QUẢN TRỊ CSDL

Giảng viên: Nguyễn Kim Sao
saonkoliver@gmail.com
0905.883.993

Nội dung

- Ngôn ngữ SQL
- Quản trị CSDL SQL Server 2008

Ngôn ngữ SQL

SQL là gì?

- Structured Query Language (SQL) là một ngôn ngữ rất phổ dụng trong lĩnh vực CSDL.
- Microsoft xây dựng Transact-SQL dựa trên ngôn ngữ vấn tin có cấu trúc chuẩn.
- Nó cung cấp một ngôn ngữ bao hàm toàn diện để định nghĩa bảng, chèn, xóa, thay đổi và truy cập dữ liệu trong bảng.
- Transact-SQL là một ngôn ngữ mạnh, nó hỗ trợ các tính năng khác như: kiểu dữ liệu, đối tượng tạm thời, thủ tục lưu trữ và thủ tục hệ thống.
- Nó còn cho phép chúng ta định nghĩa đối tượng con trỏ, khai báo biến, cấu trúc rẽ nhánh, vòng lặp, bẫy lỗi và Transaction.

Giới thiệu về SQL

- Được xem là một yếu tố chính đóng góp vào sự thành công của CSDLQH khi áp dụng vào đời sống xã hội.
- Là ngôn ngữ mức cao, người dùng chỉ cần xác định kết quả của truy vấn là gì, phần còn lại là tính toán và tối ưu hoá câu lệnh được HQTCSDL đảm nhiệm.
- 1970: SQL (Structured Query Language) bắt nguồn từ ngôn ngữ SEQUEL (Structured English QUEry Language), ngôn ngữ được thiết kế tại tập đoàn IBM nhằm đưa ra một giao diện truy vấn dữ liệu với HQTCSDL thử nghiệm SYSTEM R

Giới thiệu về SQL

- Chuẩn SQL đầu tiên có tên gọi SQL-86 (SQL1) là sự kết hợp giữa hai tổ chức ANSI và ISO.
- Sau đó, SQL1 được mở rộng với nhiều tính năng mới và được gọi là SQL-2 (SQL-92).
- Trong tương lai gần chuẩn SQL3 sẽ ra đời với các tính năng mới như hỗ trợ CSDL hướng đối tượng,…
- SQL có thể hoạt động độc lập hoặc được nhúng vào các ngôn ngữ chủ như C/C++, Pascal theo nhu cầu của người phát triển ứng dụng.

Lược đồ (schema)

- SQL1 không có lược đồ, tất cả các bảng được xem là của một lược đồ duy nhất
- Khái niệm lược đồ được đưa vào chuẩn SQL2 để nhóm các bảng, các cấu trúc thuộc về một ứng dụng nào đó.
- Một lược đồ (SQL Schema) được xác định bởi tên của nó, và bao gồm một định danh xác thực cho biết người dùng nào sở hữu lược đồ cũng như mô tả của các thành phần trong lược đồ
- Các thành phần của lược đồ bao gồm: bảng, ràng buộc, khung nhìn, vùng (domain) và các cấu trúc mô tả lược đồ
- 1 schema <=> 1 database

Khung nhìn (view)

- Là một bảng tạm thời gồm 1 số thuộc tính từ nhiều bảng khác nhau
- Ý nghĩa:
  - Truy cập đến dữ liệu dễ dàng
  - An toàn dữ liệu

NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU

- Data Definition Language (DDL), được dùng để định nghĩa (xây dựng), thay đổi hoặc xóa cấu trúc của các đối tượng CSDL, chẳng hạn: bảng, view, trigger, thủ tục lưu trữ, …
- Câu lệnh DDL:
  - CREATE object_name
  - ALTER object_name
  - DROP object_name

  Với object_name: TABLE, VIEW, …

Định nghĩa lược đồ

- Tạo lược đồ

  CREATE SCHEMA Tên_lược_đồ
  [AUTHORIZATION Tên_tác_giả]
  [DEFAULT CHARACTER SET Tập_ký_tự]
  [Danh sách các phần tử];
- Xóa lược đồ

  DROP SCHEMA Tên_lược_đồ [CASCADE | RESTRICT];

  Cascade: xóa lược đồ + các phần tử của nó

  Restrict: lược đồ được xóa khi không có đối tượng nào trong đó

Định nghĩa database

- Tạo database

CREATE DATABASE Tên_CSDL;

Ví dụ: Create Database Quanlyphongmay;

