

# SQL Việt blog

Thảo luận về lập trình và quản trị Microsoft SQL Server

## Ứng Dụng của Join Bất Cân Bằng

Guess Post: Vũ Minh Tâm

Tiếp theo bài viết [Cần thận với Join bất cân bằng](#), bài viết này mô tả một trường hợp thường gặp có thể xử lý bằng Join bất cân bằng (còn được gọi bằng Non Equi-Joins). Đó là những bài toán liên quan đến cách tính bậc thang. Ví dụ giá điện dùng trong khoảng 0-100 thì áp mức giá 100 cho 1 số, từ 101 – 200 lại áp mức giá 200. Cần tính tiền cho từng khách hàng với số điện dùng trong tháng. Hoặc trong viễn thông, áp giá cho cuộc gọi từ 0-10 giây đầu là 1 đồng, từ giây thứ 11 là 2 đồng... Trong bưu chính, khối lượng hàng hóa từ 0-200g thì mức giá là 1000 đồng, từ 200-400g là 2000 đồng... Để giải quyết bài toán này có nhiều cách, một trong số những cách thức hiệu quả là sử dụng Join bất cân bằng vì những ưu điểm sau:

- Xử lý hàng loạt, không sử dụng Cursor
- Có thể cài đặt bằng SQL, thuận tiện khi triển khai trên những hệ thống khác ngoài SQL Server như Oracle, MySQL...

### MÔ TẢ BẢNG

Bảng Price gồm có 3 cột, Lower\_Value là giá trị dưới, Upper\_Value là giá trị trên còn Price là giá trong khoảng Lower\_Value và Upper\_Value. Script tạo bảng và Insert dữ liệu mẫu như sau:

```
CREATE TABLE PRICE
(
  LOWER_VALUE INT,
  UPPER_VALUE INT,
  PRICE INT
)

GO

INSERT INTO PRICE (LOWER_VALUE, UPPER_VALUE, PRICE)
SELECT 0, 100, 5
UNION ALL
SELECT 100, 500, 10
UNION ALL
SELECT 500, 1000, 20
UNION ALL
SELECT 1000, NULL, 50

GO

SELECT *
FROM PRICE
```

LOWER_VALUE	UPPER_VALUE	PRICE
0	100	5
100	500	10
500	1000	20
1000	NULL	50

Bảng thứ hai là bảng Customer. Bảng này gồm 3 trường Customer\_ID, Customer\_Name và Value. Script tạo bảng và dữ liệu mẫu như sau:

```
CREATE TABLE CUSTOMER
(
  CUSTOMER_ID INT,
  CUSTOMER_NAME VARCHAR(50),
  VALUE INT
)

GO

INSERT INTO CUSTOMER (CUSTOMER_ID, CUSTOMER_NAME, VALUE)
SELECT 1, 'Mr Smith', 80
```

```
UNION ALL
SELECT 2, 'Mr Conrad', 240
UNION ALL
SELECT 3, 'Mrs Oreski', 800
UNION ALL
SELECT 4, 'Red Devilic', 1700

GO

SELECT *
FROM CUSTOMER
```

CUSTOMER_ID	CUSTOMER_NAME	VALUE
1	Mr Smith	80
2	Mr Conrad	240
3	Mrs Oreski	800
4	Red Devilic	1700

**YÊU CẦU BÀI TOÁN**

Cần viết truy vấn, đưa ra những dữ liệu sau:

- Customer\_ID
- Customer\_Name
- Price tương ứng với Value của khách hàng

Ví dụ với khách hàng có Customer\_ID = 2, Customer\_Name là Mr Conrad, Value = 240 sẽ được tính theo công thức sau:

100 đầu tiên nằm trong dải 0 – 100 nên có Price  $100 \times 5 = 500$   
140 sau đó nằm trong dải 101 – 500 nên có giá  $140 \times 10 = 1400$   
Vậy Price của khách hàng này là  $500 + 1400 = 1900$ . Tương tự cho các khách hàng tiếp theo.

