**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](#_Toc325664806)

[1.1 Thông tin thành viên: 1](#_Toc325664807)

[2. Các tranh chấp đồng thời đã phát hiện 1](#_Toc325664808)

[2.1 1141085 – Nguyễn Bá Ngọc 1](#_Toc325664809)

[2.1.1 Các trường hợp tranh chấp 1](#_Toc325664810)

[2.2 1141187 – Nguyễn Minh Vũ 8](#_Toc325664811)

[2.2.1 Các trường hợp tranh chấp 8](#_Toc325664812)

[2.3 1141083 – Nguyễn Minh Nghĩa 24](#_Toc325664813)

[2.3.1 Các trường hợp tranh chấp 24](#_Toc325664814)

[2.4 1141129 – Bùi Trung Tân 28](#_Toc325664815)

[2.4.1 Các trường hợp tranh chấp 28](#_Toc325664816)

[3.Các chức năng trong đề tài 39](#_Toc325664817)

[3.1.Nhân Viên Tiếp Tân 39](#_Toc325664818)

[1.1.1 Đặt bàn 39](#_Toc325664819)

[1.1.2 Danh sách Đặt bàn 40](#_Toc325664820)

[1.1.3 Xếp bàn ăn cho khách 41](#_Toc325664821)

[3.2.Nhân Viên Thu Ngân 42](#_Toc325664822)

[Thêm món ăn 42](#_Toc325664823)

[Xoá món ăn, cập nhật số lượng món ăn 42](#_Toc325664824)

[Chuyển bàn 43](#_Toc325664825)

[Ghép bàn 43](#_Toc325664826)

[Kết thúc sử dụng bàn 44](#_Toc325664827)

[3.3. Nhân Viên Quản Lý Kho 45](#_Toc325664828)

[3.3.1.Quản Lý Danh Sách Nguyên Liệu 46](#_Toc325664829)

[3.3.2.Quản Lý Danh Sách Nhà Cung Cấp 48](#_Toc325664830)

[3.3.3.Quản Lý Thông Tin Đặt Hàng 51](#_Toc325664831)

[3.3.4.Quản Lý Thông Tin Nhập Kho 53](#_Toc325664832)

[3.4.Quản Lý Nhà Hàng 54](#_Toc325664833)

[3.4.1.Quản lý Thực đơn 55](#_Toc325664834)

[3.4.2.Quản Lý Danh Sách Nhân Viên 58](#_Toc325664835)

[3.4.3.Quản Lý Danh Sách Bàn Ăn 62](#_Toc325664836)

# 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 |  | 100% |
| 3 | 1141083 | Nguyễn Minh Nghĩa |  | 100% |
| 4 | 1141129 | Bùi Trung Tân |  | 100% |

# Các tranh chấp đồng thời đã phát hiện

## 1141085 – Nguyễn Bá Ngọc

### Các trường hợp tranh chấp

#### Drity read: Nghiệp vụ Đặt Bàn Ăn.

* ***Kịch bản:***

|  |  |
| --- | --- |
| T1 | T2 |
| * Thực hiện thao tác Thêm bàn đặt |  |
|  | * **Xem danh sách đặt bàn** |
| * Hủy thao tác Thêm bàn đặt |  |

* Khi T1 đang thực hiện thao tác Thêm 1 bàn đặt mà chưa Commit, cùng lúc đó T2 giữ khóa Read Uncommitted (đọc không cần xin khóa) đọc danh sách các bàn đã được đặt trong đó có bàn mà T1 mới đặt. Lỗi Drity Read xảy ra khi T1 hủy bản vừa mới đặt.
* **Sữa lỗi:**

|  |  |
| --- | --- |
| T1 | T2 |
| * begin tran * insert into DatBan (maNH,MaBan,MaKhachHang,   ThoiGianDat,TrangThai,ThoiGianDen,sdt) values(1,3,123654789,'05/12/2012',0,'05/12/2012',0) |  |
|  | * **SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED** * **begin tran** * **select \* from datban with (rowlock,readcommitted)** * **commit tran** |
| * waitfor delay '0:0:5' * declare @sdt int * select @sdt=sdt from DatBan where maban=3 * if @@error>0 or @sdt=0   rollback tran   * else commit tran |  |

* Khi T1 đang thực hiện thao tác Thêm 1 bàn đặt mà chưa Commit, cùng lúc đó T2 đọc danh sách bàn đã được đặt “With (rowlock,readcommitted)” (xin khóa đọc những dòng dữ liệu cần đọc trong 1 bảng và nhả khóa sau khi đọc xong). Như vậy: T2 đọc dữ liệu, nếu trong dữ liệu đó có dòng của T1 đã thêm thì T2 sẽ phải chờ T1 thực hiện xong mới thực hiện được việc đọc dữ liệu.

#### Lost update: Nghiệp vụ Đặt bàn ăn.

* ***Kịch bản:***

|  |  |
| --- | --- |
| T1 | T2 |
| * Đọc danh sách đặt bàn |  |
|  | * **Đọc danh sách đặt bàn.** |
| * Cập nhật tên khách hàng đã đặt bàn. |  |
|  | * **Cập nhật tên khách hàng đã đặt bàn.** |
| * Thực hiện xong |  |
|  | * **Thực hiện xong** |

* T1 đọc danh sách bàn đã được đặt, và cập nhật tên khách hàng của bàn số 3, cùng lúc đó T2 đọc danh sách bàn đã được đặt và cập nhật tên khách hàng khác cũng trên cùng bàn số 3, cùng bàn mà T1 đã sữa. Như vậy sau khi thực hiện xong, dữ liệu của T1 không giống như dữ liệu mà mình đã thay đổi trước đó. Lý do: vì T2 đã ghi đè lên dữ liệu của T1, nên dữ liệu của T1 bị mất khi cập nhật.
* ***Sữa lỗi:***

|  |  |
| --- | --- |
| T1 | T2 |
| * begin tran * select \* from DatBan with(rowlock,xlock)   where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,GETDATE(),101)  and TrangThai=0 and maBan=3   * waitfor delay '0:0:05' | * **begin tran** |
|  | * **select \* from DatBan**   **where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,GETDATE(),101) and TrangThai=0 and maBan=3**   * **waitfor delay '0:0:5'** |
| * update Datban set maKhachHang=123654789 where MaBan=3 |  |
|  | * **update Datban set maKhachHang=234234 where MaBan=3** |
| commit tran |  |
|  | **commit tran** |

* T1 đọc danh sách bàn đã được đặt với khóa *(rowlock,xlock)* nghĩa là đặt khóa ghi cho các dòng mà T1 muốn đọc, và cập nhật tên khách hàng của bàn số 3, cùng lúc đó T2 đọc bàn mà T1 đang đọc, T2 sẽ không được đọc, sẽ phải chờ cho đến khi T1 hoàn thành xong thì T2 mới thực hiện.

#### Unrepeatable read: Nghiệp vụ Đặt bàn ăn.

* ***Kịch bản:***

|  |  |
| --- | --- |
| T1 | T2 |
| * Đọc danh sách đặt bàn |  |
|  | * **Sửa tình trạng bàn đã đặt bàn thành bàn đang sử dụng** |
| * Đọc danh sách đặt bàn |  |

* T1 đọc danh sách đặt bàn lần 1, T2 cập nhật tình trạng bàn từ được đặt thành đang sử dụng, tiếp đó T1 đọc lại danh sách đặt bàn lần 2. Thì ta thấy rằng đọc lần 1 của T1 sẽ có kết quả khác với lần đọc thứ 2. Lần đọc 1 có số dòng nhiều hơn lần đọc 2.
* ***Sữa lỗi:***

|  |  |
| --- | --- |
| T1 | T2 |
| * begin tran * select MaBan from DatBan *with (rowlock,repeatableread)*   where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,  GETDATE(),101) and TrangThai=0   * waitfor delay '0:0:05' |  |
|  | * **begin tran** * **update DatBan set TrangThai=1**   **where MaNH=1 and MaBan=4 and ThoiGianDen=CONVERT(nvarchar,GETDATE(),101)**   * **commit tran** |
| * select MaBan from DatBan   where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,  GETDATE(),101) and TrangThai=0   * commit tran |  |

* T1 đọc danh sách đặt bàn lần 1 với khóa *(rowlock,repeatableread)* nghĩa là khi T1 đọc sẽ giữ khóa đọc những dòng T1 cần đọc cho đến hết transaction.Lúc này T2 thực hiện việc cập nhật dòng dữ liệu mà T1 đọc, T1 đang giữ khóa đọc nên T2 sẽ không cấp được khóa Ghi. T2 muốn thực hiện phải chờ T1 thực hiện xong thì T2 mới thực hiện được.

#### Phantom: Nghiệp vụ Đặt bàn ăn.

* ***Kịch bản:***

|  |  |
| --- | --- |
| T1 | T2 |
| * Đọc danh sách đặt bàn |  |
|  | * **Thêm bàn đặt** |
| * Đọc danh sách đặt bàn |  |

* T1 đọc danh sách đặt bàn lần 1, T2 thực hiện thao tác thêm 1 bàn đặt, tiếp đó T1 đọc lại danh sách đặt bàn lần 2. Thì ta thấy rằng đọc lần 1 của T1 sẽ có kết quả khác với lần đọc thứ 2. Lần đọc 1 có số dòng ít hơn lần đọc 2.
* ***Sữa lỗi:***

|  |  |
| --- | --- |
| T1 | T2 |
| * begin tran * select MaBan from DatBan with (rowlock,serializable)   where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,  GETDATE(),101) and TrangThai=0   * waitfor delay '0:0:05' |  |
|  | * **begin tran** * **insert into DatBan (maNH,MaBan,MaKhachHang,ThoiGianDat,**   **TrangThai,ThoiGianDen,sdt)**  **values(1,2,123456756,'05/10/2012',0,CONVERT(nvarchar,GETDATE(),101),234234)**   * **commit tran** |
| * select MaBan from DatBan   where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,  GETDATE(),101) and TrangThai=0   * commit tran |  |

* T1 đọc danh sách đặt bàn lần 1 với khóa *(rowlock,* *serializable)* nghĩa là khi T1 đọc sẽ giữ khóa đọc những dòng T1 cần đọc cho đến hết transaction.Lúc này T2 thực hiện việc Thêm dữ liệu, T1 đang giữ khóa đọc nên T2 sẽ không cấp được khóa Ghi. T2 muốn thực hiện phải chờ T1 thực hiện xong thì T2 mới thực hiện được.

#### Deadlock: Nghiệp vụ Xếp bàn ăn cho khách.

* ***Kịch bản:***

|  |  |
| --- | --- |
| T1 | T2 |
| * Đọc danh sách bàn ăn |  |
|  | * **Đọc danh sách bàn ăn** |
| * Cập nhật tình trạng bàn ăn |  |
|  | * **Cập nhật tình trạng bàn ăn** |

* T1 đọc danh sách bàn ăn và giữ khóa đọc đến hết giao tác, T2 cũng đọc danh sách bàn ăn và giữ khóa đọc đến hết giao tác. T1 cập nhật tình trạng bàn ăn nên phải chờ cho T2 nhả khóa đọc trên bảng bản ăn, cùng lúc đó T2 cũng cập nhật tình trạng bàn ăn và cũng phải chờ cho đến khi T1 nhả khóa đọc trong bảng bàn ăn. Nên cả hai cùng chờ nhau gây lỗi conversion deadlock
* ***Sữa lỗi:***

|  |  |
| --- | --- |
| T1 | T2 |
| * SET TRANSACTION ISOLATION LEVEL repeatable read * begin tran * select \* from DatBan with (rowlock,xlock)   where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,  GETDATE(),101) and TrangThai=0   * waitfor delay '0:0:05' |  |
|  | * **SET TRANSACTION ISOLATION LEVEL**   **repeatable read**   * **begin tran** * **select \* from DatBan with (rowlock,xlock)**   **where MaNH=1 and ThoiGianDen=CONVERT(nvarchar,**  **GETDATE(),101) and TrangThai=0**   * + **waitfor delay '0:0:5'** |
| * + update Datban set TinhTrang=1 where MaBan=2 |  |
| * commit tran | * **update Datban set TinhTrang=1 where MaBan=2** |
|  | * **Commit tran** |

* T1 đọc danh sách bàn ăn và giữ khóa ghi đến hết giao tác, T2 cũng đọc danh sách bàn ăn nhưng T2 phải chờ cho đến khi T1 hoàn thành giao tác, nên sẽ không xảy ra hiện tượng conversion Deadlock.

## 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  go |
| 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  go |  |

###### Nguyên nhân tranh chấp:

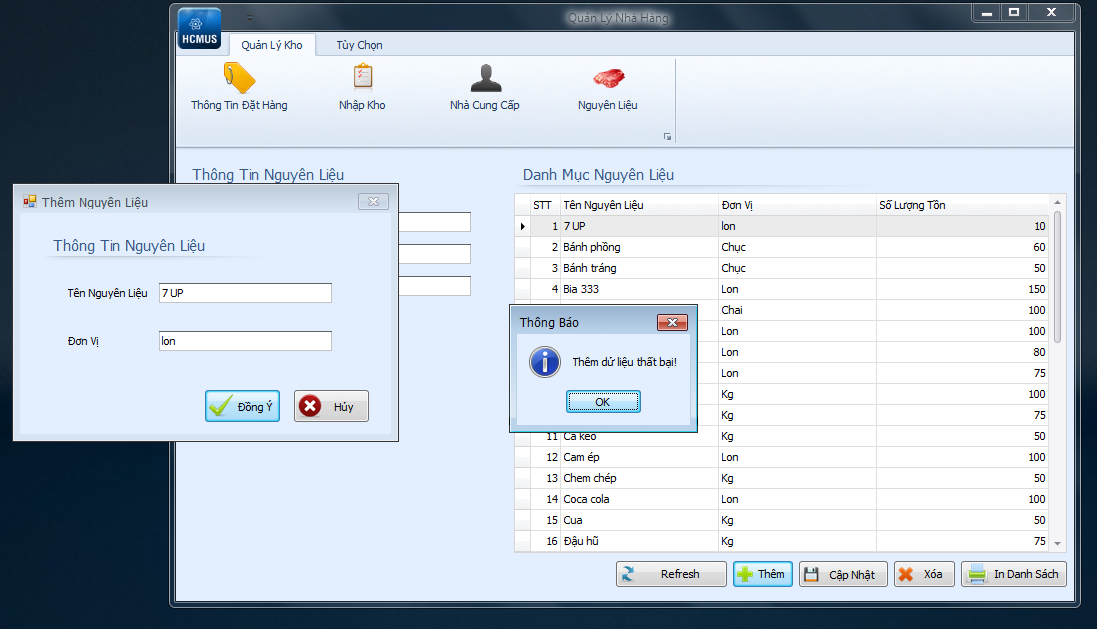
Với mức cô lập **read uncommitted** Giao tác T2 không xin khóa đọc khi lấy dử liệu, vì vậy khi T1 đang ghi dử liệu, nó vẫn có thể đọc dử liệu, dẫn đến tranh chấp

###### Cách khắc phục:

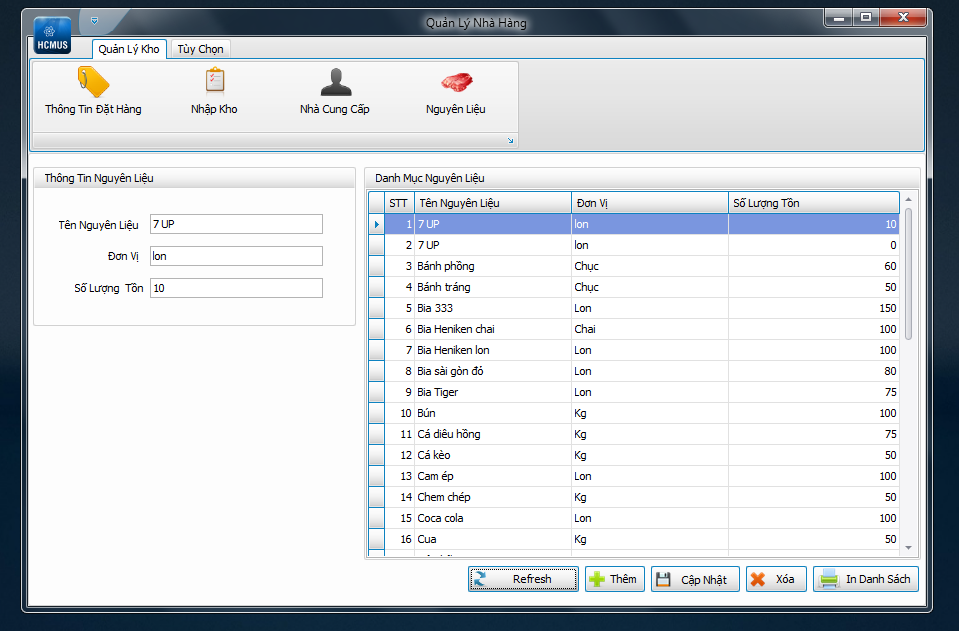
Sử dụng mức cô lập **read committed** :với mức cô lập này khi T2 cần đọc dử liệu nó phải xin khóa đọc,nhưng do T1 đang ghi, nên nó không xin được khóa đọc mà phải đợi đên khi T1 commit T2 mới có thể đọc dử liệu => Khắc phục được lỗi đọc dử liệu rác

###### 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.

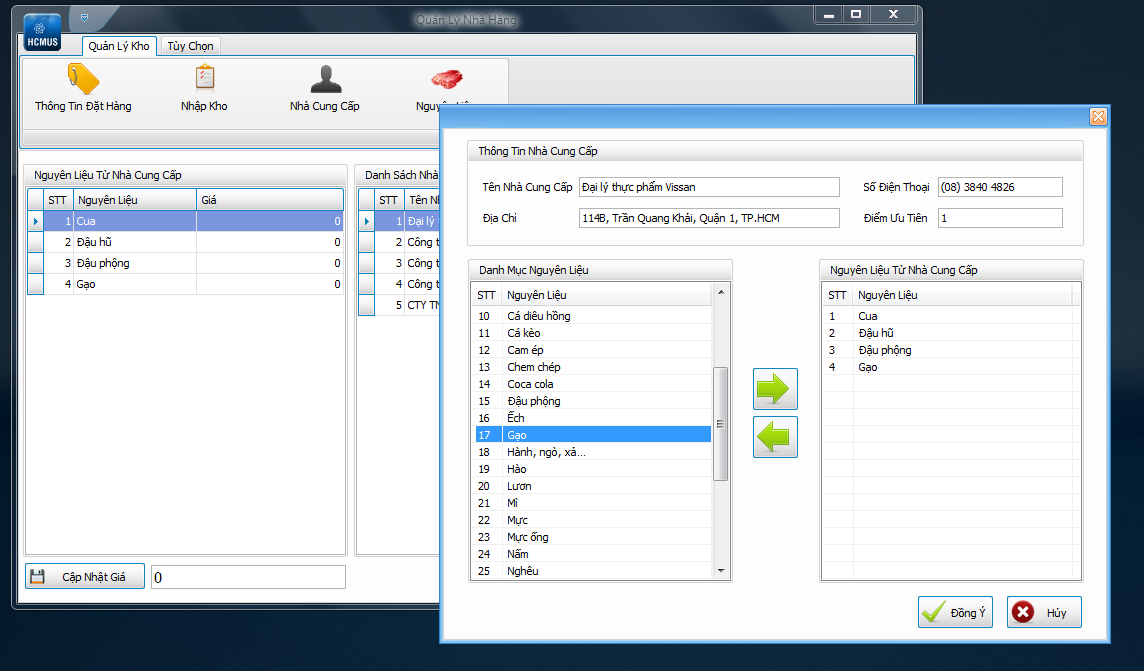
###### Cách khắc phục:

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

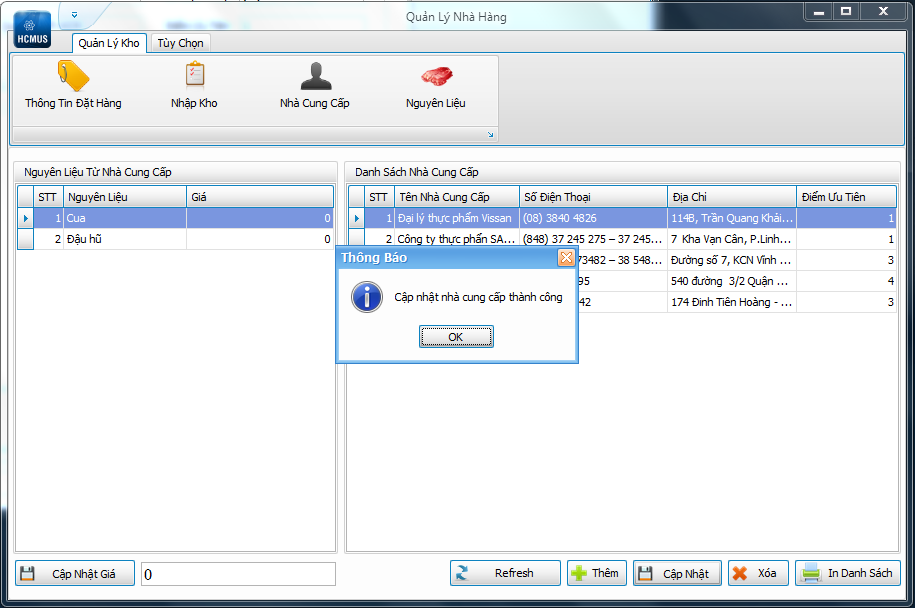
###### Giao diện chương trình:

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.

