teda či je napr. možné substituovať za nejakú regulárnu hodnotu null

2.2 Integritné ohraničenia

Aby sme mohli hovoriť o tom, či databáza je *konzistentná* (t.j. vnútorne nerozporuplná), je potrebné stanoviť podmienky, ktoré tento dobrý stav databázy charakterizujú. Nazývajú sa *integritné ohraničenia*. Rozlišujeme tieto kategórie integritných ohraničení:

- *inherentné*: sú vlastnosťami konštruktorov modelu, spĺňa ich všetko, čo je modelom namodelované (sem patrí aj neduplikačné pravidlo, ktorého existencia vyplýva z definície relácie ako (pod)množiny, ako je spomenuté vyššie),
- *explicitné*: sú explicitne zadane programátorom či aplikačným programom, napr. procedúry overujúce platnosť vstupných dát,
- *implicitné*: sú logickým dôsledkom explicitných.

Ortogonalne k tejto klasifikácii integritných ohraničení je nasledovné rozlíšenie podľa toho, či sa ohraničenia týkajú statického alebo dynamického pohľadu na databázu:

- *statické*: pravidlá, ktoré určujú, či jednotlivé stavy databázy sú konzistentné,
- *dynamické*: pravidlá určujúce konzistentnosť prechodov medzi stavmi databázy.²²

Väčšina vlastností, ktoré definujeme pri navrhovaní modelu pre neúplnú informáciu, sú inherentné integritné ohraničenia (statické aj dynamické), a to práve z dôvodu, že sú vlastnosťami navrhovaných modelov.

2.3 Nevýhody relačného modelu

Hneď na začiatku úvah o relačnom modeli spomeňme niektoré jeho nevýhody, ktoré zväčša majú vplyv na neúplnú informáciu. Vychádzame pritom najmä z článku [Kent, 1979], ako aj z [Šešera–Mičovský, 1994] a [Mrázik, 1998]. Naším cieľom pritom nie je obhajoba objektovo-orientovaných databázových systémov, ale lepšie poznanie problémovej oblasti, relačných databáz, aj s ich slabými stránkami, aby sme sa mohli dôkladne venovať problému neúplnej informácie. Poznamenajme ešte, že pojem *záznam* (ekvivalent vety resp. rekordu) v tejto časti zdôrazňuje sekvenčnosť zápisu informácie, prípadne o pevnej dĺžke. Jednotlivé „časti“ záznamu sa nazývajú *položky* (*items*), avšak budeme ich označovať *hodnoty*.²³

2.3.1 Vertikálna homogenita

Stĺpce tabuľky sú rovnaké pre všetky záznamy. Pri návrhu však môžu vyvstať nasledovné požiadavky:

1. niektoré polia nemajú hodnotu vo všetkých záznamoch, ani sa neočakáva, že ju niekedy budú mať (rodné priezvisko u mužov zväčša nemá význam)
2. jeden atribút môže mať viacero významov pre rôzne záznamy (číslo pre rôzne záznamy môže znamenať cenu alebo počet kusov), rozlíšenie významu pritom závisí od logiky aplikácie, a nie je priamo zahrnuté v databáze (problém návrhu).

2.3.2 Horizontálna homogenita

Každý záznam obsahuje tie isté polia. Prípady, keď jedno pole obsahuje viac druhov informácie (napr. auto patrí zamestnancovi, oddeleniu, pobočke alebo banke), sa riešia pomocou jednoznačných identifikátorov (uid – uniformed id), čím narastá zložitosť

²² porov. [Rishe, 1988], 25n.

²³ Presnejšie, hodnoty atribútov danej tuple (reprezentovanej záznamom), ktoré zodpovedajú jednotlivým položkám.

kontroly jednoznačnosti identifikátora (ďalšie integritné ohraničenie). Horizontálna homogenita úzko súvisí s vyššie spomínaným druhým bodom vertikálnej homogenity.

2.3.3 Mená polí sú iba symboly

Inými slovami mená polí (atribútov) v skutočnosti nič neznamenajú, sú iba nálepkami (placeholders).

2.3.4 „Čierno–biele“ videnie sveta

Relačný model neumožňuje štruktúrovanú informáciu, z čoho pramení dvojaký extrém: buď je informácia totálne známa (úplná informácia), alebo je totálne neznáma (iba string, reťazcový koment bez sémantiky). Preto sa budeme venovať aj čiastočnej informácii, porov. str. 55 a nasl.

2.3.5 Nepopísateľnosť ‘relácií’

‘Relácie’ sa nedajú popísať, ale realizujú sa pomocou cudzieho kľúča. Nie je úplne zrejmý rozdiel medzi ‘reláciou’ a atribútom, ktorý je viac technický ako konceptuálny: ak je súčasťou záznamu kľúč niektorého iného záznamu (aj inej relácie), potom ide naozaj o ‘reláciu’ medzi týmto záznamom a tým iným záznamom. Naopak, ak touto súčasťou záznamu nie je cudzí kľúč, ide o hodnotu atribútu.

2.3.6 Rôznorodosť významu informácie (ADT)

Relačný model ťažko zachytí skutočnosť, že napr. vzhľadom k americkej legislatíve identifikátor publikácie Kongresovej knižnice označuje ako knihy, tak aj platne a ďalšie nosiče informácií, avšak z pohľadu ISBN ide o rozličné objekty (knihy majú ISBN, platne nie). Pritom v oboch prípadoch ide o jednoznačné identifikátory daných diel. Pri normalizácii do 3NF sa pritom v takom prípade z dvoch kandidátov na kľúč jeden vyberie (potom sa nazýva *primárny*), avšak na druhý z kľúčov sa zväčša zabudne. A ďalej, niektoré knihy majú aj viacero ISBN, či zamestnanec môže mať viac jednoznačných identifikátorov. Relačný model neumožňuje jednoduchým spôsobom modelovať abstraktné dátové typy, prípadne ich prieniky či zjednotenia.

Predošlé nevýhody, ako už bolo spomenuté, sú uvedené preto, aby bolo zrejmé, ktoré problémy treba pri návrhu databáz s dopredu zamýšľaným obsahovaním neurčitej informácie riešiť. Každý prístup sa preto venuje nasledovným bodom:

1. vymedzenie oblasti neúplnej informácie, ktorú chceme v systéme podporovať a prípadne pomenovať aj tie oblasti, ktorým sa nebudeme venovať,
2. *statický pohľad*: formálny či neformálny popis sémantiky uvažovanej neúplnej informácie (porov. napr. [Borgida–Mylopoulos, 1981]),
3. *dynamický pohľad*: rozšírenie relačnej algebry a/alebo kalkulu o črty podporujúce neúplnú informáciu, prípadne návrh dotazovacieho jazyka.

Na rozhraní medzi klasickým relačným modelom a objektovo-orientovanými databázovými modelmi je formálny algebraický popis relačného modelu, ktorý tvorí základ objektovo-orientovaných modelov.²⁴ Databáza je chápaná ako (*mnohodruhov*á) *algebra*, pričom súčasťou databázy sú *relácie* (dáta), *operácie* (relačnej algebry a tiež

²⁴ Pozri napr. [Golshani, 1985].

aktualizačné operácie) a rovnako aj *obmedzenia* (constraints, zodpovedajú integritným ohraničeniam). Keďže sa aj v rámci klasického relačného modelu snažíme o formálnu správnosť a väčšina autorov svoje modely a výsledky viac či menej precízne popisuje, nebudeme zväčša špeciálne zdôrazňovať isté črty, pochádzajúce z formálneho algebraického pohľadu na databázu.

2.4 Interpretácia dotazov

Databázy *neobsahujú* skutočný svet. Namiesto neho obsahujú *znalosť* o skutočnom svete. Používatelia systému sa dotazujú databázového systému na skutočnosť, systém však vie zodpovedať iba dotazy podľa jeho znalostí o skutočnosti. Takto vznikajú „dva svety“: skutočný svet (realita) a „svet databázového systému“. Lipski zaviedol pre formalizáciu týchto „dvoch svetov“ pojem *externej a internej reprezentácie dotazov* ako rozlíšenie medzi skutočnosťou a faktami uchovávanými v databáze. Neúplnú informáciu pritom neodlišuje od *maybe* informácie.

- *Externá interpretácia dotazu* znamená zodpovedanie na dotaz podľa skutočnosti, podľa toho, čo v realite platí. Výsledok externej interpretácie zisťovacieho dotazu je pravdivá odpoveď o objektoch reality, výsledok externej interpretácie doplňovacieho dotazu je množina (najväčšia) objektov, ktorá má v realite vlastnosť, na ktorú sa dotazom bolo pýtané.
- *Interná interpretácia dotazu* znamená zodpovedanie na dotaz podľa údajov, ktoré sú dostupné vo (vnútornom) svete databázy, teda podľa informácií z databázy.²⁵

V databázovom svete vieme vždy dať odpoveď iba na internú interpretáciu dotazu, pretože odpovedáme na základe uchovávaných faktov. Hoci sa snažíme uchovávať v databáze naozaj úplné údaje o realite, teda chceme vedieť presne zodpovedať na dotazy a snažíme sa, aby interná reprezentácia bola rovnaká ako externá interpretácia, v mnohých prípadoch je informácia nedostupná, neúplná, poškodená či nejednoznačná. A tak vieme dať iba čiastočnú odpoveď na externe interpretované dotazy. Našou snahou je čo najviac sa priblížiť k externej interpretácii dotazu. Preto pre dotaz Q definujeme dve hranice externej interpretácie Q :

- $\llbracket Q \rrbracket_*$ – množina objektov, pre ktoré vieme na základe údajov obsiahnutých v databáze rozhodnúť, či patria do externej interpretácie systému (dolná hranica)²⁶;
- $\llbracket Q \rrbracket^*$ – množina objektov, pre ktoré s určitosťou nevieme rozhodnúť, či patria do externej interpretácie systému, avšak nemôžeme ich ani s určitosťou odmietnuť (horná hranica)²⁷.

Inými slovami, $\llbracket Q \rrbracket_*$ a $\llbracket Q \rrbracket^*$ sú najlepšie hranice externej interpretácie dotazu Q logicky odvoditeľné zo systému.²⁸ Príkladu je venovaná časť 4.4.5.

²⁵ Interná reprezentácia pripomína úradníkov, ktorí sa často riadia tým, čo majú vo svojich údajoch anie tým, čo existuje v realite. Koho nemajú v zozname, ten pre nich nejestvuje.

²⁶ angl. *lower bound*

²⁷ angl. *upper bound*

²⁸ Lipski sa taktiež snažil hornú a dolnú hranicu externej interpretácie dotazov charakterizovať syntakticky. Napr. v práci [Jaegermann–Lipski, 1983] je charakterizovaná horná a dolná hranica externej interpretácie dotazov, ktoré sa pýtajú na numerické hodnoty.

2.5 Reprezentačný systém

Relačný výraz sa nazýva β -výraz, ak obsahuje jedine relačné operátory z množiny β . Pritom uvažujeme s projekciou (P), selekciou (S), pozitívnou selekciou (S^+), zjednotením (U), joinom (J), kartézskym súčinom (X), premenovaním stĺpcov (R), rozdielom (diferenciou, D) a reštrikciou (E).

V tomto odseku budeme pod *multitabul'kou* (nad množinou atribútov M) rozumieť tabuľku/reláciu (nad množinou atribútov M) s nullovými hodnotami resp. čiastočnou informáciou.²⁹ Nech $\mathfrak{S}(M)$ je množina všetkých množín (klasických) relácií³⁰ nad množinou atribútov M . Keďže v ďalšom predpokladáme rovnakú množinu atribútov M , budeme toto slovné spojenie (i zodpovedajúce označenie) vynechávať.

Reprezentácia multitabul'ky T (označenie $Rep(T)$)³¹ je zobrazenie z množiny multitabuliek do \mathfrak{S} (množiny množín klasických relácií). Každé multitabul'ke T priraduje množinu možných relácií, ktoré T zastupuje. Často preto hovoríme o reprezentácii multitabul'ky, hoci máme na mysli výsledok aplikácie operácie Rep na túto multitabul'ku. Pritom *práve jedna relácia* z reprezentácie $Rep(T)$ je skutočná v reálnom svete. (Reprezentácia pritom môže byť prázdna množina.) Dá sa povedať, že reprezentácia je sémantická funkcia, ktorá syntaktickému objektu (tabuľke s nullmi/čiastočnou informáciou) priraduje množinu sémantických objektov (možné tabuľky-relácie s úplnou informáciou).³² Reprezentácia Rep je pritom daná modelom pre neúplnú informáciu (porov. v časti 2.7).

Reprezentačný systém je trojica $\langle \mathfrak{R}, Rep, \beta \rangle$, kde \mathfrak{R} je množina multitabuliek, Rep je reprezentácia $Rep: \mathfrak{R} \rightarrow \mathfrak{S}$, a β je množina relačných operátorov, pričom spĺňajú podmienku, ktorú popíšeme len neformálne (podrobnejšie [Imieliński–Lipski, 1984], str. 769). Požaduje sa, aby existoval spôsob, konzistentný vzhľadom na operátory v množine β , ako definovať $f(T)$ pre ľubovoľný β -výraz a multitabul'ku $T \in \mathfrak{R}$, teda aby $Rep(f(T))$ bolo v istom zmysle ekvivalentné s $f(Rep(T))$ vzhľadom k β .³³

Ekvivalencia dvoch multitabuliek je definovaná nasledovne: Nech f je relačný výraz a $X \in \mathfrak{S}$. Pripomeňme, že X je množina všetkých relácií (tabuliek) bez nullov, ktoré sú priradené nejakej multitabul'ke. f -informácia v X , označenie X^f , je $\cap f(X)$, teda tabuľka, ktorá je prienikom všetkých tabuliek množiny $\{f(t) \mid t \in X\} =: f(X)$.

Množiny tabuliek X a Y sa nazývajú β -ekvivalentné alebo β -dotazovo ekvivalentné, označenie $X \equiv_{\beta} Y$, ak pre ľubovoľný β -výraz f platí $X^f = Y^f$. Tabuľka T β -reprezentuje X , ak $Rep(T) \equiv_{\beta} X$. Teda v definícii reprezentačného systému požadujeme, aby platilo

²⁹ Tento pojem je zvolený na odlišenie od klasickej tabuľky práve z dôvodu, že multitabul'ka reprezentuje mnoho tabuliek, presnejšie (potenciálne nekonečnú) množinu relácií (totiž namiesto nullu môžeme dosadiť akúkoľvek hodnotu atribútu resp. namiesto čiastočnej informácie ľubovoľnú dosadiť zodpovedajúcu úplnú informáciu).

³⁰ teda bez nullov

³¹ nazýva sa aj *informačný obsah* (angl. *information content*) multitabul'ky T

³² Táto množina sémantických objektov tvorí pre syntaktický objekt sémantický prot'ajšok (náprotivok).

³³ *Silný reprezentačný systém* je taký, ktorý spĺňa $Rep(f(T)) = f(Rep(T))$. V ďalšom popisujeme *slabý reprezentačný systém*.

$Rep(f(T)) \equiv_{\beta} f(Rep(T))$, slovne: či najprv aplikujeme β -výraz a potom vytvoríme reprezentáciu alebo opačne, výsledky musia byť β -dotazovo ekvivalentné. Obr. 1 túto požadovanú vlastnosť názorne ukazuje.

$$\begin{array}{ccc} T & \xrightarrow{Rep} & Rep(T) \\ \downarrow f & & \downarrow f \\ f(T) & \xrightarrow{Rep} & Rep(f(T)) \equiv_{\beta} f(Rep(T)) \end{array}$$

Obr. 1: Komutujúci diagram pre reprezentáciu

Reprezentačný systém je vlastne návod, ktorým sa prirad'uje jednotlivým multitabul'kám zodpovedajúca relácia bez nullov. Význam reprezentačného systému je nasledovný. Keď vykonávame relačné operácie nad tabuľkami s nullovými/čiasťovými hodnotami, môžeme spôsobiť poškodenie informácie. Avšak keď dotazovací jazyk, ktorý má k dispozícii používateľ, je „dostatočne slabý“ (len časť relačných operátorov je povolená), nie je toto poškodenie pre používateľa viditeľné.

2.6 Relačný model a logika

Relačné databázy boli už od svojho vzniku a sú stále tradične pokladané z pohľadu logiky za *interpretácie* resp. *modely* (porov. [Ullman, 1988]). V logickom zmysle slovo „model“ znamená konečnú prvorádovú štruktúru. Každá tupľa pritom zodpovedá atomickému predikátu, pričom predikátový symbol sa stotožňuje s názvom relácie. Ak uvažujeme klasický relačný model (bez neúplnej informácie), k danej relácii existuje jediný model, formula, ktorá je disjunkciou atomických predikátov zodpovedajúcich tupliam relácie. Pri neúplnej informácii údaje uchovávané v databáze definujú *viaceré možné svety* (modely). Inými slovami, neúplnosť informácie znamená „neúplne špecifikovaný model“ a reprezentuje množinu možných úplne špecifikovaných modelov. Tento prístup sa nazýva *modelovo-teoretický prístup* (angl. označenie *model-theoretic* uhol pohľadu).

Databázu je však možno chápať aj odlišne, a to ako *teóriu*, teda ako množinu logických formúl; tento prístup sa nazýva *dokazovaco-teoretický* (angl. označenie *proof-theoretic* uhol pohľadu). Jednotlivé údaje (tuple) uchovávané v databáze tvoria axiómy, z ktorých sa výsledok dotazu odvodzuje resp. ktoré sa použijú na dôkaz výsledku. V tomto prístupe je neúplnosť informácie zahrnutá akoby prirodzenejšie: neúplná informácia znamená, že máme len obmedzené množstvo formúl vypovedajúcich o realite.

[Vardi, 1986] nazýva databázy ako interpretácie *fyzické databázy* a databázy ako teórie *logické databázy*. Keďže pojmy „fyzická databáza“ a „logická databáza“ sú mätúce vzhľadom k označeniu „fyzický“ pre spôsob uchovávaní dát na pamäťových médiách a označeniu „logický“ na vyjadrenie „súvisiaci s matematickou logikou“, budeme v ďalšom používať dlhší popisný názov.

Rozdiel v oboch prístupoch (porov. aj [Nicolas–Gallaire, 1981]) je možné badať až pri databázach s neúplnou informáciou. Jadrom rozdielu je spôsob odpovedania na dotazy a celkové chápanie toho, čo databázy vyjadrujú.

Spôsob odpovedania na dotazy:

- modelovo–teoretický prístup: výsledok dotazu je *vypočítaný*, dotaz je *vyhodnotený* vzhľadom k interpretácii,
- dokazovaco–teoretický prístup: výsledok dotazu je *odvodený* logickou inferenciou, dotaz je potrebné *dokázať* použitím inferenčného systému.

Celkové chápanie databáz uvažovaných oboma spôsobmi:

- modelovo–teoretický prístup: databáza modeluje (zobrazuje) reálny svet,
- dokazovaco–teoretický prístup: databáza modeluje (zobrazuje) našu znalosť o reálnom svete.

Príklad 2: Reprezentácia relačnej databázy ako teórie

Majme databázu dodávateľov a dodávaných komponentov, ktorá pozostáva z nasledovných poznatkov:

$|Dodávateľ| = \{\text{Anton, Braňo, Cyril, Dano}\};$

$|Pečivo| = \{\text{perník, chlieb, vianočka}\};$

$|Dodáva| = \{\langle \text{Anton, perník} \rangle, \langle \text{Braňo, chlieb} \rangle, \langle \text{Cyril, vianočka} \rangle, \langle \text{Anton, chlieb} \rangle\}.$

V tabuľkovej forme vyzerajú naše poznatky nasledovne:

Dodávateľ

Anton
Braňo
Cyril
Dano

Pečivo

perník
chlieb
vianočka

Dodáva

Dodávateľ	Pečivo
Anton	perník
Braňo	chlieb
Cyril	vianočka
Anton	chlieb

Neformálne sa uvedená databáza zapisuje nasledovne:

$Dodávateľ(\text{Anton}), Dodávateľ(\text{Braňo}), Dodávateľ(\text{Cyril}), Dodávateľ(\text{Dano}),$
 $Pečivo(\text{perník}), Pečivo(\text{chlieb}), Pečivo(\text{vianočka}),$
 $Dodáva(\text{Anton, perník}), Dodáva(\text{Braňo, chlieb}), Dodáva(\text{Cyril, vianočka}),$
 $Dodáva(\text{Anton, chlieb}),$

Reprezentácia prvorádovej databázy (používame Reiterovu syntax, definovanú na str. 38) je množina nasledovných uzavretých formúl, pričom E značí predikát rovnosti:

$(x)[Pečivo(x) \equiv E(x, \text{perník}) \vee E(x, \text{chlieb}) \vee E(x, \text{vianočka})]$
 $(x)[Dodávateľ(x) \equiv E(x, \text{Anton}) \vee E(x, \text{Braňo}) \vee E(x, \text{Cyril}) \vee E(x, \text{Dano})]$
 $(x)(y)[Dodáva(x, y) \equiv E(x, \text{Anton}) \wedge E(y, \text{perník}) \vee E(x, \text{Braňo}) \wedge E(y, \text{chlieb}) \vee$
 $E(x, \text{Cyril}) \wedge E(y, \text{vianočka}) \vee E(x, \text{Anton}) \wedge E(y, \text{chlieb})]$

Samozrejme, musí platiť $\neg E(\text{Anton, Braňo}), \neg E(\text{vianočka, chlieb}), \neg E(\text{perník, chlieb}), \neg E(\text{chlieb, perník})$ atď. (porov. UNA, str. 38), čo je všeobecná požiadavka na relačné databázy: databáza nesmie obsahovať dva rovnaké riadky.

Pre externú reprezentáciu je možné určiť aj dolnú medzu: je to najmenší model pre danú teóriu. Databáza bez neúplnej informácie, zodpovedajúca dolnej medzi externej reprezentácie, je množina všetkých logických dôsledkov pôvodnej databázy (s neúplnou informáciou).

2.7 Predpoklad uzavretého vs. otvoreného sveta

Významný predpoklad, ktorý robí Reiter na prvorádovú databázu i rozšírené relačné teórie, je *predpoklad uzavretého sveta* (skratka CWA, angl. *closed world assumption*, porov. [Reiter, 1978]). Použitie CWA je často implicitné. Dotýka sa skutočnosti, že v databázach sa uchovávajú len *pozitívne fakty*, nie však *negatívne*. Ako sa vysporiadať s faktom, ktorý v databáze nie je? Predpoklad uzavretého sveta hovorí: keďže v databáze tento fakt (jemu zodpovedajúci záznam) nie je, neexistuje ani v skutočnosti.³⁴ Databáza napr. obsahuje reláciu *Dodáva*, v ktorej sú uchovávané dvojice dodávateľov a výrobkov, označujúcu množinu znalostí o *dodávaní výrobkov*, nie však reláciu, ktorá by obsahovala údaje o *nedodávaní výrobkov* (relácia obsahujúca dvojice Dodávateľ–Pečivo s významom „daný dodávateľ *nedodáva* dané pečivo“). Podľa predpokladu uzavretého sveta teda predpokladáme, že databáza akoby obsahovala túto „duálnu reláciu“ *Nedodáva*, v ktorej sú všetky tuple, neuvedené v relácii *Dodáva*. Hoci sa zdá, že CWA okliešťa schopnosť systému modelovať realitu, je to nevyhnutný predpoklad v praktických databázach, pretože množstvo negatívnych faktov značne prevyšuje množstvo pozitívnych, ktoré sú v databáze uchovávané. Toto odôvodnenie je dostatočné na používanie predpokladu uzavretého sveta.

Predpoklad otvoreného sveta (OWA)³⁵ sa naopak v prípade, keď fakt nie je priamo zahrnutý v systéme, o tomto fakte nevyjadruje. Rozdiel medzi CWA a OWA je možné v prípade uvažovania o databázach ako teóriách demonštrovať nasledovne. Ak sa pri dokazovaní pod CWA nenájde dôkaz pre pozitívny fakt, potom sa predpokladá platnosť negácie tohoto faktu. Pri dokazovaní pod OWA sa to, že pozitívny fakt nemožno dokázať, nepokladá za dôkaz platnosti jeho negácie.

Formálne môžeme rozdiel medzi OWA a CWA zapísať nasledovne. Nech T je tabuľka s neúplnou informáciou bližšie nešpecifikovaného druhu a Val je množina valuácií pre danú neúplnú informáciu (*valuácia* je predpis (zobrazenie), ktoré priradzuje neúplnej hodnote nejakú jej zodpovedajúcu úplnú hodnotu, napr. nullu priradí regulárnu hodnotu). Potom reprezentácia tabuľky T

- v OWA interpretácii je $Rep_{OWA}(T) = \{R \mid v(T) \subseteq R, v \in Val\}$,
- v CWA interpretácii je $Rep_{CWA}(T) = \{v(T) \mid v \in Val\}$.

Rozdiel v interpretácii dotazov podľa CWA a OWA uvedieme ešte v časti 3.4.3. [Codd, 1979] poznamenáva, že v databázovom systéme je pre niektoré relácie užitočné povoliť OWA interpretáciu a pre ostatné CWA interpretáciu.

³⁴ Hoci takéto uvažovanie je blízke úradníkom, nemožno považovať CWA za synonymum obmedzenosti.

³⁵ angl. *open world assumption*

2.8 Terminologické poznámky

„Dospelí sami nikdy nič nechápu
a deti to unavuje,
keď im treba stále čosi vysvetľovať.“
Antoine de Saint-Exupéry³⁶

Niektoré pojmy vyskytujúce sa v práci vyžadujú vysvetlenie, prečo boli vybraté práve takým spôsobom, ako ich používame. Toto vysvetlenie je obsahom tohto odseku. Slovenské ekvivalenty boli konzultované s dielom [Ragan, 1998].

Kľúčový pojem, označujúci oblasť informatiky, ktorou sa v tejto práci zaoberáme, a tiež teoretický model s jeho implementáciami, „data base“ či „database“ používame vo forme „databáza“ (aj [Ragan, 1998]), hoci sa v slovenčine používa aj „dátabáza“ (napr. [Scheber, 1988]).

Pojem „tupľa“ nie je slovenského pôvodu (je poslovenčením anglického pojmu *tuple*, ktorý sa používa v anglicky písaných seriáloch a monografiách), avšak preklad *tica* by pravdepodobne nevyjadroval zamýšľaný pojem a použitie pojmu *n-tica* (ako používa napr. [Scheber, 1988]) automaticky predpokladá mohutnosť *n* komponentov. Naviac, nie vždy sa pod tupľou myslí priamo *n-tica*, konkrétne v modeloch, ktoré sú založené na zložitejších konceptoch (Biskup, Reiter). Použitie pojmu *vector* (*vektor*) ako alternatívy pre *n-ticu* podľa [Tsichritzis–Lochovsky, 1981] je síce logicky opodstatnené, avšak nie je natoľko zaužívané ako anglické *tuple*. V slovenskej informatickej terminológii sa označuje výlučne pojmom *n-tica*, avšak nesprávne sa uvádza aj ako *veta* resp. *záznam* (angl. *record*). Tieto pojmy však vyjadrujú fyzický spôsob uloženia dát, porov. napr. [Scheber, 1988], [Tsichritzis–Lochovsky, 1981].

Slovné spojenia „null“, „nonnull“, „null value“ apod. používame v slovenskom ekvivalente „null“, „nenull“, „nullová hodnota“, pretože pojem „nullová hodnota“ (s jedným l) je mätúci a zvádza k interpretácii „hodnota 0“ (nula), čo je aj jeden z významov tohto slova.

Hoci slovo „dotaz“ (označujúce výber dát) je čechizmom, je zaužívané aj v slovenskej informatickej terminológii (porov. [Scheber, 1988]) ako ekvivalent anglického „query“; v tomto zmysle ho budeme používať aj v tejto práci. Proces kladenia dotazov označujeme slovesom „dotazovať (sa)“.

Podobne pojem „query language“ používame v slovenskom ekvivalente „dotazovací jazyk“ napriek tomu, že sa používajú slovenské výrazy „zist'ovací jazyk“, „opytovací jazyk“³⁷. Jemu zodpovedajúci preklad slova „query“ ako „opyt“ alebo „otázka“ však nevyjadruje presne to, čo sa myslí pod pojmom „query“.

³⁶ Antoine de Saint-Exupéry: *Malý princ*, str. 8 (vyd. Mladé letá, Bratislava 1997)

³⁷ Posledný menovaný použitý napr. v denníku SME (27. novembra 1998), príloha Počítače, str. 16. Tento preklad znie ako „ohňová stena“ alebo prenesene „vodná priekopa“ na vyjadrenie pojmu „firewall“ v slovenčine.

Pojem „objekt“ používame v zmysle „entita“, „objekt reality“, „predmet, zviera, človek“, „pozorovaný jedinec“ apod., nepoužívame ho teda ako objekt známy z paradigiem objektovo orientovaného programovania.

Rovnako pojem „inštancia“ použitý v zmysle „databáza“, „konkrétna množina relácií“ nepoužívame v zmysle paradigiem OOP.

Termín „relácia“ (v jednoduchých úvodzovkách) používame na zdôraznenie, že ide o matematickú (nie databázovú) reláciu ako podmnožinu kartézskeho súčinu.

Termín „signatúra relácie“ znamená množinu resp. usporiadanú k -tícu množín hodnôt (resp. názvov stĺpcov) tabuľky.

Pojem „update“ (napr. update operácie) používame v zmysle „aktualizácia“, „aktualizačný“, avšak niekedy uvádzame aj pôvodný anglický pojem.

Pojem „spojenie“ (angl. „join“) používame v dvoch významoch: ako relačný join (spojenie) nad reláciami (napr. natural join), kedy ho zväčša ponechávame v anglickej podobe, a ako algebraické spojenie prvkov zväzu. Rozdiel je z kontextu jasný.

Lipski, uvedomujúc si potrebu formálne definovať modely pre neúplnú informáciu, zaviedol pojem „informačný systém“ na označenie matematického modelu pre databázy s neúplnou informáciou. Keďže však tento termín je mätúci v súvislosti s pojmom „informačný systém“ zo softvérového inžinierstva, nebudeme ho používať a namiesto neho budeme hovoriť „model databázy (s neúplnou informáciou)“ resp. „(databázový) systém“.

3 Nullové hodnoty

*Už to, že vieme, že nič nevieme,
je vedomosť, ktorá je lepšia,
ako keby sme vôbec nič nevedeli,
ani to, že nič nevieme.*

Neúplná informácia sa vyskytuje v praxi veľmi často, niekedy – žiaľ – i častejšie ako informácia úplná. Nasledujúce body naznačujú zdroje neúplnosti všeobecne i špeciálne v relačných databázach:

1. nepresnosť merania,
2. chabá ľudská pamäť („nepamätám si, ale som si istý, že značka auta začínala písmenami JV, ako moje iniciálky“),
3. nedostupnosť informácie; v konkrétnom prípade ide o aspekt bezpečnosti informačných systémov: používateľ má iba čiastočnú info o dátach v IS, jeho view (pohľad) je reštringovaný prístupovými právami,
4. viacznačnosť informácie spôsobená nepresnosťou (je to objekt z dreva, ale neviem akého) alebo tým, že informácia pochádza z viacerých zdrojov³⁸ (pri merdžovaní databáz),
5. predpoklad univerzálnej relácie³⁹,
6. aktualizálny (update) problém⁴⁰,
7. východzia (default) hodnota, ktorá ešte používateľom nebola zmenená alebo odsúhlasená.

V tejto časti sa venujeme základnému druhu neúplnej informácie, a to *nullovým hodnotám*.

3.1 Sémantika nullov

Sémantika pojmu „nullová hodnota“ (*null value*) môže byť rôzna. Podľa skupiny ANSI/X3/SPARC existuje až 14 interpretácií nullových hodnôt ([ANSI, 1975], porov. [Sugihara]):

1. neaplikovateľná – not applicable (napr. rodné meno(priezvisko) mužského zamestnanca)
2. aplikovateľná, ale ešte neexistujúca – applicable, but does not yet exist (napr. meno za vydata pre slobodnú zamestnankyňu)
3. existujúca, ale logicky nie je dovolené ju uchovávať (napr. vierovyznanie zamestnanca)
4. existujúca, ale nie je známa – not knowable (napr. posledná výška platu nového zamestnanca, ktorý pracoval pre inú spoločnosť)
5. existuje, ale ešte nie je zaznamenaná – not yet logically stored (napr. medicínska história (chorobopis) novoprijatého zamestnanca)
6. logicky uchovávaná, ale potom logicky zmazaná (subsequently logically deleted)
7. uchovávaná, ale ešte nedostupná – not yet available
8. dostupná, ale podliehajúca zmenám – available, but undergoing change (už nemusí byť platná)

³⁸ [DeMichiel, 1989]

³⁹ [Goldstein, 1981]

⁴⁰ [Lipski, 1983a]

- zmena začala, ale nová hodnota ešte nebola spočítaná
 - zmena neúplná (incomplete), aktuálne hodnoty sú čiastočne nové, čiastočne staré – môžu byť nekonzistentné
 - zmena neúplná, ale časť nových hodnôt nie je zapísaná (neprebehol commit)
 - zmena úplná, nové hodnoty však ešte nie sú zapísané (committed)
9. dostupná, ale postráda vierohodnosť – kind of suspect validity (unreliable)
 - možná chyba v konceptuálnom obdržaní dát (failure in conceptual data acquisition)
 - možná chyba vo fyzickom uchovaní dát (failure in physical data maintenance)
 10. dostupná, ale vadná (invalid)
 11. chránená pre triedu konceptuálnych objektov – secured for a class of conceptual objects
 12. chránená pre jednotlivý objekt – secured for an individual object
 13. chránená v danom čase – secured at this time
 14. odvodená (derived) z nulových hodnôt ľubovoľných predošlých typov.

Ako sme spomínali vyššie v časti venovanej nevýhodám relačného modelu (str. 9), niektoré interpretácie nulových hodnôt vyplývajú priamo z faktu, že uvažujeme o relačnom modeli. Ďalšie sú z nich odvodené, prípadne na nich nezávisiace. Avšak väčšina autorov sa zhoduje v tom, že existujú dve základné interpretácie:

- (i) **unknown** alebo tiež **missing**⁴¹: hodnota (ešte zatiaľ) nie je známa (attribute applicable but its value at present unknown), označenie **un** alebo \exists – zodpovedá ANSI nullom 3 až 13;
- (ii) **inapplicable** (alebo tiež **placeholder null**⁴²): hodnota neexistuje (does not exist) príp. nie je aplikovateľná (attribute/property inapplicable), označenie **ne** (nonexistent) – zodpovedá ANSI nullom 1 a 2.

Pre prvú z týchto interpretácií bolo rozvinutých viacero prístupov. V tejto súvislosti je pozoruhodné, že samotný pojem „null“ zahŕňa prirodzene oba významy:⁴³

1. *nula, prázdny, nulový (un)*,
2. *neplatný, nepoužiteľný (ne)*.

[Biskup, 1981] dodáva, podľa jeho slov „prinajmenšom z teoretického hľadiska“, ešte ďalší význam nullu, a to

- (iii) **arbitrary**: hodnota je ľubovoľná (attribute is applicable but its value is arbitrary), označenie \forall .

[Zaniolo, 1984] zaviedol null, ktorý zhŕňa všetky vyššie uvedené interpretácie s tým, že nevieme, o ktorú interpretáciu sa jedná, preto ešte „jeden“ význam nullu:

- (o) **no information**: neviem, aká je daná informácia, ba či vôbec existuje – viem iba, že informáciu nemám, označenie **ni**.

Bližšie o týchto dvoch interpretáciách nulov v stati venovanej Biskupovmu resp. Zaniolovmu prístupu.

⁴¹ napr. [Vassiliou, 1980], [Goldstein, 1981]

⁴² napr. [Goldstein, 1981]

⁴³ porov. [Ragan, 1998]

[Golshani, 1985] spomína špeciálny druh nullu, a to

- (e) **type error**: napr. argument je neznámeho typu (nie je súčasťou definičného oboru),

avšak tomuto typu nullov sa dá vyhnúť precíznou kontrolou typov (type checking), naviac sa vyskytuje len v algebraických formálnych modeloch. V praxi sa prejaví ako chyba aplikačného programu, preto je ťažko vôbec hovoriť o nulle „error“-typu. Preto sa ním nebudeme zaoberať.

[Codd, 1993] aj [Date, 1995] správne poznamenávajú, že hoci hovoríme „nullová hodnota“, v skutočnosti vôbec nejde o (regulárnu) hodnotu.⁴⁴ (Presnejšie, keď ideme obohatiť doménu o null, nemôžeme hovoriť, že ju obohacujeme o ďalšiu *hodnotu*, avšak ak máme doménu definovanú ako (napr.) zjednotenie množiny prirodzených čísel a špeciálneho symbolu ω pre null, v takom prípade už tento null *je* hodnotou.) Sme si vedomí aj toho, že podľa niektorých celá oblasť nullov je omylom⁴⁵, avšak my budeme pristupovať k oblasti nullov a neúplnej informácie s plnou vážnosťou.

Základné črty nullových hodnôt predstavíme na príklade jednoduchej relácie *Meno–Telefónne číslo*, kde v prvom stĺpci je uvedené meno zamestnanca a v druhom stĺpci jeho telefónne číslo. Pritom pre jednoduchosť nedefinujeme formálne ani množiny atribútov ani reláciu samotnú (pri jednotlivých prístupoch budú uvedené potrebné definície). Príklad 3 ukazuje základné dve interpretácie nullov.

Príklad 3: Fero nemá telefón, Mišo telefón má, ale nepoznám jeho číslo

Meno	Telefónne číslo
Fero	NULL (does not exist, ne)
Mišo	NULL (unknown, un)
Jano	6389432

Pozorné skúmanie klasického príkladu s telefónmi a telefónnymi číslami však vedie k tomu, že treba striktné rozlíšiť „vlastnenie telefónu“ od „mať telefónne číslo“, pretože vlastne môžu nastať až štyri prípady:

- (1) neviem, či má alebo nemá telefón,
- (2) viem, že nemá telefón,
- (3) viem, že má telefón, ale neviem jeho telefónne číslo,
- (4) viem, že má telefón a viem aj jeho telefónne číslo.⁴⁶

Z prísne formálneho hľadiska, ako pripomína aj [Zaniolo, 1984], by bolo potrebné databázu navrhnuť tak, že obsahuje aj ďalší stĺpec, ktorý bude obsahovať (taktiež potenciálne neúplnú) informáciu, či daný zamestnanec má telefón.

⁴⁴ Date: „The whole point about nulls is precisely that they are not values.“

⁴⁵ „nulls are a mistake“, [Date, 1995], str. 571

⁴⁶ [Biskup, 1981] neuvádza možnosti 1 a 3, pretože u neho dvojica $\langle \text{Meno} = m, \text{Telefón} = t \rangle \in r_2$ znamená „*m* má telefón a *t* je jeho číslo **alebo** *m* nemá telefón a *t* = Λ “. Symbol Λ pritom označuje null typu **ne** (ANSI 1).

Príklad 4: Fero nemá telefón, Mišo telefón má, ale nepoznám jeho číslo, napokon neviem, či Andrej má telefón

Meno	Má-nemá_telefón	Telefónne číslo
Andrej	NULL (neviem, un)	NULL (–, ni)
Fero	NIE (nemá telefón)	NULL (does not exist, ne)
Mišo	ÁNO (má telefón)	NULL (unknown, un)
Jano	ÁNO (má telefón)	6389432

Príklad 4 obsahuje NULL uvedený pre Andrejovo telefónne číslo, ktorý zodpovedá „no information“ nullu, ako ho zaviedol [Zaniolo, 1984]. NULL uvedený pri vlastníctve telefónu Andrejom zodpovedá klasickej interpretácii „unknown“, ANSI 4.⁴⁷ Pre danú tabuľku platí navyše integritné ohraničenie dané funkčnou závislosťou

(Telefón \in Číslo \vee Telefón = **un**) \rightarrow Má-nemá_telefón = ÁNO, inými slovami, ak mám v stĺpci telefónne číslo zamestnanca uvedené číslo alebo unknown null, znamená to, že daný zamestnanec má telefón.

Môže taktiež platiť integritné ohraničenie

(Má-nemá_telefón = **un**) \rightarrow Telefón = **ni**, vyjadrujúce fakt, že pokiaľ nie je známe, či daná osoba vlastní telefón, nevieme nič povedať o jej telefónnom čísle. V takomto prípade je neprípustné, keby tabuľka z predošlého príkladu obsahovala tupľu

\langle Meno: Jana, Má-nemá_telefón: NULL, Telefónne_číslo: 823543 \rangle .

Na druhej strane (ako uvádza poznámka 48 pod čiarou), daná tupľa vyjadruje fakt, že poznáme Janino telefónne číslo, hoci nevieme, či už Jane telefón namontovali. V takom prípade použitie predošlého integritného ohraničenia by znamenalo stratu čiastočnej informácie.

Dajú sa nájsť zodpovedajúce analógie ostatným významom nullových hodnôt⁴⁸, no je zrejmé, že takéto prísne rozlišovanie by v praxi viedlo k zbytočnému rozbujujaniu objemu dát a sotva by prispelo k sprehľadneniu uchováanej informácie.

Vynára sa ďalšia otázka, či je jednotlivý null závislý na doméne (*atribútovo závislý null*), alebo nie, teda je rovnako použiteľný pre rôzne domény. Avšak aj autori, ktorí si uvedomujú potrebu syntakticky odlišných nullov, napr. [Goldstein, 1981] či [Biskup, 1981], napokon používajú iba jeden, atribútovo nezávislý null, pričom poznamenávajú, že doménovo závislé nully komplikujú čitateľnosť článkov a neznamenajú zlepšenie dosiahnutých výsledkov.

⁴⁷ Biskupov **arbitrary** null uvedený v stĺpci Telefónne číslo by v databáze vyjadroval anekdotu o človeku, ktorý je všade: je jedno, na ktoré konkrétne číslo zavoláme, vždy sa dovoláme práve k nemu.

⁴⁸ Tu sú uvedené niektoré ďalšie možné situácie. Uvedená je dvojica \langle Má-nemá_telefón, Telefónne_číslo \rangle a pri NULLe tiež číslo zodpovedajúcej interpretácie ANSI.

- \langle ÁNO, NULL 2 \rangle Vlastní telefón, ale telekomunikácie ešte nepridelili číslo.
- \langle NULL 4, 6389432 \rangle Nevie, či už má namontovaný telefón, ale telekomunikácie číslo prideliť (číslo rezervované, aby bolo napr. zhodné s číslom mobilu alebo číslom pobočky firmy v inom okrese).
- \langle NIE, 6389432 \rangle Číslo už je pridelené, ale telefón ešte nie je namontovaný.
- \langle ÁNO, NULL 12/13 \rangle Vlastní telefón, ale má „nezverejnené“ (utajené) číslo.

Taktiež je pre manipuláciu s nullmi dôležité, či sú nully špeciálne symboly nepatriace do atribútovej domény, alebo či sú obohatením atribútovej domény, a teda jej regulárnou súčasťou. Niektorí autori sa riadia jednoduchým pravidlom: ak potrebujeme špeciálnu hodnotu pre nepredvídateľnú udalosť, jednoducho prehlásime túto hodnotu za regulárnu.

3.2 Je „inapplicable“ null opodstatnený?

Viacero autorov poznamenáva, že inapplicable null je príznak zlého návrhu schémy. Stačí si položiť otázku, prečo by mal byť Fero uvedený v tabuľke s telefónnymi číslami, ak nemá telefón. Predošlé príklady však nezahŕňajú celú reálnu schému tabuľky, keď okrem mena a telefónneho čísla býva uvedené v tabuľke množstvo iných údajov (napr. dátum narodenia, ČOP, rodné číslo). Keď je teda pri Andrejovom a Ferovom telefónnom čísle uvedený NULL, mali by sme pozmeniť schému tak, že vytvoríme dve tabuľky, pričom v prvej bude namiesto stĺpca Telefónne_číslo len indikátor, či zamestnanec má telefón, a v druhej budú uvedení zamestnanci, o ktorých je známe, že majú telefón, spolu so svojim telefónnym číslom. Tým sa odstráni ako **ne**, tak aj **ni** nully. Táto dekompozícia je podrobnejšie popísaná v časti 5.5 (str. 79).

