УДК 004.658.6 : 004.624

Фісун М.Т., Миколаївський державний гуманітарний університет ім. Петра Могили

Алгоритми та програми згортки перехресних таблиць в реляційні таблиці

Розглянуто підхід для заповнення зв'язувальних таблиць реляційних баз даних, що представляються в розгорнутому вигляді гіперкубами (у двовимірному випадку – крос-таблицями), коли одержання значень чарунок гіперкуба можна домогтися шляхом "ручного" балансування за допомогою електронної таблиці. Для балансування на гранях гіперкуба використовується таблиця Excel, потім вона експортується в таблицю бази даних Access, після чого згортається в реляційну таблицю. Цей процес повторюється для всіх атрибутів, що разом складають первинний ключ зв'язувальної таблиці бази даних. Наведено приклади реалізації згортання крос-таблиць мовами програмування SQL, VBA і Pascal.

The approach for connecting tables of the relational databases represented in a developed kind by hypercubes (in a two-dimensional case – cross-tables) when reception of values of cells of a hypercube may be to achieve by "manual" balancing with the help of a spreadsheet is considered. For balancing on the verge of a hypercube table Excel is used, then it is exported to the table of database Access, then it is turned off in the relational table. This process repeats for all attributes which together make a primary key of connecting database table. Examples of realization for turning off for cross-tables are given in programming languages SQL, VBA and Pascal.

Загальна постановка проблеми та її зв'язок з науково-практичними задачами

Як відомо, електронні таблиці є ефективним і зручним засобом при розв'язанні задач розподілення та отримання балансів [1; 2]. Будемо розглядати випадки, коли розподілення та/або отримання балансу виконується людиною, і при цьому немає потреби використовувати спеціальні моделі, наприклад, моделі лінійного програмування [3].

Системи керування базами даних (СКБД) зручні для ведення великої кількості взаємопов'язаних таблиць і для отримання у вигляді запитів, форм та звітів різного роду інформації з цих таблиць [4]. Але користувачу не дуже зручно працювати з пов'язаними таблицями в діалоговому режимі для різного роду балансувань. Проілюструємо це на прикладі задачі "постачальники (**П**) – споживачі (**С**)**".** Хай початкові умови задані такою табл. 1, комірки якої закодовані двовимірними індексами.

Таблиця 1

Початкові умови задачі розподілення

Постачальники	**Споживачі**	**С1**	**С2**	**С3**	**Разом**
	Потужності **П***i***Cj*	**50**	**60**	**30**	**140**
П1	**40**	10		10	
П2	**30**		15		
П3	**50**	30			
П4	**20**			5	
Всього	**140**				

Як бачимо, витримано баланс між загальними потужностями усіх постачальників і усіх споживачів, але поки що не зроблено закріплення постачальників за споживачами (або навпаки), яке витримує наведений баланс. Поки що в таблиці наведені окремі закріплення, які не можна змінювати, хоча це не принципово для подальшого викладення матеріалу.

Таку таблицю легко створити у програмному середовищі електронної таблиці, наприклад *Excel,* і далі розробляти варіанти розв'язання задачі розподілення (закріп-лення) з дотриманням балансу. Коли таких таблиць небагато і користувача цікавить тільки задача розподілення, то можна обійтися тільки засобами електронної таблиці. Але частіше за все користувачу потрібно створити базу даних (БД) для отримання різноманітних запитів, форм і звітів. Тому в ній, крім аналога таблиці, що наведена на рисунку, потрібні й інші таблиці.

Так, у нашому демонстраційному прикладі така таблиця-матриця після нормалізації [4] буде представлена трьома таблицями: двома основними "Постачальники" **(П)** і **"**Споживачі" **(С)** та зв'язувальною таблицею "Постачальники-споживачі" **(П_С).** Структура і вміст цих таблиць бази даних наведено нижче у табл. 2, 3, 4. Як бачимо, таблиця 1 являє собою двовимірне представлення моделі початкових умов задачі, сутність якого

Таблиця 2
Таблиця БД „П"

NP	*QP*
П1	40
П2	30
П3	50
П4	20

Таблиця 3
Таблиця БД „С"

NC	*QC*
С1	50
С2	60
С3	30

Таблиця 4
Таблиця БД „П_С"

NP	*NC*	*QTY*
П1	С1	10
П1	С3	10
П2	С2	15
П3	С1	30
П4	С3	5

У таблицях 2-4 застосовані такі позначення:

NP – код постачальника;
QP – ресурс постачальника;
NC – код споживача;
QC – потреба споживача;
QTY – кількість закріпленого ресурсу постачання/споживання.

Даталогічна модель [2] такої БД представлена на рисунку.