###### Cách khắc phục:

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

* Kkhông 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

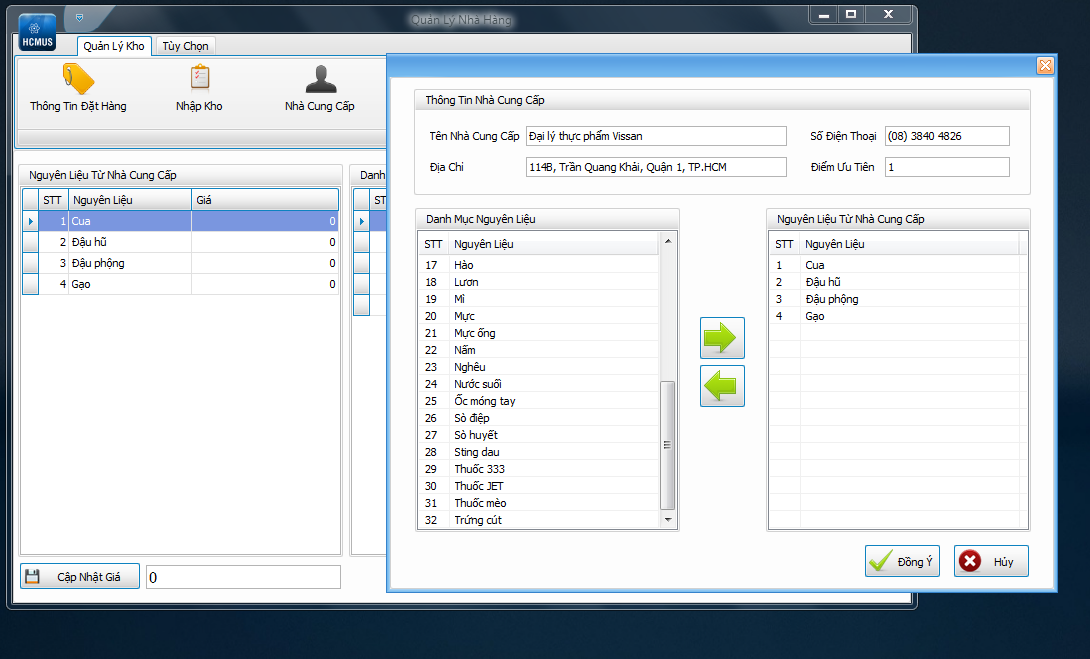
Mức cô lập : **serializable** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và giữ **Shared Lock** này đến hết giao tác .Đồng thời không cho phép insert những dòng dử liệu thỏa mãn điều kiện thiết lập **Shared Lock**

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

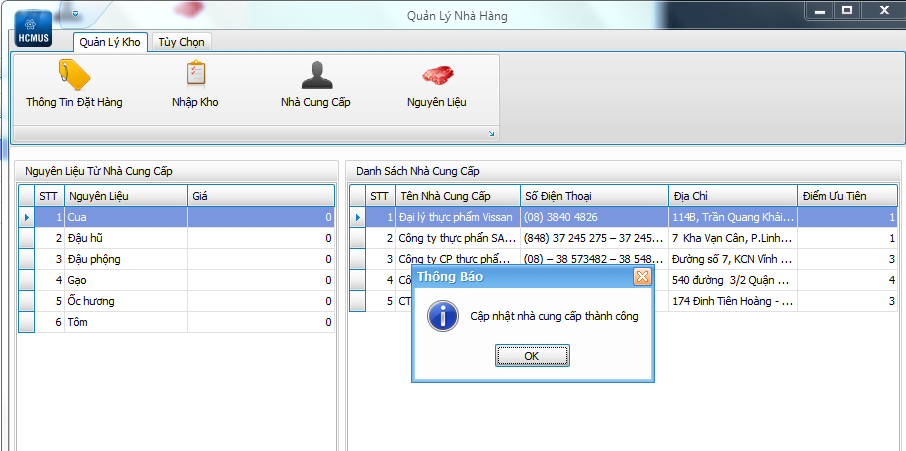
###### Giao diện chương trình:

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:



#### Lostupdate & 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

###### Cách khắc phục:

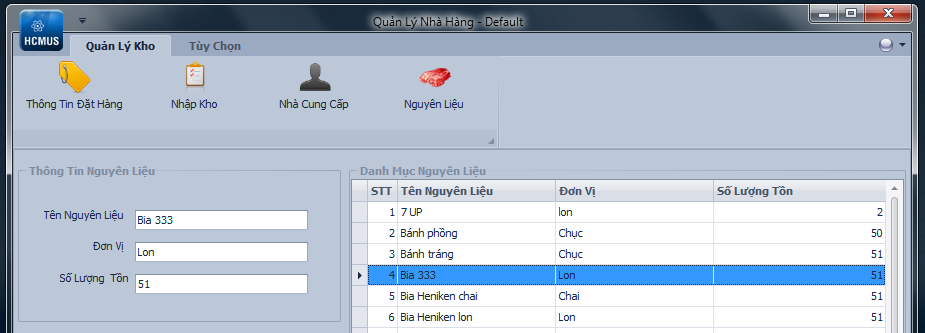
Mức cô lập : **read committed** Tạo **Shared Lock** trên đơn vị dử liệu được đọc và **Shared Lock** được giải phóng ngay sau khi đọc xong dử liệu, vì vậy khi T1 đọc dử liệu xong T2 vẫn được đọc dử liệu này => không giải quyết được tranh chấp

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

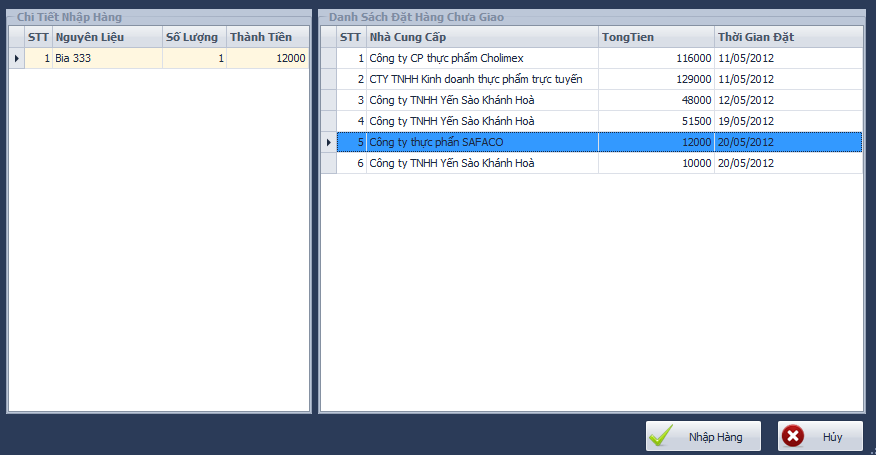
* Để giải quyết tình trạng này ta phải kết hợp Mức cô lập : **repeatable read** và Khóa **Updlock**  để khóa nhưng dòng dữ liệu khi 1 giao tác đang đọc

###### Giao diện chương trình:

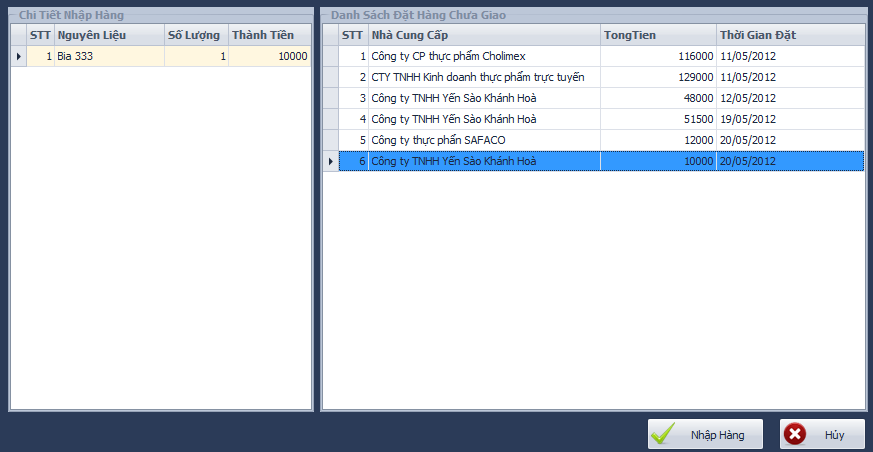
Xét Nguyên Liệu “Bia 333” số lượng tồn hiện tại =51



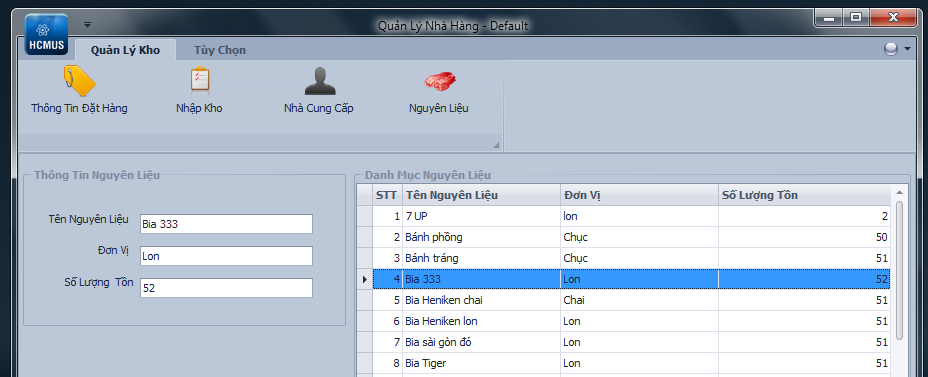
Người dùng 1: Nhập kho đơn đăt hàng từ công ty SAFACO với số lượng là 1=>cập nhật số lượng “Bia 333” tăng lên 1



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



##### Trường hợp 2:

###### Kịch bản :

|  |  |
| --- | --- |
| T1 | T2 |
| - |  |
|  | **-** |
| - |  |

###### Store procedure :

|  |  |
| --- | --- |
| T1 | T2 |
|  |  |
|  |  |
|  |  |
|  |  |

###### Nguyên nhân tranh chấp:

.

###### Cách khắc phục:

###### Giao diện chương trình:

Giao tác T1:

Giao tác T2:

## 1141083 – Nguyễn Minh Nghĩa

### Các trường hợp tranh chấp

#### Drity read

**Tranh chấp giữa giao tác Đọc danh sách Món Ăn (T2) và giao tác Thêm Món Ăn(T1)**

|  |  |  |
| --- | --- | --- |
| # | T1 | T2 |
| 1 | begin tran  insert into MonAn(TenMon,LoaiMon,Gia,DonViTinh,MaNH) values (@TenMon,@LoaiMon,@Gia,@DonViTinh,@MaNH)    if ((select count(\*) from MonAn where TenMon=@TenMon)<>1)  begin  waitfor delay '00:00:05' |  |
| 2 |  | set tran isolation level read committed  begin tran  select MaMon,TenMon,TenLoai,Gia,DonViTinh  from MonAn m,LoaiMonAn lm,NhaHang nh  where m.LoaiMon=lm.MaLoai and m.MaNH=nh.MaNH and nh.MaNH =@MaNH    Commit  go |
| 3 | rollback tran  end    if(@@error<>0)  begin  rollback tran  commit  end  else  c**o**mmit tran  go |  |
|  |  |  |

#### Lost update

**Tranh chấp giữa giao tác Cập nhật Món Ăn (T2) và giao tác Cập nhật Món Ăn (T1)**

|  |  |  |
| --- | --- | --- |
| # | T1 | T2 |
| 1 | set tran isolation level repeatable read  begin tran  if((select count(\*) from MonAn with(updlock) where MaMon=@MaMon)=1)  begin  waitfor delay '00:00:05' |  |
| 2 |  | set tran isolation level repeatable read  begin tran  if((select count(\*) from MonAn with(updlock) where MaMon=@MaMon)=1)  begin  waitfor delay '00:00:05' |
| 3 | update MonAn set TenMon=@TenMon,Gia=@gia,DonViTinh=@DVTinh where MaMon=@MaMon  waitfor delay '00:00:05' |  |
|  |  | update MonAn set TenMon=@TenMon,Gia=@gia,DonViTinh=@DVTinh where MaMon=@MaMon  waitfor delay '00:00:05' |
| 4 | if((select count(\*) from MonAn where TenMon=@TenMon)<>1)  begin  rollback tran  return  end  end  if(@@error <> 0)  begin  rollback tran  return  end    commit tran |  |
| 5 |  | if((select count(\*) from MonAn where TenMon=@TenMon)<>1)  begin  rollback tran  return  end  end  if(@@error <> 0)  begin  rollback tran  return  end    commit tran |

#### Unrepeatable tread

**Tranh chấp giữa giao tác Xóa Món Ăn (T2) và giao tác Tìm kiếm Món Ăn theo loại(T1)**

|  |  |  |
| --- | --- | --- |
| # | T1 | T2 |
| 1 | set tran isolation level repeatable read  begin tran    select @dem=count(\*) from MonAn m,LoaiMonAn lm,NhaHang nh  where m.LoaiMon=lm.MaLoai and m.MaNH=nh.MaNH and nh.MaNH =@maNH and lm.MaLoai=@maLoai    waitfor delay '00:00:05' |  |
| 2 |  | set tran isolation level read committed  begin tran  delete MonAn where MaMon=@maMon  commit  go |
| 3 | select MaMon,TenMon,TenLoai,Gia,DonViTinh  from MonAn m,LoaiMonAn lm,NhaHang nh  where m.LoaiMon=lm.MaLoai and m.MaNH=nh.MaNH and nh.MaNH =@maNH and [lm.MaLoai=@maLoai](mailto:lm.MaLoai=@maLoai)  commit tran |  |
|  |  |  |

#### Phantom

**Tranh chấp giữa giao tác Thêm Món Ăn (T2) và giao tác Tìm kiếm Món Ăn theo loại(T1)**

|  |  |  |
| --- | --- | --- |
| # | T1 | T2 |
| 1 | set tran isolation level serializable  begin tran    select @dem=count(\*) from MonAn m,LoaiMonAn lm,NhaHang nh  where m.LoaiMon=lm.MaLoai and m.MaNH=nh.MaNH and nh.MaNH =@maNH and lm.MaLoai=@maLoai    waitfor delay '00:00:05' |  |
| 2 |  | set tran isolation level read committed  begin tran  insert into MonAn(TenMon,LoaiMon,Gia,DonViTinh,MaNH) values (@TenMon,@LoaiMon,@Gia,@DonViTinh,@MaNH)    commit  go |
| 3 | select MaMon,TenMon,TenLoai,Gia,DonViTinh  from MonAn m,LoaiMonAn lm,NhaHang nh  where m.LoaiMon=lm.MaLoai and m.MaNH=nh.MaNH and nh.MaNH =@maNH and [lm.MaLoai=@maLoai](mailto:lm.MaLoai=@maLoai)  commit tran |  |
|  |  |  |

#### deadlock

## 1141129 – Bùi Trung Tân

### Các trường hợp tranh chấp

#### Drity read

###### Kịch bản :

|  |  |
| --- | --- |
| T1: thực hiện chuyển bàn ăn A sang B | T2: ghép bàn |
| Cập nhật trạng thái bàn A thành ‘trống’ |  |
|  | Lấy danh sách bàn trống để chuyển bàn.  (hoặc ghép bàn, xếp bàn…) |
| Kiểm tra bàn B có đang trống hay không:   * Nếu không còn trống (do bị giao tác khác đã thay đổi trạng thái bàn ăn này)   =>Rollback: trả lại trạng thái ban ‘đang sử dụng’ cho bàn A   * Ngược lại thì cập nhật trạng thái bàn B thành ‘đang sử dụng’.   =>Hoàn tất nghiệp vụ |  |

* Tại thời điểm T2 đã đọc dữ liệu, được trạng thái bàn A và B đều là ‘trống’ và được đọc lên.
  + Nếu T1 bị rollback, trạng thái bàn A bị gán lại thành ‘đang sử dụng’, T2 đã đọc sai dữ liệu khi đọc được bàn A.
  + Nếu T1 thành công, trạng thái bàn B bị gán thành ‘đang sử dụng’, vậy T2 đã đọc sai dữ liệu khi đọc được bàn B.

