Ekvivalence, symetrie, antisymetrie, tranzitivita v relačním modelu

Helena Palovská
Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky
nám. W. Churchilla 4, 130 67 Praha 3
e-mail: palovska@vse.cz

Abstrakt: *Efektivní způsoby implementace symetrických, antisymetrických, tranzitivních binárních vztahů a vztahů ekvivalence jsou nabídnuty.*

Klíčová slova: symetrie, antisymetrie, tranzitivita, ekvivalence, relační databáze

Abstract: *Effective ways to implement symmetric, antisymmetric, transitive binary relationships and equivalence relationships are offered.*

Keywords: symmetry, antisymmetry, transitivity, equivalence, relational database

1. Úvod

Symetrii, antisymetrii či tranzitivitu binárních vztahů, a speciálně ekvivalenci, je obtížné implementovat v relačním databázovém modelu. Přitom se jedná o jevy v realitě se vyskytující, i když ne časté. Například pokud dva spolu soupeří, je obtížné jednomu z nich přiřadit roli “A” a druhému roli “B”, když přitom reálně jsou jejich role stejné. Dalším příkladem je vztah sousedních objektů; těžko lze stanovit jednomu roli “A” a druhému roli “B”. Nebo, pokud jsou dva různé objekty ve vztahu podřízenosti, pak obvykle nesmí zároveň nastat stav, ve kterém mají role vyměněné. Další příkladem je vztah dosažitelnosti; je-li bod B dosažitelný z bodu A, a je-li bod C dosažitelný z bodu B, pak je bod C dosažitelný i z bodu A. Jiným příkladem je situace, kdy se do databáze dostávají rovnocenné záznamy, například importem z cizího zdroje; pak je třeba efektivně rozeznávat, které záznamy jsou ekvivalentní.

V relačním databázovém modelu je možné podobné požadavky zajistit, leč nebývá zřejmé, jak to udělat. Tento článek nabízí efektivní způsoby implementace tohoto druhu požadavků. Eventuální příklady kódů uvádí v jazyce Oracle.

Práce vychází z vlastní více než dvacetileté zkušenosti autorky z řešení reálných projektů a cílí na odbornou veřejnost i na studenty databázových předmětů na vysokých školách, kteří se s problémy implementace symetrie, antisymetrie, tranzitivity a ekvivalence v relačních databázích potýkají.

V takto souhrnné podobě není tato problematika řešena, i když některé dílčí výsledky lze nalézt ve volných zdrojích databázových komunit.

2. Symetrie

Nejprve je třeba objasnit, co symetrie v případě binárního vztahu je, a kdy se jedná o skutečnou symetrii, a kdy jde jen o pohled na věc.

Vztah R je symetrický, když jsou role v něm nerozlišitelné, což formálně lze vyjádřit:

$$\forall x,y : R(x,y) \Leftrightarrow R(y,x) \quad (1)$$

V úvodu uvedené příklady vztahů *soupeření*, *sousedství*, nebo např. *spolupráce*, jsou příklady symetrií. Ovšem pokud se jedná o soupeře v nějakém zápasu, tak často bývají role soupeřů označovány jako “domácí” a “hosté”, “vyzyvatel” a “obhájce”, a pod. Pak se sice z praktického pohledu na výsledky zápasů v širším kontextu nejedná o podstatný rozdíl v těch rolích, ale z jiného pohledu – organizace zápasů, jsou ty role odlišné.

Pokud má relační databáze zajistit splnění podmínky (1) pro vztah R, je třeba nějakého integritního omezení. K tomu se nabízejí dva přístupy k řešení.

2.1 Řešení č.1

Pokud bude *r* jméno tabulky zaznamenávají daný vztah, a *a* resp. *b* budou sloupce (stejného typu) pro obě role v tomto vztahu, pak pro symetrii stačí zaznamenat daný vztah pouze jednou, například do sloupce *a* nižší hodnotu a do sloupce *b* vyšší:

a	***b***
Česká Republika	Německo
Česká Republika	Polsko
Německo	Polsko

Příslušné integritní omezení:

```
ALTER TABLE r ADD UNIQUE (a,b);  /*pokud již dvojice (a,b) není primárním klíčem*/
ALTER TABLE r ADD CHECK (a<b);
```

Pro snazší vkládání dat můžeme definovat trigger, který hodnoty přehodí, pokud je třeba:

```
CREATE OR REPLACE TRIGGER r_symetrie_a_b BEFORE INSERT OR UPDATE
ON r
DECLARE c r.a%TYPE;
BEGIN
        IF :new.b<:new.a THEN
                c: = :new.a;
                :new.a := :new.b;
                :new.b := :c;
        END IF;
END;
```

Pro ověřování, zda x a y jsou ve vztahu R, lze například definovat funkci:

```
CREATE FUNCTION jsou_r (x IN r.a%TYPE , y IN r.a%TYPE) RETURN NUMBER
IS
BEGIN
        IF x<y THEN
                RETURN (SELECT count(*) FROM r WHERE r.a=x and r.b=y);
        ELSE
                RETURN (SELECT count(*) FROM r WHERE r.a=y and r.b=x);
        END IF;
END;
```

