

BÀI HỌC KINH NGHIỆM

**Lỗi không cất được chứng từ sau khi sửa
tại phân hệ mua hàng**

Trình bày: Đặng Việt Anh

BÀI HỌC KINH NGHIỆM

Ch...
Mua hàng trong nước nhập
!
Cất không thành công.
Hướng dẫn
?
X

Phiếu nhập
Hóa đơn

Mã nhà cung cấp
NCC0000116

Tên nhà cung cấp
Công ty CP Tmdv Misa 072

Ngày hạch toán
18/08/2023 10:34:48

Tổng tiền thanh toán
576.400.000

Người giao hàng

Địa chỉ
72 Xuân Đình, Btl, Hn

Ngày chứng từ
18/08/2023

Nhân viên mua hàng

Diễn giải
Mua hàng của Công ty CP Tmdv Misa 072

Số phiếu nhập
NK00071

Kèm theo
Số lượng

Chứng từ gốc

Tham chiếu hóa đơn
00000571

Tham chiếu chứng từ

Điều khoản thanh toán

Số ngày được nợ

Hạn thanh toán
DD/MM/YYYY

Hàng tiền
Chi phí

Loại tiền
USD

Tỷ giá
20.960,00

Chiết khấu
Theo mặt hàng

#	Mã hàng	Công nợ	Quy cách	ĐVT	Số lượng	Đơn giá	Thành tiền	Th
1	VTHEH46	1		Thùng	5,00	10.000,00	50.000,00	
					5,00		50.000,00	

Tổng số: 1 bản ghi
20 bản ghi trên 1 trang
Trước
1
Sau

Thêm dòng
Thêm ghi chú
Xóa hết dòng

Tổng tiền hàng
50.000,00

1.048.000.000

Tiền chiết khấu
25.000,00

524.000,00

Biểu hiện lỗi

BÀI HỌC KINH NGHIỆM

Nguyên nhân gây ra lỗi

Trong phiên bản R44 có sửa lỗi hotfix liên quan tới tính năng xem hóa đơn CQT đồng bộ từ tool khi nhấn vào drilldown số hóa đơn trên các chứng từ mua hàng, dịch vụ, trả lại hàng bán, chứng từ nghiệp vụ khác, phiếu chi, ủy nhiệm chi (lập kèm hóa đơn)

Để xem được hóa đơn đồng bộ từ tool thì phải bao gồm mẫu số, ký hiệu, số hóa đơn, ngày hóa đơn -> những thông tin này trong drilldown chưa đủ

<pre>Links: [{ Type: 'pu_invoice', RefType: MSEnum.RefType.PUInvoiceVoucherService, Model: puInvoiceModel, UseSameKeyWithMaster: false, KeyColumnInMaster: 'pu_invoice_refid' }, { Type: 'inv_bot_reference', Model: InvBotReference }], IsDraft: false, // CT lưu nhập ;</pre>	<pre>Links: [{ Type: 'pu_invoice', RefType: MSEnum.RefType.PUInvoiceVoucherService, Model: puInvoiceModel, UseSameKeyWithMaster: false, KeyColumnInMaster: 'pu_invoice_refid' }, { Type: 'inv_bot_reference', Model: InvBotReference, View: 'view_inv_bot_reference' }], IsDraft: false, // CT lưu nhập ;;</pre>
---	--

Vì không đủ thông tin nên phần links đã sửa lấy dữ liệu inv_bot_reference thông qua view

BÀI HỌC KINH NGHIỆM

Nguyên nhân gây ra lỗi

Name	×	Headers	Payload	Preview	Response	Initiator	Timing
<input type="checkbox"/> g...				▼ {Success: false, Code: 99, SubCode: 0,...}			
<input type="checkbox"/> g...				Code: 99			
<input type="checkbox"/> s...				ErrorMessage: []			
				RequestTime: "00:00:11.1636300"			
				ServerTime: "2023-11-20T10:23:47.2012018+07:00"			
				SubCode: 0			
				Success: false			
				SystemMessage: "Microsoft.EntityFrameworkCore.DbUpdateConcurrencyException: Database operation expected to affect 1 row(s) but actually affected 0 row(s). Error: The database is locked."			
				TotalTime: 0			

Sau khi không cất được thì kiểm tra xem response từ server trả về như trên
(**Database operation expected to affect 1 row(s) but actually affected 0 row(s) error**)