- Xóa database

DROP DATABASE Tên_CSDL;

Ví dụ: Drop Database Quanlyphongmay;

Định nghĩa bảng

- Tạo bảng:

CREATE TABLE Tên_bảng (
Tên_thuộc_tính kiểu_dl [ràng_buộc **DEFAULT** giá trị],
Tên_thuộc_tính kiểu_dl [ràng_buộc **DEFAULT** giá trị], ..
[**PRIMARY KEY** (DS_thuộc_tính), **UNIQUE** (DS_thuộc_tính),
FOREIGN KEY (Thuộc_tính) **REFERENCES**
Tên_bảng(Thuộc_tính) **ON DELETE** SET NULL | SET DEFAULT **ON UPDATE** CASCADE]);

- **IDENTITY**
  - CREATE TABLE <tên bảng> (tên_cột kiểu_dl [IDENTITY [(Giá_trị_đầu, tham_số_tăng)]] NOT NULL)

Kiểu dữ liệu

- Characters:
  - CHAR(20) -- fixed length
  - VARCHAR(40) -- variable length
- Numbers:
  - BIGINT, INT, SMALLINT, TINYINT
  - REAL, FLOAT -- differ in precision
  - MONEY
- Times and dates:
  - DATETIME
  - TIMESTAMP
- Binary objects (such as pictures, sound, etc.)
  - BLOB -- stands for “Binary Large OBject”
- Other…

Thuộc tính cột

- Keys:
  - PRIMARY KEY – Khóa chính
  - FOREIGN KEY – Khóa ngoại (FOREIGN KEY (Thuộc_tính)
    REFERENCES Tên_bảng(Thuộc_tính)])
  - INDEX – trường được đánh chỉ số
- Null values:
  - NOT NULL – trường bắt buộc nhập giá trị
- Default value:
  - DEFAULT ‘*value’* – giá trị được sử dụng khi người dùng không nhập dữ liệu
- UNIQUE (DS_thuộc_tính): thuộc tính có giá trị là duy nhất

Ví dụ

CREATE TABLE DEPARTMENT
(DNAME VARCHAR(15) **NOT NULL**,
DNUMBER INT **NOT NULL**,
MGRSSN CHAR(9) **NOT NULL**,
MGRSTARTDATE DATE,
PRIMARY KEY (DNUMBER),
UNIQUE (DNAME),
FOREIGN KEY (MGRSSN) **REFERENCES** EMPLOYEE(SSN));

Chỉnh sửa bảng

- ALTER TABLE: chỉnh sửa bảng sau khi đã định nghĩa bằng Create Table

  Cú pháp:

  ALTER TABLE <Table_Name>
  ALTER COLUMN [<Column_name> <New_data_type>]
  | ADD [<Column_name> <Data_Type>]
  | DROP COLUMN [<Column _Name> CASCADE|RESTRICT]

Ví dụ

- Xoá trường ADDRESS khỏi bảng EMPLOYEE:

  ALTER TABLE COMPANY**.**EMPLOYEE **DROP** ADDRESS **CASCADE**;

- Có thể sửa lại giá trị mặc định cho trường MGRSSN trong bảng DEPARTMENT như sau:

ALTER TABLE COMPANY**.**DEPARTMENT **ALTER** MGRSSN **SET DEFAULT** "333445555";

- Có thể sửa đổi các ràng buộc bằng cách xoá bỏ hoặc thêm mới chúng. Để xoá, một ràng buộc cần phải tồn tại và có một tên xác định. Ví dụ, để xoá ràng buộc có tên gọi là EMPSUPERFK trong bảng EMPLOYEE, chúng ta thực hiện:

  ALTER TABLE COMPANY**.**EMPLOYEE **DROP CONSTRAINT** EMPSUPERFK **CASCADE**;

- Có thể thêm mới một ràng buộc bằng từ khoá ADD. Ràng buộc mới thêm vào có thể có tên xác định hoặc không tên

Xóa CSDL và bảng

- DROP DATABASE/TABLE

  DROP DATABASE/TABLE

  <Database_Name/Table_Name>

Định nghĩa miền giá trị

- Tạo miền:

  CREATE DOMAIN Tên_miền AS Mô_Tả;

  – Ví dụ:

  CREATE DOMAIN MyChar AS CHAR(9)

  – Sau câu lệnh này thì có thể dùng kiểu MyChar thay thế cho CHAR(9).