**GIẢI QUYẾT BÀI TOÁN BẰNG JOIN BẤT CÂN BẰNG**

Ý tưởng của giải pháp này được nhắc đến trong cuốn sách SQL Hacks (Andrew Cumming, Gordon Russell – Publisher O’Reilly – Nov 2006). Đầu tiên ta nhận thấy trong bảng Price thì Value của khách hàng sẽ “bao” toàn bộ các bản ghi có Lower\_Value thấp hơn. Như ví dụ ở trên, khách hàng có Value là 240 sẽ liên quan đến bản ghi thứ nhất và bản ghi thứ hai trong bảng Price ( $240 > 0$  và  $240 > 100$ ).

```
SELECT *
FROM Price p JOIN Customer c
ON c.VALUE > p.lower_value;
```

LOWER_VALUE	UPPER_VALUE	PRICE	CUSTOMER_ID	CUSTOMER_NAME	VALUE
0	100	5	1	Mr Smith	80
0	100	5	2	Mr Conrad	240
100	500	10	2	Mr Conrad	240
0	100	5	3	Mrs Oreski	800
100	500	10	3	Mrs Oreski	800
500	1000	20	3	Mrs Oreski	800
0	100	5	4	Red Devilic	1700
100	500	10	4	Red Devilic	1700
500	1000	20	4	Red Devilic	1700
1000	NULL	50	4	Red Devilic	1700

Trong số những bản ghi liên quan, nếu giá trị value của khách hàng > Upper\_Value, có nghĩa ta sẽ lấy trọn dải đó (Upper\_Value – Lower\_Value) đó nhân với Price. Ngược lại, sẽ lấy Value trừ đi Lower\_Value rồi nhân với Price. Ví dụ trong trường hợp value của khách hàng là 240. Ta đã lấy trọn 100 trong khoảng 0-100 nhân với Price là 5. Còn 140 ( $240 - 100$ ) nhân với Price ứng trong khoảng 101-500 là 10.

Đến đây thì bài toán đã đơn giản hơn rất nhiều, ta lấy Extract\_Value nhân với Price, sau đó SUM( ) toàn bộ với một phép Group By theo Customer\_ID. Câu lệnh cuối cùng sẽ như sau:

```
SELECT Customer_ID, CUSTOMER_NAME, SUM(Extract_Value * Price) RESULT
FROM
(
SELECT c.Customer_ID, c.Customer_Name,c.VALUE,p.Lower_Value,
CASE WHEN c.VALUE > p.Upper_Value THEN p.Upper_Value - p.Lower_Value
ELSE c.VALUE - p.Lower_value END Extract_Value, p.Price
```

```
FROM Customer c JOIN Price p ON c.VALUE > p.Lower_Value
) TMP
GROUP BY Customer_ID, CUSTOMER_NAME
ORDER BY Customer_ID
```

Customer_ID	CUSTOMER_NAME	Result
1	Mr Smith	400
2	Mr Conrad	1900
3	Mrs Oreski	10500
4	Red Devilic	49500

Tags: [JOIN](#)

14 Comments

Posted on 29/9/2014 | Categories: [SQL Server Programming](#)

### Các bài viết tương tự


- [Loại Bỏ Bản Ghi Trùng Trong Bảng](#)
- [Các Loại JOIN Trong SQL Server](#)
- [Lấy Một Số Bản Ghi Ngẫu Nhiên](#)

### Comments

-  *Nguyễn Khánh* (01/10/2014 2:45 am)

Hay quá. cảm ơn Vũ Minh Tâm.


[Reply](#)

-  *Quan tran* (06/10/2014 11:05 pm)

Quá hay! Thank you


Mình sẽ theo dõi thường xuyên.