BÀI HỌC KINH NGHIỆM

Debug thì thấy lỗi trong SaveMasterDetail khi thực hiện lưu dữ liệu dữ vào DB

```
/// Created by DVMang - 08.02.2019
2 references | 0 changes | 0 authors, 0 changes
public async Task<ServiceResponse> SaveMasterDetail(IDbConnection cnn, DatabaseType dbType, List<MasterDetailRequest> request)
{
    ServiceResponse response = new ServiceResponse();
    List<BaseMasterModel> refreshMasters = new List<BaseMasterModel>();
    List<BaseModel> refreshDetails = new List<BaseModel>();
    dynamic result = null;
    IDbContextTransaction tran = null;
    try
    {
        //Do insert
        using (var db = _db.GetContext<TContext>(dbType, DatabaseSide.WriteSide))
        {
            var transaction = db.Database.BeginTransaction();
```

2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108

```
break;
case ModelState.Update:
    db.Entry(LinkObj).State = EntityState.Modified;
    _esTracker.AddModified(LinkObj.GetTableNameOnly());
    break;
case Mode
db.En
_esTr
break;
}
}
//await db.SaveCh
}
}
//custom thêm dữ liệu submit
await this.CustomActionBefore
//Lưu dữ liệu
await db.SaveChangesAsync();
transaction.Commit();
result = _db.BuildSubmitResult(cnn, refreshMasters.AsList<BaseModel>(), refreshDetails, dbType);
```

Exception Thrown

Microsoft.EntityFrameworkCore.DbUpdateConcurrencyException: 'Database operation expected to affect 1 row(s) but actually affected 0 row(s). Data may have been modified or deleted since entities were loaded. See <http://go.microsoft.com/fwlink/?LinkId=527962> for information on understanding and handling optimistic concurrency exceptions.'

This exception was originally thrown at this call stack:
[External Code]

Show Call Stack | View Details | Copy Details | Start Live Share session...

Exception Settings

- ☒ Break when this exception type is thrown
- Except when thrown from:
 - ☐ MISA.SME.Core.BLCommon.dll

Open Exception Settings | Edit Conditions

BÀI HỌC KINH NGHIỆM

Lỗi **“Database operation expected to affect 1 row(s) but actually affected 0 row(s)”** thường xuất hiện khi cố gắng cập nhật hoặc xóa một bản ghi trong cơ sở dữ liệu, nhưng bản ghi đó không tồn tại hoặc đã bị thay đổi kể từ khi cập nhật. Điều này thường xảy ra khi có nhiều người dùng hoặc tiến trình đang cùng lúc thao tác với cùng một bản ghi.

Lỗi trên thì có reference tới link help của MS:
See <http://go.microsoft.com/fwlink/?LinkId=527962> for information on understanding and handling optimistic concurrency exceptions.

=> Sau khi xem thì đây là một dạng lỗi về đồng bộ hóa (concurrency), và có thể xử lý nó bằng cách sử dụng các kỹ thuật như optimistic concurrency hoặc pessimistic concurrency

Optimistic concurrency

EF Core implements *optimistic concurrency*, which assumes that concurrency conflicts are relatively rare. In contrast to *pessimistic* approaches - which lock data up-front and only then proceed to modify it - optimistic concurrency takes no locks, but arranges for the data modification to fail on save if the data has changed since it was queried. This concurrency failure is reported to the application, which deals with it accordingly, possibly by retrying the entire operation on the new data.

In EF Core, optimistic concurrency is implemented by configuring a property as a *concurrency token*. The concurrency token is loaded and tracked when an entity is queried - just like any other property. Then, when an update or delete operation is performed during `SaveChanges()`, the value of the concurrency token on the database is compared against the original value read by EF Core.

To understand how this works, let's assume we're on SQL Server, and define a typical Person entity type with a special `Version` property:

```
C# Copy
public class Person
{
    public int PersonId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }

    [Timestamp]
    public byte[] Version { get; set; }
}
```

In SQL Server, this configures a concurrency token that automatically changes in the database every time the row is changed (more details are available below). With this configuration in place, let's examine what happens with a simple update operation:

```
C# Copy
var person = context.People.Single(b => b.FirstName == "John");
person.FirstName = "Paul";
context.SaveChanges();
```

BÀI HỌC KINH NGHIỆM

Review lại code

```
2024 RefreshDetails.Add(obj);  
2025 }  
2026 }  
2027 else if (obj.state == ModelState.Update)  
2028 {  
2029     db.Entry(obj).State = EntityState.Modified; ≤ 1ms elapsed  
2030     _esTracker.AddModified(obj.GetTableNameOnly());  
2031 }  
2032 }  
2033  
2034 if (details?.Count > 0)
```

