**Triggers**

Trigger là một loại stored procedure đặc biệt được execute (thực thi) một cách tự động khi có một data modification event xảy ra như Update, Insert hay Delete. Trigger được dùng để đảm bảo Data Integrity hay thực hiện các business rules nào đó.

**Khi nào ta cần sử dụng Trigger:**

* Ta chỉ sử dụng trigger khi mà các biện pháp bảo đảm data intergrity khác như Constraints không thể thỏa mãn yêu cầu của ứng dụng. Nên nhớ Constraint thuộc loại Declarative Data Integrity cho nên sẽ kiểm tra data trước khi cho phép nhập vào table trong khi Trigger thuộc loại Procedural Data Integrity nên việc insert, update, delete đã xảy ra rồi mới kích hoạt trigger. Chính vì vậy mà ta cần cân nhắc trước khi quyết định dùng loại nào trong việc đảm bảo Data Integrity.
* Khi một database được denormalized (ngược lại quá trình normalization, là một quá trình thiết kế database schema sao cho database chứa data không thừa không thiếu) sẽ có một số data thừa (redundant ) được chứa trong nhiều tables. Nghĩa là sẽ có một số data được chứa cùng một lúc ở hai hay nhiều nơi khác nhau. Khi đó để đảm bảo tính chính xác thì khi data được update ở một table này thì cũng phải được update một cách tự động ở các table còn lại bằng cách dùng Trigger.   
    
  Ví dụ: ta có table Item trong đó có field Barcode dùng để xác định một mặt hàng nào đó. Item table có vai trò như một cuốn catalog chứa những thông tin cần thiết mô tả từng mặt hàng. Ta có một table khác là Stock dùng để phản ánh món hàng có thực trong kho như được nhập về này nào được cung cấp bởi đại lý nào, số lượng bao nhiêu (tức là những thông tin về món hàng mà không thể chứa trong Item table được)...table này cũng có field Barcode để xác định món hàng trong kho. Như vậy thông tin về Barcode được chứa ở hai nơi khác nhau do đó ta cần dùng trigger để đảm bảo là Barcode ở hai nơi luôn được synchonize (đồng bộ).
* Ðôi khi ta có nhu cầu thay đổi dây chuyền (cascade) ta có thể dùng Trigger để bảo đảm chuyện đó. Nghĩa là khi có sự thay đổi nào đó ở table này thì một số table khác cũng được thay đổi theo để đảm bảo tính chính xác. Ví dụ như khi một món hàng được bán đi thì số lượng hàng trong table Item giảm đi một món đồng thời tổng số hàng trong kho (Stock table) cũng phải giảm theo một cách tự động. Như vậy ta có thể tạo một trigger trên Item table để mỗi khi một món được bán đi thì trigger sẽ được kích hoạt và giảm tổng số hàng trong Stock table.

**Ðặc điểm của Trigger:**

* Một trigger có thể làm nhiều công việc (actions) khác nhau và có thể được kích hoạt bởi nhiều hơn một event. Ví dụ ta có thể viết một trigger được kích hoạt bởi bất kỳ event nào như Update, Insert hay Delete và bên trong trigger ta sẽ viết code để giải quyết cho từng trường hợp.
* Trigger không thể được tạo ra trên temporary hay system table.
* Trigger chỉ có thể được kích hoạt một cách tự động bởi một trong các event Insert, Update, Delete mà không thể chạy manually được.
* Có thể áp dụng trigger cho View.
* Khi một trigger được kích hoạt thì data mới vừa được insert hay mới vừa được thay đổi sẽ được chứa trong Inserted table còn data mới vừa được delete được chứa trong Deleted table. Ðây là 2 table tạm chỉ chứa trên memory và chỉ có giá trị bên trong trigger mà thôi (nghĩa là chỉ nhìn thấy và được query trong trigger mà thôi). Ta có thể dùng thông tin trong 2 table này để so sánh data cũ và mới hoặc kiểm tra xem data mới vừa thay đổi có hợp lệ trước khi commit hay roll back. (Xem thêm ví dụ bên dưới)
* Có 2 loại triggers (class) : INSTEAD OF và AFTER. Loại INSTEAD OF sẽ bỏ qua (bybass) action đã kích hoạt trigger mà thay vào đó sẽ thực hiện các dòng lệnh SQL bên trong Trigger. Ví dụ ta có một Update trigger trên một table với câu INSTEAD OF thì khi table được update thay vì update SQL Server sẽ thực hiện các lệnh đã được viết sẵn bên trong trigger. Ngược lại loại AFTER (loại default tương đương với keyword FOR) sẽ thực hiện các câu lệnh bên trong trigger sau khi các action tạo nên trigger đã xảy ra rồi.