[Reply](#)

-  *BacVT* (09/10/2014 12:08 am)


sau một lúc ngồi ngẫm code thì cuối cùng cũng đã ngẫm . Ôi cảm ơn bác Tâm

[Reply](#)

-  *Đăng Khoa* (09/10/2014 11:14 pm)


Rất hay, ngẫm code và chạy thử trên SQL 1 hồi đã hiểu ra, rất hữu ích khi ứng dụng cho thực tế! 😊 Cảm ơn bác Tâm! 😊

[Reply](#)

-  *vanhungbkcbg1* (15/10/2014 5:51 am)

bài viết này quá hay,ý tưởng rất thông minh,cảm ơn anh,hi vọng anh sẽ có nhiều bài viết hay hơn nữa cảm ơn anh

[Reply](#)

-  *nguoihanoi* (21/11/2014 1:04 am)

Thấy chủ xi bảo giải pháp này là ý tưởng trong cuốn sách SQL Hacks của bọn đế quốc Mỹ mà tự ái dân tộc nổi cuộn cuộn.

Đây là giải pháp trong cuốn sách Những luật vàng trong SQL (tác giả nguoihanoi) há há.

Người Việt chúng ta cần cù, chăm chỉ, thông minh, giỏi toán, có năng khiếu tin học, và nhiều thứ linh tinh khác (cụ Khái bảo thế – cảm cái 😊) nên không cần join “Cân Bằng” hay join “Bất Cân Bằng”, chẳng where on gì rảo, chỉ rằng với đề mà vẫn sinh con đẻ cái ngon lành.

```
SELECT customer_id, customer_name, SUM(ket_qua) ket_qua FROM customer
outer apply
(
    SELECT
        iif(VALUE between lower_value and isnull(upper_value,100000000), (value-lower_value)*price,0)
        +
        iif(VALUE > upper_value, (upper_value-lower_value)*price,0)
        ket_qua FROM price
) oa
GROUP BY customer_id, customer_name
ORDER BY customer_id
```

[Reply](#)

-  *Red Devilic* (27/11/2014 12:06 pm)

Nice post,

Hồi 2k5 mình có đọc về cái Outer Apply này nhưng không để ý, cũng không xem lại cách dùng. Thanks bác đã cho mình xem lại 😊


Nhưng hình như format câu lệnh bị vỡ, bác gửi lại câu đầy đủ dc ko ?

[Reply](#)

-  *Vũ Huy Tâm* (08/12/2014 2:50 pm)

Cái này đúng là độc chiêu. Like mạnh

[Reply](#)

-  *Nguyễn Hoàng Dũng* (10/02/2015 10:42 pm)


Bác chủ xị đã nói một điểm mạnh của join bất cân bằng là ” Có thể cài đặt bằng SQL, thuận tiện khi triển khai trên những hệ thống khác ngoài SQL Server như Oracle, MySQL...”.  
 Vì vậy làm theo kiểu VN mình thì chỉ xài được trên sql server thôi. Nước ngoài họ đi trước mình rất lâu rồi, giờ họ đã lên tầm tổng quát, không còn cục bộ với một hệ thống nhất định nữa rồi.

[Reply](#)

-  *Red Devilic* (01/03/2015 10:57 pm)

Thực ra cái IFF kia cũng có thể viết lại bằng ANSI SQL mà 😊

[Reply](#)

-  *Learning* (28/11/2014 3:34 am)

```
iif(VALUE > upper_value, (upper_value-lower_value)*price,0)
=> iif(VALUE > upper_value, (upper_value-lower_value)*price,0)
```

[Reply](#)

-  *Red Devilic* (01/12/2014 3:55 am)

Thanks bác.

Chưa cập nhật cái IIF mới này 😊



Câu 1:

•

•



Reply.

## Leave a Reply

	Name (required)
	Email (will not be published) (required)
	Website

Hướng dẫn: Để nhập mã T-SQL bạn dùng thẻ `<pre lang="tsql">` và `</pre>`.  
Ví dụ: `<pre lang="tsql">SELECT * FROM MyTable</pre>`

[Post Your Comment](#)

[Like](#) 592 people like this. Be the first of your friends.