###### Stored Procedure :

|  |  |
| --- | --- |
| T1 | T2 |
| create proc sp\_ChuyenBan @kq int out, @mabanOLD int, @mabanNEW int, @mahd int  as  begin  begin tran t1  SET @kq = 1    UPDATE BanAn SET TrangThai = 0  WHERE MaBan = @mabanOLD    waitfor delay '0:0:10' |  |
|  | create proc sp\_selectBanAn @trangthai int, @manh nchar(10)  as  begin tran  IF (@trangthai = -1)  SELECT \* FROM BanAn  WHERE MaNH = @manh  ELSE  SELECT \* FROM BanAn  WHERE TrangThai = @trangthai  AND MaNH = @manh  commit tran |
| IF( (SELECT TrangThai FROM BanAn  WHERE MaBan = @mabanNEW) = 0)  begin  EXEC dbo.sp\_UpdateTrangThaiBanAn @mabanNEW, 1  UPDATE HoaDon SET MaBan = @mabanNEW WHERE MaHD = @mahd  commit tran t1  end  ELSE  begin  SET @kq = 0  rollback tran t1  end  end |  |

###### Nguyên nhân tranh chấp :

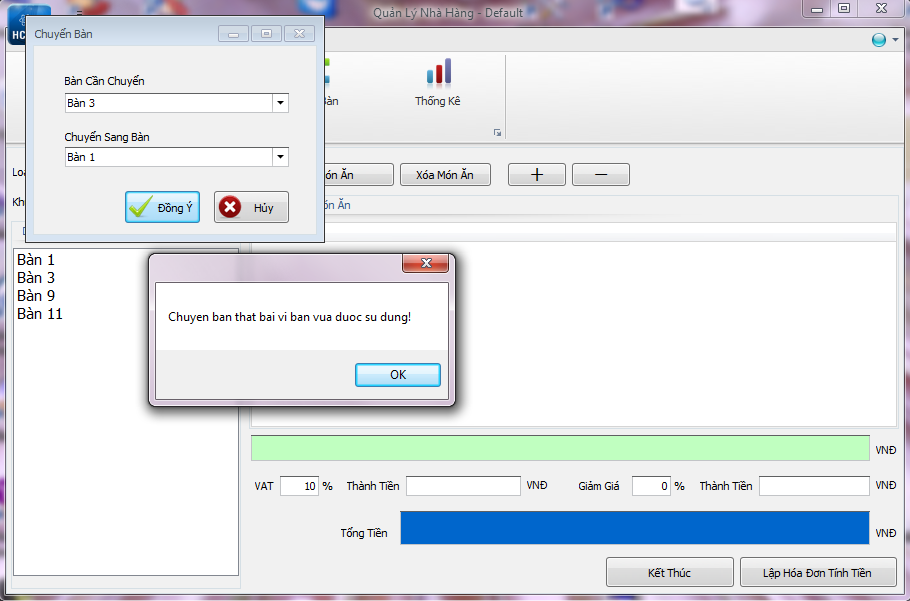
Với mức cô lập Read Uncommited, T2 đọc mà không xin khoá đọc. Như vậy bàn ăn A đang có khoá ghi (T1 update trạng thái) và T2 vẫn có thể đọc được dữ liệu, nên T2 có thể bị đọc sai.

###### Cách khắc phục:

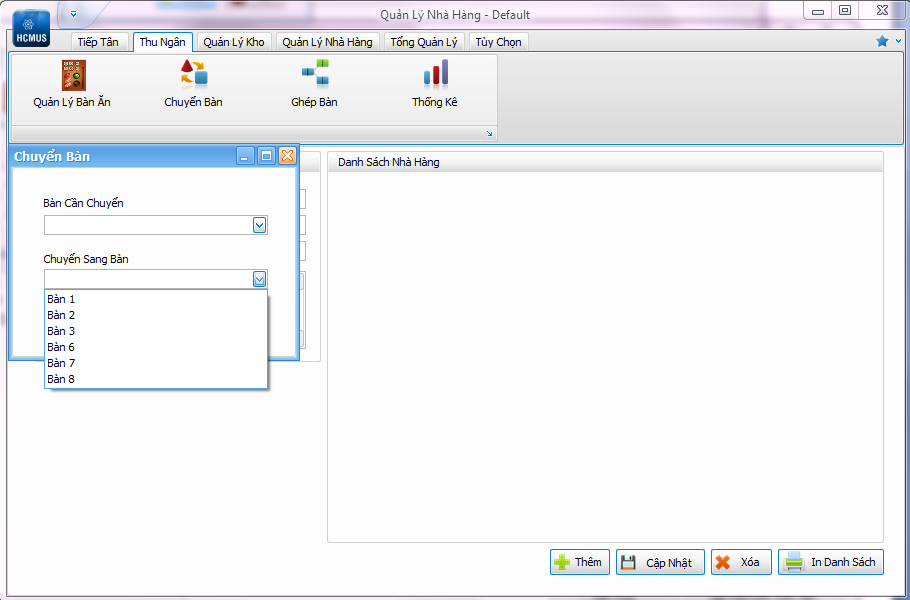
Thiết lập mức cô lập Read Commited cho T1 và T2. T2 phải xin khoá đọc trên bàn A và bàn B, như vậy T2 phải đợi T1 hoàn tất.

**Giao diện chương trình bị Dirty Read:**

Giao tác T1



Giao tác T2



Xuất hiện dữ liệu rác:

Bàn số 3

#### Lost update

###### Kịch bản :

|  |  |
| --- | --- |
| T1: thực hiện cộng thêm 1 vào số lượng của món ăn A trong chi tiết hoá đơn rồi hiển thị ra giao diện | T2: thực hiện trừ đi 1 vào số lượng của món ăn A trong chi tiết hoá đơn rồi hiển thị ra giao diện |
| Đọc số lượng hiện tại (n) của món ăn A vào biến  Cộng 1 vào biến |  |
|  | Đọc số lượng hiện tại (n) của món ăn A vào biến  Cộng 1 vào biến |
| Cập nhật lại giá trị của biến vào số lượng của món A  Số lượng mới: n + 1 |  |
| (Tại đây có một giao tác T3 đọc ra số lượng của món A) |  |
|  | Cập nhật lại giá trị của biến vào số lượng của món A  Số lượng mới: n - 1 |

* Như vậy sau khi T1 và T2 thực hiện xong, số lượng cúa món A là n – 1. Trong khi đúng ra số lượng món A phải là n.
  + Dữ liệu n + 1 đã bị ghi đè bởi n – 1
  + T3 bị dirty read vì ngay sau khi đọc xong thì giá trị vừa đọc bị thay đổi.

###### Stored Procedure :

|  |  |
| --- | --- |
| T1 | T2 |
| create proc sp\_UpdateSoLuongCTHD\_1 @mahd int, @mamon int, @soluong int  as  begin tran  SET TRAN ISOLATION LEVEL  READ UNCOMMITTED    declare @soluong\_hientai int  SELECT @soluong\_hientai = SoLuong  FROM ChiTietHoaDon  WHERE MaHD = @mahd  AND MaMon = @mamon  SET @soluong\_hientai = @soluong\_hientai + @soluong    waitfor delay '0:0:7' |  |
|  | create proc sp\_UpdateSoLuongCTHD\_1 @mahd int, @mamon int, @soluong int  as  begin tran  SET TRAN ISOLATION LEVEL  READ UNCOMMITTED    declare @soluong\_hientai int  SELECT @soluong\_hientai = SoLuong  FROM ChiTietHoaDon  WHERE MaHD = @mahd  AND MaMon = @mamon  SET @soluong\_hientai = @soluong\_hientai + @soluong  waitfor delay '0:0:7' |
| UPDATE ChiTietHoaDon SET SoLuong = @soluong\_hientai WHERE MaHD = @mahd AND MaMon = @mamon  commit tran |  |
|  | UPDATE ChiTietHoaDon SET SoLuong = @soluong\_hientai WHERE MaHD = @mahd AND MaMon = @mamon  commit tran |

###### Nguyên nhân tranh chấp :

Sau khi T1 đọc dữ liệu, T2 cũng có thể đọc dữ liệu (Shared-lock không có tranh chấp khi cấp trên cùng một đơn vị dữ liệu). Như vậy cả T1 và T2 đều đọc được trạng thái nguyên thuỷ của dữ liệu, do đó khi update vào dữ liệu đã đọc được thì đơn vị dữ liệu chỉ nhận được giá trì từ T1 hoặc T2.

T3 đọc dữ liệu ngay sau khi T1 update mà không biết rằng T2 vẫn còn update lên cùng đơn vị dữ liệu đó, nên T3 bị dirty read.

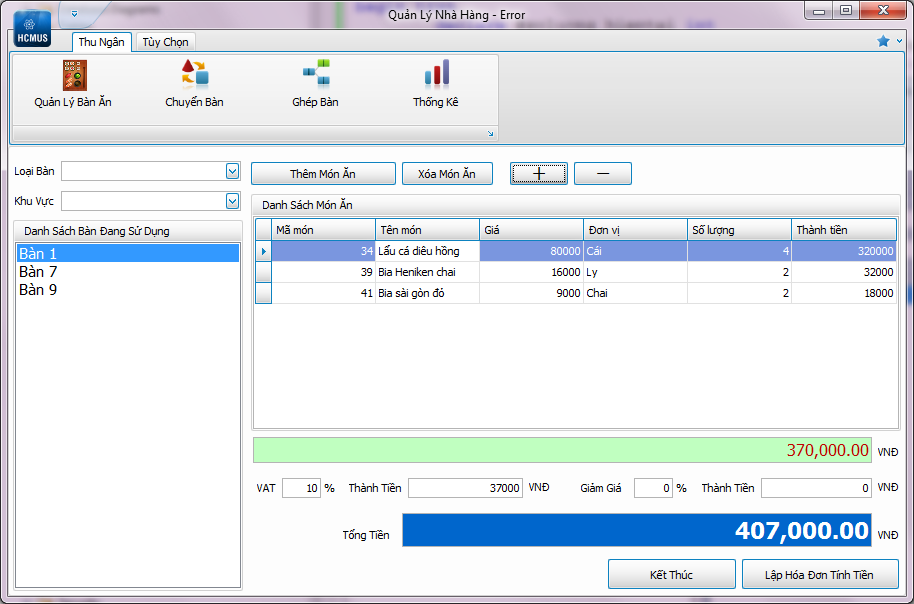
###### Cách khắc phục:

Nâng cấp khoá đọc thành Uplock, như vậy T2 phải đợi T1 hoàn tất thì mới có thể đọc và cập nhất giá trị. Theo kịch bản trên thì T3 bắt đầu sau T2, thiết lập mức cô lập Read Committed cho T3 thì T3 phải đợi T2 hoàn tất mới có thể đọc giá trị và không bị Dirty Read.

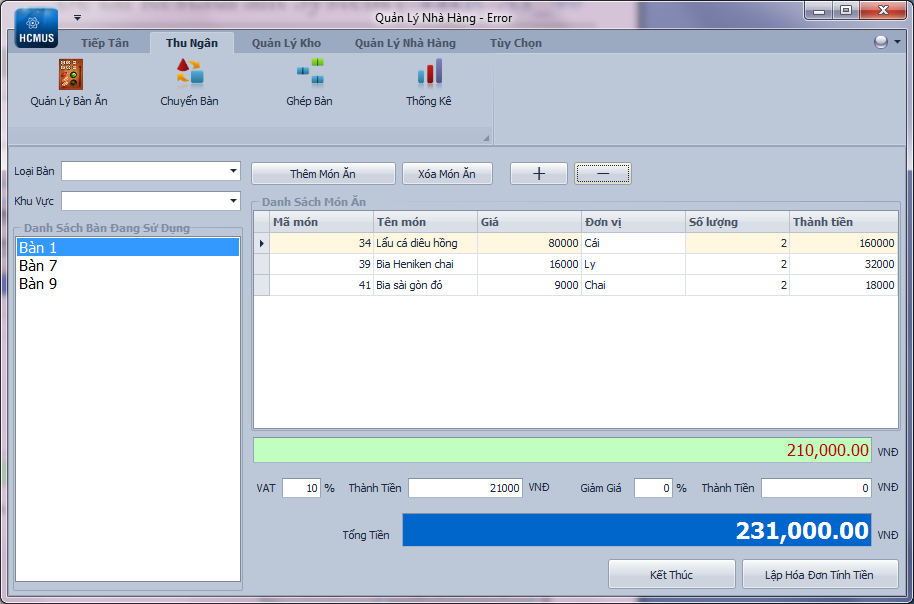
###### Giao diện chương trình bị Lost Update:

Số lượng của mã món 34 ban đầu bằng 3. T1 chạy trước tăng 1, T2 chạy sau giảm 1. Như vậy số lượng của món 34 sẽ là 2 (bị lost update) nhưng trên giao diện T1 lại hiển thị là 4 do T3 đọc được số lượng của món 34 ngay sau khi T1 ghi 4.

Giao tác T1



Giao tác T2



#### Unrepeatable tread

###### Kịch bản :

|  |  |
| --- | --- |
| T1: thực hiện thêm một món ăn A vào hoá đơn | T2: xoá món ăn A |
| Kiểm tra xem món ăn A có tồn tại hay không (đọc lần 1) |  |
|  | Xoá món ăn A |
| Nếu món ăn A có tồn tại (kiến quả của đọc lần 1) thì   * Đọc lần 2 để lấy ra giá của món ăn A * Thêm thông tin món ăn vào bảng chi tiết hoá đơn |  |

* Tại thời điểm T1 kiểm tra xem món ăn có tồn tại hay không (đọc lần 1).
  + Giả sử lần đọc đầu tiên trả ra kết quả là có, các câu lệnh bên trong hàm IF được thực hiện
  + Trước khi T1 đọc lần thứ 2, T2 đã xoá món ăn đó, làm cho T1 không thể đọc lại món ăn đó để lấy ra Giá của món ăn.

###### Stored Procedure :

|  |  |
| --- | --- |
| T1 | T2 |
| create proc sp\_insertCTHD @mahd int, @mamon int, @soluong int  as  begin tran  IF( (select COUNT(\*) from MonAn where MaMon = @mamon) = 1 )  begin  waitfor delay '0:0:10' |  |
|  | DELETE MonAn WHERE MaMon = @mamon |
| declare @dongia float  SET @dongia = 0  set @dongia = (select Gia from MonAn where MaMon = @mamon)  INSERT INTO ChiTietHoaDon(MaHD, MaMon, SoLuong, ThanhTien) VALUES(@mahd, @mamon, @soluong, @dongia \* @soluong)  end  commit tran |  |

###### Nguyên nhân tranh chấp:

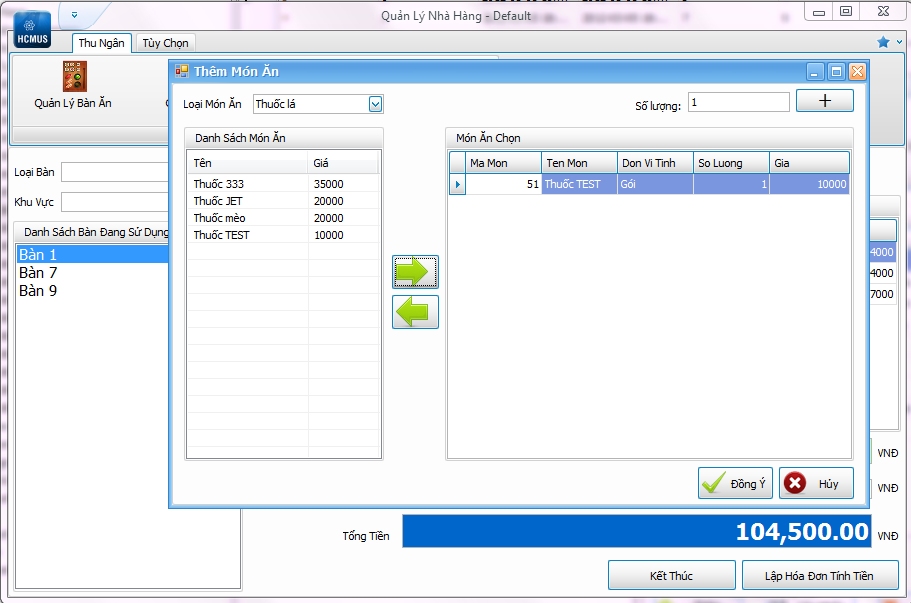
Khi T1 được thiết lập mức cô lập là Read Uncommitted hoặc Read Commited, sau lần đọc thứ nhất, khoá đọc được giải phóng trên món ăn A ngay sau khi đọc xong. Như vậy giao tác T2 có thể thay đổi thông tin món ăn A (xoá, thay đổi điều kiện lọc), làm cho T1 không thể đọc lại món ăn A ở lần đọc thứ hai.

###### Cách khắc phục:

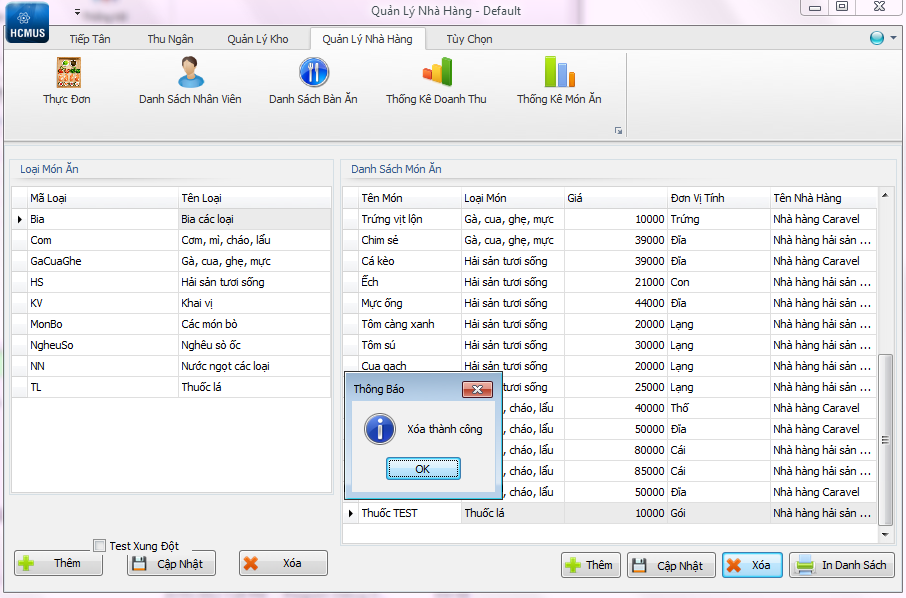
Thiết lập mức cô lập Repeatable Read cho T1. Như vậy ở lần đọc thứ nhất, khoá đọc được giữ trên món ăn A đến khi T1 kết thúc, T2 phải đợi T1 hoàn tất thì mới có thể thực hiện câu lệnh xoá món ăn.

###### Giao diện chương trình bị Unrepeatable Read:

Giao tác T1



Giao tác T2



#### Phantom

#### Deadlock

###### Kịch bản :

|  |  |
| --- | --- |
| T1: cập nhật số lượng cho chi tiết hoá đơn | T2: cập nhật số lượng cho chi tiết hoá đơn |
| Kiểm tra xem món ăn A cần cập nhật số lượng đã có trong chi tiết hoá đơn chưa. |  |
|  | Kiểm tra xem món ăn A cần cập nhật số lượng đã có trong chi tiết hoá đơn chưa. |
| Nếu có rồi thì cập nhật lại số lượng cho món ăn A |  |
|  | Nếu có rồi thì cập nhật lại số lượng cho món ăn A |

* Để không bị Unrepeatable Read, mức cô lập được sử dụng ở T1 và T2 là Repeatable Read. Như vậy xuất hiện Conversion Deadlock khi shared lock được cấp và giữ đến cuối giao tác.