  CREATE DOMAIN ratingval INTEGER DEFAULT 1 CHECK (VALUE>=1 AND VALUE<=10)

- Xóa miền:

  DROP DOMAIN Tên_miền;

Định nghĩa VIEW

– View là bảng không chứa dữ liệu, nó chỉ là truy vấn kết hợp dữ liệu từ 1 hay nhiều bảng có quan hệ với nhau và được lưu thành một đối tượng của SQL SV

– NSD có thể áp dụng ngôn ngữ thao tác dữ liệu trên các View giống như Table.

Định nghĩa view

- Tạo view:

  CREATE VIEW *Tên_khung_nhìn* [trường 1, trường 2, …] AS
  select_statement
  [WITH CHECK OPTION];
- Xóa view:

  DROP VIEW Tên_khung_nhìn;

Ví dụ 1

- Tạo view:

  CREATE VIEW SV_Tinh
  AS SELECT Hoten, Tentinh
  FROM SV, Tinh
  WHERE SV.Matinh = Tinh.Matinh
  With Check Option;
  INSERT INTO SV_Tinh(Hoten, Tinh) VALUES (“Lê Nguyễn Hà”, “Hà nội”)
- Xóa view: DROP VIEW SV_Tinh;

Ví dụ 2

- Tạo View nv_tre (nhân viên dưới 35 tuổi)

```
CREAT VIEW nv_tre (Manv, Hoten, Tuoi)
AS
   SELECT Manv, Hoten, Year(Getdate()) – Year(Ngaysinh)
   FROM    NHANVIEN
   WHERE Year(Getdate()) – Year(Ngaysinh) <= 35
```

* Sử dụng View:

```
SELECT * FROM  NV_TRE
```

☞ Nếu một thuộc tính trong View được xây dựng từ một biểu thức thì bắt buộc phải đặt tên cho thuộc tính đó.

Mục đích dùng VIEW

- Hạn chế tính phức tạp của dữ liệu đối với NSD đơn giản.
- Tạo ra bảng ảo có dữ liệu theo yêu cầu cho NSD và sử dụng trong thiết kế báo cáo.
- Hạn chế quyền truy cập dữ liệu của NSD.
- View dùng để trình bày các thông tin dẫn xuất.

Cập nhật dữ liệu thông qua View

- View định nghĩa dữ liệu trên một bảng thì có thể dùng Insert
- Nếu trong định nghĩa View có chứa mệnh đề Inner join thì không thể dùng các thao tác Insert hay Delete để thay đổi dữ liệu
- Nếu trong định nghĩa View có chứa mệnh đề With check option thì chỉ những bản ghi thỏa mãn điều kiện của View mới được Insert, Update

Xây dựng View dựa trên View khác

- Khi xóa 1 view, mọi view được xây dựng dựa trên view đó cũng bị xóa.
- Có thể dùng thủ tục sp_helptext để xem định nghĩa View
- Tạo View bằng EM
- Mã hoá View: dùng WITH ENCRYPTION
  - Không thể xem được nội dung View
  - Không thể thay đổi lại được

NGÔN NGỮ THAO TÁC DỮ LIỆU

- **SELECT:** truy vấn đến các bản ghi

  SELECT <Column_name(s)> FROM <Table_name> [WHERE <condition>][ORDER BY Column_name [<ASC>][<DESC>]]

  – [GROUP BY]: Nhóm các giá trị để áp dụng tính tổng, giá trị trung bình, đếm… (AVG(), COUNT(), MAX(), MIN(), SUM(), COUNT DISTINCT)

  – [HAVING]: cho phép tìm kiếm kết quả trên nhóm (kèm GROUP BY)

  – [[NOT] IN (SELECT…)], [[NOT] EXISTS (SELECT…)]

  – [COMPUTE function_name (column_name) [,…, function_name (column_name)][BY column_list]

- Truy vấn nhiều bảng: Union, Joins (INNER, OUTER, LEGACY JOIN,…), SubQueries (truy vấn lồng nhau)

Ngôn ngữ thao tác dữ liệu (tt)

- **INSERT:** chèn một bản ghi

  INSERT INTO <Table_name>VALUES <Values>

- **UPDATE:** chỉnh sửa dữ liệu

  UPDATE <Table_name> SET <Column_Name = Value> [WHERE <Search condition>]

- **DELETE:** xóa bản ghi

  DELETE FROM <Table_name> [WHERE <Search condition>]

