
Jistě jste si minule všimli, že problematika zahnízděných dotazů není žádná legrace – věnujeme se jí proto i v dalším pokračování našeho seriálu.

Množiny a práce s více tabulkami

Chceme-li u každého holiče mít poznámku, zda je nadprůměrně chytrý nebo nadprůměrně chytrý holič, píšeme:

SELECT RC, JMENO, IQ>(SELECT AVG(IQ) FROM CLOVEK) CHYTRY, IQ>(SELECT AVG(IQ) FROM CLOVEK WHERE PROFESE=”HOLIC”) CHYTRY_HOLIC FROM CLOVEK WHERE PROFESE=”HOLIC” ORDER BY JMENO,RC;

Zajímají-li nás okresy, kde je průměrný plat učitele větší než průměrný plat květinářky v Hradci Králové, stačí psát:

SELECT OKRES FROM CLOVEK WHERE PROFESE=”UCITEL” GROUP BY OKRES HAVING AVG(PLAT)>(SELECT AVG(PLAT) FROM CLOVEK WHERE PROFESE=”KVETINARKA” AND OKRES=”HK”) ORDER BY OKRES;

Trochu více množin

Z předchozích dílů víme, že jedna hodnota může být porovnávána s celou množinou hodnot s použitím operátoru IN nebo NOT IN. První operátor odpovídá na otázku, zda hodnota je prvkem množiny hodnot. NOT IN naopak testuje nepřítomnost hodnoty v množině. Pak je logický výraz

ROK IN (1415,1620,1621,1848,1948, 1968)

roven YES, pokud jde o smutný letopočet. Podobně výraz

KRESTNI NOT IN (”LUBOMIR”,”VASIL”,”ALOIS”)

respektuje přání rodičky před porodem.

Pro usnadnění práce s množinami slouží další operátory, které konvertují množinu na zástupnou hodnotu vhodnou pro porovnání s jinou hodnotou. Nejprve probereme operátor **ANY**, který se spokojí s existencí třeba jen jediného prvku množiny splňujícího podmínku před ANY. Ve stejném významu můžeme použít i operátor SOME. Složené operátory =ANY, >ANY, <ANY, >=ANY, <=ANY, <>ANY, LIKE ANY znamenají totéž co =SOME, >SOME, <SOME, >=SOME, <=SOME, <>SOME, LIKE SOME. Například operátor >ANY je vhodný k položení otázky, zda existuje takový prvek množiny, jehož hodnota je menší než hodnota výrazu před operátorem. Pak pravdivost následujícího výrazu ještě nezaručuje zbohatnutí:

PRACHY >ANY(5000, 321000, 500)

Znamená to pouze testování, zda máme více než pětikilo, což v praxi neznamená tolik. Rovněž chudoba není dána pravdivostí výrazu:

PRACHY <ANY(5000000, 2000, 500)

Všichni, kdo mají méně než pět melounů, si nepřipadají jako žebráci. Operátor =ANY je rovnocenný obyčejnému IN. Použití LIKE ANY je silným nástrojem pro obecné vyhledávání v textech. Máme i silnější operátor **ALL**, který trvá na současném splnění podmínky před ALL pro všechny prvky množiny uvedené za ALL. Složené operátory =ALL, >ALL, <ALL, >=ALL, <=ALL, <>ALL, LIKE ALL jsou možnými konstrukcemi. Tak operátor >ALL je vhodný k položení otázky, zda všechny prvky množiny mají menší hodnotu, než je hodnota výrazu před operátorem. Pak pravdivost výrazu již vede k optimismu:

PRACHY >ALL(50, 73254710, 800)

Pevně věřím, že čtenáři článku připadá, že hodnota následujícího výrazu v jeho případě není ani NULL, ani YES:

IQ <ALL(147, 110, 160, 105, 45)

Operátor <>ALL je rovnocenný obyčejnému NOT IN. Pokud je množina definována výčtem hodnot, je to pro aplikace málo. Naštěstí si snadno můžeme vytvořit množinu zahnízděným dotazem vhodného typu.

Jednosloupcová tabulka

Obsahuje-li tabulka právě jeden sloupec, může být použita místo množiny hodnot. Hodnoty v jednotlivých řádcích tabulky představují prvky množiny, takže jde přímo o ráj pro operátory IN, NOT IN, ALL, SOME a ANY. Pokud tabulka vytvořená poddotazem obsahuje pouze jeden řádek, je také ještě množinou o jednom prvku. Opakují-li se v tabulce stejné hodnoty, práci s množinami to nevadí, jenom SQL server se více nadře a my se déle načekáme na odpověď. Pokud při popisu projekce, která jednosloupcovou tabulku vytváří, použijeme před výrazem slovo **DISTINCT**, zamezíme tím vzniku duplicitních hodnot. Tak můžeme výrazně zmenšit počet prvků v množině a odpověď na poddotaz i na dotaz bude velmi rychlá.

Pokud v tabulce neexistuje ani jeden řádek, jde o prázdnou množinu, která je povolena. Operace s prázdnými množinami typicky vracejí hodnotu NO – s výjimkou NOT IN, který vrací YES. Udělejme další krok směrem k propasti zvané SQL a zeptejme se jinak na seznam nejmladších králíků. Nejsou to náhodou ti, jejichž věk je menší nebo roven věku jakéhokoli králíka?