**Tạo Một Trigger Như Thế Nào?**

Cú pháp căn bản để tạo ra một trigger có dạng như sau:

CREATE TRIGGER trigger\_name   
  
ON table\_name or view\_name   
  
FOR trigger\_class and trigger\_type(s)   
  
AS Transact-SQL statements

Như vậy khi tạo ra một trigger ta phải chỉ rõ là tạo ra trigger trên table nào và được trigger khi nào (insert, update hay delete. Sau chữ AS là các câu lệnh SQL xử lý công việc.

Ta hãy nghiên cứu một ứng dụng thực tiễn sau. Giả sử ta viết một application cho phép user có thể Insert, Update và Delete những thông tin nằm trong database. User này thường là những người không thông thạo lắm về computer mà chúng tôi thường gọi đùa là "bà tám". Vào một ngày đẹp trời, "bà tám" mặt mày tái xanh đến cầu cứu ta vì đã lỡ tay "delete" những thông tin khá quan trọng và hy vọng ta có thể phục hồi dữ liệu dùm. Nếu chúng ta không phòng xa trước khi viết application thì coi như cũng vô phương cứu chữa vì data đã hoàn toàn bị delete.  
  
Nhưng nếu bạn là một "guru" bạn sẽ gật gù "chuyện này khó lắm!" nhưng sau đó bạn chỉ tốn vài phút đồng hồ để rollback. Muốn làm được chuyện này chúng ta phải dùng một "chiêu" gọi là Audit (kiểm tra hay giám sát). Tức là ngoài các table chính ta sẽ thêm các table phụ gọi là Audit tables. Bất kỳ hoạt động nào đụng chạm vào một số table quan trọng trong database ta đều ghi nhận vào trong Audit table. Ví dụ khi user update hay delete một record trong table nào đó thì trước khi update hay delete ta sẽ âm thầm di chuyển record đó sang Audit table rồi mới update hay delete table chính. Như vậy nếu có chuyện gì xảy ra ta có thể dễ dàng rollback (trả record về chỗ cũ).

**Ví dụ:**

Ta có table Orders trong PracticeDB. Ðể audit các hoạt động diễn ra trên table này ta tạo ra một audit table với tên Aud\_Orders với các column giống y hệt với Orders table. Ngoài ra ta thêm vào 2 columns:

* Audit\_Type : với các giá trị có thể là 'I','U','D' để ghi nhận record được Insert, Update hay Delete
* Date\_Time\_Stamp : Data Type thuộc loại DateTime dùng để ghi nhận thời điểm xảy ra sự thay đổi, có vai trò như một con dấu.

 (Nếu trong môi trường nhiều user thì ta thêm một column UserID để ghi nhận user nào thay đổi).

 Sau đó ta sẽ tạo ra 3 trigger dùng cho việc audit  như sau:

--Insert Trigger  
 CREATE TRIGGER [AuditInsertOrders]   
 ON [dbo].[Orders]   
 FOR Insert   
 AS  
 insert into aud\_orders select \*,'I',getdate() From inserted

--Update Trigger  
 CREATE TRIGGER [AuditUpdateOrders]   
 ON [dbo].[Orders]   
 for UPDATE   
 AS  
 insert into aud\_orders select \*,'U',Getdate() from deleted   
  
 --Delete Trigger  
 CREATE TRIGGER [AuditDeleteOrders]   
 ON [dbo].[Orders]   
 FOR DELETE   
 AS  
 insert into aud\_orders select \*,'D',getdate() From deleted

Trong ví dụ trên khi user insert một record thì record mới vừa được insert sẽ nằm trong inserted table như đã trình bày ở phần trên. Do đó ta sẽ select tất cả các column trong inserted table cộng thêm Audit Type "I" và dùng hàm GetDate() trong SQL Server để lấy system date time dùng cho Date\_Time\_Stamp column, sau đó insert vào Aud\_Orders table. Tương tự với trường hợp Update và Delete, record đã được update hay delete nằm trong deleted table.

Như vậy trở lại trường hợp thí dụ ở trên nếu "bà tám" yêu cầu ta có thể vào tìm kiếm trong audit table để phục hồi lại record. Ngoài ra ta có thể dùng table này để tìm ra thủ phạm đã xoá hay sửa chữa data khi cần thiết.