Príklad 5: Dekompozícia tabuľky s telefónnymi číslami

Meno	Má-nemá_telefón
Andrej	NULL (neviem, un)
Fero	NIE (nemá telefón)
Mišo	ÁNO (má telefón)
Jano	ÁNO (má telefón)

Meno	Telefónne_číslo
Mišo	NULL (neviem, un)
Jano	6389432

3.3 Atomický null – Coddove tabuľky

[Codd, 1979] predpokladá sémantiku null hodnoty ako „value at present unknown“ (hodnota teraz ešte neznáma, „missing value“, chýbajúca hodnota, **un**) a označuje ju symbolom ω . Tento null je atomický a doménovo nezávislý. Niekedy sa pri citácii Coddových null hodnôt používajú iné symboly, napr. v [Paredaens, 1989] symbol @. My budeme odlišný symbol @ používať, keď budeme chcieť zdôrazniť, že null má rovnaké vlastnosti, ale používame ho v inom ako Coddovom modeli.⁴⁹

Tabuľky relácií alebo ich častí, v ktorých sú použité Coddove null hodnoty, sa nazývajú *Coddove tabuľky* (*Codd tables*). Platí pritom nasledujúce *obmedzenie* resp. *pravidlo*:

nully sú zakázané v každom komponente primárneho kľúča relácie.

Nazýva sa aj *entitná integrita*. Toto obmedzenie je dôležité, pretože inak by prítomnosť nullov spôsobila konflikt s definíciou primárneho kľúča. Týka sa nielen statického pohľadu na databázu, ale aj dynamického: žiadna z operácií *insert*, *update*, *delete* nesmie spôsobiť porušenie platnosti entitnej integrity. Takéto obmedzenie predpokladajú všetci autori, hoci mnohí ho ani výslovne neuvádzajú. Taktiež my v ďalšom často toto obmedzenie neuvedieme okrem prípadov, kde je potrebné výslovne ho zdôrazniť.

⁴⁹ Napr. pri Grantových intervalových hodnotách, str. 55 a nasl.

Majúc na zreteli toto obmedzenie, ľubovoľný výskyt nullov („unknown“ typu) môže byť nahradený pri operácii *update* nenullovou hodnotou a opačne, pokiaľ neexistuje explicitné integritné ohraňenie, ktoré by to nepripúšťalo.

3.3.1 Trojhodnotová logika

Zatiaľ nezodpovedanou otázkou ostáva, aká bude pravdivostná hodnota výroku $x = y$, ak jedna z hodnôt x , y alebo obidve sú nully. Z tohto dôvodu Codd rozširuje relačný kalkul a relačnú algebru o logickú neznámu hodnotu *maybe* (okrem klasických hodnôt *true* a *false*). Aby sa – rovnako ako pravdivostné hodnoty *true*, *false* – dala uchovávať v databáze, označuje ju rovnako ako null, teda „ ω “. *maybe* (možno platí, možno nie) je výsledkom každej operácie, ktorá narába s neznámou (unknown) informáciou. Logika pracujúca s tromi pravdivostnými hodnotami *true*, *false* a *maybe* sa nazýva *trojhodnotová logika* (označuje sa 3VL z angl. *three-valued logic*), ktorá je podporovaná aj SQL štandardom (SQL92). Treba poznamenať, že skutočná trojhodnotová logika je odlišná od „akoby“-trojhodnotovej logiky, kde sa *maybe* (alebo tiež *unknown*) používa len ako zástupca (angl. *placeholder*) jednej z dvoch pravdivostných hodnôt *true*, *false* (porov. [Grant, 1980], str. 365).

Stručne popíšeme trojhodnotovú logiku vrátane jej prístupu k nullovým hodnotám. Platia nasledovné pravidlá pre vyhodnocovanie výrazov:

- ak je aspoň jeden operand výrazu nullový, aj výsledok je null,
- na test toho, či je hodnota nullová, slúži logická operácia **maybe** resp. operátor IS_NUL,
- konverziu null hodnoty na nenullovú zabezpečuje operátor IF_NUL (SQL funkcia NVL),
- popri *true* a *false* je pridaná špeciálna boolovská hodnota *unk(nown)*, zastupujúca pravdivostnú hodnotu *maybe* (ω),
- výsledok porovnania s nullom je *unk*.

Hoci *unk(nown)* ako tretiu pravdivostnú hodnotu označujeme rovnako ako null (či už rovnako *unk* alebo rovnako symbolom ω), zdôrazňujeme, že ide o dve rôzne úrovne abstrakcie. V rovine relačnej algebry a vyhodnocovania dotazov je množina boolovských hodnôt obohatená o tretiu hodnotu. V rovine domén hodnôt atribútov tuplíc/relácií je ku každej množine pridaná špeciálna hodnota *null* (ozn. ω), a to aj k doméne, ktorá má dva prvky, syntakticky zapísané *true* a *false* (používateľský databázový typ *Bool(ean)* popri napr. *Int(eger)*, *Char(acter)*, *String*). Porov. [Date, 1995], str. 575.

Tabuľka 2 obsahuje pravdivostné tabuľky pre logické operácie **and**, **or** a **not** v trojhodnotovej logike; existenčný resp. univerzálny kvantifikátor sa správajú ako iterovaný **or** resp. **and**. Niekedy sa tiež definuje logická operácia **maybe** odpovedajúca na otázku, či daná hodnota je nullová.

Tabuľka 2: Pravdivostné tabuľky trojhodnotovej logiky

and	F	ω	T	or	F	ω	T	not		maybe	
F	F	F	F	F	F	ω	T	F	T	F	F
ω	F	ω	ω	ω	ω	ω	T	ω	ω	ω	T
T	F	ω	T	T	T	T	T	T	F	T	F

Ak by sme jednotlivé pravdivostné hodnoty nahradili číslami 1 (*true*), $\frac{1}{2}$ (*maybe*), 0 (*false*), dajú sa logické spojky vyjadriť ako číselné funkcie:

$$\begin{aligned}\text{not}(x) &=_{df} 1 - x, \\ x \text{ and } y &=_{df} \min(x, y), \\ x \text{ or } y &=_{df} \max(x, y),\end{aligned}$$

a napokon použitím Iversonovej konvencie môžeme zapísať

$$\text{maybe } x =_{df} (x = \frac{1}{2}).$$

3.3.2 Null-substitučný princíp

Nech S je neprázdna unárna relácia. Potom výrazom

$$\begin{aligned}\omega &\in S, \\ \langle \omega \rangle &\subseteq S\end{aligned}$$

priradíme pravdivostnú hodnotu *maybe* (ω). Môže sa to zdať kontraintuitívne, ale len dovedy, kým si uvedomíme, že každý null (teda výskyt ω) zastupuje potenciálne rôzne hodnoty. Presnejšie, pravdivostný výraz V má hodnotu ω vtedy a len vtedy, keď (po nahradení všetkých definovaných výrazov individuálnymi premennými) platia obe nasledujúce podmienky:

- (1) každý výskyt ω vo výraze V sa dá nahradiť nenullovou hodnotou (možno odlišnou pre každý výskyt) tak, že hodnota výrazu V bude T (*true*),
- (2) každý výskyt ω vo výraze V sa dá nahradiť nenullovou hodnotou (možno odlišnou pre každý výskyt) tak, že hodnota výrazu V bude F (*false*).

Tento spôsob pridelenia hodnoty ω výrazu sa nazýva *null-substitučný princíp* (*null substitution principle*). Trojhodnotová logika je s týmto princípom konzistentná. Čo sa týka testovania nerovnosti, null-substitučný princíp sa zjednoduší, pretože sa môžeme vyhnúť nahrádzaniu ω ľubovoľnou hodnotou v číselnom alebo lexikografickom usporiadaní, teda pravdivostnú hodnotu ω priradíme všetkým výrazom tvaru $x \theta y$, kde θ je jeden zo symbolov $<, \leq, =, \geq, >$ vždy, keď x alebo y je null.

Príklad 6: Vyhodnocovanie dotazov v Coddovej trojhodnotovej logike

X	Y	$X < Y$	$X = Y$	$X \equiv Y$
2	2	F	T	T
2	3	T	F	F
2	ω	M	M	F
ω	3	M	M	F
ω	ω	M	M	T

Príklad 7: Vyhodnotenie niektorých výrazov [Codd, 1979]

Nech sú dané nasledujúce relácie:

$$\begin{aligned}R &= \{\langle \omega \rangle, \langle 1 \rangle\} \\ S &= \{\langle \omega \rangle, \langle 1 \rangle, \langle 2 \rangle\} \\ T &= \{\langle \omega, 1 \rangle, \langle y, \omega \rangle\} \\ U &= \{\langle \omega, 3 \rangle, \langle x, \omega \rangle\} \\ V &= \{\langle x, \omega \rangle, \langle y, 3 \rangle, \langle z, 1 \rangle\}\end{aligned}$$

Nasledujú uvedené pravdivostné hodnoty niektorých výrazov.

$$\begin{aligned}\text{F (false):} \quad & \omega \in \emptyset \quad T \subseteq S \quad V \subseteq U \quad U \subseteq R \\ \omega \text{ (maybe):} \quad & R \subseteq S \quad S \subseteq R \quad T \subseteq U \quad U \subseteq T \quad T \subseteq V \quad U \subseteq V\end{aligned}$$

Zastavme sa pri výraze $S \subseteq R$. V pôvodnom Coddovom prístupe z r. 1975 sa vyhodnotil na *false*, na čo upozornil [Grant, 1979]. Dôvod, prečo sa $S \subseteq R$ výraz vyhodnotí *maybe* je ten, že hoci existuje veľa prípadov nahradenia unknown nullu takou hodnotou, že

$\neg S \subseteq R$, napríklad

$$R = \{\langle 0 \rangle, \langle 1 \rangle\}, S = \{\langle 0 \rangle, \langle 1 \rangle, \langle 2 \rangle\}, \text{ keď platí } R \subseteq S,$$

alebo

$$R = \{\langle 0 \rangle, \langle 1 \rangle\}, S = \{\langle 3 \rangle, \langle 1 \rangle, \langle 2 \rangle\}, \text{ keď neplatí } R \subseteq S \text{ ani } S \subseteq R,$$

existuje aj nasledujúci prípad nahradenia nullov: v relácii R nahradíme null hodnotou 2 a v relácii S jednou z hodnôt 1 alebo 2, čím dostaneme

$$R = \{\langle 1 \rangle, \langle 2 \rangle\} = S,$$

čiže platí $S \subseteq R$.

Základom množinovej teórie je operácia príslušnosti \in (byť prvkom). Bežná definícia hovorí, že výraz

$$e \in \{e_1, e_2, e_3\}$$

je ekvivalentný výrazu

$$e = e_1 \vee e = e_2 \vee e = e_3.$$

Ako pripomína [Nakano, 1990], keby sme aplikovali túto definíciu na výrazy obsahujúce tuple s nullmi, potom by neplatilo, že (použijúc predošlý príklad)

$$\langle \omega, 3 \rangle \in \{\langle \omega, 3 \rangle, \langle x, \omega \rangle\} = U,$$

pretože $\omega = \omega$ nie je pravdivé, lebo sa vyhodnotí použitím null-substitučného princípu na *maybe*. Tento výsledok je však kontraintuitívny, a ako neskôr uvidíme, v Coddovom prístupe nie je ojedinelý. Predsa požadujeme, aby o tupli bolo známe, že patrí do relácie, do ktorej v skutočnosti patrí. Preto sa zavádza *rozšírený operátor porovnania* \equiv s významom „striktného vyhodnocovania“, teda vyhodnocovania na syntaktickej úrovni. $\mu \equiv v$ je *true* práve vtedy, keď ide o dva syntakticky odlišné objekty (nezáleží pritom, či sú alebo nie sú nullu). Inak $\mu \equiv v$ je *false*. Pokiaľ μ, v nie sú nullu, správa sa \equiv ako $=$. Príklad 6 znázorňuje rozdiel vo vyhodnocovaní $a \equiv$.

Potom môžeme rozšíriť definíciu príslušnosti prvku k množine nasledovne:

$$e \in \{e_1, \dots, e_n\} \Leftrightarrow_{df} e \equiv e_1 \vee \dots \vee e \equiv e_n.$$

Každá nebázová n -árna relácia môže obsahovať n -tícu zloženú zo samých nullov, je to dovoľená n -tica; môže však byť nanajvýš jedna (pretože žiadna relácia nemôže obsahovať dve rovnaké tuple).⁵⁰ Zdalo by sa, že ak dve nullové tuple neduplikačné pravidlo považuje za rovnaké, je to v rozpore s naším ohodnocovaním výrazu $\omega = \omega$. Avšak identifikácia tuplí za účelom odstránenia duplicitných tuplí je na inej úrovni abstrakcie (je v syntaktickej rovine) ako uvažovanie o tupliach pri vyhodnocovaní výrazov. Nič teda nebráni tomu, že budú existovať dve pravidlá, každé pre inú úroveň abstrakcie:

- *duplikačné pravidlo*, ktoré zakazuje, aby sa v tabuľke nachádzali dve syntakticky rovnaké tuple, a

⁵⁰ Bázová relácia takúto nullovú tupľu obsahovať nemôže vôbec, pretože ako atribút obsahuje kľúč/časť kľúča, v ktorom je null zakázaný.

- *null-substitučný princíp*, ktorý považuje dve tuple za rôzne, pokiaľ majú alebo pri nahradení nullov konkrétnymi hodnotami môžu mať rôzne hodnoty.

Nully sú považované za rovnaké tiež za účelom usporiadania (a podľa špecifikácie systému sú v usporiadaní *pred* všetkými regulárnymi hodnotami alebo *za* všetkými regulárnymi hodnotami).

3.3.3 Relačná algebra

Z dôvodu, ktorý sme uviedli vyššie (null-substitučný princíp sa neaplikuje na množinovej úrovni), nerobia operácie \cup , \cap , \setminus (zjednotenie, prienik, rozdiel) nijaký problém, vyhodnocujú sa rovnako ako v relačnom modeli bez nullových hodnôt, taktiež \times (kartézsky súčin) ostáva „nepostihnutý“ pridaním nullových hodnôt (Príklad 8).

Príklad 8: Základné relačné operácie bez zmeneného významu

R	S	$R \cup S$	$R \cap S$	$R \setminus S$																																								
<table> <tr><th>A</th><th>B</th></tr> <tr><td>ω</td><td>ω</td></tr> <tr><td>u</td><td>ω</td></tr> <tr><td>u</td><td>1</td></tr> <tr><td>ω</td><td>1</td></tr> </table>	A	B	ω	ω	u	ω	u	1	ω	1	<table> <tr><th>A</th><th>B</th></tr> <tr><td>ω</td><td>ω</td></tr> <tr><td>u</td><td>ω</td></tr> <tr><td>u</td><td>1</td></tr> </table>	A	B	ω	ω	u	ω	u	1	<table> <tr><th>A</th><th>B</th></tr> <tr><td>ω</td><td>ω</td></tr> <tr><td>u</td><td>ω</td></tr> <tr><td>u</td><td>1</td></tr> <tr><td>ω</td><td>1</td></tr> </table>	A	B	ω	ω	u	ω	u	1	ω	1	<table> <tr><th>A</th><th>B</th></tr> <tr><td>ω</td><td>ω</td></tr> <tr><td>u</td><td>ω</td></tr> <tr><td>u</td><td>1</td></tr> </table>	A	B	ω	ω	u	ω	u	1	<table> <tr><th>A</th><th>B</th></tr> <tr><td>ω</td><td>1</td></tr> </table>	A	B	ω	1
A	B																																											
ω	ω																																											
u	ω																																											
u	1																																											
ω	1																																											
A	B																																											
ω	ω																																											
u	ω																																											
u	1																																											
A	B																																											
ω	ω																																											
u	ω																																											
u	1																																											
ω	1																																											
A	B																																											
ω	ω																																											
u	ω																																											
u	1																																											
A	B																																											
ω	1																																											

Projekcia π (Príklad 9) sa správa tak, ako sa očakáva s tým, že po klasickej projekcii sa odstránia duplicitné riadky (v tom prípade, keďže sa pohybujeme na inej úrovni abstrakcie, sa neaplikuje pravidlo potenciálnej odlišnosti nullov, ako sme spomenuli vyššie).

Príklad 9: Projekcia v relačnom modeli s null hodnotami

R

A	B	C
u	ω	ω
v	1	ω
w	ω	1
x	1	ω
y	ω	1

$\pi_{BC}R = R[B, C]$

B	C
ω	ω
1	ω
ω	1

$\pi_CR = R[C]$

C
ω
1

Selekcia⁵¹ σ_F znamená vytvorenie novej tabuľky (rovnakej relačnej schémy), ktorá obsahuje iba tuple, spĺňajúce výraz F . Tento však v trojhodnotovej logike môže nadobúdať až tri rôzne hodnoty, preto sa zavádzajú dva druhy selektu.⁵² Relácia $\sigma_F(R)$ obsahuje

- pre *true* selekciu všetky tuple $\mu \in R$, ktoré sa vyhodnotia *true*;
- pre *maybe* selekciu všetky tuple $\mu \in R$, ktoré sa vyhodnotia ω .

⁵¹ presnejšie: F -selekcia

⁵² Je jasné, že ak $F(\mu) = \text{false}$, nemôže táto tupľa patriť do selekcie.

Formálne,

$$\sigma_F(R) := \{\mu \in R \mid F(\mu) = \text{true}\},$$

$$\sigma_{\omega F}(R) := \{\mu \in R \mid F(\mu) = \omega\}.$$

V tejto súvislosti sa hovorí o *maybe tupliach*, a to s cieľom zdôrazniť, že tieto tuple nemožno s istotou vylúčiť z výsledku, avšak nemožno ani s istotou povedať, že sa vo výsledku nachádzajú. Ako zdôrazníme aj neskôr, ide o tuple ležiace medzi dolnou a hornou hranicou dotazu (výrazu) F .

*Join*⁵³ znamená spojenie párov tuplí podľa špecifikovanej podmienky θ , ktorá platí medzi určitými časťami týchto tuplí. Vyhodnocovanie podmienky pre každého kandidáta (dvojicu tuplí) má pravdivostnú hodnotu T, F alebo ω . Preto sa zavádzajú dva druhy θ -joinu:

- *true* θ -join: všetky dvojice tuplí, pre ktoré sa podmienka joinu θ vyhodnotí ako *true*;
 - *maybe* θ -join: všetky dvojice tuplí, pre ktoré sa podmienka joinu θ vyhodnotí ako ω .
- Maybe* verzia joinu sa označuje symbolom ω umiestneným ako spodný index θ -symbolu resp. symbolu operátora. Ak dotaz nie je špecifikovaný inak, rozumie sa pod ním *true* verzia.

Ak chceme z nejakej relácie dostať len tie tuple, ktoré majú v konkrétnom stĺpci (teda v hodnote nejakého atribútu) nully, stačí urobiť *maybe* $=_{\omega}$ -join tejto relácie so špeciálnou unárnou reláciou, ktorá obsahuje jediný prvok, a to ľubovoľný nenullový prvok z domény konkrétneho atribútu, a potom tento výsledok odprojektovať na atribúty uvažovanej relácie.

Príklad 10: „Selekcia“ riadkov, ktoré majú ω v B-komponente. Špeciálna relácia S obsahuje jedinou tuple, 1-ticu $\langle 3 \rangle$. Voľbou relácie S a typom joinu zabezpečíme v tejto tabuľke naozaj len riadky, kde $R[B] = \omega$.

R	
A	B
u	ω
ω	2
w	1

S
C
3

$R[B =_{\omega} C]S$ čiže $R \bowtie_{B=\omega C} S$

A	B	C
u	ω	3

$(R[B =_{\omega} C]S)[A, B]$ čiže $\pi_{AB}(R \bowtie_{B=\omega C} S)$

A	B
u	ω

⁵³ presnejšie: theta-join

Prírodné spojenie teda bude tiež dvoch druhov (Príklad 11), a to podľa toho, či sa porovnanie na rovnosť vyhodnotí T alebo ω (*true* alebo *maybe*).

Príklad 11: Dva druhy (natural) joinu. Pri maybe-joinu v $\langle 2, \omega \rangle$ je druhá hodnota možno 5, preto $\langle 2, 5, 6 \rangle$, z rovnakého dôvodu v $\langle \omega, 7 \rangle$ je prvá zložka možno 3 a možno 5.

R		S	
X	Y	Y	Z
2	3	3	4
2	ω	5	6
3	5	ω	7

true-join, $R \bowtie S$			maybe-join, $R \bowtie_{\omega} S$		
X	Y	Z	X	Y	Z
2	3	4	2	ω	7
2	5	6	2	5	6
3	5	6	2	3	7
3	5	7	3	5	7

Výsledok (zodpovedania) dotazu má teda dve „časti“: *true* a *maybe* časť. Codd preto pre každú časť odpovede definuje dva druhy operátorov:

- *true*-verzie, obsahujúce všetky tuple, pre ktoré sa podmienka vyhodnotí na *true* (*true* časť odpovede na dotaz),
- *maybe*-verzie, obsahujúce všetky tuple, pre ktoré sa podmienka vyhodnotí na *maybe* (*maybe* časť odpovede na dotaz).

Prírodné, prienik medzi týmito dvoma verziami operátorov resp. časťami odpovede na dotaz je prázdny. Ako ďalej vidno, Coddove *true*-verzie operátorov majú v podstate za cieľ počítať dolnú hranicu pre externú interpretáciu dotazov, avšak *maybe*-verzie operátorov hornú hranicu nepočítajú. Hornej hranici zodpovedá výsledok, ktorý je zjednotením výsledkov po aplikovaní *true* a *maybe* verzií operátorov. Grant definuje *true* a *maybe* verzie operátorov tak, aby vyjadrovali dolnú resp. hornú hranicu pre externú interpretáciu dotazov, pozri v ďalšom, str. 56.

Codd zavádza aj ďalšie užitočné operátory, a to *outer union* a *outer theta-join*. Sú definované nasledovne. *Outer union* je zjednotenie dvoch relácií (nie nutne kompatibilných), pričom tuple jednej relácie majú nully v stĺpcoch, ktoré pôvodne táto relácia neobsahovala a dostali sa tam z druhej relácie (a opačne). *Outer join* obsahuje všetky tuple, ktoré patria do klasického joinu uvažovaných relácií, a navyše pre každú „nespárovanú“ tuple (tuple, ktorá netvorí žiaden príspevok do joinu) každej uvažovanej relácie obsahuje túto tuple s pridanými nullmi v ostatných atribútoch. (Klasický join sa preto kvôli odlíšaniu nazýva aj *inner join*.) Formálne, nech $R = R(A, B_1)$ a $S = S(B_2, C)$ sú relácie, kompatibilné na B_1 a B_2 . (Vo všeobecnosti A, B_1, B_2, C môžu byť navzájom po dvoch disjunktné množiny atribútov.)

Príklad 12: Operácie *left outer join*, *right outer join*, *outer join*

Dodáva

Dodávateľ	Pečivo
Anton	<i>perník</i>
Braňo	<i>chlieb</i>
Cyril	<i>vianočka</i>
Anton	<i>chlieb</i>
Emil	<i>rožok</i>

D

TDodáv
Anton
Braňo
Cyril
Dano

 $D[TDodáv (=)_{left} Dodávateľ]Dodáva$

Dodávateľ	Pečivo
Anton	<i>perník</i>
Braňo	<i>chlieb</i>
Cyril	<i>vianočka</i>
Anton	<i>chlieb</i>
Dano	ω

 $D[TDodáv (=)_{right} Dodávateľ]Dodáva$

Dodávateľ	Pečivo
Anton	<i>perník</i>
Braňo	<i>chlieb</i>
Cyril	<i>vianočka</i>
Anton	<i>chlieb</i>
ω	<i>rožok</i>

 $D[TDodáv (=) Dodávateľ]Dodáva$

Dodávateľ	Pečivo
Anton	<i>perník</i>
Braňo	<i>chlieb</i>
Cyril	<i>vianočka</i>
Anton	<i>chlieb</i>
ω	<i>rožok</i>
Dano	ω

Nech $R' := R \setminus T[A, B_1]$, čiže $R \setminus \pi_{A, B_1}(T)$,

$S' := S \setminus T[B_2, C]$, čiže $S \setminus \pi_{B_2, C}(T)$.

$T := R[B_1 \theta B_2]S$ čiže $R \bowtie_{\theta} S$,

Potom *outer theta-join* je definovaný nasledovne:

$$R[B_1 (\theta) B_2]S =_{df} T \cup (R' \times \langle B_2: \omega, C: \omega \rangle) \cup (\langle A: \omega, B_1: \omega \rangle \times S').$$

Nazýva sa tiež niekedy *full outer (theta) join*. Definujú sa⁵⁴ tiež operácie *right outer join* a *left outer join*, a to nasledovne:

$$R[B_1 (\theta)_{left} B_2]S =_{df} T \cup (R' \times \langle B_2: \omega, C: \omega \rangle),$$

$$R[B_1 (\theta)_{right} B_2]S =_{df} T \cup (\langle A: \omega, B_1: \omega \rangle \times S').$$

Takto sa definuje tiež *outer natural join*.

Ako z ich definícií plynie, takéto operácie generujú jeden alebo viac nullov. Vždy sa nimi rozumejú Coddove nully typu „value at present unknown“ (**un**).

⁵⁴ najmä v komerčných produktoch, porov. [DB2v4.1, 1996]

Grant v článku [Grant, 1977] týkajúcom sa nedostatočnosti Coddovej definície z r. 1975 uvádza nasledovný príklad⁵⁵: Majme reláciu $R = R(\text{Id}, \text{Meno}, \text{Vek}, \text{Mesto})$. Nech štvorica $\langle 23, \text{John}, \text{NULL}, \text{Gainesville} \rangle$

patrí do tejto relácie R . Skúsme vyhodnotiť výraz

$$23 \in S,$$

kde S je výsledkom nasledovného dotazu:

$$(R[(\text{Meno} = \text{'John'} \text{ and } \text{Vek} = 50) \text{ or } (\text{Vek} \neq 50 \text{ and } \text{Mesto} = \text{'Gainesville'})])[Id],$$

pričom prvá (veľká) hranatá zátvorka označuje selekciu a druhá projekciu. Podľa vyhodnocovacích tabuliek pre jednotlivé logické spojky dostávame, že pravdivostná hodnota tohto výrazu je

$$(1 \text{ and } \frac{1}{2}) \text{ or } (\frac{1}{2} \text{ and } 1) = \frac{1}{2},$$

avšak 23 by sa malo vyskytovať v S určite, pretože podmienky $\text{Vek} = 50$ a $\text{Vek} \neq 50$ sa navzájom vylučujú a zvyšné časti konjunkcií sú pre p pravdivé, vyhodnotia sa *true*.

Codd uvádza v [Codd, 1979] podobný príklad, ktorý sa podľa neho zdá paradoxný „len spočiatku“. Uvažujme reláciu $R = R(\text{Meno}, \text{Vek})$. Výraz

$$(R[\text{Vek} \leq 50] \cup R[\text{Vek} > 50])[\text{Meno}]$$

nemusi nutne znamenať mená *všetkých* zamestnancov. Paradox zmizne, ak interpretujeme výraz $R[\text{Vek} \leq 50]$ ako množinu všetkých tuplí v R , ktorých vek poznáme a tento je nanajvýš 50, a $R[\text{Vek} > 50]$ ako množinu všetkých tuplí v R , ktorých vek poznáme a tento je väčší ako 50. Keby sme chceli nájsť tuple ako odpoveď na dotaz „Zamestnanci, ktorých vek je viac ako 50 alebo nanajvýš 50“, musíme ho naformulovať takto:

$$(R[\text{Vek} \leq 50 \vee \text{Vek} > 50])[\text{Meno}].$$

V „no information“ interpretácii nullov tento paradoxný prípad nenastáva, vlastne samotnou povahou **ni** nullu je zamietnutý.⁵⁶

3.3.4 Problémy Coddových tabuliek

Príklad 13 ilustruje nasledovné problémy Coddovho zaobchádzania s nulovými hodnotami okrem uvedenej Grantovej poznámky:

- veľká cena *maybe* verzií dotazov (príliš veľa potenciálnych možností): pre k výskytov null hodnoty v dotaze alebo výraze, keď každý z výskytov môže nadobudnúť postupne n_1, n_2, \dots, n_k hodnôt, je treba až $n_1 \times n_2 \times \dots \times n_k$ vyhodnocovaní.⁵⁷ Pokiaľ ide o usporiadané množiny, stačia tam napr. dve (napr. prípady $\text{Vek} = 50$, $\text{Vek} \neq 50$) resp. tri (napr. prípady $\text{Vek} < 50$, $\text{Vek} = 50$, $\text{Vek} > 50$) evaluácie pre každú null hodnotu. Hoci je tento počet menší, a to najviac 2^k resp. 3^k , je to stále veľa, [Grant, 1977];
- definícia množinového obsahovania v Coddovej trojhodnotovej logike sa líši od intuitívnej predstavy: podľa null-substitučného princípu *možno* platí výraz $R \subseteq R_1$, t.j. vyhodnocuje sa na *maybe*;

⁵⁵ Tento príklad, citovaný viacerými autormi, je jemne pozmenený, aj vzhľadom ku Coddovej poznámke v článku [Codd, 1979], ako je to uvedené v ďalšom. Je použitý tiež pri popise Lipskeho prístupu z [Lipski, 1981].

⁵⁶ Porov. stať 3.6 o Zaniolovom prístupe, str. 44.

⁵⁷ Symbol \times je tu použitý ako znak násobenia.

- takto definované množinové obsahovanie nemodeluje intuitívne chápanie dynamického správania sa systému: hoci platí $R_1 = R \cup R_2$, v predošlom príklade $R \subseteq R_1$ sa vyhodnotí ako *maybe*, z čoho plynie, že po pridaní novej informácie nová (takto obohatená) databáza obsahuje staré, pôvodné informácie iba *možno*, teda ako špekuláciu, nie *skutočne*, ako skutočnosť;
- na množinové operácie sa nevzťahujú čo i len základné vlastnosti množinovej algebry, napr. pre relácie R, R_1 :

$$(R \cap R_1) \subseteq R;$$

$$(R \cup R_1) \supseteq R;$$

$$R = R;$$

- niektoré tautológie sú vyhodnotené ako *maybe* namiesto *true* ($R = R$).

Príklad 13: Paradoxy Coddovho zaobchádzania s nullmi

R	
X	Y
ω	1
6	2

R_1	
X	Y
ω	1
6	2
7	2

R_2	
X	Y
7	2

V niektorých prípadoch by sa dalo týmto paradoxom predísť, napr. definovaním rozšírenej rovnosti \equiv , ako sme uviedli vyššie. V prípade, že porovnávame jeden a ten istý riadok relácie

obsahujúci null sám so sebou, dostávame odpoveď „možno“, čo však nie je žiadúce. Ved' intuitívne, ak sa nachádza v relácii nejaká tupľa, potom iste sa rovná sama sebe. Prečo by to nemalo platiť aj o tupliach s nullmi? Stačí v takom prípade použiť \equiv namiesto $=$.

Všeobecne pri porovnávaní je potrebné pred *sémantickým porovnaním* (porovnanie na základe hodnôt) urobiť najprv *syntaktické porovnanie* (porovnanie na úrovni symbolov). Toto sa dá zabezpečiť jednoduchým parserom výrazov. Keby sme vo vyššie uvedených príkladoch uskutočnili najskôr syntaktické porovnanie, a až potom (v prípade nezisteného výsledku) sémantické, na množinové operácie by sa vzťahovali aspoň základné vlastnosti množinovej algebry, pretože ich platnosť by sa zabezpečila už pri syntaktickom porovnaní. Podobne aj výrazy $\mu = \mu$ alebo $\mu[A] < \mu[A]$ budú vždy najprv (a definitívne) vyhodnotené na syntaktickej úrovni (v prvom prípade vždy *true*, v druhom vždy *false*).

V nasledujúcom príklade (Príklad 14), kde R aj S sú relačné schémy nad (Dodávateľ, Pečivo), oba výrazy $R = R$, $R = S$ sa vyhodnotia *maybe* pri použití klasického $=$ na prvkoch alebo na *true* pri použití \equiv na porovnávanie prvkov. Pritom sa však pri zlepšení informácie môže stať, že $R \neq S$, teda $R = S$ sa vyhodnotí na *false* – stačí, aby sme do každej tabuľky zadali iného dodávateľa. Na druhej strane, akokoľvek sa špecifikuje Dodávateľ v R , vždy bude platiť $R = R$ (malo by sa vyhodnotiť ako *true*).

Príklad 14: Ako odlišiť $R = R$ od $R = S$ možno? Použitie \equiv namiesto $=$ problém nevyrieši.

R	
Dodávateľ	Pečivo
ω	<i>perník</i>

S	
Dodávateľ	Pečivo
ω	<i>perník</i>

Je samozrejmé, že – z dôvodu neduplikačného pravidla – riadok $\langle \text{NULL}, \text{perník} \rangle$ v relačnej schéme R resp. S nemôže byť v nej uvedený dvakrát, aj keby perník dodávali

dvaja rôzni (zatiaľ nepomenovaní) dodávateľia. Ale aj v prípade, že by jeden riadok reprezentoval viacero neznámych hodnôt, sa $R = R$ vyhodnotí ako *true*. Čo v takomto prípade znamená *update* tejto tuple? Jej dočasné rozdvojenie, pričom v jednej kópii sa upresní dodávateľ („už je známe jeho meno“) a druhá kópia ostane nezmenená („ešte sa nepodarilo zistiť názov druhého dodávateľa“)? Ako vyjadriť dodatočnú informáciu, že perníky dodávajú *dvaja* dodávateľia, hoci ani jednému z nich neviem určiť meno? Týmito a ďalšími otázkami, ktoré vedú k úvahám o rozlišovaní null hodnôt, sa zaoberá časť 3.4.

Predchádzajúce príklady ilustrujú všeobecný problém pri práci s neúplnou informáciou, totiž dotazy, v ktorých sa nenachádzajú iba od seba nezávislé tabuľky (napr. ak R aj S sú dve rôzne tabuľky, dotaz tvaru „ $f(R) = g(S)$ “, kde f, g sú relačné výrazy), ale dotazy, používajúce tú istú tabuľku viackrát (napr. dotaz tvaru „ $f(R) = g(R)$ “); porov. napr. [Imieliński–Lipski, 1981], [Lipski, 1983a].

[Lipski, 1979] poznamenáva, že je prirodzené obmedziť null-substitučný princíp požiadavkou, že rozličné výskyty tej istej premennej sú vždy nahradené hodnotou rovnakou pre všetky tieto výskyty. Lenže pridanie tejto reštrikcie spôsobí, že trojhodnotová logika už nie je skutočne funkcionálna (teda pravdivostná hodnota výrazu *nie je* určená pravdivostnými hodnotami jeho podvýrazov, porov. spomínaný Coddov „spočiatku-paradox“).

Na základe definícií súvisiacich s reprezentačným systémom dospeli [Imieliński–Lipski, 1984] k nasledovným charakterizáciám Coddových tabuliek:

- PS –výrazy sa dajú korektne vyhodnocovať.⁵⁸
- PSU –výrazy ani PXE –výrazy sa nedajú korektne vyhodnocovať.
- PJ –výrazy sa nedajú korektne vyhodnocovať.

Uvedme dôvod, pre ktorý sa PJ –výrazy nedajú korektne vyhodnocovať.

Nech

$$f(R) = \pi_{AC}(R) \bowtie \pi_B(R),$$

$$T = \{\langle a_1, \omega, c_1 \rangle, \langle a_2, \omega, c_2 \rangle\},$$

pričom predpokladáme, že $a_1 \neq a_2$ a $c_1 \neq c_2$. Ľahko vidno, že $f(Rep(T)) \equiv_P Rep(T)$. Totiž keď budeme výrazom g projektovať na atribúty obsahujúce A resp. C (či obidva naraz),

$$\cap \{g(t) \mid t \in Rep(T)\} \text{ aj } \cap \{g(t) \mid t \in f(Rep(T))\}$$

budú obsahovať iba $\{\langle a_1 \rangle, \langle a_2 \rangle\}$ resp. $\{\langle c_1 \rangle, \langle c_2 \rangle\}$ (či pri oboch $\{\langle a_1, c_1 \rangle, \langle a_2, c_2 \rangle\}$). Pri projekcii na množinu atribútov, ktorá neobsahuje ani A ani C , obidve množiny $\cap g(Rep(T))$ aj $\cap g(f(Rep(T)))$ budú prázdne.

Pritom však neplatí $f(Rep(T)) \equiv_{PJ} Rep(T)$. Stačí vziať PJ –výraz g , pričom položíme

$$g(R) = \pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R)).$$

Máme

$$Rep(T)^g = \{\langle a_1, c_1 \rangle, \langle a_2, c_2 \rangle\},$$

hoci na druhej strane

$$f(Rep(T))^g = \{\langle a_1, c_1 \rangle, \langle a_2, c_2 \rangle, \langle a_1, c_2 \rangle, \langle a_2, c_1 \rangle\}.$$

⁵⁸ Práve to je dôvod, prečo ich autori v [Imieliński–Lipski, 1981] nazvali PS –tabuľky.

3.4 Indexované (značkové) nully

Problém, ako naznačiť, že dve nullové hodnoty v tabuľke, hoci ich nepoznáme, sú rovnaké alebo rôzne, riešia V-tabuľky z [Imieliński–Lipski, 1984] a v nich používané *značkové nully* (angl. *marked nulls*). [Reiter, 1986], ktorý taktiež používa takýto koncept nullov, ich nazýva *indexované nully* a keďže jeho prístup vychádza z popisu databázy založenom na prevode do prvorádovej logiky v podobe logických formúl, používa pre ne tiež označenie *Skolemove konštanty* z formálnej logiky.

3.4.1 V-tabuľky

V-tabuľky sú rozšírením Coddových tabuliek. Namiesto jednej všeobecnej null hodnoty sa v nich však vyskytuje ľubovoľné množstvo značkových nullov, pričom pre rôzne atribúty sú množiny značkových nullov disjunktné. (Táto požiadavka nie je nevyhnutná, je však logická, pretože značkový null znamená neznámu hodnotu *daného atribútu*.) Formálne, pre atribút A a jeho atribútovú doménu (nosič) D_A , množina premenných je ľubovoľná množina $Var(A)$ taká, že platí

$$Var(A) \cap D_A = \emptyset$$

a ak navyše berieme do úvahy požiadavku disjunktnosti množín, pre $D_A \neq D_B$ platí

$$Var(A) \cap Var(B) = \emptyset.$$

V-tupli $\langle a, x \rangle$, kde $x \in Var(B)$, zodpovedá formula tvaru

$$(\exists b \in B)(R(a, b)).$$

Vo všeobecnosti V-tabuľke zodpovedá formula, ktorá je existenciálne kvantifikovaná pre každú použitú premennú a obsahuje konjunkcie jednotlivých atomických formúl zodpovedajúcich riadkom tabuľky.

Napokon poznamenajme, že názov „V-tabuľky“ pochádza z iného pomenovania pre značkové nully, a to *premenné* (angl. *variables*). V staršej práci [Imieliński–Lipski, 1981] ich autori nazývali PJ-tabuľky práve z dôvodu, že dokážu korektne vyhodnocovať PJ-výrazy, ako je uvedené v nasledujúcom.

Nasledujúce výsledky z [Imieliński–Lipski, 1984] charakterizujú V-tabuľky:

- $PS^+ UJ$ -výrazy sa dajú korektne vyhodnocovať, rovnako tak aj $PS^+ UX$ -výrazy a $PS^+ UJR$ -výrazy.
- PS -výrazy sa nedajú korektne vyhodnocovať.

Tento druhý výsledok je možno prekvapujúci, pretože V-tabuľky sa zdajú „silnejšie“ ako Coddove tabuľky.

Príklad 15 (Príklad 14 modifikovaný): Perník v tabuľke R dodávajú dvaja neznámi dodávatelia, navyše $S \subseteq R$

R	
Dodávateľ	Pečivo
x	<i>perník</i>
y	<i>perník</i>

S	
Dodávateľ	Pečivo
x	<i>perník</i>

Manipulácia s V–tabuľkami je rovnaká, ako keby každý značkovaný null zastupoval konkrétnu hodnotu – teda presne taká istá ako manipulácia s relačnými tabuľkami bez nullových hodnôt. Takýto spôsob manipulácie s premennými, akoby boli správnymi výrazmi atribútových domén, sa nazýva *naivné vyhodnocovanie* (alebo *naivná evaluácia*) a tabuľky sa nazývajú *naivné tabuľky*⁵⁹. V–tabuľky však, podobne ako Coddove tabuľky, majú vážny nedostatok, uvedený vyššie: nedokážu správne vyhodnotiť všetky relačné výrazy. Stačí uvažovať selekciu $\sigma_F(T)$, kde

$$F \equiv (A = a_1 \wedge B = b) \vee (A = a_2 \wedge B \neq b)$$

nad nasledujúcou tabuľkou T (predpokladáme, že prvky a_1 a a_2 sú rôzne). Neexistuje V–tabuľka, reprezentujúca $f(\text{Rep}(T))$.

Tabuľka 3: Tabuľka T

A	B	C
a_1	y	c
a_2	y	c

3.4.2 C–tabuľky

Spomínaného nedostatku V–tabuliek správne vyjadriť ľubovoľný relačný výraz si boli Imieliński a Lipski vedomí, preto nesústredovali svoju pozornosť iba na správne narábanie s relačnými operátormi za prítomnosti null hodnôt, ale práve na narábanie s celými relačnými výrazmi.⁶⁰ Rozšírenie V–tabuliek, *C–tabuľky* (*conditional tables*), odstraňuje neúplnosť V–tabuliek a toto správne zaobchádzanie s relačnými výrazmi zabezpečuje.

C–tabuľka je V–tabuľka s pridaným stĺpcom *con* (z angl. *condition*; pre ľubovoľnú tabuľku má tento stĺpec rovnaký atribút), ktorý obsahuje *podmienky* z množiny \wp . Množina podmienok \wp je najmenšia množina obsahujúca nasledovné:

- (1) atomické podmienky tvaru $(x = a)$, $(x = y)$, kde $x, y \in \text{Var}(A)$, $a \in D_A$;
- (2) konštanty *true*, *false*;
- (3) ak ϕ a ψ sú podmienky, potom aj $\neg\phi$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$ sú podmienky.

Ak podmienka neobsahuje \neg , potom sa nazýva *pozitívna*. Namiesto $\neg(x = y)$ píšeme pre skrátenie $x \neq y$. Niekedy o časti C–tabuľky bez stĺpca *con*, ktorá zodpovedá V–tabuľke, hovoríme ako o „naivnej časti“ C–tabuľky.

C–tupľá t (riadok C–tabuľky) je zobrazenie na $X \cup \{\text{con}\}$ také, že $t[X]$ je V–tupľá (teda riadok V–tabuľky) a $t(\text{con}) \in \wp$. Podmienky sú *ekvivalentné* vtedy, ak sú rovnaké pre všetky valuácie. Nahrádzanie podmienky ekvivalentnou, ktorá je v istom zmysle jednoduchšia, sa nazýva *normalizácia*. C–tabuľke zodpovedá formula tvaru

$$c_1 \Rightarrow R(a_1) \wedge c_2 \Rightarrow R(a_2) \wedge \dots c_n \Rightarrow R(a_n),$$

pričom c_i je podmienka a a_i hodnota (pre zjednodušenie jediného) atribútu i –teho riadka C–tabuľky.

⁵⁹ porov. [Lipski, 1983b]

⁶⁰ Ako uvádza [Lipski, 1983a] či [Imieliński–Lipski, 1984], málo autorov sa zaoberá vplyvom nullových hodnôt nielen na jednotlivé relačné operátory, ale tiež celých relačných výrazov, napr. [Biskup, 1981].

