SQL Việt blog

Thảo luận về lập trình và quản trị Microsoft SQL Server

Sử Dụng Transaction Trong SQL Server

Vũ Huy Tâm

Transaction (giao dịch) được dùng để đảm bảo tính toàn vẹn dữ liệu khi xảy ra cập nhật (cập nhật xin được hiểu theo nghĩa rộng là các hành động sửa đổi dữ liệu, như INSERT, UPDATE, DELETE…). Khi một transaction bao gồm nhiều lệnh cập nhật, nó đảm bảo tất cả các cập nhật đều được thực hiện thành công, hoặc trong trường hợp một lệnh gặp sự cố toàn bộ transaction bị hủy bỏ. Khi đó dữ liệu trở về trạng thái như trước khi xảy ra transaction. Nói cách khác transaction ngăn chặn tình huống dữ liệu được cập nhật nửa chừng, trong đó một phần được cập nhật còn một phần bị bỏ qua.

Một ví dụ kinh điển về transaction là khi bạn cần thực hiện một giao dịch chuyển tiền giữa hai tài khoản ngân hàng. Giả sử bạn có hai tài khoản A và B với số tiền tương ứng là 8 tỷ và 1 tỷ; nay bạn cần chuyển bớt 2 tỷ từ tài khoản A sang tài khoản B. Sẽ có hai phép cập nhật như sau:

- trừ số tiền hiện có của tài khoản A đi 2 tỷ

- cộng thêm số tiền hiện có của tài khoản B lên 2 tỷ

Nếu hai lệnh cập nhật trên diễn ra độc lập (không nằm trong một transaction), và vì một lý do nào đó lệnh thứ hai bị lỗi, tài khoản A sẽ còn 6 tỷ và tài khoản B vẫn giữ nguyên 1 tỷ. Điều này không thể chấp nhận được vì 2 tỷ bỗng dưng biến mất! Khi thực hiện hai lệnh trên trong một transaction, nó sẽ đảm bảo:

- hoặc cả hai lệnh update đều được thực hiện thành công. Cả hai tài khoản được cập nhật với số tiền tương ứng.

- hoặc trong trường hợp giao dịch bị lỗi cả hai lệnh đều không được thực hiện. Hai tài khoản giữ nguyên số tiền như trước khi thực hiện transaction.

Trong SQL Server một transaction có đoạn code ở dạng đơn giản như sau (Bạn nên tham khảo mẫu code sử dụng transaction hoàn chỉnh ở phần dưới):

```
BEGIN TRAN
   -- lệnh 1
   -- lệnh 2
   -- ...
COMMIT
```

Bạn mở transaction bằng lệnh BEGIN TRAN và kết thúc bằng lệnh COMMIT – sau lệnh này những cập nhật dữ liệu sẽ được xác nhận vào trong database, transaction được đóng lại và các khóa (lock) trên các bảng được cập nhật được thả ra.

Về tùy chọn XACT_ABORT

Câu chuyện sẽ rất đơn giản nếu không có sự xuất hiện của tùy chọn XACT_ABORT. Đây là tùy chọn ở mức kết nối, chỉ có tác dụng trong phạm vi kết nối của bạn. XACT_ABORT nhận hai giá trị ON và OFF (OFF là giá trị mặc định). Khi tùy chọn này được đặt là OFF, SQL Server sẽ chỉ hủy bỏ lệnh gây ra lỗi trong transaction và vẫn cho các lệnh khác thực hiện tiếp, nếu lỗi xảy ra được đánh giá là không nghiêm trọng. Còn khi XACT_ABORT được đặt thành ON, SQL Server mới cư xử đúng như mong đợi – khi gặp bất kỳ lỗi nào nó hủy bỏ toàn bộ transaction và quay lui trở lại như lúc ban đầu. Ví dụ:

```
-- tạo bảng với ràng buộc cột i không được chứa giá trị 2
CREATE TABLE #t1(i INT, CONSTRAINT ck1 CHECK (i<>2) )

-- dùng giá trị mặc định XACT_ABORT = OFF
-- SET XACT_ABORT OFF

BEGIN TRAN
   INSERT #t1 SELECT 1
   INSERT #t1 SELECT 2 -- vi phạm ràng buộc
   INSERT #t1 SELECT 3
COMMIT

SELECT * FROM #t1
```

```
i
---
1
3
(2 ROW(s) affected)
```

