Các đối tượng trong SQL Server

LẠI HIỀN PHƯƠNG

EMAIL: LHPHUONG@TLU.EDU.VN

Nội dung

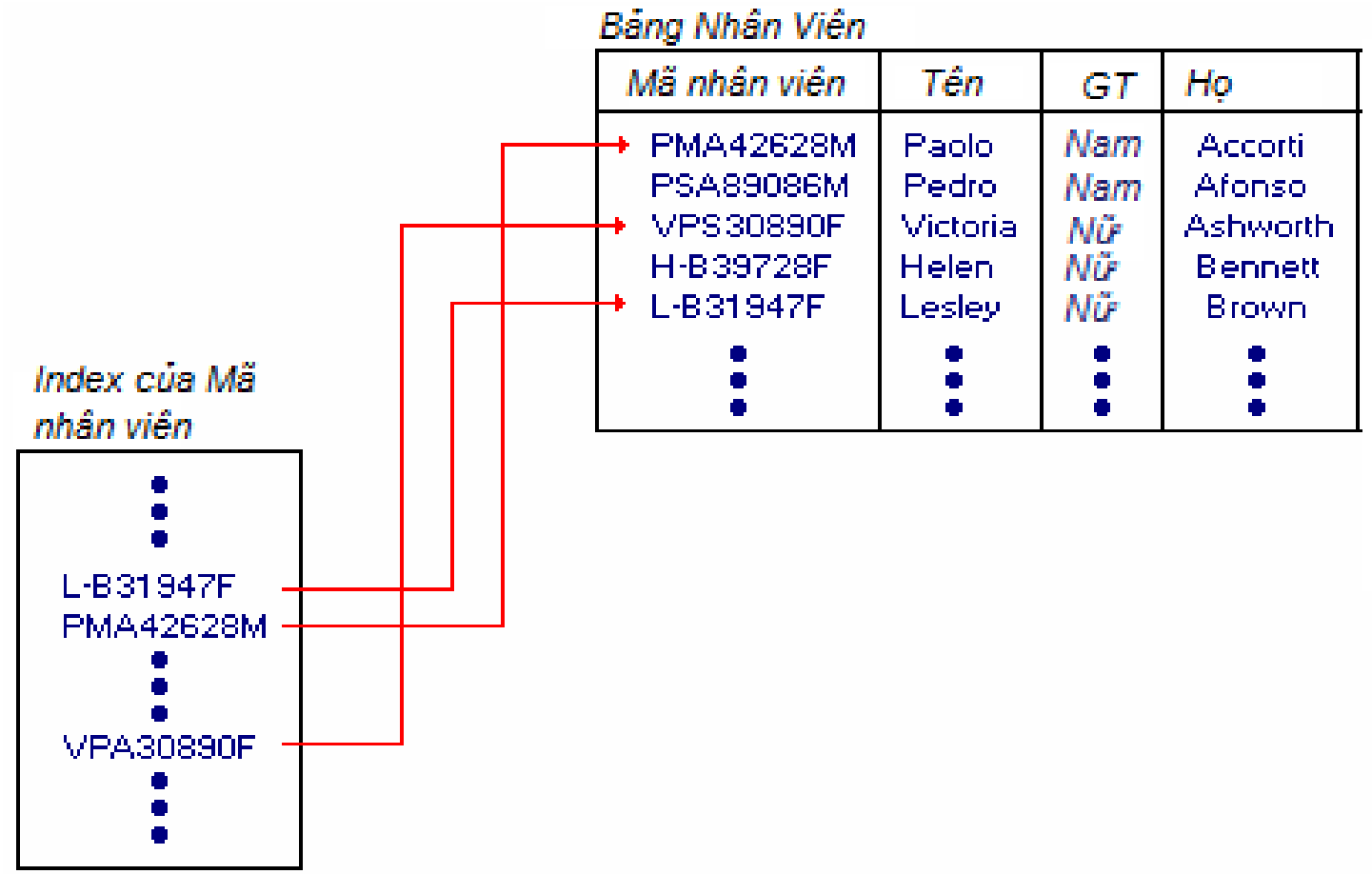
- View
- Chỉ mục
- Trigger
- Transaction và Lock