###### Stored Procedure:

|  |  |
| --- | --- |
| T1 | T2 |
| create proc sp\_UpdateSoLuongCTHD @mahd int, @mamon int, @soluong int  as  begin tran  SET TRAN ISOLATION LEVEL  REPEATABLE READ    IF( (SELECT COUNT(\*) FROM ChiTietHoaDon  WHERE MaHD = @mahd  AND MaMon = @mamon) = 1 )  begin  waitfor delay '0:0:7' |  |
|  | create proc sp\_UpdateSoLuongCTHD @mahd int, @mamon int, @soluong int  as  begin tran  SET TRAN ISOLATION LEVEL  REPEATABLE READ    IF( (SELECT COUNT(\*) FROM ChiTietHoaDon  WHERE MaHD = @mahd  AND MaMon = @mamon) = 1 )  begin  waitfor delay '0:0:7' |
| UPDATE ChiTietHoaDon SET SoLuong = @soluong WHERE MaHD = @mahd AND MaMon = @mamon  End  commit tran |  |
|  | UPDATE ChiTietHoaDon SET SoLuong = @soluong WHERE MaHD = @mahd AND MaMon = @mamon  **E**nd  commit tran |

###### Nguyên nhân tranh chấp:

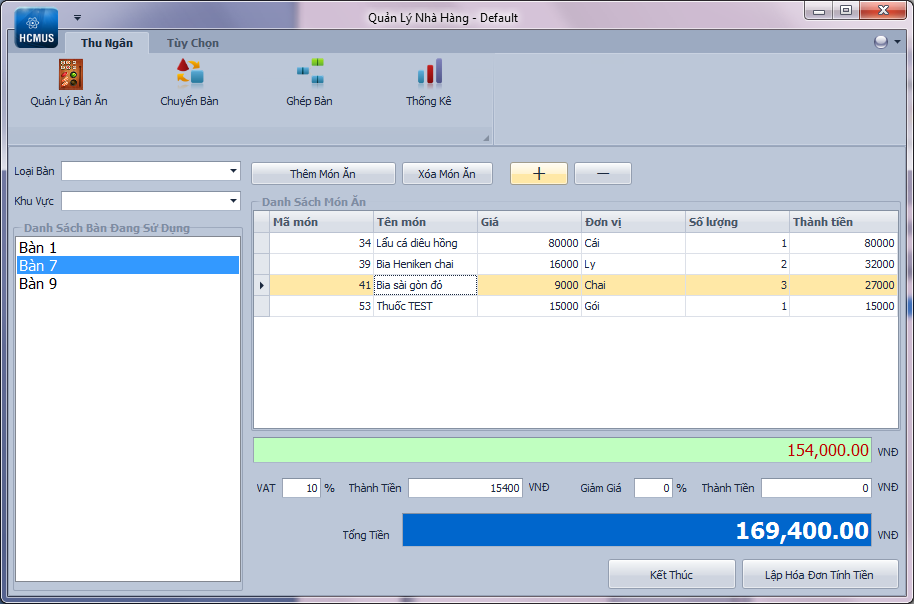
* Để cập nhật số lượng món ăn trong chi tiết hoá đơn, cần đảm bảo trong chi tiết hoá đơn có tồn tại món ăn đó, nên cần thực hiện kiểm tra
* Để đảm bảo sau khi kiểm tra, món ăn không bị thay đổi thông tin (ví dụ: xoá) thì cần thiết lập mức cô lập Repeatable Read. Như vậy ở khoá đọc được cấp và giữ trên món ăn A đến hết giao tác.
* T1 cấp shared lock trên món ăn A và giữ đến hết giao tác. T2 cấp shared lock trên món ăn A và giữ đến hết giao tác. T1 cần cập nhật giữ liệu trên món ăn A, nên phải chờ T2 kết thúc. T2 cũng cần cập nhật giữ liệu trên món ăn A và phải chờ T1 kết thúc.
* Xuất hiện Conversion Deadlock.

###### Cách khắc phục:

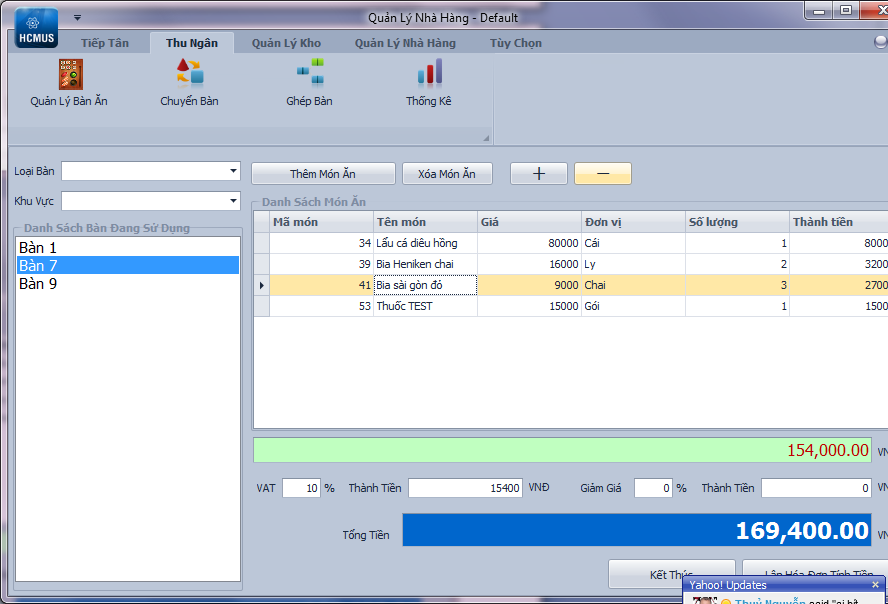
Thiết lập timeout (SET LOCK\_TIMEOUT 20000) để chỉ có một giao tác được thực thi, giao tác còn lại bị huỷ.

###### Giao diện chương trình bị Conversion Deadlock:

Giao tác T1: (tăng số lượng món ăn lên 1)

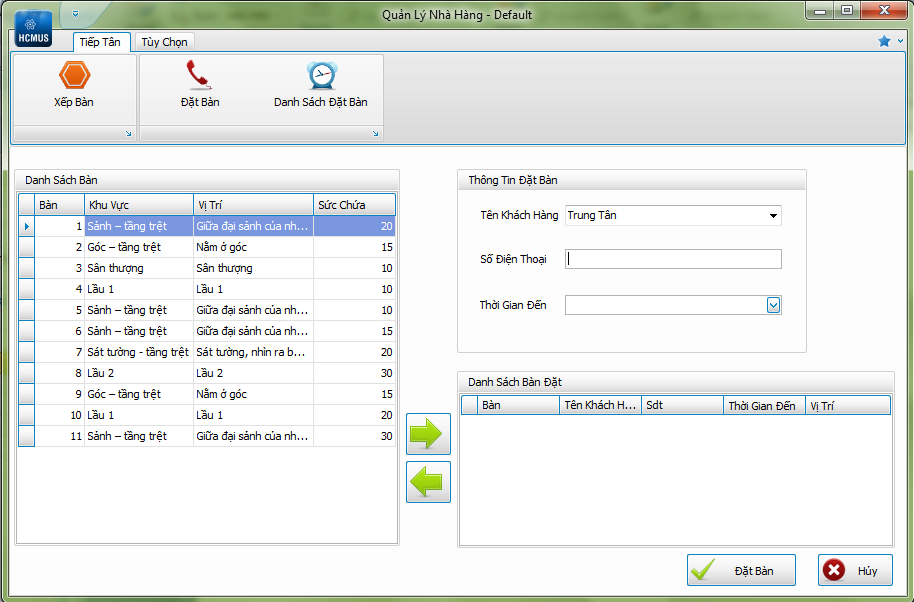
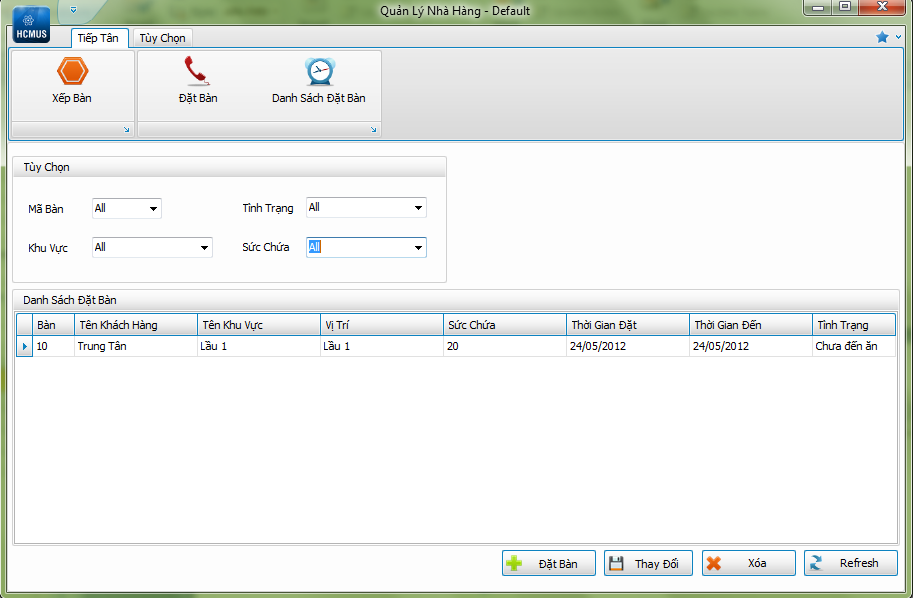


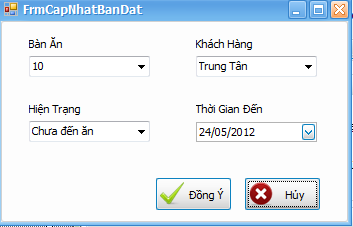
Giao tác T2: (giảm số lượng món ăn đi 1)

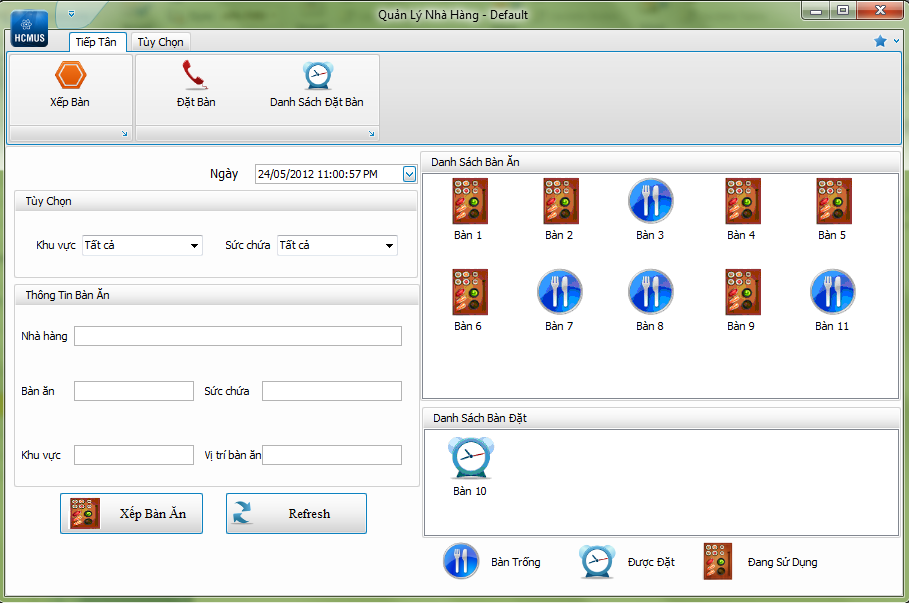


# 3.Các chức năng trong đề tài

## 3.1.Nhân Viên Tiếp Tân

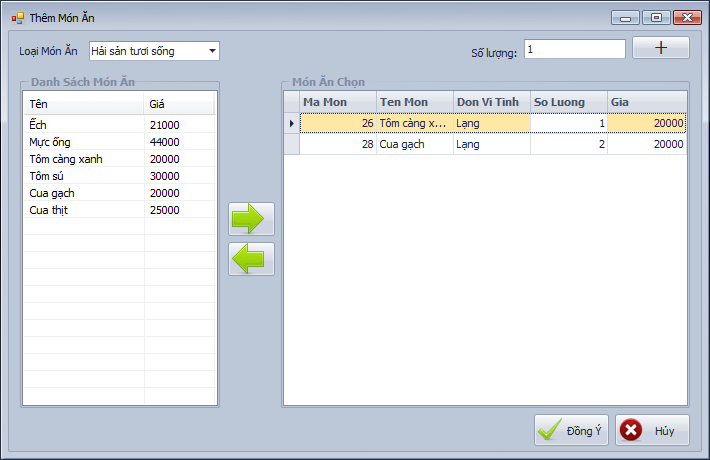
* + 1. **Đặt bàn**
    2. **Danh sách Đặt bàn**



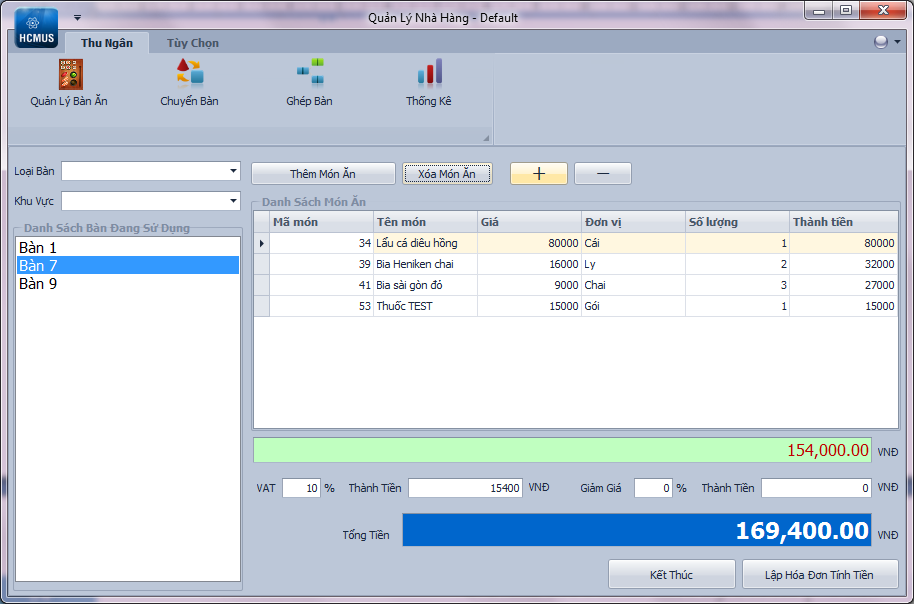
* + 1.  **Xếp bàn ăn cho khách**

## 3.2.Nhân Viên Thu Ngân

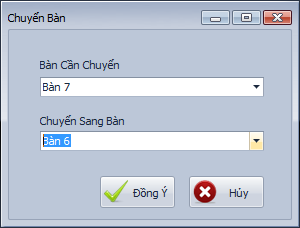
### Thêm món ăn

****

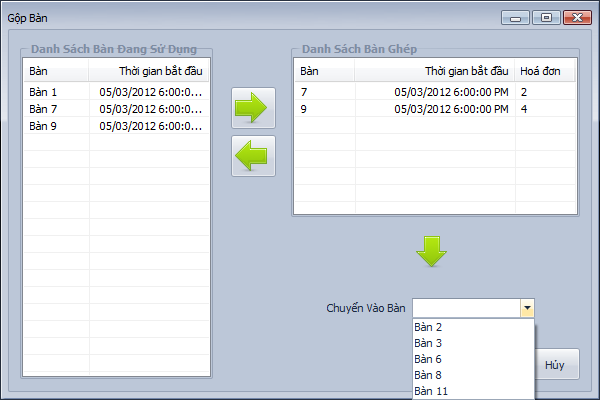
### Xoá món ăn, cập nhật số lượng món ăn

