**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**Khoa Công Nghệ Thông Tin**

**BÁO CÁO CUỐI KỲ**

**RESTAURANT SYSTEM**

**Giảng viên hướng dẫn:**

***Hoàng Anh Tú***

**Các thành viên trong nhóm 40:**

***Nguyễn Bá Ngọc – MSSV : 1141085***

***Bùi Trung Tân – MSSV :*** ***1141129***

***Nguyễn Minh Nghĩa – MSSV : 1141083***

***Nguyễn Minh Vũ – MSSV : 1141187***

## Phụ Lục

[1. Thông tin nhóm 1](#_Toc325069036)

[1.1 Thông tin thành viên: 1](#_Toc325069037)

[2. Các tranh chấp đồng thời đã phát hiện 1](#_Toc325069038)

[2.1 1141085 – Nguyễn Bá Ngọc 1](#_Toc325069039)

[2.1.1 Các trường hợp tranh chấp 1](#_Toc325069040)

[2.1.2 Xử lý các tranh chấp 2](#_Toc325069041)

[2.2 1141187 – Nguyễn Minh Vũ 2](#_Toc325069042)

[2.2.1 Các trường hợp tranh chấp 2](#_Toc325069043)

[2.2.2 Xử lý các tranh chấp 3](#_Toc325069044)

[2.3 1141083 – Nguyễn Minh Nghĩa 3](#_Toc325069045)

[2.3.1 Các trường hợp tranh chấp 3](#_Toc325069046)

[2.3.2 Xử lý các tranh chấp 3](#_Toc325069047)

[2.4 1141129 – Bùi Trung Tân 4](#_Toc325069048)

[2.4.1 Các trường hợp tranh chấp 4](#_Toc325069049)

[2.4.2 Xử lý các tranh chấp 4](#_Toc325069050)

# Thông tin nhóm

## Thông tin thành viên:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | MSSV | Họ tên | Nhóm trưởng | Đánh giá |
| 1 | 1141085 | Nguyễn Bá Ngọc | X | 100% |
| 2 | 1141187 | Nguyễn Minh Vũ |  |  |
| 3 | 1141083 | Nguyễn Minh Nghĩa |  |  |
| 4 | 1141129 | Bùi Trung Tân |  |  |

# Các tranh chấp đồng thời đã phát hiện

## 1141187 – Nguyễn Minh Vũ

### Các trường hợp tranh chấp

#### Drity read

##### Trường hợp 1 :

###### Kịch bản :

|  |  |
| --- | --- |
| T1 | T2 |
| - Thêm vào một nguyên liệu mới |  |
|  | **-** Lấy danh sách nguyên liệu để hiện thị lên màn hình |
| -Kiểm tra tên nguyên liệu mới có là duy nhất hay không, nếu có 2 nguyên liệu trùng tên và đơn vị  =>Hủy bỏ việc thêm nguyên liệu này |  |

* Giao tác 2 sẽ đọc dử liệu mà Giao tác T1 vừa thêm vào , nhưng sau đó Giao tác T1 hủy việc thêm vào này nên Giao tác T2 đọc dử liệu sai

###### Store procedure :

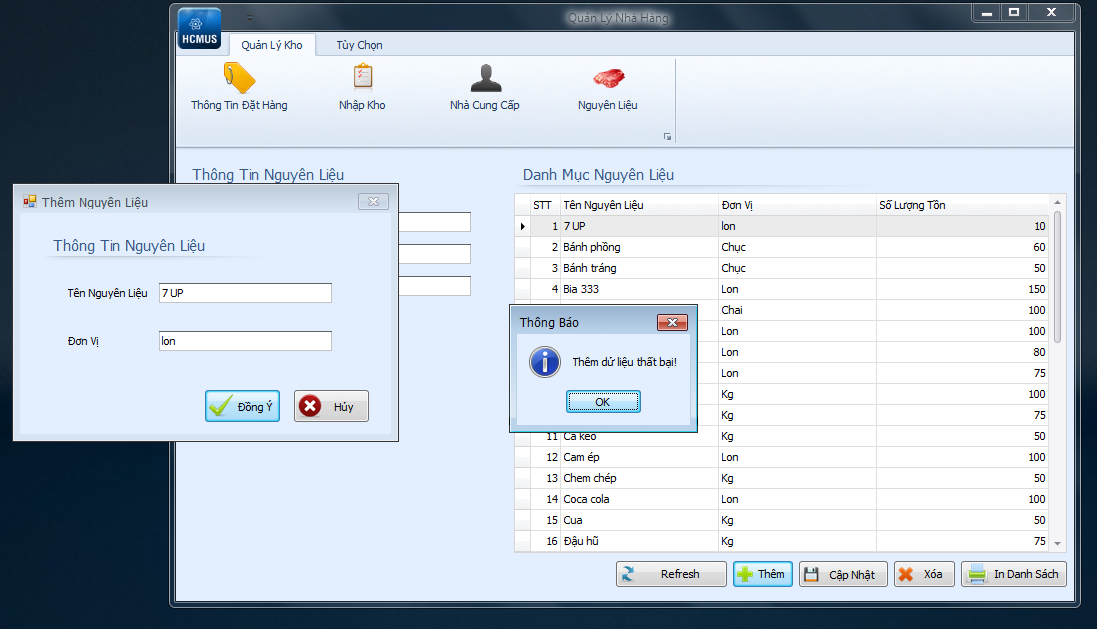
|  |  |
| --- | --- |
| T1 | T2 |
| create proc InsertNguyenLieu  @Flag int out,  @MaNH nchar(10),  @TenNL nvarchar(50),  @DonVi nvarchar(20),  @SoLuongTon int  As  set @Flag = 0  begin tran  set transaction isolation level  read uncommitted  insert into NguyenLieu  values  (  @MaNH  ,@TenNL  ,@DonVi  ,@SoLuongTon  )  waitfor delay '00:00:05' |  |
|  | create proc SelectNguyenLieu  @MaNH nchar(10)  as  begin  begin tran  set transaction isolation level  read uncommitted  select \*  from NguyenLieu nl  where nl.MaNH = @MaNH  order by  nl.TenNL    commit tran  end |
| if ((select count(\*)  from NguyenLieu  where TenNL = @TenNL  and DonVi = @DonVi  and MaNH = @MaNH)<>1)  begin  rollback tran  return  end    if (@@error<>0)  begin  rollback tran  return  end    set @Flag = 1  commit  end |  |

###### Nguyên nhân tranh chấp:

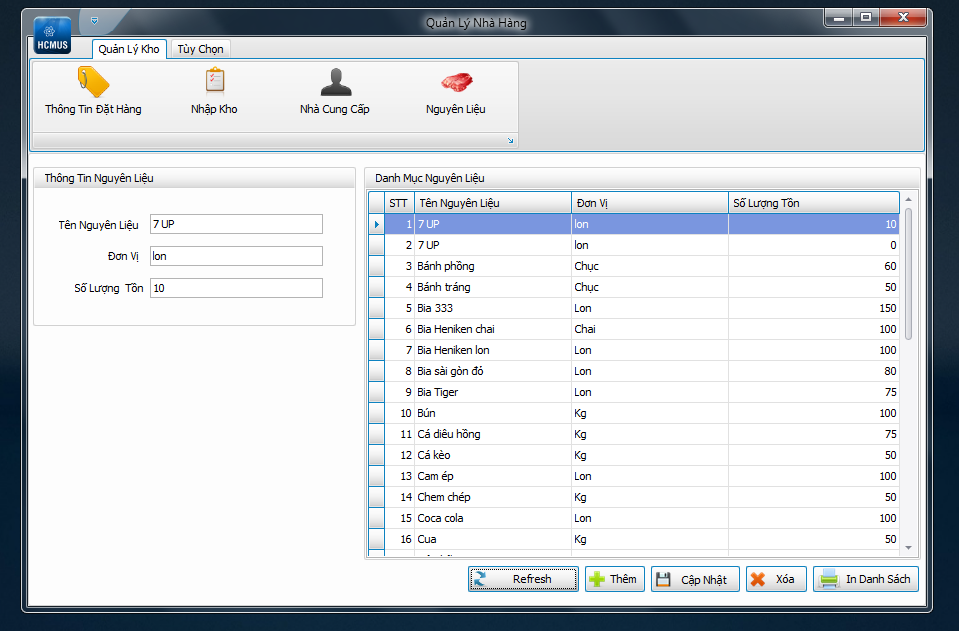
Với mức cô lập **read uncommitted** Giao tác T2 không xin khóa đọc khi lấy dử liệu, vì vậy khi T1 đang ghi dử liệu, nó vẫn có thể đọc dử liệu, dẫn đến tranh chấp

###### Giao diện chương trình:

Giao tác T1:



Giao tác T2:



Xuất hiện 2 nguyên liệu “7 UP” đơn vị là “lon”

#### Unrepeatable tread

##### Trường hợp 1 :

###### Kịch bản :

|  |  |
| --- | --- |
| T1 | T2 |
| - Kiểm tra danh sách nguyên liệu của nhà cung cấp  => Lấy danh sách nguyên liệu có trong nhà cung cấp này |  |
|  | **-** Cập nhật lại danh sách nguyên liệu của nhà cung cấp |
| - Kiểm tra danh sách nguyên liệu của nhà cung cấp  => Lấy danh sách nguyên liệu không có trong nhà cung cấp này (dùng để cập nhật thêm nguyên liệu cho chi tiết nhà cung cấp chưa có) |  |

* Hiện tại trong khó quản lý 10 danh mục nguyên liệu .Nhà cung cấp A cung cấp 4 nguyên liệu

1/Giao tác T1 lấy danh sách nguyên liệu của nhà cung cấp => có 4 nguyên liệu

2/Giao tác T2 xóa 2 nguyên liệu từ nhà cung cấp => Nhà cung cấp bây giờ có 2 nguyên liệu

3/Giao tác T1 kiểm tra danh sách nguyên liệu của nhà cung cấp => có 2 nguyên liệu

Và lấy ra danh sách 8 nguyên liệu còn lại (danh sách nguyên liệu còn lại phải là 10-4 =6)

* Giữa lần đọc dử liệu thứ nhất và lần đọc dử liệu thứ 2 của giao tác T1 có sự thay đổi, lần thứ 2 bị mất 2 nguyên liệu