Chỉ mục - Index

Khái niệm chỉ mục (Index)

- Index giúp tăng tốc độ truy vấn dữ liệu bằng cách cung cấp phương pháp truy xuất nhanh chóng tới các dòng trong bảng, tương tự như mục lục của cuốn sách
- Index được thiết lập từ một hoặc nhiều cột của bảng hay view
  - Các giá trị của index sẽ được sắp xếp và lưu trữ theo một danh sách
  - Mỗi giá trị index là duy nhất trong danh sách
  - Mỗi giá trị index sẽ liên kết đến giá trị trong bảng dữ liệu (liên kết dạng con trỏ)

Khái niệm chỉ mục (Index) (tiếp)

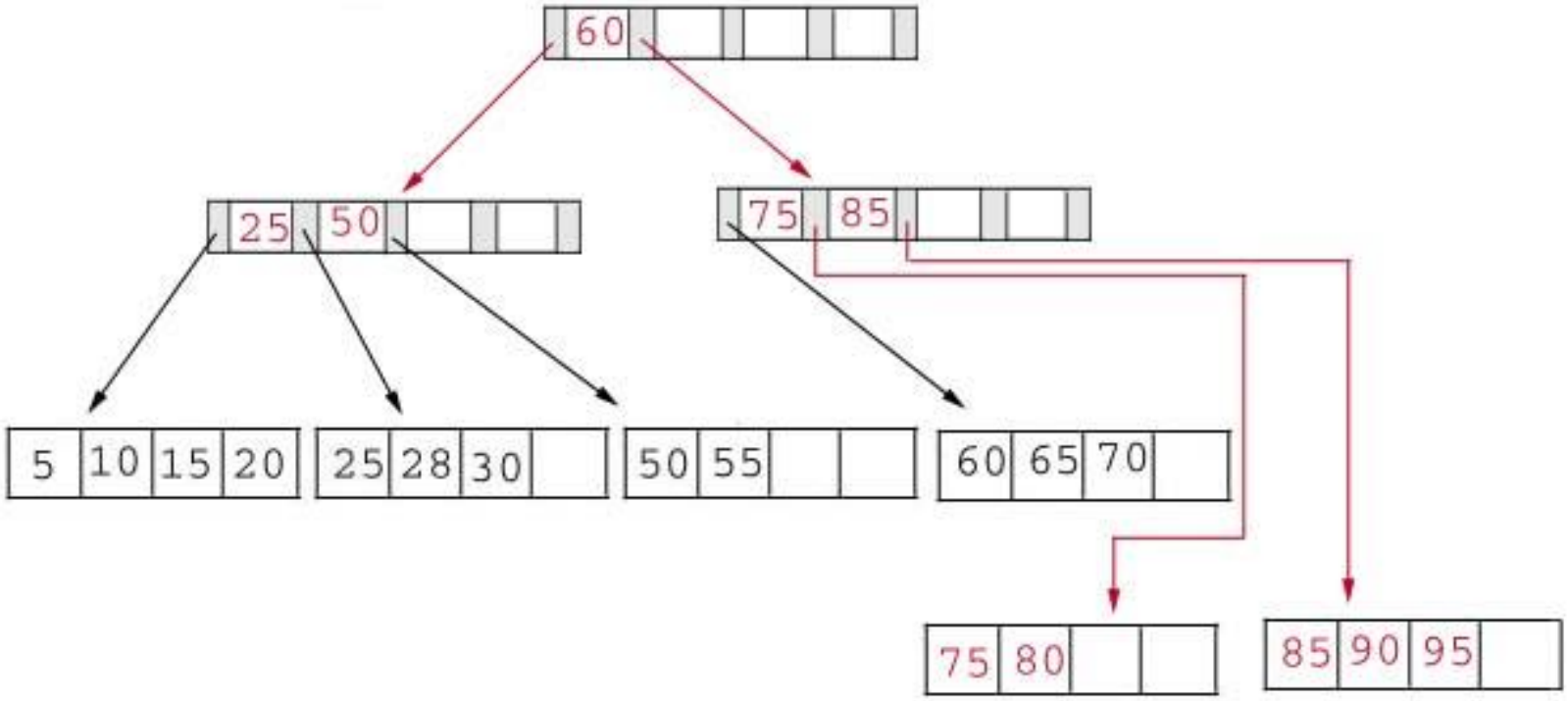


Khái niệm chỉ mục (Index) (tiếp)

- Index trong SQL Server được tạo thành từ một tập các page (các index node) và chúng được tổ chức trong một cấu trúc cây B-tree để tăng tốc độ truy xuất dữ liệu
- Khi tìm kiếm một giá trị trong một cột dữ liệu
  - Nếu không có Index, SQL Server sẽ thực hiện động tác quét qua toàn bộ bảng dữ liệu để xác định vị trí dòng cần tìm.
  - Nếu cột cần tìm tham gia tạo index, đầu tiên SQL Server sẽ tìm vị trí của giá trị này trong bảng index bằng phép duyệt cây, sau đó thực hiện tìm theo liên kết con trỏ đến bản ghi chứa giá trị tương ứng với index trong bảng dữ liệu

Khái niệm chỉ mục (Index) (tiếp)

- Ví dụ: tìm giá trị 65, 76 trong cột được tạo index



Phân loại Index

- **Clustered Index**
  - Lưu trữ và sắp xếp dữ liệu vật lý trong các bảng và view dựa trên giá trị khóa của chúng. Các cột khóa này được chỉ định trong định nghĩa index.