Như vậy trong transaction trên, lệnh insert thứ hai gây ra lỗi nhưng lệnh thứ ba vẫn tiếp tục được thực hiện, và transaction vẫn kết thúc thành công. Kết quả là bảng vẫn có hai bản ghi từ lệnh insert thứ nhất và thứ ba.
Nay hãy đặt XACT_ABORT thành ON:

```
-- tạo bảng với ràng buộc cột i không được chứa giá trị 2
CREATE TABLE #t2(i INT, CONSTRAINT ck2 CHECK (i<>2) )

SET XACT_ABORT ON

BEGIN TRAN
   INSERT #t2 SELECT 1
   INSERT #t2 SELECT 2 -- vi phạm ràng buộc
   INSERT #t2 SELECT 3
COMMIT

SELECT * FROM #t2

i
---
(0 ROW(s) affected)
```

Và bây giờ bảng không có bản ghi nào vì toàn bộ transaction đã bị hủy bỏ. Chính xác ra là các lệnh phía sau lệnh gây ra lỗi không được thực hiện tiếp, còn các lệnh thực hiện trước đó bị quay lui (ROLLBACK) trở lại.
Nói chung SET XACT_ABORT ON tránh được rất nhiều rắc rối khi dùng transaction, nó xử lý gọn ghẽ các ngoại lệ kể cả các lỗi như connection timeout hay khi user hủy bỏ thực hiện. Bản thân tôi không lý giải được tại sao Microsoft không đặt giá trị mặc định cho XACT_ABORT là ON. Thậm chí không lý giải được việc đưa ra tùy chọn này để làm gì. Tóm lại bạn luôn nên đặt SET XACT_ABORT ON vào đầu thủ tục nếu cần dùng transaction.

Mẫu code sử dụng transaction
Từ SQL Server bản 2005 trở lên bạn có thể dùng đoạn code sau:

```
SET XACT_ABORT ON
BEGIN TRAN
BEGIN TRY
   -- lệnh 1
   -- lệnh 2
  -- ...
COMMIT
END TRY
BEGIN CATCH
   ROLLBACK
   DECLARE @ErrorMessage VARCHAR(2000)
   SELECT @ErrorMessage = 'Lỗi: ' + ERROR_MESSAGE()
   RAISERROR(@ErrorMessage, 16, 1)
END CATCH
```

Đoạn lệnh trên kết hợp transaction với xử lý lỗi.
- Nó bắt đầu bằng việc đặt lựa chọn XACT_ABORT là ON để đảm bảo transaction hoạt động đúng như mong muốn.
- Sau đó là BEGIN TRAN để mở transaction.
- Tiếp đến là BEGIN TRY để mở ra khối try block (giống như try block trong C#)
- Khối try block sẽ chứa các lệnh cần thực hiện trong transaction
- Rồi đến COMMIT để kết thúc transaction và END TRY để kết thúc khối try block
- Sau đó là BEGIN CATCH (giống như catch block trong C#). Đây là phần chứa đoạn lệnh sẽ được thực hiện khi có lỗi trong phần try block.
- Trong phần catch lệnh đầu tiên là ROLLBACK để quay lui transaction.
- Sau đó dùng một biến để chứa thông báo lỗi. Bạn cũng có thể thêm các bước như lưu thông tin về lỗi vào một bảng audit, hoặc gửi email cho DBA…
- Kết thúc là RAISERROR để báo cho ứng dụng biết thủ tục đã gây ra lỗi và truyền thông báo lỗi cho ứng dụng.