###### Store procedure :

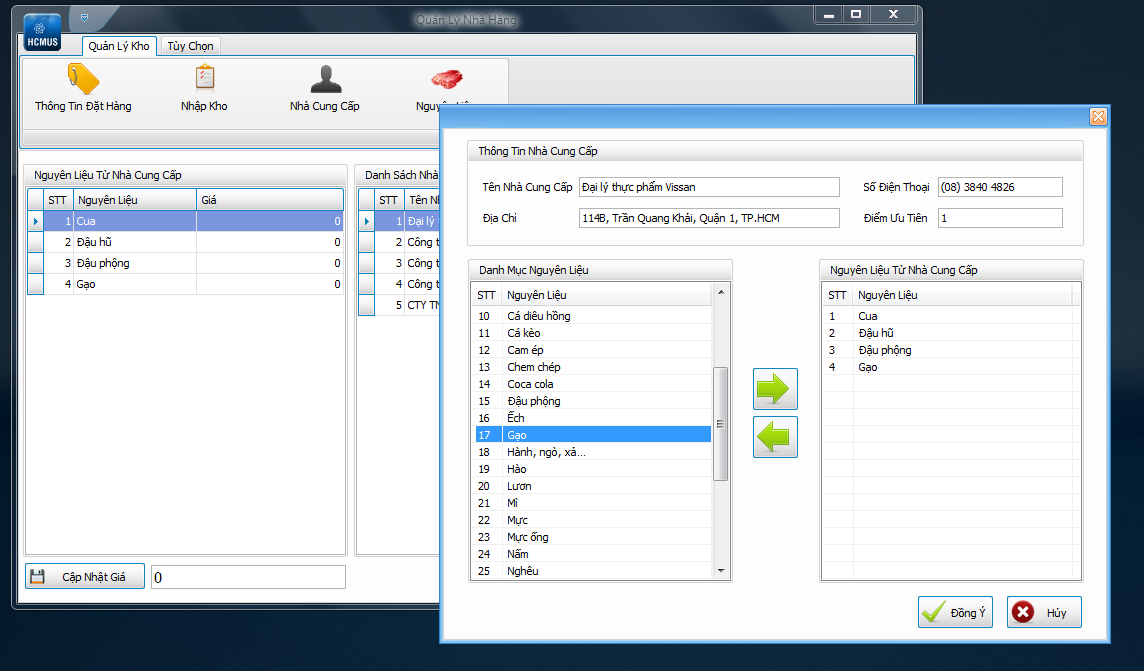
|  |  |
| --- | --- |
| T1 | T2 |
| create proc SelectNguyenLieu\_NCC  @MaNCC int,  @MaNH nchar(10)  as  begin  begin tran  set transaction isolation level  read uncommitted  select nl.\*  ,ct.Gia  from NhaCungCap ncc  ,ChiTietNCC ct  ,NguyenLieu nl  where ct.MaNCC = @MaNCC  and nl.MaNH = @MaNH  and ct.MaNL = nl.MaNL  and ct.MaNCC = ncc.MaNCC  order by nl.TenNL  waitfor delay '00:00:05' |  |
|  | create proc UpdateNhaCungCap  @Flag int out,  @MaNCC int,  @TenNCC nvarchar(50),  @sdt nvarchar(50),  @DiaChi nvarchar(50),  @DiemUuTien int,  @ChiTiet TableType\_ChiTietNCC readonly  as  begin  set @Flag = 0  begin tran  set transaction isolation level  read uncommitted    if(( select count(\*) from NhaCungCap  where MaNCC = @MaNCC )<>0)  begin  update NhaCungCap  set TenNCC = @TenNCC  ,sdt = @sdt  ,DiaChi = @DiaChi  ,DiemUuTien = @DiemUuTien  where MaNCC = @MaNCC  if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end    --Xoa nguyen lieu da bi mat trong danh sach  delete ChiTietNCC  where MaNCC = @MaNCC  and MaNL in  (select ctn.MaNL  from ChiTietNCC ctn  where ctn.MaNCC = @MaNCC  and ctn.MaNL not in  (select MaNL  from @ChiTiet))  if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  --Them cac nguyen lieu moi cap nhat  insert into ChiTietNCC  select ct.MaNL  ,ct.MaNCC  ,ct.Gia  from @ChiTiet ct  where ct.MaNL not in  (select ctn.MaNL  from ChiTietNCC ctn  where ctn.MaNCC = @MaNCC)    if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  end  set @Flag = 1  commit tran  end |
| select \*  from NguyenLieu nl1  where nl1.MaNH = @MaNH  and nl1.MaNL not in  (select nl.MaNL  from NguyenLieu nl  ,ChiTietNCC ct  where ct.MaNCC = @MaNCC  and nl.MaNH = @MaNH  and nl.MaNL = ct.MaNL)  order by nl1.TenNL |  |

###### Nguyên nhân tranh chấp:

Với mức cô lập **read uncommitted** khống thiết lập **Shared Lock** trên đơn vị dử liệu được đọc, vì vậy khi T1 đọc dử liệu lần 1T2 có quyền ghi dử liệu(update,delete), dẫn đến lần đọc dử liệu thứ 2 của T1 có sự thay đổi.

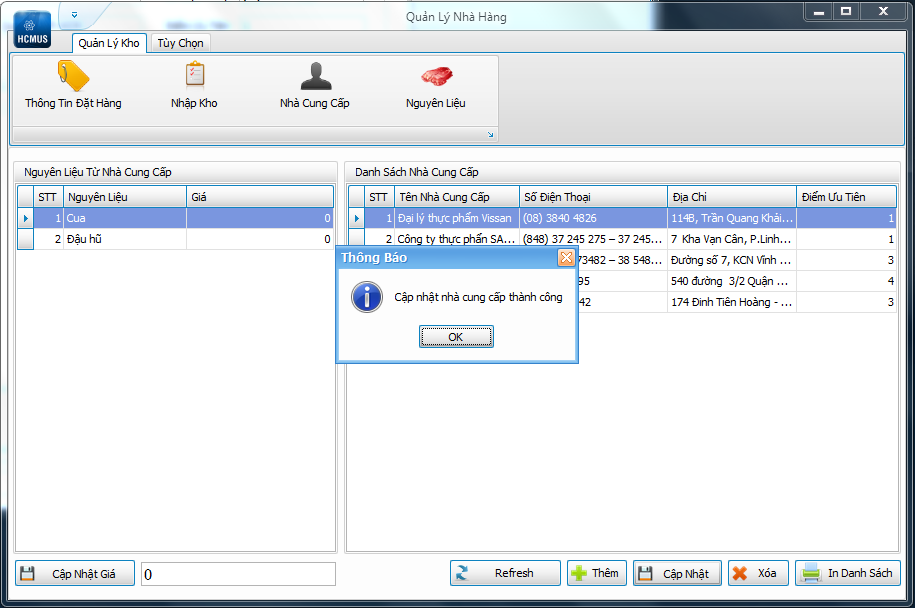
###### Giao diện chương trình:

Giao tác T1:



Lần đọc dử liệu thứ 2 trong danh mục nguyên liệu còn lại xuất hiện “Gạo” và “Đậu phộng” do lúc kiểm tra danh sách nguyên liệu từ nhà cung cấp bị mất 2 nguyên liệu này(T2 xóa)

Giao tác T2:



#### Phantom

##### Trường hợp 1 :

###### Kịch bản :

|  |  |
| --- | --- |
| T1 | T2 |
| - Kiểm tra danh sách nguyên liệu của nhà cung cấp  => Lấy danh sách nguyên liệu có trong nhà cung cấp này |  |
|  | **-** Cập nhật lại danh sách nguyên liệu của nhà cung cấp |
| - Kiểm tra danh sách nguyên liệu của nhà cung cấp  => Lấy danh sách nguyên liệu không có trong nhà cung cấp này (dùng để cập nhật thêm nguyên liệu cho chi tiết nhà cung cấp chưa có) |  |

* Hiện tại trong khó quản lý 10 danh mục nguyên liệu .Nhà cung cấp A cung cấp 4 nguyên liệu

1/Giao tác T1 lấy danh sách nguyên liệu của nhà cung cấp => có 4 nguyên liệu

2/Giao tác T2 thêm 2 nguyên liệu từ nhà cung cấp => Nhà cung cấp bây giờ có 6 nguyên liệu

3/Giao tác T1 kiểm tra danh sách nguyên liệu của nhà cung cấp => có 6 nguyên liệu

Và lấy ra danh sách 4 nguyên liệu còn lại (danh sách nguyên liệu còn lại phải là 10-4 =6)

* Giữa lần đọc dử liệu thứ nhất và lần đọc dử liệu thứ 2 của giao tác T1 có sự thay đổi, lần thứ 2 xuất hiện thêm 2 nguyên liệu