  - Mỗi bảng hoặc view chỉ có duy nhất một Clustered index vì bản thân các dòng dữ liệu được lưu trữ và sắp xếp vật lý theo giá trị của cột trong index.
  - Khi một table có một clustered index thì gọi là clustered table

Phân loại Index (tiếp)

- **Non-Clustered Index**
  - Index được lưu ở một vùng khác so với bản thân dữ liệu. Mỗi index chứa các giá trị của các cột khóa trong khai báo của index và có con trỏ tới dòng dữ liệu tương ứng trong bảng.
  - Dữ liệu không sắp xếp ở dạng vật lý mà chỉ sắp xếp logic, tức là chỉ có các giá trị khóa trong index được sắp xếp
  - Một bảng có thể có tối đa 249 Non-Clustered Index
  - Mặc định lệnh CREATE INDEX tạo ra non-clustered index

Phân loại Index (tiếp)

- **Phân loại theo cách khác, ta có**
  - ***Composite index***: là kiểu index có nhiều hơn một cột, có thể là clustered hoặc non-clustered index
  - ***Unique index***: là kiểu index dùng để đảm báo tính duy nhất trong các cột được tạo index.
    - Khi định nghĩa **Primary Key**, SQL Server tự động tạo ra một **unique clustered index** nếu chưa có một clustered index nào tồn tại trên bảng hoặc view
    - Khi định nghĩa một ràng buộc **Unique**, SQL Server tự động tạo một **unique non-clustered index**. Người dùng có thể tạo unique clustered index nếu chưa có một clustered index nào được tạo trước đó trên bảng.