Príklad 16: Postupné vyhodnocovanie výrazu $f(R) = \sigma_{C=c}(\pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R)))$ nad T

$$T$$

A	B	C	con
a	b	z	$z \neq c$
a	y	c	$y \neq b$
x	b	c	$x \neq a$

$$U = \pi_{AB}(R) \bowtie \pi_{BC}(R)$$

A	B	C	con
a	b	z	$z \neq c \wedge z \neq c \wedge b = b$
a	b	c	$z \neq c \wedge y \neq b \wedge b = y$
a	b	c	$z \neq c \wedge x \neq a \wedge b = b$
a	y	z	$y \neq b \wedge z \neq c \wedge y = b$
a	y	c	$y \neq b \wedge y \neq b \wedge y = y$
a	y	c	$y \neq b \wedge x \neq a \wedge y = b$
x	b	z	$x \neq a \wedge z \neq c \wedge b = b$
x	b	c	$x \neq a \wedge y \neq b \wedge b = y$
x	b	c	$x \neq a \wedge x \neq a \wedge b = b$

U po ekvivalentných úpravách⁶¹

A	B	C	con
a	b	z	$z \neq c$
a	b	c	$z \neq c \wedge x \neq a$
a	y	c	$y \neq b$
x	b	z	$x \neq a \wedge z \neq c$
x	b	c	$x \neq a$

$$W = \pi_{AC}(U)$$

A	C	con
a	z	$z \neq c$
a	c	$(z \neq c \wedge x \neq a) \vee y \neq b$
x	z	$x \neq a \wedge z \neq c$
x	c	$x \neq a$

$$\text{napokon } f(R) = \sigma_{C=c}(W)$$

A	C	con
a	z	$z \neq c \wedge z = c$
a	c	$((z \neq c \wedge x \neq a) \vee y \neq b) \wedge z = c$
x	z	$x \neq a \wedge z \neq c \wedge z = c$
x	c	$x \neq a \wedge z = c$

⁶¹ V pôvodnom článku je tretí riadok tabuľky U uvedený chybné. Správne napr. v [Paredaens, 1989].

a po ekvivalentných úpravách (pretože v riadkoch, ktoré v tabuľke zostali, je podmienka $z = c$ ekvivalentná s *true*)

A	C	con
a	c	$y \neq b$
x	c	$x \neq a$

Príklad 17: Reprezentácia čiastočnej znalosti C–tabuľkou.

R		
Dodávateľ	Pečivo	con
x	perník	$x \neq y$
y	chlieb	$x \neq y$

S		
Dodávateľ	Pečivo	con
x	perník	$x \neq \text{Anton} \wedge x \neq \text{Braňo}$

R: Nieкто dodáva *perník* a nieкто *chlieb*; viem však, že sú to odlišní dodávateľia.

S: Nieкто dodáva *perník*. Hoci neviem, kto je to, viem, že to nie je ani Anton ani Braňo.

Pri istom druhu interpretácie domén (napr. keď „vieme“, že daný stĺpec má doménový obor reálne čísla) teoreticky potrebujeme nekonečne veľa značkových nullov.

Našťastie, vždy uvažujeme *konečné* množiny tuplů (konečné databázy), hoci aj v nich množstvo premenných môže nekontrolovateľne (exponenciálne) rásť. Stačí si uvedomiť, že ak x označuje neznámu celočíselnú hodnotu, pri zadaní dvojnásobku tejto hodnoty $2.x$ už reprezentuje inú hodnotu, rovnako neznámu (vieme však o nej, že je *párna*).

Nasledujúce výsledky z [Imieliński–Lipski, 1984] charakterizujú správne zaobchádzanie s relačnými výrazmi v C–tabuľkách:

- *PSUJ*–výrazy sa dajú korektne vyhodnocovať,
- *PSUJRD*–výrazy sa dajú korektne vyhodnocovať pod CWA interpretáciou.

Teda syntaktické vyhodnocovanie dotazov nad C–tabuľkami je sémanticky správne. Toto je veľká prednosť C–tabuliek.

Nasledujúci výsledok platí pre Coddove tabuľky, V–tabuľky aj C–tabuľky:

- pre ľubovoľnú množinu $\Omega \subseteq PSS^+UJR$ platí: Ω –výrazy sa dajú korektne vyhodnocovať pod OWA práve vtedy, keď sa dajú správne vyhodnocovať pod CWA interpretáciou.

Grahne jemne rozširuje pojem C–tabuľky tým, že pôvodné podmienky nazýva *lokálne* (týkajú sa iba jedného riadku) a pridáva k tabuľke *globálnu podmienku*, teda podmienku, ktorá platí ako integritné ohraňenie pre celú tabuľku (porov. [Abiteboul–Grahne, 1985]). Globálna podmienka sa vlastne aplikuje na každú tupľu v konjunkcii s lokálnou podmienkou tejto tuple. Stačí teda nahradiť všetky lokálne podmienky novými – ktoré vzniknú konjunkciou pôvodnej podmienky s globálnou podmienkou –, čím odstránime globálnu podmienku. Potom už ľahko vidno, že aj na modifikované C–tabuľky s globálnymi podmienkami sa vzťahujú výsledky Imieliňského a Lipskeho.

Každá tupľa, ktorá nemá v stĺpci *con* pravdivostnú hodnotu *true*, je pokladaná za *maybe* informáciu. Naviac, keby sme množinu podmienok definovali tak, že okrem konštánt (pravdivostných hodnôt) *true* a *false* by obsahovala aj konstantu *maybe*, dostaneme model pre C–tabuľky, ktorý je schopný reprezentovať aj *maybe* informáciu priamo. Nevýhodou

tejto reprezentácie je, že sa nedá nijakým spôsobom naznačiť, ktorý komponent je zdrojom „maybe charakteru“ informácie. Taktiež keď tupľa v stĺpci *con* obsahuje disjunkciu, ide v podstate o disjunktívnu informáciu. Podrobnejšie o týchto ďalších vlastnostiach C–tabuliek pozri časť 4.4, str. 62 a nasl.

O funkčných závislostiach pozri časť 5.4.

Čo sa týka *implementácie*, C–tabuľky sú kritizované jednak kvôli náročnosti reprezentácie podmienok (stĺpec *con*), dvak kvôli zložitosti vyhodnocovania dotazov. V tejto súvislosti sú možné nasledovné riešenia:⁶²

1. Databáza je uchovávaná ako C–tabuľka, avšak pre účely dotazovania je použitá iba „naivná časť“ tabuliek. Teda odpovede na dotazy sú iba aproximáciami. Celá informácia C–tabuliek je použitá kvôli aktualizáčnym operáciám (porov. časť 3.7 na str. 53). Preto obsah databázy sa čo najviac podobá realite (hoci zodpovedanie na dotazy túto „presnosť“ môže skresľovať).
2. Ak sa predpokladá, že sa podmienky vyskytujú zriedka, teda keď slúžia najmä na vysporiadanie sa s výnimkami, potom argument o zložitosti C–tabuliek nie je namieste. Fyzická implementácia totiž poskytne všetky výhody C–tabuliek, a pritom narábanie s väčšinou informácií sa môže realizovať nástrojmi pre V–tabuľky.

3.4.3 Reiterov prístup

Podľa Reitera rôzne prístupy k nullom a manipuláciám s nimi v databázach stroskotali kvôli sémantickým ťažkostiam pochádzajúcim z nevhodného teoretického základu databázovej teórie. Už [Gallaire–Minker, 1978] zhŕňajú názory na vzťah medzi matematickou logikou a databázovou teóriou. Analógia relácie (tabuľky) a predikátu je známa. Ako sme uviedli vyššie (str. 14), riadok tabuľky pritom zodpovedá jednému prípadu, v ktorom predikát platí. Nullové hodnoty však spôsobujú, že táto analógia nie je priamočiara. Zatiaľ čo pred Reiterom sa databáza pokladala za druh modelu pre množinu výrazov, čo pri použití nullových hodnôt vedie k viachodnotovým logikám⁶³, Reiter vo svojich článkoch zastáva názor, že databázu treba chápať ako *teóriu*, teda ako množinu logických formúl. Navyše, Reiter zdôrazňuje, že je potrebné pozorne skúmať aj implicitné predpoklady, ktoré ležia v pozadí relačného modelu. Prvému z predpokladov (CWA) sme sa venovali v samostatnej časti 2.7 (str. 16), druhý len spomenieme. Ide o *predpoklad jednoznačného mena* (angl. *unique name assumption*, skratka UNA). Toto je predpoklad, že jednotlivé objekty v databáze, napr. Anton a Braňo alebo *chlieb* a *perník*, sú navzájom rôzne.

Definujme teda prvorádové databázy podľa Reitera.

Relačný jazyk je prvorádový jazyk zložený zo slov (slovo sa nazýva dobre vytvorená formula⁶⁴, v ďalšom „wff“ alebo iba „formula“) nad abecedou symbolov pre premenné, konštanty, predikáty, interpunkciu, logické konštanty a kvantifikátory, pričom abeceda neobsahuje funkčné symboly iné ako nulárne konštanty. V abecede sa nachádza aj špeciálny binárny predikát *E* (equality), ktorý reprezentuje rovnakosť objektov. Abeceda

⁶² porov. [Abiteboul–Grahne, 1985], str. 11

⁶³ tento spôsob zvolil napr. aj [Biskup, 1981], vid' nižšie

⁶⁴ angl. *well-formed formula*, odtiaľ skratka wff

tiež obsahuje nenulovú množinu unárnych predikátov, ktoré sa nazývajú *jednoduché typy*, pritom však nie všetky unárne predikáty musia byť jednoduché typy.

Term relačného jazyka je premenná alebo konštanta z abecedy. Ak P je n -árny predikátový symbol a t_1, \dots, t_n sú termy, potom $P(t_1, \dots, t_n)$ je *atomická formula*. Ak t_1, \dots, t_n sú konštanty, nazýva sa táto atomická formula *uzavretá* (angl. *ground*).

Namiesto niektorých bežne používaných formúl sa používajú nasledovné skratky⁶⁵:

$$(x/\tau)(W) \text{ namiesto } (x)(\tau(x) \Rightarrow W)$$

$$(Ex/\tau)(W) \text{ namiesto } (Ex)(\tau(x) \wedge W),$$

kde τ je jednoduchý typ. Pod $(x/\tau)(W)$ sa rozumie „pre všetky x ktoré sú τ platí W “ a $(Ex/\tau)(W)$ značí „existuje x ktoré je τ , že platí W “.

Konečná podmnožina R formúl relačného jazyka sa nazýva *rozšírená relačná teória*, je definovaná platnosťou nasledovných podmienok:

- (1) Pre každý n -árny predikát P z abecedy odlišný od E (pritom však obsahujúci jednoduché typy) R obsahuje *jedinú* formulu tvaru

$$(\mathbf{x})P(\mathbf{x}) \equiv E(\mathbf{x}, \mathbf{c}_1) \vee \dots \vee E(\mathbf{x}, \mathbf{c}_r),$$

kde $\mathbf{x} = x_1, \dots, x_n$ je postupnosť rôznych premenných, $(\mathbf{x})P(\mathbf{x})$ je skratka za $(x_1) \dots (x_n)P(x_1, \dots, x_n)$ a \mathbf{c}_i sú n -tice konštánt z abecedy.

V prípade, že $r = 0$, zodpovedajúca formula je $(\mathbf{x})\neg P(\mathbf{x})$; nazýva sa *rozširujúci axióm* pre P v R .

- (2) R obsahuje nula alebo viac *axióm jednoznačnosti*⁶⁶ tvaru $\neg E(c, c')$ pre rôzne konštanty c, c' z abecedy.

- (3) Nič iné nie je v R .

(Prvorádová) *databáza* je množina uzavretých formúl (wff).

Dotaz je ľubovoľný výraz tvaru

$$\langle x_1/\tau_1, \dots, x_n/\tau_n \mid W(x_1, \dots, x_n) \rangle,$$

kde x_1, \dots, x_n sú navzájom rôzne premenné z abecedy, τ_1, \dots, τ_n sú jednoduché typy z abecedy a W je wff s voľnými premennými x_1, \dots, x_n , ktorá obsahuje jedine typované kvantifikátory. Aj prípad $n = 0$ je prípustný; v takom prípade má dotaz tvar $\langle \mid W \rangle$, kde W nemá voľné premenné. Neformálne, dotaz $\langle \mathbf{x}/\tau \mid W(\mathbf{x}) \rangle$ má značiť množinu všetkých n -tíc $\mathbf{x} = x_1, \dots, x_n$ takých, že každé x_i je typu τ_i a databáza spĺňa $W(\mathbf{x})$.

n -tica $\mathbf{c} = \langle c_1, \dots, c_n \rangle$ konštánt z abecedy sa nazýva *odpoveď* na dotaz $\langle \mathbf{x}/\tau \mid W(\mathbf{x}) \rangle$ vzhľadom k prvorádovej databáze DB, ak všetky formuly $\tau_i(c_i)$, $i = 1, \dots, n$, a $W(\mathbf{c})$ sú pravdivé v každom modeli prvorádovej databázy DB v predikátovom kalkule prvého rádu s rovnosťou. [Reiter, 1986] dokázal, že táto definícia je ekvivalentná nasledovnej: n -tica $\mathbf{c} = \langle c_1, \dots, c_n \rangle$ konštánt z abecedy je *odpoveďou* na dotaz $\langle \mathbf{x}/\tau \mid W(\mathbf{x}) \rangle$ vzhľadom k prvorádovej databáze DB, ak platí

- (1) $DB \vdash \tau_i(c_i), i = 1, \dots, n,$
- (2) $DB \vdash W(\mathbf{c}),$

⁶⁵ Ide o typovo ohraničené kvantifikátory. Platnosť kvantifikátorov môže byť ohraničená jedine príslušnosťou k typu, nie ľubovoľným predikátom.

⁶⁶ angl. *unique name axiom*, ktorý formalizuje UNA

kde $S \vdash w$ značí dokázateľnosť formuly w v predikátovom kalkule prvého rádu s rovnosťou s predpokladmi (hypotézami) S . V článku [Reiter, 1986] je ukázaný spôsob, ako tieto odpovede získať, pričom záujem o konštrukciu odpovede vedie k definícii algebraických operátorov a napokon k algoritmu na vyhodnocovanie dotazov založenom na týchto operátoroch.

Rozšírené relačné teórie nerobia žiaden formálny rozdiel medzi nullmi, ktoré označujú „neznáme“ prípady, a „obyčajnými“ konštantami, ktoré označujú regulárne hodnoty. (Jediným faktorom, ktorý odlišuje nullové hodnoty od regulárnych hodnôt, je neprítomnosť axiómu jednoznačnosti pre ω , teda neexistujú formuly $\neg E(c, \omega)$ pre konštanty c .) A tak Reiterov prístup nezachytáva celú šírku rôznych nullových hodnôt, je však cenný v spôsobe pohľadu na neznáme „údaje“: v ponímaní rozšírenej relačnej teórie sú neznáme hodnoty zapríčinené neprítomnosťou nejakých axiém jednoznačnosti. Indexovanú nullovú hodnotu dostaneme tak, že odstránime existenčnú kvantifikáciu z prvorádovej formuly a nahradíme ju *Skolemovou konštantou* (v skutočnosti pre Reiterovu teóriu to znamená pridanie axiém jednoznačnosti pre nully, teda formúl $\neg E(\omega_1, \omega_2)$ pre nully ω_i).

Príklad 2 ilustruje Reiterov spôsob zápisu databázy ako teórie. Príklad 18 tento spôsob reprezentácie rozširuje – ukazuje, ako reprezentovať neúplnú informáciu zavedením vyššie spomínaného nullu ako Skolemovej konštanty – a ďalej ho využijeme na ilustráciu rozdielov CWA a OWA interpretácie.

Príklad 18: Reiterov spôsob reprezentácie čiastočnej informácie

Poznatok

„Niektorí dodávajú *perník* a niektorí *chlieb*; viem však, že to nie je ten istý dodávateľ.“

(Príklad 2, R) je zapísaný takto:

$(Ex/\text{Dodávateľ})(Ey/\text{Dodávateľ})\text{Dodáva}(x, \text{perník}) \wedge \text{Dodáva}(y, \text{chlieb}) \wedge \neg E(x, y)$

pričom zodpovedá formule

$$\text{Dodávateľ}(\omega_1) \wedge \text{Dodávateľ}(\omega_2) \wedge \text{Dodáva}(\omega_1, \text{perník}) \wedge \\ \text{Dodáva}(\omega_2, \text{chlieb}) \wedge \neg E(\omega_1, \omega_2)$$

po odstránení existenčného kvantifikátora.

Podobne poznatok

„Niektorí dodávajú *perník*, ale neviem, kto je to.

Viem však, že to nie je ani Anton ani Braňo.“

(Príklad 2, S) formálne zapísaný vyzerá nasledovne:

$(Ex/\text{Dodávateľ})\text{Dodáva}(x, \text{perník}) \wedge \neg E(x, \text{Anton}) \wedge \neg E(x, \text{Braňo})$

a po eliminácii existenčného kvantifikátora

$$\text{Dodávateľ}(\omega) \wedge \text{Dodáva}(\omega, \text{perník}) \wedge \neg E(\omega, \text{Anton}) \wedge \neg E(\omega, \text{Braňo})$$

Niektoré pozitívne dotazy zapísané formálne:

- „Dodávatelia, ktorí dodávajú viac ako jedno pečivo.“
 $\langle x/\text{Dodávateľ} \mid (Ey/\text{Pečivo})(Ez/\text{Pečivo})\neg E(y, z) \wedge \text{Dodáva}(x, y) \wedge \text{Dodáva}(x, z) \rangle$
- „Dodávatelia, ktorí dodávajú všetky druhy pečiva.“
 $\langle x/\text{Dodávateľ} \mid \text{Pečivo}(y) \Rightarrow \text{Dodáva}(x, y) \rangle$
- „Dvojice navzájom rôznych dodávateľov, ktorí dodávajú rovnaké druhy pečiva.“
 $\langle x/\text{Dodávateľ}, y/\text{Dodávateľ} \mid \neg E(x, y) \wedge (Ez/\text{Pečivo})\text{Dodáva}(x, z) \equiv \text{Dodáva}(y, z) \rangle$

Na vyhodnotení negatívneho dotazu „Dodávatelia, ktorí nedodávajú chlieb“ ukážeme rozdiel medzi CWA a OWA. Dotaz formálne vyzerá

$$Q := \langle x/\text{Dodávateľ} \mid \neg \text{Dodáva}(x, \text{chlieb}) \rangle.$$

Odpoveď v OWA (označenie $\llbracket Q \rrbracket_{\text{OWA}}$) je \emptyset (porov. [Reiter, 1978]). Intuitívne by sme chceli, aby odpoveď bola $\{\text{Cyril}, \text{Dano}\}$, teda

$$|\text{Dodávateľ}| \setminus \llbracket \langle x/\text{Dodávateľ} \mid \text{Dodáva}(x, \text{chlieb}) \rangle \rrbracket_{\text{OWA}}.$$

Príčina tohoto výsledku je, že z našej prvorádovej databázy sa nedá odvodiť, že

$$\neg \text{Dodáva}(\text{Cyril}, \text{chlieb})$$

je pravdivý. Čo sa týka formálnej logiky, pravdivostná hodnota výrazu

$$\text{Dodáva}(\text{Cyril}, \text{chlieb})$$

je neznáma.

Odpoveď v CWA implicitne predpokladá existenciu nasledovných znalostí:

$$\begin{aligned} |\neg \text{Dodáva}| = \{ & \langle \text{Anton}, \text{vianočka} \rangle, \langle \text{Braňo}, \text{perník} \rangle, \langle \text{Braňo}, \text{vianočka} \rangle, \\ & \langle \text{Cyril}, \text{perník} \rangle, \langle \text{Cyril}, \text{chlieb} \rangle, \langle \text{Dano}, \text{perník} \rangle, \\ & \langle \text{Dano}, \text{chlieb} \rangle, \langle \text{Dano}, \text{vianočka} \rangle \}. \end{aligned}$$

Je zrejmé, že množstvo negatívnych faktov značne presahuje množstvo pozitívnych faktov, čo je aj dôvodom, prečo sa v drvinej väčšine reálnych aplikácií negatívna informácia nechováva. Potom v našom príklade $\llbracket Q \rrbracket_{\text{CWA}} = \{\text{Cyril}, \text{Dano}\}$.

[Vardi, 1986] sa zaoberal dotazovaním v takomto modeli databáz. Jeho výsledky rozširujú Reiterove výsledky z jeho práce [Reiter, 1986], lenže nie sú povzbudivé: cena, ktorú zaplatíme za elegantný spôsob reprezentovania neúplnej informácie v databázach pohľadom na databázu ako teóriu, je príliš vysoká. Vyhodnocovanie dotazov je vo všeobecnosti príliš pomalé. Zatiaľ čo prvorádové dotazy nad klasickými relačnými databázami môžu byť vyhodnotené v logaritmickom priestore, rovnaká množina dotazov nad logickými databázami je co-NP-úplná (jej doplnok je NP-úplná množina). Ako Reiter, tak Vardi navrhli algoritmy rovnakej zložitosti ako pre relačné databázy. (Reiterov algoritmus je založený na rekurzívnej dekompozícii zložitých dotazov do jednoduchších vhodnou množinovo-teoretickou operáciou na dotazoch.) Tieto algoritmy sú iba aproximačné, nedávajú celú odpoveď na dotaz, len jej aproximáciu. Sú *korektné*, čo znamená, ich výsledok je vždy podmnožinou správnej odpovede. Pre databázy bez neúplnej informácie resp. databázy s nullmi spolu s dotazmi bez negácie (pozitívne dotazy) sú tieto algoritmy aj *úplné*, teda správna odpoveď je zahrnutá v ich výsledku.

3.5 Biskupove nully

[Biskup, 1981] rozpracoval myšlienku pochádzajúcu od Vassilioua, a to použiť usporiadanie na zvýšenie čiastočného charakteru neúplnej informácie. Najskôr však popíšme Biskupove nully. Okrem nullu, použitého vo vyššie popísaných prístupoch (vyjadrujúci „existing but unknown“ interpretáciu; Biskup ho označuje symbolom existenčného kvantifikátora \exists), používa aj null s významom „hodnota je ľubovoľná“, „nestarám sa o hodnotu, môže tam byť akákoľvek“ označenie symbolom všeobecného kvantifikátora \forall . Teda jeho doménou je množina *regulárnych hodnôt* V spolu s dvoma symbolmi pre nully, teda množina $V^* := V \cup \{\exists, \forall\}$. Relácia nad takýmito obohatenými doménami sa nazýva *zovšeobecnená relácia*⁶⁷ (teda relácia s Biskupovými nullovými

⁶⁷ [Biskup, 1982a] ju nazýva *interná relácia* (angl. *internal relation*), aby zdôraznil, že ide o internú reprezentáciu podľa Lipskeho.

hodnotami). Jednotlivé prvky zovšeobecnenej relácie sa nazývajú *zovšeobecnené tuple*. Sú to tuple nadobúdajúce hodnoty z množiny V^* . Tupľa s regulárnymi hodnotami znamená uzavretý údaj (bez kvantifikátorov) v predikátovej logike.⁶⁸ Zápis o (zovšeobecnenej) tupli μ

$$\mu := \langle \forall, a, \exists, \forall, b \rangle \in R$$

znamená

„existuje x_3 také, že pre všetky x_1 a pre všetky x_4 platí $\langle x_1, a, x_3, x_4, b \rangle$ “.

Všetky existenčné kvantifikátory pritom predchádzajú všetky všeobecné kvantifikátory.⁶⁹

Každý zovšeobecnený tuple μ zovšeobecnenej relácie R zodpovedá formula F_μ zostrojená nasledovne: najprv existenčne kvantifikované premenné, zodpovedajúce „unknown“ nullom, potom všeobecne kvantifikované premenné, zodpovedajúce „arbitrary“ nullom, napokon predikát zodpovedajúci relácii R , obsahujúci všetky kvantifikované premenné a (prípadne) konštanty. V tabuľkovej reprezentácii zovšeobecnenej tupli μ zodpovedá množina nenullových tuplí α takých, ktoré sú *získané vhodnou substitúciou nullov*, a to presnejšie

- v atribútoch, kde μ má konštantu, α má tú istú hodnotu,
- v atribútoch, kde μ má \exists -null, α má ľubovoľnú hodnotu,
- ak má μ v nejakom atribúte \forall -null, existuje toľko (iba v tomto atribúte navzájom rôznych) tuplí β_i , aká je mohutnosť domény zodpovedajúcej danému atribútu.⁷⁰

Podobne relácii R nad V_i^* zodpovedá množina kvantifikovaných tvrdení $\{F_\mu \mid \mu \in R\}$.

V tabuľkovej reprezentácii je to množina (klasických) relácií r bez nullových hodnôt nad množinou stĺpcov V_i .

S reláciami sa narába podľa *predpokladu uzavretého sveta* (CWA), avšak iba v nasledujúcom zmysle. Nech R je zovšeobecnená relácia a α je tupľa bez nullov. Ak α nemôže byť získaná vhodnou substitúciou nullov zo žiadnej tuple relácie R , predpokladáme, že *nie je* prvkom nijakej relácie r zodpovedajúcej zovšeobecnenej relácii R . Je užitočné si všimnúť, že ak je súčasťou R tupľa $\langle \exists, \exists, \dots, \exists \rangle$ zo samých \exists -nullov (ktorá je zovšeobecným reprezentantom pre ľubovoľnú nenullovú tupľu), znamená to, že v tabuľke nemáme žiadnu negatívnu informáciu, inými slovami, že pracujeme pod *predpokladom otvoreného sveta* (OWA).

Biskup definuje ‘reláciu’ \preceq na množine V^* takto:

$$x \preceq y \Leftrightarrow_{df} x = \exists \text{ alebo } y = \forall \text{ alebo } x = y.$$

Ide teda o čiastočné usporiadanie⁷¹, v ktorom sú jednotlivé navzájom rôzne regulárne hodnoty neporovnateľné, \exists má úlohu spodného prvku (dolník) a \forall má úlohu horného prvku (horník). Biskup zdôrazňuje, že hoci majú s Vassiliouom rovnaké usporiadanie, ich

⁶⁸ Pretože Biskup prijíma pohľad na reláciu ako interpretáciu resp. model (v logickom zmysle slova) množiny výrazov.

⁶⁹ Samozrejme, pre úplnosť by bolo treba špecifikovať množiny, ktorých prvkami môžu byť x_i , na to by však bolo potrebné formálne definovať reláciu resp. zovšeobecnenú reláciu.

⁷⁰ Táto definícia nie je príliš formálna a odráža význam, že je to množina všetkých tuplí, pre ktoré predikát F_μ platí.

⁷¹ je reflexívne, tranzitívne a antisymetrické

horníky majú rozličné významy. ‘Relácia’ \preceq rozšírená na relácie je čiastočné usporiadanie (ba čo viac, zväz) vzhľadom na množinu všetkých zovšeobecnených relácií.

V-relácia (z angl. *value-relation*) je dvojica $\langle \text{relácia bez nullov, signatúra relácie} \rangle$. Medzi *v-reláciou* a zovšeobecnenou reláciou je rovnaký vzťah, ako medzi reláciou a multireláciou v zmysle [Imieliński–Lipski, 1984]: zovšeobecnenú reláciu môže „spĺňať“ mnoho *v-relácií*. Biskup ukázal, že usporiadanie na zovšeobecnených tupliach, teda syntaktických objektoch, korešponduje s usporiadaním na formulách (logické vyplývanie), teda na sémantických objektoch.

Z rozšírenia relačných operácií načrtujeme rozšírenie pre prirodzený join. Je potrebné, aby sme modifikovali definíciu spojenia tuplí aj pre Biskupove zovšeobecnené tuple, a to nasledovne: Hovoríme, že zovšeobecnené tuple μ a ν (nad množinami atribútov M resp. N) *mačujú*, ak pre každý atribút A množiny $M \cap N$ platí

$$\mu[A] \leq \nu[A] \text{ alebo } \nu[A] \leq \mu[A].$$

Ak tuple μ a ν mačujú, ich *spojenie* je tupľa τ (označenie $\mu \bowtie \nu$) nad $M \cup N$, definovaná nasledovne:

- $\tau[A] =_{df} \mu[A]$, ak $A \in M \setminus N$,
- $\tau[A] =_{df} \nu[A]$, ak $A \in N \setminus M$,
- ak $A \in M \cap N$, tak

$$\tau[A] =_{df} \begin{cases} \exists, & \text{ak } \mu[A] = \exists \wedge \nu[A] = \forall \text{ alebo } \nu[A] = \exists \wedge \mu[A] = \forall, \\ a, & \text{ak } \mu[A] = a \wedge \nu[A] \in \{a, \forall\} \text{ alebo } \nu[A] = a \wedge \mu[A] \in \{a, \forall\}, \\ \forall, & \text{ak } \mu[A] = \nu[A] = \forall. \end{cases}$$

V prípade, že chceme vo výsledku zahrnúť aj *maybe*-zovšeobecnené tuple, treba definovať spojenie nasledovne:

$$\tau[A] =_{df} \begin{cases} \exists, & \text{ak } \mu[A] = \exists \wedge \nu[A] \in \{\exists, \forall\} \text{ alebo } \nu[A] = \exists \wedge \mu[A] \in \{\exists, \forall\}, \\ a, & \text{ak } \mu[A] = a \wedge \nu[A] \in \{\exists, a, \forall\} \text{ alebo } \nu[A] = a \wedge \mu[A] \in \{\exists, a, \forall\}, \\ \forall, & \text{ak } \mu[A] = \nu[A] = \forall. \end{cases}$$

Toto spojenie označujeme $\mu \bowtie_B \nu$. (Rozdiel oproti $\mu \bowtie \nu$ je v tom, že sme tiež zahrnuli tuple, v ktorých je *maybe* $\exists = \exists$ a tiež $\exists = a$. *Maybe* tuple však generuje jedine \exists , nikdy nie \forall .)

Prirodzený *true*-join je definovaný štandardne, t.j. $R \bowtie S =_{df} \{\mu \bowtie \nu \mid \mu \in R \wedge \nu \in S\}$.

Biskupov join je definovaný nasledovne: $R \bowtie_B S =_{df} \{\mu \bowtie_B \nu \mid \mu \in R \wedge \nu \in S\}$.

Prirodzený *maybe*-join v Coddovom zmysle je vlastne rozdiel Biskupovho natural joinu a prirodzeného *true*-joinu, pretože \bowtie_B -spojenie tuplí obsahuje ako *true*-tuple, tak aj *maybe*-tuple.

[Biskup, 1982] ukázal, že takýto join je korektný a reštringovaný, teda korektne vyhodnocuje výsledok (výsledok neobsahuje žiadne nesprávne tuple) a obsahuje najväčší počet takýchto tuplí (keby sme chceli zväčšiť ich počet, stratilo by sa kritérium korektnosti). Ostatným operátorom relačnej algebry je venovaná rozsiahla štúdia [Biskup, 1983].

Vyhodnocovanie dotazov je však správne len v prípade, že sú jednotlivé relácie nezávislé. Teda napr. v prípade výrazu typu $f(T) \bowtie g(T)$ nemusí vrátiť správny výsledok. Biskup si bol tohoto nedostatku vedomý.

Biskup⁷² poznamenáva, že je možné rozšíriť jeho pohľad na nully zakomponovaním myšlienky *indexovaných nullov*, pričom by išlo len o indexovanie nullov typu „unknown“ (teda \exists_1, \exists_2 atď.).

3.6 „No information“ nully

[Zaniolo, 1984] navrhuje zabudnúť na rozdiel medzi dvoma principiálne rôznymi interpretáciami nullov, ako sme ich uviedli vyššie, pretože podľa jeho slov to nie je ani potrebné, ani žiaduce. V jeho riešení null stojí aj na mieste neznámej, aj neexistujúcej informácie (**ni**). Príklad 3 by potom interpretoval null hodnotu takto: „Nemám jeho telefónne číslo“ (z ľubovoľných dôvodov). Táto interpretácia nullov sa nazýva aj „no information“. Tento prístup zvolilo viacero autorov (napr. [Levene-Loizou, 1993]) a aj mnoho implementácií databázových programov. Je pozoruhodné, že tento model pre nully bol študovaný až po výsledkoch dosiahnutých Coddom, Lipskim, Biskupom a ďalšími, keď sa ukázalo, že tieto prístupy neprinesli v danej oblasti definitívne riešenie. No information null je totiž najzákladnejšou a najjednoduchšou interpretáciou nullovej hodnoty.

Na ilustráciu uvažujme náš príklad databázy zamestnancov spolu s ich telefónnymi číslami. Predstavme si, že databázový administrátor sa rozhodne⁷³ rozšíriť schému relácie o ďalší stĺpec Klapka, ktorý bude perspektívne obsahovať internú telefónnu klapku do kancelárie daného zamestnanca. Táto zmena neznamená, že všetci zamestnanci budú mať telefónnu klapku pridelenú naraz a dokonca, že všetci zamestnanci budú mať *vôbec* pridelenú klapku. Tento stĺpec sa bude vyplňať postupne, ako bude postupovať budovanie telefónnej siete, teda klapka bude zaznamenaná, akonáhle sa stane dostupnou.

Nuž a teraz stojí náš databázový administrátor pred problémom, ako zaobchádzať s takto rozšírenou databázou, hoci zmenil iba schému a nepridal ani jednu novú informáciu do tabuľky. Bežným riešením je uvažovať takú databázu, ako je na nasledujúcej tabuľke.

Príklad 19: Príklad 3 vzápätí po rozšírení o stĺpec Klapka

Meno	Telefónne číslo	Klapka
Fero	NULL (does not exist, ne)	– (ni)
Mišo	NULL (neviem, un)	– (ni)
Jano	6389432	– (ni)

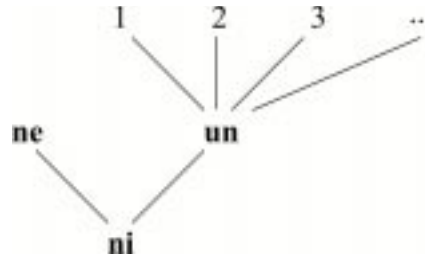
Všetky položky stĺpca Klapka našej tabuľky boli vyplnené symbolom „–“, ktorý označuje, že neviem o hodnote vlastne nič (**ni**): ani či existuje (a ak áno, aká presne je), ani či neexistuje.

Príklad 4 tiež uvádza takýto null, ktorý pri Andrejovom telefónnom čísle zohráva úlohu nijakej informácie, pretože neviem ani to, či Andrej má telefón.

⁷² [Biskup, 1982b], str. 46

⁷³ po rozhodnutí manažmentu podniku, ktorý odsúhlasil, že firma si vybuduje internú telefónnu sieť

„No information“ null modeluje všetky druhy chýbajúcej alebo neúplnej informácie, čo je jeho výhodou, avšak cena, ktorú treba zaplatiť za jednoduchú manipuláciu s ním, je strata akejkoľvek čiastkovej (čiastočnej) informácie. Obr. 2 ukazuje vzťah medzi základnými nullmi **un**, **ne** a **ni**. Čiara zdola hore pritom znamená „je všeobecnejší než“, „zahŕňa“.



Obr. 2: Vzťahy medzi základnými nullmi

Základný argument odôvodňujúci použitie „no information“ nullov je založený na pozorovaní, že databáza je vždy iba aproximáciou, priblížením sa k reálnemu svetu. Samozrejme, ako je popísané nižšie, pokiaľ sa jednotlivé domény štruktúrujú (napríklad algebraicky), čím sa zvýši ich komplexnosť, pričom sa popíše, čo znamená pojem „lepšie aproximovať“, je možné túto aproximáciu zachytiť, nie však vo všeobecnom prípade, keď o doménach nemáme žiadnu informáciu. Práve preto jediná vec, ktorú o atribúte vieme je, že ho nevieme.

Rovnako ako Codd, aj Zaniolo rozširuje každú doménu atribútov o špeciálnu hodnotu, práve „no information“ null **ni**, ktorý býva v tabuľkách značený „–“. Príklad 19 ukazuje reláciu, ktorá má hodnoty z takto rozšírených domén, ktoré sa nazývajú *rozšírené domény*. Z algebraického hľadiska tvorí rozšírená doména D spolu s „no information“ null hodnotou *diskrétnu cpo* (D_{ni}, \leq) , pričom jediné vzťahy medzi prvkami na $D_{ni} =_{df} D \cup \{\mathbf{ni}\}$ sú

$$\mathbf{ni} \leq a \text{ pre všetky } a \in D,$$

$$a \leq a \text{ pre všetky } a \in D$$

a úlohu prvku \perp (dolníka) hrá null hodnota **ni**. Ekvivalentne je možné rozšíriť aj domény o Coddovu null hodnotu @.

Hovoríme, že tupľa s je *viac informatívna* ako tupľa t (označenie $s \geq t$), ak platí, že vo všetkých atribútoch, kde t nie je **ni**, sú hodnoty atribútov v s ne-**ni** (nenullové) a sú rovnaké ako zodpovedajúce hodnoty v t . Napríklad

$$\begin{aligned} \langle a, b \rangle &\geq \langle a, - \rangle, \\ \langle a, b \rangle &\geq \langle a, b, - \rangle, \\ \langle a, b, - \rangle &\geq \langle a, b \rangle, \\ \langle a, b \rangle &\geq \langle a, b \rangle. \end{aligned}$$

Dve tuple sú (*informáciou*) *ekvivalentné*⁷⁴ (označenie $s \equiv t$), ak sú navzájom jedna druhej viac informatívne, teda platí $s \geq t$ aj $s \leq t$. Pretože \geq je reflexívna a tranzitívna, je ‘relácia’ \equiv reláciou ekvivalencie, lebo je aj symetrická. Napríklad $\langle a, b \rangle \equiv \langle a, b \rangle$ alebo aj $\langle a, b \rangle \equiv \langle a, b, - \rangle$.

⁷⁴ angl. *information-wise equivalent*

Z takto definovanej ekvivalencie vychádza presne opačný prístup k zaobchádzaniu s nullmi ako Coddov null-substitučný princíp: dve tuple s **ni** sú totiž ekvivalentné, čiže sa – zdanlivo optimisticky – predpokladá, že dva nully budú nahradené rovnakou hodnotou. Táto zdanlivá optimistickosť prístupu sa objasní, keď zopakujeme, čo sa myslí pod ekvivalenciou tuplí. Tá totiž znamená, v akom vzťahu sú naše *momentálne*, aktuálne znalosti o tupliach (teda o objektoch reálneho sveta). Keď teda o dvoch zodpovedajúcich si atribútoch vieme len to, že nepoznáme ich hodnoty, znamená to, že v prípade oboch tuplí sme na tom „rovnako“. Toto je Zaniolov prístup. Codd naopak predpokladá, že hoci *teraz* o hodnotách atribútov nič nevieme, je možné, že *v budúcnosti* budeme niečo vedieť, preto ešte teraz nemôžeme prehlásiť tuple za rovnaké. Vidno, že obaja autori sa na tento problém pozerajú akoby z dvoch opačných koncov, pričom Zaniolov prístup by sa dal formulovať nasledovne: „Keďže teraz o oboch hodnotách nič neviem, sú pre mňa rovnaké; ak sa neskôr ukáže opak, zmením svoje tvrdenie“, zatiaľ čo Coddov prístup by sa dal formulovať takto: „Hoci teraz o hodnotách nič neviem, nič ma neoprávňuje tvrdiť, že sú rovnaké; ak sa neskôr ukáže opak, zmením svoje tvrdenie“.

Tabuľka 4 zobrazuje pravdivostné tabuľky pre logické operácie **and**, **or** a **not** pre Zaniolove **ni** nully; existenčný resp. univerzálny kvantifikátor sa správajú ako iterovaný **or** resp. **and**. Zaniolo používa rovnakú trojhodnotovú logiku ako Codd.

Tabuľka 4: Pravdivostné tabuľky trojhodnotovej logiky

and	F	ni	T	or	F	ni	T	not	
F	F	F	F	F	F	ni	T	F	T
ni	F	ni	ni	ni	ni	ni	T	ni	ni
T	F	ni	T	T	T	T	T	T	F

V Zaniolovom prístupe, podobne ako u Biskupa, sa už črtajú zárodoky algebraického prístupu k zaobchádzaniu s nullovými hodnotami. Definujme *priese*k (*meet*) dvoch tuplí s , t ako tuple p (označenie $s \wedge t$) nasledovne (hodnota i -teho atribútu je označená s_i , t_i resp. p_i):

$$p_i =_{df} \begin{cases} s_i, & \text{ak } s_i = t_i \\ \mathbf{ni}, & \text{ak } s_i \neq t_i \end{cases}$$

Je pozoruhodné, že pre vyššie uvedenú definíciu nezáleží na tom, či platí **ni** = **ni** alebo **ni** ≠ **ni**. Je zrejmé, že ak platí $s \equiv \hat{s}$, potom $s \wedge t \equiv \hat{s} \wedge t$. Preto pokiaľ nerozlišujeme ekvivalentné tuple, vždy existuje priesek dvoch ľubovoľných tuplí a je daný jednoznačne. Priesek dvoch tuplí je viac informatívny ako ľubovoľná tuple a menej informatívna ako obe tieto tuple. Ba čo viac, priesek je najviac informatívna tuple a spomedzi všetkých tuplí, ktoré sú menej informatívne ako tuple s , t .

Hoci priesek existuje pre ľubovoľné dve tuple (totálne alebo obsahujúce **ni**), spojenie nemusí existovať, preto najprv predpokladáme, že tuple s , t sú *spojiteľné*, teda platí

ak $s_i \neq t_i$, potom platí $s_i = \mathbf{ni}$ alebo $t_i = \mathbf{ni}$.

Ak sú dve tuple s , t *spojiteľné*, ich *spojenie* (*join*) je tuple p (označenie $s \vee t$), definovaná nasledovne:

$$p_i =_{df} \begin{cases} s_i, & \text{ak } s_i \geq t_i, \\ t_i, & \text{ak } \neg(s_i \geq t_i). \end{cases}$$

Ak sú tuple s, \hat{s} spojitelné a platí $s \equiv \hat{s}$, potom platí $s \vee t \equiv \hat{s} \vee t$ a $s \wedge t \equiv \hat{s} \wedge t$. Keď stotožníme ekvivalentné tuple, relácia \geq definuje nielen čiastočné usporiadanie, ale dokonca polozväz.

Význam spojenia je nasledovný: Priesek dvoch tuplí je najmenej informatívna tupľa spomedzi všetkých tuplí, ktoré sú viac informatívne ako tuple s, t .

Poznamenajme, že prieseok a spojenie pre spojitelné tuple sú definované aj pre nekompatibilné tuple, hoci to v definícii priamo nie je vidieť. Uvedme niekoľko príkladov:

$$\begin{aligned} \langle A: a, B: b \rangle \text{ a } \langle A: \hat{a}, B: b \rangle &\text{ nie sú spojitelné,} \\ \langle A: a, B: b \rangle \vee \langle A: a, B: b \rangle &= \langle A: a, B: b \rangle, \\ \langle A: a, B: b \rangle \vee \langle A: -, B: b \rangle &= \langle A: a, B: b \rangle, \\ \langle A: -, B: b \rangle \vee \langle A: -, B: b \rangle &= \langle A: -, B: b \rangle, \\ \langle A: a, B: - \rangle \vee \langle A: -, B: b \rangle &= \langle A: a, B: b \rangle, \\ \langle A: a, B: b \rangle \vee \langle A: a, C: c \rangle &= \langle A: a, B: b, C: c \rangle, \\ \langle A: a, B: b \rangle \vee \langle C: c \rangle &= \langle A: a, B: b, C: c \rangle, \\ \langle A: a, B: b \rangle \vee \langle C: b, D: d \rangle &= \langle A: a, B: b, C: c, D: d \rangle, \\ \langle A: a, B: - \rangle \vee \langle C: b, D: d \rangle &= \langle A: a, B: -, C: c, D: d \rangle, \\ \langle A: a, B: - \rangle \vee \langle C: -, D: d \rangle &= \langle A: a, B: -, C: -, D: d \rangle. \end{aligned}$$

Pojem „byť viac informatívny“ sa ďalej rozširuje na relácie. Hovoríme, že relácia R *zahrňa*⁷⁵ reláciu S , ak platí

pre každú nenullovú tupľu $s \in S$ existuje tupľa $r \in R$ taká, že $r \geq s$.⁷⁶

Zapisujeme $R \supseteq S$. Táto ‘relácia’ je tranzitívna a reflexívna. Napríklad platí

$$\begin{aligned} \{\langle a, b_1 \rangle, \langle a, b_2 \rangle\} &\supseteq \{\langle a, b_1 \rangle, \langle a, - \rangle\}, \\ \{\langle a, b \rangle\} &\supseteq \{\langle a, b \rangle, \langle a, - \rangle\}, \\ \{\langle a, b \rangle, \langle a, - \rangle\} &\supseteq \{\langle a, b \rangle\}. \end{aligned}$$

Teraz zadefinujeme, čo znamená ekvivalentnosť relácií. Hovoríme, že relácie R, S sú (*informáciou*) *ekvivalentné* a zapisujeme $R \equiv S$, ak $R \supseteq S$ a $S \supseteq R$, teda ak sa navzájom zahrňajú.⁷⁷ ‘Relácia’ \equiv je vskutku reláciou ekvivalencie a faktorizuje množinu všetkých relácií na disjunktné triedy ekvivalencie. Môžeme tak použiť základný mechanizmus použitý pri rozšírení prirodzených čísel na racionálne čísla (pozri napr. [MacLane–Birkhoff, 1973], str. 168) na zavedenie pojmu *rozšírená relácia* (skrátene *x-relácia*).