###### Store procedure :

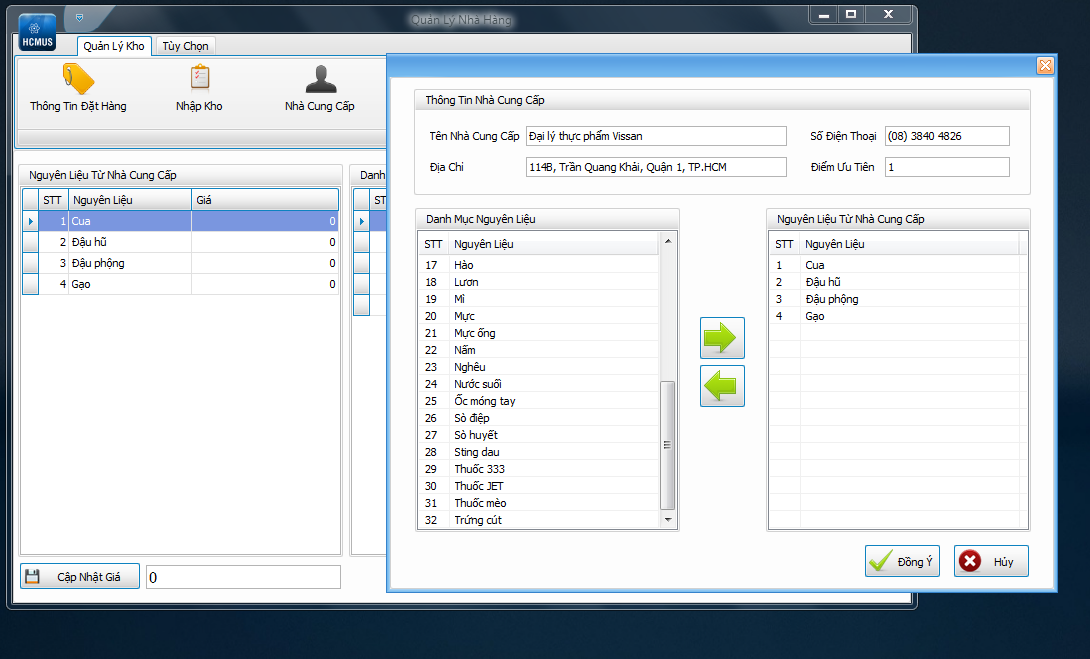
|  |  |
| --- | --- |
| T1 | T2 |
| create proc SelectNguyenLieu\_NCC  @MaNCC int,  @MaNH nchar(10)  as  begin  begin tran  set transaction isolation level  read uncommitted  select nl.\*  ,ct.Gia  from NhaCungCap ncc  ,ChiTietNCC ct  ,NguyenLieu nl  where ct.MaNCC = @MaNCC  and nl.MaNH = @MaNH  and ct.MaNL = nl.MaNL  and ct.MaNCC = ncc.MaNCC  order by nl.TenNL  waitfor delay '00:00:05' |  |
|  | create proc UpdateNhaCungCap  @Flag int out,  @MaNCC int,  @TenNCC nvarchar(50),  @sdt nvarchar(50),  @DiaChi nvarchar(50),  @DiemUuTien int,  @ChiTiet TableType\_ChiTietNCC readonly  as  begin  set @Flag = 0  begin tran  set transaction isolation level  read uncommitted    if(( select count(\*) from NhaCungCap  where MaNCC = @MaNCC )<>0)  begin  update NhaCungCap  set TenNCC = @TenNCC  ,sdt = @sdt  ,DiaChi = @DiaChi  ,DiemUuTien = @DiemUuTien  where MaNCC = @MaNCC  if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end    --Xoa nguyen lieu da bi mat trong danh sach  delete ChiTietNCC  where MaNCC = @MaNCC  and MaNL in  (select ctn.MaNL  from ChiTietNCC ctn  where ctn.MaNCC = @MaNCC  and ctn.MaNL not in  (select MaNL  from @ChiTiet))  if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  --Them cac nguyen lieu moi cap nhat  insert into ChiTietNCC  select ct.MaNL  ,ct.MaNCC  ,ct.Gia  from @ChiTiet ct  where ct.MaNL not in  (select ctn.MaNL  from ChiTietNCC ctn  where ctn.MaNCC = @MaNCC)    if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  end  set @Flag = 1  commit tran  end |
| select \*  from NguyenLieu nl1  where nl1.MaNH = @MaNH  and nl1.MaNL not in  (select nl.MaNL  from NguyenLieu nl  ,ChiTietNCC ct  where ct.MaNCC = @MaNCC  and nl.MaNH = @MaNH  and nl.MaNL = ct.MaNL)  order by nl1.TenNL |  |

###### Nguyên nhân tranh chấp:

Với mức cô lập **read uncommitted** khống thiết lập **Shared Lock** trên đơn vị dử liệu được đọc, vì vậy khi T1 đọc dử liệu lần 1T2 có quyền ghi dử liệu(update,delete), dẫn đến lần đọc dử liệu thứ 2 của T1 có sự thay đổi.

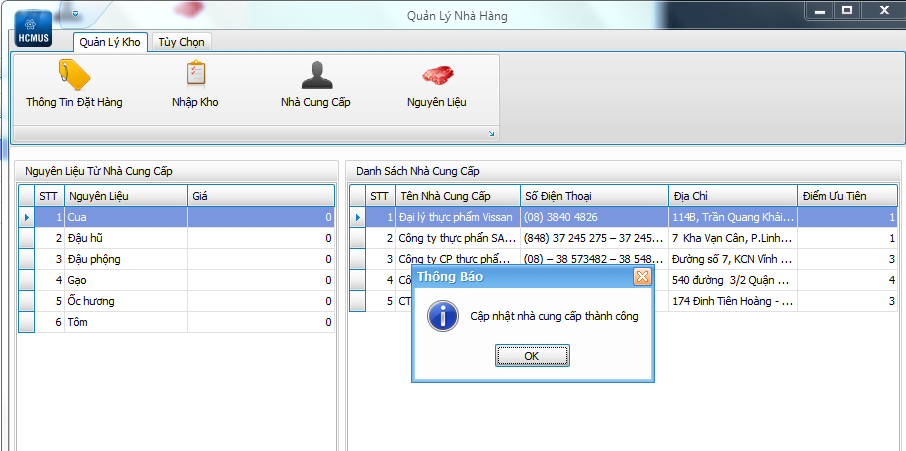
###### Giao diện chương trình:

Giao tác T1:



Trong danh mục Nguyên liệu bị mất 2 nguyên liệu “Ốc hương” và “Tôm” do nó xuất hiện trong danh sách nguyên liệu của nhà cung cấp

Giao tác T2:



#### Lost update & deadlock

##### Trường hợp 1 :

###### Kịch bản :

|  |  |
| --- | --- |
| T1 | T2 |
| - thêm hóa đơn nhập kho, cập nhật lại tình trạng đơn đặt hàng là “Đã giao” |  |
| - Lấy số lượng tồn của nguyên liệu |  |
|  | **-** thêm hóa đơn nhập kho, cập nhật lại tình trạng đơn đặt hàng là “Đã giao” |
|  | **-** Lấy số lượng tồn của nguyên liệu |
| - Cập nhật lại số lượng tồn của nguyên liệu = số lượng tồn củ + với số lượng nguyên liệu có trong chi tiết hóa đơn |  |
|  | **-** Cập nhật lại số lượng tồn của nguyên liệu = số lượng tồn củ + với số lượng nguyên liệu có trong chi tiết hóa đơn |

* Xét trường hợp 1 nguyên liệu có số lượng tồn = 50,
* Người dùng 1 tiến hành nhập kho vào số lượng nguyên liệu này là 5 và cập nhật số lượng tồn của nguyên liệu = 50 +5 = 55
* Người dùng 2 cũng tiến hành nhập kho với số lượng nguyên liệu là 10 và cập nhất số lượng tồn = 50 +10 =60
* Trong khi đó số lượng nguyên liệu thực tế trong kho là 50 + 5 +10 = 65

###### Store procedure :

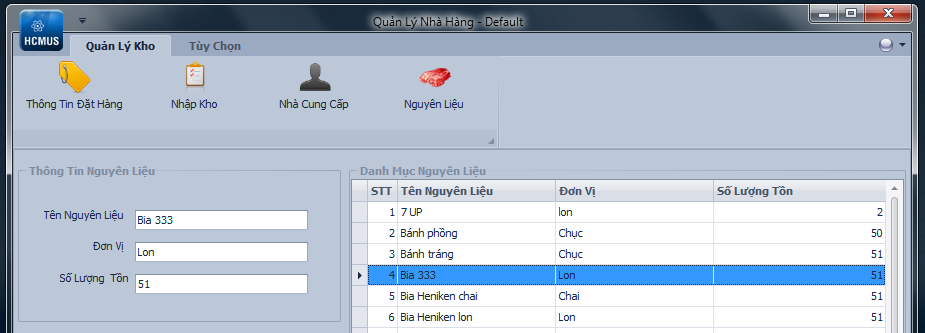
|  |  |
| --- | --- |
| T1 | T2 |
| create proc UpdateTinhTrangDatHang  @Flag int out,  @MaHoaDon int,  @TinhTrang nvarchar(50)  as  begin  set @Flag = 0  begin tran  set transaction isolation level  read uncommitted  if(@TinhTrang <> N'Đã Giao')  begin  update DatHang  set TinhTrang = @TinhTrang  where MaHoaDon = @MaHoaDon    if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  end  else  begin  update DatHang  set TinhTrang = @TinhTrang  where MaHoaDon = @MaHoaDon  if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  --Cap nhat so luong nguyen lieu ton  declare @soluongton int,  @soluongthem int,  @MaNL int  declare @cur cursor  set @cur = cursor for  select MaNL  from ChiTietDatHang ctdh  where ctdh.MaHoaDon = @MaHoaDon    open @cur  fetch next from @cur into @MaNL  while @@FETCH\_STATUS = 0  begin  set @soluongthem = (  select ctdh.SoLuong  from ChiTietDatHang ctdh  where ctdh.MaHoaDon = @MaHoaDon  and MaNL = @MaNL)  set @soluongton =  (select SoLuongTon  from NguyenLieu  where MaNL = @MaNL)  waitfor delay '00:00:06' |  |
|  | create proc UpdateTinhTrangDatHang  @Flag int out,  @MaHoaDon int,  @TinhTrang nvarchar(50)  as  begin  set @Flag = 0  begin tran  set transaction isolation level  read uncommitted  if(@TinhTrang <> N'Đã Giao')  begin  update DatHang  set TinhTrang = @TinhTrang  where MaHoaDon = @MaHoaDon    if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  end  else  begin  update DatHang  set TinhTrang = @TinhTrang  where MaHoaDon = @MaHoaDon  if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  --Cap nhat so luong nguyen lieu ton  declare @soluongton int,  @soluongthem int,  @MaNL int  declare @cur cursor  set @cur = cursor for  select MaNL  from ChiTietDatHang ctdh  where ctdh.MaHoaDon = @MaHoaDon    open @cur  fetch next from @cur into @MaNL  while @@FETCH\_STATUS = 0  begin  set @soluongthem = (  select ctdh.SoLuong  from ChiTietDatHang ctdh  where ctdh.MaHoaDon = @MaHoaDon  and MaNL = @MaNL)  set @soluongton =  (select SoLuongTon  from NguyenLieu  where MaNL = @MaNL)  waitfor delay '00:00:06' |
| update NguyenLieu  set SoLuongTon = @soluongthem+@soluongton  where MaNL = @MaNL    if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  fetch next from @cur into @MaNL  end  end  set @Flag = 1  commit tran  end |  |
|  | update NguyenLieu  set SoLuongTon = @soluongthem+@soluongton  where MaNL = @MaNL    if(@@ERROR<>0)  begin  set @Flag = 0  rollback  return  end  fetch next from @cur into @MaNL  end  end  set @Flag = 1  commit tran  end |