****

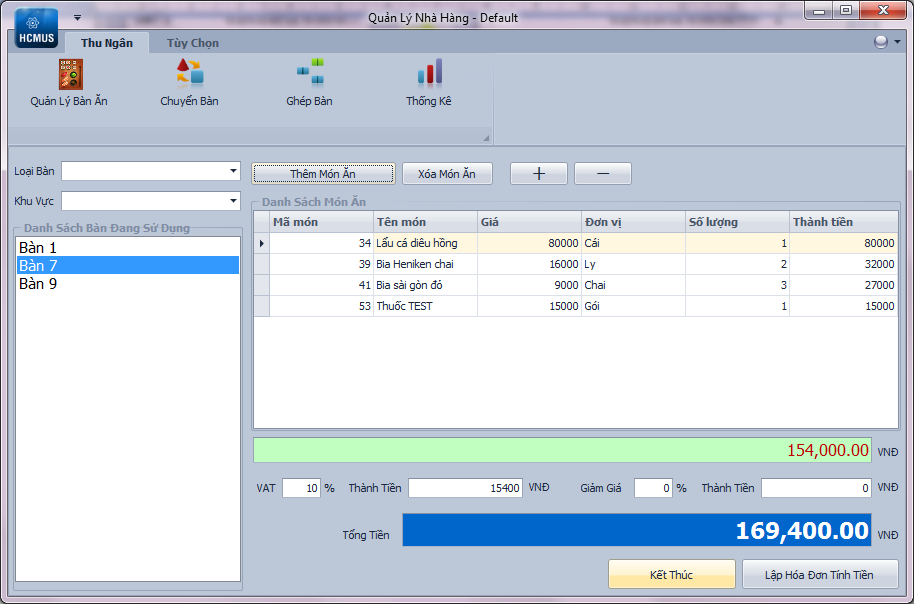
### Chuyển bàn

****

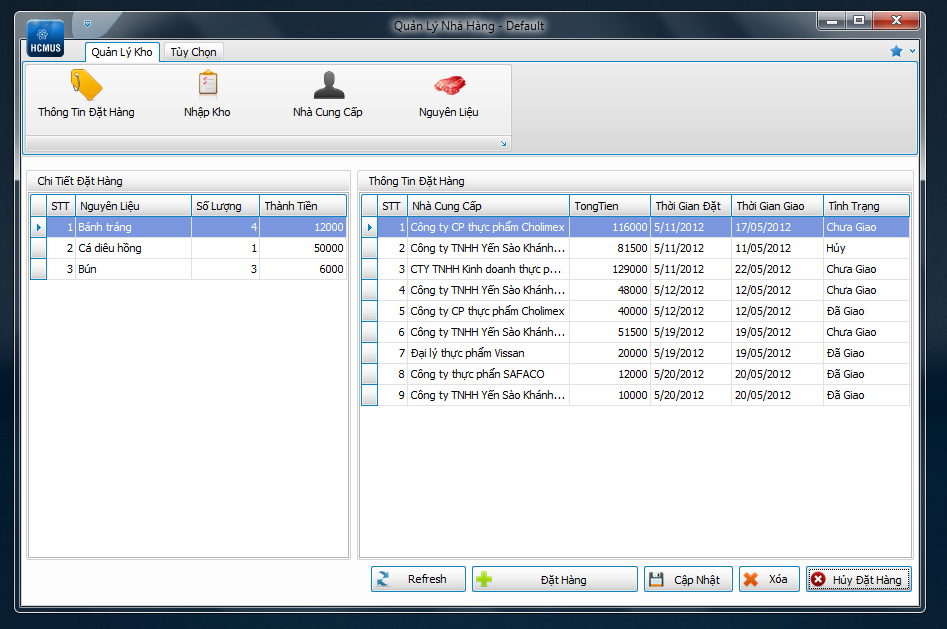
### Ghép bàn

****

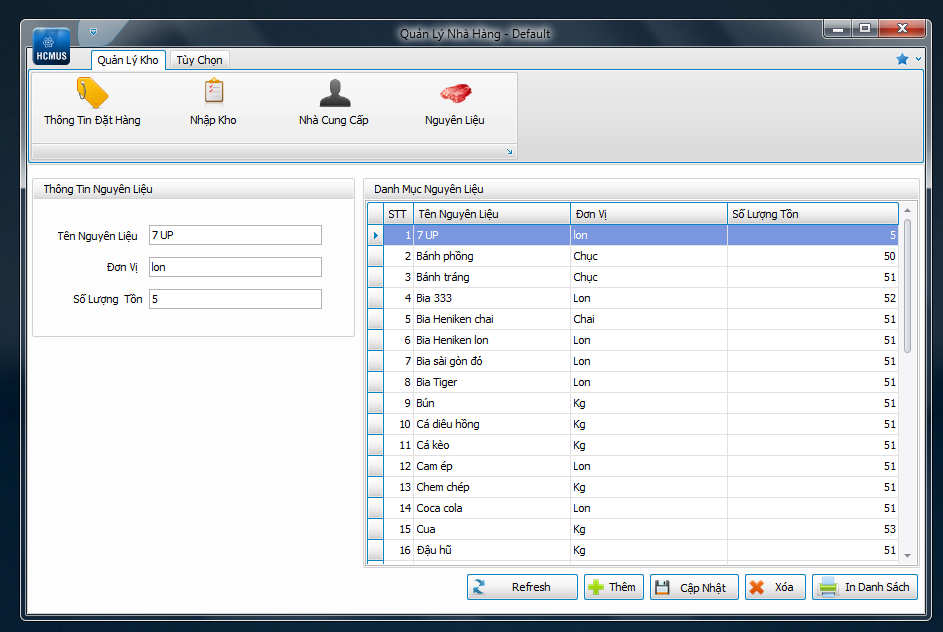
### Kết thúc sử dụng bàn

****

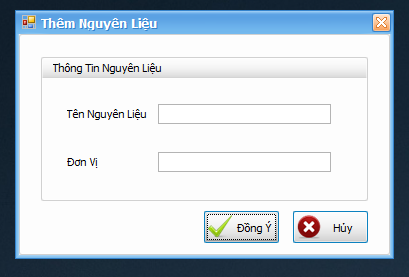
## 3.3. Nhân Viên Quản Lý Kho



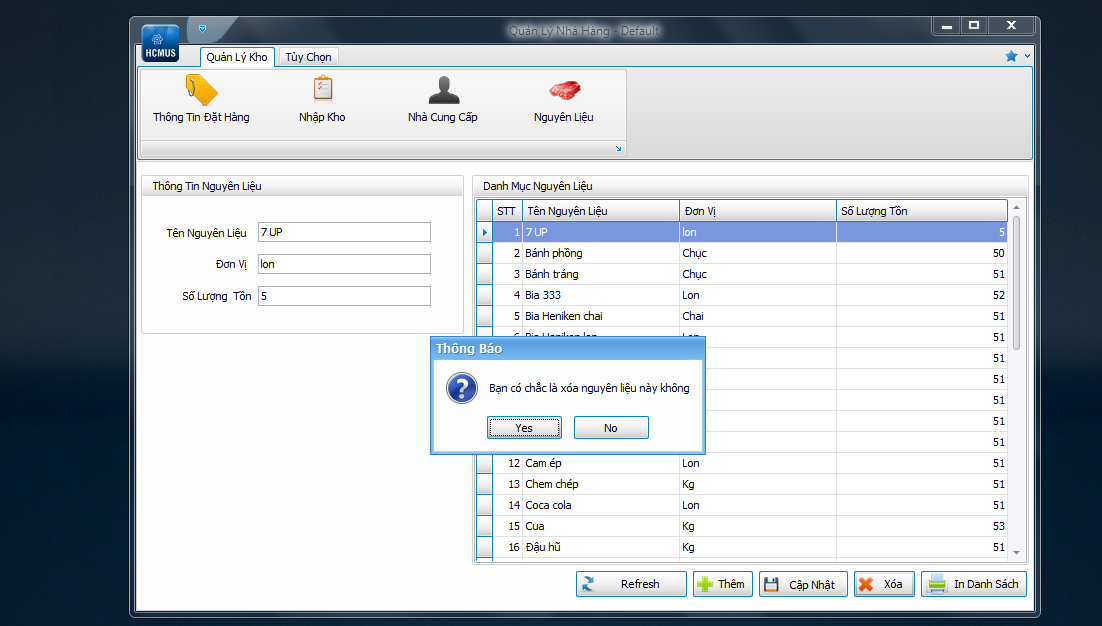
### 3.3.1.Quản Lý Danh Sách Nguyên Liệu



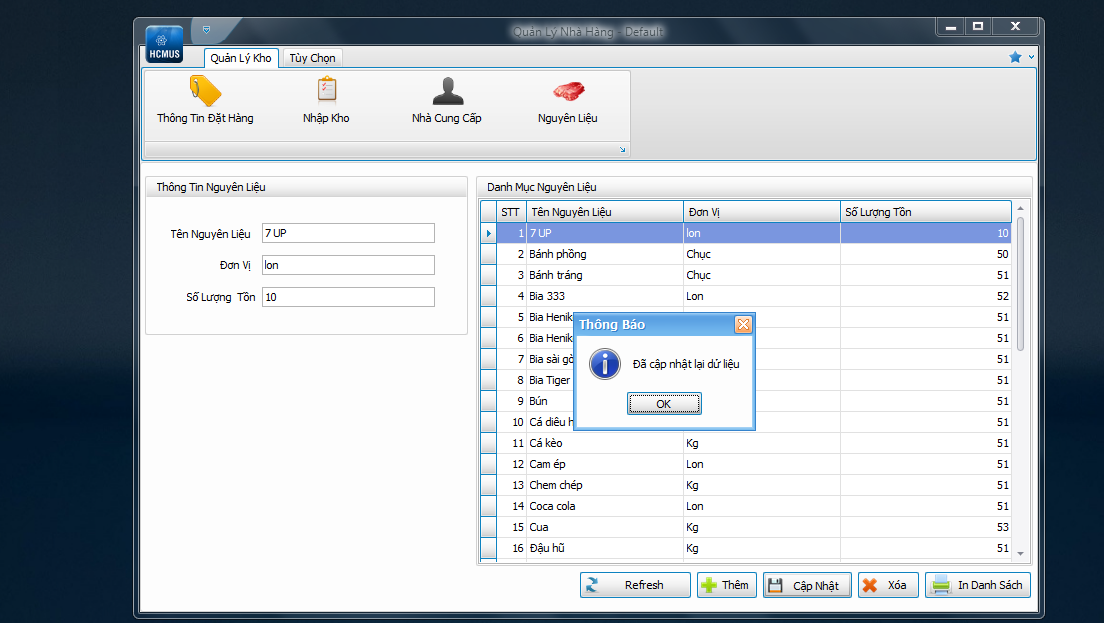
#### 3.3.1.1.Thêm Nguyên Liệu



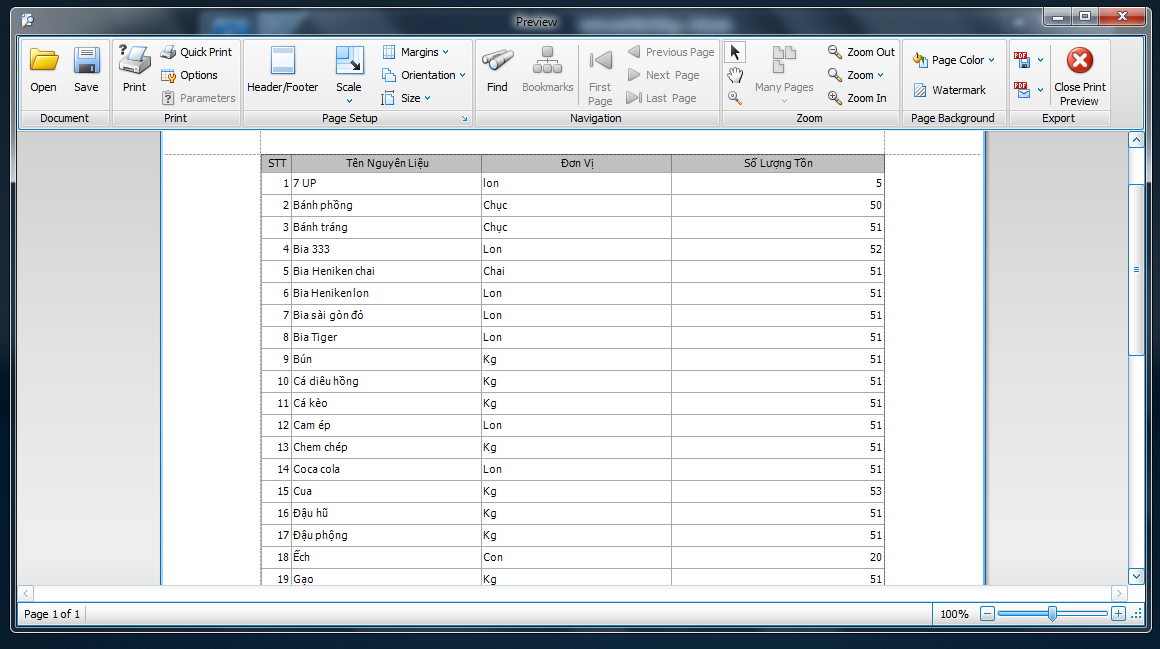
#### 3.3.1.2.Xóa Nguyên Liệu



#### 3.3.1.3.Cập Nhật Nguyên Liệu

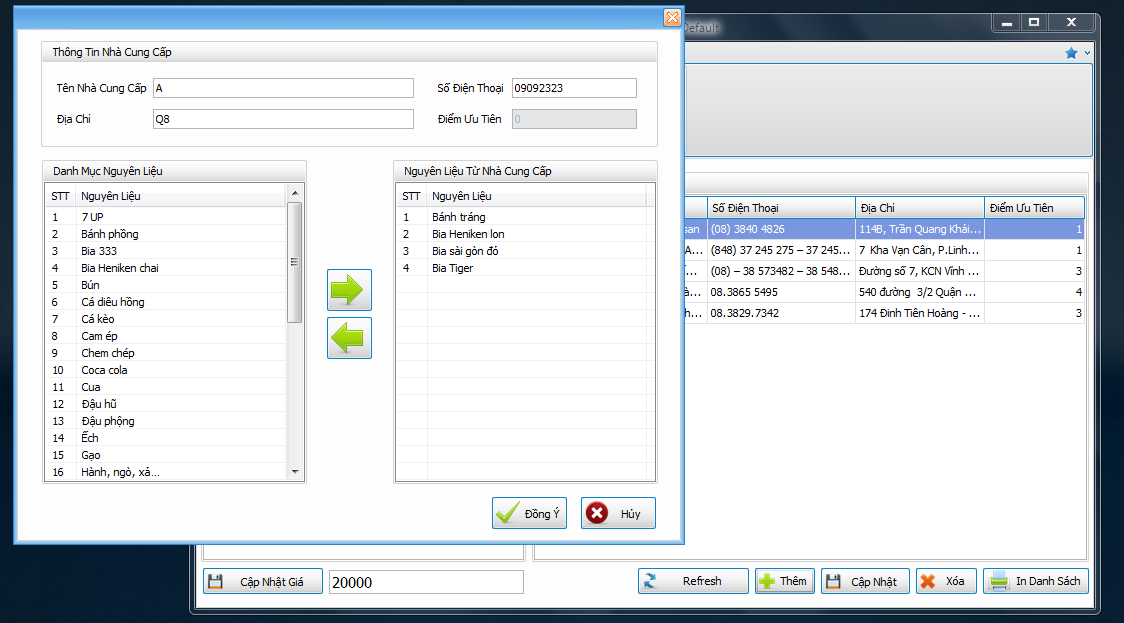


#### 3.3.1.4.In Danh Sách Nguyên Liệu

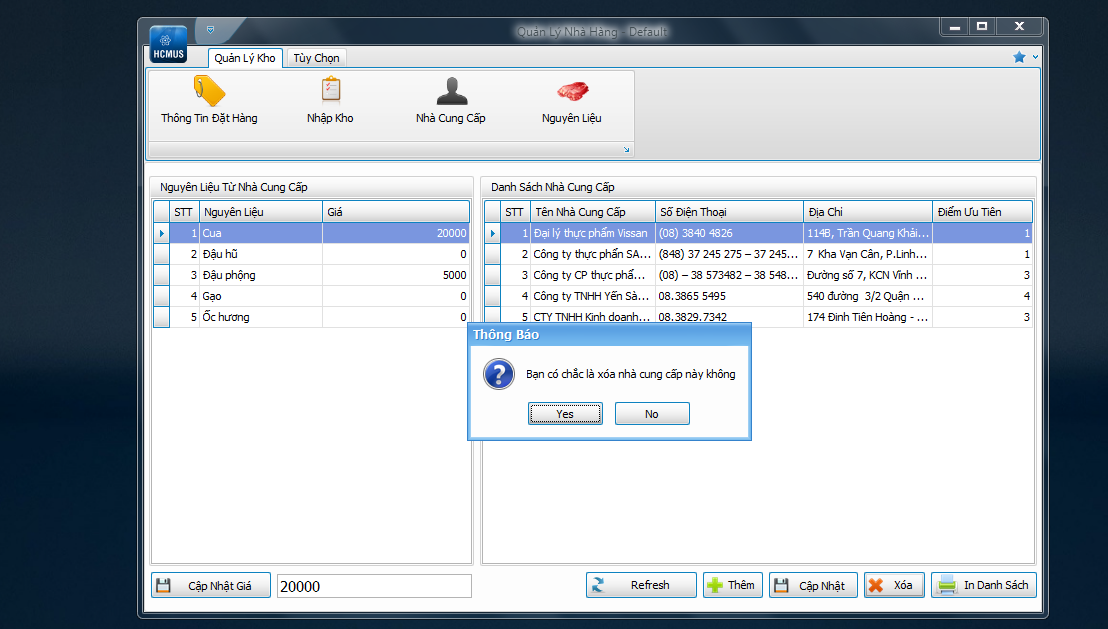


### 3.3.2.Quản Lý Danh Sách Nhà Cung Cấp

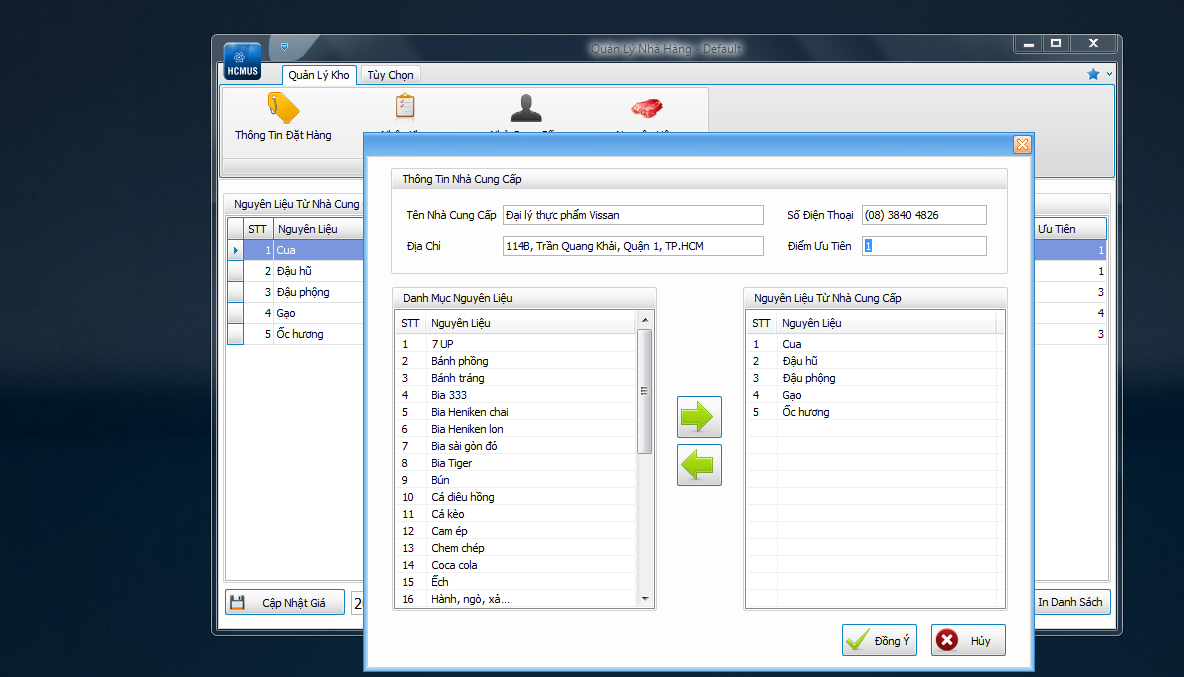
