**Trường Đại học Khoa học tự nhiên**

**Khoa Công Nghệ Thông Tin**

BÁO CÁO ĐỒ ÁN CUỐI KÌ

**Môn : Hệ Quản Trị Cơ Sở Dữ Liệu**

# Quản Lý Quảng Cáo

**GVHD : Hồ Lê Thị Kim Nhung**

**Nhóm Connecting®:**

| | |
|---|---|
| **1. Huỳnh Thanh Bình** | **1041013** |
| **2. Nguyễn Đào Minh Đức** | **1041036** |
| **3. Trương Như Kỳ** | **1041072** |
| **4. Phạm Thành Luân** | **1041082** |

*Tp Hồ Chí Minh, ngày 01 tháng 5 năm 2011*

## 1. Liệt kê và Giải thích chức năng :

+ Use case là quản lý quảng cáo

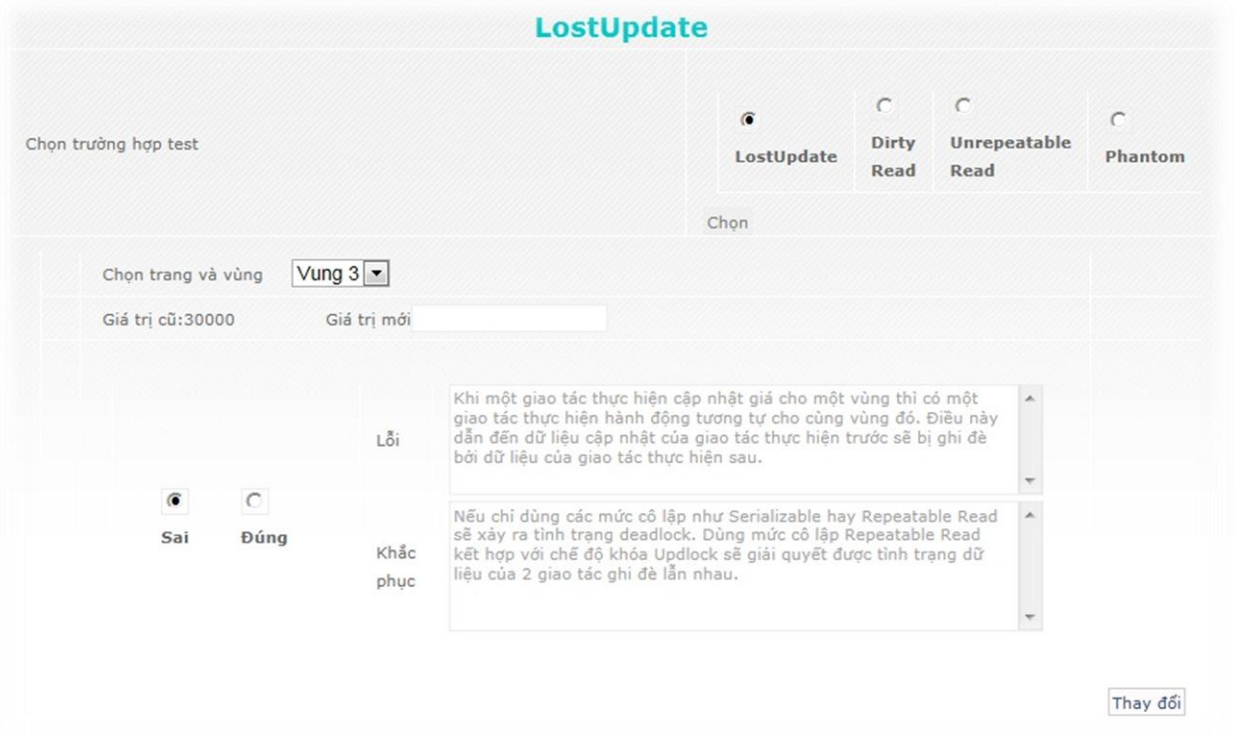
+ Tập các kịch bản trong usercase, và các actor  tương ứng

### 1.1 Lost Updates

Trường hợp 1:

☞ Actor  trong TH này là 2 người dùng admin sẽ đồng thời cập nhật  giá cho vùng , thuộc 1 trang  trong  số báo,admin1 chọn số báo 2, trang 3 , vùng 1 để cập nhật lại giá vùng này là gấp đôi giá trị ban đầu, và admin2 cũng chọn cùng vùng trên để cập nhật giá gấp 3 giá ban đầu (cả 2 admin đều chọn radiobutton **sai** ), thì sẽ dẫn đến việc dữ liệu của admin thứ 1, bị admin thứ 2 ghi đè lên khi này gia vùng là gấp 3 giá trị ban đầu → dữ liệu mất tính nhất quán

☞ Khi cả 2 người dùng admin chọn radiobutton **đúng** , đây là TH đã được  chỉnh sửa đúng nên giá của vùng sẽ được cập nhật thành gấp 6 lần giá trị ban đầu.



Trường hợp 2:

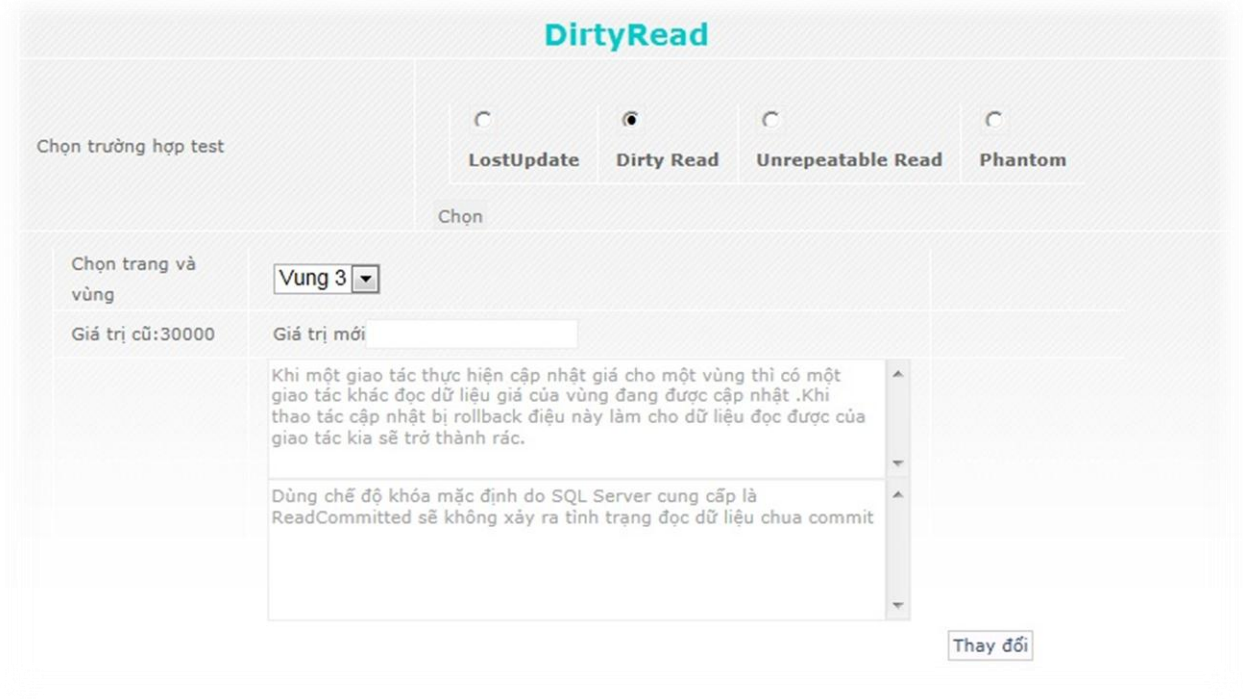
☞ Actor trong TH này là 1 người dùng khách hàng sẽ có 1 hợp đồng để lưu thông tin khách hàng, tổng số tiền phải thanh toán, tình trang thanh toán,thông tin vùng chọn đăng quảng cáo…. , trong TH khách hàng này thực hiện chọn thêm 1 vùng mới bất kỳ trong cùng số báo vào hợp đồng hiện tại, khi đó trong hợp đồng tổng số tiền đang là 50 triệu vnd. Chọn thêm 2 vùng có giá lần lượt là 5 triệu vnd và 6 triệu (chọn radiobutton **sai)**, thì sẽ dẫn đến việc hợp đồng sẽ ghi nhận tổng tiền sẽ là 56 triệu vnd.