Bảng tạm

- Khi nào dùng bảng tạm?
- Bảng tạm được lưu trữ tại đâu?
- Loại bảng tạm: cục bộ (#), toàn bộ (##)
- Cú pháp

```
CREATE TABLE #temp
(
……
)
INSERT #temp
SELECT *
FROM SourceTable
```

29

NGÔN NGỮ ĐIỀU KHIỂN DỮ LIỆU – DCL

- Ngôn ngữ điều khiển dữ liệu dùng để thiết lập quyền truy cập trên các đối tượng cơ sở dữ liệu
- Ngôn ngữ điều khiển dữ liệu được sử dụng để bảo mật cơ sở dữ liệu
- Các quyền được điều khiển bằng cách sử dụng các câu lệnh GRANT, REVOKE và DENY

Quyền người dùng

- Sự phân chia khả năng quản trị và sử dụng hệ quản trị cơ sở dữ liệu SQL Server.
- Hình thành theo cơ cấu
  - Người đăng nhập (login)
  - Người dùng (user)
  - Quyền hạn (permission)
  - Nhóm quyền (role)
- Người đăng nhập được thể hiện là mỗi một người dùng với một số quyền hạn ứng với một dữ liệu.

Đăng nhập (login)

- Tạo người đăng nhập

```
exec      sp_addlogin
          'tên_login' ,  'mật_khẩu'  [, 'dữ_liệu']

create  login   tên_login
    with          password = 'mật_khẩu'
                  [, default_database = dữ_liệu ]
```

Đăng nhập (login)

- Xóa người đăng nhập

  exec sp_droplogin 'tên_login'

  drop login tên_login
- Thay đổi người đăng nhập

  alter login { [enable | disable]
  | with { password = '…'
  | default_database = … }
  }

33

Người dùng (user)

- Tạo người dùng cho từng dữ liệu

  exec sp_adduser 'tên_login' , 'tên_user'
  [, 'tên_role']

  create user tên_user for login tên_login
- Xóa người dùng

  exec sp_dropuser 'tên_user'

  drop user tên_user

34

Nhóm quyền (role)

- Tạo nhóm quyền

  exec sp_addrole 'tên_role'

  create role tên_user
- Xóa nhóm quyền

  exec sp_droprole 'tên_role'

  drop role tên_role

Người dùng & nhóm quyền

- Gắn người dùng với nhóm quyền

  exec sp_addrolemember 'tên_role' , 'tên_user'
- Xóa nhóm quyền

  exec sp_droprolemember 'tên_role' , 'tên_user'

Server:
sysadmin
bulkadmin
dbcreator
diskadmin
processadmin
securityadmin
serveradmin
setupadmin

Database:
db_owner
db_securityadmin
db_accessadmin
db_backupoperator
db_ddladmin
db_datawriter
db_datareader
db_denydatawriter
db_denydatareader

Ngôn ngữ điều khiển dữ liệu (tt)

- Câu lệnh GRANT
  - Đặc quyền Grant được sử dụng khi cơ sở dữ liệu được chia sẻ với các người dùng khác.

  Cú pháp:

  GRANT {ALL | statement[,...]} ON Table_Name TO Security_Account [,...]

  [WITH GRANT OPTION]
- Ví dụ: gán quyền SELECT cho người dùng JOHN trên bảng Employee

  GRANT SELECT ON Employee TO JOHN

Ngôn ngữ điều khiển dữ liệu (tt)

- Câu lệnh REVOKE
  - Lệnh REVOKE dùng để xóa các quyền đã gán trên các đối tượng của người dùng trong cơ sở dữ liệu hiện hành

  Cú pháp:

  REVOKE {ALL | statement[,...]} ON Table_Name

  FROM Security_Account [,...]
- Ví dụ: Câu lệnh trên xóa quyền SELECT của người dùng JOHN đối với bảng Employee

  REVOKE SELECT ON Employee FROM JOHN

Ngôn ngữ điều khiển dữ liệu – DCL (tt)

- Câu lệnh DENY
  - Lệnh DENY dùng để ngăn quyền của người dùng

    DENY {ALL | statement[,...]} ON Table_Name TO Security_Account [,...]
  - Ví dụ: Câu lệnh ngăn quyền SELECT trên bảng Employee của người dùng JOHN

    DENY SELECT ON Employee FROM JOHN

Lập trình trong SQL Server

Biến trong SQL

- Biến là vùng nhớ trong bộ nhớ được đặt tên để chứa giá trị dữ liệu.
- Biến có thể phân thành 2 loại: Biến cục bộ và biến toàn bộ. Dữ liệu có thể truyền cho câu lệnh SQL bằng biến cục bộ.
- Local Variables (Biến cục bộ): Trong Transact-SQL, biến cục bộ được khai báo và sử dụng tạm thời khi thực thi câu lệnh SQL.