###### Nguyên nhân tranh chấp:

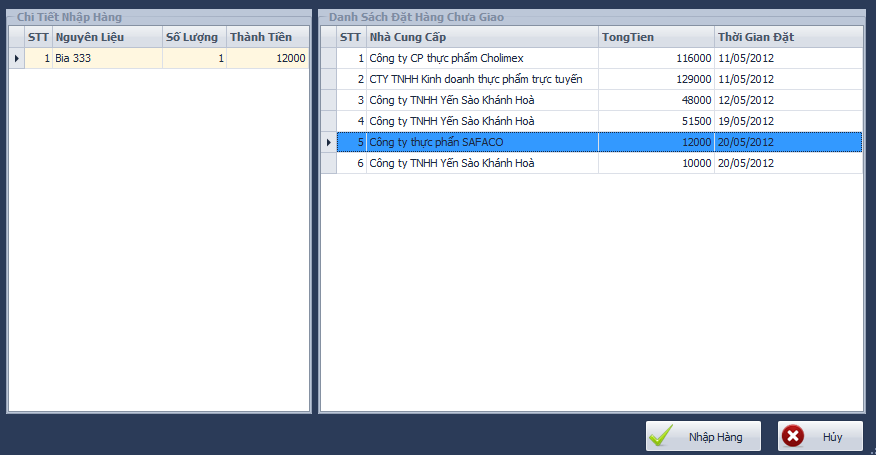
Với mức cô lập **read uncommitted** khống thiết lập **Shared Lock** trên đơn vị dử liệu được đọc, vì vậy khi T1 đọc dử liệu A thì T2 vẫn được đọc dử A này, nên khi T2 cập nhật dử liệu với thông tin đọc được là A thì trên đãi T1 đã cập nhật lại dử liệu A này thành giá trị khác rồi => dẫn đến T2 cập nhật dử liệu với thông tin sai

###### Giao diện chương trình:

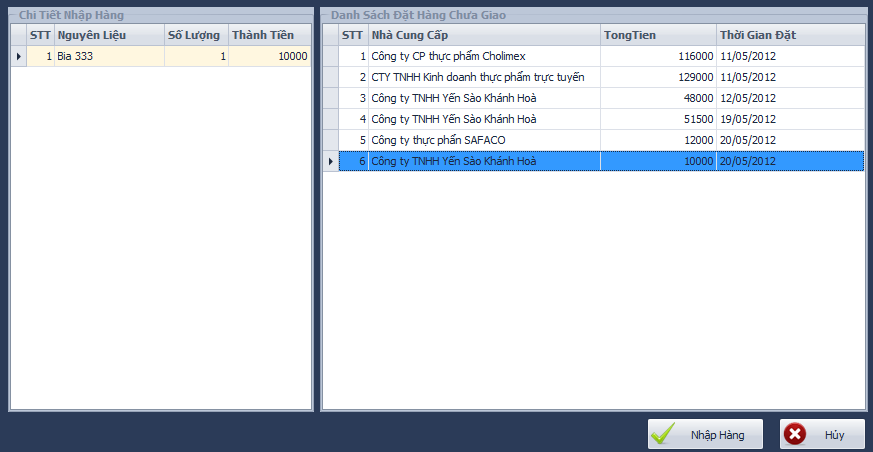
Xét Nguyên Liệu “Bia 333” số lượng tồn hiện tại =51



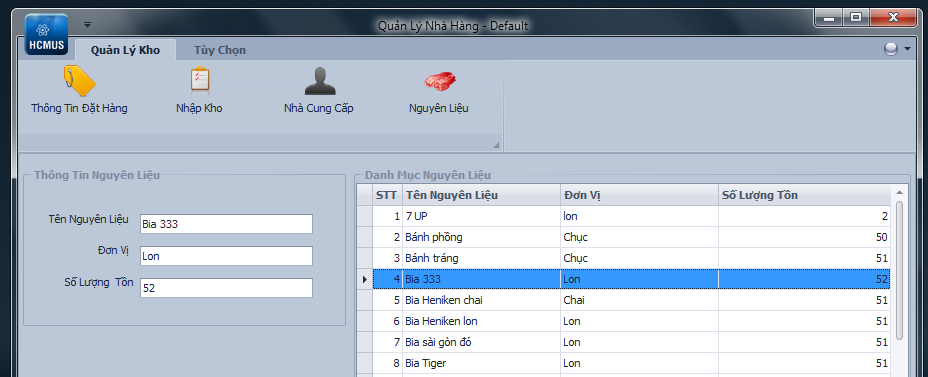
Người dùng 1: Nhập kho đơn đăt hàng từ công ty SAFACO với số lượng là 1=>cập nhật số lượng “Bia 333” tăng lên 1



Người dùng 2: Nhập kho đơn đăt hàng từ công ty Yến Sào Khánh Hòa với số lượng là 1=>cập nhật số lượng “Bia 333” tăng lên 1



Kết quả : sau 2 lần tăng số lượng tồn kết quả = 52 . Nhưng thực tế phải là 53



### Xử lý các tranh chấp

#### Drity read

Sử dụng mức cô lập **read committed** :với mức cô lập này khi T2 cần đọc dử liệu nó phải xin khóa đọc,nhưng do T1 đang ghi, nên nó không xin được khóa đọc mà phải đợi đên khi T1 commit T2 mới có thể đọc dử liệu => Khắc phục được lỗi đọc dử liệu rác

#### Unrepeatable tread

Mức cô lập : **read committed** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và **Shared Lock** được giải phóng ngay sau khi đọc xong dử liệu, vì vậy khi T1 đọc dử liệu xong T2 vẫn có thể update hoặc delete dử liệu này => không giải quyết được tranh chấp

Mức cô lập : **repeatable read** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và giữ **Shared Lock** này đến hết giao tác . Vì vậy ghi T1 đọc dử liệu T2 không thể update hoặc delete những đơn vị dử liệu mà T1 đang giữ **Shared Lock** => Khắc phục được tranh chấp dử liệu

#### Phantom

Mức cô lập : **read committed** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và **Shared Lock** được giải phóng ngay sau khi đọc xong dử liệu, vì vậy khi T1 đọc dử liệu xong T2 vẫn có thể insert thêm dử liệu

* Không giải quyết được tranh chấp

Mức cô lập : **repeatable read** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và giữ **Shared Lock** này đến hết giao tác . Vì vậy ghi T1 đọc dử liệu T2 vẫn có thể insert dử liệu => không giải quyết được tranh chấp

Mức cô lập : **serializable** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và giữ **Shared Lock** này đến hết giao tác .Đồng thời không cho phép insert những dòng dử liệu thỏa mãn điều kiện thiết lập **Shared Lock**

Do đó khi T1 đọc dử liệu trên bảng ChiTietNCC thì T2 không được insert dử liệu lên bảng này

* Khắc phục được tranh chấp

#### Lost update & deadlock

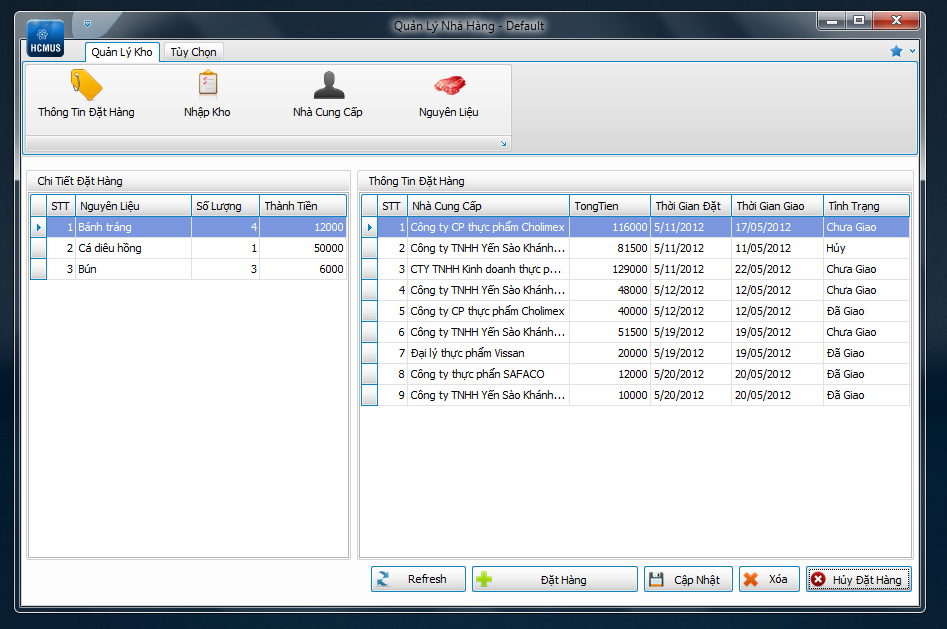
Mức cô lập : **read committed** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và **Shared Lock** được giải phóng ngay sau khi đọc xong dử liệu, vì vậy khi T1 đọc dử liệu xong T2 vẫn được đọc dử liệu này => không giải quyết được tranh chấp

Mức cô lập : **repeatable read** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và giữ **Shared Lock** này đến hết giao tác . với mức cô lập này T1 đọc dử liệu T2 vẫn được đọc=> Khi T2 phát khóa **Shared Lock** thì T1 không được ghi do T2 đang giữ khóa đọc nên bị deadlock, trong khi đó T2 sau khi đọc xong lại xin khóa ghi trong khi T1 lại đang giữ khóa **Shared Lock** nên cũng bị deadlock => chường trình bị treo