Chú ý khi tạo Index

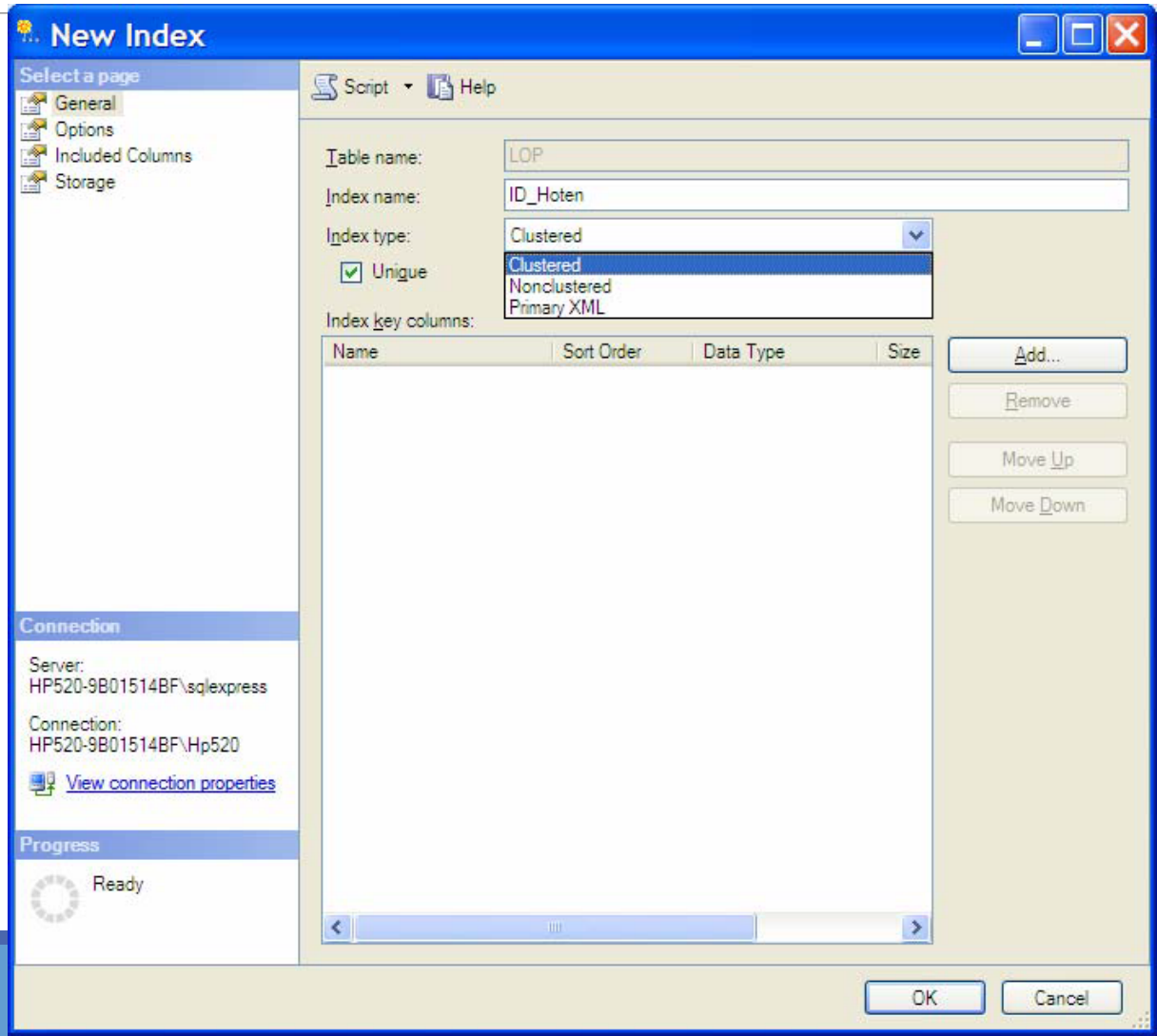
- Index có thể chiếm nhiều không gian của ổ cứng, do đó không nên tạo quá nhiều Index nếu không thực sự cần
- Index sẽ được tự động cập nhật khi bản thân các dòng dữ liệu được cập nhật, có thể giảm hiệu xuất xử lý dữ liệu
- Đối với các bảng được cập nhật dữ liệu thường xuyên, nên sử dụng càng ít cột càng tốt trong một index
- Đối với bảng có nhiều dữ liệu nhưng tần suất cập nhật dữ liệu thấp, nên sử dụng nhiều index để tăng hiệu suất truy vấn

