How should I go about implementing an "autonumber" field in SQL Server 2005?

Asked 15 years, 9 months ago Modified 15 years, 9 months ago Viewed 1k times

3

I'm aware of IDENTITY fields but I have a feeling that I couldn't use one to solve my problem.

Let's say I have multiple clients. Each client has multiple orders. Each client needs to have their orders numbered sequentially, specific to them.

Example table structure:

```
Orders:
OrderID | ClientID | ClientOrderID | etc...
```

Some example rows for this table would be:

```
OrderID | ClientID | ClientOrderID | etc...
1 | 1 | 1 | ...
2 | 1 | 2 | ...
3 | 2 | 1 | ...
4 | 3 | 1 | ...
5 | 1 | 3 | ...
6 | 2 | 2 | ...
```

I know the naive way would be to take the MAX ClientOrderID for any client and use that value for INSERTs but that would be subject to concurrency issues. I was considering using a transaction but I'm not quite sure what the broadest isolation scope that can be used for this. I'll be using LINQ to SQL but I have feeling that isn't relevant.

sql-server-2005 transactions identity auto-increment transaction-isolation

Share Improve this question Follow

asked Mar 23, 2009 at 21:54

llamaoo7

4,331 ● 2 ● 24 ● 22

So, I have to ask...WHY do you want to do this? – Micky McQuade Mar 23, 2009 at 21:56

This feels like someone not accustomed to data storage is asking you to do this. Create dates are a more natural way of doing this. – Nick DeVore Mar 23, 2009 at 22:05

I could probably just assign a time stamp to each order and then doing a COUNT to get the order # for the client. This shouldn't be a performance problem as the table won't get much larger than 1000 records. – llamaoo7 Mar 23, 2009 at 22:16

While ordering by date or even the orderID column will give you the same results, the whole sequence numbering system will fall apart when somebody decides to delete an order ... and they will, even if it's not in the spec. I think he's doing the right thing by giving the client what they asked for. – John MacIntyre Mar 23, 2009 at 22:24

4 Answers

Sorted by: Highest score (default)

2

Somebody correct me if I'm wrong, but as long as your MAX() call is in the same step as your insert, you won't have a problem with concurrency.

So, you could **not** do

```
select @newOrderID=max(ClientOrderID) + 1
from orders
where clientid=@myClientID;

insert into ( ClientID, ClientOrderID, ...)
values( @myClientID, @newOrderID, ...);
```

But you **can** do

```
insert into ( ClientID, ClientOrderID, ...)
select @myClientID, max(ClientOrderID) + 1, ...
from orders
where clientid=@myClientID;
```

I'm assuming OrderID is an identity column.

Again, if I'm incorrect on this, please let me know. Preferably with a URL

Share

Improve this answer

Follow

edited Mar 23, 2009 at 22:25

answered Mar 23, 2009 at 22:18

John MacIntyre

13k ● 13 ● 69 ● 107

1

You could use a Repository pattern to handle your Orders and let it control the number of each specific clients order number. If you implement the OrderRepository correctly it could control the concurrency and number the order before saving it to the database (let the repository and not the db set the number).

Repository pattern: http://martinfowler.com/eaaCatalog/repository.html

Share Improve this answer Follow

answered Mar 23, 2009 at 22:01

Per Hornshøj-Schierbeck

15.2k ● 21 ● 83 ● 103

1

One possibility (though I don't like to do this) is to have a lookup table that would tell you the greatest Order Number given for each vendor. Inside of a transaction, you'd fetch the most recent one from VendorOrderNumber, save your new order, increment the value in VendorOrderNumber, commit transaction.

Share Improve this answer Follow

answered Mar 23, 2009 at 22:03

Nick DeVore

10.2k ● 3 ● 40 ● 41

1

This is an odd way to store data, but assuming you need it, there is nothing built-in that you can use.

Your suggestion of Max(ClientOrderID) is straight forward and pretty easy to implement (follow John MacIntyre's advice). It will probably work acceptably well on tables with a few thousand orders. As the table grows this approach will of course slow down.

Nick DeVore's suggestion of a lookup table is a little messier to implement but won't substantially be affected by data growth.

Depending on where/when you actually need the ClientOrderID, you could calculate the id when needed like this:

```
SELECT *,
ROW_NUMBER() OVER(ORDER BY OrderID) AS ClientOrderID
FROM Orders
WHERE ClientID = 1
```

This assumes that the ClientOrderIDs are in the same sequence as the OrderID. Without actually persisting the ID, it is awkward to use as a key to anything else. This approach should not be affected by data growth.

Share Improve this answer Follow

answered Mar 23, 2009 at 22:33

ScottS

8,543 ● 5 ● 31 ● 50