#### 3.3.2.1.Thêm Nhà Cung Cấp Mới



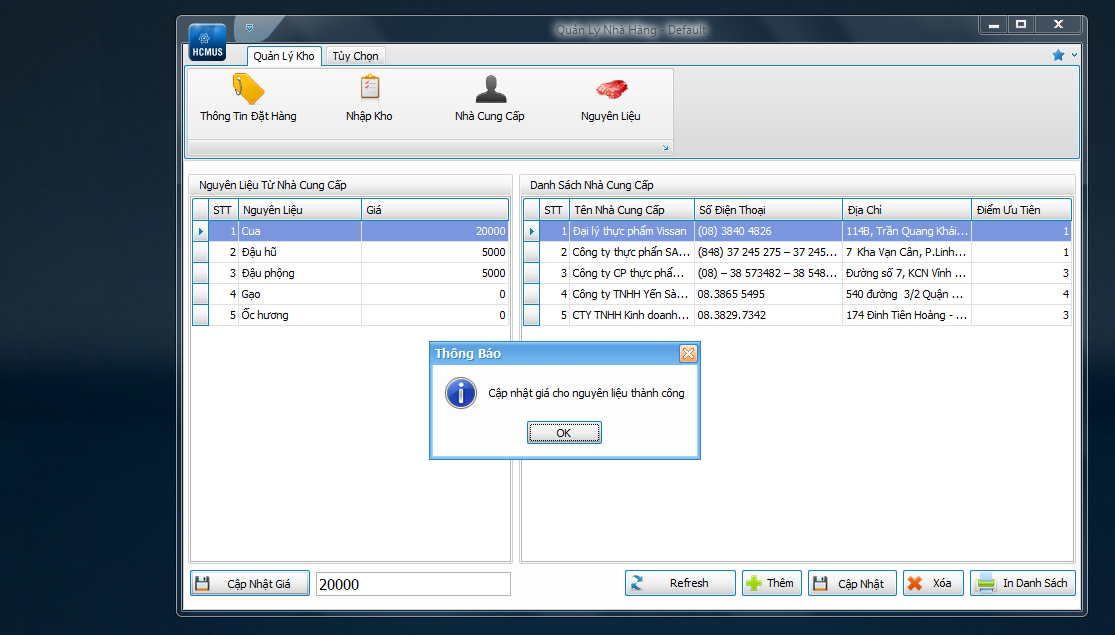
#### 3.3.2.2.Xóa Nhà Cung Cấp



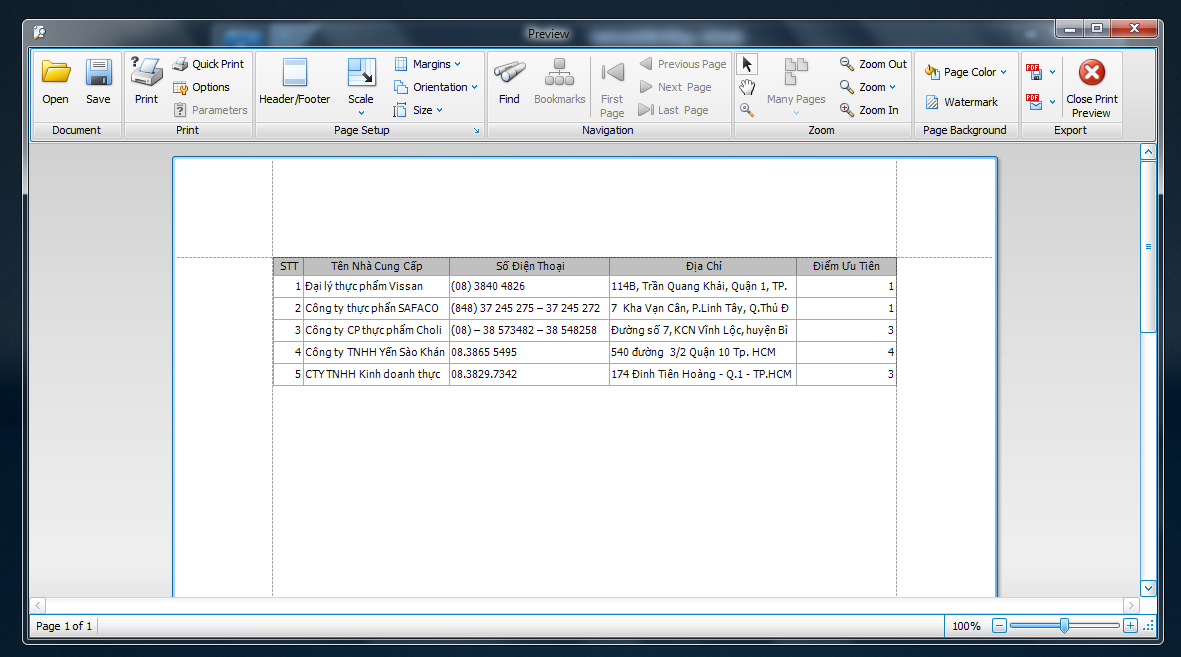
#### 3.3.2.3.Cập Nhật Nhà Cung Cấp



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



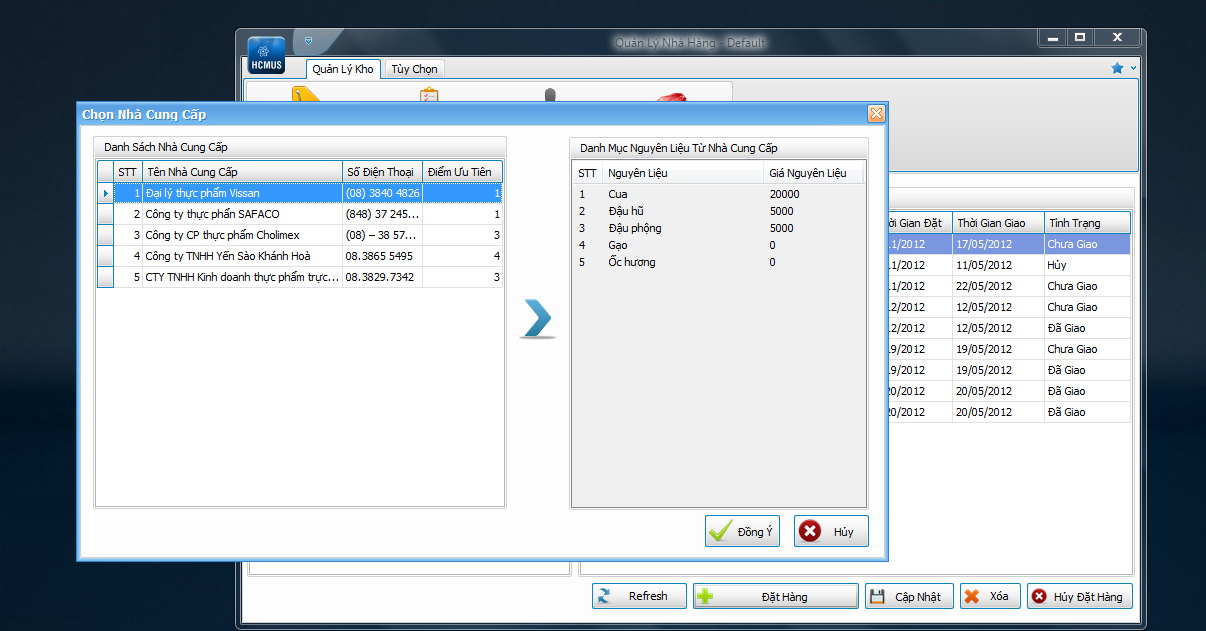
#### 3.3.2.5.In Danh Sách Nhà Cung Cấp



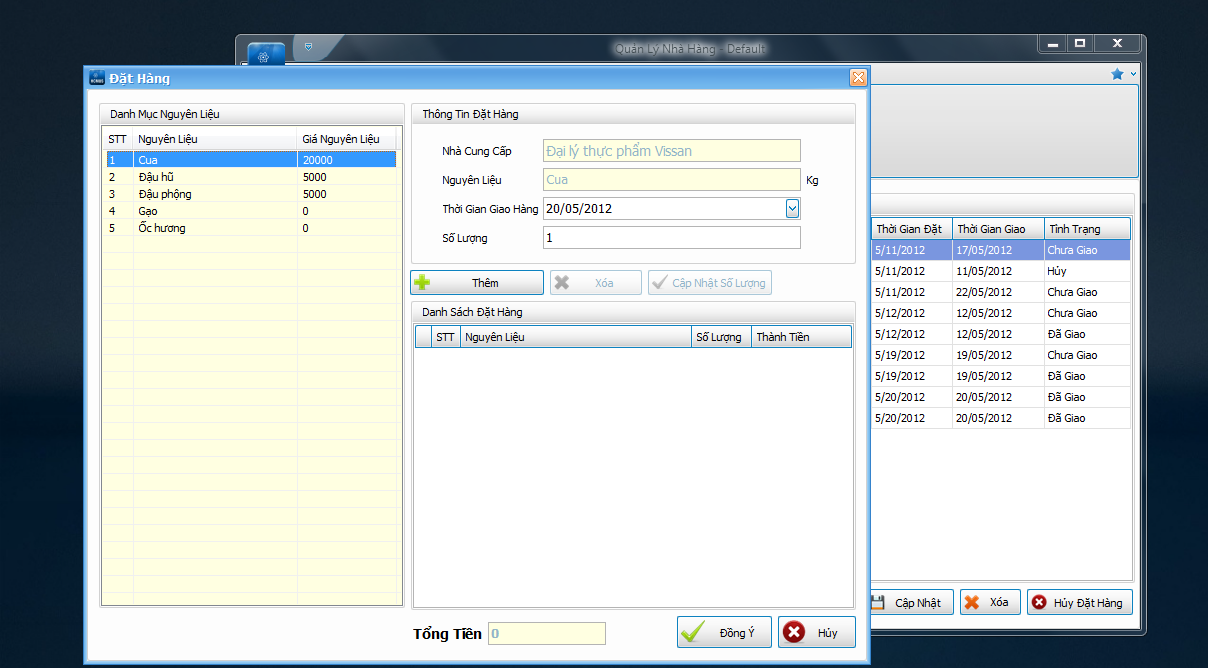
### 3.3.3.Quản Lý Thông Tin Đặt Hàng

#### 3.3.3.1.Thêm Đơn Đặt Hàng Mới

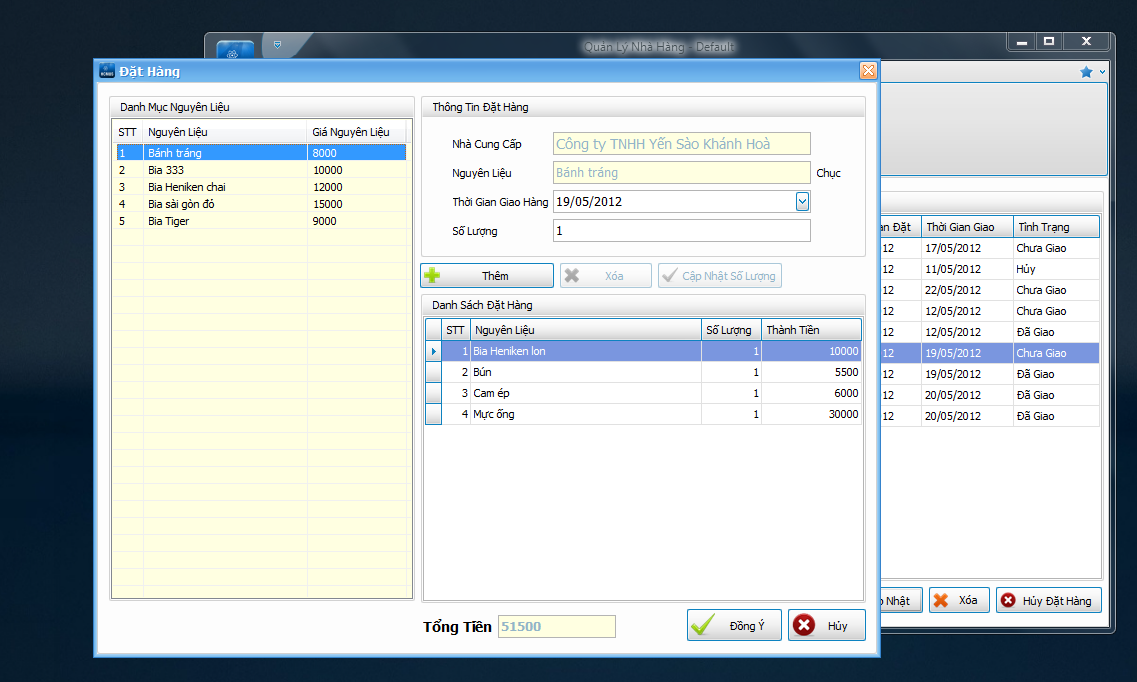
##### a.Chọn Nhà Cung Cấp



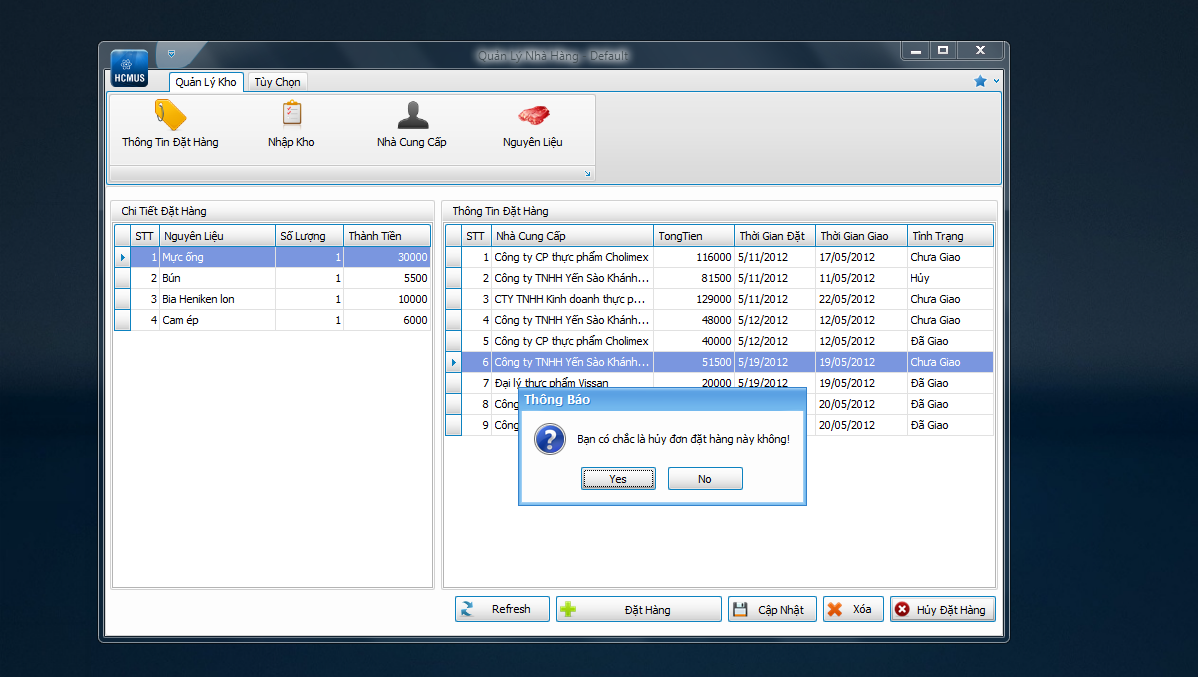
##### b.Chọn Nguyên Liệu



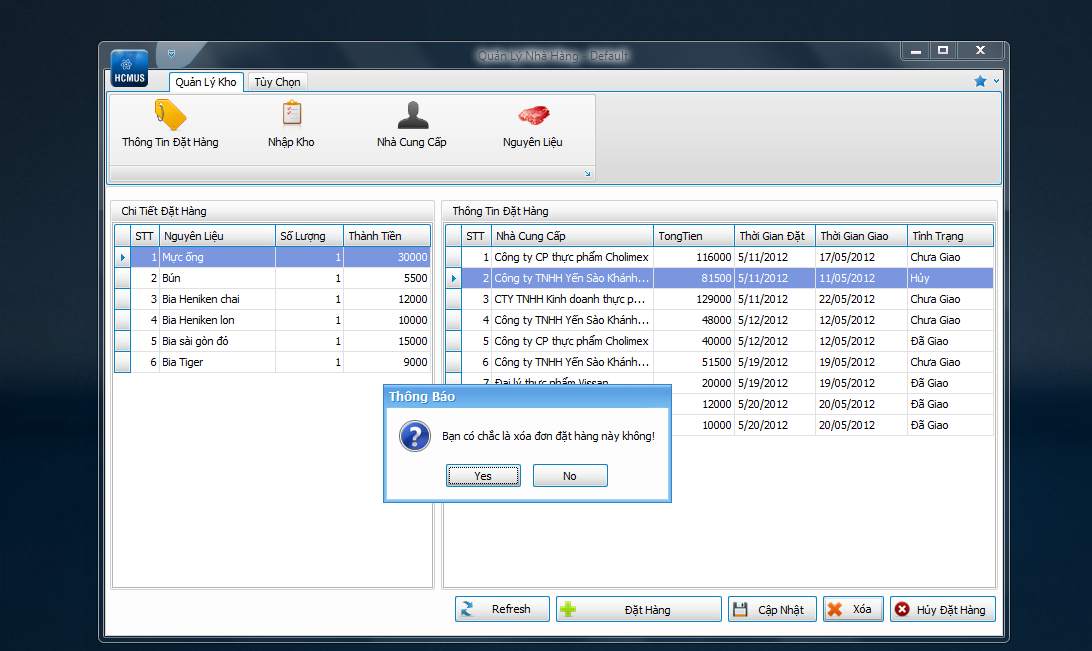
#### 3.3.3.2.Cập Nhật Thông Tin Đơn Đặt Hàng Chưa Giao