*Đề nghị ghi rõ nguồn gốc khi trích dẫn hoặc đăng lại nội dung từ blog này*

[Giới Thiệu](#)

[SQL Việt Q&A](#) 

[Hội thảo SQL Server – HCMC 2013](#) 

[Seminar Presentation: Index trong SQL Server - Download](#)

Search

PHẢN HỒI MỚI

- toru on [Vi sao nên tránh viết SQL code trong ứng dụng](#)
- toru on [DELETE và TRUNCATE](#)
- Lê Lam on [Các Chế Độ Phục Hồi Của Database](#)
- in áo theo yêu cầu on [Cấu Hình SQL Server Để Gửi Mail](#)
- john thornhill on [Thêm Dữ Liệu Mới Vào Bảng Dùng LEFT JOIN](#)
- Tho Nguyen on [Tạo SQL Job Dùng Để Backup Database Hàng Ngày.](#)
- Phat Dat on [Tạo SQL Job Dùng Để Backup Database Hàng Ngày.](#)
- Amazon Solar led light on [Tận mạn về NULL](#)
- Amazon Solar lights for garden on [Tận mạn về NULL](#)
- thiết kế phòng ngủ nhỏ 6m2 on [Cấu Hình SQL Server Để Gửi Mail](#)
- CườngMH on [Tạo SQL Job Dùng Để Backup Database Hàng Ngày.](#)
- uniwersytet on [Viết Resume Xin Việc Như Thế Nào](#)
- Darren Slonski on [Các Kiểu Backup Trong SQL Server](#)
- Benton Pontillo on [Tạo SQL Job Dùng Để Backup Database Hàng Ngày.](#)
- HOAT on [Di chuyển Database File](#)
- NGUYỄN NHƯ HOẠT on [Sql Động](#)
- Nguyen Manh Cường on [Bảng Tam và Biến Kiểu Bảng](#)
- Thắng on [Làm Sao Giảm Bớt Dung Lượng File LOG Của SQL Server](#)
- Thắng on [Làm Sao Giảm Bớt Dung Lượng File LOG Của SQL Server](#)
- Lý Mạc Sầu on [Các Loại JOIN Trong SQL Server](#)

BÀI MỚI

- [Agile Và Tư Duy Chiều Ngang](#) 🗨️29
- [Ghép Nối Nhiều Bản Ghi Vào Một Dòng](#) 🗨️45
- [Log File Để Làm Gì](#) 🗨️2
- [Ứng Dụng của Join Bất Cân Bằng](#) 🗨️15
- [Cẩn Thận Với Join Bất Cân Bằng](#) 🗨️8
- [Kinh Nghiệm Phòng Vấn](#) 🗨️0
- [Viết Resume Xin Việc Như Thế Nào](#) 🗨️37
- [Cấu Hình SQL Server Để Gửi Mail](#) 🗨️32
- [User Lạc Mẹo](#) 🗨️8
- [Thiết Lập Cho xp\\_cmdshell](#) 🗨️2
- [\[LINH TINH\] Từ Vựng Lớp 3](#) 🗨️22
- [LEFT JOIN Với Mệnh Đề WHERE](#) 🗨️15
- [Di chuyển Database File](#) 🗨️2
- [Khai Trương Trang SQL Việt Q&A](#) 🗨️26
- [Tâm Sự Cuối Năm](#) 🗨️10
- [Partition Bảng Đã Có Sẵn](#) 🗨️16
- [Bảng Tạm và Biến Kiểu Bảng](#) 🗨️39
- [Tham Số Fill Factor](#) 🗨️4
- [Hội thảo SQL Server – HCMC 2013](#) 🗨️68
- [Tản mạn về NULL](#) 🗨️53

[Xem toàn bộ các bài post...](#)