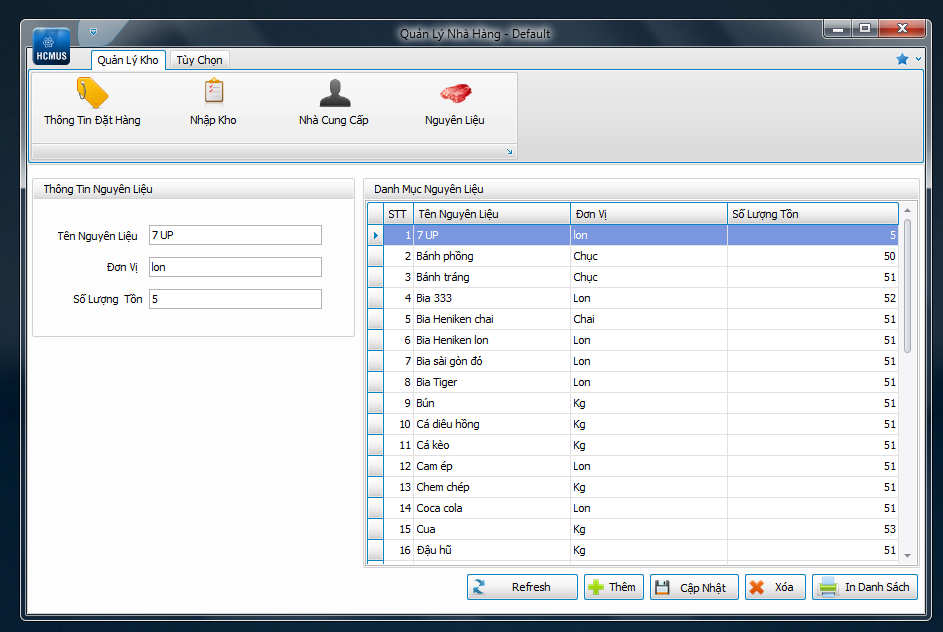
* Để giải quyết tình trạng này ta phải kết hợp Mức cô lập : **repeatable read** và Khóa **Updlock**  để khóa nhưng dòng dữ liệu khi 1 giao tác đang đọc không cho các giao tác khác cập nhật lên những dòng dữ liệu này