Phiên bản áp dụng: SQL Server 2005 trở lên

Tags: BEGIN CATCH, BEGIN TRAN, BEGIN TRY, COMMIT, END CATCH, END TRY, ERROR_MESSAGE(), RAISERROR, ROLLBACK, transaction, XACT_ABORT

12 Comments
Posted on 19/2/2011 | Categories: SQL Server Programming

Các bài viết tương tự

- Các Mức Isolation Level
- Di Chuyển Dữ Liệu Qua Bảng Khác
- Làm Sao Giảm Bớt Dung Lượng File LOG Của SQL Server

Comments

- *DamVu* (19/02/2011 7:17 am)

  Một bài viết hay, cảm ơn anh đã chia sẻ

  Reply

- *Red Devilic* (20/02/2011 7:29 pm)

  Tùy chọn XACT_ABORT thực ra có rất có ý nghĩa anh Tâm ah.

  Ví dụ 1 trường hợp cần insert, hoặc load từ file, v.v…. rất lớn vào 1 bảng. Như kiểu BULK INSERT vậy. Nếu như chỉ có 1,2 dòng bị lỗi, hoặc 1 số rất ít bị lỗi mà các dòng còn lại vẫn INSERT được bình thường thì sẽ rất phí phạm nếu như phải ROLLBACK lại toàn bộ.

  Tuy nhiên có vẻ các Developer của MS cũng hơi vô lý khi để giá trị default là OFF . Là ON thì phù hợp với mục đích sử dụng chung hơn.

  Reply

- *Vũ Huy Tâm* (22/02/2011 9:17 am)

  Chào bác Red Devilic,

  bác có thể cho ví dụ về dùng XACT_ABORT để giúp load file được không? Theo tôi hiểu thì BULK INSERT là một transaction, trừ khi bác đặt batch size thì nó sẽ tách thành nhiều transaction và commit theo từng batch. Còn nếu không thì nó commit hoặc rollback toàn bộ (giống như 1 lệnh UPDATE) không phụ thuộc vào giá trị của XACT_ABORT

  Reply

- *Tony* (06/07/2011 6:00 am)

  MÌNH test trên sql 2005 bị lỗi này

  [CODE]
  CREATE TABLE #t2(i INT, CONSTRAINT ck2 CHECK (i2))

  SET XACT_ABORT ON
  BEGIN TRAN
  BEGIN TRY
  INSERT #t2 SELECT 1
  INSERT #t2 SELECT 2 — vi phạm ràng buộc
  INSERT #t2 SELECT 3
  COMMIT
  END TRY
  BEGIN CATCH
  ROLLBACK
  DECLARE @ErrorMessage VARCHAR(2000)
  SELECT @ErrorMessage = 'Lỗi: ' + ERROR_MESSAGE()
  RAISERROR(@ErrorMessage, 16, 1)

END
[/CODE]

Msg 102, Level 15, State 1, Line 16
Incorrect syntax near 'END'.

Reply

- *Vũ Huy Tâm* (06/07/2011 10:37 am)

hì hì, phải là END CATCH mới đúng. Cám ơn Tony, tôi đã sửa lại trong bài rồi

Reply

- *Neo An* (25/06/2012 3:27 am)

Thanks for sharing, man

Reply

- *Pe sua* (09/07/2012 4:17 am)

Bạn ơi cho mình hỏi với, mình viết đoạn lệnh sau trên nền ứng dụng Visual Fox,nhưng không thấy có hiệu quả(khi có lỗi không thông báo), Lỗi ở đây là mình cố tính cho dữ liệu trường text có chưa kí tự " ' " (dấu nháy đơn),nhưng vẫn lưu được bình thường ko báo lỗi nhưng thực ra là Insert không thành công vì trường dữ liệu có chứa kí tự Khóa của SQL
Đoạn lệnh như sau

SQLEXEC(p_gnConnHandle,"SET XACT_ABORT ON")— Lệnh SQLEXEC là lệnh của VS FOX để làm việc với database
SQLEXEC(p_gnConnHandle,"BEGIN TRANSACTION")—-Gửi lệnh này SQL
SQLEXEC(p_gnConnHandle,"BEGIN TRY")