Rozšírená relácia je trieda ekvivalencie vzhľadom $k \equiv$ na množine všetkých relácií. Trieda ekvivalencie, do ktorej patrí S , sa označuje $[S]$ resp. \hat{S} a hovoríme, že S je *reprezentantom* $[S]$. Tiež hovoríme, že $[S]$ je *uzáverom* relácie S . Podobne ako sme definovali „zahrňanie“ na reláciách, definujeme „obsahovanie“ aj na rozšírených reláciách. Rozšírená relácia $[R]$ *obsahuje* $[S]$ (zápis $[R] \supseteq [S]$), ak R zahrňa S . Správanie tejto ‘relácie’ nezávisí od výberu reprezentanta triedy: ak

⁷⁵ angl. *subsumes*

⁷⁶ Ako uvidíme neskôr, takéto usporiadanie je možné definovať viacerými spôsobmi.

⁷⁷ Ako vidieť, platí napr. $\{\langle a, b \rangle\} \equiv \{\langle a, b \rangle, \langle a, - \rangle\}$.

$$R_1 \cong R_2 \text{ a } S_1 \cong S_2,$$

potom

$$[R_1] \supseteq [S_1] \text{ platí práve vtedy, keď } [R_2] \supseteq [S_2].$$

Takže $[R] = [S]$ práve vtedy, keď $[R] \supseteq [S]$ a $[S] \supseteq [R]$.

Hovoríme, že tupľa μ *x-patrí* do $[S]$ (alebo je *x-prvkom* $[S]$) a zapisujeme

$$\mu \in [S],$$

ak existuje taký prvok $S \in [S]$, že $\mu \in S$. Platí, že tupľa μ *x-patrí* *x-relácii* práve vtedy, ak jej reprezentant obsahuje tupľu, ktorá je viac informatívna ako μ . Používame skrátený zápis

$$\mu \notin [S]$$

namiesto $\neg(\mu \in [S])$.

Pre množinu tuplí

$$M := \{t_1, t_2, \dots, t_n\}$$

definujeme *uzáver* množiny M ako najmenšiu triedu ekvivalencie, do ktorej patrí množina M , zapisujeme

$$[M]_{=nt} [\{t_1, t_2, \dots, t_n\}].$$

Tabuľka 5 obsahuje definície základných množinových operácií na *x-reláciách* použitím tejto notácie. *x-prienik* je definovaný uvedeným spôsobom preto, aby zahŕňal „spoločnú informáciu“ zahrnutú v oboch *x-reláciách*.

Tabuľka 5: Operácie na x-reláciách

$$\begin{array}{lll} \text{zjednotenie:} & [R] \cup [S] & =_{df} [\{\mu \mid \mu \in R \text{ alebo } \mu \in S\}] \\ \text{x-prienik:} & [R] \cap [S] & =_{df} [\{\mu \wedge \nu \mid \mu \in R \text{ a } \nu \in S\}] \\ \text{rozdiel:} & [R] \setminus [S] & =_{df} [\{\mu \mid \mu \in R \text{ a } \mu \notin S\}] \end{array}$$

Samozrejme, že nezáleží na tom, ktorého reprezentanta vezmeme za predstaviteľa triedy ekvivalencie na výpočet zjednotenia, *x-prieniku* a rozdielu, inými slovami ak

$$[R_1] = [R_2] \text{ a } [S_1] = [S_2],$$

potom platí

$$\begin{aligned} [R_1] \cup [S_1] &= [R_2] \cup [S_2], \\ [R_1] \cap [S_1] &= [R_2] \cap [S_2], \\ [R_1] \setminus [S_1] &= [R_2] \setminus [S_2]. \end{aligned}$$

Prečo je *x-prienik* definovaný odlišne ako v klasickej relačnej algebre? *Prienik* má označovať informáciu obsiahnutú ako v jednej, tak aj v druhej relácii. Uvedieme nasledujúce dve relácie, nech pritom prvý atribút označuje dodávateľa a druhý dodávané pečivo, ako bolo uvedené v niekoľkých predchádzajúcich príkladoch:

$$\begin{aligned} [R_1] &:= [\langle \text{Anton}, \text{perník} \rangle], \\ [R_2] &:= [\langle \text{Anton}, \text{chlieb} \rangle]. \end{aligned}$$

Akú informáciu nesie R_1 ? Vraví „Anton dodáva *perník*.“ Podobne R_2 značí „Anton dodáva *chlieb*.“ $[R_1]$ znamená množinu všetkých relácií, ktoré sú informačne ekvivalentné s R_1 . Keďže $[R_1]$ a $[R_2]$ označujú triedu ekvivalencie podľa ‘relácie’ \cong , obsahujú aj reláciu $\{\langle \text{Anton}, \text{perník} \rangle, \langle \text{Anton}, - \rangle\}$

resp.

$$\{\langle \text{Anton}, \text{chlieb} \rangle, \langle \text{Anton}, - \rangle\}^{78}$$

Čo označuje x -prienik $[R_1]$ a $[R_2]$? Stačí uvážiť, že spoločná informácia v $[R_1]$ aj $[R_2]$ je tá, že „Anton dodáva“. Je síce pravda, že nemožno povedať ani „Anton dodáva *perník*“ ani „Anton dodáva *chlieb*“, ale zato informácia, že „Anton dodáva“, je *prienikom* toho, o čom vypovedajú $[R_1]$ a $[R_2]$. Preto

$$[R_1] [\cap] [S_1] = [\langle \text{Anton}, - \rangle].$$

Na tomto mieste uveďme, že zjednotenie a x -prienik definujú najmenšiu hornú hranicu a najväčšiu dolnú hranicu vzhľadom k čiastočnému usporiadaniu \sqsubseteq :

$$\text{ak } [U] \sqsupseteq [R] \text{ a } [U] \sqsupseteq [S], \text{ potom } [U] \sqsupseteq [R] \cup [S]$$

(ak $\mu \in [R] \cup [S]$, potom $\mu \in [R]$ alebo $\mu \in [S]$; nech $\mu \in [R]$: potom $\mu \in [U]$, lebo $[U] \sqsupseteq [R]$) a tiež

$$\text{ak } [U] \sqsubseteq [R] \text{ a } [U] \sqsubseteq [S], \text{ potom } [U] \sqsubseteq [R] \cap [S].$$

Tak sme ukázali, že množina všetkých x -relácií spolu s operáciou \sqsubseteq tvorí zväz, ktorý je navyše *distributívny*, pretože z definícií vyplýva

$$[R] [\cap] ([S] \cup [T]) = ([R] [\cap] [S]) \cup ([R] [\cap] [T]),$$

$$[R] \cup ([S] [\cap] [T]) = ([R] \cup [S]) [\cap] ([R] \cup [T]).$$

(Stačí dokázať iba jednu z uvedených vlastností, pretože platnosť druhej vyplýva z duálnosti.)

Tento zväz má dolníka $\perp := [\emptyset]$ ⁷⁹, pretože pre každú x -reláciu $[R]$ platí

$$[R] [\cap] [\emptyset] = [\emptyset].$$

Horníka τ ⁸⁰ možno charakterizovať jeho reprezentantom tak, že sú to všetky možné n -tice (kde n označuje počet všetkých atribútov, vždy konečný počet; nech sú to A_1 až A_n):

$$H := D(A_1) \times \dots \times D(A_n),$$

takže $\tau = [H]$. Treba poznamenať, že H nie je množinou všetkých tuplíc, pretože niektoré tuple v H obsiahnuté nie sú (dajme tomu, že $a_1 \in D(A_1)$, $a_2 \in D(A_2)$, $n = 3$; tupľa $\langle a_1, a_2 \rangle$ nepatrí do H , hoci $\langle a_1, a_2, - \rangle$, ktorá je s ňou informáciou ekvivalentná, do H patrí). Vo všeobecnosti nemá každá x -relácia komplement.⁸¹ Rovnako nie vždy platí, že $([R] \setminus [S]) [\cap] [S] = \emptyset$ ⁸², ale platí pre ľubovoľné x -relácie $[R]$, $[S]$ také, že $[R] \sqsupseteq [S]$,

$$([R] \setminus [S]) \cup [S] = [R].$$

Taktiež z definícií máme

$$\text{ak } [T] \cup [S] = [R], \text{ tak } [T] \sqsupseteq ([R] \setminus [S]).$$

⁷⁸ Pretože $\{\langle \text{Anton}, \text{chlieb} \rangle, \langle \text{Anton}, - \rangle\} \sqsubseteq \{\langle \text{Anton}, \text{chlieb} \rangle\}$.

⁷⁹ $[\emptyset]$ má jediného reprezentanta, prázdnu reláciu.

⁸⁰ platí: $[R] \cup \tau = \tau$

⁸¹ Nech $n = 2$, $D(A_1) = \{a\}$, $D(A_2) = \{b_1, b_2\}$. Tuple

$$\lambda_1 := \langle a, b_1 \rangle, \lambda_2 := \langle a, b_2 \rangle$$

sa nachádzajú v H , čiže sú x -prvkami τ . Nech $[R]$ x -obsahuje λ_1 , ale neobsahuje λ_2 . Potom x -relácia $[R']$, o ktorej požadujeme, že $[R] \cup [R'] = \tau$, musí x -obsahovať λ_2 ($\lambda_2 \in [R']$). Avšak potom tupľa $\langle a, - \rangle$ patrí ako do $[R]$, tak aj $[R']$, čiže $[R] \cap [R'] \neq \emptyset$.

⁸² Nech $R := \{\lambda_1\}$, $S := \{\lambda_2\}$, potom $\langle a, - \rangle \in ([R] [\cap] [S])$, ale $\langle a, - \rangle \notin [S]$, čiže $([R] [\cap] [S]) \setminus [S] \neq \emptyset$.

Vidno, že $([R] \setminus [S])$ je vzhľadom k \sqsubseteq najmenšia x -relácia, ktorej zjednotenie s $[S]$ dáva $[R]$. Význam takto načrtnutého *minimálneho reprezentanta* je dôležitý pre narábanie s x -reláciami. Presnejšie, hovoríme, že relácia M je *minimálny reprezentant* rozšírenej relácie $[R]$ (samozrejme, $[M] = [R]$), ak žiadna vlastná podmnožina relácie M nie je reprezentantom $[R]$, inými slovami, ak platí

$$\forall N (N \subsetneq M \Rightarrow [N] \subsetneq [R]).$$

Skonstruovať minimálneho reprezentanta je ľahké: stačí odstrániť nullovú tupľu – ak existuje – a postupne odstraňovať všetky tuple, ktoré sú menej informatívne. Tento proces je možné chápať ako rozšírenie procesu, ktorým sa odstraňujú duplicitné tuple (dodržiavajúc neduplikačné pravidlo) v relačnom modeli bez nullových hodnôt. Aby bol minimálny reprezentant jednoznačný, treba ešte istým spôsobom obmedziť prebytočné atribúty. Príklad 3 a Príklad 19 ukazujú dvoch minimálnych reprezentantov vzhľadom na odlišné množiny atribútov, pritom však v druhom prípade je atribút „Klapka“ akoby zbytočný. Preto hovoríme, že množina atribútov W je *rozsahom* rozšírenej relácie $[R]$, keď sa $[R]$ dá reprezentovať reláciou nad W , ale nedá sa reprezentovať nijakou reláciou nad ľubovoľnou vlastnou podmnožinou atribútov z W . Teda v procese konštruovania minimálneho reprezentanta napokon odstránime atribúty, pre ktoré sú všetky hodnoty tuplí **ni**, aby sme našli rozsahovo najmenšieho minimálneho reprezentanta.

Napokon je potrebné definovať základné relačné operácie, aby sa dali efektívne vyhodnocovať. Tabuľka 5 uvádza totiž operácie, na ktorých vypočítanie sa uvažuje neúmerne množstvo tuplí kvôli použitiu operátora \in . Je možné odvodiť ekvivalentné formulácie definícií, používajúce \in namiesto \in . Sú uvedené v nasledujúcej tabuľke. Tabuľka 6 tiež obsahuje definíciu elementárnej selekcie (θ znamená operátor ako $<$, \leq , $=$, \geq , $>$; k je konštanta, A, B sú atribúty), kartézskeho súčinu, joinu a projekcie (N je podmnožina atribútov relácie R).

Tabuľka 6: Efektívne vypočítateľné operácie na x -reláciách

<i>zjednotenie:</i>	$[R] \cup [S]$	$=_{df} [\{\mu \mid \mu \in R \text{ alebo } \mu \in S\}]$
<i>x-priemik:</i>	$[R] [\cap] [S]$	$=_{df} [\{\mu \wedge \nu \mid \mu \in R \text{ a } \nu \in S\}]$
<i>rozdiel:</i>	$[R] \setminus [S]$	$=_{df} [\{\mu \mid \mu \in R \text{ a } \forall \nu \in S: \neg(\nu \geq \mu)\}]$
<i>selekcia:</i>	$\sigma_{A\theta B}[R]$	$=_{df} [\{\mu \mid \mu \in R, \text{ je } AB\text{-totálna a } \mu[A] \theta \mu[B]\}]^{83}$
	$\sigma_{A\theta k}[R]$	$=_{df} [\{\mu \mid \mu \in R, \text{ je } A\text{-totálna a } \mu[A] \theta k\}]$
<i>kartézsky súčin:</i>	$[R] [\times] [S]$	$=_{df} [\{\mu \vee \nu \mid \mu \in R, \nu \in S, \text{ nie sú nullové}\}]^{84}$
<i>join:</i>	$[R] [\bowtie] [S]$	$=_{df} [\{\mu \vee \nu \mid \mu \in R, \nu \in S, \text{ obe sú } M\text{-totálne}\}]^{85}$
<i>projekcia:</i>	$\pi_W[R]$	$=_{df} [\{\mu[W] \mid \mu \in R\}]^{86}$

Poznámka: Pri kartézskom súčine uvažujeme o tých atribútoch, ktoré majú R a S rovnaké, ako o odlišných atribútoch z hľadiska definície spojenia. Pri joinu rovnaké atribúty

⁸³ Aby nedošlo k nedorozumeniu, poznamenajme, že hranaté zátvorky v $\mu[A]$ znamenajú „reštrikcia tuple μ na atribút A “, $[R]$ je uzáver R .

⁸⁴ Pritom uvažujeme o množinách atribútov R a S ako o disjunktných množinách; ak sa prekrývajú, treba rovnaké atribúty syntakticky odlišiť (premenovaním).

⁸⁵ Množina atribútov M je priemik množín atribútov relácií R a S .

⁸⁶ Pripomeňme opäť, že hranaté zátvorky v $\mu[W]$ znamenajú „reštrikcia tuple μ na množinu atribútov W “.

nerozlišujeme. Toto zabezpečí elimináciu rovnakých stĺpcov vo výslednom join. Pri join požadujeme, aby boli všetky tuple z R aj S totálne na prieniku atribútov relácií R a S .

Jednoduchá implementácia zjednotenia vyžaduje čas rádovo $|R| + |S|$, projekcie a selekcie $|R|$, x -prieniku, kartézského súčinu, joinu a rozdielu rádovo $|R| \times |S|$, pričom tento horný odhad sa dá zlepšiť použitím techník ako je hašovanie.

Aký je vzťah klasických relácií bez nullov a x -relácií? Ak R je beznullová relácia, $[R]$ je totálna (neobsahuje nully). Navyše pre dve rôzne beznullové relácie sú ich uzávery rôzne, takže existuje jednoznačná korešpondencia – bijekcia – medzi beznullovými reláciami a totálnymi x -reláciami. Táto korešpondencia zachováva všetky relačné operátory: zjednotenie (U), rozdiel (D), kartézsky súčin (X), selekciu (S), projekciu (P). Stačí totiž poznamenať, že

1. pre dve kompatibilné nenullové relácie R_1, R_2 platí:
 - ak $R_1 \cup R_2 = R_3$, potom $[R_1] \cup [R_2] = [R_3]$,
 - ak $R_1 \setminus R_2 = R_3$, potom $[R_1] \setminus [R_2] = [R_3]$,
 - ak $R_1 \supseteq R_2$, potom $[R_1] \supseteq [R_2]$,
2. pre dve nenullové relácie R_1, R_2 platí:
 - ak $R_1 \times R_2 = R_3$, potom $[R_1] \times [R_2] = [R_3]$,
3. pre nenullovú reláciu R , ktorá atribút(y) A (resp. B) platí:
 - ak $\sigma_{A\theta k} R = R_1$, potom $\sigma_{A\theta k} [R] = [R_1]$,
 - ak $\sigma_{A\theta B} R = R_1$, potom $\sigma_{A\theta B} [R] = [R_1]$,
4. pre nenullovú reláciu R a podmnožinu jej atribútov W platí:
 - ak $\pi_W R = R_1$, potom $\pi_W [R] = [R_1]$.

Preto sme použili rovnaké symboly pre operácie ako na nenullových, tak na rozšírených reláciách. Dôležitosť tohoto výsledku je v tom, že s reláciami bez nullov v Zaniolovom systéme nemusíme zaobchádzať *ináč* ako v klasickej relačnej algebre nad reláciami bez nullov. (Podobne sa v algebre ukazuje, že s prirodzenými číslami na množine racionálnych čísel sa manipuluje rovnako ako na samotnej množine prirodzených čísel *bez zmeny* operácií. Porov. napr. [MacLane–Birkhoff, 1973].) Teda rozšírenie relačného modelu na rozšírené relácie je konzistentné rozšírenie.

Napokon uveďme príklad (založený na dodávaní pečiva), ktorý ukazuje, že Zaniolova teória nie je iba pekná, ale aj použiteľná pri zodpovedaní na dotazy. Definujeme operátor *podielu*, ktorý nám ukáže väčšiu vhodnosť Zaniolovho prístupu pri zodpovedaní na dotazy ako Coddovho. Nech množina atribútov relácie S je podmnožina atribútov relácie R a nech M sú ostatné atribúty relácie R . Podiel $R \div S$ je definovaný nasledovne:

$$R \div S =_{df} \pi_M R \setminus \pi_M((\pi_M R \times S) \setminus R).$$

Preto pre rozšírené relácie definujeme

$$[R] \div [S] =_{df} \pi_M [R] \setminus \pi_M((\pi_M [R] \times [S]) \setminus [R_M]),$$

pričom R_M označuje množinu všetkých M -totálnych tuplů v R .

Tentokrát náš príklad dodávania pečiva je daný nasledujúcou tabuľkou.

Dodáva

Dodávateľ	Pečivo
Anton	<i>perník</i>
Anton	<i>chlieb</i>
Anton	–
Braňo	<i>perník</i>
Braňo	–
Cyril	–
Dano	<i>vianočka</i>

Uvažujme nasledujúci dotaz:

$Q :=$ „Všetci dodávatelia, ktorí dodávajú každé pečivo dodávané Braňom“.

Odpoveď A na dotaz môže zistená relačným výrazom

$$A := [Dodáva] \div [B],$$

kde B označuje množinu pečív, ktoré dodáva Braňo, formálne

$$[B] := \pi_{\text{Pečivo}}(\sigma_{\text{Dodávateľ} = \text{Braňo}}[Dodáva])$$

Aký bude výsledok $[B]$? Z tabuľky vidno, že je to $\{\textit{perník}, -\}$. Aký výsledok dá Coddova manipulácia s nullmi? *True* verzia selektu dá výsledok $\{\textit{perník}, -\}$, *maybe* verzia dá výsledok \emptyset .

Výsledok A podľa Coddova bude

$$A_{\text{TRUE}} = \emptyset \text{ (teda žiaden dodávateľ)}^{87} \text{ pri } \textit{true}\text{-vyhodnocovaní,}$$

$$A_{\text{MAYBE}} = \{\text{Anton, Braňo, Cyril}\} \text{ pri } \textit{maybe}\text{-vyhodnocovaní.}$$

Avšak použijúc Zaniolovu definíciu *podielu* dostaneme

$$[A_{\text{Zaniolo}}] = [\{\text{Anton, Braňo}\}].$$

Z jednotlivých uvedených výsledkov vyplýva, že treba preformulovať pôvodný dotaz Q , aby presnejšie vyjadroval, čo sme dosiahli.

Coddovo *true*–vyhodnocovanie implementuje nasledovný pozmenený dotaz $Q_{\text{TRUE}} =$

„Všetci dodávatelia, ktorí *určite* dodávajú každé pečivo, ktoré *môže* dodávať Braňo.“

Coddovo *maybe*–vyhodnocovanie implementuje nasledovný pozmenený dotaz $Q_{\text{MAYBE}} =$

„Všetci dodávatelia, ktorí *možno* dodávajú každé pečivo, ktoré *určite* dodáva Braňo.“

Zaniolovo vyhodnocovanie implementuje nasledovný pozmenený dotaz $Q_{\text{Zaniolo}} =$

„Všetci dodávatelia, ktorí *určite* dodávajú každé pečivo, ktoré *určite* dodáva Braňo.“

Takéto preformulovanie je potrebné, pretože vidno, že záleží na tom, ako interpretujeme univerzálny kvantifikátor na vyjadrenie toho, čo sme *naozaj* chceli dosiahnuť. Najmä

⁸⁷ paradox tohoto typu (vlastne ani Braňo nedodáva všetky druhy pečiva, ktoré dodáva Braňo), je spomínaný aj vyššie (str. 32)

v prípadoch, keď sa vyskytujú v dotaze viacej ráz. Zaniolo si zvolil ten prístup, že každý výskyt slova „každý“, „všetky“ (resp. ekvivalentné) interpretuje ako „určite“. Tým sa vyhne v poznámke 87 spomínanému Coddovmu paradoxu, podrobnejšie pozri str. 32.

Naostatok poznamenajme pre zhrnutie, že Zaniolov prístup zjednodušuje sémantiku nullov a navyše nevedie k paradoxom ako Coddov prístup. [Libkin, 1994] rozširuje Zaniolovu myšlienku o použití aproximácií na lepšie charakterizovanie „neúplnosti“ informácie, keď uvažuje o zložitejších algebraických štruktúrach na reprezentovanie štruktúry domén. Formálne definuje aj spôsob narábania s takouto informáciou, pričom zaviedol jazyk OR-SML na manipuláciu s dátami, v ktorých je neúplnosť charakterizovaná aproximáciou. Modelom tohto typu pre *nepresnú informáciu* sa v tejto práci nezaobráame.

3.7 Update operácie

Zatiaľ sme sa zaoberali vplyvom nullov na jednotlivé relačné operácie, avšak nie na *update* operácie. Preto v tejto časti stručne zhrnieme vplyv neúplnej informácie na tieto operácie.

- *Insertion* (vloženie) tuple do tabuľky: vykonáme nasledujúce dva kroky.
 1. *Syntaktický krok*: keďže sa jedná o zjednotenie, ide o jednoduché pridanie tuple do tabuľky, avšak – ako sme spomenuli vyššie – na tejto úrovni uvažujeme v zmysle *neduplikačného pravidla*, a to aj pre iné modely ako Coddove nullové hodnoty. Inými slovami, pridanie tuple $\mu := \langle \forall, a, \exists, \forall, b \rangle$ v Biskupovom modeli pre neúplnú informáciu do tabuľky, ktorá takúto zovšeobecnenú tupľu už obsahuje, nespôsobí žiadnu zmenu tabuľky, pretože už je v tabuľke obsiahnutá. (Pretože na tejto úrovni ide o syntaktickú totožnosť objektov, predpokladáme platnosť rovností $\omega \equiv \omega, \exists \equiv \exists, \forall \equiv \forall, - \equiv -$.)
 2. *Sémantický krok*: Po syntaktickom vložení je potrebné vykonať sémantickú kontrolu, či teda je prípustné danú tupľu (obsahujúcu neúplnú informáciu) do tabuľky vložiť. Nie je napr. možné vložiť tupľu, obsahujúcu null v atribúte, ktorý je súčasťou kľúča, do Coddovej tabuľky. Je na návrhárovi systému, ktoré ďalšie tuple zakáže. Napr. ak chceme Biskupov model pre nullové hodnoty reprezentovať podľa CWA predpokladu, musíme zakázať tuple tvaru $\langle \exists, \exists, \dots, \exists \rangle$ zo samých \exists -nullov (porov. str. 42).
- *Deletion* (vymazanie) z tabuľky: vykonáme nasledujúci krok resp. dva kroky:
 1. *Vyhľadanie*: zistíme, či sa takýto syntaktický objekt v tabuľke vyskytuje (porovnávame syntakticky, teda pomocou \equiv),
 2. *Vymazanie*: ak sa takýto objekt ako tupľa, ktorej vymazanie požadujeme, v tabuľke vyskytuje (a z dôvodu *neduplikačného pravidla* sa vyskytuje nanajvyš raz), zrušíme ju, odstránime.
- *Modifikácia* tuple/tuplí nachádzajúcej sa v tabuľke:
 - (a) nahradenie tuple $\mu \in T$ tupľou μ' : zrušenie tuple (deletion) a následne pridanie tuple (insertion),
 - (b) nahradenie tuplí spĺňajúcich F tak, aby spĺňali F' , pričom F' je výraz neobsahujúci nerovnosti (substitúcia):
 1. najprv vyberieme všetky tuple, vyhovujúce podmienke F ,
 2. naraďíme hodnoty týchto tuplí podľa substitúcie F' ,
 3. vykonáme vloženie tuplí do tabuľky (insertion), kde sa skontroluje prípadná syntaktická rovnosť tuplí (*neduplikačné pravidlo*).

Ak chceme povoliť iba modifikáciu, ktorá bude spôsobovať iba vzrastanie úplnosti informácie, zakazujúc „zhoršenie“ našej znalosti, treba dotazovací jazyk (resp. operáciu modifikácie) implementovať vzhľadom k nasledujúcemu jednoduchému princípu: „Kedykoľvek je informácia pridaná do (neúplnej) databázy, nasledujúce odpovede na dotazy nesmú byť v rozpore s predošlými, nesmú im odporovať a nesmú byť menej informatívne.“⁸⁸

Nasledujúce výsledky z [Abiteboul–Grahne, 1985] charakterizujú aktualizčné operácie pre Coddove, V–tabuľky a C–tabuľky:

- Coddove tabuľky sú reprezentačným systémom s relačnými operáciami PS (projekcia, selekcia) spolu s update operáciami vloženia, integrácie a vymazania⁸⁹,
- V–tabuľky sú reprezentačným systémom s relačnými operáciami $PS^+ UJR$ spolu s update operáciami vloženia, integrácie a subjekcie⁹⁰,
- V–tabuľky nie sú reprezentačným systémom, ak obsahujú nasledovné operácie:
 - PS^+ a vymazanie⁹¹,
 - PS^+ a modifikácia,
 - PS^+ a rozšírenie (augmentation),
 - PS^+ a subjekcia.
- C–tabuľky sú reprezentačný systém so všetkými operáciami relačnej algebry a všetkými aktualizáčnymi operáciami.

⁸⁸ Porov. [Golshani, 1985], str. 293.

⁸⁹ ak je vymazávanie s podmienkou, táto musí byť úplná, teda podmienka F bez neúplnej informácie

⁹⁰ subjekcia musí byť iba pozitívnymi závislosťami bez konštánt

⁹¹ hoci by aj bolo len s úplnou podmienkou

4 Čiastočná informácia

„Lebo poznávame len čiastočne...“
sv. Pavol⁹²

Už úvahy o viacerých nullových hodnotách nám dávali dôvod k hlbšiemu zamysleniu sa nad neúplnou informáciou. V tejto časti sa budeme venovať zložitejšej reprezentácii neúplnej informácie, keď máme len čiastočnú informáciu o hodnote atribútu. V prípade, že poznáme štruktúru domény, môžeme hovoriť aj o miere čiastočnosti informácie, a teda informáciu usporiadať podľa čiastočnosti či naopak, podľa úplnosti.

4.1 Podmnožinová konštrukcia

Všeobecným modelom, spomínaným v [Lipski, 1979] či [Grant, 1980], je relačný model rozšírený nasledovným spôsobom. Pre domény D_1, \dots, D_n relácia $R \subseteq D_1 \times \dots \times D_n$ obsahuje tuple tvaru $\langle a_1, \dots, a_n \rangle$ pozostávajúce z prvkov domén. Zovšeobecnenie vedie k tomu, že umožníme, aby tuple obsahovali nie jednotlivé prvky domén, ale *podmnožiny* domén. *Všeobecná tupľa* nad D_1, \dots, D_n je teda n -tica $\langle M_1, \dots, M_n \rangle$, kde $M_i \subseteq D_i$. V tomto zmysle klasickej tupli $\langle a_1, \dots, a_n \rangle$ zodpovedá všeobecná tupľa $\langle \{a_1\}, \dots, \{a_n\} \rangle$.

Všeobecná relácia R nad D_1, \dots, D_n je konečná množina všeobecných tuplí nad D_1, \dots, D_n , teda

$$R \subseteq 2^{D_1 \times \dots \times D_n},$$

$$R = \{ \langle M_1, \dots, M_n \rangle \mid M_1 \times \dots \times M_n \subseteq D_1 \times \dots \times D_n \}.$$

Tento všeobecný relačný model sa nazýva aj *podmnožinová konštrukcia*. Samozrejme, všeobecnej relácii zodpovedá jedna alebo viac možných reprezentácií klasickými reláciami. Od interpretovania všeobecnej tuple závisí množina prípustných reprezentácií, ktoré určujú, o aký druh čiastočnej informácie sa jedná.

- $\langle M_1, \dots, M_n \rangle$ znamená, že v každej reprezentácii sa nachádza *práve jedna* tupľa $\langle a_1, \dots, a_n \rangle$, kde $a_i \in M_i$, $i \in \{1, \dots, n\}$. Takýto druh čiastočnej informácie sa nazýva *exkluzívna disjunktívna informácia* alebo tiež *alternatívna informácia*.⁹³
- $\langle M_1, \dots, M_n \rangle$ znamená, že v každej reprezentácii sa nachádza *aspoň jedna* tupľa $\langle a_1, \dots, a_n \rangle$, kde $a_i \in M_i$, $i \in \{1, \dots, n\}$. Tento druh čiastočnej informácie sa nazýva *inkluzívna disjunktívna informácia*. Pozri časť 4.4 (str. 62).
- $\langle M_1, \dots, M_n \rangle$ znamená, že v každej reprezentácii sa nachádzajú *všetky* tuple $\langle a_1, \dots, a_n \rangle$, kde $a_i \in M_i$, $i \in \{1, \dots, n\}$. V tomto prípade vlastne nejde o druh čiastočnej informácie, ale o rozšírenie relačného modelu o množinové atribúty. Všeobecná relácia sa dá previesť do ekvivalentnej množiny klasických relácií spôsobom popísaným v dodatku, str. 95.

4.2 Číselné domény: intervalové hodnoty

Prvým článkom o čiastočnej informácii pre dva typy domén, číselné a reťazcové, bol článok [Grant, 1980], neskôr jeho myšlienku prepracovali a rozšírili [Ola-Özsoyoglu,

⁹² Biblia, 1Kor 13, 9

⁹³ Porov. [Homenda, 1991]

1993]. Všeobecná množinová konštrukcia popísaná vyššie nie je príliš praktická, a navyše sa vlastne vymyká relačnému modelu – podobá sa skôr na objektovo-orientovaný prístup k dátam. Preto v tejto časti študujeme jednoduchšie modely pre špeciálne dátové domény. Ako prvé sa ponúkajú *číselné hodnoty*. Rozumieme pod nimi hodnoty z množín prirodzených, celých a reálnych čísel. Tieto množiny, svorne nazývané *číselné domény*, rozširujeme o ďalší typ hodnôt (okrem prvku domény, teda konkrétneho čísla), ktoré nazývame *intervalové hodnoty* alebo krátko *interval* (Grant) resp. *množiny intervalov* (Ola, Özsoyoglu). Z teoretického hľadiska je rozšírenie uvedeným spôsobom korektné a nevedie k nekonzistencii s prvou normálnou formou, avšak v praxi je interval vždy dvojica, a teda komplexnejší objekt, ktorý je reprezentovaný zložitejšie (prinajmenšom ako dvojica dát). Takže použitie takejto domény rozšírenej o intervaly buď vedie k praktickým problémom s 1NF alebo k neefektívnemu uchovávaní dát, keď sa budú aj úplné číselné dáta uchovávať ako dvojica napr. s prázdnu druhou zložkou (ak budú prázdne obe zložky, ide o **un**–null). Toto sú implementačné ťažkosti, ktoré okrajovo spomína [Štuller, 1997], str. 40.

4.2.1 Grantove intervalové hodnoty

Interval nad číselnou doménou D je dvojica $[i, j]$ čísel, pričom $i, j \in D$ a navyše $i < j$.

Obohatená doména D' pre pôvodnú číselnú doménu D je množina

$$D' = D \cup \{[i, j] \mid i, j \in D, i < j\} \cup \{@\},$$

pre nečíselnú doménu je to tá istá doména.⁹⁴ Symbol $@$ označuje Coddovu null hodnotu unknown typu (**un**).

Grantova relácia nad doménami D_1, \dots, D_n je relácia $R \subseteq D'_1 \times \dots \times D'_n$ obsahujúca tuple tvaru $\langle a_1, \dots, a_n \rangle$, pričom pre číselné domény a_i je

- úplná číselná hodnota,
- interval,
- **un**–null.

Pre danú Grantovu reláciu existuje viacero zodpovedajúcich klasických relácií. Množina všetkých týchto tabuliek sa nazýva *reprezentácia*.⁹⁵ Formálne, *reprezentácia prvku* $a \in D'$ (označenie $\llbracket a \rrbracket$) je množina, závisiaca od hodnoty a , definovaná nasledovne:

$$\llbracket a \rrbracket =_{df} \begin{cases} \{a\}, & \text{ak } a \in D, \\ \{n \in D \mid i \leq n \leq j\}, & \text{ak } a \text{ je interval } [i, j], \\ D, & \text{ak } a = @ \in D', \end{cases}$$

pričom v prvom prípade, ak $D \neq D'$, tak a je úplná číselná hodnota.

Prvok (v množinovom zmysle) reprezentácie prvku $a \in D'$ sa nazýva *vlastná substitúcia prvku* a . Podobne, *reprezentácia tuple* $\mu = \langle a_1, \dots, a_n \rangle \in D'_1 \times \dots \times D'_n$ je množina tuplů nad D_1, \dots, D_n , ktorú označujeme $\llbracket \mu \rrbracket$ a definujeme nasledovne:

$$\llbracket \mu \rrbracket =_{df} \{ \langle a'_1, \dots, a'_n \rangle \mid a'_i \in \llbracket a_i \rrbracket, i \in \{1, \dots, n\} \}.^{\text{96}}$$

⁹⁴ Grant vo svojom článku hovorí aj o reťazcových doménach, avšak z dôvodu zreteľného odlišenia sa reťazcovým hodnotám venujeme osobitne, pretože Grantov prístup zovšeobecňujeme.

⁹⁵ Hoci Grant používa pojem *range* (rozsah), budeme sa držať jednotnej terminológie v celej práci (v súlade s Lipskim).

⁹⁶ Rovnako definujeme aj reprezentáciu množiny. Nech $A \in D_i$ je podmnožina hodnôt obohatenej domény. Potom *reprezentácia množiny* A je množina $\llbracket A \rrbracket =_{df} \{ \llbracket a \rrbracket \mid a \in A \}$.

Reprezentácia Grantovej relácie R je množina relácií $\llbracket R \rrbracket$, ktorej prvky sú jednotlivé *vlastné substitúcie relácie R* . Jednotlivé vlastné substitúcie sú definované nasledovne:

$$S_k =_{df} \cup_{i \in I} \{v_i \mid v_i \in \llbracket \mu_i \rrbracket\},$$

celá reprezentácia je množina všetkých vlastných substitúcií, teda

$$\llbracket R \rrbracket =_{df} \{S_k \mid k \in K\}$$

pričom $R = \{\mu_i \mid i \in I\}$ (I je indexová množina pre tuple relácie R) a K je indexová množina obsahujúca index každej vlastnej substitúcie S_k relácie R . Hoci neuvádzame návod na skonštruovanie jednotlivých relácií S_k , stačí si uvedomiť, že pri implementácii problémy nenastanú, pretože jednotlivé v_i by sme z množín $\llbracket \mu_i \rrbracket$ vyberali nejakým pevne daným spôsobom, napr. podľa fyzického uloženia (cyklom typu „for“), pretože ide o dobre usporiadanú množinu hodnôt, prirodzené čísla.

Výhodou použitia vlastnej substitúcie prvku, tuple resp. tabuľky je, že logika použitá na vyhodnocovanie dotazov, nie je trojhodnotová, ale klasická dvojhodnotová. Nevýhodou je, že už mohutnosti množín $\llbracket a \rrbracket$ (o množinách $\llbracket \mu \rrbracket$ resp. $\llbracket R \rrbracket$ ani nehovoriac) sú obrovské, pre reálne čísla sú tieto množiny nespočítateľné. Uvažovať o praktickom využití sa dá nanajvýš v prípade, ak sa obmedzíme na množinu celých čísel alebo ak zadefinujeme operátory iným spôsobom, ako to robí Grant.

Pomocou pojmov vlastnej substitúcie definujeme aj *true* a *maybe* verzie predikátov. Grant definuje *true* a *maybe* verzie predikátov odlišne od Codda (str. 31), a to práve kvôli tomu, aby naozaj vyjadrovali dolnú a hornú hranicu formuly podľa Lipskeho (str. 12).

Nech P je predikát vzťahujúci sa na položky Grantových tabuliek resp. tuple. Máme na mysli predikáty rovnosti $=$, \neq a medzi číselnými typmi aj predikáty porovnania $<$, \leq , \geq , $>$ resp. operácie \in , \subseteq . Potom predikátu P zodpovedajúci *true*–predikát P_{TRUE} je pravdivý práve vtedy, ak pre všetky vlastné substitúcie jeho premenných resp. tuplí a/alebo tabuliek platí P . Predikátu P zodpovedajúci *maybe*–predikát P_{MAYBE} je pravdivý práve vtedy, ak P platí aspoň pre jednu vlastnú substitúciu jeho premenných resp. tuplí a/alebo tabuliek. Formálne, nech a_1, \dots, a_n sú všetky premenné v $P(a_1, \dots, a_n)$, potom

$$P_{\text{TRUE}}(a_1, \dots, a_n) \Leftrightarrow_{df} P(a'_1, \dots, a'_n) \text{ pre všetky } a'_i \in \llbracket a_i \rrbracket, i \in \{1, \dots, n\},$$

$$P_{\text{MAYBE}}(a_1, \dots, a_n) \Leftrightarrow_{df} \exists a'_1 \in \llbracket a_1 \rrbracket, \dots, \exists a'_n \in \llbracket a_n \rrbracket: P(a'_1, \dots, a'_n)$$

resp. nech $\mu_1, \dots, \mu_n, R_1, \dots, R_m$ sú všetky tuple a relácie v $P(\mu_1, \dots, \mu_n, R_1, \dots, R_m)$, potom

$$P_{\text{TRUE}}(\mu_1, \dots, \mu_n, R_1, \dots, R_m) \Leftrightarrow_{df} P(\mu'_1, \dots, \mu'_n, R_1, \dots, R_m)$$

$$\text{pre všetky } \mu'_i \in \llbracket \mu_i \rrbracket, i \in \{1, \dots, n\},$$

$$P_{\text{MAYBE}}(\mu_1, \dots, \mu_n, R_1, \dots, R_m) \Leftrightarrow_{df} \exists \mu'_1 \in \llbracket \mu_1 \rrbracket, \dots, \exists \mu'_n \in \llbracket \mu_n \rrbracket:$$

$$P(\mu'_1, \dots, \mu'_n, R_1, \dots, R_m).^{97}$$

Ak nevznikne nedorozumenie, budeme P_{TRUE} resp. P_{MAYBE} skracovať na P_T resp. P_M .

Na základe takýchto definícií je možné definovať *true* a *maybe* verzie pre jednotlivé predikáty rovnosti a porovnania (uvažujeme o číselných typoch), napr.

$$a =_T b \Leftrightarrow a, b \text{ sú úplné číselné hodnoty a } a = b,$$

$$a =_M b \Leftrightarrow \llbracket a \rrbracket \cap \llbracket b \rrbracket \neq \emptyset,$$

⁹⁷ Pritom prirodzene $P(\mu'_1, \dots, \mu'_n, R_1, \dots, R_m)$ platí, ak existujú reprezentácie R'_1, \dots, R'_m Grantových relácií R_1, \dots, R_m také, že $P(\mu'_1, \dots, \mu'_n, R'_1, \dots, R'_m)$.

$$a <_T b \Leftrightarrow \sup\{x \mid x \in \llbracket a \rrbracket\} < \inf\{y \mid y \in \llbracket b \rrbracket\},$$

$$a \leq_T b \Leftrightarrow \sup\{x \mid x \in \llbracket a \rrbracket\} \leq \inf\{y \mid y \in \llbracket b \rrbracket\}$$

a podobne, pričom $a, b \in D'$. Poznamenajme, že $a \leq_T b$ nie je to isté ako $a <_T b \vee a =_T b$.

Rovnako definujeme *true* a *maybe* verzie množinových operácií nad hodnotami tuplí, tupľami a reláciami:

$$a \in_T A \Leftrightarrow \text{pre každé } a' \in \llbracket a \rrbracket \text{ platí } a' \in \llbracket A \rrbracket,$$

$$a \in_M A \Leftrightarrow \llbracket a \rrbracket \cap \llbracket A \rrbracket \neq \emptyset,$$

$$\mu \in_T R \Leftrightarrow \text{pre každé } v \in \llbracket \mu \rrbracket \text{ platí } v \in R,$$

$$\mu \in_M R \Leftrightarrow \llbracket \mu \rrbracket \cap \llbracket R \rrbracket \neq \emptyset.$$

Operácie nad tabuľkami sú troch druhov: *true*, *maybe* a *množinovo-teoretické*. *True*-verzie obsahujú len tie tuple, ktoré *musia byť* vo výslednej relácii, *maybe*-verzie obsahujú tie tuple, ktoré *môžu byť* vo výslednej relácii, a napokon *množinovo-teoretické* verzie dostaneme tak, že s Grantovou reláciou zaobchádzame ako s množinou (možno aj s duplikátmi). Načrtne niektoré z nich.⁹⁸

Zjednotenie relácií: postačí množinovo-teoretická verzia, hoci by sme mohli uvažovať o relácii obsahovania na intervaloch ($[i_1, j_1] \in [i_2, j_2] \Leftrightarrow_{df} i_1 \geq i_2 \wedge j_1 \leq j_2$) a v takom prípade zvážiť, či pridať tupľu $[3, 4]$ do tabuľky, kde už tupľa $[1, 5]$ je.

Prienik relácií: prienik intervalov je daný klasicky, teda

$$[i_1, j_1] \cap [i_2, j_2] =_{df} \begin{cases} [i, j], & \text{ak } i := \max(i_1, i_2) < \min(j_1, j_2) =: j, \\ i, & \text{ak } i := \max(i_1, i_2) = \min(j_1, j_2), \\ \emptyset, & \text{ak } \max(i_1, i_2) > \min(j_1, j_2). \end{cases}$$

Rovnako zadefinujeme aj prienik intervalu s regulárnou hodnotou a prienik dvoch regulárnych hodnôt⁹⁹. Položíme:

$$[i, j] \cap k =_{df} [i, j] \cap [k, k]$$

a

$$i \cap j =_{df} [i, i] \cap [j, j].$$

Prienik tuplí je daný nasledovne. Nech μ, ν sú definované nad množinou atribútov W . Ak pre všetky $A \subseteq W$ prienik $\mu[A] \cap \nu[A]$ je neprázdny, potom $\mu \cap \nu =_{df} \tau$, kde $\tau[A] =_{df} \mu[A] \cap \nu[A]$ pre všetky $A \subseteq W$. V opačnom prípade definujeme $\mu \cap \nu =_{df} \emptyset$.