Ðể tạo ra hay xem một trigger bằng Enterprise Manager bạn làm như sau: Right-Click lên table mà bạn muốn tạo trigger->All Tasks-> Manage Triggers.

Lưu ý: Ðôi Khi ta chỉ muốn trigger thực sự hoạt động khi một hay vài column nào đó được Update chứ không phải bất kỳ column nào. Khi đó ta có thể dùng hàm Update(Column\_Name) để kiểm tra xem column nào đó có bị update hay không.

**Ví dụ:**

Tạo một trigger cho Customer table. Bên trong Trigger (sau chữ AS) ta có thể kiểm tra xem nếu column First\_Name hay Last\_Name bị thay đổi thì mới hành động nếu không thì không làm gì cả

IF UPDATE (first\_name) OR UPDATE (Last\_Name)  
 BEGIN  
 Do some conditional processing

when either of these columns are updated.  
 END

Nếu muốn kiểm tra nhiều columns ta có thể dùng hàm khác là Columns\_Updated() . Xin xem thêm trong SQL Server Books Online để biết thêm chi tiết về cách sử dụng.

**Views**

Ðịnh nghĩa một cách đơn giản thì view trong SQL Server tương tự như Query trong Access database. View có thể được xem như một table ảo mà data của nó được select từ một stored query. Ðối với programmer thì view không khác chi so với table và có thể đặt ở vị trí của table trong các câu lệnh SQL. Ðặc điểm của View là ta có thể join data từ nhiều table và trả về một recordset đơn. Ngoài ra ta có thể "xào nấu" data (manipulate data) trước khi trả về cho user bằng cách dùng một số logic checking như (if, case...).

**Ví dụ:**

Create View OrderReport  
 As   
 Select OrderID,  
 (case when [Name] is null then 'New Customer'  
 else [Name]  
 end )As CustomerName,  
 ProductName,  
 DateProcessed  
 From Customers Right Outer Join Orders on

Customers.CustomerID=Orders.CustomerID

 Trong ví dụ trên ta chủ yếu trả về data từ Orders table trong PracticeDB nhưng thay vì display CustomerID vốn không có ý nhiều ý nghĩa đối với user ta sẽ display tên của customer bằng cách join với Customer table. Nếu Customer Name là Null nghĩa là tên của customer đã đặt order không tồn tại trong system. Thay vì để Null ta sẽ display "New Customer" để dễ nhìn hơn cho user.

Nói chung câu lệnh SQL trong View có thể từ rất đơn giản như select toàn bộ data từ một table cho đến rất phức tạp với nhiều tính năng programming của T-SQL.

**View Thường Ðược Dùng Vào Việc Gì?**

View thường được sử dùng vào một số công việc sau:

* Tập trung vào một số data nhất định : ta thường dùng view để select một số data mà user quan tâm hay chịu trách nhiệm và loại bỏ những data không cần thiết.   
    
  Ví dụ: Giả sử trong table ta có column "Deleted" với giá trị là True hay False để đánh dấu một record bị delete hay không. Việc này đôi khi được dùng cho việc Audit. Nghĩa là trong một ứng dụng nào đó khi user delete một record nào đó, thay vì ta physically delete record ta chỉ logically delete bằng cách đánh dấu record là đã được "Deleted" để đề phòng user yêu cầu roll back. Như vậy chủ yếu ta chỉ quan tâm đến data chưa delete còn data đã được đánh dấu deleted chỉ được để ý khi nào cần roll back hay audit mà thôi. Trong trường hợp này ta có thể tạo ra một view select data mà Deleted=False và làm việc chủ yếu trên view thay vì toàn bộ table.
* Ðơn giản hóa việc xử lý data: Ðôi khi ta có những query rất phức tạp và sử dụng thường xuyên ta có thể chuyển nó thành View và đối xử nó như một table, như vậy sẽ làm cho việc xử lý data dễ dàng hơn.
* Customize data: Ta có thể dùng view để làm cho users thấy data từ những góc độ khác nhau mặc dù họ đang dùng một nguồn data giống nhau. Ví dụ: Ta có thể tạo ra views trong đó những thông tin về customer được thể hiện khác nhau tùy login ID là normal user hay manager.
* Export và Import data: Ðôi khi ta muốn export data từ SQL Server sang các ứng dụng khác như Excel chẳng hạn ta có thể dùng view để join nhiều table và export dùng bcp.

Khi sử dụng view ta có thể select,insert, update, delete data bình thường như với một table.

**Ví dụ:**

Select \* From OrderReport  
 Where DateProcessed <'2003-01-01'