Chú ý khi tạo Index (tiếp)

- Cần cân nhắc việc sử dụng index trên các bảng nhỏ vì việc tìm kiếm trên index có thể mất nhiều thời gian hơn duyệt bảng để tìm dữ liệu.
- Đối với clustered index, cố gắng giữ cho độ dài các cột được lập index càng ngắn càng tốt
- Đối với composite index:
  - Cột nào thường được sử dụng trong các biểu thức so sánh ở mệnh đề where sẽ được liệt kê đầu tiên.
  - Với các cột tiếp theo, cột nào có tính duy nhất của giá trị trong cột càng cao thì càng được liệt kê trước.

Tạo index bằng SQL Server Management Studio

- Mở rộng Table hoặc view muốn tạo index
- Nhấn chuột phải lên Indexes, chọn New Index
  - Index name: tên chỉ mục muốn tạo
  - Index type: kiểu chỉ mục
  - Index key columns: xác định các trường khóa của index



Tạo index bằng T-SQL

- Cú pháp:

```
CREATE [UNIQUE] [CLUSTERED] [NON CLUSTERED]
INDEX  tên_index
ON tên_bảng (tên_cột_1, tên_cột_2, …)
```

- Các tùy chọn:
  - UNIQUE có thể được chọn đồng thời với các tùy chọn khác
  - Chỉ được chọn hoặc CLUSTERED hoặc NON CLUSTERED
  - Mặc định là NON CLUSTERED nếu không chỉ định

Tạo index bằng T-SQL (tiếp)

- **Ví dụ**: Tạo Non-clustered index cho cột NgayDatHang của bảng DonHang trong CSDL QuanLyKhachHang

```
CREATE INDEX  index_NgayDatHang
ON DonHang (NgayDatHang)
```

Xem index bằng T-SQL

- Để xem index của một bảng hay view:
  - Cú pháp:

    Sp_helpindex Tên_bảng_hoặc_Tên_view
  - Ví dụ:

    Sp_helpindex DonHang

Xóa index bằng T-SQL

- Để xóa index của một bảng hay view:
  - Cú pháp:

    Drop index Tên_bảng.Tên_index
  - Ví dụ:

    Drop index DonHang.index_NgayDatHang

Sử dụng index trong câu truy vấn

- Trong một câu lệnh SQL, một điều kiện tìm kiếm ở mệnh đề WHERE được gọi là sargable (Search Argument-Able) nếu index có thể được sử dụng khi thực hiện câu lênh
- Ví dụ:

```
SELECT * from SinhVien WHERE MaSV = 15
```

Điều kiện MaSV = 15 là sargable vì nó cho phép index trên cột MaSV được sử dụng

Sử dụng index trong câu truy vấn (tiếp)

- Cần viết code sao cho các điều kiện tìm kiếm trở thành **sargable** vì index giúp tăng hiệu năng của câu lệnh lên rất nhiều
- **Nguyên tắc**:
  - Cột cần tìm phải đứng một mình ở một phía của biểu thức hay nói cách khác là không có hàm số hay phép tính toán nào áp dụng trên cột đó

Sử dụng index trong câu truy vấn (tiếp)

- Ví dụ:

```
-- câu lệnh 1 (non-sargable)
SELECT * FROM Individual
WHERE CustomerID+2 = 11002

-- câu lệnh 2 (sargable)
SELECT * FROM Individual
WHERE CustomerID = 11000
```

- Câu lệnh 1 index không được sử dụng vì khi áp dụng một phép tính toán trên cột, hệ thống phải thực hiện tính toán trên từng node trên cây index trước khi so sánh, do vậy phải duyệt tuần tự qua tất cả các node thay vì tìm theo kiểu nhị phân như với câu lệnh 2

Sử dụng index trong câu truy vấn (tiếp)