Souhrnné dotazy týkající se vztahu R je nejlépe formulovat nadvakrát – například počet sousedů objektu x spočítáme:

```
SELECT SUM(poc)
FROM
        (SELECT COUNT(*) poc FROM r WHERE a=x )
        /*využije se index na sloupec a*/
        UNION
        (SELECT COUNT(*) FROM r WHERE b=x)
        /*využije se index na sloupec b*/
;
```

3. Řešení č.2

Pro implementaci symetrie lze zvolit i jiný typ řešení. Ten vychází z toho, že dvojice objektů v symetrii tvoří pár, tedy množinu se dvěma prvky. Vytvoříme tedy reprezentaci onoho páru nějakým identifikátorem. Pak modelujeme vztah každého z členů páru k tomuto páru – vztah členství – což je obecně vztah kardinality n:m – jeden objekt může být v různých párech s různými objekty. Vytvoříme tabulku vztahu *členství v párech*, například:

stát	**id páru**
Česká republika	1
Německo	1
Česká republika	2
Polsko	2
Německo	3
Polsko	3

Tento typ řešení bude nejspíš vyžadovat zaručení, že pár má přesně dva prvky. Vzhledem k postupnému zapisování záznamů do databáze nelze na její úrovni zaručit, že pro pár máme přesně dva záznamy o členství jeho prvků, ale můžeme definovat omezení pro to, aby počet prvků páru nebyl větší. Nejefektivnější je použít umělý pomocný sloupec pro pseudo-role v páru, například:

```
ALTER TABLE páry ADD COLUMN role CHAR(1) CHECK role IN (a,b) NOT NULL;
ALTER TABLE páry ADD UNIQUE (id_paru,role);
```

V tomto řešení se souhrny počítají lépe, například počet sousedů objektu x:

```
SELECT COUNT(*) FROM páry where stát = x;
```

4. Antisymetrie

Požadavek na antisymetrii je velmi vzácný, nicméně se vyskytuje. Jedná se o to, že pokud jsou dva objekty v daném vztahu, pak již nesmí být v onom vztahu s vyměněnými rolemi.

Formálně:
$\forall x,y : R(x,y) \Rightarrow \neg R(y,x)$

Například pokud osoba A je ručitelem osoby B, nemůže pak být osoba B ručitelem osoby A.

Pro řešení takového požadavku lze použít analogii řešení č.1 pro symetrii, pouze je potřeba vytvořit odvozený sloupec či alespoň index na výraz jej odvozující; oním výrazem bude dvojice obou členů uspořádaná dle velikosti. Definice indexu:

```
CREATE UNIQUE INDEX antisym_r ON r
        (CASE a<b THEN a ELSE b , CASE a<b THEN b ELSE a);
```

5. Tranzitivita

Tranzitivita, přenositelnost, binárního vztahu znamená, že pokud jsou dva objekty v daném vztahu, a druhý z nich je v tom vztahu s nějakým třetím objektem, pak i první z objektů je s tím třetím objektem v daném vztahu. Formálně:

$\forall x,y,z : R(x,y) \,\&\, R(y,z) \Rightarrow R(x,z)$

V kombinaci s antisymetrií se jedná o různá *částečná uspořádání*, například slovníkových hesel či klíčových slov a jejich vztahu nadřazenosti, nebo obecných vztahů celek-část.

V kombinaci se symetrií se jedná o *ekvivalenci*, viz kap. 6.

V obecném případě může jít například o *propojitelnost* z jednoho bodu v síti do druhého bodu – může existovat cesta z bodu A do bodu B, přičemž cesta z bodu B do bodu A může ale také nemusí existovat – v silniční síti jsou například úseky jednosměrné i úseky obousměrné. Jako další příklad může sloužit síť veřejné dopravy.

U částečného uspořádání či propojitelnosti je otázkou, zda v databázi ukládat pouze základní neodvoditelnou informaci, například pouze vztah bezprostředně nadřízeného a podřízeného, či všechny odvozené, či jen některé odvozené, například do určitého počtu kroků odvození. Například u sítě veřejné dopravy můžeme ukládat pouze přímé spoje, nebo spoje do pěti přesedání…, nebo dokonce ukládat pouze přímá spojení bez mezizastávek, a vše ostatní odvozovat. Zjevně se zde jedná o optimalizační úlohu, jejíž řešení závisí na požadované funkčnosti příslušného informačního systému.

5.1 Práce s odvozenými záznamy

V případě, že databáze uchovává záznamy odvozené ze základních instancí vztahu R, je potřeba efektivně řešit situace, kdy dojde k rušení některé základní instance (např. dočasná či trvalá neprůchodnost úseku spojení).