# 3.Các chức năng trong đề tài

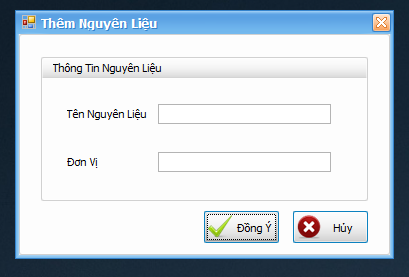
## 3.3. Nhân Viên Quản Lý Kho



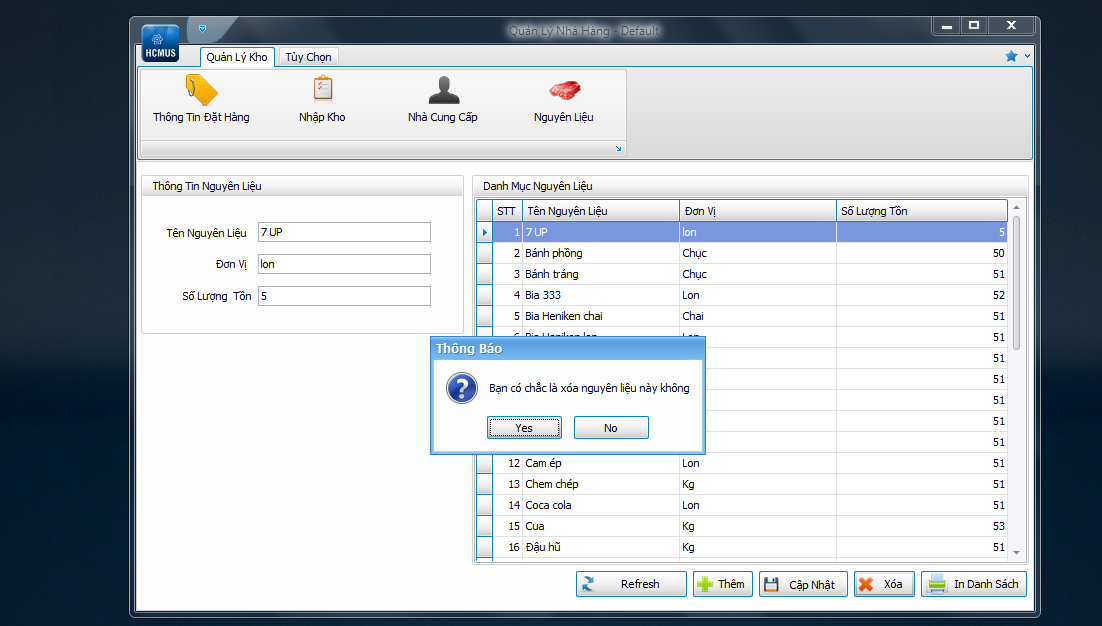
### 3.3.1.Quản Lý Danh Sách Nguyên Liệu



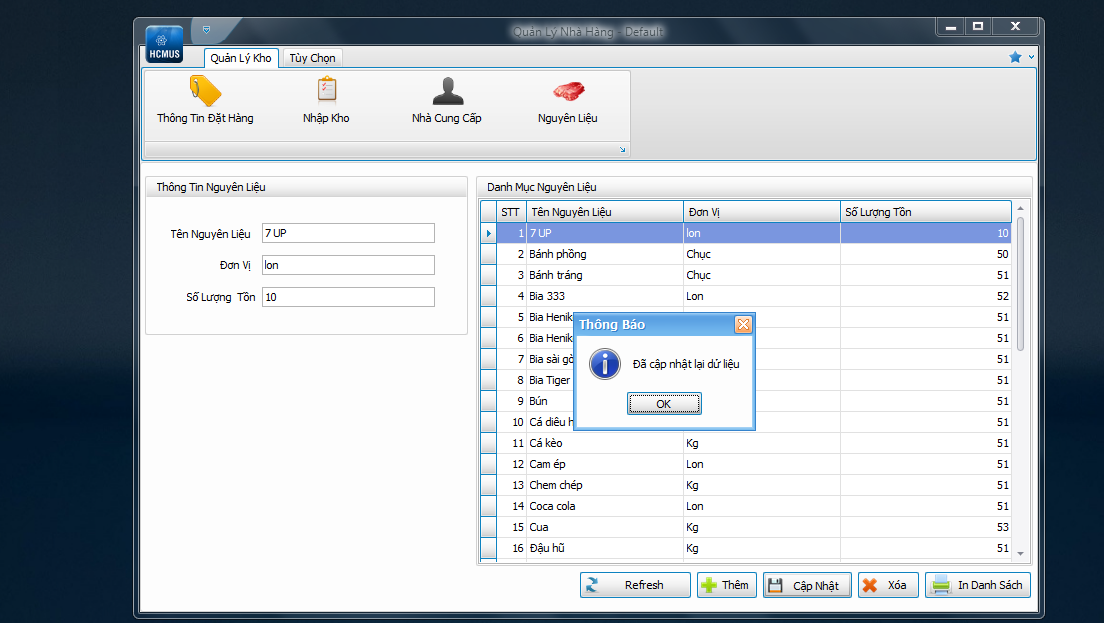
#### 3.3.1.1.Thêm Nguyên Liệu



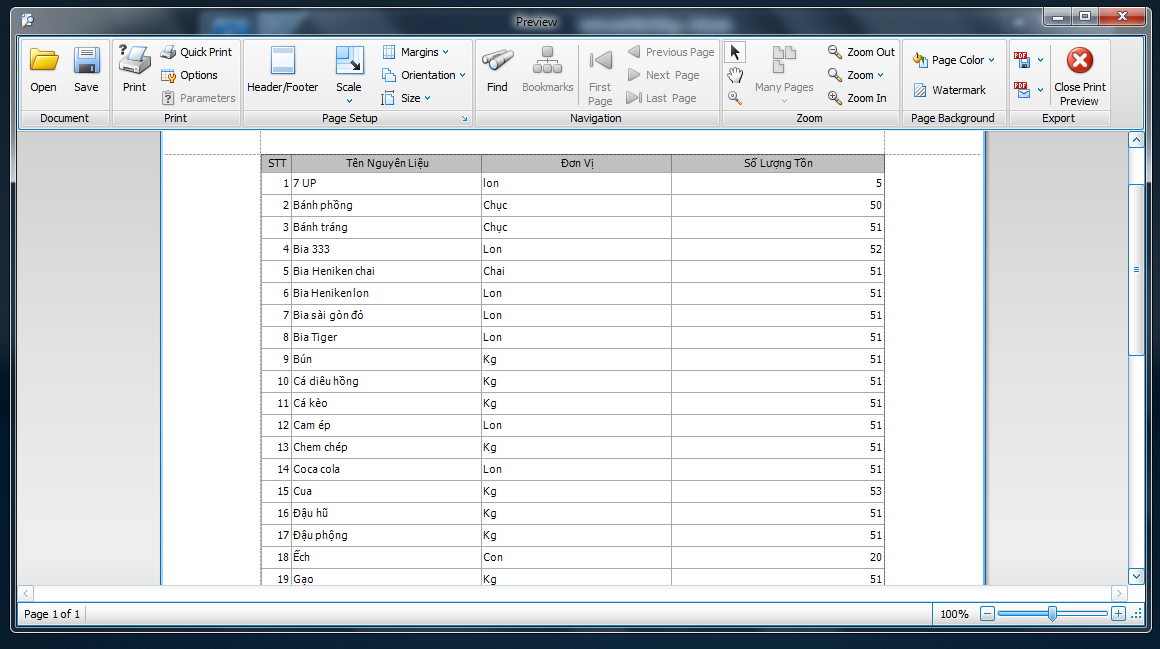
#### 3.3.1.2.Xóa Nguyên Liệu



#### 3.3.1.3.Cập Nhật Nguyên Liệu

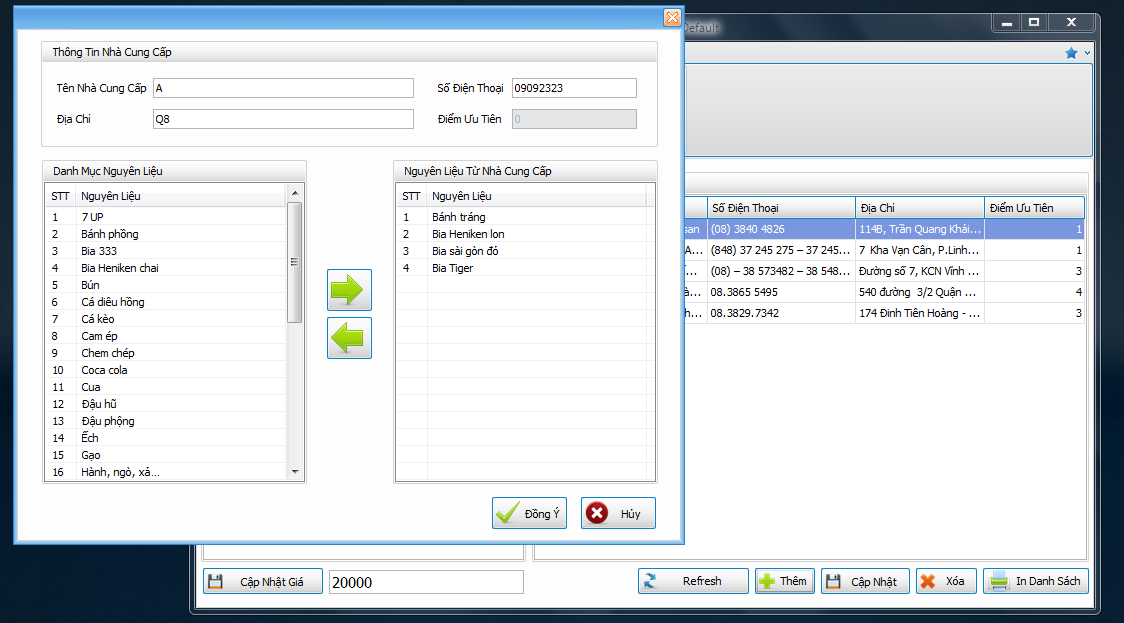


#### 3.3.1.4.In Danh Sách Nguyên Liệu

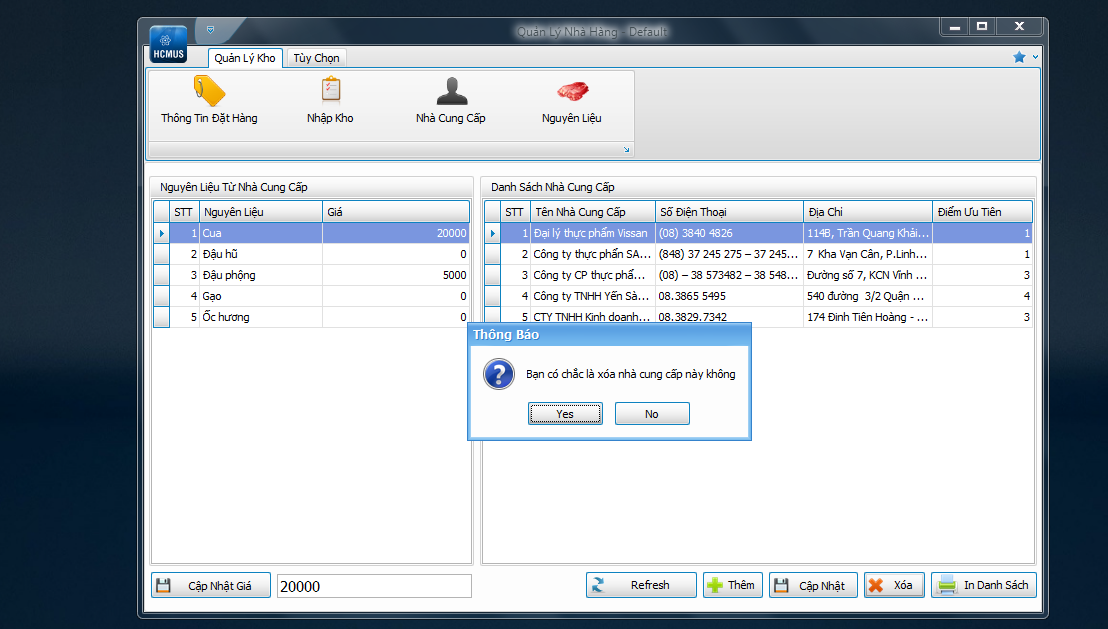