- Trên thực tế, với ví dụ trên, thời gian chạy là:

```
Câu lệnh 1 (non-sargable):
Table 'Individual'. Scan count 1, logical reads 3088, physical reads 35
CPU time = 0 ms, elapsed time = 259 ms.

Câu lệnh 2 (sargable):
Table 'Individual'. Scan count 0, logical reads 3, physical reads 3
CPU time = 0 ms, elapsed time = 19 ms.
```

Sử dụng index trong câu truy vấn (tiếp)

- **Ví dụ**: khi tìm tất cả các đơn hàng được thực hiện trong tháng 9 năm 2017

```
-- cách 1: non-sargeable
select * from DonHang
where month(NgayDatHang) = 9
and year(NgayDatHang) = 2017

-- cách 2: sargeable
select * from DonHang
where NgayDatHang >= '20170901' and NgayDatHang <= '20170930'
```

Sử dụng index trong câu truy vấn (tiếp)

- **Ví dụ 2**: tìm tất cả các khách hàng có họ tên bắt đầu bằng chữ N

```
-- các cách non-sargeable
select * from KhachHang
where Left(HoTen,1) = 'N'
select * from KhachHang
where substring(HoTen,1,1) = 'N'

-- cách sargeable
select * from KhachHang
where Hoten like 'N%'
```

Trigger

Khái niệm Trigger

- Trigger là một kiểu stored procedure đặc biệt
  - Trigger không có tham số đầu vào, đầu ra
  - Không thể thực thi bằng tay bằng lệnh EXECUTE
  - Trigger được kích hoạt thực hiện một cách tự động khi có các sửa đổi trên dữ liệu (INSERT, UPDATE, DELETE) hoặc các sửa đổi lược đồ dữ liệu (CREATE, ALTER, DROP) liên quan đến Trigger

Khái niệm Trigger (tiếp)

- Trigger được sử dụng trong việc
  - Kiểm tra dữ liệu nhập, đảm bảo sự toàn vẹn cho dữ liệu bằng cách ngăn không cho những thay đổi không nhất quán được thực hiện
    - ***Ví dụ***: Lương của nhân viên không thể cao hơn lương của người quản lý
  - Dùng để tính toán, cập nhật dữ liệu tự động
    - ***Ví dụ***: Tính lại giá trị trường **ThanhTien** trong bảng **DonHang** khi giá trị **DonGia** của sản phẩm được thay đổi.

Các loại Trigger

- Trigger được chia thành 2 nhóm
  - DML triggers (hay Standard Triggers): thực thi khi người sử dụng sửa đổi dữ liệu thông qua các lệnh thao tác dữ liệu INSERT, UPDATE, DELETE trên bảng hoặc View
  - DDL Triggers: thực thi khi có các sự kiện định nghĩa lược đồ dữ liệu thông qua các lệnh CREATE, ALTER và DROP. Đây là nhóm mới, được bổ sung từ SQL Server 2005 Database Engine.

Tạo DML Trigger

- Cú pháp:

```
CREATE TRIGGER Tên_trigger
ON Tên_bảng_hoặc_tên_view
[WITH ENCRYPTION]
{FOR | AFTER | INSTEAD OF}
{[DELETE] [,] [INSERT] [,] [UPDATE]}
AS Câu_lệnh_SQL
```

Tạo DML Trigger (tiếp)

- AFTER:
  - Trigger được thực thi sau khi tất cả các câu lệnh SQL gây ra trigger được thực thi thành công
  - AFTER là kiểu mặc định nếu từ khóa FOR được dùng
  - Không thể định nghĩa AFTER Trigger cho view
- INSTEAD OF:
  - Trigger được thực thi thay cho các câu lệnh SQL gây ra trigger
  - INSTEAD OF trigger dùng được cho cả bảng và view

Tạo DML Trigger (tiếp)

