# Ràng buộc

# Khái quát về View

- View là 1 bảng ảo (virtual table) cho phép truy xuất vào 1 tập con các cột từ 1 hay nhiều bảng
- View bảo đảm được tính bảo mật cho dữ liệu nhờ vào việc hạn chế truy xuất (restricting access), chỉ cho phép hiển thị:
  - 1 số hàng của bảng
  - 1 số cột của bảng
  - Chỉ 1 số hàng và cột của bảng
  - Một tập con của 1 view khác hay 1 tập con của 1 vài view hay bảng
  - Số liệu thống kê trong 1 bảng được cho.

# Khái quát về View

- Khi 1 view đã được định nghĩa, thì nó có thể được dùng như bất kỳ bảng nào khác trong cơ sở dữ liệu.
- Mặc dù view tương tự như bảng, nhưng nó không được lưu trữ thực sự trong cơ sở dữ liệu. Nó chỉ suy dẫn các giá trị từ những bảng dữ liệu gốc.

# Khái quát về View

- View có nhiều ưu điểm :
  - Cung cấp dữ liệu thích hợp cho người dùng
  - Che giấu sự phức tạp của dữ liệu
  - Tổ chức dữ liệu từ các nguồn không đồng nhất
- View được dùng với 2 lý do cơ bản sau:
  - Làm đơn giản hoá các truy vấn phức tạp đối với người dùng
  - Để hạn chế người dùng không được xem dữ liệu trực tiếp từ các bảng

# Lệnh CREATE VIEW

- **CREATE VIEW**
  **[ < *database_name* >.] [ < *owner* > . ] *view_name***
  **[ ( *column* [ ,...*n* ] ) ]**
  **AS**
  ***select_statement***
  **[ WITH CHECK OPTION ]**
- *Column: là tên cột được dùng trong view. Chỉ nên đặt tên cột cho view khi:*
  - Cột được suy dẫn từ 1 biểu thức số học (arithmetic expression) hay 1 hằng số (constant),
  - Khi 2 hay nhiều cột có thể có tên trùng nhau
- Nếu không đặt tên cho cột của view thì nó sẽ có cùng tên với các cột trong mệnh đề SELECT.

# Lệnh CREATE VIEW

- [ WITH CHECK OPTION ]: bắt buộc tất cả các kênh chỉnh sửa dữ liệu liên quan đến view đều phải tuân theo điều kiện lọc trong mệnh đề select. Khi 1 hàng bị sửa đổi thông qua view, tuỳ chọn WITH CHECK OPTION này bảo đảm là dữ liệu vẫn còn nhìn thấy được thông qua view sau khi việc sửa đổi này được thực hiện.

# Lệnh CREATE VIEW

- Ví dụ:
  - View được tạo ra để khôi phục toàn bộ các hàng từ 1 bảng với điều kiên lọc là lương nhân viên < $30,000.
  - Nếu lương của nhân viên tăng thành $32,000, thì khi truy vấn từ view sẽ không còn được nhìn thấy nhân viên này nữa
  - Nếu mệnh đề WITH CHECK OPTION được dùng khi định nghĩa view, thì các hàng sẽ không thể bị chỉnh sửa theo cách mà làm cho chúng bị biến mất khỏi view. Bất kỳ lệnh sửa đổi nào mà gây ra vấn đề này sẽ bị loại trừ và hiển thị thông báo lỗi.

# Ví dụ

```
CREATE VIEW CAonly
AS SELECT au_lname, au_fname, city, state
FROM authors
WHERE state = 'CA' WITH CHECK OPTION
```

```
UPDATE CAOnly
Set state = 'KS' WHERE state ='CA' and
  au_fname ='Ann'
```

Lệnh update này có thực hiện được không?

# Sửa đổi, xoá và đặt tên lại view

- Sửa đổi View

```
ALTER VIEW view_name
 [(column_name)]
AS select_statement
  [WITH CHECK OPTION]
```

- Xoá View

```
DROP VIEW view_name
```

- Đặt lại tên View

```
sp_rename old_viewname, new_viewname
```

# Chỉnh sửa dữ liệu thông qua view

- **Dữ liệu trong bảng gốc (base table) có thể được sửa đổi thông qua việc sửa đổi dữ liệu trong view**
- **Có 1 số hạn chế khi sửa đổi dữ liệu thông qua view:**
  - Không thể sửa đổi dữ liệu trong 1 view nếu việc sửa đổi này cùng 1 lúc ảnh hưởng đến nhiều hơn 1 bảng gốc.
  - Không thể thay đổi một cột mà nó là kết quả của một biểu thức tính toán

# VIEW

1. **Tạo view V1 danh sách các sinh viên khoa "Công nghệ Thông tin"**
2. **Tạo view V2 danh sách sinh viên khoá 2002-2006**
3. **Tạo view V3 danh sách các môn mà chưa sinh viên nào theo học**
4. **Tạo view V4 danh sách sinh viên thi đậu môn Toán Cao cấp A1**
5. **Tìm sinh viên có điểm thi Toán Cao Cấp A1 cao nhất từ view V4**
6. **Đổi tên view V2 thành VSV02**

# Ràng buộc Check

Kiểm tra miền giá trị :

ALTER TABLE Tên_bảng

ADD [ CONSTRAINT CK_Tên_bảng_Tên_cột ]

CHECK (Biểu_thức_luận_lý)

Biểu thức với các toán tử số học, toán tử quan hệ hay từ khoá IN, LIKE, BETWEEN.