Даталогічна модель гіпотетичної бази даних про постачання

Ставиться задача закріпити споживачів за постачальниками, при цьому необхідно не порушити баланс між сумарними потужностями усіх постачальників й усіх споживачів.

Якщо не задано жодного елемента матриці, то маємо систему $m+n$ рівнянь (умови балансу) з m змінними. Можна показати, що така система рівнянь має багато рішень, і вони можуть бути знайдені, у загальному випадку, методом перебору, алгоритм якого залежить від умов конкретної задачі. Якщо для такої неповної системи лінійних рівнянь сформульована цільова функція (максимум прибутку, мінімум витрат на перевезення і т.п.), то така задача зводиться до моделі лінійного програмування, при цьому рівняння балансу будуть обмеженнями. Такі задачі розв'язуються відповідними програмами. Але в реальній практиці економічних розрахунків дуже часто зустрічаються випадки, коли оптимізаційна задача не ставиться, тоді їх зручно розв'язувати за допомогою електронних таблиць у діалоговому режимі, тому що користувач відчуває умови своєї задачі і може досить швидко її розв'язати. У табл. 5 наведено одне з рішень, яке отримано в *Exel*. Курсивом показані значення, що

Таблиця 5

Результат закріплення постачальників за споживачами

	Споживачі	**С1**	**С2**	**С3**	**Разом**
Постачальники	**Потужності П*i*\С*j***	50	60	30	**140**
П1	**40**	10	20	10	40
П2	**30**	0	15	15	30
П3	**50**	30	20		50
П4	**20**	10	5	5	20
Всього	**140**	50	60	30	**140**

Якщо тепер зробити зворотне перетворення *Excel*-таблиці у таблиці бази даних (наприклад, *Access*) [], то вміст таблиць **П** і **С** не зміниться, а вміст таблиці **П_С** наведено у табл. 6.

Таблиця 6

Таблиця БД „П_С" після розподілення

NP	*NC*	*QTY*
П1	С1	10
П1	С2	20
П1	С3	10
П2	С2	15
П2	С3	15
П3	С1	30
П3	С2	20
П4	С1	10
П4	С2	5
П4	С3	5

Будемо називати перетворення нормалізованих таблиць типу, що наведені в таблицях 1 і 5, в таблиці типу, що наведені в таблицях 4 і 6, *згорткою*, а зворотні перетворення – *розгорткою*. В сучасних СКБД поки що на верхньому рівні програмування (засобами візуального програмування або мовою *SQL* реалізована тільки процедура розгортки, яка здійснюється, зокрема, в середовищі *Access* груповим запитом *Crosstab* або інструкцією *Transform*) в останніх версіях СКБД *Access*

– запитом *Pivot*, але процедура згортки не реалізована безпосередньо груповим оператором, але потреба мати груповий запит зворотної дії є актуальною.

Тому **метою досліджень** у даній статті є розробка деяких прийомів реалізації групової операції згортки перехресних таблиць у звичайну реляційну таблицю бази даних.

Результати досліджень. Найпростішим прийомом можна вважати програмування запиту на створення таблиці мовою *SQL* з багаторазовим використанням оператора *Union*. Нижче наводиться текст програми (запиту) мовою *SQL* (середовище СКБД *Access* [5]) для нашого прикладу, при цьому розглядається тільки центральне питання – перетворення крос-таблиці, що наведена в таблиці 5, у зв'язувальну таблицю **П_С**(*NP, NC, QTY*), що наведена в таблиці 6. При цьому питання реалізації розгортки (запит *Crosstab*), а також питання імпорту/експорту *Access-Excel* тут не розглядаються. Тому, якщо заголовок крос-таблиці буде **П_С_Cross** (*NP, C1, C2, C3*), тоді *SQL*-запит у середовищі може бути таким:

```
Select П_С_Cross.NP, "C1" As NC,
П_С_Cross.[C1] As QTY
From П_С_Cross
Where (((П_С_Cross.[C1]) Is Not Null))
Union
Select П_С_Cross.NP, "C2" As NC,
П_С_Cross.[C2] As QTY
From П_С_Cross
Where (((П_С_Cross.[C2]) Is Not Null))
Union
Select П_С_Cross.NP, "C3" As NC,
П_С_Cross.[C3] As QTY
From П_С_Cross
Where (((П_С_Cross.[C3]) Is Not Null))
```

В результаті цього запиту буде отримана віртуальна таблиця бази даних **П_С**, структура і вміст якої наведено в таблиці 6. При необхідності можна одразу її створити як таблицю бази даних. Як бачимо із тексту *SQL*-програми, вона містить сукупність (за кількістю стовпців, що згортаються) однотипних *Select*-операторів, об'єднаних *Union*-операторами. В *Select*-операторах додається новий атрибут, що відповідає „горизонтальному виміру" в *Cross*-таблиці (в нашому прикладі це номер споживача – *NC*), і покроково заповнюються значення цього атрибута як назва стовпця *Cross*-таблиці та вибираються значущі комірки із даного стовпця.