## CHỦ ĐỀ

- [Bảo mật](#) (4)
- [Bên trong SQL Server](#) (3)
- [Community](#) (2)
- [Cộng đồng](#) (3)
- [Database Administration](#) (27)
- [Download](#) (2)
- [Eventual Consistency](#) (2)
- [Function](#) (2)
- [IDENTITY](#) (2)
- [Index](#) (10)
- [Linh tinh](#) (11)
- [Performance tuning](#) (11)
- [SQL Server Programming](#) (34)
- [Sql động](#) (4)
- [SSIS](#) (1)
- [Stored Procedure](#) (3)
- [Table partitioning](#) (6)
- [Thiết kế database](#) (12)
- [Tip & Trick](#) (8)
- [Uncategorized](#) (1)
- [View](#) (1)

## NHÃN

[backup](#) [Bookmark Lookup](#) [CASCADE DELETE](#) [CASE](#) [CHECKIDENT](#) [clustered index](#) [Common Table Expression](#) [CONVERT](#) [Covering Index](#) [CROSS JOIN](#) [Data file](#) [DATEPART](#) [DATETIME](#) [DBCC](#) [De-duplicate](#) [Default file location](#) [DELETE](#) [differential backup](#) [Dynamic Search](#) [Dynamic SQL](#) [Eventual Consistency](#) [EXEC\(\)](#) [Execution plan](#) [EXISTS](#) [filegroup](#) [File location](#) [filtered index](#) [FOREIGN KEY](#) [full backup](#) [FULL OUTER JOIN](#) [GROUP BY](#) [HAVING](#) [IDENTITY](#) [IDENTITY INCREMENT](#) [IDENTITY SEED](#) [IDENTITY](#) [INSERT IF EXISTS](#) [Include](#) [Index](#) [index partitioning](#) [Index Seek](#) [INFORMATION\\_SCHEMA](#) [INNER JOIN](#) [INSERT](#) [ISNULL](#) [JOIN](#) [Key Lookup](#) [LEFT](#) [LEFT JOIN](#) [LIKE %](#) [Log file](#) [login](#) [MySpace](#) [NEWID\(\)](#) [Nhật quán cuối cùng](#) [Non-clustered index](#) [NoSQL](#) [NOT IN](#) [NULL](#) [OBJECT\\_ID](#) [OUTER JOIN](#) [OUTPUT](#) [OUTPUT INTO](#) [Parameter](#) [PARTITION](#) [Performance](#) [phân đoạn](#) [PRIMARY KEY](#) [RANDOM](#) [recovery model](#) [RESEED](#) [RESET IDENTITY](#) [restore](#) [restore log](#) [ROUTINES](#) [ROW\\_NUMBER](#) [Sarg](#) [Sargable](#) [SCOPE\\_IDENTITY](#) [Search Condition](#) [SELECT](#) [sp\\_executesql](#) [sp\\_helpfile](#) [Sql động](#) [Stored Procedure](#) [SUBSTRING](#) [SYS.IDENTITY\\_COLUMNS](#) [table-valued function](#) [table partitioning](#) [Tempdb](#) [Tham số](#) [Thủ tục](#) [transaction log backup](#) [TRUNCATE](#) [Tìm kiếm động](#) [UNION ALL](#) [unique constraint](#) [unique index](#) [UPDATE](#) [user](#)

## Blog Việt

- [ddth.com \(SQL Server forum\)](#)
- [cione – Đào tạo trực tuyến](#)
- [Buu Nguyen blog](#)
- [Trang CSDL trên facebook](#)

## Blog Anh

- [SQLBlog.com](#)
- [SQL Server Central](#)
- [SQL Authority](#)
- [Database management and analytics](#)
- [SQL Server Customer Advisory Team](#)
- [SQL University](#)
- [SQL Server Pedia](#)
- [Brent Ozar's blog](#)



© SQL Việt blog 2019 · Powered by [Wordpress](#) · Design by [Thomas Klaiber](#)