Các lệnh quét các dòng trong bảng của Fox và insert len table trong Database

SQLEXEC(p_gnConnHandle,"COMMIT TRANSACTION")
SQLEXEC(p_gnConnHandle,"END TRY")
SQLEXEC(p_gnConnHandle,"BEGIN CATCH")
SQLEXEC(p_gnConnHandle,"ROLLBACK")
SQLEXEC(p_gnConnHandle,"DECLARE @ErrorMessage CHAR(2000)")
SQLEXEC(p_gnConnHandle,"SELECT @ErrorMessage = 'Lçi: '+ ERROR_MESSAGE()")
SQLEXEC(p_gnConnHandle,"RAISERROR(@ErrorMessage, 16, 1)")
SQLEXEC(p_gnConnHandle,"END CATCH")

Đây là đoạn lệnh trích dẫn thôi
Mong bạn chỉ giùm mình làm sao để dùng được Transaction trong trường hợp này,nếu có thể cho mình xin Nick YM để trình bày rõ hơn,Thanks!

Reply

- *ecopmc* (01/11/2012 5:01 am)

```
BEGIN TRAN
BEGIN TRY
   -- lệnh 1
   -- lệnh 2
  -- ...
COMMIT
END TRY
BEGIN CATCH
   ROLLBACK TRAN
   DECLARE @ErrorMessage VARCHAR(2000)
   SELECT @ErrorMessage = 'Lỗi: ' + ERROR_MESSAGE()
   RAISERROR(@ErrorMessage, 16, 1)
END CATCH
```

chỉ cần cấu trúc thế này thôi là đc rồi, ko cần quan tâm set xact_abort là ON hay OFF
vì nếu có lỗi thì Rollback rồi

nên dùng set xact_abort ON trong 1 cấu trúc đơn giản khi người dùng ko muốn bắt try catch giống thế này

```
SET XACT_ABORT ON

BEGIN TRAN
   INSERT #t2 SELECT 1
   INSERT #t2 SELECT 2 -- vi phạm ràng buộc
   INSERT #t2 SELECT 3
COMMIT
```

Reply

- *Vũ Huy Tâm* (01/11/2012 9:20 am)

  Vấn đề là TRY CATCH không bắt được một số loại lỗi, như lỗi timeout hoặc tên bảng/tên cột không tồn tại. Khi gặp những lỗi này chương trình sẽ gục ngã ngay tại đó và không nhảy được tới phần BEGIN CATCH để mà rollback. Nếu xact_abort là OFF, các lệnh chạy trước lệnh chứa lỗi vẫn thực hiện thành công. Chỉ khi xact_abort là ON thì toàn bộ các lệnh mới được tự động rollback. Bạn có thể thử đoạn lệnh sau với hai lựa chọn xact_abort ON và OFF:

  ```
  CREATE TABLE #t2(i INT, CONSTRAINT ck2 CHECK (i<>2) )
  SET XACT_ABORT OFF
  -- SET XACT_ABORT ON
  BEGIN TRAN
  BEGIN TRY
     INSERT #t2 SELECT 1
     INSERT #t2 SELECT 3
     INSERT #t3 SELECT 100 -- bảng #t3 không tồn tại
  COMMIT
  END TRY
  BEGIN CATCH
     ROLLBACK TRAN
     DECLARE @ErrorMessage VARCHAR(2000)
     SELECT @ErrorMessage = 'Lỗi: ' + ERROR_MESSAGE()
     RAISERROR(@ErrorMessage, 16, 1)
  END CATCH
  SELECT * FROM #t2
  ```

  Reply

- *ecopmc* (01/11/2012 9:59 pm)

  Thanks bạn rất nhiều, đúng là như bạn nói. Try catch ko bắt đc những lỗi nhu vậy
  ^^

  Reply

- *ecopmc* (02/11/2012 12:16 am)

  ah ah, đóng góp ý kiến thêm xíu nữa,

  đoạn này nếu chạy trên SQL SERVER 2008 R2 thì default XactAbort luôn ON nha.

  ```
  CREATE TABLE #t2(i INT, CONSTRAINT ck2 CHECK (i2) )
  --Nghĩa là ở đây không cần quan tâm tới XactAbort là ON hay OFF. Các bản SQL khác thì mình không dám chắc
  BEGIN TRAN
  BEGIN TRY
     INSERT #t2 SELECT 1
     INSERT #t2 SELECT 3
     INSERT #t3 SELECT 100 -- bảng #t3 không tồn tại
  COMMIT
  END TRY
  BEGIN CATCH
     ROLLBACK TRAN
     DECLARE @ErrorMessage VARCHAR(2000)
     SELECT @ErrorMessage = 'Lỗi: ' + ERROR_MESSAGE()
     RAISERROR(@ErrorMessage, 16, 1)
  END CATCH
  SELECT * FROM #t2
  ```