Таким чином, узагальнену задачу можна сформулювати таким чином.

Маємо таблицю бази даних **A** з кардинальним числом ***m*,** до складу якої входить ключовий атрибут *a*, таблицю **B** з кардинальним числом ***n*,** до складу якої входить ключовий атрибут *b*, та зв'язувальну таблицю **A_B,** до складу якої входять атрибути *a* і *b* як зовнішні ключі, що разом складають первинний ключ, а також атрибут *c***.** При цьому атрибути *a, b* і *c* можуть бути складеними. Для розв′язання задачі розподілення (визначення $c\mathbf{ab}$ за допомогою електронної таблиці *Excel* і наступного занесення цих даних до бази даних *Access*) пропонується така послідовність дій.

1. В середовищі *Access* над таблицею **A_B** здійснюється *Crosstab*-запит, де "заголовками" рядків будуть значення атрибута ***a***, "заголовками" стовпців-атрибутів – значення атрибута ***b***. Крос-таблиця, що буде отримана, з математичної точки зори є матрицею інциденцій відношень **A** і **B**, тобто на перерізах рядків ***ai*** та стовпців ***bj*** відображається факт існування взаємозв'язку між цими відношеннями та властивість цього відношення. Хай ім'я крос-таблиці буде **A_B_Cros.**
2. Одним із можливих варіантів експорту (самий простий – за допомогою *Clipboard*) переносимо **A_B_Cros** у середовище *Excel***.**
3. Виконуємо балансування даних таблиці з перевіркою обмежень по рядках і по стовпцях.
4. Шляхом імпорту із *Excel* створюємо нову таблицю **A_B_Cros** в середовищі *Access***.**
5. За допомогою *SQL*-запиту, аналогічного вищенаведеному, отримуємо нову

таблицю бази даних **A_B.**

Якщо кількість стовпців крос-таблиці велика, то скрипт (код) *SQL*-запиту буде довгим (кількість інструкцій *"select… from … where"*, з'єднаних інструкцією "*union"*, дорівнюватиме кількості стовпців крос-таблиці, що згортаються), тому для таких випадків краще розробити програму (наприклад, мовою *VBA,* якщо обрана СКБД *Access*), яка б виконувала операцію згортки. Нижче наводиться текст такої програми (модуля), який прив'язаний до відповідної форми. Вона перетворює перехресну *Excel*-таблицю П_С_Cross.xls в згорнуту реляційну таблицю бази даних *Access.*

```
Private Sub Zgortka_Click()
Dim WB As Excel.Workbook,
      WS As Excel.Worksheet, n, k
Set WB = Excel.Workbooks.Open("d:\...\
П_С_Cross.xls")
Set WS = WB.Worksheets("CrosT")
'Перебираємо комірки таблиці спочатку по
колонках, потім – по рядках'
For k = 2 To 4 ' по колонках C1, C2, C3
For n = 2 To 5 ' по рядках
'Якщо комірка не пуста, то переносимо із
перехресної таблиці до згорнутої'
If WS.Cells(n, k) <> "" Then
NP = WS.Cells(n,1)
                  NC = WS.Cells(1,k)
                  QTY = WS.Cells(n, k)
'Далі здійснюється перехід на новий
рядок таблиці БД,'  яка заповнюється
кнопкою форми "MakeTable"'
DoCmd.GoToRecord acDataForm,
"MakeTable", acNext
End If
Next n
Next k
'Далі здійснюється відключення Excel від
таблиці'
Set WS = Nothing
Set WB = Nothing
Excel.Workbooks.Close
End Sub
```

Таку ж структуру й алгоритм має програма згортання перехресної таблиці, що існує в базі даних СКБД *Access*.

