УДК 681.3.06

Глибовець М. М., Кулябко П. П.

ЗАПИТИ З МНОЖИННИМИ ПОРІВНЯННЯМИ У МОВІ SQL

На основі порівняльного аналізу виразових можливостей SQL-92 стандарта і класичного SQL(SEQUEL) пропонуються схеми реалізації складних запитів з множинними порівняннями засобами SQL Access.

Мова SQL на сьогоднішній день є найбільш поширеною мовою запитів СУБД реляційного типу і підтримується практично всіма програмними продуктами, які представлені на ринку технологій, пов'язаних з базами даних та інформаційними системами.

Виробники програмних продуктів при розробці мовних процесорів SQL орієнтуються на стандарт 1992 року або SQL-92 [3]. Проте ця мова має більш давню історію, бо поширилася ще у 1974 році [1] під назвою SEQUEL.

Мова SEQUEL була задумана як мова функціонального типу з відповідними засобами та технологічними прийомами програмування. Під тиском користувачів, переважна більшість яких була зорієнтована на процедурний стиль програмування, в наступній версії мови SEQUEL/2 [2] з'являються додаткові засоби, такі як GROUP BY та множинні функції типу SET.

Пізніше, на початку 80-х років, мова SEQUEL/2 отримала нову назву SQL, а згодом у нових версіях цієї мови зникають можливості для множинного порівняння, що було зумовлено значною складністю алгоритмів порівняння множин, а відтак намаганням розробників зорієнтувати користувачів на застосування більш ефективних схем запитів пошуку даних. Але, на нашу думку, деякі схеми запитів класичного SEQUEL/2 більш зручні для користувача (принаймні з логічної точки зору), бо мають стрункішу логічну структуру і чіткіше семантичне тлумачення через теорію множин та теорію відображень у порівнянні з функціонально еквівалентними їм схемами запитів, що виражені засобами SQL-92. Проілюструємо це твердження рядом прикладів.

Для розгляду прикладів типів запитів опишемо структуру таблиць з предметної сфери "Розклад вищого навчального закладу".

Лектор	Код лектора	Прізвище	Організація	Телефон	Науковий ступінь
Lect	CdL	Name	Org	Phone	Degree

Предмет	Код предмета	Назва	Кількість годин	Тип	Контроль
Subj	CdS	Nm	Hours	Tp	Testing

Група	Код групи	Курс	Кафедра	Факультет	Кількість чоловік
Group	CdG	Course	Caf	Faculty	Quant

Розклад	Код лектора	Код предмета	Код ГОУПИ	Аудиторія	День	Пара	Тиждень
Sch	CdL	CdS	CdG	Aud	Day	TmL	Week

Розглянемо тип запиту, що пропонується в [4]:

"Знайти прізвища лекторів, які читають всі предмети."

В термінах мови SEQUEL (класичний) цей запит може бути представлений так:

```
SELECT DISTINCT Lect.Nam
FROM Lect
WHERE Lect.CdL IN
    (SELECT Sch.CdL
    FROM Sch
    GROUP BY Sch.CdL
    HAVING SET(Sch.CdS) =
            (SELECT Subj.CdS
            FROM Subj))
```

Але в SQL-92 (і також в SQL Access) немає засобів для множинного порівняння, тому потрібна суттєва модифікація.

```
1)  варіант (кількісний) /* Count*/
SELECT Lect.Name
FROM Lect
WHERE Lect.CdL IN (SELECT Sch.CdL
    FROM Sch
    GROUP BY Sch.CdL
            HAVING Count(Sch.CdS) =
            = (SELECT Count(Subj.CdS)
              FROM Subj))
```

Така технологія дуже зручна, але, на жаль, значною мірою специфічна, бо фактично відбувається заміна початкового запиту на інший:

"Знайти прізвища лекторів, які читають таку ж *кількість* предметів, що дорівнює *кількості* **всіх** існуючих предметів."

У наведеному контексті ці запити семантично еквівалентні, але подібний прийом буде некоректним, якщо відносно існуючих предметів з'явиться певна умова (обмеження), і тоді кількісне порівняння не зможе замінити порівняння множин при збереженні семантичної еквівалентності.

```
2) варіант (ні –ні ) /* not – not */
SELECT DISTINCT Lect.Name
FROM Lect
WHERE NOT EXISTS (SELECT *
                  FROM Subj
WHERE NOT EXISTS (SELECT *
                  FROM Sch
WHERE Lect.CdL = Sch.Cdl AND Subj.Cds =
= Sch.CdS))
```

Цей варіант, що теж наводиться в [4], передбачає перетворення початкового запиту на рівні природної мови до такого (семантично еквівалентного початковому):

"Знайти прізвища лекторів, для яких *не* існує предметів, які б вони *не* читали."

Специфіка цього варіанта полягає в необхідності перетворень початкового запиту на рівні природної мови, що, по-перше, в багатьох випадках досить складно, особливо з урахуванням обгрунтування еквівалентності перетворення, а, по-друге, важко описати технологію такого перетворення в зв'язку з неформальністю та неоднозначністю природної мови.

Інший підхід до проблеми множинного порівняння може базуватися на тій обставині, що з усіх операцій реляційної алгебри Кодда тільки операція ділення не дозволяє безпосереднього представлення засобами SQL-92 (чи SQL Access). Але операція ділення не є незалежною, тобто вона може бути виражена через інші операції реляційної алгебри. Зокрема, в [4] наводиться відповідна формула.