True-prienik relácií: nemôže obsahovať prienik intervalov aj za predpokladu, že existuje, pretože táto informácia je *maybe*-druhu. Hoci napr. $[1, 10] \cap [7, 23] = [7, 10] \neq \emptyset$, predsa nevieme s istotou zaručiť, či prvá hodnota nie je 5 a druhá 13 (v tom prípade je ich

⁹⁸ Poznamenajme, že všeobecné definície pre operátor O tvaru

$$O_T(R_1, \dots, R_n) =_{df} \{\mu \mid \mu \in_T O(R_1, \dots, R_n)\}$$

a

$$O_M(R_1, \dots, R_n) =_{df} \{\mu \mid \mu \in_M O(R_1, \dots, R_n)\}$$

nedávajú správny výsledok (porov. napr. pre \cup).

⁹⁹ pre úplnosť dodávame obe definície

prienik prázdny a ani jedna z hodnôt 5, 13 nepatrí do $[7, 10]$). Preto definujeme prienik len na neintervaloch, a to takto:

$$R \cap_T S =_{df} \{\mu \mid \mu \in R \wedge \mu \in S, \mu \text{ neobsahuje interval}\}.$$

Maybe-prienik naopak bude takéto hodnoty obsahovať. Definujeme ho nasledovne:

$$R \cap_M S =_{df} \{\mu \cap \nu \mid \mu \in R \wedge \nu \in S\}.$$

Príklad 20 ukazuje, že *množinovo-teoretický* prienik môže byť odlišný od oboch spomínaných prienikov.

Príklad 20: Dve Grantové relácie s navzájom rozličnými prienikmi

R		S		$R \cap S$		$R \cap_T S$		$R \cap_M S$	
A	B	A	B	A	B	A	B	A	B
1	2	1	2	1	2	1	2	1	2
[1, 9]	3	[1, 9]	3	[1, 9]	3			[1, 9]	3
[7, 8]	[4, 7]	[5, 8]	6					[7, 8]	6
		8	5					8	5

True/maybe rozdiel iba definujeme:

$$R \setminus_T S =_{df} \{\mu \mid \mu \in R \wedge (\forall \nu \in S: \mu \neq \nu)\},$$

$$R \setminus_M S =_{df} \{\mu \mid \mu \in R \wedge (\exists \nu \in \llbracket \mu \rrbracket: \nu \notin S)\}.$$

True/maybe F-selekcia obsahuje tie tuple, pre ktoré sa výraz F vyhodnotí *true/maybe*.

Rovnako tak aj *true/maybe F-join*. Kartézsky súčin je definovaný štandardne.

Výsledky však nie sú uspokojivé: hoci platia niektoré vzťahy, napr.

$$\llbracket R \cup S \rrbracket = \llbracket R \rrbracket \cup \llbracket S \rrbracket, \llbracket R \cap_M S \rrbracket = \llbracket R \rrbracket \cap \llbracket S \rrbracket,$$

neplatia mnohé iné, napr. každá z nasledovných množín môže byť iná:

$$\llbracket R \setminus S \rrbracket, \llbracket R \rrbracket \setminus \llbracket S \rrbracket, \llbracket R \setminus_T S \rrbracket, \llbracket R \setminus_M S \rrbracket.$$

Obr. 1 v tomto prípade nekomutuje.

4.2.2 Množinové intervalové hodnoty

[Ola–Özsoyoglu, 1993] rozšírili Grantovu myšlienku a uvažujú obohatenie domény D nielen o (jednoduché) intervaly, ale o množiny intervalov. Regulárna hodnota je pritom práve jedna z daného intervalu (exkluzívna disjunktívna informácia).¹⁰⁰ V prípade prirodzených čísel ide vlastne iba o zosťučnenie množinového zápisu. Autori ukázali, že vyhodnocovanie dotazov v nimi navrhnutých modeloch je *zdravé* (nie sú odvodené žiadne nesprávne výsledky), avšak nie *úplné* (všetky správne výsledky sú odvodené). Navyiac, ako ukázali [Chiu–Chen, 1996], tri z piatich modelov navrhnutých v článku [Ola–Özsoyoglu, 1993] nie sú ani zdravé. Preto sa týmto prístupom nebudeme zaoberať, iba pre príklad uvedieme tabuľku. Hodnota $\{[10, 15], [28, 31]\}$ pre Jánov výplatný deň znamená, že dostáva výplatu buď v rozmedzí 10.–15. dňa v mesiaci, alebo medzi 28. a 31. dňa v mesiaci (závisí od solventnosti firmy v strede mesiaca).

¹⁰⁰ Navyiac tupliam priradili „typ“, ktorý okrem príznaku *true* resp. *maybe* (ako ho majú napr. I–tabuľky) obsahuje ďalšie dva typy, označujúce tuple, ktoré môžu obsahovať čiastočnú informáciu a musia/nemusia byť jednoznačne reprezentované.

Príklad 21: Tabuľka s neúplnou informáciou podľa [Ola–Özsoyoglu, 1993]¹⁰¹

Zamestnanec	Výplatný_deň
Dominik	15
Ján	{[10, 15], [28, 31]}
Juraj	{15, 28}

4.3 Reťazcové domény: regulárne výrazy

[Grant, 1980] síce uvažuje aj o reťazcoch, ale iba o obmedzenom spôsobe uchovania čiastočnej informácie o reťazcoch: umožňuje namiesto ľubovoľného znaku dať symbol „{“ označujúci ľubovoľný znak (nad danou abecedou symbolov)¹⁰². Navrhujeme všeobecnejší prístup, a to umožniť obmedzené uchovanie regulárneho výrazu (až na ľubovoľnú iteráciu). Dôvodom je, že mnohé čiastočné poznatky o reťazcových hodnotách sa len ťažko uchovávajú. Napríklad svedok krádeže si spomenie len na posledné dvoj písmenie značky kradnutého auta, ktoré zahliadol, pretože sú („náhodou“) totožné s jeho iniciálkami: táto informácia sa dá pohodlne zapísať regulárnym výrazom. So stĺpcami takýchto hodnôt sa potom ľahšie pracuje a k joinu, kartézskemu súčinu či selekcii stačí poznať narábanie s regulárnymi výrazmi, teda počítanie prieniku, zjednotenia a komplementu, prípadne zret'azenia.

Nasleduje jednoduchá syntax pre regulárne výrazy. Predpokladáme pritom, že (konečná) *abeceda* je tvorená pevne danou množinou symbolov.¹⁰³ Označíme ju $\Sigma = \{a_1, a_2, \dots, a_n\}$. *Jazyk* (nad Σ) je množina reťazcov z abecedy Σ , ktoré sa v kontexte teórie formálnych jazykov nazývajú *slová*.

Regulárny výraz je definovaný nasledovne:

- (1) prázdne slovo (označenie ϵ) je regulárny výraz,
- (2) ľubovoľný znak *abecedy* je regulárny výraz,
- (3) zret'azenie regulárnych výrazov A_1, A_2 (označenie $A_1.A_2$ alebo iba A_1A_2) je regulárny výraz,
- (4) ak A_1, A_2 sú regulárne výrazy, potom aj $(A_1 + A_2)$ je regulárny výraz,¹⁰⁴
- (5) iterácia regulárneho výrazu A (označenie A^*) je regulárny výraz,
- (6) nič iné nie je regulárny výraz.

Používame nasledovnú konvenciu, ktorá približuje teoretické (informatické) označovanie regulárnych výrazov zrozumiteľnejšiemu (programátorskému):

- (1) konkrétny jednotlivý symbol abecedy reprezentujeme ním samotným,
- (2) prázdne slovo reprezentujeme prázdny reťazcom „“ (t.j. dve úvodzovky vedľa seba),

¹⁰¹ vynechali sme stĺpec označujúci typ tuple

¹⁰² ako sa používa napr. v implementáciách lexikálnych analyzátorov (Lex)

¹⁰³ V praxi ide najmä o ASCII/Unicode/ISO-8859-X znakové sady. Znak/reťazce kvôli odlíšeniu od ostatného textu uvádzame odlišným, neproporcionálnym písmom, navyše medzi dve úvodzovky. Znak, ktoré majú definované špeciálne významy (nazývajú sa *wildcards*), sa zobrazujú použitím znaku spätnej lomky (*backslash*, znak „\“) pred daný znak, ako je to bežné z programovacieho jazyka C. Napríklad namiesto znaku „?“ treba použiť reťazec „\?“ , alebo namiesto znaku „)“ reťazec „\)“.

¹⁰⁴ niekedy sa nazýva aj *alternatíva*

- (3) ľubovoľný¹⁰⁵ znak reprezentujeme pomocou „?“ („s?m“ znamená „som“, „sem“ aj „sám“); otáznik „?“ je teda skratkou za regulárny výraz $(a_1 + a_2 + \dots + a_n) = \sum_{a \in \Sigma} a$,
- (4) ľubovoľný reťazec reprezentujeme pomocou „*“ („ročn*“ znamená napr. „ročný“, „ročne“, „ročník“); hviezdička, ktorá nenasleduje za pravou zátvorkou, je skratkou za regulárny výraz $(a_1 + a_2 + \dots + a_n)^*$,
- (5) práve jeden z množiny znakov: zapíšeme možnosti do hranatých zátvoriek [] („[kp]rst“ znamená „krst“ alebo „prst“), čo je len iná forma písania alternatívy, bez znakov plus,
- (6) ľubovoľný znak z daného rozsahu (intervalu): zapíšeme začiatok a koniec intervalu do hranatých zátvoriek a medzi ne dáme pomlčku: [–], pričom ukladáme obmedzenie, že hranice rozsahu musia byť vo vzostupnom poradí,
- (7) na iteráciu používame hviezdičku, avšak musí nasledovať za pravou zátvorkou, ktorou sa končí (dobře uzátvorkovaný) iterovaný výraz.

Ľahko nahliadnuť, že pomocou vyššie uvedenej konvencie sa dá reprezentovať ľubovoľný regulárny výraz a opačne. Navrhujeme používať túto konvenciu. Tiež z nej už vidieť, ako budeme reprezentovať neúplnú informáciu:

- keď nepoznáme *jednotlivé písmenko*, ale vieme, že má konkrétnu pozíciu v slove, na jeho miesto dáme znak ?,
- keď nepoznáme *časť slova*, ktorá má konkrétny počet znakov, na jej miesto analogicky dáme zodpovedajúci počet otáznikov,
- keď nepoznáme *ani len dĺžku neznámej časti slova*, zapíšeme na dané miesto v slove hviezdičku, znak *,
- keď poznáme iba *niekoľko možností* daného písmenka (alebo časť abecedy, kde sa určite nachádza), zapíšeme tieto možnosti do hranatých zátvoriek resp. pomocou intervalu.

Regulárny výraz vlastne zastupuje množinu slov, ktoré ním môžu byť reprezentované (hovoríme, že jazyk $L(A)$ zodpovedá regulárnemu výrazu A). Pokiaľ regulárny výraz reprezentuje jediné slovo¹⁰⁶, považujeme túto hodnotu za úplnú, v opačnom prípade ide o neúplnú informáciu.

Zhrnieme stručne známe výsledky manipulácie s regulárnymi výrazmi podľa teórie jazykov a automatov, ako ju podávajú napr. [Hopcroft–Ullman, 1978]. Tieto výsledky vlastne charakterizujú dynamický pohľad na databázu obsahujúcu neúplnú informáciu pre reťazcové domény vyjadrenú pomocou regulárnych výrazov (vyššie uvedenou konvenciou). Ukazujú, že porovnávanie regulárnych výrazov pomocou prípadného vstavaného „kompilátora“ nespôsobuje ťažkosti.

¹⁰⁵ v zmysle „neznámy“ alebo „nezáleží-na-ňom“

¹⁰⁶ Ako vyplýva z definície, v prípade, že regulárny výraz je daný zretežením symbolov abecedy, jemu zodpovedajúci jazyk obsahuje jediné slovo, a to práve to slovo, ktoré je regulárnym výrazom. Vtedy tieto dva pojmy stotožňujeme. Ľahko sa dá overiť, že regulárny výraz reprezentuje jednoslovný jazyk práve vtedy, keď sám je reprezentovaný slovom (zretežením symbolov abecedy).

Nech A, A_1, A_2 sú regulárne výrazy. Nasledujúce problémy sú triviálne alebo rozhodnuteľné:

- $L(A) \neq \emptyset$, problém prázdnoty jazyka: práve vtedy, keď zodpovedajúci automat s n stavmi akceptuje slovo kratšej dĺžky ako n ,
- $|L(A)| = \infty$, problém nekonečnosti jazyka: práve vtedy, keď zodpovedajúci automat akceptuje slovo l dĺžky $n \leq |l| < 2n$,
- $L_1 = L_2$, problém rovnosti jazykov: práve vtedy, keď $L((L_1 \cap \overline{L_2} \cup (\overline{L_1} \cap L_2)) = \emptyset$,
- $L_1 \subseteq L_2$, problém inklúzie jazykov: práve vtedy, keď $L_2 = L_1 \cup L_2$.

Mnohé databázové systémy poskytujú časť tejto funkčnosti, a to vyhľadávanie reťazcových údajov na základe zadaného regulárneho výrazu (azda s obmedzením na nekonečnú iteráciu). Teória pattern matchingu a vyhľadávania reťazcových informácií (i na čiastočnú zhodu) v databázach sa stále rozvíja, a to aj v súčasnosti.¹⁰⁷ Modelu sa nebudeme podrobnejšie venovať, ostáva ako podnet pre ďalšiu výskumnú prácu.

4.4 Disjunktívna informácia

Ako sme spomínali vyššie, existuje niekoľko druhov disjunktívnej informácie. My sa budeme zaoberať *exkluzívnou* disjunktívnou informáciou a budeme skráteno hovoriť o *disjunktívnej informácii*. Disjunktívna informácia sa zapíše ľahko v dokazovaco-teoretickom uhle pohľadu¹⁰⁸. Keby sme napr. chceli zaznamenať skutočnosť, že dodávateľ Cyril dodáva *perník* alebo *chlieb*, neformálne by sme túto skutočnosť vyjadrujúcu inkluzívny disjunktívny fakt zaznačili nasledovne:

$$\text{Dodáva}(\text{Cyril}, \text{perník}) \vee \text{Dodáva}(\text{Cyril}, \text{chlieb}).$$

Tento istý fakt, interpretovaný exkluzívne, zaznačíme takto:

$$\text{Dodáva}(\text{Cyril}, \text{perník}) \otimes \text{Dodáva}(\text{Cyril}, \text{chlieb}).^{109}$$

Ako sme uviedli vo všeobecnom popise podmnožinovej konštrukcie, dá sa takýto fakt zapísať pomocou všeobecnej tuple¹¹⁰

$$\text{Dodáva}(\text{Cyril}, \{\text{perník}, \text{chlieb}\}).$$

Prehľadom tabuľkových reprezentácií všeobecnej relácie (pre inkluzívnu/exkluzívnu disjunktívnu informáciu) sa budeme zaoberať v nasledovných odsekoch.

4.4.1 I-tabuľky

I-tabuľky zaviedli a skúmali Liu a Sunderraman vo svojej práci [Liu–Sunderraman, 1990]. *I-tabuľka* (z angl. *indefinite table*) obsahuje okrem regulárnych hodnôt aj množinovú notáciu zapísanú exkluzívnu disjunktívnu informáciu. I-tabuľky sú zložené z troch komponentov:

1. *definite component*: obsahuje normálne tuple,
2. *indefinite component*: obsahuje všeobecné tuple,
3. *maybe component*: obsahuje *maybe* tuple (porov. str. 4).

¹⁰⁷ Porov. napr. štúdiu [Marcinčák–Matula–Semanišin, 1999] prednesenú na nedávnej konferencii UNINFOS '99.

¹⁰⁸ pozri časť 2.6, str. 14

¹⁰⁹ Symbol \otimes znamená logickú spojku xor (exkluzívny or).

¹¹⁰ nazýva sa aj *indefinite tuple*

V I–tabuľkách je teda uložená dvojaká informácia:

1. *sure* (istá), reprezentovaná prvými dvoma komponentmi,
2. *maybe*, reprezentovaná tretím komponentom.

Príklad 2 s pridanou všeobecnou tupľou *Dodáva*(Cyril, {*perník*, *chlieb*}) je reprezentovaný nasledujúcou I–tabuľkou (bez *maybe* komponentu).

Príklad 22: Stručne a precízne (s dvoma komponentmi) zapísaná I–tabuľka

Dodáva		Dodáva	
Dodávateľ	Pečivo	Dodávateľ	Pečivo
Anton	<i>perník</i>	Anton	<i>perník</i>
Braňo	<i>chlieb</i>	Braňo	<i>chlieb</i>
Cyril	<i>vianočka</i>	Cyril	<i>vianočka</i>
Anton	<i>chlieb</i>	Anton	<i>chlieb</i>
Cyril	{ <i>perník</i> , <i>chlieb</i> }		
		Cyril	<i>perník</i>
		Cyril	<i>chlieb</i>

Liu a Sunderraman navrhli operátor *REDUCE* (s časovou zložitou $O(n \cdot \log n)$, kde n označuje počet tuplí), ktorý odstraňuje redundantnú informáciu obsiahnutú v I–tabuľke. Dá sa povedať, že tým sa každá I–tabuľka redukuje do normálneho tvaru. Pôvodom tejto redundancie však je *maybe* komponent tabuľky, preto sa redukciou nebudeme bližšie zaoberať.

4.4.2 M–tabuľky

I–tabuľky však dokážu reprezentovať iba typ disjunktívnej informácie, kde sa vyskytuje jediný predikát v disjunkcii, teda napr.

$$P(\mu_1) \vee P(\mu_2) \vee \dots \vee P(\mu_n),$$

nie však informáciu typu

$$P_1(\mu_1) \vee P_2(\mu_2) \vee \dots \vee P_n(\mu_n),^{111}$$

kde P_1, P_2, \dots, P_n môžu byť rôzne predikáty. Pritom P resp. P_i zodpovedá jednej relácii.

Liu a Sunderraman¹¹² sa venovali rozšíreniu I–tabuliek, aby dokázali reprezentovať aj takúto informáciu. Rozšírené I–tabuľky sa nazývajú *M–tabuľky* (z angl. *multitable*).

„Tajomstvo“ tohoto rozšírenia spočíva v tom, že kým jednotlivé relácie P_i sú v klasickom relačnom modeli uchovávané zvlášť (a teda tabuľka zodpovedá relácii a databáza je množina tabuliek), v M–tabuľkách sú všetky relácie uchovávané spolu v jednej relácii (teda tabuľka zodpovedá množine relácií, databáza je síce tiež množina tabuliek, avšak môže sa stať, že aj pri množstve relácií zodpovedá databáze jediná M–tabuľka).

Príklad 23: M–tabuľka pre relácie *Ujo*, *Teta*

Nech relácia *Otec*(a, b) resp. *Brat*(a, b) znamená, že a je otec b čka resp. a je bratom b čka. Nasledujúca M–tabuľka *Otec–Brat* reprezentuje tieto čiastočné poznatky:

- (1) *Otec*(Tomáš, Stanislav),
- (2) *Otec*(Radoslav, Ján) \vee *Otec*(Radoslav, Dominik),

¹¹¹ V prípade exkluzívnej informácie namiesto symbolov \vee sú \otimes .

¹¹² [Liu–Sunderraman, 1989] a [Liu–Sunderraman, 1991]

- (3) *Brat*(Marek, Tomáš),
 (4) *Brat*(Ľubor, Ján) \vee *Brat*(Ľubor, Dominik),
 (5) *Otec*(Martin, Ján) \vee *Brat*(Martin, Ján),
 (6) *Otec*(Miroslav, Tomáš) \vee *Otec*(Miroslav, Stanislav)
 \vee *Brat*(Miroslav, Tomáš) \vee *Brat*(Miroslav, Stanislav)

Otec-Brat

<i>Otec</i>	<i>Osoba</i>	<i>Brat</i>	<i>Osoba</i>
Tomáš	Stanislav		
Radoslav	Ján		
Radoslav	Dominik		
		Marek	Tomáš
		Ľubor	Ján
		Ľubor	Dominik
Martin	Ján	Martin	Ján
Miroslav	Tomáš	Miroslav	Tomáš
Miroslav	Stanislav	Miroslav	Stanislav

M–tabuľky správne modelujú obidve hranice externej interpretácie dotazov.

4.4.3 MPV–tabuľky

Nazývajú sa tiež *tabuľky so značkovateľnými čiastočnými hodnotami*¹¹³, pretože čiastočná informácia je v nich reprezentovaná množinami možných hodnôt (exkluzívne), ktoré sú značkovateľné. Obsahuje tiež stĺpec *status*, v ktorom je hodnota *true* alebo *maybe* podľa toho, či sa jedná o *true*-tupľu alebo *maybe*-tupľu. Príklad 24 ukazuje jednoduchú MPV–tabuľku. Obsahuje čiastočnú informáciu, že nie je úplne jasné, ktoré z mien Andrej resp. Ondrej je správne¹¹⁴. Martin a Andrej (resp. Ondrej) študujú ten istý odbor, ktorý je buď *informatika*, alebo *fyzika*. Marián študuje tiež informatiku alebo fyziku, lenže nemusí to byť ten istý odbor, ako Martin a Andrej (resp. Ondrej). Napokon Pavol študuje matematiku alebo fyziku.

Príklad 24: Tabuľka so značkovateľnými čiastočnými hodnotami

Študenti

Meno	Odbor	<i>status</i>
Ján	<i>informatika</i>	<i>true</i>
Martin	{ <i>informatika</i> , <i>fyzika</i> } _x	<i>true</i>
Pavol	{ <i>matematika</i> , <i>fyzika</i> } _y	<i>true</i>
{Andrej, Ondrej} _u	{ <i>informatika</i> , <i>fyzika</i> } _x	<i>true</i>
Marián	{ <i>informatika</i> , <i>fyzika</i> } _z	<i>true</i>

Ako sme spomínali už v časti 3.4.2 venovanej C–tabuľkám (str. 37), C–tabuľky sú tiež schopné reprezentovať disjunktívnu informáciu, a to dokonca so značkovateľnými

¹¹³ angl. *tables with marked partial values*

¹¹⁴ pretože ide o dvojtvár, meniny majú v ten istý deň, hoci Ondrej sa postupne na úkor formy mena Andrej vytráca

čiasťočnými hodnotami. Príklad 25 ukazuje, ako možno predchádzajúcu MPV–tabuľku previesť do C–tabuľky. Ako vidieť, klasická C–tabuľka (keď výraz uvedený v podmienkovom stĺpci nemôže obsahovať *maybe*) nedokáže rozlíšiť medzi *maybe* a neúplnou informáciou. Pridanie *maybe* hodnoty pre podmienkové výrazy obsiahnuté v stĺpci *con* tento rozdiel dokáže čiastočne simulovať; nie však úplne: tupľa

$\langle \text{Peter, manažment, true} \rangle$

bude reprezentovaná rovnako v MPV– aj C–tabuľke, avšak nedokážeme rozlíšiť tuple

$\langle \text{Marián, } \{ \text{informatika, fyzika} \}_z, \text{true} \rangle$

a

$\langle \text{Marián, } \{ \text{informatika, fyzika} \}_z, \text{maybe} \rangle,$

pretože obidve sú v C–tabuľke reprezentované tupľou

$\langle \text{Marián, } z, z = \text{informatika} \vee z = \text{fyzika} \rangle.$

Príklad 25: C–tabuľka zodpovedajúca predošlej MPV–tabuľke

Študenti'

Meno	Odbor	con
Ján	<i>informatika</i>	<i>true</i>
Martin	<i>x</i>	$x = \text{informatika} \vee x = \text{fyzika}$
Pavol	<i>y</i>	$y = \text{matematika} \vee y = \text{fyzika}$
<i>u</i>	<i>x</i>	$(u = \text{Andrej} \vee u = \text{Ondrej}) \wedge (x = \text{informatika} \vee x = \text{fyzika})$
Marián	<i>z</i>	$z = \text{informatika} \vee z = \text{fyzika}$

4.4.4 Pv–tabuľky

Cieľom práce [Chiu–Chen, 1995]¹¹⁵ bolo odstrániť časť nedostatkov MPV–tabuliek. Všimnime si totiž, čo sa stane s tabuľkou *Študenti* po dotaze, konkrétne selekcii „študenti, ktorí študujú odbor *informatika*“. Príklad 26 tento výsledok ukazuje, a to pre obidve tabuľky (MPV–tabuľku *Študenti*, jej zodpovedajúcu C–tabuľku *Študenti'*).

Príklad 26: Selekcia nad tabuľkami *Študenti*, *Študenti'*

$\sigma_{\text{Odbor} = \text{informatika}}(\text{Študenti})$

Meno	Odbor	status
Ján	<i>informatika</i>	<i>true</i>
Martin	$\{ \text{informatika} \}_x$	<i>maybe</i>
$\{ \text{Andrej, Ondrej} \}_u$	$\{ \text{informatika} \}_x$	<i>maybe</i>
Marián	$\{ \text{informatika} \}_z$	<i>maybe</i>

$\sigma_{\text{Odbor} = \text{informatika}}(\text{Študenti}')$

Meno	Odbor	con
Ján	<i>informatika</i>	<i>true</i>
Martin	<i>x</i>	$x = \text{informatika}$
<i>u</i>	<i>x</i>	$(u = \text{Andrej} \vee u = \text{Ondrej}) \wedge (x = \text{informatika})$
Marián	<i>z</i>	$z = \text{informatika}$

¹¹⁵ aj predošlých prác týchto autorov

Vo výsledných tabuľkách nie je vidieť, ktorý atribút prispieva k *maybe* charakteru informácie (ako je tomu konieckoncov v MPV– aj C–tabuľkách celkovo). Ďalej, v oboch tabuľkách sa stratí informácia, že Martin, Marián a Andrej (resp. Ondrej) možno študujú fyziku.

Cieľom Pv–tabuliek je preklenúť tieto nedostatky. Pv–tupľa je dvojica, ktorá obsahuje množinu premenných a množinu hodnôt, pričom práve jedna z nich v realite platí. Ak množina premenných je prázdna, ide o *úplnú informáciu*, v opačnom prípade o čiastočnú informáciu. Pri vytvorení Pv–tabuľky (resp. novej Pv–tuple) je množina premenných jednoprvková. Namiesto

$$\langle \langle \{u\}, \{\text{Andrej, Ondrej}\} \rangle, \langle \{x\}, \{\text{informatika}\} \rangle \rangle$$

píšeme skrátené

$$\langle (u, \text{Andrej Ondrej}), (x, \text{informatika}) \rangle.$$

Pomocou Pv–tabuliek sa dá reprezentovať aj *maybe* informácia a v rámci nej identifikovať atribúty, ktoré vlastne dávajú tejto informácii *maybe* charakter, avšak my sa *maybe*-Pv–tupľami nebudeme zaoberať.

Druhá odlišnosť je v tom, že hodnoty v množine hodnôt môžu byť aj „negatívne“, inými slovami, zahrnutá je tiež negatívna informácia (nad danou hodnotou je pruh). Napr. výsledok joinovania tuplí

Pavol	$\{\text{matematika, fyzika}\}_y$
-------	-----------------------------------

(zapišeme ju Pv–tupľou $\langle (\emptyset, \text{Pavol}), (y, \text{matematika fyzika}) \rangle$)

a

Marián	$\{\text{informatika, fyzika}\}_z$
--------	------------------------------------

(zodpovedá jej Pv–tupľu $\langle (\emptyset, \text{Marián}), (x, \text{informatika fyzika}) \rangle$)

nad atribútom Odbor je tupľu, ktorá má v hodnote atribútu Odbor uvedenú hodnotu

$$(x \ y, \overline{\text{matematika}} \ \overline{\text{informatika}} \ \overline{\text{fyzika}}),$$

ktorá zachytáva aj fakt, že hodnota premennej x resp. y síce môže byť *matematika*, *informatika* alebo *fyzika*, avšak jedine v prípade, že to bude *fyzika*, nebude táto tupľu vylúčená z odpovede.

Príklad 27: Pv–tabuľka „Študenti“ a selekcia na informatiku

Študenti

Meno	Odbor
$(\emptyset, \text{Ján})$	$(\emptyset, \text{informatika})$
$(\emptyset, \text{Martin})$	$(x, \text{informatika fyzika})$
$(\emptyset, \text{Pavol})$	$(y, \text{matematika fyzika})$
$(u, \text{Andrej, Ondrej})$	$(x, \text{informatika fyzika})$
$(\emptyset, \text{Marián})$	$(z, \text{informatika fyzika})$

$\sigma_{\text{Odbor} = \text{informatika}}(\text{Študenti})$

Meno	Odbor
$(\emptyset, \text{Ján})$	$(\emptyset, \text{informatika})$
$(\emptyset, \text{Martin})$	$(x, \text{informatika} \ \overline{\text{fyzika}})$
$(u, \text{Andrej, Ondrej})$	$(x, \text{informatika} \ \overline{\text{fyzika}})$
$(\emptyset, \text{Marián})$	$(z, \text{informatika} \ \overline{\text{fyzika}})$

Príklad 27 pre ilustráciu uvádza výsledok selekcie $\sigma_{\text{Odbor} = \text{informatika}}(\text{Študenti})$. Zachováva aj časť informácie, že napr. Martin môže študovať aj fyziku (v tom prípade, ako naznačuje údaj *fyzika*, však nebude zahrnutý vo výslednej odpovedi. Logická formula, ktorá zodpovedá hodnote atribútu Odbor „Martinovej“ P_v-tuple, je

$$(x = \text{informatika} \vee x = \text{fyzika}) \wedge (x \neq \text{fyzika}).$$

Hoci táto formula je logicky ekvivalentná formule $x = \text{informatika}$, jej sémantika je bohatšia. Obsahuje totiž aj informáciu, že Martin eventuálne študuje fyziku.

Chiu a Chen ukázali, že relačné operácie zadané nad P_v-tabuľkami sa vyhodnocujú správnym spôsobom, a to na základe vlastností, ktoré vyplývajú z rôznych vzťahov medzi P_v-tupľami (tieto vzťahy pomáhajú pri odstraňovaní redundancií v P_v-tabuľkách).

V článku [Chiu–Chen, 1995] tiež autori ukázali, ako zhusteným spôsobom reprezentovať výsledok zodpovedania dotazov. Napokon autori navrhli logicky ekvivalentnú transformáciu P_v-tabuliek do C-tabuliek s tým, že – vďaka obmedzeniu C-tabuliek – sa zotrie rozdiel medzi neúplnou informáciou a *maybe* informáciou.

4.4.5 Príklad: interpretácia dotazov

V nasledujúcom príklade¹¹⁶ ilustrujeme základné koncepty všeobecného prístupu Lipskeho a tiež sa dotkneme externej a internej interpretácie dotazov. Uvažujeme o všeobecnej množinovej konštrukcii pre exkluzívnu disjunktívnu informáciu. Nech databáza obsahuje štyri objekty *a*, *b*, *c*, *d*. Nech množina objektov, o ktorých vieme, že sú červené, pozostáva len z prvku *a*, pričom jediný objekt, o ktorom vieme, že nie je červený, je *d* (teda nevieme, či *b* a *c* sú alebo nie sú červené). Potom *externou interpretáciou* dotazu

červené? := „zoznam všetkých červených objektov“

je množina všetkých objektov, ktoré sú *skutočnosťou* (v realite, teda nie v systéme, ktorý vypovedá o objektoch, ale v skutočnom svete, ktorého sú objekty prvkami) červené.

Môže to byť teda množina {*a*}, {*a*, *b*}, {*a*, *c*} alebo {*a*, *b*, *c*}. Samozrejme informácia obsiahnutá v databáze nie je dostatočná na to, aby sme mohli rozhodnúť, ktorá z týchto množín to je. V tomto zmysle je naša interpretácia „externá“ vzhľadom k systému.¹¹⁷

V našom príklade

$$\begin{aligned} \llbracket \text{červené?} \rrbracket_* &= \{a\}, \\ \llbracket \text{červené?} \rrbracket^* &= \{a, b, c\}. \end{aligned}$$

Je ľahké vypočítať $\llbracket Q \rrbracket_*$ a $\llbracket Q \rrbracket^*$, keď *Q* vyjadruje elementárnu vlastnosť¹¹⁸ (napr. byť červený), pretože takáto informácia sa väčšinou nachádza v databáze. Problém však nastáva v prípadoch, keď je dotaz (napr. boolovskou) kombináciou elementárnych vlastností. Vtedy nám nestačí preskúmať databázu, potrebujeme hodnoty $\llbracket Q \rrbracket_*$ a $\llbracket Q \rrbracket^*$ odvodiť (vypočítať).

¹¹⁶ Pozri [Lipski, 1981]. Tento príklad pochádza z [Jaegermann, 1975].

¹¹⁷ Naopak, ako sme spomínali, *interná interpretácia* dotazov sa týka informácií uložených v systéme, nie o svete samotnom.

¹¹⁸ *Elementárna vlastnosť* je vlastnosť prvku databázy, ktorá sa dá priamo zistiť (v relačnom modeli kontrolou stĺpca) z databázy.

Na objasnenie tejto ťažkosti uvažujme nasledujúci príklad. Predstavme si lekársku databázu, ktorá obsahuje údaje o krvnej skupine množstva pacientov resp. darcov krvi (teda krvné skupiny H^{119} , A, B, AB). Niekedy sa stane, že máme iba čiastočnú informáciu o krvnej skupine jedinca, napríklad odvodením z krvných skupín rodičov alebo z čiastkových testov. Napríklad ak jeden z rodičov má krvnú skupinu AB, potom dieťa nemôže mať skupinu H; ak rodičia majú skupiny A a H, potomok môže mať krvnú skupinu jedine A alebo H. Predstavme si, že hľadáme kandidátov na transfúziu skupiny A alebo H. Vtedy vyhovuje aj jedinec, ktorého rodičia majú krvné skupiny A a H, hoci jeho krvnú skupinu presne nevieme. Stačí vedieť, že jeho krvná skupina je v množine $\{A, H\}$. Inými slovami, každá takáto osoba je v množine

$$[[\text{krvná_skupina} = A \text{ alebo } \text{krvná_skupina} = H]]^*$$

a teda v externej interpretácii dotazu „ $\text{krvná_skupina} = A \text{ alebo } \text{krvná_skupina} = H$ “. Samozrejme, nie každá takáto osoba musí byť v množine

$$[[\text{krvná_skupina} = A]]^* \cup [[\text{krvná_skupina} = H]]^*$$

(napríklad náš kandidát, ktorého krvnú skupinu vieme vymedziť iba množinou $\{A, H\}$). Tento príklad ukazuje, že

$$[[Q_1 \text{ alebo } Q_2]]^*$$

obyčajne nemôže byť počítané ako zjednotenie

$$[[Q_1]]^* \cup [[Q_2]]^*.$$

Podobný princíp platí pre áno-nie (zist'ovacie) dotazy. Pokračujme v príklade s farebnými objektmi, pričom trochu rozšírime našu databázu. Nech obsahuje prvky $\{a, b, c, d, e\}$, pričom farba žiadneho z nich nie je presne známa, ale máme čiastočnú znalosť o farbách takúto:

možné červené objekty: a, b
 možné modré objekty: a, b, c, d, e
 možné zelené objekty: a, c
 možné biele objekty: a, b, c
 možné čierne objekty: b, c

Pritom vylučujeme možnosť, že objekt má inú farbu, aká je uvedená v našom zozname.¹²⁰ Inými slovami, objekt b môže mať červenú, modrú, bielu alebo čiernu farbu (nemôže mať zelenú), ale nemôže mať ani ružovú farbu, o ktorej naša databáza nehovorí.

Teraz uvažujme dotaz $Q :=$ „Máme v našej zbierke objekty všetkých farieb?“ Môžeme naň dať zodpovedajúcu odpoveď, hoci nepoznáme presne farby jednotlivých objektov, ba dokonca ani to, či objekty nemajú náhodou inú farbu. Pretože sú len tri objekty a, b a c na reprezentáciu štyroch farieb: červenej, zelenej, bielej a čiernej. Teda máme $[[Q]]^* =$ „Nie“.

¹¹⁹ Častejšie sa používa klasické označenie 0 (nula).

¹²⁰ Klasický predpoklad pre (exkluzívnu) disjunktívnu informáciu: v realite má daný objekt farbu práve z množiny, ktorá neúplnú informáciu reprezentuje.

Interná interpretácia poukazuje na našu znalosť o objekte. Interná reprezentácia dotazu Q sa označuje $\llbracket Q \rrbracket$ a znamená „z databázy sa dá odvodiť, že prvok spĺňa Q “. V prípade úplnej informácie sa interná reprezentácia správa normálne ako prirodzená reprezentácia. Definujeme totiž

$$\begin{aligned}\llbracket Q_1 \text{ a } Q_2 \rrbracket &= \llbracket Q_1 \rrbracket \cap \llbracket Q_2 \rrbracket \\ \llbracket Q_1 \text{ alebo } Q_2 \rrbracket &= \llbracket Q_1 \rrbracket \cup \llbracket Q_2 \rrbracket \\ \llbracket \text{not } Q \rrbracket &= \text{komplement k } \llbracket Q \rrbracket\end{aligned}$$

Treba pripomenúť, že Q znamená „vieme, že má/majú vlastnosť Q “ len v prípade, že sa jedná o elementárnu vlastnosť. Pre zložené vlastnosti platí, že treba presne kopírovať definíciu v slovnom vyjadrovaní významu. Napríklad

červená **alebo** modrá

znamená

vieme, že je červená alebo vieme, že je modrá,

nie však

vieme, že je [červená alebo modrá].

Uvažujme naše farebné objekty, množinu ktorých opäť rozšírime. Nech naša báza dát obsahuje prvky $\{a, b, c, d, e, f, g\}$, pričom čiastočná znalosť o farbách je nasledovná:

možné červené objekty: b, c, f, g

možné biele objekty: a, b, c, d

možné čierne objekty: b, c, e, f

Lahko môžeme nahliadnuť, že pre dotaz „biele **alebo** čierne“ platia nasledujúce rovnosti:

$$\begin{aligned}\llbracket \text{biele alebo čierne} \rrbracket_* &= \{a, d, e\} \\ \llbracket \text{biele alebo čierne} \rrbracket^* &= \{a, b, c, d, e, f\} \\ \llbracket \text{biele alebo čierne} \rrbracket &= \llbracket \text{biele} \rrbracket \cup \llbracket \text{čierne} \rrbracket = \{a\} \cup \{e\} = \{a, e\}\end{aligned}$$

Za povšimnutie stojí, že nie vždy platí $\llbracket Q \rrbracket \subseteq \llbracket Q \rrbracket_*$, napríklad

$$\begin{aligned}\llbracket \text{not biele} \rrbracket &= \{b, c, d, e, f, g\}, \\ \llbracket \text{not biele} \rrbracket_* &= \{e, f, g\}.\end{aligned}$$

Ako sme už spomínali, interná reprezentácia nie je veľmi zaujímavá, pokiaľ používame taký istý jazyk ako v prípade úplnej informácie. Avšak môžeme obohatiť jazyk dotazov o unárny operátor **určite** s nasledujúcou interpretáciou: $\llbracket \text{určite } Q \rrbracket$ je množina objektov, ktoré patria do $\llbracket Q \rrbracket$ v každom možnom zúplnení¹²¹ nášho súčasného poznania.

Aký je vzťah medzi externou a internou interpretáciou? Videli sme, že v externej interpretácii dotaz vypovedá priamo o skutočnom svete. Keďže však informácia

¹²¹ Hovoríme, že systém S_1 je *rozšírenie* systému S_2 a zapisujeme $S_1 \geq S_2$, ak pre každý atribút i a každý objekt x platí $\beta_1^i(x) \subseteq \beta_2^i(x)$, pričom $\beta^i(x)$ znamená množinu všetkých hodnôt atribútu i objektu x , ktoré sú zachytené v danom systéme. Systém S_1 je *zúplnenie* systému S_2 , ak pre každé rozšírenie S systému S_1 platí $S = S_1$. Neformálne: lepšia, väčšia, úplnejšia informácia o objekte (*lepšia aproximácia informácie*) znamená zmenšenie neurčitosti, neúplnosti o atribútoch objektu, teda vlastne zmenšenie množiny možných hodnôt atribútu/atribútov. Informácia je úplná, keď k nej už nemožno nič viac pridať. Aproximatívnu informáciou sa zaoberá napr. [Libkin, 1994].

v systéme o vonkajších vlastnostiach objektov je vo všeobecnosti neúplná, externá interpretácia dotazu nie je zväčša prístupná pre systém, a teda ani pre používateľa. Informácia obsiahnutá v systéme nám umožňuje určiť len určité hranice externej interpretácie. Tieto hranice sa študujú najmä v článku [Lipski, 1979]. Naproti tomu, naša informácia o vnútornej vlastnosti objektu je pochopiteľne úplná (pretože vieme presne, čo vieme). Interná interpretácia termu nie je nič iné ako množina objektov, pre ktoré informácia obsiahnutá v systéme spĺňa podmienky vyjadrené týmto termom. Voľne povedané, interná reprezentácia termu je ekvivalentná externej interpretácii tohto termu v „umelej realite“, v ktorej každá externá vlastnosť vyjadrená deskriptorom $\langle i, A \rangle$ je nahradená vlastnosťou „systému je známe, že hodnota atribútu i v A je takáto“.¹²²

Hoci nás vždy zaujíma informácia o skutočnosti, (ako používatelia databázového systému) máme k dispozícii vždy len informácie obsiahnuté v systéme, a nie vždy z nich vieme presne dedukovať informácie o realite – vieme však poskytnúť hranice, v ktorých sa pohybujú. Interná reprezentácia navyše umožňuje správcovi databázy alebo ľuďom zodpovedným za zber údajov a napĺňanie bázy dát kontrolovať, či sú v systéme podstatné „diery“ a nakoľko treba zmeniť model reality ponúkaný systémom, aby odrážal realitu vernejšie.

Grantov príklad [Grant, 1977] uvažuje aj Lipski, keď ilustruje použitie externej a internej interpretácie. Uvažujme opäť reláciu $R = R(\text{Id}, \text{Meno}, \text{Vek}, \text{Mesto})$, term S daný nasledovne:

$$S = (R[(\text{Meno} = \text{'John'} \text{ and } \text{Vek} = 50) \text{ or } (\text{Vek} \neq 50 \text{ and } \text{Mesto} = \text{'Gainesville'})])[\text{Id}]$$

a napokon štvoricu $\langle 23, \text{John}, \text{NULL}, \text{Gainesville} \rangle$, ktorá patrí do R . V Lipskeho modeli zodpovedá tejto tupli objekt x , pre ktorý platí

$$\begin{aligned}\beta_{\text{Id}}(x) &= \{23\} \\ \beta_{\text{Meno}}(x) &= \{\text{John}\} \\ \beta_{\text{Vek}}(x) &= D_{\text{Vek}} = \text{množina možných hodnôt Veku} \\ \beta_{\text{Mesto}}(x) &= \{\text{Gainesville}\}.\end{aligned}$$

Pomocou teorém článku [Lipski, 1979] sa dá ukázať, že $x \in \llbracket S \rrbracket_*$ a tiež $x \in \llbracket S \rrbracket^*$.

¹²² angl. *known in the system to have the value of attribute i in A*

5 Logické aspekty neúplnej informácie

„Tí, ktorí robia chyby,
mýlia sa z nedostatku poznania.“
Platón¹²³

5.1 Pohľady (views)

Vzhľadom na rôznorodosť informácií uchovávaných v databázových systémoch je potrebné, aby sa informácie prezentovali rozličným spôsobom. V etape analýzy a návrhu informačného systému sa analytici a návrhári zamýšľajú nad rôznymi úlohami, ktoré používatelia systému môžu zastávať. Je úlohou *konceptuálneho modelovania* čo najvernejšie zachytiť realitu (pozri ďalej). Pritom jeden (fyzický) používateľ prichádza do styku so systémom ako potenciálne viac (logických) používateľov¹²⁴, napr. manažér firmy jednak ako riadiaci pracovník, jednak ako bežný používateľ komunikačného systému. Pre rozličných logických používateľov potom existujú odlišné reprezentácie aj tých istých údajov, nazývané *pohľady*. Práve pohľady (angl. *views*) tvoria vrchnú vrstvu v *trojschémovej architektúre* (ANSI). V nasledujúcich príkladoch sa nevyhneme uvažovaniu na úrovni konceptuálneho modelovania či fyzického návrhu systému, avšak tvoria prirodzený obal na presnejšie vyjadrenie vplyvu uvažovania o pohľadoch.