- DELETE, INSERT, UPDATE:
  - Xác định câu lệnh mà khi thực thi trên bảng hoặc view sẽ gây ra Trigger
- WITH ENCRYPTION:
  - Mã hóa nội dung, ngăn chặn người dùng khác xem nội dung của trigger

Chú ý

- Trước khi có bất cứ câu lệnh nào chứa trigger được thực sự thực hiện, SQL Server tạo ra 2 bảng đặc biệt lưu trong bộ nhớ có cùng cấu trúc với bảng mà trên đó trigger được tạo
  - Bảng **INSERTED**: chứa các giá trị đang được add vào bảng
  - Bảng **DELETED**: chứa các giá trị đang bị xóa từ bảng
- Với **INSERT** Trigger, chỉ có bảng **INSERTED** được tạo
- Với **DELETE** Trigger, chỉ có bảng **DELETED** được tạo
- Với **UPDATE** trigger, cả 2 bảng **INSERTED** và **DELETED** được tạo

Chú ý (tiếp)

- Trong câu lệnh SQL của trigger, có thể thực hiện truy vấn 2 bảng INSERTED và DELETED để kiểm tra điều kiện toàn vẹn dữ liệu
- Trong câu lệnh SQL của trigger, có thể sử dụng:
  - IF UPDATE(tên cột): được dùng trong các trigger INSERT, UPDATE được dùng để kiểm tra xem có các sửa đổi trên cột chỉ định không
  - IF UPDATE(tên cột) [{AND | OR} UPDATE (tên_cột)]: được dùng khi kiểm trả các sửa đổi trên nhiều cột

Insert trigger: ví dụ

- Tạo trigger hiển thị thông báo mỗi khi thực hiện chèn thành công một bản ghi vào bảng KhachHang của CSDL QuanLyKhachHang

```
create Trigger insert trigger KH
ON KhachHang
FOR INSERT
AS Print N'Bạn đã chèn thành công'
```

Insert trigger: ví dụ

- Tạo trigger thực hiện tự động tính trường ThanhTien của bảng SP_DonHang(IDDonHang, IDSanPham, SoLuong, ThanhTien) khi thêm một bản ghi mới gồm IDDonHang, IDSanPham và SoLuong

```
create trigger ThanhTien
ON SP_DonHang
For Insert As
If ((select IDSanPham from inserted) is not null)
Begin
    Update SP_DonHang
    Set ThanhTien = SoLuong * DonGia
    from SanPham, (select IDSanPham, IDDonHang from Inserted) as I
    where SanPham.IDSanPham = SP_DonHang.IDSanPham
    and SP_DonHang.IDSanPham = I.IDSanPham
    and SP_DonHang.IDDonHang = I.IDDonHang
End
```

Insert trigger: ví dụ 2

- Trigger trong ví dụ sẽ được kích hoạt khi câu lệnh insert vào bảng SP_DonHang được thực hiện. Ví dụ

```
insert into SP_DonHang(IDDonHang,IDSanPham,SoLuong)
values (1,3,2)
```

Update trigger: ví dụ

- Tạo trigger hiển thị thông báo mỗi khi thực hiện cập nhật thành công một bản ghi của bảng KhachHang của CSDL QuanLyKhachHang

```
CREATE TRIGGER update_trigger_KH
ON KhachHang
FOR Update
AS
Print N'Bạn đã cập nhật thành công bảng KhachHang'
```

Update trigger: ví dụ 2

- Tạo trigger kiểm tra nếu người dùng muốn sửa IDKhachHang của bảng khách hàng thì không cho phép và hiển thị thông báo

```
CREATE TRIGGER update_IDKH
ON KhachHang
FOR Update AS
If UPDATE(IDKhachHang)
Begin
    Print N'Không thể thay đổi giá trị của trường IDKhachHang'
    ROLLBACK TRANSACTION
End
```

Update trigger: ví dụ 2

- Trigger trong ví dụ 2 sẽ được kích hoạt khi có người dùng muốn sửa đổi trường IDKhachHang trong bảng KhachHang

```
Update KhachHang
set IDKhachHang = 2
where IDKhachHang = 1
```