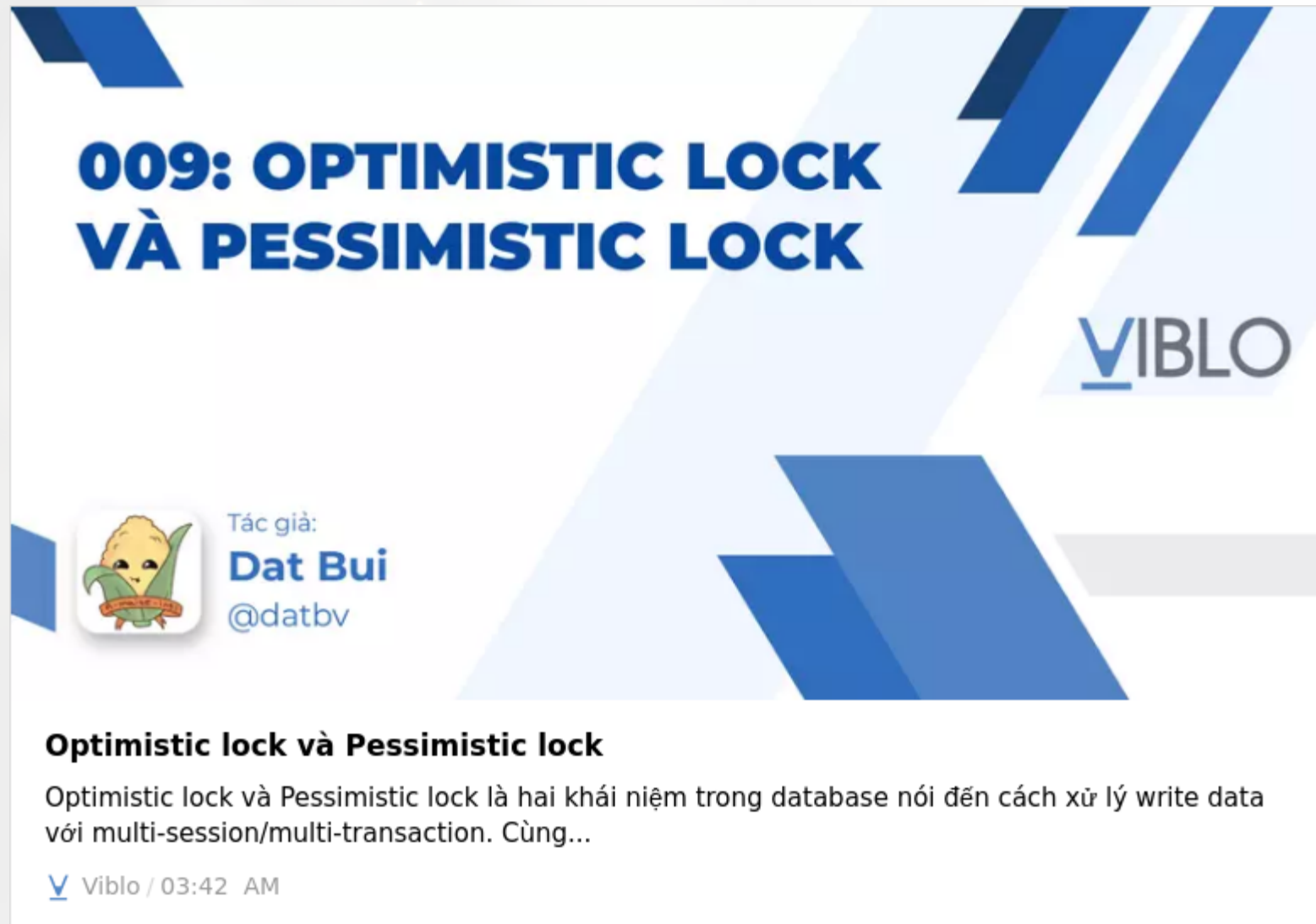
=> Lệnh `_dbContext.Entry(entity).State = EntityState.Modified;` trong Entity Framework sử dụng cơ chế Optimistic Locking

Để thực hiện được thì sẽ phải Fetch data kèm theo `edit_version` hiện tại


Kiểm tra view thì không thấy lấy ra cột này, bổ sung lấy ra thêm cột `xmin` as `edit_version` (trong postgresql để quản lý transaction của 1 row dựa vào `xmin`)

Any Question?

Link tham khảo



009: OPTIMISTIC LOCK VÀ PESSIMISTIC LOCK

 Tác giả: **Dat Bui**
@datbv

Optimistic lock và Pessimistic lock

Optimistic lock và Pessimistic lock là hai khái niệm trong database nói đến cách xử lý write data với multi-session/multi-transaction. Cùng...

[V](#) Viblo / 03:42 AM



Microsoft Learn

Handling Concurrency Conflicts - EF Core

Managing conflicts when the same data is updated concurrently with Entity Framework Core

 MicrosoftLearn /