Uvažujme jednoduchú databázovú schému s atribútmi

Meno, Adresa, Telefón, Oddelenie, Platová_trieda, Základ_Platu, Príplatky, Plat,

pričom atribút *Platová_trieda* označuje *internú podnikovú* platovú triedu, ktorá určuje základ platu (atribút *Základ_platu*) potenciálne neúplne (platová trieda č. 7 teda môže znamenať interval [1100, 1300]¹²⁵, základ platu teda môže byť napr. 1234). *Plat* je určený na základe základu závislého čiastočne od platovej triedy, a príplatku. *Príplatok* má právo určovať a meniť jedine vedúci oddelenia, platovú triedu určuje kvalifikácia zamestnanca (teda menia a určujú ju pracovníčky personálneho oddelenia) a napokon základ platu mení na príkaz vedúceho pracovníčka PO. Napokon poznamenajme, že primárnym kľúčom je interný atribút *id*.

Napokon uvažujme, ktoré atribúty môže vidieť jednotlivý používateľ systému. *Meno, Adresa, Telefónne_číslo* a *Oddelenie* sú verejné pre všetkých zamestnancov.

Platová_trieda, Základ_Platu, Príplatky a *Plat* sú viditeľné len pre

- zamestnanca samotného (o ktorého sa jedná),
- pracovníčky PO,
- vedúceho oddelenia (v ktorom daný zamestnanec pracuje), príp. šéfa firmy.¹²⁶

Nasleduje neformálne (syntaxou jazyka C) zapísaná časť kódu, ktorá má na starosti sprístupnenie hodnôt atribútov:

¹²³ Platón: *Protagoras*, I, 357d. (vyd. ISE, ed. OIKÚMENÉ, Praha 1994)

¹²⁴ V modelovacom jazyku UML sa logický užívateľ nazýva *aktor*.

¹²⁵ uvažujeme vymyslený príklad v bližšie nešpecifikovanej mene

¹²⁶ Neuvažujeme o tom, že zamestnanec môže byť zamestnaný vo viacerých oddeleniach firmy. Taktiež neuvažujeme o tom, že pracovníčka PO môže síce údaje o Príplatku čítať, nie však modifikovať.

```

get_data(Data data)
{
    ...
    SQL_GET(T); // nejaké načítanie danej tuple T
    ...
    data.meno = T.Meno;
    data.adresa = T.Adresa;
    data.telefon = T.Telefón;
    if((user.id == data.id) ||
        (user.role == pers_dept) ||
        (user.role == chief && user.dept == data.oddelenie))
    {
        data.plat_trieda = T.Platová_Trieda;
        data.zaklad_platu = T.Základ_platu; // prípadne upravený podľa plat. triedy
        data.priplatky = T.Príplatky;
        data.plat = T.Plat; // tu môže byť aj výpočet
    }
    else
    {
        data.plat_trieda = VIEW_NULL;
        data.zaklad_platu = VIEW_NULL;
        data.priplatky = VIEW_NULL;
        data.plat = VIEW_NULL;
    }
    ...
}

```

Pozrime sa na uvažovaný príklad podrobne. Ak používateľ nemá dovolené vidieť platové náležitosti, naplnia sa dané premenné špeciálnou hodnotou `VIEW_NULL`, ktorá znamená práve spomínané: nejde o klasický null v zmysle **un** resp. **ne** pre danú tabuľku (reláciu) databázy, ale ide vlastne sémanticky o **ni** null v zmysle pohľadu na tabuľku. Inými slovami, pre neautorizovaného používateľa hodnota atribútu *Príplatky* neexistuje, nemá (mať) zmysel. Prirodzene, spracovanie údajov `VIEW_NULL` môže mať rozličnú podobu, a to napr.

- v aplikáciách s grafickým používateľským rozhraním zašednutie polí vo formulári či vyplnenie daného miesta textom typu „Nie je umožnené vidieť pre tento druh pohľadu“ (najjednoduchšie riešenie),
- nezobrazenie daných polí.

Niekedy je vhodnejší druhý spôsob (pri individuálnom výpise zamestnancov), inokedy prvý spôsob (pri spoločnom výpise typu „browse“ všetkých zamestnancov).

Avšak nemožno zabudnúť, že s danou informáciou sa potenciálne pracuje, preto treba zvážiť, ktorý prístup zvoliť:

- zvoliť hodnotu `VIEW_NULL` tak, aby sa s ňou dalo regulárne narábať, pričom sa nebudeme (v ďalšom) starať o to, či hodnota bola `VIEW_NULL` alebo regulárna hodnota (v prípade platu hodnota 0, nula),
- zvoliť zvláštnu hodnotu `VIEW_NULL` a podľa toho, ako bude zvolená, s ňou narábať: v prípade platu upraviť sémantiku všetkých funkcií narábajúcich s platom tak, aby vedeli počítať aj s hodnotou `VIEW_NULL`.

Druhý prístup je náročnejší, avšak formálne čistejší. Takýmto spôsobom sa dajú implementovať napríklad Coddove null hodnoty, je potrebné však predefinovať operátor rovnosti na dátových typoch, obsahujúcich null, a rozšíriť testovanie o Coddovu pravdivostnú hodnotu *maybe*.

5.2 Konceptuálne modelovanie

Konceptuálny model systému zahŕňa celú jeho „logiku“, teda logický návrh databáz, základné postupy pri používaní systému (tiež sa hovorí o *scenároch*), identifikujú sa logickí používatelia a pomocou modelovacieho jazyka (napr. SSADM, OMT, Booch, OOSE, UML) sa navrhuje štruktúra systému nezávislá na implementačnom prostredí a na hardvérových charakteristikách. Konceptuálny model sa vytvára na základe analýzy a požiadaviek zákazníka. Práve v tejto etape je treba zvažovať, pre ktoré atribúty uvažovaných objektov (entít) pripustíme, aby neboli definované, resp. aby obsahovali nullové hodnoty. Vzhľadom na tému tejto práce je potrebné uvažovať o nasledovnom:

- Ktoré atribúty môžu ako hodnoty obsahovať null hodnoty? Pre ktoré atribúty budeme pri zadávaní dát vyžadovať od používateľa zadanie hodnoty (tzv. *požadovaná hodnota*)?
- Aké null hodnoty dovoľíme takýmto atribútom? Ako budeme modelovať domény obsahujúce neúplnú informáciu? Aké formálne modely zvolíme?

Keď už sme zodpovedali na tieto základné otázky, je nutné uvažovať aj o vplyve našich odpovedí na zvyšok systému. Ak pripustíme prítomnosť neúplnej informácie, treba s ňou počítať vo *všetkých* častiach systému. Napr. ak povolíme, aby objem paliva v nádrži bol ako neúplná informácia vyjadrený intervalom (prípadne s najvyšším dovoleným rozsahom), je potrebné uvažovať o tom, že takáto informácia ovplyvní výpočet odhadovanej prejdenej vzdialenosti a navyše, ak sa takýto údaj uchováva, treba aj pre daný atribút povoliť neúplnú informáciu alebo nezabudnúť na logickej úrovni navrhnúť spôsob, akým sa daná vypočítaná neúplná informácia „vtesná“ do hodnoty tohto atribútu.

5.3 Návrh databáz s čiastočnou informáciou

V etape vývoja softvéru, ktorá sa zaoberá návrhom budúceho informačného systému, sa navrhuje aj fyzická štruktúra používaných a uchovávaných dát. Keď už konkrétne uvažujeme (najmä v etape návrhu a implementácie) o rozdelení do tabuliek a relácií¹²⁷, treba si položiť nasledujúce otázky:

- Ako budeme fyzicky reprezentovať čiastočné hodnoty pre tie atribúty, ktoré v konceptuálnom modelovaní pripustili neúplnú informáciu? Ak nedovoľíme nullové hodnoty resp. iné čiastočné hodnoty, hoci neúplnú alebo chýbajúcu informáciu modelovať chceme (čo sme navrhli v etape konceptuálneho modelovania), aké špeciálne hodnoty budeme používať na vyjadrenie neúplnej informácie?
- Akým spôsobom budeme zaobchádzať s neúplnou informáciou, ak ju povolíme? Umožníme používateľovi viacero spôsobov manipulácie s nullmi?
- Ako budeme postupovať pri spájaní (merdžovaní) databáz z rôznych zdrojov (napr. z viacerých pobočiek) v prípade, či už budú obsahovať rôzne/nekonzistentné údaje alebo nullové/čiastočné hodnoty?

Základná otázka, či pripustiť nullové hodnoty, je v tom, či sa takéto rozšírenie a určite aj skomplikovanie oplatí. Treba mať na zreteli, že keď je pravdepodobnosť výskytu neúplnej alebo čiastočnej informácie malá, teda keď je ich výskyt zriedkavým

¹²⁷ za predpokladu, že budeme implementovať dátové štruktúry pomocou relačných databáz, ktorých terminológiou sa v ďalšom vyjadrujeme, hoci takéto uvažovanie je potrebné uskutočniť bez ohľadu na konkrétnu použitú databázovú architektúru

prípadoch¹²⁸, použitie mechanizmov umožňujúcich narábanie s nullovými hodnotami zrejme minie pôvodný cieľ, a to jednoduchšiu a zreteľnú manipuláciu s dátami. Ako poznamenáva už [Grant, 1980], mali by sme zvážiť, v ktorých stĺpcoch je neúplná informácia prevládajúca, a len pre tieto stĺpce uvažovať o otázkach nullových hodnôt. V ďalšom čiastočne načrtneme odpovede na položené otázky. Poznamenajme, že neexistuje žiadna „kuchárka“, ktorá by návrhárovi databázového systému dokázala tieto otázky vyriešiť bez ľudského umu a inteligencie. Fyzický aspekt uchovávaní neúplnej informácie je uvažovaný v ďalšej časti.

5.3.1 Ako reprezentovať čiastočné hodnoty

Na túto otázku sme čiastočne zodpovedali v kapitolách o nullových hodnotách a neúplnej informácii, preto je iba na zvážení, ktorý model najlepšie vyhovuje konkrétnym požiadavkám.

Za predpokladu, že chceme modelovať neúplnú/neznámu informáciu, avšak nepovolíme napr. nullové hodnoty, je potrebné vyhradiť špeciálne hodnoty pre neúplné/neznáme hodnoty. Tu nasleduje niekoľko príkladov:

- *reťazcové hodnoty*: hviezdička môže označovať neznámy reťazec, otáznik neznámy znak,
- *kladné celočíselné alebo reálne hodnoty*: napr. -1 môže mať špeciálny význam, alebo ľubovoľné záporné číslo,
- *reálne hodnoty*: podľa oblasti, napr. ak má reálne číslo vyjadrovať teplotu ľudského tela, hodnoty nižšie ako, povedzme, 20°C (špeciálne 0°C) môžu značiť chýbajúcu informáciu, alebo pokiaľ ide o výsledok merania, je potrebné uchovať okrem nameranej hodnoty aj odchýlku merania (teda napr. $37,4^{\circ}\text{C} \pm 0,3^{\circ}\text{C}$).

5.3.2 Ako zaobchádzať s neúplnou informáciou

Čiastočne na túto otázku je daná odpoveď tým, aký druh neúplnej informácie chceme modelovať. Pre konkrétny model je potrebné rozhodnúť sa, ktoré operácie relačnej algebry podporovať, a to aj na základe ich zložitosti. Taktiež je potrebné rozhodnúť sa, či umožníme používateľovi viacero spôsobov manipulácie s neúplnou informáciou. Napr. pre nully budeme používať outer alebo inner join? Dovolíme používateľovi vybrať si pre každý jednotlivý operátor *true* alebo *maybe* verziu?

Je možné, že sa v etape návrhu systému ukáže ako najvhodnejšie nasledovné riešenie:¹²⁹

- používatelia potrebujú pracovať s neúplnou informáciou, preto na úrovni pohľadov (views) im ponúkneme nástroje na prácu s neúplnou informáciou,
- vnútorná reprezentácia neúplnej informácie v systéme je odlišná od tej, ktorú vidia používatelia – interne je neúplná informácia reprezentovaná úplnou informáciou.

¹²⁸ O tom, koľko je „málo“ a čo znamená „zriedkavo“, je potrebné pri návrhu urobiť diskusiu, prípadne prejednať so zákazníkom. Podľa objemu spracovávaných dát ide o 0.1%, 0.5%, 1%, 2% a pod. Použitie štatistických metód na porovnávanie vplyvu použitia či nepoužitia nullových hodnôt (pri ich predpokladanom výskyte) na výkon databázy ovplyvňujúci rýchlosť vyhľadávania, spájania databáz a pod. nie je predmetom tejto práce.

¹²⁹ môžeme ho nazvať *ilúzia neúplnej informácie*

V takom prípade je potrebné určiť dva spôsoby zaobchádzania s neúplnou informáciou: na úrovni konceptuálneho modelu (ktorému potom zodpovedajú používateľské pohľady) a na úrovni internej reprezentácie.

5.3.3 Ako sa postaviť k heterogénnemu prostrediu

Najmä v distribuovanom prostredí, keď jednotlivé databázy alebo ich časti sú umiestnené na rôznych systémoch a sú spracovávané v rozličných databázových prostrediach, je potrebné už pri návrhu vziať do úvahy heterogénnosť prostredia. Tejto problematike je podrobne venovaná napr. práca [DeMichiel, 1989], kde používa autorka pojem „virtuálne atribúty“ na riešenie nekonzistencie dát. Avšak predpokladáme, že tento problém sa v praxi rieši a bude riešiť najmä nasledovnými spôsobmi:

1. typovanie (šablóny alebo rozhrania), ako ho poznáme napr. zo špecifikácie štandardu CORBA (porov. [Mrázik, 1998]),
2. štandardizácia databázových formátov (v minulosti sa používal najmä jednoduchý a pre potreby mnohých relačných databáz postačujúci formát dbf, v súčasnosti firemné štandardy napr. Oracle, Microsoft Access).

5.4 Neúplná informácia a funkčné závislosti

Dôležitá časť poznatkov o reálnom svete sa v relačnom modeli zachytáva pomocou dátových, špeciálne funkčných závislostí. Preto sa budeme teraz stručne venovať vplyvu funkčných závislostí na niektoré modely pre neúplnú informáciu.

5.4.1 Coddove tabuľky

Nech F je množina funkčných závislostí a T je Coddova tabuľka. $chase_F(T)$ je tabuľka, ktorú dostaneme z T opakovaným aplikovaním nasledovného pravidla: „ak $\mu, v \in T$, $X \rightarrow Y \in F$, a $\mu[X] = v[X]$ sú nenulové, potom identifikuj $\mu[Y]$ s $v[Y]$ “, kde *identifikácia* znamená:

ak $\mu[Y]$ je null a $v[Y]$ je regulárna hodnota, nahradíme $\mu[Y]$ hodnotou $v[Y]$ (a opačne, teda vždy namiesto nullu napíšeme danú konkrétnu hodnotu).

Ako sme spomínali vyššie, pre účely funkčných závislostí de facto predpokladáme, že $\omega = \omega$ (presnejšie, namiesto toho použijeme $\omega \equiv \omega$). Teda dve tuple, zhodné v hodnote atribútu X , ktoré obsahujú pre atribút Y null, neporušujú integritné ohraničenie dané funkčnou závislosťou $X \rightarrow Y$. Môžeme napísať, že platnosť funkčnej závislosti $X \rightarrow Y$ v relácii R je daná nasledovne:¹³⁰

$$\text{platnosť}_1(X \rightarrow Y, R) =_{df} \begin{cases} \text{true}, & \text{ak } \forall \mu, v \in R : \\ & (\mu[X] \neq v[X] \vee \mu[X] = v[X] \Rightarrow \\ & (\mu[X] = v[X] \vee JN(\mu[Y], v[Y])), \\ \text{false} & \text{inak.} \end{cases}$$

kde

$$JN(m, n) := [m \equiv \omega \wedge \neg(n \equiv \omega)] \vee [n \equiv \omega \wedge \neg(m \equiv \omega)],$$

inými slovami, práve jedna z hodnôt m, n je null (odtiaľ skratka JN).

¹³⁰ porov. [Jajodia–Springsteel, 1990]

K nullom v atribúte X môžeme pristupovať dvojako. Prvý prístup je vlastne zahrnutý vo vyššie uvedenej platnosti pre funkčnú závislosť, ktorú nenarušia tuple $\langle X: \omega, Y: 1 \rangle$, $\langle X: \omega, Y: 2 \rangle$. Ak však uvažujeme o doméne $D(Y)$ s konečným množstvom prvkov, môžeme ešte pridať jedno obmedzenie založené na známom *holubníkovom princípe*, a to nasledovné: Hoci by aj každý z nullov v atribúte X bol substituovaný inou hodnotou, počet tuplů, ktoré majú rôzne Y hodnoty, je nanajvýš $|D(Y)|$. Formálne, rozšírená platnosť funkčnej závislosti $X \rightarrow Y$ v relácii R je daná takto:

$$\text{platnosť}_2(X \rightarrow Y, R) =_{df} \text{platnosť}_1(X \rightarrow Y, R) \wedge |\{\mu[Y] \mid \mu[X] \equiv \omega, \mu \in R\}| < |D(Y)|.$$

Tento druhý prístup lepšie zachytáva význam funkčných závislostí, pretože akoby simuluje všetky reprezentácie. To už je náznak všeobecného riešenia, pozri časť 5.4.3.

5.4.2 V–tabuľky

Neúplný model pre „svet“ s funkčnými závislosťami je dvojica $\langle T, F \rangle$, kde T je V–tabuľka a F je množina funkčných závislostí. Ak množinu všetkých relácií, ktoré spĺňajú všetky závislosti z množiny F označíme $Sat(F)$, potom môžeme definovať

$$Rep(\langle T, F \rangle) = Rep(T) \cap Sat(F).$$

Prirodzená je otázka, ako využiť znalosť reprezentovaných závislostí pri zodpovedaní (vyhodnocovaní) dotazov. Ako navrhli Imieliński s Lipskim (porov. [Lipski, 1983a]), poznatky obsiahnuté v F môžu byť vyjadrené vo vhodnej modifikácii pôvodnej tabuľky T , a to pomocou *chase procesu* (porov. [Ullman, 1988]), ktorý môžeme popísať nasledovne:¹³¹

Nech F je množina funkčných závislostí a T je tabuľka. $chase_F(T)$ je tabuľka, ktorú dostaneme z T opakovaným aplikovaním nasledovného pravidla: „ak $\mu, \nu \in T$, $X \rightarrow Y \in F$, a $\mu[X] = \nu[X]$ sú nenullové, potom identifikuj $\mu[Y]$ s $\nu[Y]$ “, kde *identifikácia* znamená, že

- ak $\mu[Y]$ je premenná a $\nu[Y]$ je regulárna hodnota, nahradíme $\mu[Y]$ hodnotou $\nu[Y]$ (a opačne, teda vždy namiesto premennej napíšeme danú konkrétnu hodnotu),
- ak obidve hodnoty sú *odlišné* premenné (napr. $\mu[Y] = x$, $\nu[Y] = y$), premenné stotožníme, presnejšie, jednu nahradíme druhou nejakým pevne vopred daným spôsobom (napr. podľa abecedy vždy nahrádzame premennou, čo je v lexikografickom usporiadaní skôr),
- v ostatných prípadoch nevykonáme žiadnu zmenu.

Platí nasledovné tvrdenie. Pre ľubovoľnú V–tabuľku T a konečnú množinu F funkčných závislostí¹³² platí:

$$Rep(T) \cap Sat(F) \equiv_{PS^+ UJR} Rep(chase_F(T)).$$

¹³¹ ide o rozšírenie chase procedúry pre Coddove tabuľky, ktorá je uvedená vyššie

¹³² v skutočnosti môže obsahovať ľubovoľné zovšeobecnené závislosti (porov. [Ullman, 1988])

Ako vidno z definície, v prípade, že funkčná závislosť je „neúplne“ narušená (teda keď $\mu[X] = v[X]$ je rovnaká premenná a $\mu[Y] \neq v[Y]$ sú rozličné konštanty), *chase* proces túto skutočnosť nezachytí. Inými slovami, pre množinu funkčných závislostí $F = \{A \rightarrow B\}$ a tabuľku

A	B
x	b ₁
x	b ₂

kde $b_1 \neq b_2$, platí $\text{chase}_F(T) = T$, avšak relácia $R = \{\langle a, b_1 \rangle, \langle a, b_2 \rangle\}$, ktorá je v $\text{Rep}(\text{chase}_F(T))$, nie je v $\text{Rep}(T) \cap \text{Sat}(F)$, pretože porušuje závislosť $A \rightarrow B$.

Ekvivalenciu v predošlom tvrdení by bolo možné nahradiť rovnosťou v prípade, že by sme pridali ešte jedno pravidlo pre identifikáciu hodnôt, totiž pre tento uvedený prípad:

- ak $\mu[X] = v[X]$ je rovnaká premenná a $\mu[Y] \neq v[Y]$ sú rozličné konštanty, tabuľka porušuje funkčnú závislosť, a tým aj integritné ohraničenie,
- ak $\mu[X] = v[X]$ je rovnaká premenná a $\mu[Y] \neq v[Y]$ sú rozličné premenné, identifikuj ich, ako keby $\mu[X] = v[X]$ bola konštanta.

5.4.3 Funkčné závislosti vyjadrené reprezentáciou

[Vassiliou, 1980] definuje funkčnú závislosť nasledovne:¹³³

$$\text{platnosť}_3(X \rightarrow Y, R) =_{df} \sqcup \{\text{platnosť}(X \rightarrow Y, R') \mid R' \in \text{Rep}(R)\},$$

kde \sqcup značí najmenšiu hornú hranicu nad množinou s hodnotami $\{true, false\}$ definovanú takto:¹³⁴

$$\sqcup M =_{df} \begin{cases} true, & \text{ak } M = \{true\}, \\ false, & \text{ak } M = \{false\}, \\ unknown, & \text{ak } M = \{true, false\}. \end{cases}$$

Niekedy hovoríme o dvoch druhoch funkčných závislostí: *true*-závislosti (platí vo všetkých reprezentáciách) a *maybe*-závislosti (závislosť platí aspoň v jednej reprezentácii).¹³⁵ Presnejšie, *true*-funkčná závislosť $X \rightarrow Y$ nad reláciou R platí práve vtedy, keď

$$\begin{aligned} & \text{platnosť}_3(X \rightarrow Y, R) = true, \\ \text{maybe-funkčná závislosť platí práve vtedy, keď} & \\ & \text{platnosť}_3(X \rightarrow Y, R) \neq false. \end{aligned}$$

Ak máme testovať viacero funkčných závislostí, musia platiť nielen každá zvlášť, ale aj spoločne. Príklad 28 ukazuje, že je potrebné ich platnosť testovať *simultánne* (nie iba *sekvenčne*).

¹³³ [Vassiliou, 1980], str. 262 zapisuje platnosť funkčnej závislosti trochu odlišným spôsobom; jeho myšlienku sme zapísali uvedeným spôsobom z dôvodu, aby nadväzovala na predchádzajúce úvahy.

¹³⁴ Naozaj ide iba o tieto tri prípady, pretože M je množina hodnôt (nie multimnožina).

¹³⁵ Porov. napr. [Grant, 1980].

Príklad 28: Sekvenčné a simultánne vyhodnocovanie FD

Uvažujme reláciu R a množinu funkčných závislostí $F = \{F_1, F_2\}$, kde $F_1 := A \rightarrow B$, $F_2 := B \rightarrow C$.

$$R$$

A	B	C
a	ω	c_1
a	ω	c_2

Keď vyhodnocujeme funkčné závislosti F_1, F_2 samostatne (sekvenčne), tak obe platia, t.j. platí

$$\text{platnosť}_3(A \rightarrow B, R) = \text{platnosť}_3(B \rightarrow C, R) = \text{true},$$

avšak ľahko vidno, že spolu nemôžu platiť. (Naviac, stačí uvážiť, že tranzitívne implikujú platnosť FD $A \rightarrow C$, ktorá je však zjavne narušená v relácii R .)

Preto platnosť množiny funkčných závislostí F nemôžeme definovať rovnako ako pre klasické relácie (pozri časť 2.1.1.3), ale definujeme ju takto:

$$\text{Platnosť}(F, R) =_{df} \sqcup \{ \text{Platnosť}(F, R') \mid R' \in \text{Rep}(R) \}.$$

Vassiliou ukázal, že pre takto definované funkčné závislosti a ich platnosť sú Armstrongove odvodzovacie pravidlá na reláciách s nullmi zdravé a úplné.

Uvedené časti popisujú teoretickú podobu testovania funkčných závislostí, ktorá má nasledovnú podobu: dostaneme reláciu a množinu funkčných závislostí a máme zodpovedať na otázku, či táto relácia spĺňa integritné ohraničenie definované množinou závislostí. V praktických aplikáciách uplatňujeme metódu *pridávania a kontroly*, teda testujeme integritné ohraničenie (platnosť množiny funkčných závislostí) pri každom pridaní a modifikácii tuple. Ak tuple porušuje FD, nedovolíme ju pridať resp. zmeniť. Podobne sa správame aj k pridaniu celej tabuľky do tabuľky, ktoré realizujeme ako iterované pridávanie tuple.

5.4.4 Neúplná informácia a normálne formy

Ak vieme otestovať, či daná funkčná závislosť platí, potom vieme otestovať, či sa relácia nachádza v 2NF. Stačí zistiť, či všetky neklúčové atribúty závisia na celom kľúči relácie. Overiť, či sa daná tabuľka nachádza v 3NF, vyžaduje okrem kontroly 2NF aj testovanie, či v tabuľke nie sú tranzitívne závislosti. Otestovanie, či tabuľka je v BCNF (je potrebné zistiť, či každý atribút, na ktorom sú nejaké iné atribúty závislé,¹³⁶ je súčasťou nejakého nadkľúča), nevyžaduje špeciálne zaobchádzanie s neúplnou informáciou. Ponechávame ďalšie otázky o normálnych formách nezodpovedané s tým, že zdôrazníme nasledujúce doporučené. Relácie treba navrhovať tak, aby pre každú reláciu existoval jediný nadkľúč, a to primárny kľúč, naviac, každá tabuľka by mala „nieť ľarchu reprezentovať“ iba jedinou funkčnú závislosť. Týmto budeme tvoriť tabuľky automaticky v 3NF resp. BCNF.

¹³⁶ nazýva sa aj *determinant*

5.5 Dekompozícia tabuliek odstraňujúca nully

Ako sme spomenuli vyššie (str. 23), je možné tabuľky obsahujúce základné interpretácie nullov (unknown, inapplicable, no information) dekomponovať na ekvivalentné tabuľky len s jedným typom nullu unknown typu, ktorý je v podstate trikom odstrániteľný.

V nasledujúcom odseku navrhujeme spôsob, ktorým túto dekompozíciu realizovať, pričom taktiež popíšeme výhody a nevýhody nášho prístupu. Tento odsek ukazuje, že bez nullov (teda aspoň uvažovaných druhov) sa vieme zaobiť, ak v etape návrhu navrhujeme schému tabuľky priamo takú istú, ako popisujeme pri dekompozícii.

5.5.1 Jeden atribút s nullmi

Najprv precízne popíšeme vyššie uvedený príklad dekompozície, keď uvažovaná tabuľka obsahuje práve jeden stĺpec pripúšťajúci všetky tri základné druhy nullov (**un**, **ne**, **ni**).

Nech W je množina atribútov relácie R , ktorá obsahuje jediný atribút A , ktorý má hodnoty z domény D_A obsahujúcej regulárne hodnoty, alebo niektorý z uvedených nullov, teda

$$D(A) = D_A \cup \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\}.$$

Relácii R bude zodpovedať dvojica relácií R', R_A takých, že pôvodná relácia sa dá ľahko rekonštruovať (neskôr popíšeme, čo pod *ľahkým rekonštruovaním* myslíme). Usporiadaná dvojica relácií $\langle R', R_A \rangle$ sa nazýva *dekompozícia tabuľky R* a označuje $\Delta(R)$.¹³⁷ Relácie R', R_A sú vytvorené nasledovným spôsobom. Pre každú tupľu $\mu \in R$ zostrojíme *modifikovanú tupľu* μ' (a hovoríme, že tupľu μ *zodpovedá* (modifikovaná) tupľu μ') nasledovne:

$$\mu' =_{df} \begin{cases} \mu[A/\text{ÁNO}] \in R', & \text{ak } \pi_A(\mu) = \mathbf{un}, \\ \mu[A/\text{NIE}] \in R', & \text{ak } \pi_A(\mu) = \mathbf{ne}, \\ \mu[A/\text{NEVIEM}] \in R', & \text{ak } \pi_A(\mu) = \mathbf{ni}, \\ \mu \in R_A, & \text{ak } \pi_A(\mu) \notin \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\}, \end{cases}$$

pričom

- $\mu[A/x]$ znamená substitúciu aktuálnej hodnoty atribútu A tuple μ hodnotou $x \in \{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$,
- tupľu zaradíme v prvých troch prípadoch do R' , v poslednom do R_A , čo je naznačené vyššie zápisom „patrí do množiny“.

Jednotlivé hodnoty ÁNO, NIE, NEVIEM nazývame *indikátory*.

Dekompozícia sa dá zapísať aj množinovo, a to nasledovne:

$$\begin{aligned} \Delta(R) := & \langle \{ \mu[A/\text{ÁNO}] \mid \mu \in R \wedge \pi_A(\mu) = \mathbf{un} \} \\ & \cup \{ \mu[A/\text{NIE}] \mid \mu \in R \wedge \pi_A(\mu) = \mathbf{ne} \} \\ & \cup \{ \mu[A/\text{NEVIEM}] \mid \mu \in R \wedge \pi_A(\mu) = \mathbf{ni} \}, \\ & \{ \mu \in R \mid \pi_A(\mu) \notin \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\} \} \rangle. \end{aligned}$$

Relácia R' má relačnú schému $W \setminus A, A'$, kde $D(A') = \{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$. Atribút A' , ktorého hodnotou je indikátor, vlastne obsahuje túto informáciu: „Existuje v relácii R hodnota atribútu A ?“¹³⁸ Pritom sa v tejto relácii nevyskytujú tuple s úplnou informáciou.

¹³⁷ Symbol Δ (delta) pre $\Delta(R)$ sme zvolili, aby naznačoval, že ide o dekompozíciu tabuľky R , ako aj na odlíšenie od $D(R)$, čo značí doménu zodpovedajúcu atribútu R .

¹³⁸ Inými slovami, indikuje, či hodnota atribútu existuje. Preto *indikátor*.

Relácia R_A má schému W . Obsahuje iba tuple, ktoré v pôvodnej relácii mali v atribúte A regulárnu hodnotu. Takže priamo z definície platí

$$R' \cap R_A = \emptyset,$$

$$R' \cup R_A = R.$$

Keď si uvedomíme, že relačné schémy relácií R' a R_A sú rôzne, potom môžeme vysloviť, že pre relácie R, S nad tou istou relačnou schémou platí

$$R' \cap S_A = \emptyset,$$

$$S' \cap R_A = \emptyset.$$

Priamo z definície tiež plynie, že ak R je relácia bez nullov, potom $\Delta(R) = \langle R, \emptyset \rangle$.

Zostrojenie relácií R' a R_A z relácie R možno chápať ako použitie dvoch operátorov (čiarka, index A) či operátora Δ na pôvodnú reláciu R . Operátor $'$ znamená vlastne istý druh selekcie nullových tuplí so zmenou hodnoty nullového atribútu, operátor $_A$ znamená selekciu nenullových tuplí. Operátor Δ je zložený z dvoch zložiek, operátorov čiarky $'$ a indexu A . K tomuto matematickému pohľadu existuje aj informatický prot'ajšok. Vyššie popísaný postup tvorby relácií R', R_A totiž neformálne určuje algoritmus konštrukcie $\Delta(R)$, teda algoritmus, ktorým sa dekomponuje relácia R na dvojicu relácií R', R_A . Je vidieť, že časovú zložitosť vzhľadom k počtu tuplí má lineárnu, inými slovami, ak počet tuplí relácie R je n , algoritmus trvá rádovo $O(n)$. Veľkosť nových tabuliek je (až na iné záhlavia) rovnaká ako veľkosť tabuľky zodpovedajúcej pôvodnej relácii, pretože každá tupľa relácie R sa nachádza práve v jednej z relácií R', R_A . Ak uvažujeme, že pôvodná relácia je reprezentovaná záznamami pevnej dĺžky, možno predpokladať, že veľkosť nových tabuliek bude menšia ako veľkosť pôvodnej tabuľky, pretože na reprezentáciu neúplnej informácie sa v pôvodnej tabuľke spotreboval rovnaký priestor ako na reprezentáciu regulárnych hodnôt, avšak v novej tabuľke zodpovedajúcej relácii R' stačí na reprezentáciu hodnoty atribútu A' priestor dvoch bitov (kódovanie indikátorov).

Keďže relácia R_A obsahuje iba tuple relácie R neobsahujúce nully, sama neobsahuje nullové hodnoty. Podobne relácia R' obsahuje síce tuple relácie R , ktoré pôvodne obsahovali nully, avšak algoritmom dekompozície boli tieto nully nahradené indikátormi, regulárnymi hodnotami z množiny $\{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$. Výsledné relácie teda neobsahujú nullové hodnoty. Vyššie spomínaný trik spočíva v zavedení indikátorov.

Vysvetlíme, čo znamená *lahko rekonštruovať* pôvodnú tabuľku. Nech R' a R_A sú tabuľky, definované takto: R' aj R_A majú (až na atribút A) rovnakú schému. Presnejšie, R' má relačnú schému $W \setminus A, A'$, kde $D(A') = \{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$. Relácia R_A má schému W . Naviac, žiaden z atribútov množiny $W \setminus A$ nesmie nadobúdať nullové hodnoty. Jediný atribút, ktorý môže obsahovať nullové hodnoty typu **un**, **ne** resp. **ni** je A . Ak tabuľky R' a R_A spĺňajú tieto vlastnosti, nazývajú sa *rekonštruovateľné*. Nech relácia R'' obsahuje všetky tuple relácie R_A a naviac pre každú tupľu μ relácie R' obsahuje tupľu μ' (tupľu μ upravenú nasledovne):

$$\mu'[A] =_{df} \begin{cases} \mu[A/\text{un}] \in R'', & \text{ak } \pi_A(\mu) = \text{ÁNO}, \\ \mu[A/\text{ne}] \in R'', & \text{ak } \pi_A(\mu) = \text{NIE}, \\ \mu[A/\text{ni}] \in R'', & \text{ak } \pi_A(\mu) = \text{NEVIEM}, \end{cases}$$

naviac hovoríme, že (modifikovanej) tupli μ *zodpovedá* tupľa μ' .

Píšeme $R'' = \nabla(R', R_A)$ a hovoríme, že R'' je *rekonštrukciou* tabuliek R' , R_A . Je zrejmé, že $R = R''$. Vyplýva to z definovania spätnej úpravy tuplí relácie R' , ktorá bola definovaná ako inverzná operácia ku konštrukcii tabuľky R' .¹³⁹ Navyše platia nasledujúce rovnosti:

$$\nabla(\Delta(R)) = R,$$

$$\Delta(\nabla(R)) = R$$

pre ľubovoľnú reláciu R obsahujúcu tri základné druhy nullov (**un**, **ne**, **ni**). Platí teda

$$\nabla = \Delta^{-1},$$

čiže operácia rekonštrukcie je inverzná operácia k operácii dekompozície (a opačne).

Rekonštrukcia sa dá zapísať aj množinovo, a to nasledovne:

$$\begin{aligned} R'' := & \{ \mu[A/\mathbf{un}] \mid \mu \in R' \wedge \pi_A(\mu) = \text{ÁNO} \} \\ & \cup \{ \mu[A/\mathbf{ne}] \mid \mu \in R' \wedge \pi_A(\mu) = \text{NIE} \} \\ & \cup \{ \mu[A/\mathbf{ni}] \mid \mu \in R' \wedge \pi_A(\mu) = \text{NEVIEM} \} \cup R_A. \end{aligned}$$

Uvedená rekonštrukcia trvá vzhľadom k počtu prvkov pôvodnej relácie rádovo $O(n)$, pretože pre každú tupľu sa buď vykoná len prečítanie a zápis (tuple z R_A) alebo navyše porovnanie a modifikácia jedného atribútu (tuple z R'). Z tohoto dôvodu sme rekonštrukciu považovali za *ľahkú*.

Teraz popíšeme, ako vyzerá manipulácia s novými dekomponovanými tabuľkami. *Príslušnosť tuple* k relácii $\Delta(R)$ – teda vyhľadanie tuple alebo aj jej odstránenie – sa zrealizuje najjednoduchšie paralelným prehľadávaním tabuliek R' a R_A (a prípadným nasledovným odstránením nájdenej tuple). Ak uvažujeme sekvenčnú architektúru, jednoduchá simulácia paralelizmu spôsobom jeden krok v tabuľke R' , jeden krok v tabuľke R_A (prípadne proporčne vzhľadom k veľkostiam tabuliek R' a R_A), stále bude časová náročnosť nanajvýš konštantne ráz väčšia¹⁴⁰ ako v prípade sekvenčného prehľadávania tabuľky R . Rozdiely sú ešte menšie, keď sú tuple relácie R nejakým spôsobom usporiadané, pretože algoritmus dekompozície toto usporiadanie zachováva a prehľadávanie slovníkovým spôsobom ho využíva v oboch komponentoch.

Pridanie novej tuple do $\Delta(R)$ bude predchádzané najprv testom hodnoty atribútu A , či (ne)obsahuje nullové hodnoty. Podľa toho sa priamo pridá do relácie R_A (keď neobsahuje null), alebo sa hodnota atribútu A modifikuje vyššie popísaným spôsobom a takto upravená tupľa sa pridá do relácie R' .

Rovnosť dvoch dekomponovaných tabuliek definujeme nasledovne:

$$\Delta(R_1) = \Delta(R_2) \Leftrightarrow_{df} R_1' = R_2' \wedge R_{A1} = R_{A2},$$

pričom R_i' píšeme namiesto $(R_i)'$ a R_{iA} píšeme namiesto $(R_i)_A$. Táto rovnosť sa prakticky otestuje rovnako, ako v prípade klasických tabuliek, pričom opäť môžeme uvažovať o paralelnom testovaní. Jeho výhoda je v tom, že nerovnosť tabuliek môže zistiť skôr (rýchlejšie), ako keby sme testovali $R_1 = R_2$, pretože obidve porovnávané dvojice tabuliek

¹³⁹ Treba však poznamenať, že táto rovnosť platí množinovo (R a R'' sú navzájom ekvivalentné ako množiny, vzhľadom k prvkom), keď relácie sú definované ako *množiny* tuplí.

¹⁴⁰ Ak sa v tabuľke R nachádza $2n + 1$ tuplí, pričom prvá polovica ($n + 1$) tuplí obsahuje v atribúte A regulárnu hodnotu a zvyšných n tuplí obsahuje v tomto atribúte null, pri sekvenčnom vyhľadávaní ($n + 1$)-vej tuple sa vykoná $2n + 1$ krokov, v pôvodnej tabuľke $n + 1$ krokov, čo je maximálne dvakrát viac. Pritom predpokladáme, že sa sekvenčne prehľadáva spôsobom krok v R' , krok v R_A .

sú menšie ako pôvodné tabuľky. Testovanie rovnosti trvá v najhoršom prípade rovnaký čas (rovnaké množstvo porovnaní, ak tabuľky R_1 resp. R_2 obsahujú len nenullové alebo len nullové tuple) ako v prípade $R_1 = R_2$.

Zjednotenie dekomponovaných relácií definujeme nasledovne:

$$\Delta(R_1) \cup \Delta(R_2) =_{df} \langle R_1' \cup R_2', R_{1A} \cup R_{2A} \rangle,$$

kde

$$\Delta(R_i) = \langle R_i', R_{iA} \rangle, i \in \{1, 2\}.$$

Treba pritom overiť, či platí

$$\Delta(R_1) \cup \Delta(R_2) = \Delta(R_1 \cup R_2),$$

inými slovami, či nami zadefinovaná dekompozícia zachováva zjednotenie relácií.

Poznamenajme, že $\Delta(R_1 \cup R_2)$ vyzerá z definície nasledovne:

$$\Delta(R_1 \cup R_2) = \langle (R_1 \cup R_2)', (R_1 \cup R_2)_A \rangle.$$

Na overenie potrebujeme ukázať, že

$$\langle R_1' \cup R_2', R_{1A} \cup R_{2A} \rangle = \langle (R_1 \cup R_2)', (R_1 \cup R_2)_A \rangle.$$

Vezmime tuple $\mu \in R_1' \cup R_2'$. Nech bez ujmy na všeobecnosti tuple μ sa nachádza v R_1' , čo znamená, že tuple $\mu_1 \in R_1$ (ktorá zodpovedá tuple μ) obsahuje null. Rovnako však $\mu_1 \in R_1 \cup R_2$, takže μ (tuple zodpovedajúca tuple μ_1) sa nachádza v tabuľke $(R_1 \cup R_2)'$. Tuple zodpovedajúca tuple μ_1 je naozaj rovnaká ako μ , preto sme ju aj označili μ . Vyplyva to z definície modifikovanej tuple. Tým sme ukázali, že platí

$$R_1' \cup R_2' \subseteq (R_1 \cup R_2)'.$$

To isté platí aj o tuple $\mu \in R_{1A} \cup R_{2A}$ (tuple μ v R_1 resp. R_2 neobsahuje null). Teda máme

$$R_{1A} \cup R_{2A} \subseteq (R_1 \cup R_2)_A.$$

Opačne, vezmime tuple $\mu \in (R_1 \cup R_2)'$. Tejto tuple zodpovedá v relácii $(R_1 \cup R_2)$ tuple μ_1 obsahujúca null. Z definície zjednotenia máme $\mu_1 \in R_1 \vee \mu_1 \in R_2$. Nech (opäť bez ujmy na všeobecnosti) $\mu_1 \in R_1$. μ_1 však obsahuje null, takže $\mu_1 \in R_1' \subseteq R_1' \cup R_2'$. Tým sme ukázali, že platí

$$R_1' \cup R_2' \supseteq (R_1 \cup R_2)'.$$

Rovnako sa ukáže, že $R_{1A} \cup R_{2A} \supseteq (R_1 \cup R_2)_A$, čím je dôkaz ukončený.

Zjednotenie relácií možno chápať aj ako iterované pridávanie nových tuple. Pokiaľ sa jedná o dve tabuľky rovnakého typu (dve nedekomponované alebo dve dekomponované tabuľky), nezáleží, či pridávame tuple prvej tabuľky do druhej alebo naopak. Na poradi zjednotencov však záleží, ak zjednocujeme tabuľku a dekomponovanú tabuľku.

Definujeme preto

$$R_1 \cup \Delta(R_2) =_{df} \Delta(R_1) \cup \Delta(R_2),$$

$$\Delta(R_1) \cup R_2 =_{df} \Delta(R_1) \cup \Delta(R_2).$$

Pre úplnosť ešte dodajme, že platí

$$R_1 \cup R_2 = \nabla(\Delta(R_1) \cup \Delta(R_2)).$$

Vezmime tuple $\mu \in R_1 \cup R_2$. Nech bez ujmy na všeobecnosti sa tuple μ nachádza v R_1 . Tuple μ buď obsahuje null alebo nie. Uvažujme najskôr prípad, keď null obsahuje, čo znamená, že tuple μ' (ktorá zodpovedá tuple μ) patrí do R_1' . Rovnako však $\mu' \in R_1' \cup R_2'$, takže k nej zodpovedajúca tuple μ (označili sme ju rovnako, pretože je rovnaká ako μ) sa nachádza v tabuľke $\nabla(\langle R_1' \cup R_2', S_A \rangle)$ pre ľubovoľné S_A , teda aj v tabuľke

$$\nabla(\langle R_1' \cup R_2', R_{1A} \cup R_{2A} \rangle) = \nabla(\Delta(R_1) \cup \Delta(R_2)).$$

Uvažujme teraz prípad, keď tupľa μ neobsahuje null. Znamená to, že tupľa $\mu' = \mu$ (rovnosť tu vyplýva z definície: ak tupľa neobsahuje null, nie je modifikovaná) patrí do R_{1A} . Rovnako potom $\mu \in R_{1A} \cup R_{2A}$, takže μ sa nachádza v tabuľke $\nabla(\langle S', R_{1A} \cup R_{2A} \rangle)$ pre ľubovoľné S' , teda aj v tabuľke

$$\nabla(\langle R_1' \cup R_2', R_{1A} \cup R_{2A} \rangle) = \nabla(\Delta(R_1) \cup \Delta(R_2)).$$

Tým sme ukázali, že platí

$$R_1 \cup R_2 \subseteq \nabla(\Delta(R_1) \cup \Delta(R_2)).$$

Obrátene, nech $\mu \in \nabla(\Delta(R_1) \cup \Delta(R_2))$. Najskôr, nech μ obsahuje null, čiže jemu zodpovedajúca tupľa μ' je súčasťou prvého komponentu výrazu $\Delta(R_1) \cup \Delta(R_2)$, teda $\mu' \in R_1' \cup R_2'$. Nech (bez ujmy na všeobecnosti) $\mu' \in R_1'$. Znamená to, že tupľa zodpovedajúca tupli μ' patrí do R_1 . Ako sme už viac ráz spomínali vyššie, upravenie (modifikácia) tuple je inverzná operácia, teda môžeme spokojne napísať $\mu \in R_1$, a teda aj $\mu \in R_1 \cup R_2$. Napokon nech μ neobsahuje null. Potom $\mu \in R_{1A} \cup R_{2A}$; nech bez ujmy na všeobecnosti $\mu \in R_{1A}$. Takže $\mu \in R_1$, a teda aj $\mu \in R_1 \cup R_2$. Ukázali sme opačnú inklúziu,

$$R_1 \cup R_2 \supseteq \nabla(\Delta(R_1) \cup \Delta(R_2)),$$

čím je dôkaz skončený.