☞ Khi 2 khách hàng chọn radiobutton **đúng** và thực hiện thao tác như trên thì tổng tiền là 61 triệu vnd.

## 1.2 Dirty Read

☞ Actor trong TH này là giữa 1 người dùng admin và 1 người dùng khách hàng, admin sẽ cập nhật lại giá mới cho vùng1 , thuộc trang 2 của số báo 1 là 5 triệu vnd (giá ban đầu 10 triệu vnd), thì trong thời gian việc cập nhật bị gián đoạn (cập nhật chưa thành công), thời điểm đó 1 khách hàng đọc thông tin giá của vùng 1 trang 2, số báo 1 sẽ là 5 triệu vnd.

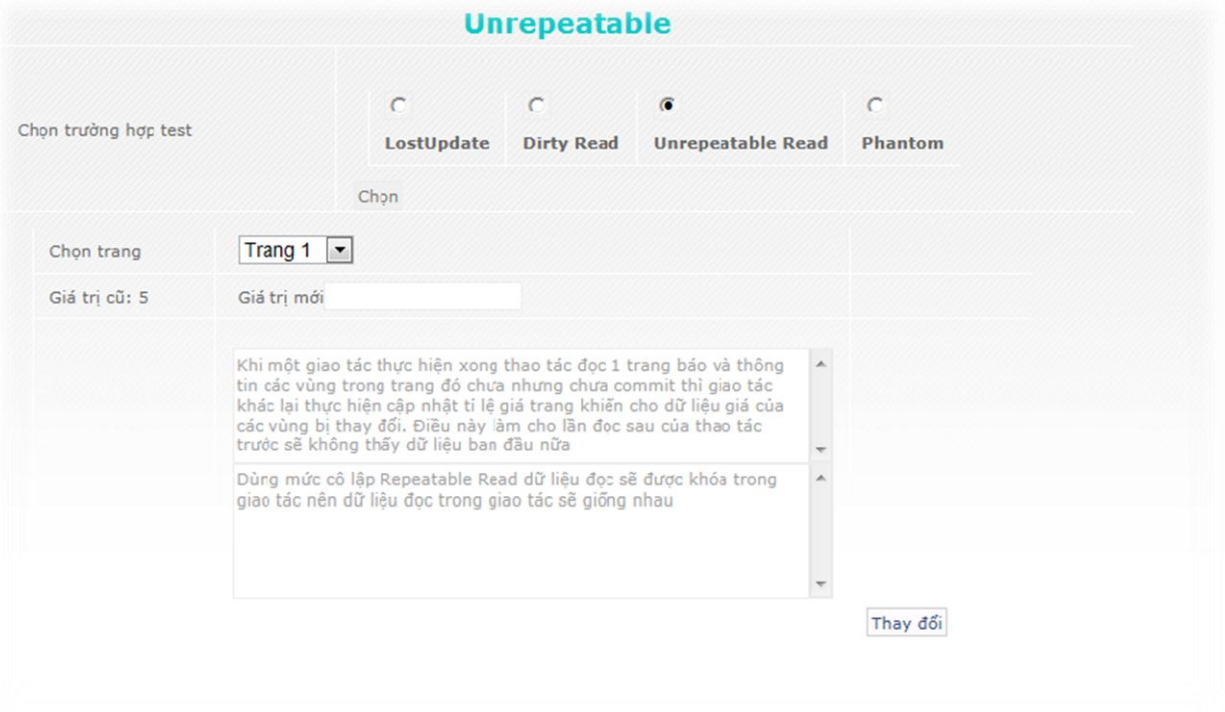
☞ Trường hợp đúng là với thao tác trên thì khách hàng sẽ đọc được thông tin giá vùng 1, trang 2, số báo 1 là 10 triệu vnd.




## 1.3 Unrepeatable Read

☞ Actor trong TH này là giữa 1 admin và 1 khách hàng, khách hàng tìm vùng 1 số báo 1 để đang quảng cáo thì với lần đọc đầu tiên là giá 10 triệu vnd (nhưng chưa chọn vùng để đăng quảng cáo) và trong khi đó admin lại cập nhật giá vùng trên là 15 triệu vnd ,dẫn đến khi khách hàng, chọn lại vùng 1 số báo 1 thì giá lúc này là 15 triệu vnd.

☞ Khi cả 2 người dùng, với thao tác theo trình tự như trên (khắc phục lỗi unrepeatable read) sau 2 lần đọc khách hàng vẫn đọc được giá vùng 1 số báo 1 là 10 triệu vnd.




## 1.4 Phantom

Trường hợp 1:

☞ Actor trong TH này là giữa 2 admin, admin1 tìm và thống kê trong số báo 1, có 10 hợp đồng quá thời gian thực hiện việc thanh toán tiền trước 2 ngày , admin2 lại tìm xóa những hợp đồng thuộc số báo trên trên , dẫn đến việc đọc dữ liệu của admin1 ở lần thứ 2 thì số hợp đồng là 0(cả 2 người dùng đều chọn radiobutton **sai**).

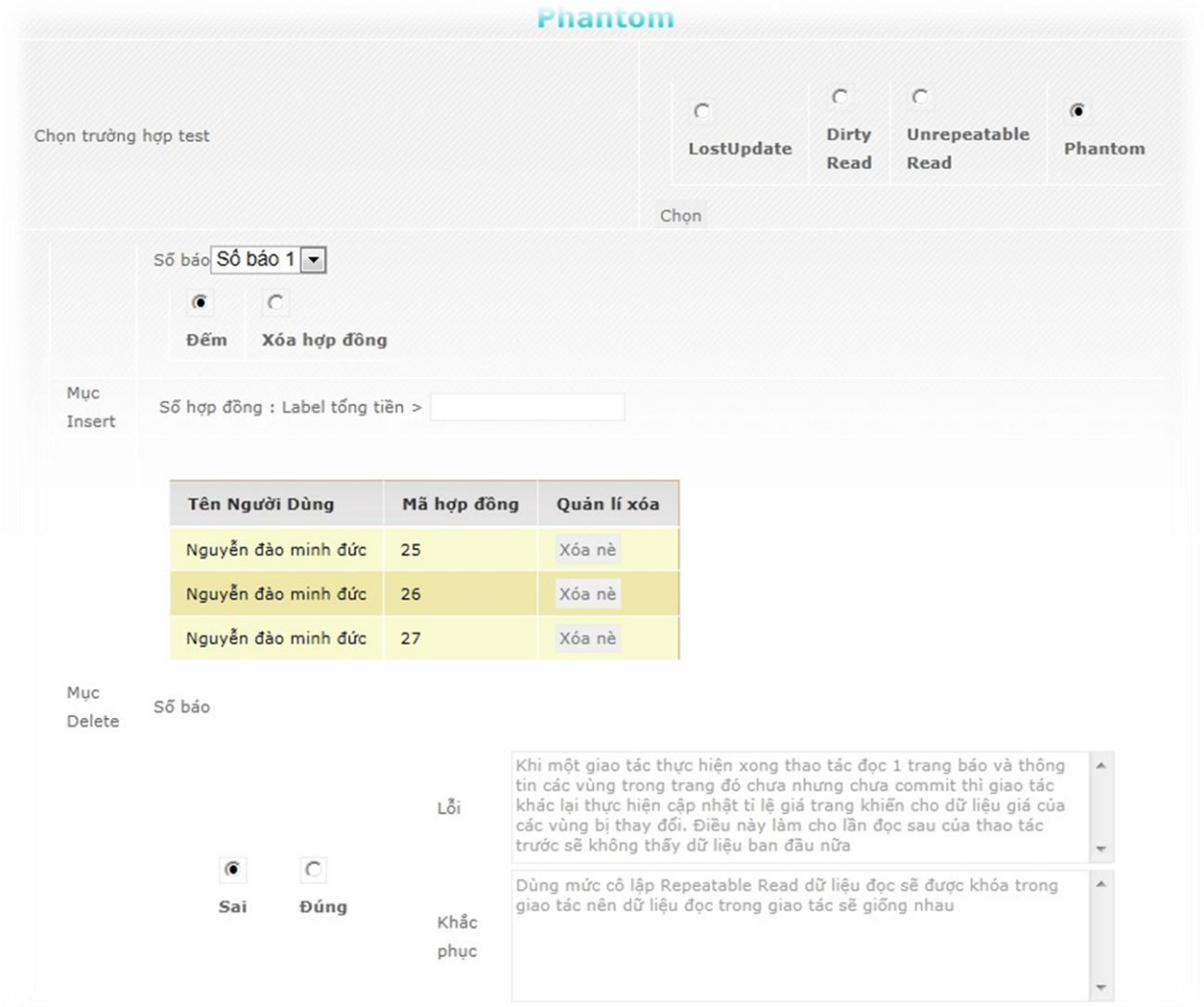
☞ Khi 2 admin chọn radiobutton **đúng** , với thao tác như trên thì số hợp đồng quá thời gian 2 ngày mà admin1 đọc được qua 2 lần vẫn là 10 nhưng trong khi đó admin đã xóa hết 10 hợp đồng này.