### 3.3.2.Quản Lý Danh Sách Nhà Cung Cấp

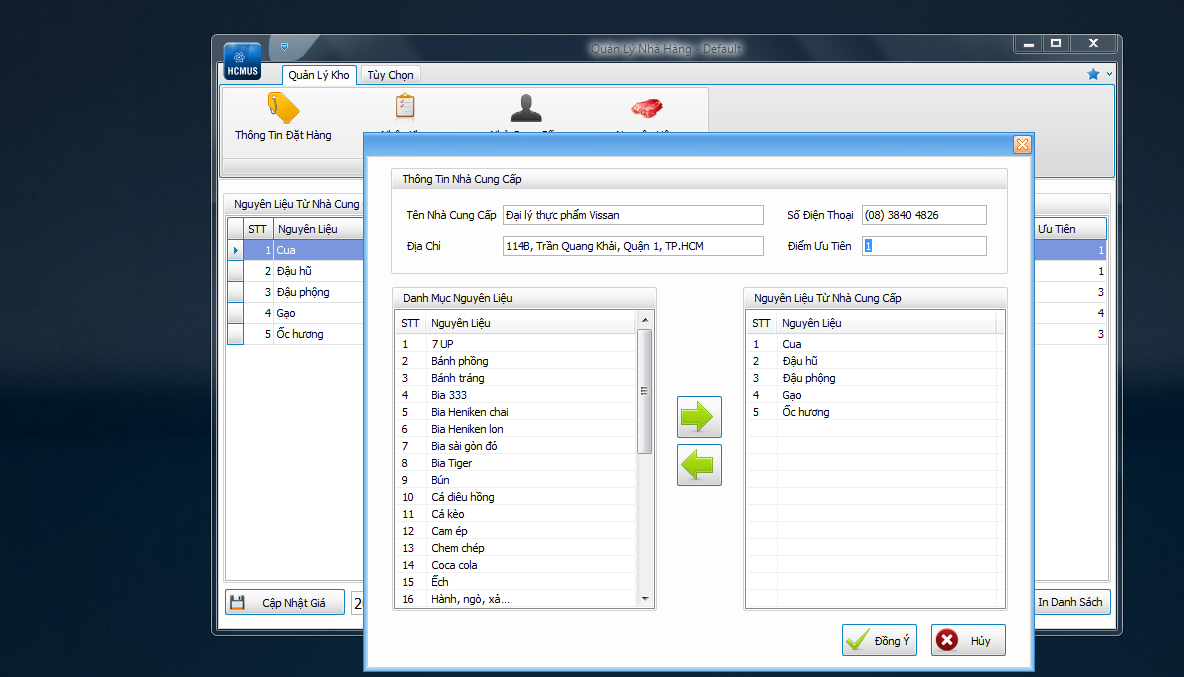
#### 3.3.2.1.Thêm Nhà Cung Cấp Mới



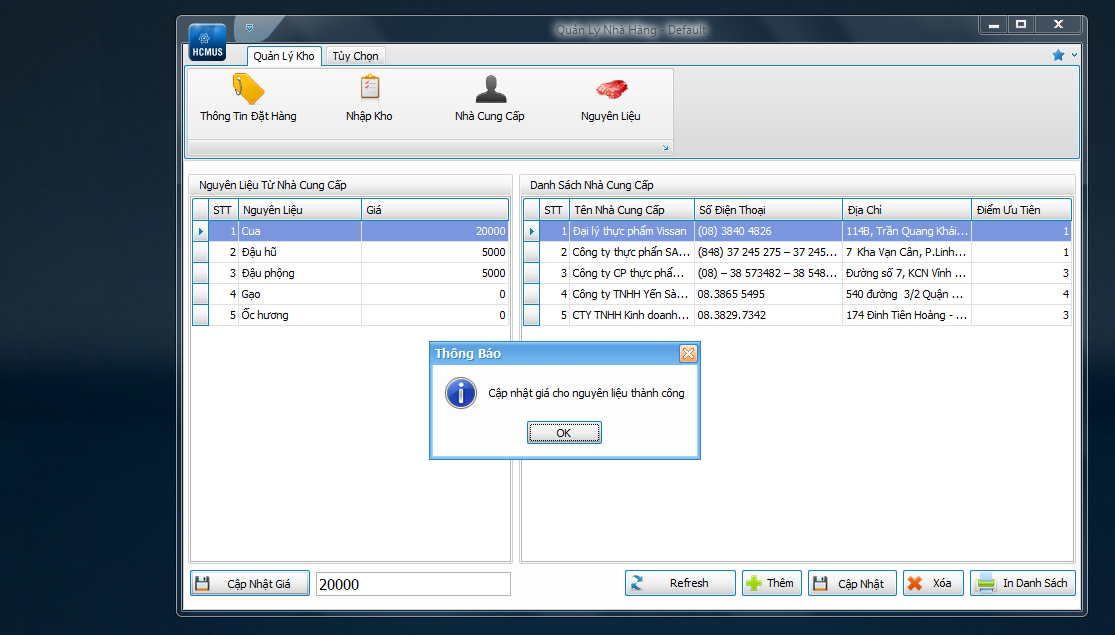
#### 3.3.2.2.Xóa Nhà Cung Cấp



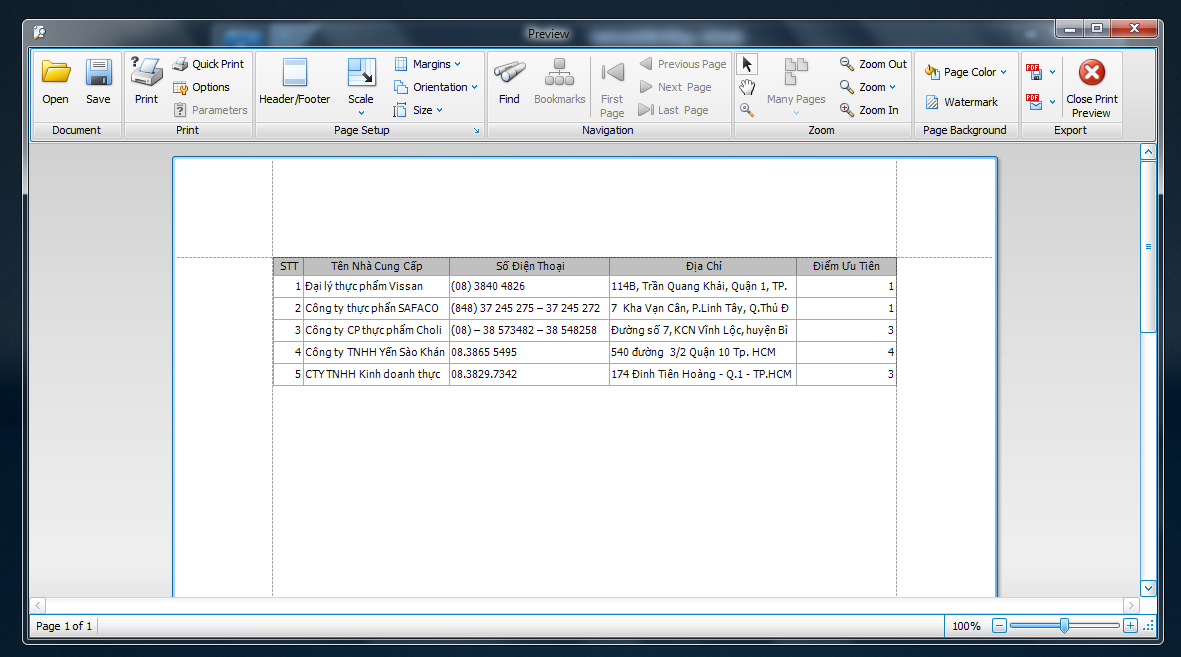
#### 3.3.2.3.Cập Nhật Nhà Cung Cấp



#### 3.3.2.4.Cập Nhật Giá Nguyên Liệu Nhà Cung Cấp



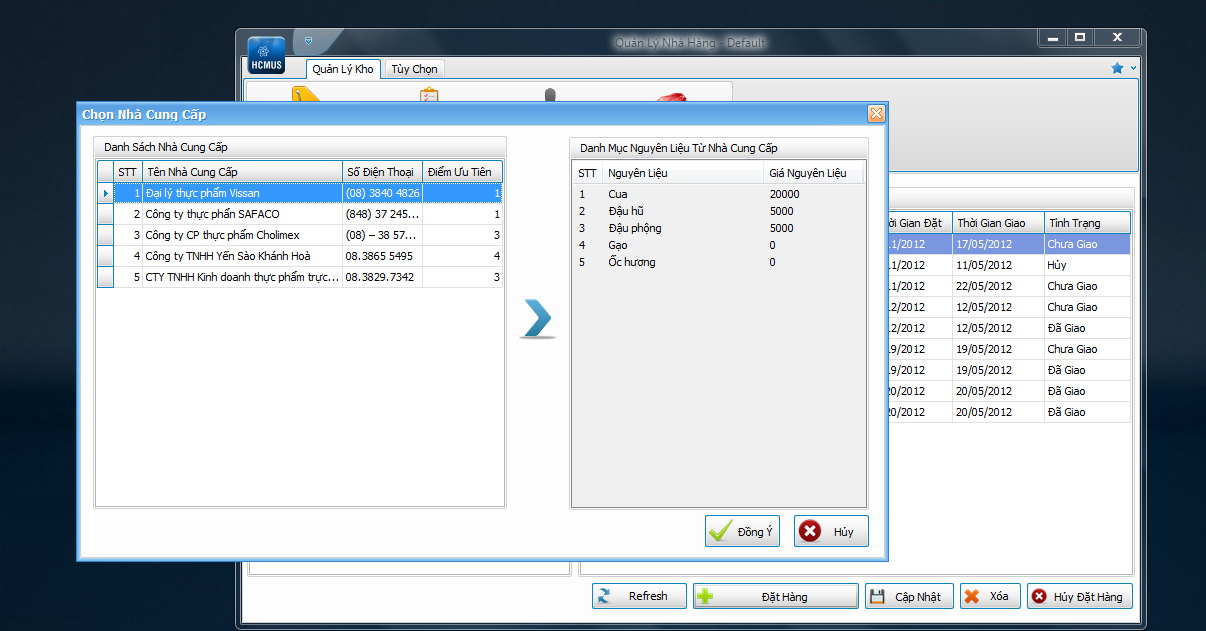
#### 3.3.2.5.In Danh Sách Nhà Cung Cấp



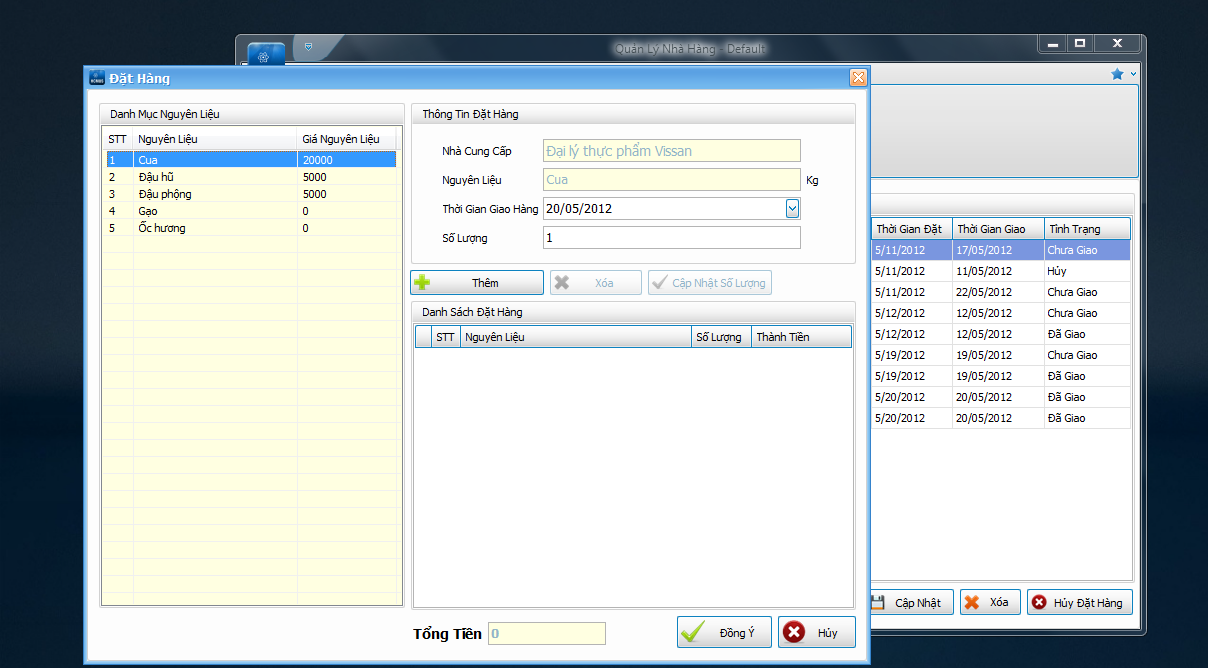
### 3.3.3.Quản Lý Thông Tin Đặt Hàng

#### 3.3.3.1.Thêm Đơn Đặt Hàng Mới

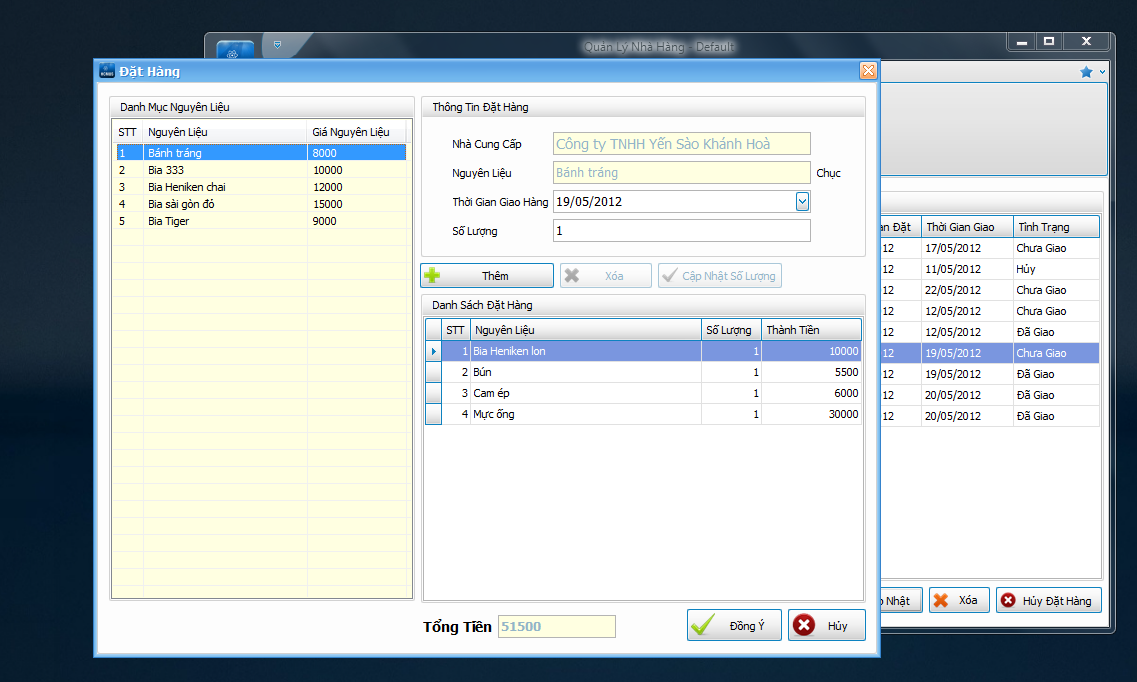