Prienik dekomponovaných relácií definujeme podobne ako zjednotenie, a to nasledovne:

$$\Delta(R_1) \cap \Delta(R_2) =_{df} \langle R_1' \cap R_2', R_{1A} \cap R_{2A} \rangle,$$

kde

$$\Delta(R_i) = \langle R_i', R_{iA} \rangle, i \in \{1, 2\}.$$

Treba pritom overiť, či platí

$$\Delta(R_1) \cap \Delta(R_2) = \Delta(R_1 \cap R_2),$$

inými slovami, či nami zadefinovaná dekompozícia zachováva prienik relácií. Dôkaz vykonáme len sčasti. Vezmime tupľu $\mu \in R_1' \cap R_2'$, teda tupľa μ sa nachádza v R_1' aj v R_2' , čo znamená, že tupľa $\mu_1 \in R_1$ (ktorá zodpovedá tupli μ v R_1) obsahuje null a aj tupľa $\mu_2 \in R_2$ (ktorá zodpovedá tupli μ v R_2) obsahuje null. Navyiac, tieto dve tuple sú rovnaké, ako vyplýva z definície modifikovanej tuple. Teda $\mu_1 \in R_1 \cap R_2$, takže μ (tupľa zodpovedajúca tupli $\mu_1 = \mu_2$) sa nachádza v tabuľke $(R_1 \cap R_2)'$. Tým sme ukázali, že platí

$$R_1' \cap R_2' \subseteq (R_1 \cap R_2)'.$$

Opačne, vezmime tupľu $\mu \in (R_1 \cap R_2)'$. Tejto tupli zodpovedá v relácii $(R_1 \cap R_2)$ tupľa μ_1 obsahujúca null. Z definície prieniku máme $\mu_1 \in R_1 \wedge \mu_1 \in R_2$. Keďže μ_1 obsahuje null, platí $\mu_1 \in R_1' \wedge \mu_1 \in R_2'$, a teda aj $\mu_1 \in R_1' \cap R_2'$. Tým sme ukázali, že platí

$$R_1' \cap R_2' \supseteq (R_1 \cap R_2)',$$

teda že platí

$$R_1' \cap R_2' = (R_1 \cap R_2)'.$$

Rovnako sa ukáže, že $R_{1A} \cap R_{2A} = (R_1 \cap R_2)_A$, čím je dôkaz zachovania prieniku operátorom Δ ukončený.

Rovnako ako v prípade zjednotenia, definujeme prienik aj pre tabuľku a dekomponovanú tabuľku takto:

$$R_1 \cap \Delta(R_2) =_{df} \Delta(R_1) \cap \Delta(R_2),$$

$$\Delta(R_1) \cap R_2 =_{df} \Delta(R_1) \cap \Delta(R_2).$$

Aj pre prienik platí $R_1 \cap R_2 = \nabla(\Delta(R_1) \cap \Delta(R_2))$. Dôkaz sa vykoná podobne ako pre zjednotenie.

Selekcia tuplí dekomponovanej tabuľky bude rovnaká (teda nezmenená) v prípade, že výraz F , pomocou ktorého sa selektuje, neobsahuje nullovú hodnotu. Vtedy definujeme

$$\sigma_F(\Delta(R)) =_{df} \langle \sigma_F(R'), \sigma_F(R_A) \rangle.$$

Pre výraz F obsahujúci nullové hodnoty definujeme jemu zodpovedajúci výraz F' taký, že

$$F' = F[\mathbf{un}/\text{ÁNO}][\mathbf{ne}/\text{NIE}][\mathbf{ni}/\text{NEVIEM}],$$

pričom $F[z/x]$ znamená substitúciu nullovej hodnoty $z \in \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\}$ výrazu F hodnotou $x \in \{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$. Inverzná operácia, ktorou z výrazu F' dostaneme výraz F'' , pričom ľahko vidno, že $F = F''$, je definovaná nasledovne:

$$F'' = F'[\text{ÁNO}/\mathbf{un}][\text{NIE}/\mathbf{ne}][\text{NEVIEM}/\mathbf{ni}].$$

Ešte je potrebné zdefinovať, akým spôsobom budeme vyhodnocovať výrazy obsahujúce nully. Hoci sme pri príslušnosti tuple do relácie porovnávali tupľu s tupľou, doteraz to nebolo potrebné, pretože išlo o porovnávanie tuple ako celku. Vtedy napr.

$(\mathbf{un} = \mathbf{un}) = \text{true}$, $(\mathbf{un} = \mathbf{ni}) = \text{false}$, $(\mathbf{un} = 23) = \text{false}$. V ďalšom budeme takéto porovnávanie označovať symbolom \equiv , ktorý označuje „striktné“ resp. syntaktické porovnanie. Budeme používať trojhodnotovú logiku, vedomí si jej nedostatkov a paradoxov, ako boli spomenuté v časti o Coddovom prístupe (str. 23 a ďalej). Tabuľka 2 uvádza spôsob vyhodnocovania výrazov obsahujúcich tri pravdivostné hodnoty.

Tabuľka 7: Narábanie s nullovými hodnotami (vyjadrené tabuľkou resp. vzťahom)

$g = h$	a	\mathbf{ni}	\mathbf{un}	\mathbf{ne}
a	T	M	M	F
\mathbf{ni}	M	M	M	F
\mathbf{un}	M	M	M	F
\mathbf{ne}	F	F	F	F

$$g = h =_{df} \begin{cases} \text{true,} & \text{ak } g, h \text{ sú nenullové a } g \equiv h, \\ \text{false,} & \text{ak } g, h \text{ sú nenullové a } \neg(g \equiv h) \\ \text{maybe} & \text{alebo } g \equiv \mathbf{ne} \vee h \equiv \mathbf{ne}, \\ & \text{inak.} \end{cases}$$

Tabuľka 7 uvádza výsledok aplikácie porovnania na rôzne druhy hodnôt atribútov, a je konštanta (nenullová hodnota). Pritom síce platí, že $\mathbf{ne} \equiv \mathbf{ne}$, avšak $\neg(\mathbf{ne} = x)$ pre ľubovoľné x , čo vyplýva zo sémantiky „inapplicable“ nullu. Pri porovnaní sa totiž pýtame na hodnotu, a \mathbf{ne} null znamená, že tam ani žiadna nie je (teda sa ničomu „ani náhodou“ nemôže rovnať), teda odpoveď nemôže byť ani M (*maybe*).

Tabuľka 8: Rovnosť hodnôt pomocnej domény

$=$	a	NEVIEM	ÁNO	NIE
a	T	M	M	F
NEVIEM	M	M	M	F
ÁNO	M	M	M	F
NIE	F	F	F	F

Tabuľka 8 uvádza, ako budeme pristupovať k porovnaniu hodnôt dekomponovaných tabuliek. Vidno, že obe tabuľky sú v podstate totožné, teda zmena hodnôt v dekomponovaných tabuľkách (vlastne len v prvých komponentoch) zachová sémantiku nullových hodnôt nezmenenú.

Pre ostatné operácie definujeme ich význam na regulárnych hodnotách a pre nullové hodnoty určíme rovnaké vyhodnocovanie, ako vyhodnocovanie $=$.

Teraz sme pripravení definovať selekciu nad dekomponovanými reláciami:

$$\sigma_F(\Delta(R)) =_{df} \langle \sigma_G(R'), \sigma_G(R_A) \rangle,$$

pričom pre názornosť sme namiesto F' uviedli G . Vidieť, že prípad, keď F neobsahuje nully, je špeciálnym prípadom tejto definície. Uviedli sme ho však preto, lebo vo väčšine prípadov sa dotazujeme len na regulárne hodnoty. Používateľ (zo svojho pohľadu) ani nemusí vedieť, že (a či vôbec) informácia je vnútorne reprezentovaná pomocou nullových hodnôt. Dotazy na tuple obsahujúce nullové hodnoty sú skôr záležitosťou administrátora databázy alebo vysoko pokročilých používateľov.

Požadovaná vlastnosť takto definovanej selekcie je:

$$\Delta(\sigma_F R) = \sigma_F(\Delta(R)).$$

K dôkazu je potrebné ukázať, že pre ľubovoľnú tuple μ s tromi uvažovanými nullmi platí

$$F(\mu) \Leftrightarrow F'(\mu').$$

Stačí uvážiť, že elementárne zložky týchto dvoch výrokov sa vyhodnotia rovnako (vzhľadom na rovnakú definíciu porovnania a požiadavku, aby pre nullové hodnoty boli operácie dedefinované rovnako ako pre $=$, teda porovnanie) a logické výrazy zostrojené pomocou logických spojok sa vyhodnocujú rovnako, trojhodnotovou logikou.

Projekcia dekomponovaných tabuliek nespôsobuje ťažkosti. Definujeme

$$\pi_W(\Delta(R)) =_{df} \langle \pi_W(R'), \pi_W(R_A) \rangle,$$

kde W je podmnožina množiny atribútov relácie R . Ľahko sa dá ukázať, že

$$\pi_W(\Delta(R)) = \Delta(\pi_W(R)).$$

Aby sme mohli uvažovať o operáciách typu join, treba najprv pripomenúť dôležité obmedzenie, ktoré istým spôsobom znižuje pôsobnosť nami definovaných operácií tohoto typu, a to fakt, že sa (zatiaľ) zaoberáme len reláciami, obsahujúcimi nullové hodnoty v najviac jednom atribúte. Teda join pripustíme len s reláciou, ktorá neobsahuje nully a kartézsky súčin dovoľíme len s klasickou reláciou (inak by nám vznikli dva stĺpce s nullmi). Formálne definovať ho však nebudeme, pretože definícia je analogická ako predchádzajúce. Budeme sa v nasledujúcom odseku venovať dekompozíciám tabuliek, pre ktoré dovoľíme nullové hodnoty vo viacerých atribútoch.

5.5.2 Viacero atribútov s nullmi

Vo všeobecnosti môže relácia obsahovať nully vo viacerých atribútoch. Nech relácia R obsahuje nullové hodnoty práve v atribútoch A_1, \dots, A_m ($m \geq 2$). Nech W je množina atribútov relácie R a nech A je množina všetkých atribútov s nullmi, teda

$$A =_{mt} A_1 \cup \dots \cup A_m.$$

Hodnoty atribútov A_i sú regulárne hodnoty domény D_i alebo niektorý z uvedených nullov, teda

$$D(A_i) = D_i \cup \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\}, 1 \leq i \leq m.$$

Atribúty z množiny $W \setminus A$, ktoré sú primárnym kľúčom, označíme K . Pre zjednodušenie predpokladáme, že relácia R má primárny kľúč zložený z jediného atribútu, teda $|K| = 1$. Toto zjednodušenie nie je obmedzujúce, najviac spôsobí menej priestorovo efektívnu reprezentáciu dekomponovaných relácií, ktorá sa dá odstrániť definovaním jednoatribútového primárneho kľúča alebo jednoznačnou identifikáciou tuple.

Jednoznačná identifikácia tuple znamená, že každá tuple pôvodnej relácie R by mala jednoznačný identifikátor id , ktorého hodnoty budú napr. (automaticky generované) čísla. Zároveň by sme pre jednoznačnú identifikáciu tuple pridali funkčnú závislosť $id \rightarrow K$, teda identifikátor by nahrádzal a naviac jednoznačne určoval primárny kľúč. Avšak identifikátor by bol súčasťou internej reprezentácie dát, neprístupnej pre používateľa. Používateľ teda nebude mať vôbec poňatia o tom, že jeho dáta sú vo viacerých tabuľkách,

naviac poprepájaných cez kľúč *id*. Inherentne teda aj zakážeme pýtať sa na tuple pomocou *id*, množina atribútov selekcie bude vždy interne obohatená o atribút *id*. Pritom stojí za zmienku, že dotazy, pýtajúce sa na množinu atribútov bez kľúča, presnejšie projekcie na množinu atribútov, z ktorej vylúčime kľúč, *netvorí zmysluplné tabuľky*, pretože nemajú určený primárny kľúč. Napríklad dotaz s výslednou tabuľkou s dvojicami plat–telefónne_číslo: táto tabuľka nie je príliš zmysluplná sama osebe, naviac ak sa v nej vyskytujú tuple (dvojice)

$\langle 20.000, 313 \rangle,$
 $\langle 17.500, 313 \rangle,$
 $\langle 20.000, 312 \rangle,$

ktoré znamenajú, že dvaja kolegovia sediaci v tej istej kancelárii s klapkou 313 majú rôzny plat, avšak jeden z nich má rovnaký plat ako kolega zo susednej kancelárie, nemôže obsahovať kľúč. Ak nejaké dva atribúty A, B spolu súvisia, existuje medzi nimi funkčná závislosť, či už priama ($A \rightarrow B$ resp. $B \rightarrow A$), tranzitívna (napr. $A \rightarrow C \rightarrow B$), alebo závisia od toho istého atribútu (napr. $C \rightarrow AB$).¹⁴¹ Keď ich odprojektujeme na časť atribútov, ktoré od seba závisia, takáto tabuľka sama osebe nemá žiaden význam (obsahuje len torzo dát, ktoré kedysi spolu súviseli), jej význam je iba v tom, že môže ísť o odpoveď na dotaz o hodnotách atribútov.

Poznamenajme, že v ďalšom budeme namiesto $\pi_{A_i}(\mu)$ písať $\pi_{A_i}(\mu)$. Relácii R bude zodpovedať $(m + 1)$ -tica relácií R', R_1, \dots, R_m takých, že pôvodná relácia sa dá ľahko rekonštruovať (neskôr popíšeme, čo pod *ľahkým rekonštruovaním* myslíme). Usporiadaná $(m + 1)$ -tica relácií $\langle R', R_1, \dots, R_m \rangle$ sa nazýva *dekompozícia tabuľky* R a označuje $\Delta(R)$. Relácia R' je tvorená tupľami relácie R , ktoré majú namiesto atribútov A_1, \dots, A_m nové atribúty A'_1, \dots, A'_m nadobúdajúce hodnoty z množiny {ÁNO, NIE, NEVIEM}, a to aj v prípade, že pôvodná tupľa neobsahovala v žiadnom atribúte null. To je významný rozdiel oproti prípadu jedného null pripúšťajúceho atribútu. Relácie R_1, \dots, R_m sú vytvorené nasledovným spôsobom. Pre každú tupľu $\mu \in R$ skonštruujeme *modifikovanú tupľu* $\mu' \in R'$ nasledovne:

- pre všetky i od 1 do m ,

$$\mu'[A_i] =_{df} \begin{cases} \mu[A_i/\text{ÁNO}] \in R'', & \text{ak } \pi_{A_i}(\mu) = \mathbf{un}, \\ \mu[A_i/\text{NIE}] \in R'', & \text{ak } \pi_{A_i}(\mu) = \mathbf{ne}, \\ \mu[A_i/\text{NEVIEM}] \in R'', & \text{ak } \pi_{A_i}(\mu) = \mathbf{ni}, \\ \text{ÁNO}, & \text{ak } \pi_{A_i}(\mu) \notin \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\}, \end{cases}$$

naviac v poslednom prípade ($\pi_{A_i}(\mu) \cap \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\} = \emptyset$) pridáme tupľu

$v_i =_{nt} \pi_K(\mu) \bowtie \pi_{A_i}(\mu)$ do relácie R_i ,

- pre všetky ostatné atribúty $B \notin A$ (vrátane kľúča) položíme $\mu'[B] = \mu[B]$.

Teda tupli $\mu \in R$ zodpovedá najviac $(m + 1)$ -tica¹⁴² tuplí μ', v_i (pre všetky také i , že $\mu[A_i]$ nie je null).

¹⁴¹ Keď zavedieme jednoznačnú identifikáciu tuplí spolu so špeciálnym atribútom *id*, tieto funkčné závislosti sa nám nestratia. Naopak, ako sme spomenuli, pribudne funkčná závislosť $id \rightarrow K$.

¹⁴² presnejšie: $1 +$ počet atribútov A_i , v ktorých má tupľa nenullovú hodnotu

Tým vlastne vytvoríme tabuľky R_i obsahujúce dvojice kľúč–atribút pre tie tuple relácie R , ktoré mali nenullovú hodnotu v atribúte A_i . Idea vytvorenia binárnych relácií bola prevzatá z [Kowalski, 1978]. V tomto článku autor poznamenáva: „[V tomto prístupe] neznáma informácia sa ľahšie ignoruje a nová informácia sa ľahšie pridáva. Ba čo viac, treba zdôrazniť, že binárne relácie ponúkajú lepší model dátových tabuliek“, totiž „lepšie formalizujú požadovanú vlastnosť, že nezáleží na poradí stĺpcov tabuľky“.

Dekompozícia $\Delta(R) = \langle R', R_1, \dots, R_m \rangle$ sa dá zapísať aj množinovo, a to nasledovne:

$$R' := \{\mu' \mid \mu \in R\},$$

$$R_i := \{\pi_K(\mu) \bowtie \pi_{A_i}(\mu) \mid \mu \in R \wedge \pi_{A_i}(\mu) \cap \{\mathbf{un}, \mathbf{ne}, \mathbf{ni}\} = \emptyset\}, i \in \{1, \dots, m\}.$$

Relácia R' má relačnú schému $W \setminus A, A'$, kde $D(A'_i) = \{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$ pre všetky $i \in \{1, \dots, m\}$, a navyše

$$A' =_{nt} A'_1 \cup \dots \cup A'_m.$$

Priamo z definície máme

$$R' \cap R_i = \emptyset.$$

Ak pochopíme vyššie uvedený postup ako algoritmus tvorby $\Delta(R)$, má zmysel analyzovať jeho zložitosť. Časová zložitosť vzhľadom k počtu tuplí je lineárna, inými slovami, ak počet tuplí relácie R je n , algoritmus trvá rádovo $O(n)$, pretože sa raz prebehne tabuľkou R a pritom sa v každom kroku nanajvýš m krát vkladá tupľa do tabuľky R_i . Pritom berieme hodnotu m za konštantnú pre danú databázu.

Keďže relácie R_i obsahujú iba tuple relácie R neobsahujúce nully, a to presnejšie binárne relácie kľúč–atribút A_i , neobsahujú nijaké nullové hodnoty. Relácia R' obsahuje síce tuple relácie R , ktoré pôvodne obsahovali nully, avšak algoritmom dekompozície boli tieto nully nahradené regulárnymi hodnotami z množiny $\{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$. Výsledné relácie teda neobsahujú nullové hodnoty.

Lahká rekonštrukcia pôvodnej tabuľky presne sleduje dekompozíciu, podobne ako v prípade jediného atribútu s nullom. Nech R' má relačnú schému $W \setminus A, A'$ ($\forall A_i \in A': D(A_i) = \{\text{ÁNO}, \text{NIE}, \text{NEVIEM}\}$). Relácie R_i majú schému K, A_i . Navyše, žiaden z atribútov množiny $W \setminus A$ nesmie nadobúdať nullové hodnoty. Jedine atribúty A_i môžu obsahovať nullové hodnoty typu **un**, **ne** resp. **ni**. Pre každú tupľu $\mu' \in R'$, pokiaľ aspoň v jednom z atribútov množiny A' má hodnotu NIE alebo NEVIEM, musia existovať tuple s rovnakou hodnotou kľúča v tabuľkách R_i (pre každé i také, že atribút A'_i má hodnotu z $\{\text{NIE}, \text{NEVIEM}\}$). Navyše musí platiť

$$\pi_K(R_i) \subseteq \pi_K(R') \text{ pre všetky } i \in \{1, \dots, m\}$$

a relácie R_i musia spĺňať integritné ohraničenia, že $K \rightarrow A_i$ (inak: hodnota kľúča sa v R_i vyskytne najviac raz). Ak tabuľky $\langle R', R_1, \dots, R_m \rangle$ spĺňajú tieto vlastnosti, nazývajú sa *rekonštruovateľné*. Pre tupľu μ' definujeme upravenú tupľu μ'' takto:

$$\mu''[A_i] =_{df} \begin{cases} \tau[A_i], & \text{ak } \pi_{A_i}(\mu') = \text{ÁNO} \wedge (\exists \tau \in R_i : \tau[K] = \mu'[K]), \\ \mu'[A_i/\mathbf{un}], & \text{ak } \pi_{A_i}(\mu') = \text{ÁNO} \wedge \mu'[K] \notin \pi_K(R_i), \\ \mu'[A_i/\mathbf{ne}], & \text{ak } \pi_{A_i}(\mu') = \text{NIE}, \\ \mu'[A_i/\mathbf{ni}], & \text{ak } \pi_{A_i}(\mu') = \text{NEVIEM}, \\ \mu'[A_i] & \text{inak.} \end{cases}$$

Relácia R'' obsahuje pre každú tupľu μ' relácie R' jej zodpovedajúcu upravenú tupľu μ'' . Píšeme $R'' = \nabla(R', R_1, \dots, R_m)$ a hovoríme, že R'' je *rekonštrukciou* tabuliek R', R_1, \dots, R_m .

Je zrejmé, že $R = R''$. Vyplýva to z definovania spätnej úpravy tuplí relácie R' , ktorá bola definovaná ako inverzná operácia ku konštrukcii tabuľky R' . Aj pre prípad viacerých stĺpcov platia nasledujúce rovnosti:

$$\begin{aligned}\nabla(\Delta(R)) &= R, \\ \Delta(\nabla(R)) &= R, \\ \nabla &= \Delta^{-1}.\end{aligned}$$

Operácia rekonštrukcie sa správa presne ako v prípade jedného nulového atribútu. Vzhľadom k počtu prvkov pôvodnej relácie trvá rekonštrukcia rádovo $O(m \cdot n \cdot \log n_i)$, kde n_i je maximum z $\{|R_1|, \dots, |R_m|\}$, a to preto, lebo pre každú tupľu sa vykoná nanajvýš m operácií, zahŕňajúcich prečítanie a zápis (tuple obsahujúce NIE a NEVIEM) a navyše vyhľadanie tuple s rovnakou hodnotou kľúča v tabuľke R_i , podľa ktorej sa modifikuje hodnota atribútu A_i . Keďže m je konštantou pre tabuľku, považujeme rekonštrukciu za *ľahkú*.

Teraz popíšeme stručnejšie ako v jednoatribútovom prípade, ako vyzerá manipulácia s novými dekomponovanými tabuľkami. *Príslušnosť tuple* k relácii $\Delta(R)$ – teda vyhľadanie tuple alebo aj jej odstránenie – sa zrealizuje najjednoduchšie paralelným prehľadávaním tabuliek R' a R_i (a prípadným nasledovným odstránením nájdenej tuple). *Pridanie novej tuple* do $\Delta(R)$ bude predchádzané najprv testom hodnoty atribútu A , či (ne)obsahuje nulové hodnoty. Podľa toho sa priamo pridá do relácie R_A (keď neobsahuje null), alebo sa hodnota atribútu A modifikuje vyššie popísaným spôsobom a takto upravená tupľa sa pridá do relácie R' .

Rovnosť dvoch dekomponovaných tabuliek definujeme nasledovne:

$$\Delta(R) = \Delta(S) \Leftrightarrow_{df} R' = S' \wedge \forall i \in \{1, \dots, m\}: R_i = S_i.$$

Táto rovnosť sa otestuje rovnako ako v prípade klasických tabuliek.

Zjednotenie (kompatibilných) dekomponovaných relácií:

$$\Delta(R) \cup \Delta(S) =_{df} \langle R' \cup S', R_1 \cup S_1, \dots, R_m \cup S_m \rangle.$$

Poznamenajme, že zjednotenie relácií $R_i \cup S_i$ musí zachovať integritné ohraničenie, že hodnota kľúča je jednoznačná (ak teda pre $\mu \in R_i$, $\nu \in S_i$ platí $\mu \neq \nu \wedge \mu[K] = \nu[K]$, potom ani jedna z tuplí μ , ν nie je v $R_i \cup S_i$).

Treba pritom overiť, či platí

$$\Delta(R) \cup \Delta(S) = \Delta(R \cup S),$$

inými slovami, či nami zadefinovaná dekompozícia zachováva zjednotenie relácií.

Poznamenajme, že $\Delta(R \cup S)$ vyzerá z definície nasledovne:

$$\Delta(R \cup S) = \langle (R \cup S)', (R \cup S)_1, \dots, (R \cup S)_m \rangle.$$

Na overenie potrebujeme ukázať, že

$$\langle R' \cup S', R_1 \cup S_1, \dots, R_m \cup S_m \rangle = \langle (R \cup S)', (R \cup S)_1, \dots, (R \cup S)_m \rangle.$$

Vezmime tupľu $\mu \in R' \cup S'$ a všetky tuple z tabuliek $R_i \cup S_i$, ktoré majú v kľúči hodnotu $\mu[K]$ (označíme ich ν_k). Nech bez ujmy na všeobecnosti tupľa μ sa nachádza v R' , čo znamená, že všetky tuple ν_k sa nachádzajú (aspoň) v R_k . Potom existuje tupľa $\mu_1 \in R$,

ktorej zodpovedajú tuple $\mu = (\mu_1)', v_k$.¹⁴³ Potom $\mu_1 \in R \cup S \supseteq R$ a $\mu \in (R \cup S)'$, $v_k \in (R \cup S)_k$. Platí to aj opačne. Vezmime $\mu \in (R \cup S)'$ a všetky tuple z tabuliek $(R \cup S)_m$, ktoré majú v kľúči hodnotu $\mu[K]$ (označíme ich v_k). Potom existuje tupľa $\mu_1 \in R \cup S$, ktorej zodpovedajú tuple $\mu = (\mu_1)', v_k$. Nech $\mu_1 \in R$. Potom nej zodpovedajúce tuple $\mu = (\mu_1)'$ resp. v_k patria do relácií R' resp. R_k , čo znamená, že patria aj do $R' \cup S'$ resp. $R_k \cup S_k$.

Prienik dekomponovaných relácií definujeme podobne ako zjednotenie, a to nasledovne:

$$\Delta(R) \cap \Delta(S) =_{df} \langle R' \cap S', R_1 \cap S_1, \dots, R_m \cap S_m \rangle.$$

Treba pritom overiť, či platí

$$\Delta(R) \cap \Delta(S) = \Delta(R \cap S),$$

presnejšie

$$\langle R' \cap S', R_1 \cap S_1, \dots, R_m \cap S_m \rangle = \langle (R \cap S)', (R \cap S)_1, \dots, (R \cap S)_m \rangle,$$

inými slovami, či nami zadefinovaná dekompozícia zachováva prienik relácií.

Vezmime tupľu $\mu \in R' \cap S'$ a všetky tuple z tabuliek $R_i \cap S_i$, ktoré majú v kľúči hodnotu $\mu[K]$ (označíme ich v_k). Teda tupľa μ sa nachádza v R' aj v S' , čo znamená, že všetky tuple v_k sa nachádzajú aj v R_k , aj v S_k . Potom existuje tupľa $\mu_1 \in R$ a tupľa $\mu_2 \in S$, ktorej zodpovedajú tuple $\mu = (\mu_1)', v_k$ a tuple $\mu = (\mu_2)', v_k$. Takže $\mu_1 = \mu_2$. Potom $\mu_1 \in R \cap S$ a $\mu \in (R \cap S)', v_k \in (R \cap S)_k$. Platí to aj opačne. Vezmime $\mu \in (R \cap S)'$ a všetky tuple z tabuliek $(R \cap S)_m$, ktoré majú v kľúči hodnotu $\mu[K]$ (označíme ich v_k). Potom existuje tupľa $\mu_1 \in R \cap S$, ktorej zodpovedajú tuple $\mu = (\mu_1)', v_k$, a tieto sú v oboch reláciách R' , S' resp. R_k, S_k , čo znamená, že patria aj do $R' \cap S'$ resp. $R_k \cap S_k$.

Rozdiel dekomponovaných relácií definujeme odlišne ako zjednotenie a prienik, a to nasledovne:

$$\Delta(R) \setminus \Delta(S) =_{df} \langle R' \setminus S', T_1, \dots, T_m \rangle,$$

kde

$$P := \pi_K(R' \setminus S'),$$

$$T_i := \{v \in R_i \mid v[K] \in P\} \text{ pre každé } i \in \{1, \dots, m\}.$$

Odlišnosť spočíva v tom, že nemôžeme jednoducho počítat rozdiel množín $R_i \setminus S_i$, pretože môže sa stať, že existuje tupľa $\mu \in R$ a $v \in S$, ktoré sú odlišné, majú rovnakú hodnotu kľúča a jedného z atribútov A_j . Takto by sme síce odstránili tupľu v z $R' \setminus S'$, avšak zničili by sme aj tupľu $\mu_j = v_j \in R_j \cap S_j$. Treba pritom podotknúť, že jednoznačnosť kľúča „ospravedľňuje“ našu na prvý pohľad bezhlavú aplikáciu rozdielu \setminus na relácie R', S' . Dá sa ukázať, že platí

$$\Delta(R) \setminus \Delta(S) = \Delta(R \setminus S).$$

¹⁴³ Môže sa stať, žeby pre nejaké τ v relácii $R_i \cup S_i$ (niektoré z v_k) bolo $\tau[K] = \mu[K]$, a pritom $\mu[A'_i] \neq \tau[A'_i]$? (Predpokladáme, že $\mu \in R'$.) Ak by sa tak stalo, tak buď $\tau \in R_i$ alebo $\tau \in S_i$. Prvá možnosť nemôže nastať, lebo by to znamenalo, že v tabuľka R bola chybné dekomponovaná. Druhá možnosť: ak $\tau \in S_i$, tak potom je to v protiklade s dodržaním integritného ohraničenia o jednoznačnosti hodnoty kľúča, pretože to znamená, že pôvodné tuple v R a S mali odlišnú hodnotu v atribúte A_i , a teda v zjednotení $R_i \cup S_i$ nemajú čo robiť.

Projekcia dekomponovaných tabuliek nespôsobuje ťažkosti (pritom zakazujeme odprojektovať kľúč). Definujeme

$$\pi_W(\langle R', R_1, \dots, R_m \rangle) =_{df} \langle S', S_1, \dots, S_k \rangle,$$

kde W je podmnožina množiny atribútov relácie R , $K \notin W$, nech pre zjednodušenie $W \cap A = \{A_1, \dots, A_k\}$ a dekomponovaná relácia S je daná nasledovne:

$$S' := \pi_W(R'),$$

$$S_j := \pi_W(R_j) \text{ pre každé } j \in \{1, \dots, k\}.$$

Teda relácie, ktoré neobsahujú atribút z W nie sú vo výslednej $\pi_W(\Delta(R))$.

Ľahko sa dá ukázať, že

$$\pi_W(\Delta(R)) = \Delta(\pi_W(R)).$$

Selekcia tuplí dekomponovanej tabuľky je najkomplikovanejšia operácia, pretože výraz F , pomocou ktorého sa selektuje, môže obsahovať výraz používajúci viacero atribútov, v ktorých sú nullové hodnoty, a ktoré uchováваме *každý zvlášť* (v samostatnej tabuľke R_i). Definícia true-selekcie bude teda trochu krkolomná, avšak našim prvoradým cieľom je poskytnúť alternatívu k trom typom nullov, a to model bez nullov.

True-selekciu definujeme rekurzívne vzhľadom na výraz F :

$$\sigma_F(T) =_{df} \begin{cases} \sigma_{F_1}(T) \cup \sigma_{F_2}(T), & \text{ak } F \equiv F_1 \vee F_2, \\ \sigma_{F_1}(T) \cap \sigma_{F_2}(T), & \text{ak } F \equiv F_1 \wedge F_2, \\ T \setminus \sigma_{F_1}(T), & \text{ak } F \equiv \neg F_1, \end{cases}$$

a pre možnosť $F \equiv (M = a)$ resp. $F \equiv (M = N)$ pre $M, N \in W$, $a \in D(M)$ (a je konštanta) už špeciálne pre dekomponované tabuľky ($\Delta(R) = \langle R', R_1, \dots, R_m \rangle$) definujeme takto:

$$\sigma_F(\Delta(R)) =_{df} \langle S', S_1, \dots, S_k \rangle,$$

kde S', S_1, \dots, S_k sú podľa jednotlivých možností dané nasledovne.

1. prípad: $F \equiv (M = a)$, $M \in W \setminus A$ alebo $F \equiv (M = N)$, $M, N \in W \setminus A$.

$$S' := \sigma_F(R'),$$

$$S'_i := \{v \in R_i \mid v[K] \in \pi_K(S')\} \text{ pre každé } i \in \{1, \dots, m\}.$$

2. prípad: $F \equiv (M = a)$, $M = A_k$ pre nejaké konkrétne $k \in \{1, \dots, m\}$.

$$S'_k := \sigma_{M=a}(R'_k),$$

$$S' := \{\mu \in R' \mid \mu[K] \in \pi_K(S'_k)\},$$

$$S'_i := \{v \in R_i \mid v[K] \in \pi_K(S'_k)\} \text{ pre každé } i \in \{1, \dots, k-1, k+1, \dots, m\}.$$

3. prípad: $F \equiv (M = N)$, $M = A_k$ pre nejaké konkrétne $k \in \{1, \dots, m\}$, $N \in W \setminus A$.

$$P := \pi_K(\sigma_F(R_k \bowtie \pi_{KN}(R'))),$$

$$S' := R' \cap P,$$

$$S'_i := R'_i \cap P \text{ pre každé } i \in \{1, \dots, m\}.$$

4. prípad: $F \equiv (M = N)$, $M = A_k$, $N = A_p$ pre nejaké konkrétne $k \neq p \in \{1, \dots, m\}$.

$$P := \pi_K(\sigma_F(R_k \bowtie R_p)),$$

$$S' := R' \cap P,$$

$$S'_i := R'_i \cap P \text{ pre každé } i \in \{1, \dots, m\}.$$

Pritom vo všetkých pravých stranách výrazov uvedených v predchádzajúcich prípadoch sú použité operácie na klasických reláciách. Možno sa zdá, že sme zabudli na porovnania medzi jednotlivými nullovými hodnotami, avšak poznamenajme, že náš model vyhodnocuje každé porovnanie s nulom (v prípade dekomponovaných tabuliek s indikátorom ÁNO, NIE, NEVIEM) buď *maybe* alebo *false* (Tabuľka 8).

Maybe-selekcia: Zdefinujeme ju pomocou *true*-selekcie, a to na základe nasledovného poznatku: Nech $F(\mu) = \text{false}$, potom $\mu \in \sigma_{\neg F}(R)$, pretože $\neg F(\mu) = \text{true}$. Definujeme

$$\sigma_{\omega F}(\Delta R) =_{df} R \setminus (\sigma_F(\Delta R) \cup \sigma_{\neg F}(\Delta R)).$$

Kartézsky súčin definujeme najprv pre súčin jednotupľovej dekomponovanej relácie $J = \langle \gamma', J_1, \dots, J_m \rangle$ s dekomponovanou reláciou $\langle S', S_1, \dots, S_k \rangle$. Pritom J_i obsahuje najviac jednu tupľu (podľa hodnoty $\gamma[A_i]$). Označíme ju χ_i .¹⁴⁴ Keďže vznikajú úplne nové tuple, treba im priradiť úplne novú kľúčovú hodnotu. Pre hodnotu kľúča *id* označíme túto novú hodnotu $\Phi(id)$.¹⁴⁵ Definujeme

$$\langle \gamma', J_1, \dots, J_m \rangle \times \langle S', S_1, \dots, S_k \rangle =_{df} \langle T', T_1, \dots, T_k \rangle,$$

kde

$$\begin{aligned} T' &:= \{(\gamma' \times \mu')[K/\Phi(\mu'[K])] \mid \mu' \in S'\}, \\ T_i &:= \{v[K/\Phi(v[K])] \mid v \in S_i\} \text{ pre každé } i \in \{1, \dots, m\}. \end{aligned}$$

Teraz sme už pripravení definovať kartézsky súčin pre dve dekomponované relácie. Kladieme

$$\langle R', R_1, \dots, R_m \rangle \times \langle S', S_1, \dots, S_k \rangle =_{df} \bigvee_{\substack{\gamma' \in R', \\ \chi_i \in R_i, \\ \forall i: \chi_i[K] = \gamma'[K]}} \langle \gamma', \{\chi_1\}, \dots, \{\chi_m\} \rangle \times \langle S', S_1, \dots, S_k \rangle,$$

pričom možno niektoré $\{\chi_i\}$ budú pre mnohé γ' prázdne.

Dokážeme, že pre *true*-selekciu platí:

$$\Delta(\sigma_F R) = \sigma_F(\Delta(R)).$$

Dôkaz vykonáme štruktúrnou indukciou vzhľadom na výraz F .

1. $F \equiv F_1 \wedge F_2$: Podľa definície $\Delta(\sigma_F R) = \Delta(\sigma_{F_1}(R) \cap \sigma_{F_2}(R))$, čo podľa rovnosti pre dekompozíciu prieniku je rovné $\Delta \sigma_{F_1}(R) \cap \Delta \sigma_{F_2}(R) =: P$. Podľa indukčného predpokladu (lebo výrazy F_1, F_2 sú jednoduchšie ako F) platí $\Delta \sigma_{F_i}(R) = \sigma_{F_i}(\Delta(R))$ pre oboje $i \in \{1, 2\}$, a teda prienik P sa podľa definície rovná $\sigma_F(\Delta(R))$.
2. $F \equiv F_1 \vee F_2$: Ako predošlý prípad.
3. $F \equiv \neg F_1$: $\Delta(\sigma_F R) = \Delta(R \setminus \sigma_{F_1}(R)) \stackrel{(1)}{=} \Delta(R) \setminus \Delta(\sigma_{F_1}(R)) \stackrel{(2)}{=} \Delta(R) \setminus \sigma_{F_1}(\Delta(R)) = \sigma_F(\Delta(R))$, pričom rovnosť (1) platí na základe rovnosti pre rozdiel a rovnosť (2) platí podľa indukčného predpokladu.
4. $F \equiv (M = a)$ resp. $F \equiv (M = N)$ pre $M, N \in W$, $a \in D(M)$ (a je konštanta): Ukážeme pre 2. prípad. Teda $F \equiv (M = a)$, $M = A_k$ pre nejaké konkrétne $k \in \{1, \dots, m\}$. Nech $\mu \in R$ je taká, že platí $F(\mu)$, t.j. $\mu[A_k] = a$. Ukážeme, že $\langle \mu', \mu_1, \dots, \mu_m \rangle$ (jej zodpovedajúce tuple po dekompozícii) patria ako do $\Delta(\sigma_F R)$, tak aj do $\sigma_F(\Delta(R))$. Keďže platí $F(\mu)$, tak $\mu \in \sigma_F R$, a teda potom $\langle \mu', \mu_1, \dots, \mu_m \rangle \in \Delta(\sigma_F R)$. Pre druhú množinu, $\langle \mu', \mu_1, \dots, \mu_m \rangle \in \Delta(R)$. Potom z definície *true*-selekcie $\mu_k \in S'_k$, a rovnako $\mu' \in S'$, pretože $\mu'[K] = \mu_k[K] \in \pi_K(S'_k)$. Rovnako tak ostatné $\mu_i \in S'_i$ pre každé $i \in \{1, \dots, k -$

¹⁴⁴ Teda ak $J_i \neq \emptyset$, tak $\{\chi_i\} = J_i$ a $\chi_i[K] = \gamma'[K]$.

¹⁴⁵ Použitý znak symbolizuje, že ide o istý druh orákula (preškrtnuté O). Naviac, Φ (*fi*) naznačuje, že v konkrétnej aplikácii musí ísť o funkciu. Takéto použitie „neznámej funkcie“ pre generovanie jednoznačných, ešte v systéme neobsiahnutých dát, je v prácach bežné.

$1, k + 1, \dots, m\}$. Keďže kľúč sa vyskytuje v reláciách R', R_i jednoznačne, platí aj tvrdenie pre $\mu \in R$ je také, že neplatí $F(\mu)$, t.j. $\mu[A_k] \neq a$. Tým sme ukázali, že platí

$$\Delta(\sigma_{A_k=a}(R)) = \sigma_{A_k=a}(\Delta(R)).$$

Ostatné prípady sa ukážu analogicky.

Napokon F -join definujeme klasicky, t.j. $\Delta(R) \bowtie_F \Delta(S) =_{df} \sigma_F(\Delta(R) \times \Delta(S))$.

Všetky relačné operátory boli definované tak, aby zachovali výsledok aj po aplikovaní dekompozície na pôvodné tabuľky, aj po aplikovaní na najprv dekomponované tabuľky.

Čo sa týka reprezentácie, je potrebné definovať reprezentáciu pre tabuľky s nullmi typu **un**, **ne** a **ni**, rovnako ako reprezentáciu pre dekomponované tabuľky. Reprezentáciu iba naznačíme:

1. krok: Namiesto tabuľky s **ni** nullom utvoríme dve tabuľky, pričom jedna z nich má na mieste **ni** nullu **un** null, druhá tam má **ne** null. (*Odstránenie ni nullov.*)
2. krok: Pre každú tupľu, ktorá obsahuje **ne** null(y) urobíme nasledovné: Vytvor tupľu nad schémou bez atribútov, kde má **ne** null(y) a zaraď ju do tabuľky s takouto schémou; ak taká zatiaľ neexistuje, tak ju vytvor. (*Odstránenie ne nullov.*)
3. krok: Každá tabuľka vytvorená predchádzajúcimi krokmi bude obsahovať nanajvyš **un** null(y). Pre každú tabuľku vytvor $Rep(T)$ tak, ako keby to bola – akože de facto aj je – Coddova tabuľka.

Výsledná reprezentácia je zjednotenie všetkých reprezentácií. Poznamenajme, že na rozdiel od klasickej reprezentácie, keď výsledkom sú relácie s rovnakou relačnou schémou, sa v prípade práve definovanej reprezentácie relácií s nullmi typu **un**, **ne** a **ni** môžu vyskytnúť dve zásadné odlišnosti:

1. Prvok reprezentácie relácie nie je nevyhnutne tabuľka, ale môže byť aj „databáza“, množina relácií.
2. Reprezentácia relácie obsahuje relácie (resp. podľa predošlého bodu aj množiny relácií) nad rôznymi schémami.

Reprezentáciu dekomponovaných tabuliek by sme definovali tak, aby platilo

$$Rep(\Delta(R)) = \Delta Rep(R).$$

Potom z uvedeného vyplýva, že tabuľky s nullmi typu **un**, **ne** a **ni** sú nanajvyš tak dobré ako Coddove tabuľky (pretože Coddove tabuľky tvoria podmnožinu tabuliek s nullmi typu **un**, **ne** a **ni**). Sú charakterizované výsledkami z časti 3.3.4.

Ako sa „zozložitia dotazy“ na dekomponované tabuľky? Najcitlivejšie miesto je selekcia, a teda aj join. Takže možno konštatovať, že dotazy sa veľmi skomplikujú, avšak cieľom je ukázať, že bez nullov sa môžeme istým spôsobom zaobiť.