Trường hợp 2:

☞ Actor trong TH này là giữa 1 admin và 1 khách hàng, admin1 vừa tìm và thống kê có 10 hợp đồng có tổng tiền >20 triệu vnd, trong số báo 1,thì khi đó có khách hàng vừa lập 1 hợp đồng mới có tổng tiền là 30 triệu vnd trong số báo 1, dẫn đến việc admin1 thực hiện lại thao tác trên

thì tổng số hợp đồng là 11(cả 2 người dùng đều chọn radiobutton **sai**).

☞ Khi cả 2 người dùng chọn radiobutton **đúng** , với thao tác như trên (Th chọn radiobutton sai) thì tổng số hợp đồng có tổng tiền >20 triệu vnd là 10 (dù trong đã có 1khách hàng lập 1 hợp đồng > 30 triệu).




Admin :

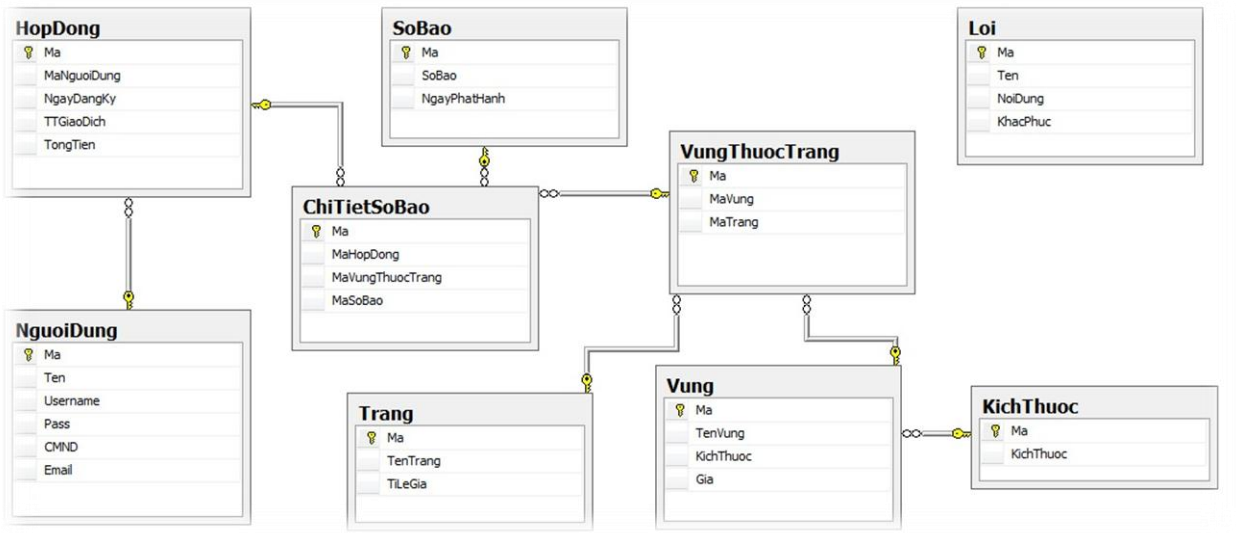
- Cập nhật giá vùng
- Cập nhật tỉ lệ giá trang
- Tìm danh sách khách hàng hết hạn giao dịch
- Xóa những hợp đồng hết hạn
- Đếm số lượng hợp đồng theo tổng tiền

User :

- Thêm hợp đồng
- Đăng kí

## 2.Thiết Kế Dữ Liệu:

### 2.1 Mô hình quan hệ:



## 2.2 Diễn giải :

### 2.2.1 Bảng SoBao:

Chứa thông tin về từng số báo gồm: Ma, SoBao, NgayPhatHanh

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt từng số báo |
| 2 | SoBao | Chuỗi | Tên từng số báo |
| 4 | NgayPhatHanh | Ngày | Ngày phát hành của số báo |

### 2.2.2Bảng Trang:

Chứa thông tin về trang báo: Ma, TenTrang

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt từng trang báo |
| 2 | TenTrang | Chuỗi | Tên từng trang báo |
| 3 | TiLeGia | Số nguyên | Tỉ lệ giá của từng trang |

### 2.2.3 Bảng Vung:

Chứa thông tin về vùng: Ma, TenVung, KichThuoc, Gia

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt từng vùng |
| 2 | TenVung | Chuỗi | Tên của vùng |
| 3 | KichThuoc | Số nguyên | Kích thước của vùng |
| 4 | Gia | Số nguyên | Giá của vùng |

### 2.2.4 Bảng VungThuocTrang:

Chứa thông tin chi tiết về các vùng thuộc về cùng 1 trang gồm: Ma, MaVung, MaTrang.

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt chi tiết danh sách vùng |
| 2 | MaVung | Số nguyên | Mã của từng vùng |
| 3 | MaTrang | Số nguyên | Cho biết vùng thuộc trang nào |

### 2.2.5 Bảng NguoiDung

Chứa thông tin về người dùng: Ma, Ten, Username, Pass, CMND, Email

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt người dùng |
| 2 | Ten | Chuỗi | Tên người dùng |
| 3 | Uername | Chuỗi | Tên đăng nhập |
| 4 | Pass | Chuỗi | Password |
| 5 | CMND | Số nguyên | Chứng minh nhân dân |
| 6 | Email | Chuỗi | email |

### 2.2.6 Bảng HopDong

Chứa thông tin về hợp đồng đăng ký của người dùng: Ma, MaNguoiDung, NgayDangKy, TTGiaoDich, TongTien

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt mỗi hợp đồng |
| 2 | MaNguoiDung | Số nguyên | Mã của người dùng đăng ký hợp đồng |
| 3 | NgayDangKy | Ngày | Ngày đăng ký |
| 4 | TTGiaoDich | Số nguyên | Trạng thái giao dịch |
| 5 | TongTien | Số nguyên | Tổng tiền của hợp đồng |

Trạng thái giao dịch: khách hàng đã đến công ty giao dịch chưa

- Giá trị 0: chưa giao dịch.
- Giá trị 1: đã giao dịch.