SELECT ID, JMENO FROM SAMEC WHERE VEK<=ALL(SELECT DISTINCT VEK FROM SAMEC) ORDER BY JMENO, ID;

Pak už je pouhá hračka zeptat se na seznam nejvypasenějších nejvýše dvouletých králíků:

SELECT ID, JMENO, VEK FROM SAMEC WHERE VEK<=2 AND HMOTNOST **>=**ALL(SELECT DISTINCT HMOTNOST FROM SAMEC WHERE VEK<=2) ORDER BY VEK, JMENO, ID;

Někteří vaši kamarádi mají možná stejná příjmení jako slavné osobnosti, přestože sami slavní nejsou:

SELECT RC, PRIJMENI, JMENO FROM CLOVEK WHERE PRIJMENI IN (SELECT DISTINCT PRIJMENI FROM CLOVEK WHERE SLAVNY) AND NOT SLAVNY ORDER BY PRIJMENI, JMENO, RC;

Kdo nemá rád IN, ptá se jinak:

SELECT RC, PRIJMENI, JMENO FROM CLOVEK WHERE PRIJMENI =ANY(SELECT DISTINCT PRIJMENI FROM CLOVEK WHERE SLAVNY) AND NOT SLAVNY ORDER BY PRIJMENI, JMENO, RC;

Jiní vaši kamarádi mají stejné křestní jméno jako vaši nepřátelé. Proto na mejdan pozvete jenom ty, kteří vám nebudou připomínat nevyřízené účty:

SELECT RC, PRIJMENI, JMENO FROM CLOVEK WHERE JMENO NOT IN (SELECT DISTINCT JMENO FROM CLOVEK WHERE NEPRITEL) AND KAMARAD ORDER BY PRIJMENI, JMENO, RC;

Kdo nemá rád NOT IN, může postupovat i jinak:

SELECT RC, PRIJMENI, JMENO FROM CLOVEK WHERE JMENO **<>**ALL(SELECT DISTINCT JMENO FROM CLOVEK WHERE NEPRITEL) AND KAMARAD ORDER BY PRIJMENI, JMENO, RC;

Množina může být ku prospěchu nejen za WHERE, ale i při popisu projekce nebo za HAVING. Chceme-li u lidí mít poznámku, zda jsou podezřelí, stačí vědět, že podezřelý je každý zločinec, každý se zločineckým příjmením nebo obyvatel města, kde bydlí nejméně 10 zločinců:

SELECT RC, PRIJMENI, JMENO FROM CLOVEK, ZLOCINEC OR PRIJMENI =ANY(SELECT DISTINCT PRIJMENI FROM CLOVEK WHERE ZLOCINEC) OR PSC IN(SELECT PSC FROM CLOVEK WHERE ZLOCINEC GROUP BY PSC HAVING COUNT(PSC)**>=**10) PODEZIRAN ORDER BY PRIJMENI, JMENO, RC;

Seznam okresů s největším počtem trvale bydlících zločinců by se také hodil. V takovém seznamu bude patrně jeden okres, ale není vyloučeno, že tam bude více okresů. Trochu si zaagregujeme v hlavním dotazu i v poddotazu, aby se nám podařilo přejít od jednotlivých zločinců ke zločineckým okresům:

SELECT OKRES FROM CLOVEK WHERE ZLOCINEC GROUP BY OKRES HAVING COUNT(RCISLO)**>=**ALL(SELECT COUNT(RCISLO) FROM CLOVEK WHERE ZLOCINEC GROUP BY OKRES) ORDER BY OKRES;

Položme si otázku, na co by se mohla hodit tabulka o neomezeném počtu řádků a sloupců uvnitř zahnízděného dotazu.

Existence řádků tabulky

V obecné tabulce s více sloupci je uložena směs hodnot různého typu, které těžko budeme hromadně porovnávat s hodnotou jednoho typu. Navíc to není k ničemu. Jediné, co je zajímavé testovat na tabulce o neurčeném počtu sloupců, je, zda obsahuje alespoň jeden řádek. To se hodí spíše

pro řízení výpočtů než pro zahnízďování dotazů. Operátor **EXISTS** určí, zda existuje alespoň jeden řádek tabulky vygenerované poddotazem. Operátor NOT EXISTS má pouze opačný význam. Za operátorem se nachází zahnízděná tabulka a před ním není hodnota. Pokud bych chtěl poslat všem nepodvodníkům seznam všech podvodníků, pak přílohu dopisu stvořím snadno. Stačí napsat příkaz:

```
SELECT RC, PRIJMENI, JMENO, BYDLISTE, PSC FROM CLOVEK WHERE PODVODNIK
ORDER BY PRIJMENI, JMENO, RC;
```

Jak ale vygenerovat samolepky na obálky? Takhle to dělají začátečníci nebo pesimisté:

```
SELECT PRIJMENI, JMENO, BYDLISTE, PSC FROM CLOVEK WHERE NOT PODVODNIK
ORDER BY PSC, PRIJMENI, JMENO;
```