Pro takový případ je užitečné umět rozlišit, které záznamy jsou “odvozené”, a ze kterých základních instancí. Prvním krokem je tedy indikátor “základnosti”, tedy například sloupec *zakladni* s hodnotami 0 nebo 1. Dále vztah *je_odvozen_z* mezi odvozenými a základními záznamy. Tento vztah je obecně kardinality n:m, tudíž

v relačním databázovém modelu vyžaduje tabulku. Taková tabulka, obecně s kvadraticky vyšším počtem záznamů než má tabulka *r* pro vztah R[1], by případně sloužila pouze pro efektivní nalezení dotčených odvozených záznamů z dočasně či trvale rušeného základního záznamu. Samozřejmě je ji možno využít pro zapsání dalších informací užitečných při dotazování databáze, například pro informaci o pořadí toho kterého základního úseku spojení v tom kterém spojení odvozeném.

Poslední zmíněná eventuální potřebná informace může být efektivněji zapsána v dalším sloupci tabulky *r*, jenž by u ne-základních záznamů nesl v semi-strukturované podobě informaci o celé propojené "trase" (ve formátu JSON, XML...).

Ještě je třeba se zabývat případem, kdy do systému vstupuje nový spoj dvou již v síti existujících bodů. Pokud databáze označuje ty z odvozených "tras", jež jsou nějakým způsobem optimální, pak je třeba posoudit, zda daný nový spoj není lepší, než eventuální stávající trasa označená jako optimální. Pokud ano, pak se změny mohou dotknout dalších tras doposud označených jako optimální. Na místě je mít efektivní metodu jejich nalezení. K tomu může sloužit modifikace tabulky pro vztah *je_odvozen_z* zmíněné výše, a to její rozšíření na všechny odvozené záznamy z tabulky *r* a vztah vyjadřující, že trasa představená daným záznamem z *r obsahuje* trasu představenou druhým záznamem z *r*. Mohutnost takovéto pomocné tabulky by se opět zvýšila[2].

5.2 Hledání základních záznamů

Pokud se mají spolehlivě detekovat základní, neodvoditelné záznamy, pak je opět třeba řešit případ přidávání nových spojů. Je třeba zjistit, zda potenciální nový záznam není odvoditelný ze stávajících. Bez pomocných odvozených datových struktur, jako například těch zmíněných v části 5.1, je třeba hledat cestu z výchozího do koncového prvku nového spoje ve stávajících záznamech[3]. Ostatně taková může být jedna z potřebných funkcí aplikace, která databázi využívá.

6. Ekvivalence

Pokud binární vztah R vyjadřuje ekvivalenci, je nutně symetrický a tranzitivní. Proto lze celou doménu porovnávaných objektů rozdělit do disjunktních podmnožin vzájemně ekvivalentních objektů, tzv. *tříd ekvivalence*[4].

Efektivním řešením pro zaznamenání ekvivalence je proto umělý atribut porovnávaných objektů, který jim přiřadí identifikátor třídy ekvivalence, do které patří. Pro identifikaci tříd ekvivalence poslouží jakákoliv uměle konstruovaná hodnota. Pokud je výskyt ekvivalentních vzájemně různých objektů vzácný, lze příslušný sloupec ponechávat prázdný kromě případů zjištěné výjimky. Například tabulka:

[1]To je v případě uložení všech odvozených záznamů. Jinak, pokud například ukládáme jen do n "přestupů", bude mohutnost maximálně n-krát větší.

[2]Pokud například ukládáme jen do n "přestupů", pak by byla až 2^n -krát větší.

[3]Jedná se o tzv. path-finding algoritmus, viz např. (Cormen et. all, 2009, kap. VI)

[4]Každý prvek domény považujeme za ekvivalentní se sebou samým.

ID	termín	ekvivalence
1	okurka	-
2	ječmen	1
3	jablko	-
4	řepa	-
5	zelí	2
6	jecmen	1
7	kel	2
8	slunečnice	-
9	zeli	2

zaznamenává, že termíny “ječmen” a “jecmen” jsou ekvivalentní, a “zelí”, “kel” a “zeli” jsou ekvivalentní.

Také lze pro identifikátor třídy ekvivalence zvolit identifikátor kteréhokoli z jejích prvků, například toho, který byl v databázi zapsán jako první − stal by se jakýmsi jejím reprezentantem. Pak by příslušný sloupec v tabulce mohl být cizím klíčem. Tato volba by však později mohla vést k problémům, pokud by záznam s reprezentantem třídy měl být někdy v budoucnu smazán:

ID	termín	ekvivalence
1	okurka	-
2	ječmen	2
3	jablko	-
4	řepa	-
5	zelí	5
6	jecmen	2
7	kel	5
8	slunečnice	-
9	zeli	5

Reference

Cormen,T.H. & Leiserson,Ch.E. & Rivest,R. L. & Stein,C., 2009: Introduction to Algorithms, 3rd ed., The MIT Press, Dostupné z: https://labs.xjtudlc.com/labs/wldmt/reading%20list/books/Algorithms%20and%20optimization/Introduction%20to%20Algorithms.pdf

JEL Classification: M10, C88