### 2.2.7 Bảng ChiTietSoBao

Chứa thông tin về chi tiết hợp đồng của khách hàng thuộc số báo gồm: Ma, MaHopDong, MaVungThuocTrang, MaSoBao

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt chi tiết số báo |
| 2 | MaHopDong | Số nguyên | Cho biết chi tiết số báo này thuộc về hợp đồng nào |
| 3 | MaVungThuocTrang | Số nguyên | Cho biết vùng đăng ký |
| 4 | MaSoBao | Số nguyên | Cho biết thuộc về số báo nào |

### 2.2.8 Bảng KichThuoc

Chứa thông tin về kích thước vùng: Ma, KichThuoc

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt từng kích thước |
| 2 | KichThuoc | Số nguyên | Kích thước |

### 2.2.9.Bảng Loi

Chứa thông báo lỗi về xử lý đồng thời: Ma, TenLoi, NoiDung, KhacPhuc

| STT | Thuộc tính | Kiểu dữ liệu | Diễn giải |
|---|---|---|---|
| 1 | Ma | Số nguyên | Khóa chính để phân biệt từng loại lỗi |
| 2 | Ten | Chuỗi | Tên lỗi |
| 3 | NoiDung | Chuỗi | Nội dung chi tiết lỗi trong trường hợp |
| 4 | KhacPhuc | Số nguyên | Cho biết cách khắc phục lỗi |

## 2.3.Mô tả khóa ngoại :

| STT | Table nguồn | Table Đích | Quan hệ |
|---|---|---|---|
| 1 | ChiTietSoBao (MaSoBao) | SoBao (Ma) | Chi tiết số báo thuộc về số báo nào |
| 2 | ChiTietSoBao (MaHopDong) | HopDong (Ma) | Chi tiết số báo thuộc về hợp đồng |
| 3 | ChiTietSoBao (MaVungThuocTrang) | VungThuocTrang (Ma) | Vùng đăng ký của hợp đồng |
| 4 | HopDong (MaNguoiDung) | NguoiDung (Ma) | Khách hàng chủ hợp đồng |
| 5 | VungThuocTrang (MaVung) | Vung (Ma) | Liên hệ giữa vùng thuộc về trang |
| 6 | VungThuocTrang (MaTrang) | Trang (Ma) | Liên hệ giữa vùng thuộc về trang |
| 7 | Vung (KichThuoc) | KichThuoc (Ma) | Kích thước của mỗi vùng |
| 8 | Loi (KhacPhuc) | SuaLoi (Ma) | Cách khắc phục lỗi |

## 3.Tiên Đoán Các Vấn Đề:

### 3.1.Lost Update:

#### ✦Trường hợp 1:

- Khi  2 giao tác cùng thực hiện cập nhật giá cho 1 vùng sẽ xảy ra đụng độ

CapNhatGiaVung

- Tham số truyền vào: @mavung, @heso
- Đọc giá hiện tại của vùng và gán vào biến @ght
- Cập nhật giá vùng theo công thức: giá mới = @ght * @heso

-Store CapNhatGiaVung mô tả công việc cập nhật giá cho 1 vùng theo cách tăng theo hệ số như giá có thể tăng gấp đôi.

-Tình huống đụng độ xảy ra như sau: giả sử cùng cập nhật giá cho 1 vùng có mã = 3 và giá hiện tại của vùng = 2

| T1 (cập nhật giá theo hệ số = 2 ) | T2 (cập nhật giá theo hệ số = 3 ) |
|---|---|
| Đọc giá hiện tại: 2 | |
| | Đọc giá hiện tại: 2 |
| Cập nhật giá mới = 2 * 2 | |
| | Cập nhật giá mới = 2 * 3 |
| | |
| Commit | Commit |

-Kết luận: với giá ban đầu của vùng là = 2 thì sau 2 lần cập nhật thì giá mới sẽ = 12. Nhưng sau khi chạy xong 2 giao tác thì giá mới chỉ là 6 do dữ liệu cập nhật của T1 đã bị T2 ghi đè → mất tính nhất quán dữ liệu.

✦Trường hợp 2:

-Khi 2 giao tác cùng thực hiện cập nhật tổng tiền cho 1 hợp đồng sẽ xảy ra đụng độ.

CapNhatTongTien

- Tham số truyền vào: @mahd, @sotien
- Đọc tổng tiền hiện tại của hợp đồng và gán vào biến @t
- Cập nhật tổng tiền theo công thức: tổng tiền = @t + @sotien

-Store CapNhatTongTien mô tả công việc cập nhật số tiền tổng cộng cho 1 hợp đồng với thao tác chọn thêm vùng mới. Tổng số tiền sẽ phải cộng thêm số tiền của vùng mới.

-Tình huống đụng độ xảy ra như sau: giả sử cùng cập nhật tổng tiền cho 1 hợp đồng có mã = 1, tổng số tiền hiện tại = 0 và số tiền mới cho 2 giao tác lần lượt là 2, 3

| T1 (cập nhật theo số tiền mới = 2 ) | T2 (cập nhật theo số tiền mới = 3) |
|---|---|
| Đọc số tiền tổng hiện tại: 0 | |
| | Đọc số tiền tổng hiện tại: 0 |
| Cập nhật số tiền tổng mới = 0 + 2 | |
| | Cập nhật số tiền tổng mới = 0 + 3 |
| | |
| Commit | Commit |

-Kết luận: dữ liệu ban đầu tổng số tiền là = 0. Sau khi 2 lần cập nhật thì số tiền tổng cộng phải là = 5. Tuy nhiên, tổng mới hiện tại lại = 3 do dữ liệu của T1 đã bị mất → mất tính nhất quán dữ liệu.

## 3.2.Dirty Read:

-Khi 1 giao tác đang thực hiện việc cập nhật giá cho 1 vùng thì lại có 1 giao tác khác đọc dữ liệu đó lên.

RollCapNhatGiaVung

- Tham số truyền vào: @mavung, @gia
- Cập nhật giá = @gia
- Rollback thao tác cập nhật.

DirtyRead

- Tham số truyền vào: @mavtt.
- Đọc giá của vùng

-Store RollCapNhatGiaVung mô tả việc cập nhật giá cho 1 vùng nhưng vì lý do nào đó việc cập nhật không thực hiện được hoàn toàn nên giao tác bị *rollback*.

-Store DirtyRead mô tả việc đọc dữ liệu của 1 vùng.

-Tình huống đụng độ xảy ra như sau: giả sử việc cập nhật giá và đọc dữ liệu của vùng có cùng mã = 1, giá hiện tại = 3, mã vùng thuộc trang tương ứng = 1

| T1 (RollCapNhatGiaVung1, 2) | T2 (DirtyRead 1) |
|---|---|
| Cập nhật giá mới = 2 | |
| | Đọc giá hiện tại: 2 |
| Rollback | |
| | Commit |

-Kết luận: theo như câu lệnh *Update* của T1 thì giá của vùng thay đổi theo giá mới truyền vào là 2 lúc này thì giao tác T2 đọc được giá mới cập nhật là 2. Nhưng giao tác T1 bị *rollback* nên dữ liệu đọc được của T2 là dữ liệu sai →mất tính nhất quán dữ liệu đối với giao tác T2.

## 3.3.Unrepeatable Read:

-Tình huống xảy ra khi có 1 giao tác đọc dữ liệu của một trang lên cùng lúc đó lại có 1 giao tác khác thực hiện việc cập nhật dữ liệu trên trang này.

DocDuLieuTrang

- Tham số truyền vào: @matrang
- Đọc dữ liệu của trang, tỉ lệ giá

CapNhatTiLeGiaTrang

- Tham số truyền vào: @matrang, @tlgia
- Cập nhật dữ liệu của trang, tỉ lệ giá = @tlgia

-Store DocDuLieuTrang mô tả việc đọc dữ liệu tỉ lệ giá của 1 trang để tính toán giá cho những vùng thuộc trang đó.

-Store CapNhatTiLeGiaTrang mô tả việc thực hiện cập nhật tỉ lệ giá trang cho 1 trang nào đó.

-Tình huống đụng độ xảy ra như sau: giả sử ta cùng đọc dữ liệu và cập nhật tỉ lệ giá cho 1 trang có mã = 1, tỉ lệ giá =2

| T1 (DocDuLieuTrang  1) | T2 (CapNhatTiLeGiaTrang 1, 2) |
|---|---|
| Đọc dữ liệu trang có mã = 1 | |
| | Cập nhật tỉ lệ giá trang = 2 cho trang có mã =1 |
| | |
| Đọc dữ liệu trang có mã = 1 | |
| Commit | Commit |

-Kết luận: thao tác đọc của T1 lần đầu cho ra kết quả là dữ liệu đã có trong csdl. Nhưng sau khi thao tác cập nhật của T2 hoàn thành thì dữ liệu đã bị thay đổi làm cho lần đọc sau của T1 sẽ cho ra 1 kết quả khác  → mất tính nhất quán dữ liệu đối với giao tác T1.

## 3.4.Phantom:

### Delete

- Khi có 1 giao tác thực hiện thao tác trên 1 tập dữ liệu là hợp đồng các khách  hàng đăng ký chọn vùng nhưng đã hết hạn giao dịch là 2 ngày thì có 1 giao tác khác lại thực hiện việc thêm hoặc xóa 1 hợp đồng khách hàng nằm trong danh sách các khách hàng trên.