##### a.Chọn Nhà Cung Cấp



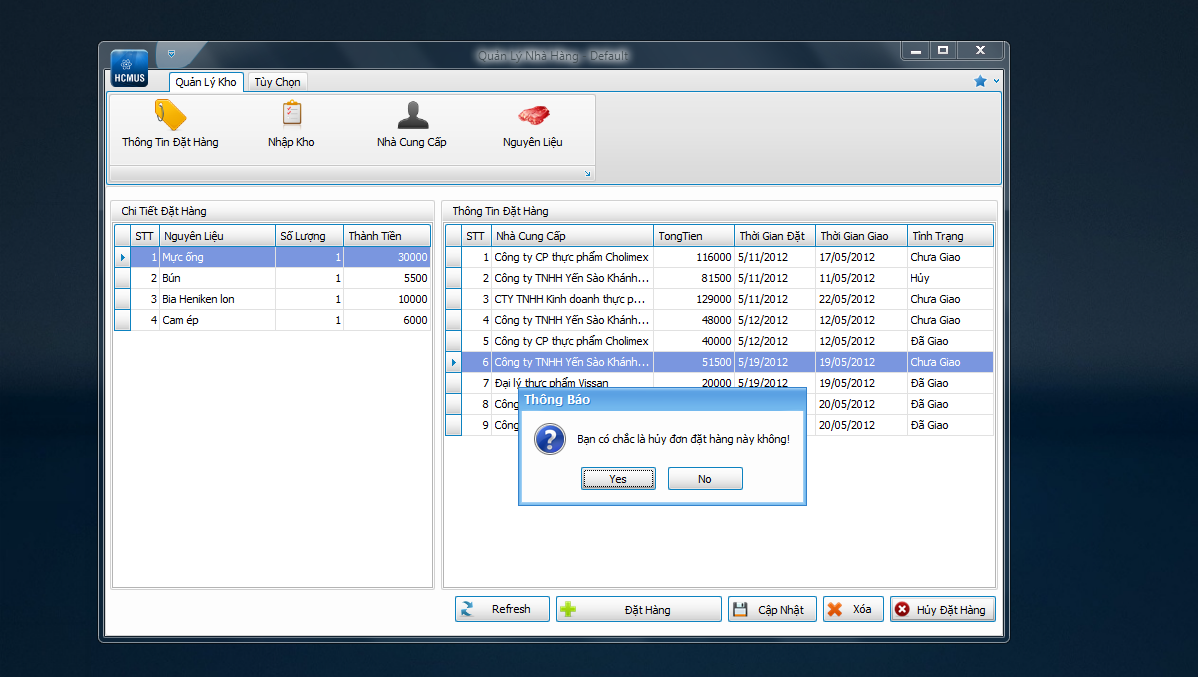
##### b.Chọn Nguyên Liệu



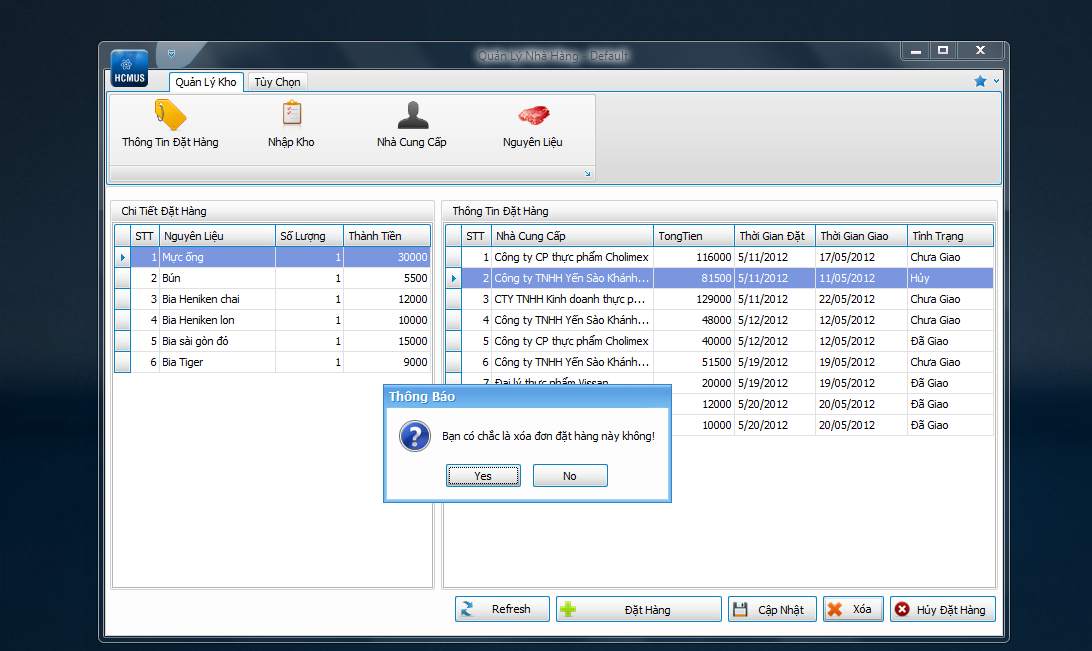
#### 3.3.3.2.Cập Nhật Thông Tin Đơn Đặt Hàng Chưa Giao



#### 3.3.3.4.Hủy Đơn Đặt Hàng Chưa Giao

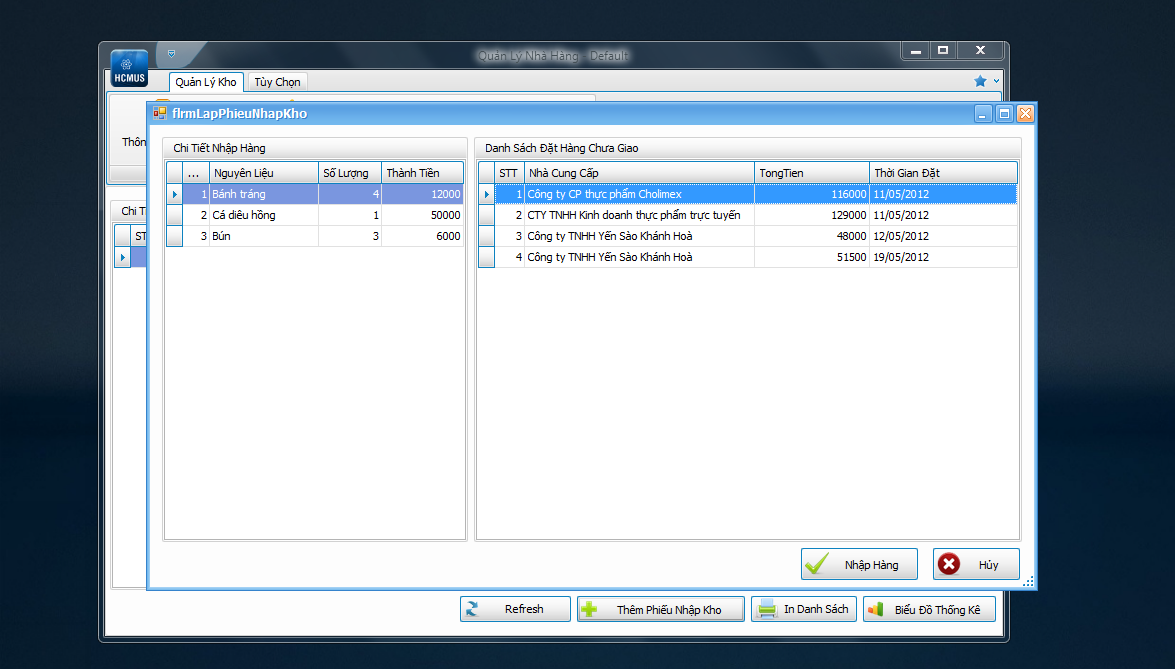


#### 3.3.3.5.Xóa Đơn Đặt Hàng “Chưa Giao” Hoặc Đã “Hủy”

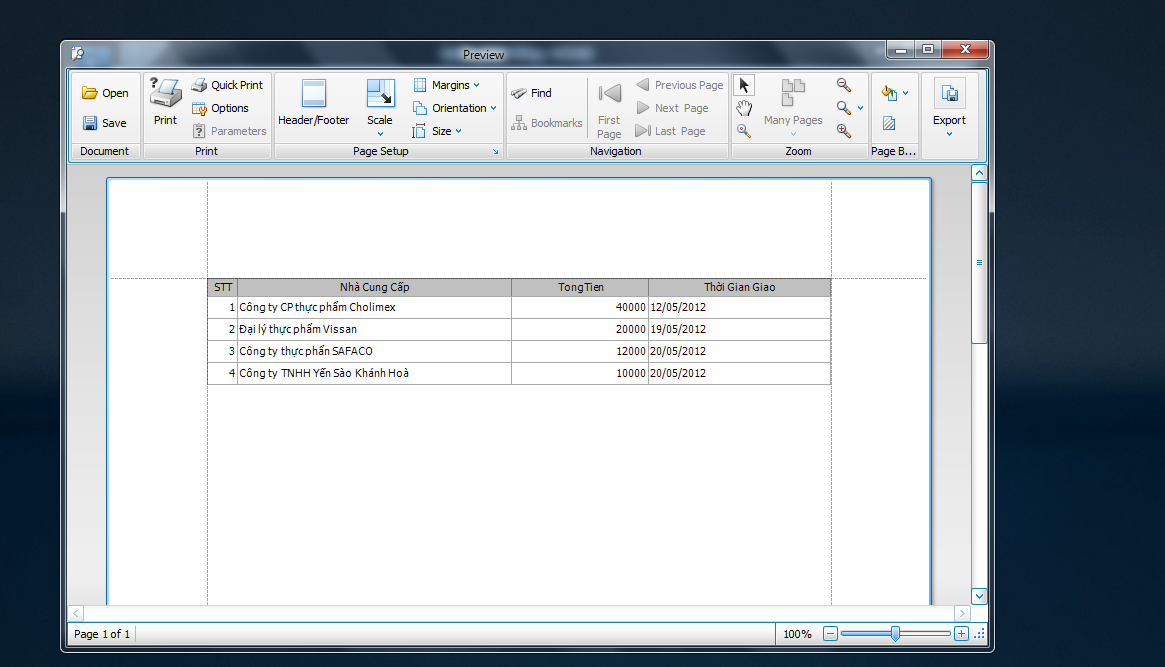


### 3.3.4.Quản Lý Thông Tin Nhập Kho

#### 3.3.4.1.Thêm Phiếu Nhập Kho



#### 3.3.4.2.In Danh Sách Phiếu Nhập Kho



#### 3.3.4.3.Thống Kê Thông Tin Nhập Kho