  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

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

Giới Thiệu

Tip & Trick

Trang Hỏi - Đáp

Seminar Presentation: Index trong SQL Server - Download

Search

PHẢN HỒI MỚI

- Vth90 on Hỏi – Đáp
- NaCl on Hỏi – Đáp
- Vth90 on Hỏi – Đáp
- Vth90 on Hỏi – Đáp
- Vũ Huy Tâm on Một Ví Dụ Sao Lưu/Khôi Phục Dữ Liệu
- Vũ Huy Tâm on Table Partitioning – Các Khái Niệm Cơ Bản
- Vũ Huy Tâm on Làm Sao Giảm Bớt Dung Lượng File LOG Của SQL Server
- Omni Nguyen on Làm Sao Giảm Bớt Dung Lượng File LOG Của SQL Server
- huu vinh on Table Partitioning – Các Khái Niệm Cơ Bản
- Omni Nguyen on Một Ví Dụ Sao Lưu/Khôi Phục Dữ Liệu
- Omni Nguyen on Một Ví Dụ Sao Lưu/Khôi Phục Dữ Liệu
- NaCl on Tìm Kiếm Với Nhiều Tham Số
- Quân on Hỏi – Đáp
- Vth90 on Hỏi – Đáp
- nguoihanoi on Hỏi – Đáp
- Vth90 on Hỏi – Đáp
- Vth90 on Hỏi – Đáp
- nguoihanoi on Hỏi – Đáp
- Vth90 on Hỏi – Đáp
- Vth90 on Hỏi – Đáp

BÀI MỚI

- SQL Saturday
- Thứ Tự Xử Lý Điều Kiện Ở Mệnh Đề WHERE
- Các Mức Isolation Level
- Một Số Khái Niệm Về Bảo Mật Trong SQL Server
- Filegroup Trong SQL Server
- Lựa Chọn Khóa Chính: Khóa Đại Diện vs. Khóa Tự Nhiên
- Compress Backup
- Import File vào Database Dùng Integration Service
- Di Chuyển Dữ Liệu Qua Bảng Khác
- Tìm Xem File Thuộc Database Nào
- Xem Thông Tin File Backup
- Tạo Linked Server
- Tránh Dùng COUNT(DISTINCT)
- Lấy Ngày Cuối Của Tháng
- Các Chế Độ Phục Hồi Của Database
- Cione
- Các Loại Ràng Buộc Trong SQL Server
- Tìm Xem Cột Nằm Trong Bảng Nào
- Table Partitioning – Split và Merge
- Tìm Thời Điểm SQL Server Start Gần Đây Nhất

Xem toàn bộ các bài post...

CHỦ ĐỀ

- Bảo mật (2)
- Bên trong SQL Server (1)
- Database Administration (19)
- Download (2)
- Eventual Consistency (2)
- Function (2)
- IDENTITY (2)
- Index (7)
- Linh tinh (6)
- Performance tuning (10)
- SQL Server Programming (27)
- Sql động (4)
- SSIS (1)
- Stored Procedure (2)
- Table partitioning (4)
- Thiết kế database (10)
- Tip & Trick (7)
- View (1)

NHÃN

ANSI JOIN 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 Seek INFORMATION_SCHEMA INNER JOIN INSERT ISNULL JOIN Key Lookup LEFT LEFT JOIN LIKE % Log file MySpace Nested Loop Join NEWID() Nhất 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 RIGHT JOIN 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

Blog Việt

- ddth.com (SQL Server forum)
- cione – Đào tạo trực tuyến
- Buu Nguyen blog
- sqlvn.com

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