DSKhachHangChuaGD

- Tham số truyền vào: @masb.
- Tìm đọc những hợp đồng đã hết hạn theo mã số báo.

XoaHopDongHetHan

- Tham số truyền vào: @masb.
- Tìm những hợp đồng đã hết hạn theo mã số báo.
- Xóa 1 hợp đồng đã hết hạn bất kỳ

-Store DSKhachHangChuaGD mô tả việc tìm kiếm những hợp đồng của khách hàng đã hết hạn.

-Store XoaHopDongHetHan mô tả việc xóa đi 1 hợp đồng của khách hàng đã hết hạn giao dịch.

-Tình huống đụng độ xảy ra như sau: giả sử 2 store thuộc 2 giao tác thực hiện việc tìm kiếm và xóa theo cùng 1 số báo có mã =2.

| T1 (DSKhachHangChuaGD  2 ) | T2 (XoaHopDongHetHan 2) |
|---|---|
| Tìm đọc hợp đồng theo mã số báo: 2 | |
| | Tìm hợp đồng theo mã số báo: 2 |
| | Xóa 1 hợp đồng bất kỳ |
| Tìm đọc hợp đồng theo mã số báo: 2 | |
| | Commit |
| Commit | |

-Kết luận: khi T1 chạy xong câu lệnh tìm đọc những hợp đồng hết hạn thì cho kết quả là 1 danh sách khách hàng.  Đến khi T2 cũng chạy xong câu lệnh này và tìm ra được 1 danh sách tương tự sau đó T2 xóa đi 1 hợp đồng bất kỳ và *commit*. Lúc này T1 đọc lại danh sách đó thì thấy kết quả thay đổi → mất tính nhất quán dữ liệu đối với giao tác T1.

## Insert:

- Trường hợp 1 giao tác sẽ thực hiện thao tác tra số lượng hợp đồng theo tổng tiền, vd trong trường hợp trên là tổng tiền > 15000.Nhưng trong lúc đó thì có 1 hợp đồng mới của 1 khách hàng được thêm vào.

DemHopDong

- Tìm đọc những hợp đồng có tổng tiền >15000.

ThemHopDong

- Tham số truyền vào: @mand, @ngay, @mavungtt, @masb.
- Thêm vào 1 hợp đồng mới ứng với chi tiết số báo của hợp đồng đó.

-Store DemHopDong mô tả việc tìm kiếm những hợp đồng có tổng tiền >15000.

-Store ThemHopDong mô tả việc thêm mới đi 1 hợp đồng của ứng với chi tiết số báo.

-Tình huống đụng độ xảy ra như sau: giả sử 2 store thuộc 2 giao tác thực hiện việc tra số lượng hợp đồng có tổng tiền >15000 và thêm mới 1 hợp đồng.

| T1 (DemHopDong ) | T2 (ThemHopDong) |
|---|---|
| Tra số lượng hợp đồng có tổng tiền >15000 | |
| | Thêm mới 1 hợp đồng theo chi tiết số báo |
| | |
| Tra số lượng hợp đồng có tổng tiền >15000 | |
| | Commit |
| Commit | |

-Kết luận: khi T1 chạy xong câu lệnh tra số lượng hợp đồng có tổng tiền >15000 thì cho kết quả là 1 danh sách các hợp đồng ứng với các số báo của hợp đồng đó. Đến khi T2 thêm mới 1 hợp đồng thì lúc này T1 thực hiện đọc lại số lượng hợp đồng thì thấy kết quả thay đổi → mất tính nhất quán dữ liệu đối với giao tác T1

# 4.Xử Lý Vấn Đề

## 4.1.Lost Update:

### Trường hợp 1:

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc CapNhatGiaVung_f<br>@mavung int, @heso int<br>As<br>Begin<br>  begin transaction<br>  declare @ght int<br>  select @ght=Gia From Vung<br>  where Ma = @mavung<br>  waitfor delay '00:00:05'<br><br>  update Vung<br>  set  Gia = @ght * @heso<br>  where Ma = @mavung<br>  if(@@error<> 0)<br>  begin<br>  rollback transaction<br>  return<br>  end<br>  commit transaction<br>end | |
| exec CapNhatGiaVung_f 1, 2 | exec CapNhatGiaVung_f 1, 3 |

-Khi  T1 thực hiện việc cập nhật giá cho 1 vùng, T2 cũng thực hiện hành động tương tự cùng cho vùng đó. Lúc này dữ liệu cập nhật của giao tác thực hiện trước sẽ bị dữ liệu của giao tác thực hiện sau ghi đè lên dẫn đến việc dữ liệu cập nhật của T1 bị mất.

### **Khắc phục:

-Để giải quyết tình huống này gây nên nhiều trường hợp dẫn đến *deadlock* nếu ta chỉ thiết lập các mức cô lập.

-Đầu tiên ta xem xét đến mức cô lập Serializable:

```
Create proc CapNhatGiaVung @mavung int, @heso int
As
Begin
      Begint ransaction
      set transaction isolation level serializable
            declare @ght int
            select @ght=Gia from Vung
            where Ma = @mavung
            waitfor delay '00:00:05'
```

```
            update Vung
            set  Gia = @ght * @heso
            where Ma = @mavung


      if(@@error<> 0)
            begin
                  rollback transaction
                  return
            end
      commit transaction
end
```

-Với mức cô lập này khi T1 chạy đến câu lệch *Select* nó được phát khóa đọc trên đơn vị dữ liệu vùng với mã tương ứng và chờ. Còn T2 khi đến lệnh *Select* nó cũng được phát khóa đọc trên dữ liệu này, 2 khóa đọc trên cùng 1 đơn vị dữ liệu thì tương thích với nhau.

-Nhưng trường hợp khóa ghi sau đây thì không tương thích với bất kỳ khóa nào khác.

-Khi cả 2 giao tác chuyển qua câu lệnh *Update* thì giao tác T1 yêu cầu phát khóa ghi nhưng lúc này dữ liệu đang bị khóa bởi khóa đọc của T2 nên T1 phải chờ T2 bỏ khóa trên dữ liệu vùng này. Tương tự, T2 khi chuyển qua câu lệnh *Update* cũng yêu cầu phát khóa ghi trên dữ liệu và cũng phải chờ T1 bỏ khóa đọc trên dữ liệu vì lúc này T1 vẫn giữ khóa đọc do việc xin phát khóa ghi chưa được đáp ứng . 2 giao tác chờ khóa lẫn nhau. Việc này dẫn đến tình trạng *deadlock* xảy ra.

-Tiếp theo ta thử với mức cô lập Repeatable Read

-Tình huống tương tự cũng xảy ra cho mức cô lập này. Khi cả 2 giao tác cùng chạy đến lệnh Select cùng yêu cầu phát khóa đọc trên dữ liệu nay.

-Khi cả hai cùng chuyển qua lệch *Update* thì lại cùng yêu cầu phát khóa ghi đẫn đến tình trạng chờ khóa lẫn nhau *deadlock* xảy ra.

-Cả 2 mức cô lập trên đều dẫn đến tình trạng *conversion deadlock*.

-Sau đây là cách giải quyết tình trạng *deadlock*.

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Createproc CapNhatGiaVung<br>@mavung int, @heso int<br>As<br>Begin<br>  begin transaction<br>  set transaction isolation level repeatable read<br>  declare @ght int<br>  Select @ght=Gia<br>  from Vung with(Updlock)<br>  where Ma = @mavung<br>  waitfor delay '00:00:05'<br><br>  update Vung<br>  set  Gia = @ght * @heso<br>  where Ma = @mavung<br><br>if(@@error<> 0)<br>  begin<br>  rollback transaction<br>  return<br>  end<br>  commit transaction<br>end | |
| exec CapNhatGiaVung 1, 2 | exec CapNhatGiaVung 1, 3 |

-Ta kết hợp mức cô lập với chế độ khóa lock hints của SQL Server cung cấp.