Delete trigger: ví dụ

- Tạo trigger hiển thị thông báo mỗi khi thực hiện xóa thành công một bản ghi của bảng KhachHang của CSDL QuanLyKhachHang

```
CREATE TRIGGER delete_trigger_KH
ON KhachHang
FOR DELETE
AS
Print N'Bạn đã xóa thành công bảng KhachHang'
```

Delete trigger: ví dụ 2

- Tạo trigger sao cho khi xóa 1 đơn hàng trong bảng DonHang, tất cả các dòng tương ứng trong bảng SP_DonHang cũng bị xóa

```
CREATE TRIGGER delete_trigger_DonHang
ON DonHang
FOR DELETE
AS
BEGIN
    DELETE from SP_DonHang
    where IDDonHang = (Select IDDonHang from Deleted)
    Print N'Xóa thành công'
END
```

Sửa đổi Trigger

- Cú pháp:

```
ALTER TRIGGER Tên_trigger
ON Tên_bảng_hoặc_tên_view
[WITH ENCRYPTION]
{FOR | AFTER | INSTEAD OF}
{[DELETE] [,] [INSERT] [,] [UPDATE]}
AS Câu_lệnh_SQL
```

Xóa Trigger

- Cú pháp:

  DROP TRIGGER Tên_trigger

INSTEAD OF Trigger

- INSTEAD OF trigger thường dùng cho View nhằm:
  - Cập nhật nhiều bảng một lúc trong một khung nhìn
  - Tăng điều kiện ràng buộc trên các thuộc tính so với CHECK
  - Đánh giá trạng thái của bảng trước hoặc sau khi cập nhật dữ liệu và thực thi một số nhiệm vụ như in thông báo lỗi, sửa đổi bảng khác
  - Cho phép một phần tập hợp câu lệnh bị từ chối trong khi các phần còn lại vẫn được thực thi thành công
- INSTEAD OF Trigger cũng sử dụng các bảng logic Inserted, Deleted để lưu những thay đổi về dữ liệu khi Trigger đang được thực thi

INSTEAD OF Trigger: ví dụ

- Có View NV_DV lấy thông tin tà 2 bảng NhanVien và DonVi

```
Create view NV_DV
AS
Select
    MasoNV,Hodem,Ten,ngaysinh,gioitinh,diachi,nv.MasoDV,Te
    nDV,MasoNQL
From NhanVien nv, DonVi dv
Where nv.MasoDV=dv.MasoDV
```

INSTEAD OF Trigger: ví dụ

- Tạo trigger trên View NV_DV dùng để chèn dữ liệu vào các bảng tương ứng khi chèn một bản ghi vào view

```
Create Trigger chen_NVDV
On NV_DV  Instead of Insert
As
Begin
    Insert into NhanVien
    (MasoNV,Hodem,Ten,ngaysinh,gioitinh,diachi,MasoDV)
    Select MasoNV,Hodem,Ten,ngaysinh,gioitinh,diachi,MasoDV
    From inserted;
    Insert into DonVi(MasoDV,TenDV,MasoNQL)
    Select MasoDV,TenDV,MasoNQL
    From inserted;
End
```

Bài tập

Các bài sau làm việc với CSDL QuanLyKhachHang

- Bài 1: Viết Trigger thực hiện cập nhật lại ThanhTien của bảng SP_DonHang và TongTien của bảng DonHang mỗi khi trường DonGia trong bảng SanPham được cập nhật
- Bài 2: Viết Trigger thực hiện đảm bảo khi chèn một bản ghi vào bảng SP_DonHang thì IDKhachHang phải tồn tại trong bảng KhachHang và IDSanPham phải tồn tại trong bảng SanPham
- Bài 3: Viết Trigger thực hiện xóa các bản ghi tương ứng trong bảng DonHang và SP_DonHang khi xóa một khách hàng khỏi bảng KhachHang