Profesionální optimisté šetřící papír do tiskárny raději píší:

```
SELECT PRIJMENI, JMENO, BYDLISTE, PSC FROM CLOVEK WHERE EXISTS (SELECT *
FROM CLOVEK WHERE PODVODNIK ) AND NOT PODVODNIK ORDER BY PSC, PRIJMENI,
JMENO;
```

Nebudu přeci psát milionům nepodvodníků o tom, že seznam podvodníků je prázdný. Jistě by to šlo napsat i bez EXISTS:

```
SELECT PRIJMENI, JMENO, BYDLISTE, PSC FROM CLOVEK WHERE (SELECT
COUNT(RCISLO) FROM CLOVEK WHERE PODVODNIK )>0 AND NOT PODVODNIK ORDER BY
PSC, PRIJMENI, JMENO;
```

Vidíte, že cesty k zvládnutí rozmanitých dotazů do jedné tabulky jsou velmi členité a strukturovanost z nich jenom čiší. Někteří čtenáři si možná kladou otázku, proč jsem v úvodu seriálu rozsekával problém na několik tabulek a proč stále ještě tajím, jak to všechno spojit do jednoho celku ku všeobecnému užitku. Nechtěl jsem vás zatím příliš rozptylovat. Počínaje následující malou kapitolou už se nebudu vícetabulkovým dotazům vyhýbat.

Práce se dvěma tabulkami

Ono velké tajemství zní: **poddotaz může směřovat i do jiné zdrojové tabulky**. Nejprve si novou možnost vychutnáme na dvojicích tabulek v 5NF obrněných doménovou a entitní integritou, které mají

k sobě relaci N:1. Co kdyby nás zajímali lidé, kteří nemají ani jeden účet a je jim víc než 18? Chceme jim poslat skvělou nabídku od naší nové banky GOLD TRANSILVANIA Ltd:

SELECT PRIJMENI, JMENO, BYDLISTE, PSC FROM CLOVEK WHERE VEK>=18 AND RCISLO NOT IN(SELECT DISTINCT RC FROM UCET)ORDER BY PSC, PRIJMENI, JMENO;

Zajímavý je též dotaz do rejstříku trestů a registru firem:

SELECT RC, PRIJMENI, JMENO FROM REJSTRIK WHERE TRESTAN AND RC=ANY(SELECT DISTINCT RC FROM REG_FIREM) ORDER BY PRIJMENI, JMENO, RC;

Sněhurka šílela z milých a závislých trpaslíků. V tabulce trpaslíků se dobře loví pomocí podřízené tabulky vlastností, která obsahuje číslo trpaslíka a název vlastnosti jako složený klíč:

SELECT JMENO FROM TRPASLIK WHERE CISLO IN(SELECT DISTINCT CISLO FROM VLASTNOST WHERE NAZEV=”MILY”) AND CISLO IN(SELECT DISTINCT CISLO FROM VLASTNOST WHERE NAZEV=”ZAVISLY”)ORDER BY JMENO;

Práce s více tabulkami

Z více tabulek nemusíte mít obavy. Pokud je databázový systém dobře navržen, jde všechno hladce. Vzpomeňme si na první díl seriálu a na číselník receptů, číselník surovin a na spojovací entitu DAVKA, která obsahovala dva klíčové sloupce CIR a CIS pro orientaci v číselnících a jeden neklíčový sloupec pro množství suroviny v rámci receptu. Mám teď zrovna chuť na KARI koření a nechce se mi je jíst samotné. Poddotazem budu muset zjistit číslo suroviny, kterého se zmocní nadřízený poddotaz, a stanoví čísla receptů, které potřebují KARI, a konečně hlavní dotaz vypíše názvy receptů s KARI podle abecedy. Takže pozor na hnízdo v hnízdě:

SELECT CIR, NAZEV FROM RECEPT WHERE CIR=ANY(SELECT CIR FROM DAVKA WHERE CIS=ANY(SELECT CIS FROM SUROVINA WHERE NAZEV =”KARI”)) ORDER BY NAZEV, CIR;

Některé suroviny nejsou k ničemu a chci si je uvědomit:

SELECT CIS, NAZEV FROM SUROVINA WHERE CIS NOT IN (SELECT DISTINCT CIS FROM DAVKA) ORDER BY NAZEV, CIS;

Chci založit GOULASH PUB, s.r.o., a zajímá mě, z čeho se vlastně guláš dělá. Musím se prohrabat recepty, které připomínají guláš, a opsat si jejich čísla. Pak se zanořím do dávek a najdu čísla potenciálních surovin. Pak teprv budu moci abecedně vypsat suroviny do gulášů:

```
SELECT CIS, NAZEV FROM SUROVINA WHERE CIS IN (SELECT DISTINCT CIS FROM
DAVKA WHERE CIR IN (SELECT CIR FROM RECEPT WHERE NAZEV LIKE ”%GULAS%”)) ORDER
BY NAZEV, CIS;
```

Dobrou chuť – i na spojování tabulek pomocí joinů v dalším dílu!

Jaromír Kukal