-Khi 2 giao tác T1 và T2 chạy đồng thời đến câu lệch *Select* cả 2 cùng yêu cầu phát khóa *Updlock*. Nhưng trong 1 thời điểm SQL Server  chỉ cho phép *1 Updlock* trên 1 đơn vị dữ liệu nên giao tác nào thực hiện trước sẽ được cấp *Updlock* trước và giao tác kia sẽ phai chờ đến khi giao tác này thực hiện xong *commit* xuống csdl thì giao tác còn lại mới được dùng đơn vị dữ liệu đó.

-Do việc xin cấp phát khóa ghi không xảy ra đồng thời nên tình trạng *deadlock* sẽ không xảy ra.

-Dữ liệu cũng được cập nhật đầy đủ không bị mất mát dữ liệu.

## Trường hợp 2:

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc CapNhatTongTien_f @mahd int, @sotien int<br>As<br>Begin<br>begin transaction<br>declare @t int<br>select @t = TongTien<br>from HopDong<br>where Ma = @mahd<br>waitfor delay '00:00:05'<br>update HopDong<br>set TongTien = @t + @sotien<br>where Ma = @mahd<br>if ( @@error <> 0)<br>begin<br>rollback transaction<br>return<br>end<br>commit transaction<br>end | |
| exec CapNhatTongTien_f 1, 1 | exec CapNhatTongTien_f 1, 2 |

-Khi T1 thực hiện việc cập nhật tổng tiền cho 1 hợp đồng của khách hàng, T2 cũng thực hiện hành động tương tự cùng hợp đồng đó. Lúc này dữ liệu cập nhật của giao tác thực hiện trước sẽ bị dữ liệu của giao tác thực hiện sau ghi đè lên dẫn đến việc dữ liệu cập nhật của T1 bị mất.

**Khắc phục:

- Nếu ta chỉ thiết lập các mức cô lập để giải quyết tình huống này gây nên nhiều trường hợp dẫn đến *deadlock* như trường hợp trên.

-Đầu tiên ta xem xét đến mức cô lập Repeatable Read

```
Create proc CapNhatTongTien @mahd int, @sotien int
As
Begin
     begin transaction
     set transaction isolation level repeatable read
           declare @t int
           Select @t = TongTien From HopDong
           Where Ma = @mahd
           waitfor delay '00:00:05'

           Update HopDong
```

```
            Set  TongTien = @t + @sotien
            Where Ma = @mahd


    if ( @@error <> 0)
     begin
           rollback transaction
           return
     end
     commit transaction
end
```

-Với mức cô lập này khi T1 chạy đến câu lệch *Select* nó được phát khóa đọc trên đơn vị dữ liệu hợp đồng với mã tương ứng và chờ. Còn T2 khi đến lệch *Select* nó cũng được phát khóa đọc trên dữ liệu này, 2 khóa đọc trên cùng 1 đơn vị dữ liệu thì tương thích với nhau.

-Nhưng trường hợp khóa ghi sau đây thì không tương thích với bất kỳ khóa nào khác.

-Khi cả 2 giao tác chuyển qua câu lệnh *Update* thì T1 sẽ chờ khóa đọc của T2 kia được gỡ bỏ và T2 cũng sẽ chờ T1 giải phóng khóa đọc vì khóa ghi mà cả hai yêu cầu phát khóa ghi trên đơn vị dữ liệu không tương thích với các loại khóa khác. Xảy ra tình trạng giống như trường hợp 1 tình trạng dealock xảy ra.

-Tiếp theo ta thử với mức cô lập Serializable

-Tình huống tương tự cũng xảy ra cho mức cô lập này. Khi cả 2 giao tác cùng chạy đến lệnh *Select* cùng yêu cầu phát khóa đọc trên dữ liệu nay.

-Khi cả hai cùng chuyển qua lệch *Update* thì lại cùng yêu cầu phát khóa ghi đẫn đến tình trạng chờ khóa lẫn nhau *deadlock* xảy ra.

-Cả 2 mức cô lập trên đều dẫn đến tình trạng *conversion deadlock*.

-Sau đây là cách giải quyết tình trạng *deadlock*.

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc CapNhatGiaVung<br>@mavung int, @heso int<br>As<br>Begin<br>begin transaction<br>set transaction isolation<br>level repeatable read<br>declare @ght int<br>select @ght=Gia<br>from Vung with(Updlock)<br>where Ma = @mavung<br>waitfor delay '00:00:05'<br><br>update Vung<br>set Gia = @ght * @heso<br>where Ma = @mavung<br><br>if(@@error<> 0)<br>begin<br>rollback transaction<br>return<br>end<br>commit transaction<br>end | |
| exec CapNhatGiaVung 1, 2 | exec CapNhatGiaVung 1, 3 |

-Ta kết hợp mức cô lập với chế độ khóa lock hints của SQL Server cung cấp.

-Khi 2 giao tác T1 và T2 chạy đồng thời đến câu lệch Select cả 2 cùng yêu cầu phát khóa *Updlock*. Nhưng trong 1 thời điểm SQL Server chỉ cho phép 1 *Updlock* trên 1 đơn vị dữ liệu nên giao tác nào thực hiện trước sẽ được cấp *Updlock* trước và giao tác kia sẽ phai chờ đến khi giao tác này thực hiện xong *commit* xuống csdl thì giao tác còn lại mới được dùng đơn vị dữ liệu đó.

-Do việc xin cấp phát khóa ghi không xảy ra đồng thời nên tình trạng deadlock sẽ không xảy ra.

-Dữ liệu cũng được cập nhật đầy đủ không bị mất mát dữ liệu.

## 4.2 Dirty Read

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc RollCapNhatGiaVung<br>@mavung int, @gia int<br>As<br>Begin<br>  begin transaction<br>  update Vung<br>  set  Gia = @gia<br>  where Ma = @mavung<br>  waitfor delay '00:00:05'<br><br>  rollback transaction<br>end | Create proc DirtyRead @mavttint<br>As<br>Begin<br>  select v.Gia, t.TiLeGia, kt.KichThuoc<br>  from VungThuocTrang vtt with (Nolock), Vung v with (Nolock), KichThuoc kt with (Nolock), Trang t with (Nolock)<br>  where vtt.MaVung = v.Ma and v.KichThuoc = kt.Ma and vtt.Ma = @mavtt and t.Ma = vtt.MaTrang |
| exec RollCapNhatGiaVung 1, 1 | exec DirtyRead 1 |

-Khi T1 update lại giá của 1 vùng có mã = 1 với giá mới = 1 đồng thời lại có 1 khách hàng T2 đọc dữ liệu vùng đó lên. Giao tác của T1 bị *rollback* khiến cho dữ liệu của T2 đọc lên là dữ liệu rác, không có trên csdl.

-Khi ta đặt khóa Nolock hay Read Uncommitted tương đương với việc không đặt khóa thì dữ liệu đọc của T2 sẽ sai vì dữ liệu đó chưa được *commit*.

**Khắc phục:

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc RollCapNhatGiaVung<br>@mavung int, @gia int<br>As<br>Begin<br>  begin transaction<br>  update Vung<br>  set  Gia = @gia<br>  where Ma = @mavung<br>  waitfor delay '00:00:05'<br><br>  rollback transaction<br>end | Begin<br>  select v.Gia, t.TiLeGia, kt.KichThuoc<br>  from VungThuocTrang vtt, Vung v, KichThuoc kt, Trang t<br>  where vtt.MaVung = v.Ma and v.KichThuoc = kt.Ma and vtt.Ma = @mavtt and t.Ma = vtt.MaTrang<br>End |
| exec RollCapNhatGiaVung 1, 1 | exec DirtyRead 1 |

-Trả lại thiết lập khóa mặc định của SQL  cho câu lệch Select như vậy sẽ không có hiện tượng trên.

-Đây thực sự là trường hợp giả lập vì câu lệch Select đã được SQL thiết lập chế độ khóa mặc định là Readcommitted tạo khóa đọc

và chỉ đọc những dữ liệu đã được *commit* → tình trạng đọc dữ liệu rác của T2 được giải quyết.