Čo sa týka veľkosti nových tabuliek, situácia je komplikovanejšia ako v prípade jedného stĺpca pripúšťajúceho nullu. Veľkosť tabuľky R' oproti pôvodnej tabuľke R závisí od hustoty nullov, presnejšie, od nasledovných parametrov:

- m : počet stĺpcov pripúšťajúcich nullu,
- n : počet tuplí relácie R ,
- $n(i)$: počet tuplí, ktoré majú v atribúte A_i nenullovú hodnotu,
- $V(J)$: potrebná dátová veľkosť na uchovanie hodnoty atribútu A (pritom $J \in \cup_{i \in \{1, \dots, m\}} \{A_i, A'_i\} \cup \{K\}$),
- Ost : veľkosť tabuľky R bez stĺpcov A_i („ostatné stĺpce“).

Objem tabuľky R označíme $V(R)$, objem tabuľky ΔR označíme $V(\Delta R)$. Ich jednotlivé hodnoty sú dané vzťahmi uvedenými v časti 7.3. Závislosť medzi $V(R)$ a $V(\Delta R)$ určuje, či je použitie nami definovanej dekompozície efektívnejšie, ako pôvodná reprezentácia tabuľky R . Uvažujeme pritom, že pôvodná relácia je reprezentovaná záznamami pevnej dĺžky a že na reprezentáciu neúplnej informácie v nových tabuľkách (atribúty A'_i) stačí priestor dvoch bitov (kódovanie troch hodnôt ÁNO, NIE, NEVIEM).

Ako vyplýva z výpočtov, zlepšenia nastávajú v prípadoch, keď sú hodnoty $V(A_i)$ rádovo väčšie ako hodnoty $V(A'_i)$, čo však väčšinou nastáva, pretože sa ukazuje užitočné predpokladať, že atribúty pripúšťajúce nulové hodnoty nadobúdajú napr. reťazcové hodnoty. Ďalšia kritická hodnota, od ktorej závisí, či dekompozícia nebude dokonca zhoršovať objem relácie, je počet (ne)nulových tuplí resp. pomer nulových a nenulových tuplí. Pre podrobnejšie údaje pozri dodatok, časť 7.3 (str. 96). Prínos dekompozície však nie je v redukcii priestoru na uchovanie dát, ale v tom, že bez nulov (základných typov) sa dá zaobiť.

6 Fyzické aspekty neúplnej informácie

„Fyzické médiá zastarajú veľmi rýchlo,
aj softvér po pár rokoch už bude úplne iný,
a ten dnešný bude nepoužiteľný.
Ale dáta – dáta sú večné.“
Ján Štunc

Základným fyzickým aspektom každej informácie je náročnosť jej uchovávanania na záznamových médiách, aby to bolo výhodné objemovo a súčasne z hľadiska prístupu k dátam, ako sme to spomenuli už pri návrhu dekompozície. Keďže vieme, že v oblasti hardvéru a fyzického uloženia dát ide pokrok azda rýchlejšie, ako dokáže priemerný informatik sledovať, zhrnieme fyzické aspekty neúplnej informácie na jednu hutnú stranu. Podľa typu nullov resp. neúplnej informácie môžeme navrhnúť:

- *nully* (atomické, Biskupove, Zaniolove): reprezentovať ako *bitové hodnoty*, a to aj v prípade viacerých druhov nullov (porov. časť 5.5, str. 79), prístup je rýchly, aj porovnávanie; resp. reprezentovať ako Oracle, teda iba 1 bytom (dĺžka záznamu),
- *indexované nully*: reprezentovať ako *celočíselné hodnoty* (rôzne číslo znamenajúce rozličnú indexovanú nullovú hodnotu), resp. ako smerníky do tabuliek obsahujúcich nullové hodnoty (v prípade, že na úrovni systému sa poskytuje mechanizmus odlíšenia smerníka od regulárnej hodnoty)¹⁴⁶, pritom je prirodzené, aby systém poskytoval isté obmedzenia na počet rôznych nullových hodnôt,
- *intervalové hodnoty*: dvojica hodnôt daného typu, pričom technicky je to riešiteľné dvojakým spôsobom:
 1. regulárna hodnota alebo prvá časť intervalu je uchovávaná v prvom stĺpci, prípadná druhá hodnota intervalového nullu je uchovávaná v druhom stĺpci, ktorý pre regulárnu hodnotu nemá žiadny význam (môže to byť naznačené špeciálnym symbolom)¹⁴⁷,
 2. regulárna hodnota resp. prvá časť intervalu ako v predošlom prípade, a príznak, ktorý značí (podľa 0/1), či ide o regulárnu hodnotu alebo o intervalovú hodnotu; ak ide o interval, druhá časť je uložená v inej, špeciálnej tabuľke¹⁴⁸,
- *regulárne výrazy*: uchovávať ako reťazce (stringy), pričom je potrebný mechanizmus skenovania a parsovania reťazca za účelom vybudovania vnútornej reprezentácie regulárneho výrazu za účelom vyhodnocovania dotazov; takýto mechanizmus už mnohé softvérové balíky obsahujú (tzv. vnútorný kompilátor),
- *množinová neúplná informácia*: keďže dopredu nie je možné odhadnúť počet členov množiny, ktorá reprezentuje neúplnú informáciu, najlepší spôsob je reprezentovať túto množinu ako (hašovaný) zoznam, prípadne strom (ak ide o porovnateľné hodnoty) alebo ekvivalentnú štruktúru, pritom fyzicky bude samozrejme reprezentovaná ako sekvencia niekde v oddelenom segmente daného databázového súboru; prístup aj rekonštrukcia takejto neúplnej informácie je náročná na čas.

¹⁴⁶ porov. nevýhodu relačného modelu, ktorá sa nazýva *vertikálna homogenita*, časť 2.3.1, str. 10

¹⁴⁷ riešenie spočívajúce vo vložení rovnakej hodnoty, ako je v prvom stĺpci, pre regulárnu hodnotu, nie je rozumné z hľadiska zložitejšieho mechanizmu zmeny hodnoty – bolo by potrebné meniť nie jednu, ale obe hodnoty, pričom treba zaručiť, že sú obidve rovnaké (ľahko môže nastať narušenie takto definovanej integrity)

¹⁴⁸ takto sme postupovali aj v nami navrhovanej dekompozícii

7 Dodatok

7.1 Prevod všeobecných relácií na klasické relácie

Nech I je konečná množina indexov $\{1, \dots, m\}$ a všeobecná relácia

$$R = \{\vartheta_i \mid i \in I, \vartheta_i = \langle M_1, \dots, M_n \rangle, M_1 \times \dots \times M_n \subseteq D_1 \times \dots \times D_n\}.$$

Predpokladáme interpretáciu všeobecných relácií tak, že ich reprezentácie obsahujú *všetky* tuple zložené z prvkov, ktoré sa vyskytujú v zodpovedajúcich podmnožinách domén D_i daných komponentov všeobecnej tuple. Ponúka sa teda priamočiare riešenie: skonštruovanie klasickej relácie

$$R' = \cup_{i \in I} \{\langle a_1, \dots, a_n \rangle \mid a_j \in M_j, j \in \{1, \dots, n\}, \vartheta_i = \langle M_1, \dots, M_n \rangle\}.$$

Opačne, ak máme reláciu R' , zloženú z tuplíc $\mu_i = \langle a_{i1}, \dots, a_{in} \rangle, i \in I$, jedna z k nej ekvivalentných všeobecných relácií vyzerá nasledovne:

$$R = \{\{\langle a_{i1} \rangle, \dots, \langle a_{in} \rangle\} \mid i \in I, \mu_i \in R'\}.$$

Ak chceme zmenšiť počet všeobecných tuplíc, musíme nanajvýš v kvadratickom čase prebehnúť nimi všetkými a pozlúčovať do väčších množín. Toto zlučovanie môže byť implementované rozlične, a teda hľadanie všeobecnej relácie k danej relácii nezachováva syntaktickú ekvivalenciu všeobecných relácií. Uvádza to aj nasledujúci príklad. Hoci všeobecné relácie

$$R_1 := \{\{\langle 1, 2 \rangle, \langle a, b \rangle\}, \{\langle 1 \rangle, \langle a, c \rangle\}\}$$

$$R_2 := \{\{\langle 1 \rangle, \langle a, b, c \rangle\}, \{\langle 2 \rangle, \langle a, b \rangle\}\}$$

sú syntakticky rôzne, im ekvivalentná relácia

$$\{\langle 1, a \rangle, \langle 1, b \rangle, \langle 1, c \rangle, \langle 2, a \rangle, \langle 2, b \rangle\}$$

je pre ne rovnaká. Zlučovaním totiž môžeme dospieť ako k R_1 , tak aj k R_2 . Späťne teda nevieme skonštruovať pôvodnú všeobecnú reláciu a ponúka sa nám viacero možností.

Táto nejednoznačnosť je však v poriadku, ak si uvedomíme, že *vzhľadom k našej interpretácii* všeobecných relácií sú R_1, R_2 nerozlíšiteľné, teda (sémanticky) *ekvivalentné*.

7.2 Značenie

označenie, symbol

vysvetlenie

subskript $_{nt}$	notačne (označenie)
subskript $_{df}$	definitoricky
$:=$	priradenie hodnoty symbolu
$\langle \dots \rangle$, napr. $\langle a, b, c \rangle$	usporiadaná n -tica, trojica prvkov a, b, c z množiny $A \times B \times C$ ($a \in A, b \in B, c \in C$)
$\langle A: a, B: b \rangle$	to isté ako $\langle a, b \rangle \in A \times B$
malé grécke písmená, napr. μ	tuple
$\bowtie, \pi, \sigma, \cup, \cap, \setminus$	relačné operácie
alebo, a resp. not, and, or, maybe	logické operácie
$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow, \otimes$	logické spojky (and, or, not, impl, eq, xor)
<i>true, T, false, F, maybe, unk, M</i> resp. ω	logické konštanty (pravdivostné hodnoty)
$D(A)$	doména zodpovedajúca atribútu A
\equiv	syntaktická rovnosť (ekvivalencia)
un	„unknown“ null
ni	„no information“ null
ne	„nonexistent“/„inapplicable“ null
Δ, ∇	dekompozícia, rekonštrukcia

7.3 Charakteristika navrhnuitej dekompozície

V tejto časti dodatku zhŕňame výsledky výpočtov, ktoré charakterizujú navrhovanú dekompozíciu a jej použiteľnosť v praxi. Veľkosť tabuľky R' oproti pôvodnej tabuľke R Budeme značiť ako v časti 5.5.2, teda nasledovne:

- n : počet tuplí relácie R ,
- m : počet stĺpcov pripúšťajúcich nully,
- $n(i)$: počet tuplí, ktoré majú v atribúte A_i nenullovú hodnotu,
- $V(J)$: potrebná dátová veľkosť na uchovanie hodnoty atribútu A (pritom $J \in \cup_{i \in \{1, \dots, m\}} \{A_i, A'_i\} \cup \{K\}$),
- Ost : veľkosť tabuľky R bez stĺpcov A_i („ostatné stĺpce“).

Pre objem tabuľky R (označenie $V(R)$), objem tabuľky ΔR (označenie $V(\Delta R)$) platia nasledovné vzťahy:

$$\begin{aligned} V(R) &= n \cdot \left(\sum_{i=1}^m V(A_i) + Ost \right), \\ V(R') &= n \cdot \left(\sum_{i=1}^m V(A'_i) + Ost \right), \\ V(R_{A_i}) &= n(i) \cdot (V(K) + V(A_i)), \\ V(\Delta R) &= V(R') + \sum_{i=1}^m V(R_{A_i}). \end{aligned}$$

Budeme sa sústreďovať na závislosť medzi $V(R)$ a $V(\Delta R)$, ktorá určuje, či je použitie nami definovanej dekompozície objemovo efektívnejšie ako pôvodná reprezentácia tabuľky R . Uvažujeme o databáze s $n = 1000$ tupľami. Hodnoty objemov dát sú v bytoch. Na veľkosť kľúča postačia 4 byty, čo je 32 bitov (dá sa reprezentovať číslo 0 až 4 294 967 295, teda približne 4 miliardy rôznych tuplí). Predpokladáme ďalej, že všetky nahradené nully (atribúty A'_i) sa reprezentujú 1 bitom, teda $V(A'_i) = 1\text{bit}$.

Napokon ešte uvádzame tieto štatistické hodnoty:

- $\%nullcol$: percento nullových stĺpcov (vzhľadom k veľkosti),
- $\%nulls$: priemerné percento pomeru počtu nullových tuplí k nenullovým v stĺpcoch, čo povoľujú nully.

Sú vyjadrené nasledovne:

$$\begin{aligned} \%nullcol &= \frac{\sum_{i=1}^m V(A_i)}{\sum_{i=1}^m V(A_i) + Ost}, \\ \%nulls &= \frac{\sum_{i=1}^m V(A_i) \cdot \left(1 - \frac{n(i)}{n}\right)}{m \cdot \sum_{i=1}^m V(A_i)}. \end{aligned}$$

Predpokladajme, že nully sú dovolené v $m = 5$ stĺpcoch. Pre zjednodušenie budeme všetky hodnoty $V(A_i)$ pre atribúty s nullmi považovať za rovnaké. Nasledujúce body charakterizujú databázy podľa toho, „ako husto“ nullové hodnoty obsahujú.

1. *Hustá databáza*: je databáza, kde sa nulové hodnoty vyskytujú v tretine až polovici stĺpcov (nielen čo sa počtu týka, ale čo sa objemu dát týka), a to dosť husto, presnejšie, viac ako 50%. Objem dekomponovanej relácie je zhruba o 10–30% menší ako pôvodná relácia.

$m = 5, V(A_i) = 10$ bytov							
$n(1)$	200	200	100	500	200	100	500
$n(2)$	200	200	200	500	200	200	500
$n(3)$	200	200	300	500	200	300	500
$n(4)$	200	200	400	500	200	400	500
$n(5)$	200	200	500	500	200	500	500
<i>Ost</i>	200	100	100	100	50	50	50
$\%nullcol$	20%	33,3%	33,3%	33,3%	50%	50%	50%
$\%nulls$	80%	80%	70%	50%	80%	70%	50%
$V(\Delta R):V(R)$	85,9%	74,4%	81,1%	90,4%	64,6%	71,6%	85,6%

2. *Stredne hustá databáza*: je databáza, kde sa nulové hodnoty vyskytujú nanajvýš v tretine stĺpcov, a to dosť husto, presnejšie, viac ako 50%. Objem dekomponovanej relácie je približne rovnaký ako pôvodná relácia.

$m = 5, V(A_i) = 4$ byty							
$n(1)$	200	200	100	500	200	100	500
$n(2)$	200	200	200	500	200	200	500
$n(3)$	200	200	300	500	200	300	500
$n(4)$	200	200	400	500	200	400	500
$n(5)$	200	200	500	500	200	500	500
<i>Ost</i>	200	100	100	100	50	50	50
$\%nullcol$	9,1%	16,7%	16,7%	16,7%	28,6%	28,6%	28,6%
$\%nulls$	80%	80%	70%	50%	80%	70%	50%
$V(\Delta R):V(R)$	94,8%	90,5%	93,9%	100,5%	83,8%	89,5%	100,9%

3. *Stredne riedka databáza*: je databáza, kde sa nulové hodnoty vyskytujú nanajvýš v tretine stĺpcov, a to pomerne riedko, nanajvýš 30%. Objem dekomponovanej relácie je priemerne o 10% väčší, ako pôvodná relácia.

$m = 5, V(A_i) = 4$ byty					
$n(1)$	800	800	800	1000	1000
$n(2)$	800	700	800	900	900
$n(3)$	800	600	800	800	800
$n(4)$	800	500	800	900	900
$n(5)$	800	400	800	900	900
<i>Ost</i>	100	100	200	100	200
$\%nullcol$	16,7%	16,7%	9,1%	16,7%	9,1%
$\%nulls$	20%	40%	20%	10%	10%
$V(\Delta R):V(R)$	110,5%	103,9%	105,7%	113,9%	107,6%

4. *Riedka databáza*: je databáza, kde sa nulové hodnoty vyskytujú nanajvýš v 10% stĺpcov, a to veľmi zriedka, nanajvýš 5%. Objem dekomponovanej relácie je o 10% väčší, ako pôvodná relácia, čo je značne veľa.

	$m = 5, V(A_i) = 4$ byty			
$n(1)$	1000	1000	1000	1000
$n(2)$	1000	1000	1000	1000
$n(3)$	900	1000	1000	1000
$n(4)$	900	900	900	1000
$n(5)$	900	900	900	900
<i>Ost</i>	200	200	500	400
%nullcol	9,1%	9,1%	3,8%	4,8%
%nulls	6%	4%	4%	2%
$V(\Delta R):V(R)$	108,3%	108,6%	103,7%	104,7%

5. *Extrémne databázy*: databázy, kde sa nulové hodnoty vyskytujú v malom množstve stĺpcov, ale veľmi často, a naopak, vo veľa atribútoch, ale zriedkavo.

	$m = 5, V(A_i) = 4$ byty			
$n(1)$	200	100	990	900
$n(2)$	200	200	990	900
$n(3)$	200	300	990	900
$n(4)$	200	400	900	900
$n(5)$	200	500	900	900
<i>Ost</i>	200	500	10	20
%nullcol	9,1%	3,8%	66,7%	50%
%nulls	80%	70%	4,6%	10%
$V(\Delta R):V(R)$	94,8%	98,6%	162,6%	141,6%

Z tabuľky vidieť, že najmenej „príjemný“ prípad pre dekompozíciu nastáva vtedy, ak sa nully vyskytujú v takmer všetkých stĺpcoch, avšak celkovo sa v nich vyskytujú zriedka. Odtiaľ plynie dôležité pravidlo pre etapu návrhu tabuliek: koncentrujte výskyt nullov do čo najmenšieho množstva atribútov.

Celkovo možno povedať, že dekompozícia neprináša výrazné zlepšenie, čo sa týka objemu, avšak neprináša – až na „nepříjemný“ prípad, ktorý treba odstrániť v etape návrhu relácií – ani zhoršenie. Ako sme už spomenuli, jej základný prínos nie je v redukcii priestoru na uchovanie dát, ale v teoretickom poznatku, že bez nullov (základných typov) sa dá zaobísť.

7.4 Nulové hodnoty v praxi

7.4.1 Null v štandarde SQL92

NULL v SQL je vlastne len značka, ktorá sa správa ako Coddova hodnota ω (null typu „unknown“, **un**). Štandard podporuje trojhodnotovú logiku s nullmi. Null v skutočnosti je iba značka, nie hodnota. Nie je možné porovnávať s nullom. Test sa musí vykonať pomocou operátorov IS NULL alebo IS NOT NULL, ktoré zodpovedajú operátoru IS_UNK z trojhodnotovej logiky. Tieto operátory vždy vracajú hodnotu z $\{true, false\}$.

Revízia SQL štandardu známa ako SQL3 zavádza novú, bohatšiu podporu pre narábanie s chýbajúcou a neúplnou informáciou. Sú zavedené používateľsky definované nully. Používateľ sám môže zdefinovať vlastný null typ (NULL CLASS v SQL3) a špecifikovať, ktorý ním zdefinovaný nullový *typ aplikovať na ktorú doménu. Taktiež je zavedený predikát, ktorý dokáže rozlíšiť dva nully. Je to predikát DISTINCT, ktorý sa používa nasledovne:

hodnota₁ IS DISTINCT FROM hodnota₂

a vyhodnotí sa na *false*, ak obe sú nenullové a *hodnota₁ = hodnota₂* alebo obe sú toho istého null typu. V opačnom prípade sa predošlý výraz vyhodnotí na *true*.

Je pridaný vstavaný dátový typ **BOOLEAN**. Keďže SQL je postavené na základoch trojhodnotovej logiky, tento typ obsahuje tri hodnoty, *true*, *false* a *unknown*. Tretia pravdivostná hodnota, *unknown* (alebo len *unk*), vlastne reprezentuje null. Je výsledkom porovnania medzi výrazmi, z ktorých aspoň jeden má hodnotu null. (Tým samozrejme vznikajú paradoxy, ktoré boli spomínané v Coddovom prístupe, str. 31. V tomto smere je štandard SQL dogmatickým a nevystihuje požadované správanie. Štandard by mal vychádzať z precízneho popisu správania sa navrhovaných nullov, a potom by sa mal zaoberať návrhom operácií/spôsobov, ako toto želané správanie sa dosiahnuť.)

Po podrobnom skúmaní sémantiky jednotlivých operátorov SQL je zrejmé, prečo (pri manipulácii s pravdivostnou hodnotou *unknown*) nasledujúce výrazy nie sú ekvivalentné:

p IS NOT TRUE

a

NOT *p*.

Stačí si uvedomiť, že prvý výraz znamená (je isté, že) „*p* nie je pravdivé“, kým druhý (je isté, že) „*p* je nepravdivé“.¹⁴⁹

Čo sa týka vyhodnocovania dotazov a výrazov, nie sú tuple vyhodnotené *unknown* zahrnuté do odpovedí. (Ide o Coddovu *true*-verziu dotazov.) Presnejšie, keď SQL aplikuje **WHERE** časť selektu na tabuľku *T*, vylúči všetky riadky tabuľky *T*, pre ktoré sa podmienkový výraz tej **WHERE** časti vyhodnotí inak ako *true*, teda na *false* alebo *unknown*. Dá sa teda povedať, že za účelom vyhodnocovania **WHERE** klauzúl považuje SQL pravdivostnú hodnotu *unknown* ekvivalentnú hodnote *false*.

Avšak za účelom vyhodnocovania integritných ohraničení sa *unknown* v SQL správa ako *true*, inými slovami, tuple, ktoré vzhľadom k integritným ohraničeniam sú vyhodnotené ako *unknown*, nie sú vylúčené z databázy. Teda ako by boli vyhodnotené na *true*, sú v nej ponechané.

Poznamenajme, že v štandarde pre objektové databázy skupiny ODMG 2.0¹⁵⁰ sa taktiež uvažuje o nulle typu „unknown“ (**un**). Pre každý literálový typ (sú to preddefinované typy, napr. **float** alebo **Set<>**) existuje ešte jeden literálový typ podporujúci nullovú hodnotu (značia sa potom s prefixom „nullable_“, napr. **nullable_float** alebo

¹⁴⁹ Inými slovami, v angličtine **NOT** v **NOT TRUE** nie je „**NOT**“ trojhodnotovej logiky a **NOT** trojhodnotovej logiky nie je „not“ obyčajnej angličtiny. V slovenčine by sa dal tento paradox vyjadriť asi nasledovne: „nepravdivý nie je nie-pravdivý“.

¹⁵⁰ porov. [Mrázik, 1998]

nullable_Set<>). Táto verzia typu s nullom je ten istý ako pôvodný typ, avšak je rozšírený o nullovú hodnotu „nil“. Sémantika nullu je rovnaká ako v štandarde SQL92. Pripomíname, že hoci sú mená nullov rôznych typov rovnaké, ide o rôzne nully.

7.4.2 Null v SQL Oracle7 Server

Oracle7 Server patrí medzi robustné databázové servery. Použitie nullov v Oracli úplne sleduje štandard ANSI/ISO SQL92. Existuje jediný null, ktorá indikuje „missing, unknown, or inapplicable data“ (Oracle7 Server, Release 7.3, *Concepts* resp. *SQL Reference*, 1996). Null nesmie byť použitý s cieľom mať význam nejakej inej hodnoty, napr. nuly. Default nastavenie pre každý stĺpec je NULL, čo značí, že môže obsahovať null. Stĺpec môže obsahovať nully, pokiaľ nad ním nebolo definované integritné ohraničenie NOT NULL resp. PRIMARY KEY, ktoré automaticky zahŕňa aj ohraničenie NOT NULL na stĺpci. V takom prípade nemožno do tabuľky pridať žiaden riadok obsahujúci v takýchto stĺpcoch null.

Keďže Oracle má vlastnú internú architektúru uchovávaní záznamov nie ako klasických rekordov, ale v podstate ako objektov (v zmysle OOP), prítomnosť nullu je indikovaná nulou v zodpovedajúcom poli riadka, ktoré znamená počet bytov na uchovanie danej hodnoty. Keď sú nully uvedené na konci záznamu, môže ich byť ľubovoľný počet, veľkosť, ktorú zaberajú, je 1 byte. Preto (z priestorových dôvodov) sa v Oracli odporúča stĺpce, kde sa očakávajú nullové hodnoty, uvádzať na konci tabuľky. Nully nie sú indexované (ani za účelom vytvorenia novej tabuľky).

Vyhodnocovanie dotazov je rovnaké ako v trojhodnotovej logike podľa Codda či ako v SQL štandarde. Ak chceme identifikovať nully, treba použiť predikát IS NULL. Keď chceme použiť nejakú hodnotu, pričom nevieme, či náhodou nie je nullová, poslúži nám funkcia NVL s dvoma argumentmi $expr_1$, $expr_2$, ktorá sa správa nasledovne: ak $expr_1$ je null, vráť $expr_2$, inak (teda ak $expr_1$ nie je null), vráť $expr_1$. Inými slovami, zabezpečí konverziu na regulárnu hodnotu v prípade, že prvý vstupný parameter je null.

Agregačné funkcie nully väčšinou ignorujú (teda súčet čísel 10, null, null, null, 20 bude 30). Selekcia ignoruje unknown odpovede, teda ide o *true*-selekciu.

Outer-join v Oracli sa správa ako štandardný outer join. Syntax má tú vlastnosť, že o aký druh outer-joinu ide, sa indikuje vo WHERE klauzule, a to tak, že sa dané stĺpce označia postfixom (+).

7.4.3 Null v Microsoft Access 97

Microsoft Access 97 patrí medzi kancelárske databázové produkty. V Microsoft Access 97 je v podstate implementovaný SQL štandard: existuje jediná vopred definovaná konštantná generická null hodnota Null indikujúca „missing“ alebo „unknown“ dáta v poli. V špeciálnych prípadoch môžeme implementovať **un** a **ne** nully, ako je uvedené nižšie.

Pri definovaní návrhu databázy je treba určiť, či daný stĺpec môže alebo nesmie obsahovať nullové hodnoty (napríklad tak, že Validation Rule sa nastaví na jednoduché Null<>"«Expr»" a naviac sa môže zakázať vkladanie reťazcov prázdnej dĺžky, prípadne sa prepínač Required prepne na Yes).

Null hodnotu môžeme použiť vo výrazoch a môže byť vložený do polí, pre ktoré je informácia neznáma, a tiež do výrazov a dotazov. (Vo Visual Basicu kľúčové slovo **Null** indikuje nulovú hodnotu.) Niektoré polia, napríklad tie, ktoré sú definované ako obsahujúce primárny kľúč, nemôžu obsahovať nulové hodnoty.

Keď pole neobsahuje žiadnu hodnotu, obsahuje vlastne hodnotu **Null**. V skutočnosti pre polia typu **Text**, **Memo** a **Hyperlink** môže v takomto prípade okrem **Null** obsahovať aj reťazec nulovej dĺžky. Rozdiel medzi nimi je popísaný nižšie. Nasledujú niektoré základné črty zaobchádzania s **Null** v MS Accessse.

Na testovanie, či hodnota je alebo nie je null, slúži funkcia **IsNull**. Pri joinovaní tabuliek v dotaze výsledok obsahuje len tie záznamy, ktoré nemajú **Null** v poliach, podľa ktorých sa tabuľky spájajú. Je možné používať **Nully** aj na dotazovanie či zisťovanie **Null** hodnôt. Pri použití s agregáčnymi funkciami (výpočet súčtu, priemeru, počtu alebo iného množstva) nad číselnými dátami, záznamy s nulovými hodnotami nebudú do výpočtu zahrnuté. Pri použití aritmetického operátora (ako sú +, -, *, /) vo výraze, pričom niektorý z operandov je **Null**, výsledok celého výrazu bude tiež **Null**. Môžeme vytvoriť výraz, ktorý konvertuje (v číselnej doméne) null hodnoty na matematickú nulu. Na konvertovanie **Null** na nulu slúži funkcia **Nz**. Pri triedení vo vzostupnom poradí budú záznamy obsahujúce v tomto poli **Null** zaradené na začiatok zoznamu.

Microsoft Access umožňuje rozlišovať medzi dvoma typmi nullu, hoci len pri poliach typu **Text**, **Memo** a **Hyperlink** práve na zdôraznenie, že informácia môže existovať ale jej hodnota nie je práve teraz známa alebo nie je aplikovateľná v danom zázname. Keď necháme pole voľné (nevyplnené), znamená to, že interne je v tomto poli uložený **Null** s významom **unknown (un)**. Vloženie reťazca nulovej dĺžky (napísanie dvoch úvodzoviek "") znamená null s významom **inapplicable (ne)**.

8 Záver

„Avoid nulls.“
Chris Date¹⁵¹

Cieľom tejto práce bolo poskytnúť ucelený pohľad na problematiku neúplnej informácie (a špeciálne nullových hodnôt) v relačných databázach. Aj keď výskum v teoretickej oblasti napreduje hlavne smerom k oblastiam, ktoré v práci neboli uvažované, predsa sme pokladali za dôležité zhrnúť doterajší vývoj v teórii práve v spomínaných smeroch. Zamerali sme sa najmä na teóriu nullových hodnôt a neúplnej informácie, avšak letmú pozornosť sme venovali aj praktickým dopadom neúplnej informácie na proces vývoja softvéru, pričom sme sa snažili „neuletieť príliš vysoko“ v teórii a „nezájsť implementačne hlboko“ v praktickejšie orientovaných častiach. Z tohoto dôvodu môže práca slúžiť aj ako materiál ku kurzu o neúplnej informácii ako doplnkovému predmetu k základnej prednáške „Databázové systémy“. Lahko sme tiež nahliadli do štandardu SQL, ktorý sa v súčasnosti v praxi implementuje v každom „slušnom“ databázovom prostredí. Napokon, prezentovali sme ideu dekomponovania relácie s tromi základnými typmi nullov, ktorou je ilustrovaná myšlienka zbavenia sa nullov resp. spôsobu, ako sa bez nich zaobísť.

Ako každá diplomová práca, aj práca predkladaná je neúplná. V prípade tejto práce je o dôvod na neúplnosť viac, ako to aj vyplýva z jej témy. Môže byť azda práca o neúplnej informácii úplná? Pre tých, ktorí majú radi neúplnú informáciu, je práca aspoň prehľadom jednotlivých prístupov, a pre ostatných, ktorí neúplnú informáciu radi nemajú, medzi ktorých patrí napr. Chris Date (ktorý poznamenáva, aby sme sa vyhýbali nullom), poskytuje objektívne dôvody, prečo by mali vo svojom postoji zotrvať. Ako sme skonštatovali už v úvode, definitívne riešenie problému neúplnej informácie zatiaľ neexistuje a teoretické výskumy pokračujú ďalej. Dúfajme, že po čase budeme vedieť viac.

¹⁵¹ [Date–Darwen, 1993], str. 243. Ide o celý text kapitoly 16.6, ktorá má názov „A RECOMMENDATION“, čo je nielen kuriózne, ale aj veľavravné.

9 Literatúra a použité pramene

Poznámka: Nie všetky tu menované pramene sú výslovne citované v texte práce.

- [Abiteboul–Grahne, 1985] Serge Abiteboul, Gösta Grahne: Update semantics for incomplete databases. *Proceedings of 11th International Conference on Very Large Data Bases*, Stockholm, Sweden, IEEE 1985, 1-12.
- [Abiteboul–Kanellakis–Grahne, 1991] Serge Abiteboul, Paris Kanellakis, Gösta Grahne: On the representation and querying of sets of possible worlds. *Theoretical Computer Science* 78 (1991), 159-187.
- [ANSI, 1975] ANSI/X3/SPARC, Study group on data base management systems: *Interim Report*, Feb. 1975. Tiež v *FDT (Bulletin of ACM SIGMOD)* Vol. 2, No. 2 (1975).
- [Barbará et al., 1992] Daniel Barbará, Hector Garcia-Molina, Daryl Porter: The Management of probabilistic data. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 5 (Oct. 1992), 487-502.
- [Biskup, 1981] Joachim Biskup: A formal approach to null values in database relations. In: *Advances in Data Base Theory*, H. Gallaire, J. Minker, J. M. Nicolas (eds.), Plenum Press, New York 1981, 299-341.
- [Biskup, 1982] Joachim Biskup: Extending the relational algebra for relations with maybe tuples and existential and universal null values. Jan. 1982* *Fundamenta Informaticae*.
- [Biskup, 1983] Joachim Biskup: Foundations of Codd's relational maybe operations. *ACM Transactions on Database Systems* Vol. 8, No. 4 (Dec. 1983), 608-636.
- [Borgida–Mylopoulos, 1981] Alex Borgida, John Mylopoulos: Semantic models in databases: Some formal aspects. *Department of Computer Science, University of Toronto*. Nedatované (po 1981).
- [Codd, 1979] E. F. Codd: Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems* Vol. 4, No. 4 (Dec. 1979), 397-434.
- [Date, 1995] Chris J. Date: *An introduction to Database systems. Sixth edition*. Addison-Wesley Publishing Company, 1995, kap. 20: Missing Information, 570-592.
- [Date–Darwen, 1993] Chris J. Date, Hugh Darwen: *A guide to the SQL Standard*. Addison-Wesley Publishing Company, 1993, kap. 16: Missing Information and Nulls, 219-243.
- [DB2v4.1, 1996] *DB2 Version 4.1: Join Enhancements*. Chuck Anesi 1996.
<http://www.anesi.com/v41003.htm>.
- [DeMichiel, 1989] Linda DeMichiel: Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 4 (Dec. 1989), 485-493.
- [DeMichiel, 1990] Linda DeMichiel: Resolving database incompatibility: An approach to performing relational operations over mismatched domains. In: *A mediator architecture for abstract data access*, Wiederhold et al., *Stanford University Report* No. STAN-CS-90-1303 (Feb. 1990), 80-103.
- [Eco, 1997] Umberto Eco: *Jak napsat diplomovou práci*. Votobia, Olomouc 1997.

* Článok je staršou verziou publikovaného materiálu (resp. verzia „submitted for publication“).

- [Goldstein, 1981] Billie S. Goldstein: Constraints on null values in relational databases. *Proceedings of 7th International Conference on Very Large Data Bases*, Cannes, France, IEEE 1981, 101-110.
- [Golshani, 1985] Forouzan Golshani: Growing certainty with null values. *Information Systems Vol. 10*, No. 3 (1985), 289-297.
- [Grant, 1977] John Grant: Null values in a relational data base. *Information Processing Letters Vol. 6*, No. 5 (Oct. 1977), 156-157.
- [Grant, 1979] John Grant: Partial values in a tabular database model. *Information Processing Letters Vol. 9*, No. 2 (Aug. 1979), 97-99.
- [Grant, 1980] John Grant: Incomplete information in a relational database. *Fundamenta informaticae III*, No. 3 (1980), 363-378.
- [Grant–Minker, 1986] John Grant, Jack Minker: Answering queries in indefinite databases and the null value problem. *Advances in Computing Research, Vol. 3* (1986), 247-267.
- [Homenda, 1991] Waldysław Homenda: Databases with alternative information. *IEEE Transactions on Knowledge and Data Engineering, Vol. 3*, No. 3 (Sep. 1991), 384-386.
- [Hopcroft–Ullman, 1978] John E. Hopcroft, Jeffrey D. Ullman: *Formálne jazyky a automaty*. Alfa, Bratislava 1978.
- [Chen–Chiu–Tseng, 1996] Arbee L. P. Chen, Jui-Shang Chiu, Frank S. C. Tseng: Evaluating aggregate operations over imprecise data. *IEEE Transactions on Knowledge and Data Engineering, Vol. 8*, No. 2 (Apr. 1996), 273-284.
- [Chiu–Chen, 1995] Jui-Shang Chiu, Arbee L. P. Chen: An exploration among exclusive disjunctive data. *IEEE Transactions on Knowledge and Data Engineering, Vol. 7*, No. 6 (Dec. 1995), 928-940.
- [Chiu–Chen, 1996] Jui-Shang Chiu, Arbee L. P. Chen: A note on „Incomplete relational database models based on intervals“. *IEEE Transactions on Knowledge and Data Engineering, Vol. 8*, No. 1 (Feb. 1996), 189-191.
- [Imieliński–Lipski, 1981] Tomasz Imieliński, Witold Lipski, Jr.: On representing incomplete information in a relational database. *Proceedings of 7th International Conference on Very Large Data Bases*, Cannes, France, IEEE 1981, 388-397.
- [Imieliński–Lipski, 1984] Tomasz Imieliński, Witold Lipski, Jr.: Incomplete information in relational databases. *Journal of the Association for Computing Machinery Vol. 31*, No. 4 (Oct. 1984), 761-791.
- [Jaegermann, 1975] Michał Jaegermann: Information storage and retrieval systems – mathematical foundations. Part IV. Systems with incomplete information. *CC PAS Report No. 215*, Warsaw, Poland 1975, 1-78.
- [Jaegermann–Lipski, 1983] Michał Jaegermann, Witold Lipski, Jr.: Numerical queries in incomplete information data bases. *Fundamenta Informaticae VI*, No. 3-4 (1983).
- [Jajodia–Springsteel, 1990] Sushil Jajodia, Frederick N. Springsteel: Lossless outer join with incomplete information. *BIT 30* (1990), 34-41.
- [Katuščák, 1998] Dušan Katuščák: *Ako písať vysokoškolské a kvalifikačné práce*. Stimul, Bratislava 1998.
- [Kent, 1979] William Kent: Limitations of record based information models. *ACM Transactions on Database Systems Vol. 4*, No. 1 (Mar. 1979).
- [Kolibiar, 1991] Milan Kolibiar a kol.: *Algebra a príbuzné disciplíny*. Alfa, Bratislava 1991.
- [Kowalski, 1978] Robert Kowalski: Logic for data description. In: *Logic and Data Bases*, H. Gallaire, J. Minker (eds.), Plenum Press, New York 1978, 77-103.

- [Levene-Loizou, 1993] Mark Levene, George Loizou: Semantics for null extended nested relations. *ACM Transactions on Database Systems Vol. 18*, No. 3 (Sep. 1993), 414-459.
- [Libkin, 1994] Leonid Libkin: *Aspects of partial information in databases*. PhD. Thesis. University of Pennsylvania 1994, 262 str.
- [Lipski, 1977] Witold Lipski, Jr.: On the logic of incomplete information. *Proceedings of 6th International Symposium on Mathematical Foundations of Computer Science*, Tatranská Lomnica, Czechoslovakia, T. Gruska (ed.), Springer-Verlag, Berlin 1977, 374-381.
- [Lipski, 1979] Witold Lipski, Jr.: On semantic issues connected with incomplete information Databases. *ACM Transactions on Database Systems Vol. 4*, No. 3 (Sep. 1979), 262-296.
- [Lipski, 1981] Witold Lipski, Jr.: On databases with incomplete information. *Journal of the Association for Computing Machinery Vol. 28*, No. 1 (Jan. 1981), 41-70.
- [Lipski, 1983a] Witold Lipski, Jr.: Logical problems related to incomplete information in databases. *Rapport de Recherche No. 138, Universite de Paris-Sud* (Sep. 1983).
- [Lipski, 1983b] Witold Lipski, Jr.: On relational algebra with marked nulls. *Rapport de Recherche No. 140, Universite de Paris-Sud* (Nov. 1983).
- [Liu-Sunderraman, 1989] Ken-Chih Liu, Rajshekhar Sunderraman: General indefinite maybe information in relational databases. *Information Processing 89*, G. X. Ritter (ed.), Elsevier Science Publishers B.V. (North-Holland), IFIP 1989, 809-814.
- [Liu-Sunderraman, 1990] Ken-Chih Liu, Rajshekhar Sunderraman: Indefinite and maybe information in relational databases. *ACM Transactions on Database Systems Vol. 15*, No. 1 (Mar. 1990), 1-39.
- [Liu-Sunderraman, 1991] Ken-Chih Liu, Rajshekhar Sunderraman: A generalized relational model for indefinite and maybe information. *IEEE Transactions on Knowledge and Data Engineering, Vol. 3*, No. 1 (Mar. 1991), 65-76.
- [MacLane-Birkhoff, 1973] Saunders MacLane, Garrett Birkhoff: *Algebra*. Alfa, Bratislava 1973.
- [Manna, 1981] Zohar Manna: *Matematická teorie programů*. SNTL, Praha 1981.
- [Marcinčák-Matula-Semanišin, 1999] Peter Marcinčák, Marcel Matula, Gabriel Semanišin: Detekcia „podobných“ záznamov v databázových systémoch. *Zborník UNINFOS'99*, Bratislava 1999, 85-90.
- [Mrázik, 1998] Augustín Mrázik: *Objektovo-orientované databázové systémy*. Prednášky na MFF UK, zimný semester 1998/99. Tiež <http://object.dcs.fmph.uniba.sk>, <http://www.omg.org>, <http://www.odmg.org>.
- [Nakano, 1990] Ryohei Nakano: Translation with optimization from relational calculus to relational algebra. *ACM Transactions on Database Systems Vol. 15*, No. 4 (Dec. 1990), 544-545.
- [Nicolas-Gallaire, 1981] Jean-Marie Nicolas, Hervé Gallaire: Data base: Theory vs. Interpretation. In: *Logic and Data Bases*, J. M. Nicolas, H. Gallaire (eds.), Plenum Press, New York 1981.
- [Ola-Özsoyoglu, 1993] Adegbemiga Ola, Gultekin Özsoyoglu: Incomplete relational database models based on intervals. *IEEE Transactions on Knowledge and Data Engineering, Vol. 5*, No. 2 (Apr. 1993), 293-308.
- [Paredaens, 1989] Jan Paredaens et al.: *The Structure of the Relational Database Model*. Springer-Verlag 1989, kap.6: Incomplete Information, 156-176.
- [Ragan, 1998] Ján Ragan: *Anglicko-slovenský slovník výpočtovej techniky*. SPN, Bratislava 1998.

- [Reiter, 1978] Raymond Reiter: On closed world data bases. In: *Logic and Data Bases*, H. Gallaire, J. Minker (eds.), Plenum Press, New York 1978, 55-76.
- [Reiter, 1986] Raymond Reiter: A sound and sometimes complete query evaluation algorithm for relational databases with null values. *Journal of the Association for Computing Machinery Vol. 33*, No. 2 (Apr. 1986), 349-370.
- [Rishe, 1988] Naphtali Rishe: *Database design fundamentals*. Prentice-Hall, London 1988.
- [Scheber, 1988] Anton Scheber: *Databázové systémy*. Alfa, Bratislava 1988.
- [Sugihara] <http://www.ics.hawaii.edu/~sugihara/course/ics421s95/note/1-23n03>,
<http://www.ics.hawaii.edu/~sugihara/course/ics421s95/note/2-06n07>.
- [Swamy] Shekhar Swamy: *Handling Incomplete Information In Query Processing*.
<http://www.cs.rpi.edu/~swamys/adb/query/query.html>.
- [Šešera–Mičovský, 1994] Ľubor Šešera, Aleš Mičovský: *Objektovo-orientovaná tvorba systémov a jazyk C++*. Perfekt, Bratislava 1994.
- [Štuller, 1997] Július Štuller: Incomplete information in database systems. COFAX, jún 1997.
- [Takahashi, 1993] Yoshikane Takahashi: Fuzzy database query language and their relational completeness theorem. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 1 (Feb. 1993), 122-125.
- [Tsichritzis–Lochovsky, 1987] Dionysios C. Tsichritzis, Frederick H. Lochovsky: *Databázové systémy*. SNTL, Praha 1987.
- [Ullman, 1988] Jeffrey D. Ullman: *Principles of database and knowledge-base systems Volume 1*. Computer Science Press, Rockville 1988.
- [Vardi, 1986] Moshe Y. Vardi: Querying logical databases. *Journal of Computer and System Sciences Vol. 33*, No. 2 (Oct. 1986), 142-160.
- [Vassiliou, 1980] Yannis Vassiliou: Functional dependencies and incomplete information. *Proceedings of 6th International Conference on Very Large Data Bases*, Montreal, Canada, IEEE 1980, 260-269.
- [Zaniolo, 1984] Carlo Zaniolo: Database systems with null values. *Journal of Computer and System Sciences Vol. 28*, No. 1 (Feb. 1984), 142-166.
- [Zaniolo] Carlo Zaniolo: Incomplete information and null values: an Overview. Nedatované (po 1980).
- [Zendulka, 1996] Jaroslav Zendulka: Modelling of missing (unknown) information in database and logic simulation languages. *Advanced Simulation of Systems '96* (1996), 7-12.