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

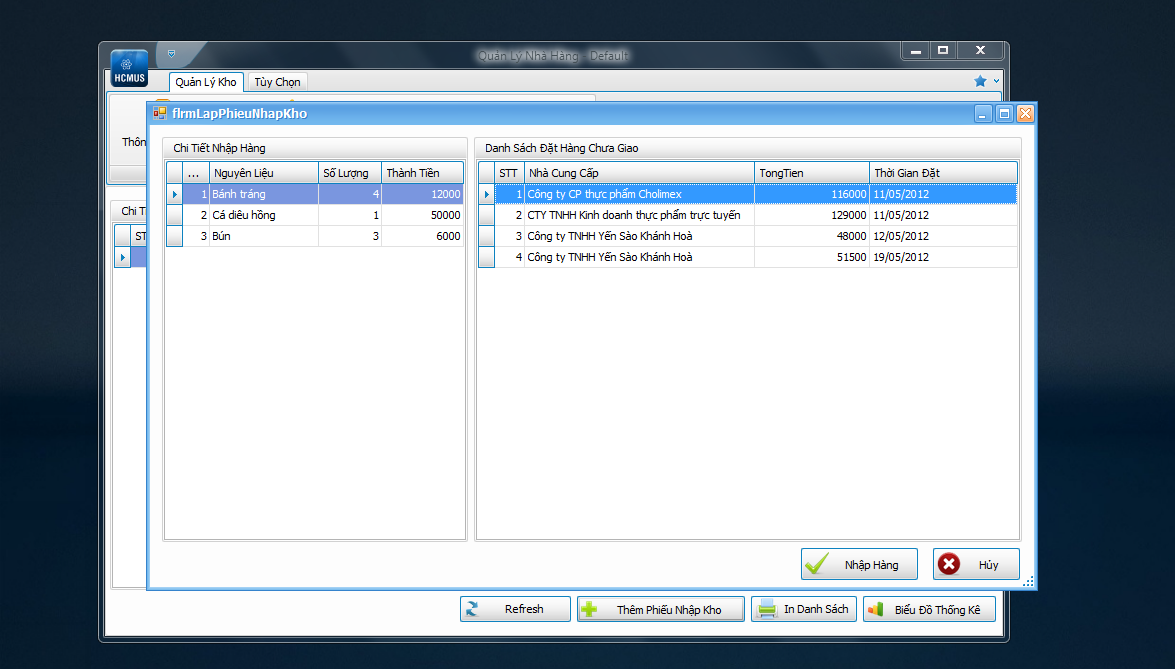


#### 3.3.3.5.Xóa Đơn Đặt Hàng “Chưa Giao” Hoặc Đã “Hủy”

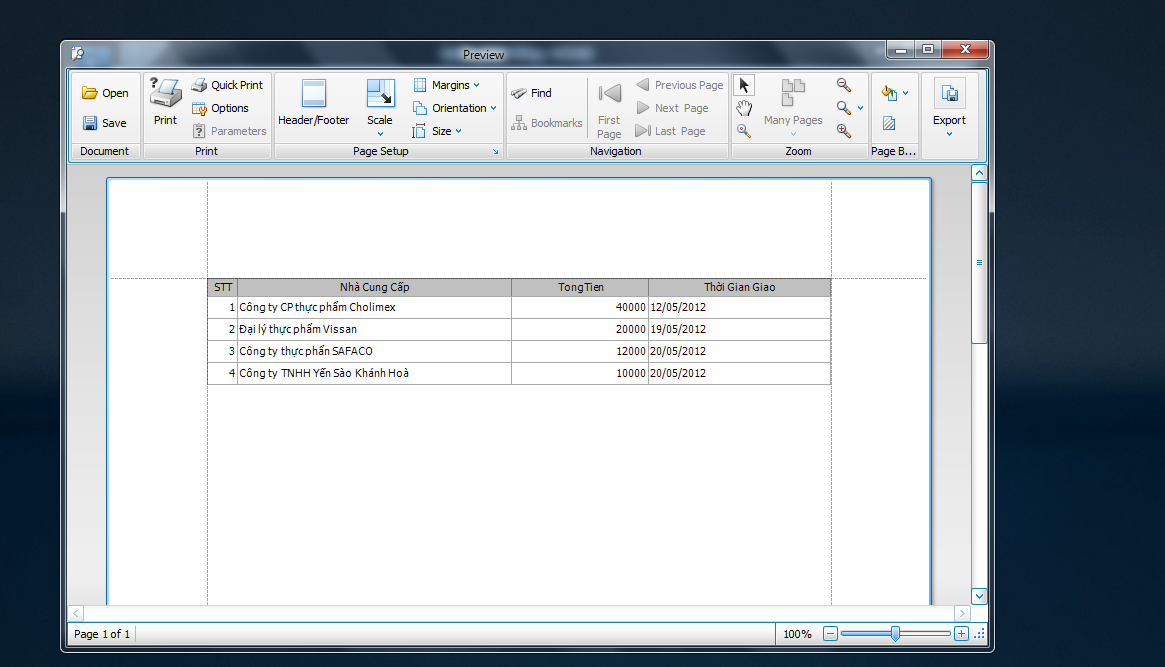


### 3.3.4.Quản Lý Thông Tin Nhập Kho

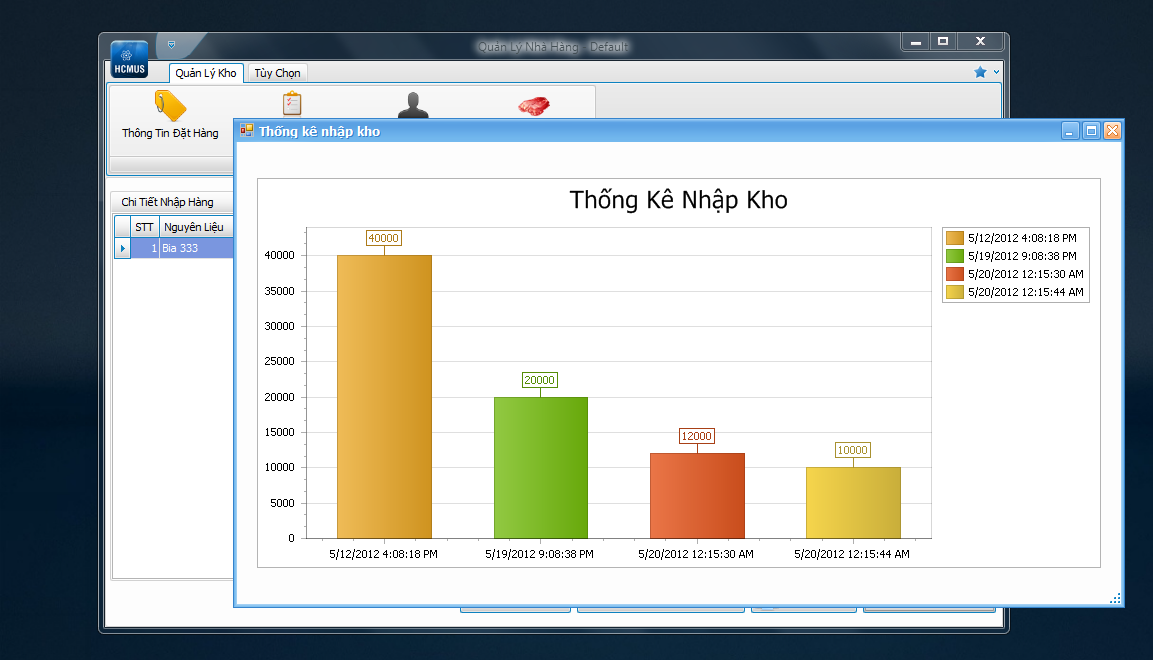
#### 3.3.4.1.Thêm Phiếu Nhập Kho



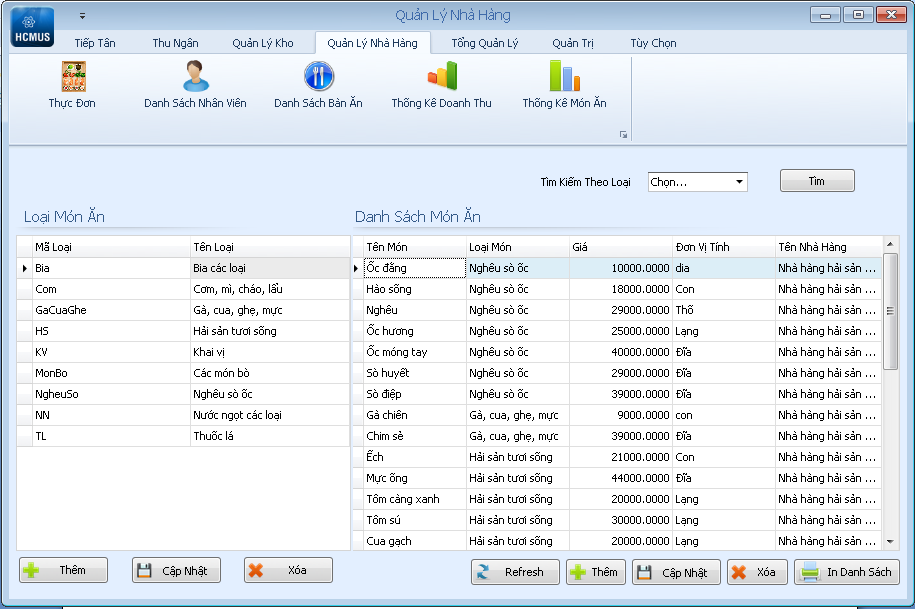
#### 3.3.4.2.In Danh Sách Phiếu Nhập Kho