## 4.3 Unrepeatable Read:

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc DocDuLieuTrang_f<br>@matrang int<br>As<br>Begin<br>begin transaction<br>select t.TenTrang, v.KichThuoc, v.Gia, t.TiLeGia<br>from Vung v, Trang t, VungThuocTrang vtt<br>where v.Ma = vtt.MaVung and t.Ma = vtt.MaTrang and t.Ma = @matrang<br>waitfor delay '00:00:05'<br><br>select t.TenTrang, v.KichThuoc, v.Gia, t.TiLeGia<br>from Vung v, Trang t, VungThuocTrang vtt<br>where v.Ma = vtt.MaVung and t.Ma = vtt.MaTrang and t.Ma = @matrang<br>if(@@error<> 0)<br>begin<br>rollback transaction<br>return<br>end<br>commit transaction<br>end | Create proc CapNhatTiLeGiaTrang<br>@matrang int, @tlgia int<br>As<br>Begin<br>begin transaction<br>update Trang<br>set TiLeGia = @tlgia<br>where Ma = @matrang<br><br>if(@@error<> 0)<br>begin<br>rollback transaction<br>return<br>end<br>commit transaction<br>end |
| exec DocDuLieuTrang_f 1 | exec CapNhatTiLeGiaTrang 1, 3 |

-Trong khi T1 thực hiện việc đọc dữ liệu vùng của trang có mã = 1 với tỉ lệ giá = 3 thì trong lúc này giao tác T2 lại thực hiện cập nhật tỉ lệ giá của trang đó làm cho dữ liệu tính toán giá của những vùng thuộc trang đó không giống với kết quả ban đầu T1 đọc lên.

-Xảy ra tình trạng T1 không thể đọc lại dữ liệu ban đầu.

**Khắc phục:

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc DocDuLieuTrang<br>@matrang int<br>As<br>Begin<br>  begin transaction<br>  set transaction isolation level repeatable read<br>  select t.TenTrang, v.KichThuoc, v.Gia, t.TiLeGia<br>  from Vung v, Trang t, VungThuocTrang vtt<br>  where v.Ma = vtt.MaVung and t.Ma = vtt.MaTrang and t.Ma = @matrang<br>  waitfor delay '00:00:05'<br><br>  select t.TenTrang, v.KichThuoc, v.Gia, t.TiLeGia<br>  from Vung v, Trang t, VungThuocTrang vtt<br>  where v.Ma = vtt.MaVung and t.Ma = vtt.MaTrang and t.Ma = @matrang<br>  if(@@error<> 0)<br>  begin<br>  rollback transaction<br>  return<br>  end<br>  commit transaction<br>end | Create proc CapNhatTiLeGiaTrang<br>@matrang int, @tlgia int<br>As<br>Begin<br>  begin transaction<br>  update Trang<br>  set  TiLeGia = @tlgia<br>  where Ma = @matrang<br><br>  if(@@error<> 0)<br>  begin<br>  rollback transaction<br>  return<br>  end<br>  commit transaction<br>end |
| exec DocDuLieuTrang 1 | exec CapNhatTiLeGiaTrang 1, 3 |

-Khi ta thiết lập mức cô lập Repeatable Read cho giao tác T1 thì mọi dữ liệu đọc trong giao tác đều được phát khóa đọc và khóa đọc này được giữ đến hết giao tác.

-Giao tác T1 thực hiện đến câu lệch *Select* thì được phát khóa đọc trên dữ liệu thuộc trang 1 đến đây giao tác T2 thực hiện câu lệch *Update* nên T2 sẽ yêu cầu phát khóa ghi trên dữ liệu trang 1. Do dữ liệu đã có khóa đọc của T1 nên yêu cầu của T2 không được đáp ứng do đó T2 phải chờ T1 thực hiện xong giao tác vì quy định của mức cô lập trên T1 → tình trạng không thể đọc lại dữ liệu của T1 được giải quyết.

## 4.4 Phantom:

### Delete

-Tìm những hợp đồng của khách hàng chưa đến công ty thực hiện giao dịch trong vòng 2 ngày trong 1 số báo. Trong lúc này thì một giao tác khác lại xóa đi 1 hợp đồng trong danh sách đó làm cho khách hàng có hợp đồng đó chưa được xử lý.

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc DSKhachHangChuaGD_f<br>@masb int<br>as<br>Begin<br>  begin transaction<br>  set transaction isolation level read uncommitted<br>  select  nd.Ten, hd.Ma<br>  from SoBao sb ,NguoiDung nd ,HopDong hd ,ChiTietSoBao ctsb<br>  where sb.Ma = ctsb.MaSoBao and nd.Ma = hd.MaNguoiDung and hd.Ma = ctsb.MaHopDong and ctsb.MaSoBao = @masb and hd.TTGiaoDich = 0<br>  anddatediff(day, hd.NgayDangKy,getdate())> 2<br>  groupby  nd.Ten, hd.Ma<br>  waitfor delay '00:00:05'<br><br>  select  nd.Ten, hd.Ma<br>  from SoBao sb ,NguoiDung nd ,HopDong hd ,ChiTietSoBao ctsb<br>  where sb.Ma = ctsb.MaSoBao and nd.Ma = hd.MaNguoiDung and hd.Ma = ctsb.MaHopDong and ctsb.MaSoBao = @masb and hd.TTGiaoDich = 0<br>  anddatediff(day, hd.NgayDangKy,getdate())> 2<br>  groupby  nd.Ten, hd.Ma<br><br>if(@@error<> 0)<br>  begin<br>  rollback transaction<br>  return<br>  end<br>  commit transaction<br>end | Create proc XoaHopDongHetHan<br>@mahd int<br>as<br>Begin<br>  begin transaction<br>  delete<br>  from HopDong<br>  where Ma = @mahd<br><br>  if ( @@error <> 0)<br>  begin<br>  rollback transaction<br>  return<br>  end<br>  commit transaction<br>end |
| exec DSKhachHangChuaGD_f 1 | exec XoaHopDongHetHan 1 |

-Khi chạy T1 giao tác sẽ tìm danh sách những khách hàng có hợp đồng trong số báo đó nhưng chưa đến công ty thực hiện giao dịch để xử lý. Với mức cô lập Read Uncommitted được thiết lập cho T1 sẽ không thiết lập khóa đọc trên đơn vị dữ liệu đọc và đọc dữ liệu chưa *commit* .

-Khi T1 đọc lần đầu sẽ có kết quả là 1 danh sách sau đó chờ. Trong lúc đó T2 đã chạy và *delete* 1 hợp đồng của 1 khách hàng trong danh sách của T1 do mức cô lập đã thiết lập trước đó cho T1 nên T2 có quyền *delete*. Việc này dẫn đến lần đọc sau của T1 sẽ cho ra kết quả là danh sách đó đã bị mất đi một khách hàng → hiện tượng bóng ma là khách hàng bị mất đi trong danh sách.

**Khắc phục:

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|

```
Create proc DSKhachHangChuaGD
@masb int
as
Begin
   begin transaction
      set transaction isolation
level serializable
      select  nd.Ten, hd.Ma
      from SoBao sb ,NguoiDung nd
,HopDong hd ,ChiTietSoBao ctsb
      where sb.Ma = ctsb.MaSoBao
and nd.Ma = hd.MaNguoiDung and
hd.Ma = ctsb.MaHopDong and
ctsb.MaSoBao = @masb and
hd.TTGiaoDich = 0
anddatediff(day,
hd.NgayDangKy,getdate())> 2
      groupby  nd.Ten, hd.Ma
      waitfor delay '00:00:05'

      select  nd.Ten, hd.Ma
      from SoBao sb ,NguoiDung nd
,HopDong hd ,ChiTietSoBao ctsb
      where sb.Ma = ctsb.MaSoBao
and nd.Ma = hd.MaNguoiDung and
hd.Ma = ctsb.MaHopDong and
ctsb.MaSoBao = @masb and
hd.TTGiaoDich = 0
anddatediff(day,
hd.NgayDangKy,getdate())> 2
      groupby  nd.Ten, hd.Ma
if(@@error<> 0)
      begin
            rollback transaction
            return
      end
```

| | |
|---|---|
| commit transaction<br>end | |
| exec DSKhachHangChuaGD 1 | exec XoaHopDongHetHan 1 |

-Với trường hợp này ta dùng mức cô lập Serializable sẽ tạo *shared lock* trên dữ liệu đọc và được giữ đến hết giao tác. Điểm mạnh của mức cô lập này là nó không cho ghi những dòng dữ liệu thỏa điều kiện thiết lập.