Нижче наводиться ще текст програми мовою *Pascal* [6], оскільки він частіш за все використовується для опису алгоритмів. Крім того, в архітектурах розподілених баз даних „клієнт-сервер" для розробки прикладних програм, що працюють із серверною базою даних, використовується програмне середовище *Delphi* [7], в якому мова *Pascal* є його підмножиною.

```
program Zgortka (E,R);
{Програма працює з двома файлами:
E – вхідний файл, який вище представлено
таблицею 4, тобто крос-таблицею.
Будемо вважати, що його наповнення
здійснено за межами даної програми;
R – вихідний файл, який вище представлено
таблицею П_С (див. табл. 6, тобто він
буде результатом згортки).
   Алгоритм програми полягає у
покроковому зчитуванні кожного запису із
файла E і перевірці значення кожної
комірки на перерізі активного рядку
(запису) і стовпця, що згортається, на
наявність там значення C[i,j], де i – номер
рядка, j – номер стовпця, що
розглядається. Оскільки для полів файла у
ПАСКАЛі відсутнє значення nil, то у
нашому прикладі ознакою відсутності
значення є будь-яке негативне число.
   Якщо C[i,j] має позитивне значення, то
до файла R додається ще один запис за
правилами, що наведені у коментарях до
відповідних фрагментів програми.}


type
RecE = record // Опис запису вхідного
файла E.
NP:string[10];          C1,C2,C3:real;
end;
RecR = record // Опис запису вихідного
файла R.
NP,NC:string[10];   QTY:real;
end;
var
C1,C2,C3:string[5];
// Змінні для збереження назв стовпців,
// що згортаються
   E:file of RecE;   // Вхідний файл
   R:file of RecR;   // Вихідний файл
   Erec:RecE;
//  Змінна запису вхідного файла
   Rrec:RecR;
```

```
 // Змінна запису вихідного файла
begin
  C1:='C1'; C2:='C2'; C3:='C3';
   reset(E);   rewrite(R);
  while not Eof(E) do //Організуємо цикл,
доки не дійдемо до кінця  файла
     begin
       read(E,Erec);
// Зчитуємо черговий запис вхідного файла
       if Erec.C1>0   then
// Перевіряємо на значення комірку першого
стовпця, що згортається
          begin
            RRec.NP:=ERec.NP;
 // Цей і наступні два оператори
            Rrec.NC:=C1; //  формують запис
            Rrec.QTY:=Erec.C1; //  для
додавання до вихідного файла
            write (R,RRec);
//Додавання сформованого запису до
вихідного файла
          end;  // if Erec.C1>0
       if Erec.C2>0   then
// Перевіряємо на значення комірку другого
стовпця   і виконуємо аналогічні дії, що і
для "C1"
          begin
            RRec.NP:=ERec.NP;  Rrec.NC:=C2;
Rrec.QTY:=Erec.C2;
            write (R,RRec);
          end;  // if Erec.C2>0
       if Erec.C3>0   then
// Перевіряємо на значення комірку
третього стовпця
          begin
// і виконуємо аналогічні дії, що і для "C1"
            RRec.NP:=ERec.NP;  Rrec.NC:=C3;
Rrec.QTY:=Erec.C3;
            write (R,RRec);
          end;  // if Erec.C3>0
//  Такі ж дії виконуються і для інших
стовпців, якщо вони ще є у вхідному файлі
     end;{while not Eof(E)}
  CloseFile(R);    CloseFile(E);
  // Команди виводу на екран вмісту
вихідного файла R тут не наводяться
 end.
```

Висновки

За допомогою такої методології можна розв'язувати задачі не тільки для двовимірних таблиць, але й для n-мірних крос-таблиць, виконуючи поступово вищенаведені балансування на кожній поверхні гіперкуба. Зокрема, такі програми можна використовувати при створенні бази даних інформаційної системи кафедри ВНЗ при складанні робочого навчального плану [8]. Серед таблиць названої бази даних є

Література

1. Додж М., Стинсон К. Эффективная работа з Microsoft Excel 2002. – СПб.: Издательство "Питер", 2003. – 1056 с.
2. Куперштейн В. Современные информационные технологии в делопроизводстве и управлении. – СПб.: Издательство "Питер", 2000. – 805 с.
3. Зайченко Ю.П. Дослідження операцій: Підручник для ВНЗ. – Київ: ЗАТ "ВІПОЛ", 2000. – 688 с.
4. Дейт К. Дж. Введение в системы баз данных. – 7-е изд. – М.: Изд. дом "Вильямс", 2001. – 1072 с.Пасько В. Access-97. Система управления реляционными базами данных. – Дюссельдорф; Москва; Санкт-Петербург; Киев: Изд. гр. BHV, 1997. – 480 с.
5. Марченко Ю.А., Марченко Л.А. Программирование в среде Borland Pascal 7.0. – К.: Юпитер, 1997. – 496 с.
6. Канту М. Delphi 2 для Windows 95/NT. Полный курс: В 2 т. – Т. 2. – М.: Малип., 1997. – 400 с.
7. Фісун М.Т. Автоматизація складання робочого навчального плану / Вестник Херсонского государственного технического университета. – 2003. – № 2(18). – С. 440-445.

Стаття надійшла до редколегії 15.01.2004 р.