Якщо задані дві реляції A[X,Y] і B[Y], то

$$(A[Y \div Y]B) \equiv (A[X] \setminus ((A[X] \otimes B) \setminus A)[X]),$$

де ÷ — операція ділення; [ ] — операція проекції; \ — операція різниці; ⊗ — декартів добуток.

Усі операції, що входять у праву частину тотожності, допускають безпосереднє представлення засобами SQL-92.

Більш технологічним, на наш погляд, є підхід, який можна вважати (до певної міри) розширенням варіанта "ні-ні", але з формальними перетвореннями на рівні теоретико-множинних співвідношень.

Розглянемо запит:

"Знайти коди лекторів, які читають принаймні **всім** групам кафедри МІ".

В термінах класичного SEQUEL цей запит може виглядати, наприклад, так:

```
SELECT DISTINCT Cdl
FROM Sch
GROUP BY Cdl
HAVING SET(Cdg) /* A */
            CONTAINS
    (SELECT Cdg /* B */
    FROM Grp
    WHERE Grp.Caf = "mi")
```

Якщо множину кодів груп SET(Cdg) позначимо через A, а множину (SELECT Cdg ...) — через B, то отримаємо співвідношення

$$(A \supseteq B) \Leftrightarrow (B \setminus A = \varnothing) \text{ або } (B \cap (-A) = \varnothing),$$

(останнє особливо важливо для SQL Access в зв'язку з відсутністю операції EXCEPT). Відповідно до останнього можемо отримати таку SQL-програму:

```
/* CONTAINS — NOT STRICT*/
SELECT DISTINCT x.Cdl
FROM Sch AS x
WHERE NOT EXISTS (SELECT *
    FROM Grp
    WHERE Grp.Caf = "mi"
    AND
    (Grp.Cdg NOT IN (Select Sch.Cdg
                     FROM Sch
                     WHERE Sch.Cdl =
                     = x.Cdl)))
```

Якщо потрібне строге включення, то маємо

$$(A \supset B) \Leftrightarrow ((B \setminus A = \varnothing) \;\&\; (A \setminus B \neq \varnothing)) \text{ або}$$

$$(B \cap (-A) = \varnothing) \;\&\; (A \cap (-B) \neq \varnothing)$$

```
/* CONTAINS — STRICT */
SELECT DISTINCT x.Cdl
```

```
FROM Sch AS x
WHERE NOT EXISTS (SELECT *
                  FROM Grp
WHERE Grp.Caf = "mi"
    AND
(Grp.Cdg NOT IN (Select Sch.Cdg
                 FROM Sch
WHERE Sch.Cdl = x.Cdl)))
    AND EXISTS
(Select Sch.Cdg
From Sch
WHERE (Sch.Cdl = x.Cdl)
    AND
(Sch.Cdg NOT IN (Select Grp.Cdg
FROM Grp
WHERE Grp.Caf = "mi" )))
```

Для випадку рівності множин маємо:

$(A = B) \Leftrightarrow ((B \setminus A = \varnothing) \,\&\, (A \setminus B = \varnothing))$ або

$(B \cap (-A) = \varnothing) \,\&\, (A \cap (-B) = \varnothing)$

```
/* EXACTLY */
SELECT DISTINCT x.Cdl
FROM Sch AS x
WHERE NOT EXISTS (SELECT *
    FROM Grp
    WHERE Grp.Caf = "mi"
        AND
    (Grp.Cdg NOT IN (Select Sch.Cdg
        FROM Sch
        WHERE Sch.Cdl = x.Cdl)))
            AND NOT EXISTS
            (Select Sch.Cdg
FROM Sch
WHERE (Sch.Cdl = x.Cdl)
AND (Sch.Cdg NOT IN (Select Grp.Cdg
FROM Grp
WHERE Grp.Caf = "mi" )))
```

Наприкінці зауважимо, що хоча складні запити з множинними порівняннями в практичній роботі займають відносно невелику частку у загальній сукупності запитів, але виразити їх простими типами запитів (не виходячи за межі мови SQL) не вдається. Крім того, без можливостей множинного порівняння мова втрачає властивість *реляційної повноти* [4].

1. *Chamberlin D. D., Boyce R. F.* SEQUEL: A Structured English Query Language *II* Proc. ACM SIGMOD Workshop on Data Description, Access and Control.— Ann Arbor, Mich., 1974.
2. *Chamberlin D. D.* et al. SEQUEL/2: A Unified Approach to Data Definition, Manipulation, and Control *II* IBM J. R&D.— 1976— Vol. 20.— № 6; 1977.— Vol. 21.— № 1.
3. *Date C J., Darven H.* A guide to the SQL Standard.— Reading, Mass.: Addison-Wesley, 1993.
4. *Дейт К. Дж.* Введение в системы баз данных / 6-е изд.— К.: Диалектика, 1998.

Glibovets M. M., Kulyabko P. P.

QUERIES WITH SET COMPARISONS IN SQL

On the base of comparative analysis of SQL-92 standard expressive possibilities and classical SQL (SEQUEL) expressive possibilities are offered schemes to realization of complex queries with set comparisons by SQL Access facilities.