-Khi T1 chạy xong câu *Select* và chờ, T2 chạy đến câu lệnh tạo cursor thì tạo khóa đọc 2 khóa đọc ở 2 giao tác tương thích. Nhưng khi T2 chạy đến lệch *delete* sẽ bị buộc dừng lại vì mức cô lập Serializable trên đơn vị dữ liệu này (trường hợp này là những khách hàng đã hết hạn giao dịch trong 1 số báo) đang được thiết lập trên T1 không cho phép *delete* trên dữ liệu dù thỏa điều kiện thiết lập khóa nên T2 không thể xin khóa ghi tren dữ liệu đó không thể *delete* khách hàng này trong lúc này được mà phải chờ T1 kết thúc → giải quyết hiện tượng bóng ma

Insert:

-Tra cứu số lượng hợp đồng theo tổng. Trong lúc này thì một giao tác khác lại thêm vào 1 hợp đồng mới ứng theo chi tiết số.

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc DemHopdong_f @masb int, @tongtien int<br>As<br>Begin<br>  begin transaction<br>  set transaction isolation level serializable<br>  select COUNT(*) as SoHopDong<br>  from SoBao sb,ChiTietSoBao ctsb ,HopDong hd,NguoiDung nd<br>  where sb.Ma=ctsb.MaSoBao and hd.Ma=ctsb.MaHopDong and nd.Ma=hd.MaNguoiDung and sb.Ma=@masb and hd.TongTien>@tongtien<br>  group by  sb.SoBao<br><br>  waitfor delay '00:00:05'<br><br>  select COUNT(*) as SoHopDong<br>  from SoBao sb,ChiTietSoBao ctsb ,HopDong hd,NguoiDung nd | Create proc ThemHopDong @mand int, @ngay datetime,@mavungtt int,@masb int, @tongtien int<br>as<br>Begin<br>  begin transaction<br>  declare @i int<br>  set @i =( Select MAX(Ma) from HopDong) + 1<br>  insert into HopDong values(@i,@mand,@ngay,0,@tongtien);<br>  declare @ii int<br>  set @ii =( Select MAX(Ma) from ChiTietSoBao ) + 1<br>  insert into ChiTietSoBao values(@ii,@i,@mavungtt,@masb);<br><br>  if ( @@error <> 0)<br>  begin<br>  rollback transaction<br>  return<br>  end<br>  commit transaction<br>end |

| | |
|---|---|
| where sb.Ma=ctsb.MaSoBao and hd.Ma=ctsb.MaHopDong and nd.Ma=hd.MaNguoiDung and sb.Ma=@masb and hd.TongTien>@tongtien<br>group by sb.SoBao<br>if (@@error<> 0)<br>begin<br>rollback transaction<br>return<br>end<br>commit transaction<br>end | |
| exec DemHopdong_f 1,15000 | exec ThemHopDong 1, '06-02-2011',1,1,20000 |

-Khi ta chạy 2 transaction đồng thời T1 và T2, thì kết quả T1 sẽ cho ra 2 kết quả khác nhau, do trong quá trình T1 chạy, thông qua khoảng thời gian chờ "waitfordelay'00:00:05'", T2 đã thêm vào 1 hợp đồng mới thỏa điều kiện thiết lập khóa của T1(là khóa Repeatable Read) ứng với khách hàng đã có tài khoản, và đã chọn 1 số báo nhất định (mà TH này là thông qua 2 bảng HopDong, ChiTietSoBao) .

-TH này dẫn đến T1 sẽ có sự sai lệch khi đọc trên cùng 1 CSDL nhưng cho 2 kết quả khác nhau, và việc thêm mới dữ liệu của T2 giốngnhư " bóng ma" → hiện tượng bóng ma.

**Khắc phục:

| Transaction 1 (T1) | Transaction 2 (T2) |
|---|---|
| Create proc DemHopdong_f @masb int, @tongtien int<br>As<br>Begin<br>begin transaction<br>set transaction isolation level serializable<br>select COUNT(*) as SoHopDong<br>from SoBao sb,ChiTietSoBao ctsb ,HopDong hd,NguoiDung nd<br>where sb.Ma=ctsb.MaSoBao and hd.Ma=ctsb.MaHopDong and nd.Ma=hd.MaNguoiDung and sb.Ma=@masb and hd.TongTien>@tongtien<br>group by sb.SoBao<br><br>waitfor delay '00:00:05'<br><br>select COUNT(*) as SoHopDong<br>from SoBao sb,ChiTietSoBao ctsb ,HopDong hd,NguoiDung nd<br>where sb.Ma=ctsb.MaSoBao and hd.Ma=ctsb.MaHopDong and nd.Ma=hd.MaNguoiDung and sb.Ma=@masb and hd.TongTien>@tongtien<br>group by sb.SoBao<br>if (@@error<> 0)<br>begin<br>rollback transaction<br>return<br>end<br>commit transaction<br>end | Create proc ThemHopDong @mand int, @ngay datetime,@mavungtt int,@masb int<br>as<br>Begin<br>begin transaction<br>declare @i int<br>set @i =(Selectcount(*)from HopDong )+ 1<br>insert into HopDong values(@i,@mand,@ngay,0,20000);<br>declare @ii int<br>set @ii =(Selectcount(*)from ChiTietSoBao )+ 1<br>insert into ChiTietSoBao values(@ii,@i,@mavungtt,@masb);<br><br>if (@@error<> 0)<br>begin<br>rollback transaction<br>return<br>end<br>commit transaction<br>end |
| exec DemHopdong_f 1,15000 | exec ThemHopDong9,'06-01-2011',1,2 |

-Do T1 ban đầu sử dụng mức cô lập Repeatable Read,với mức cô lập này, nó sẽ tạo *shared lock* trên đơn vị dữ liệu mà T1 đang đọc ,cho đến hết giao tác, nhưng khi T2 thực hiện thao tác thêm 1 dòng dữ liệu thỏa điều kiện của T1 thì việc đụng độ vẫn xảy ra.

- Nên để giải quyết trường hợp này , ta cần sử dụng mức cô lập Serializable,để ngăn việc *insert* 1 dòng dữ liệu thỏa mãn điều kiện thiết lập *shared lock*. Như trong TH trên thì khi T1 chạy mức cô lập Serializable, sẽ phát khóa đọc trên các bảng

(SoBao,ChiTietSoBao,HopDong,NguoiDung), và ở T2 cũng sẽ phát khóa đọc (do SQL Server tự động thiết lập) cho các bảng (HopDong,ChiTietSoBao), nhưng khi T2 không thể thực hiện việc thêm 1 dòng dữ liệu vào 2 bảng HopDong và ChiTietSoBao, dù cả ban đầu 2 khóa đọc của 2 giao tác là tương thích nhau → giải quyết hiện tượng bóng ma.

## 5.Bổ sung

### 5.1 Trigger cho việc kiểm tra xóa hợp đồng khách hàng đã hết hạn giaodịch

+Nếu như việc kiểm tra đã hết hạn là đúng thì sẽ cho xóa.

+Nếu sai thì sẽ xuất hiện thông báo và phục hồi lại hợp đồng của khách hàng đó.

```
Create trigger trg_KiemTraGiaoDich on HopDong
For Delete
As
Begin

      if exists (Select * From Deleted D
                  Where  TTGiaoDich = 0 and datediff(day,
NgayDangKy, getdate()) < 2 )
      Begin
            Raiserror('Chưa hết hạn 2 ngày', 0, 1)
            Rollback transaction
      End

End
```

### 5.2 Trigger cho việc kiểm tra thêm 1 hợp đồng mới.

+Nếu ngày đăng ký đã qua (số báo cũ) thì xuất hiện thông báo và không cho thêm hợp đồng.

+Nếu ngày đăng ký là tương lai (số báo sắp phát hành) thì cho phép thêm hợp đồng.

```
Create trigger trg_KiemTraSoBaoCu on HopDong
For Insert
As
Begin

      if exists (Select * From Inserted I
                  Where Convert(varchar(10),NgayDangKy,101) <=
Convert (varchar(10),getdate(),101 ) )
      Begin
            Raiserror('Đây là số báo cũ', 0, 1)
            Rollback transaction
      End
```

*The End*