# Ràng buộc Check

**Ví dụ 1:** CHECK (cContractRecuiterCode LIKE '[0-9][0-9][0-9][0-9]')

**Ví dụ 2:** CHECK (mTotalPaid BETWEEN 0 AND 50000)

**Ví dụ 3:** CONSTRAINT chkCity CHECK(cCity IN ('Berkeley', 'Boston', 'Chicago', ' Dallas‘))

# Constraint

Hủy một Constraint :

ALTER TABLE Tên_bảng

DROP CONSTRAINT Tên_constraint [ , ...]

Tắt các Constraint :

ALTER TABLE Tên_bảng

NOCHECK CONSTRAINT ALL| Tên_constraint [ , ...]

Bật các Constraint :

ALTER TABLE Tên_bảng

CHECK CONSTRAINT ALL| Tên_constraint [ , ...]

**CHECK constraint**

7. ChuongTrinh.ma chỉ có thể là ‘CQ’ hoặc ‘CD’ hoặc ‘TC’
8. Chỉ có 2 học kỳ là ‘HK1’ và ‘HK2’
9. Số tiết lý thuyết (GiangKhoa.soTietLyThuyet) tối đa là 120

# Rules

- **Rule cung cấp 1 cơ chế để thực thi bảo toàn domain (domain integrity) các cột hay kiểu dữ liệu người dùng (user-defined datatypes)**
- ***Các rule được dùng để:***
  - ***Thực thi việc ràng buộc mà không làm thay đổi cấu trúc bảng.***
  - ***Nếu bảo toàn người dùng áp dụng cho nhiều cột của cùng 1 hay nhiều bảng, nên tạo các rule. Ngược lại nếu bảo toàn chỉ áp dụng cho 1 cột của bảng, nên dùng ràng buộc constraint.***

# Rules

- Cú pháp lệnh CREATE RULE

```
CREATE RULE [owner.]rulename AS
        condition_expression
```

- Biến (variable) được xác định trong biểu thức điều kiện (condition_expression) phải được đặt trước biểu tượng @.
- Ví dụ:

1. CREATE RULE dept_name_rule

AS @deptname NOT IN ('accounts','stores')

2. CREATE RULE min_price_rule

AS @minprice >= $5000

# Rules

- Để xem thông tin của 1 rule:

  *sp_help <rule name>*

- Để xem nội dung của 1 rule:

  *sp_helptext <rule* name>

- Để huỷ 1 rule

  DROP RULE { *rule* } [ ,...*n* ]

# Gắn rule vào 1 cột hay kiểu dữ liệu người dùng

```
sp_bindrule [ @rulename = ] 'rule' ,
    [ @objname = ] 'object_name'
  [ , [ @futureonly = ] 'futureonly_flag' ]
```

- **@Object_name:** là cột của bảng hay kiểu dữ liệu người dùng mà rule được gắn vào. Nếu object_name không ở dạng table.column, thì nó được xem là kiểu dữ liệu của người dùng. Mặc định các cột hiện tại của kiểu dữ liệu người dùng thừa kế rule.
- **@futureonly**: được dùng chỉ khi gắn kết 1 rule vào kiểu dữ liệu người dùng. Tham số này dùng để tránh các cột hiện có của kiểu dữ liệu người dùng không kế thừa rule mới này.

# Ví dụ 1

- Gắn kết rule today vào cột hire_date của bảng employee

  **sp_bindrule 'today', 'employees.[hire date]'**

- Gắn kết rule rule_ssn vào kiểu dữ liệu người dùng ssn

  sp_bindrule 'rule_ssn', 'ssn'

# Hạn chế khi dùng Rules

- Mỗi lần chỉ có 1 rule được gắn kết vào 1 cột hay 1 kiểu dữ liệu của người dùng.
- Nếu 1 rule được gắn kết vào 1 kiểu dữ liệu người dùng, nó không thay thế được rule đã được gắn kết trước đó vào các cột của kiểu dữ liệu người dùng.
- Nếu 1 rule mới được gắn kết vào 1 cột hay kiểu dữ liệu của người dùng đang gắn kết vào 1 rule khác, thì rule mới này sẽ thay thế rule cũ.
- Các rule không áp dụng được cho dữ liệu đã có sẵn trong bảng. Các dữ liệu có sẵn không phải tuân theo điều kiện được xác định bởi rule.

# Xóa gắn kết rule khỏi 1 cột hay kiểu dữ liệu người dùng

- **Cú pháp:**

```
sp_unbindrule [@objname =] 'object_name'
    [, [@futureonly =] 'futureonly_flag']
```

- **Ví dụ:**
  - Xóa rule khỏi cột hire_date của bảng employee

```
sp_unbindrule 'employees.[hire_date]'
```

  - Xoá rule khỏi kiểu dữ liệu người dùng

```
sp_unbindrule ssn
```

# Các ví dụ

- **CREATE RULE typBook AS @list IN ('business', 'mod_cook', 'popular_comp', 'psychology')**
- **sp_bindrule 'typBook', 'Titles.Type'**

➔ Gắn kết rule typBook vào cột Type của bảng Titles.

- **sp_unbindrule 'Titles.Type'**

➔ Xóa các rule đã gắn kết vào cột Type vào bảng Titles.

- **Sp_addType BookType, 'char(20)','NOT NULL'**

- **Sp_bindrule 'typBook', BookType**

➔ Gắn kết rule typBook vào kiểu dữ liệu người dùng

- **sp_unbindrule 'BookType'**

➔ Xoá gắn kết rule đã gắn kết vào kiểu dữ liệu của người dùng

**Ràng buộc RULE**

10. Số tiết thực hành (GiangKhoa.soTietThucHanh) tối đa là 120

11. Số tín chỉ (GiangKhoa.soTinChi) của một môn học tối đa là 6

12. Điểm thi (KetQua.diem) được chấm theo thang điểm 10, chính xác đến 1 chữ số sau dấu phẩy)