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



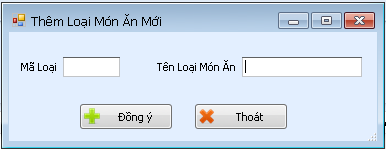
## 3.4.Quản Lý Nhà Hàng



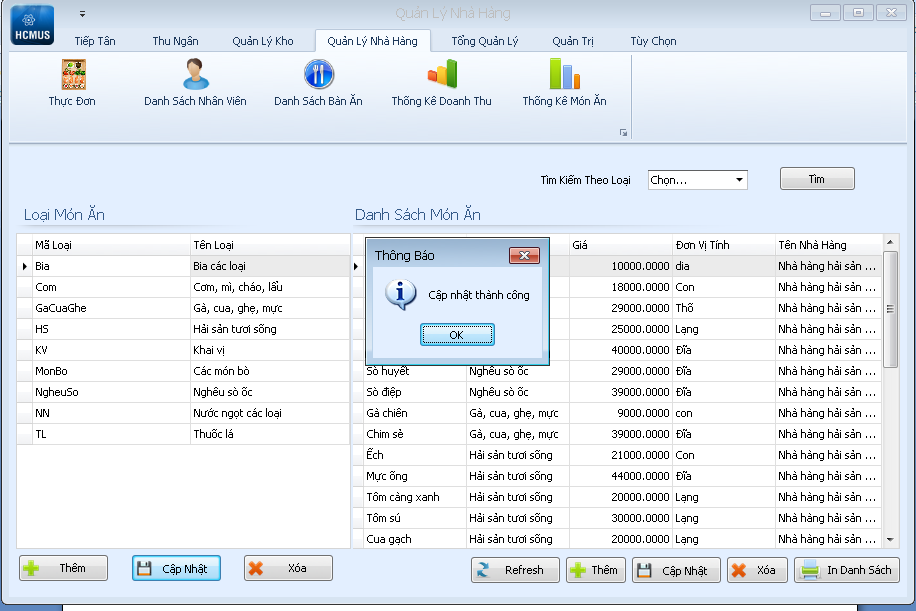
### 3.4.1.Quản lý Thực đơn

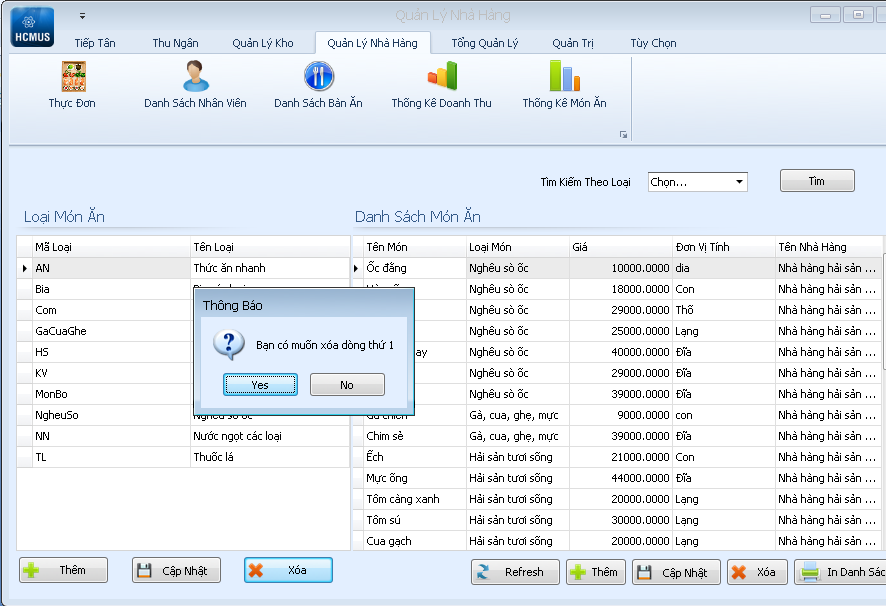
#### Quản lý Loại Món Ăn

***+ Thêm Loại Món Ăn***

******

***+ Cập nhật Loại Món Ăn***

******

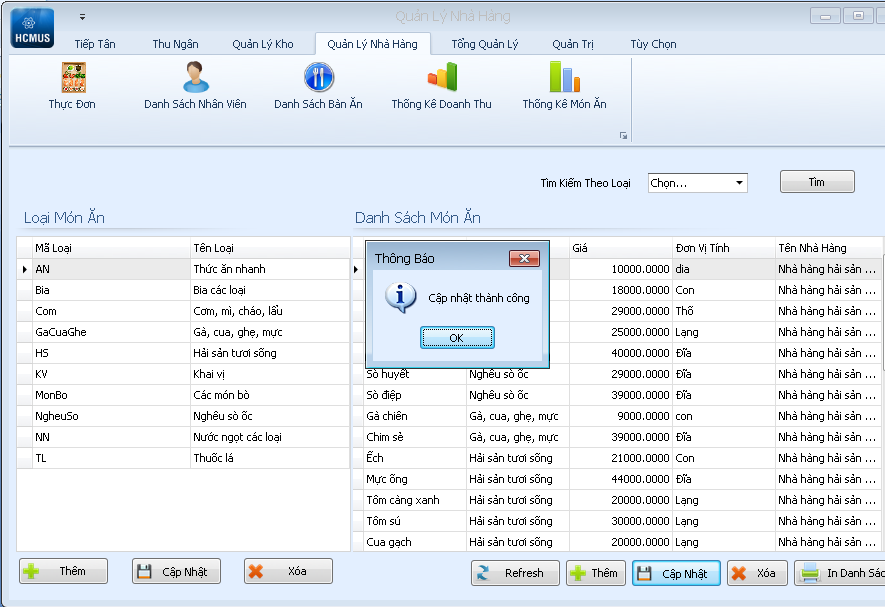
***+Xóa Loại Món Ăn***

#### Quản lý Món Ăn

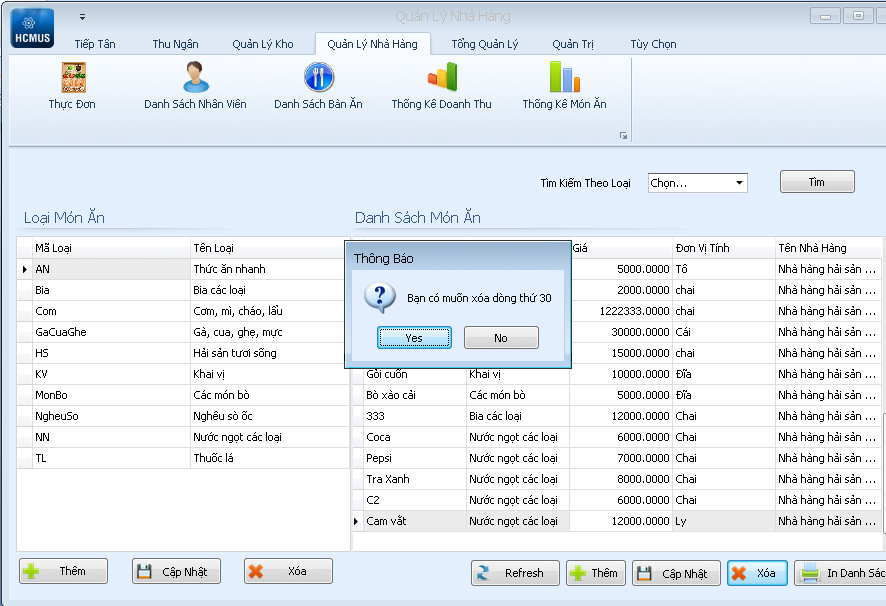
***+ Thêm Món Ăn***

******

***+ Cập nhật Món Ăn***

******

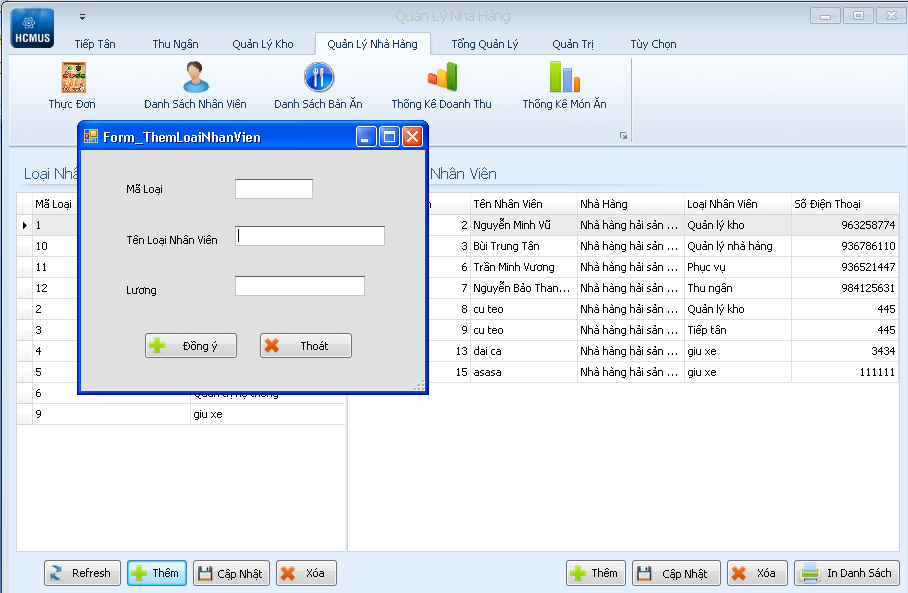
***+ Xóa Món Ăn***

******

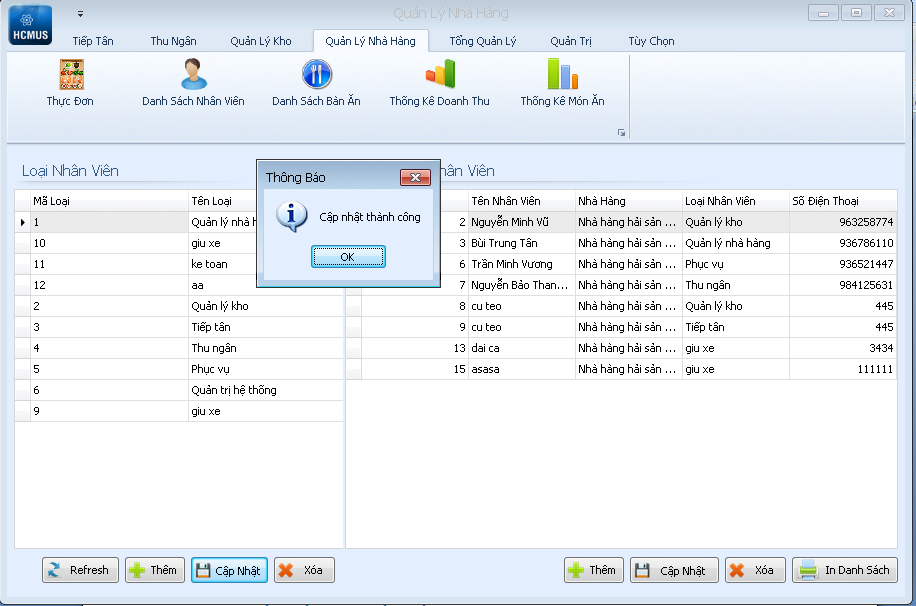
### 3.4.2.Quản Lý Danh Sách Nhân Viên

#### Quản lý Loại Nhân Viên

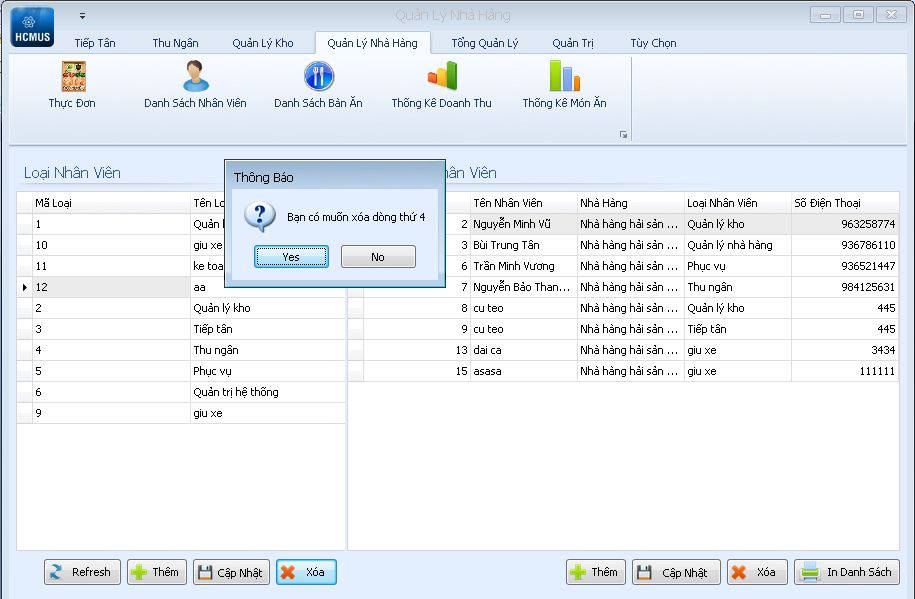
+ Thêm Loại Nhân Viên

******

***+ Cập nhật Loại Nhân Viên***

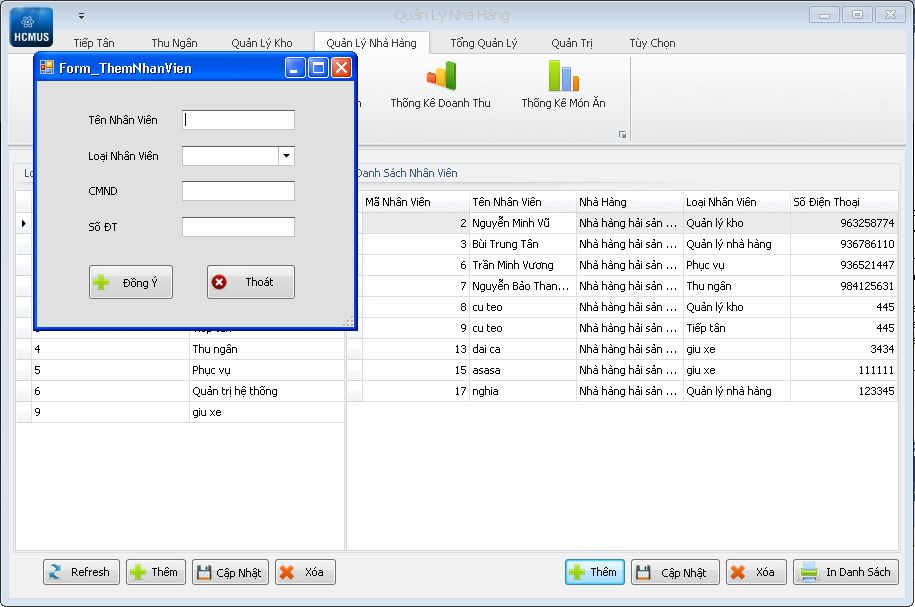
******

***+ Xóa Loại Nhân Viên***

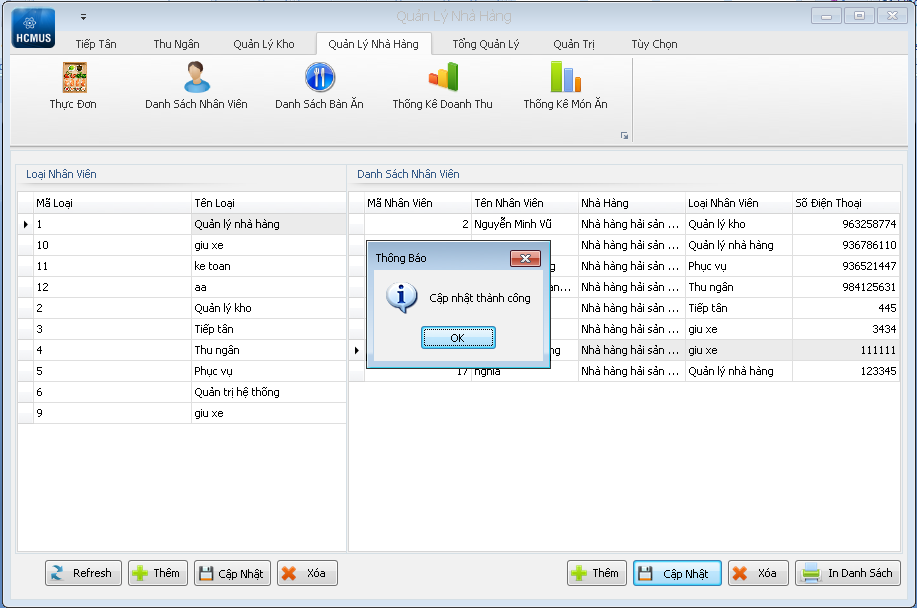
******

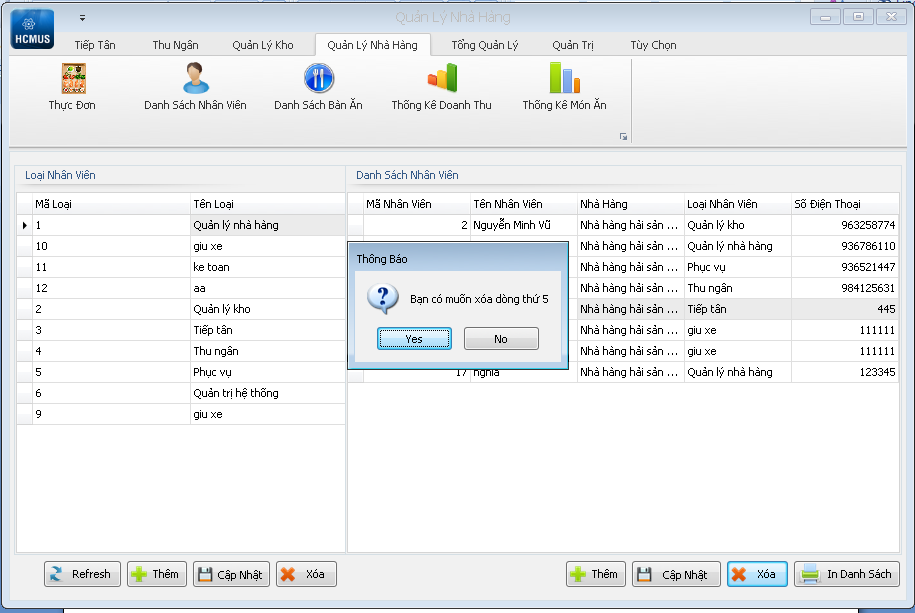
#### Quản lý Nhân Viên

***+ Thêm Nhân viên***

******

***+Cập nhật Nhân viên***

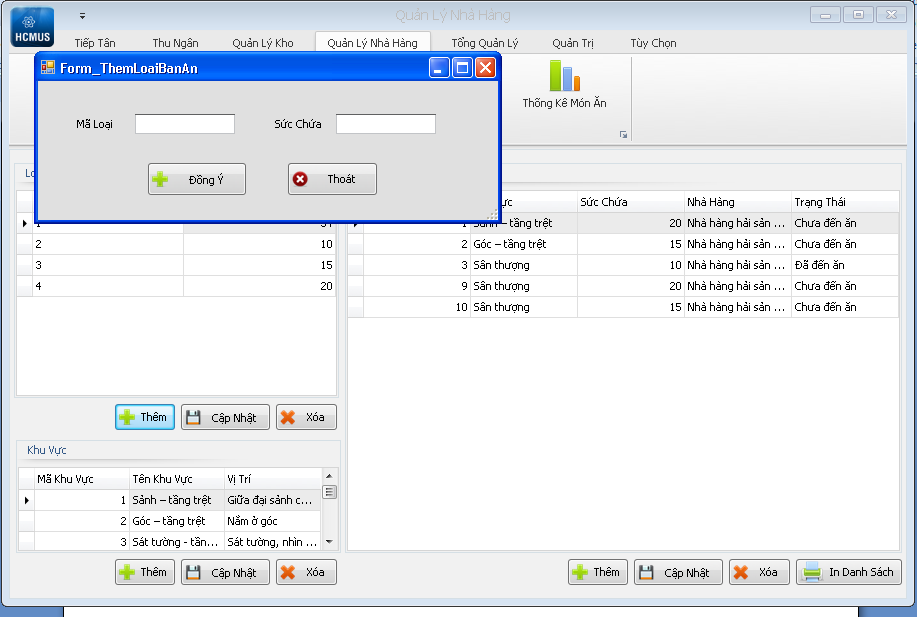
******

***+Xóa Nhân Viên***

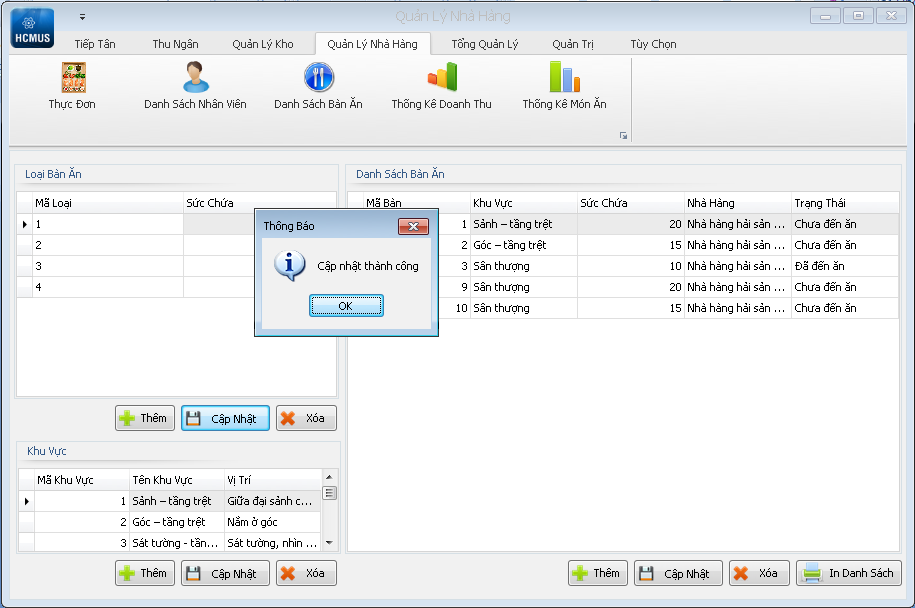
### 3.4.3.Quản Lý Danh Sách Bàn Ăn

#### Quản Lý Loại Bàn Ăn

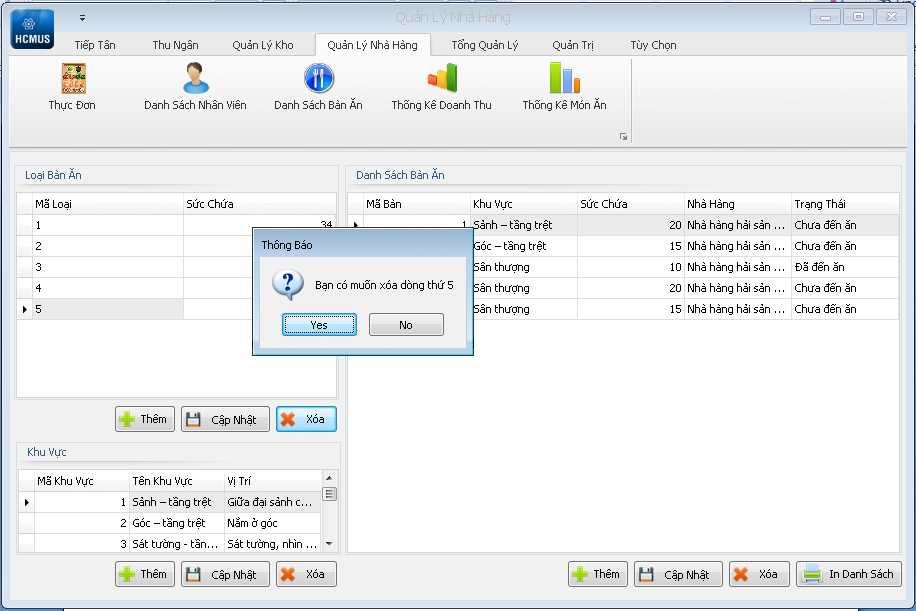
***+ Thêm Loại Bàn Ăn***

******

***+Cập nhật Loại Bàn Ăn***

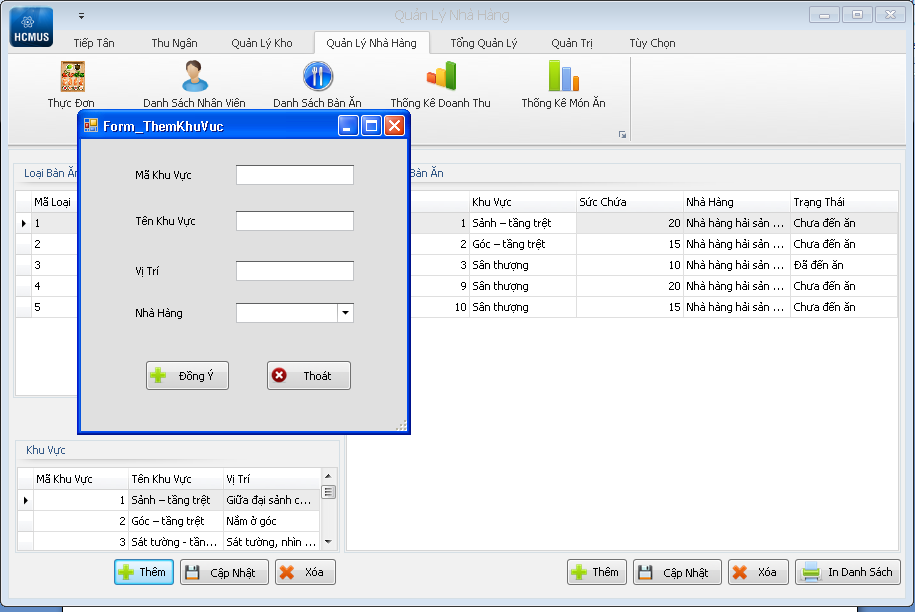
******

***+Xóa Loại Bàn Ăn***

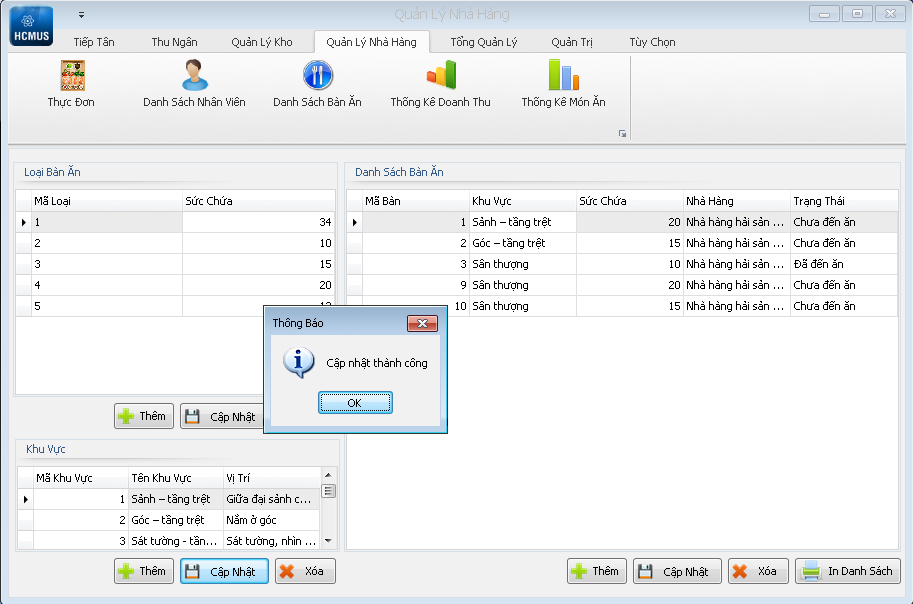
******

#### Quản Lý Khu Vực

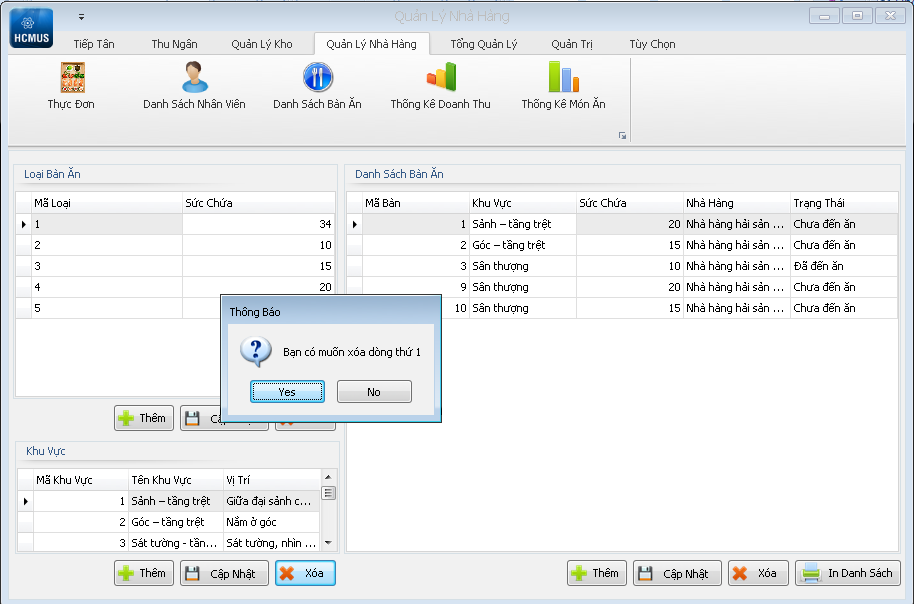
***+ Thêm Khu Vực***

******

***+ Cập nhật Khu vực***

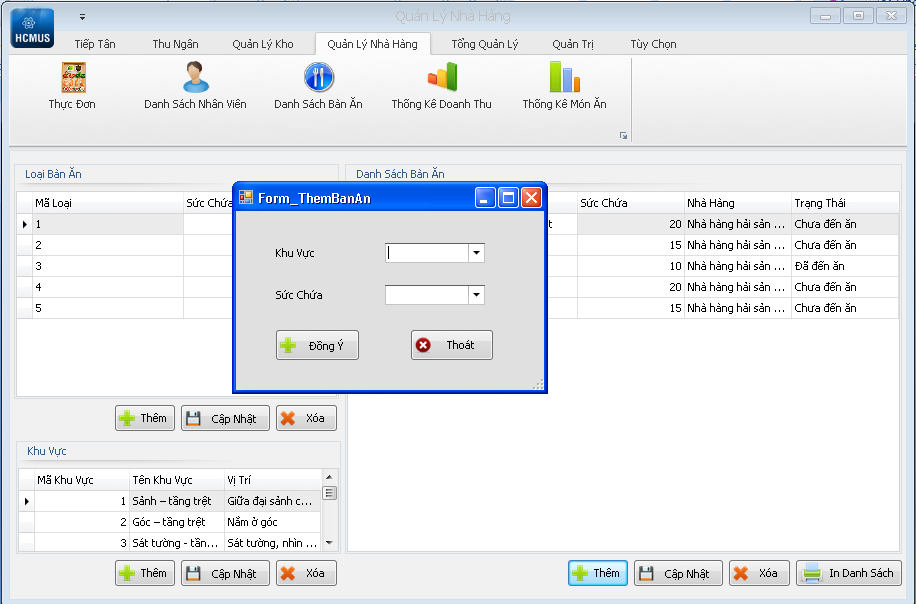
******

***+ Xóa Khu Vực***

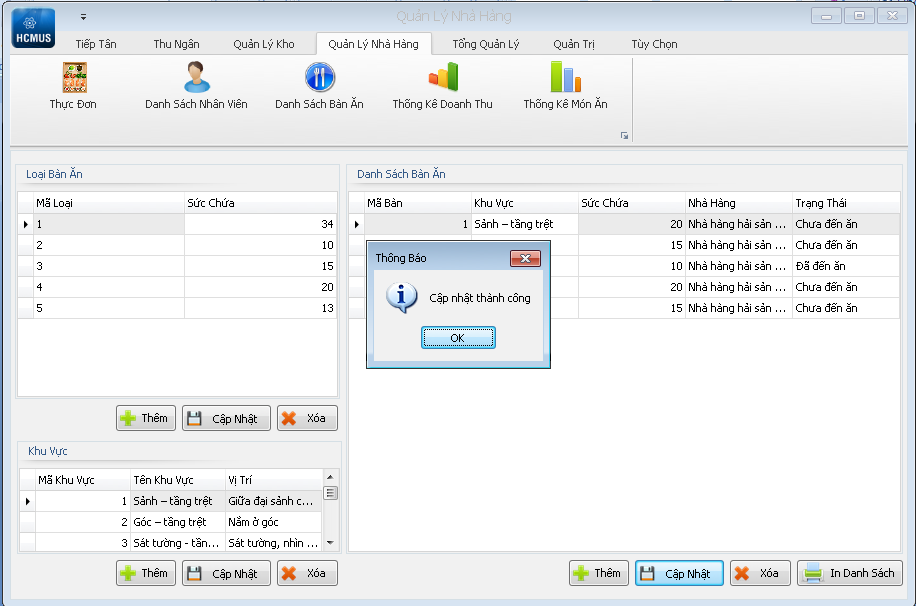
******

#### Quản Lý Danh sách Bàn Ăn

+ Thêm Bản Ăn



+Cập nhật Bàn Ăn



+ Xóa Bàn Ăn