  Cú pháp:

  DECLARE
  {
  @local_variable_name [AS] data_type
  }

Biến trong SQL (tt)

- Gán giá trị cho biến: dùng SET hoặc SELECT

  SET @local_variable = value

  Hoặc

  SELECT @local_variable = value
- Xem giá trị hiện hành của biến

  PRINT @biến
- Đổi kiểu dữ liệu
  - CAST (@biến AS kiểu dữ liệu)
  - CONVERT: chuyển đổi dạng ngày tháng

Biến trong SQL (tt)

- Global Variables (Biến toàn cục):
  - Biến toàn cục là biến có sẵn và hệ thống quản lý, được đặt tên bắt đầu bởi hai ký hiệu @.
  - Ví dụ:
    - SELECT @@VERSION AS 'SQL Server version‘
  - Một số biến thường dung
    - @@RowCount: tổng số bản ghi
    - @@Error: số mã lỗi của câu lệnh gần nhất
    - @@Fetch_Status: trạng thái việc đọc dữ liệu theo từng bản ghi (cursor)

Kiểu dữ liệu

Data Types in SQL Server			
tinyint	real	datetime	varbinary
smallint	char	smalldatetime	uniqueidentifier
int	nchar	image	numeric
bigint	varchar	money	timestamp
bit	nvarchar	smallmoney	sql_variant
decimal	text	xml	table
float	ntext	cursor	binary

Kiểu dữ liệu (tt)

Kiểu dữ liệu	Kích thước	Miền giá trị dữ liệu lưu trữ
> Các kiểu dữ liệu dạng số nguyên		
Int	4 bytes	từ-2,147,483,648đến +2,147,483,647
SmallInt	2 bytes	từ -32768 đến +32767
TinyInt	1 byte	từ 0 đến 255
Bit	1 byte	0, 1 hoặc Null
> Các kiểu dữ liệu dạng số thập phân		
Decimal, Numeric	17bytes	từ -10^38 đến +10^38
> Các kiểu dữ liệu dạng số thực		
Float	8 bytes	từ -1.79E+308 đến +1.79E+308
Real	4 bytes	từ -3.40E+38 đến +3.40E+38

Kiểu dữ liệu (tt)

> Các kiểu dữ liệu dạng chuỗi có độ dài cố định

Char	N bytes	từ 1 đến 8000 ký tự, mỗi ký tự là một byte

> Các kiểu dữ liệu dạng chuỗi có độ dài biến đổi

VarChar	N bytes	từ 1 đến 8000 ký tự, mỗi ký tự là 1 byte
Text	N bytes	từ 1 đến 2,147,483,647 ký tự, mỗi ký tự là 1 byte

> Các kiểu dữ liệu dạng chuỗi dùng font chữ Unicode

NChar	2*N bytes	từ 1 đến 4000 ký tự, mỗi ký tự là 2 bytes
NVarChar	2*N bytes	từ 1 đến 4000 ký tự, mỗi ký tự là 2 bytes
NText	2*N bytes	từ 1 đến 1,073,741,823 ký tự, mỗi ký tự là 2 bytes

Kiểu dữ liệu (tt)

> Các kiểu dữ liệu dạng tiền tệ

Money	8 bytes	từ -922,337,203,685,477.5808 đến +922,337,203,685,477.5807
SmallMoney	4 bytes	từ -214,748.3648 đến + 214,748.3647

> Các kiểu dữ liệu dạng ngày và giờ

DateTime	8 bytes	từ01/01/1753đến31/12/9999
SmallDateTime	4 bytes	từ01/01/1900đến06/06/2079

> Các kiểu dữ liệu dạng chuỗi nhị phân (Binary String)

Binary	N bytes	từ 1 đến 8000 bytes
VarBinary	N bytes	từ 1 đến 8000 bytes
Image	N bytes	từ 1 đến 2,147,483,647 bytes

Định nghĩa chú thích

- Microsoft SQL Server hỗ trợ hai loại chú thích sau:
  - -- chú thích một dòng
  - /* . . . */ chú thích nhiều dòng

Cấu trúc lệnh

- Cấu trúc lệnh IF

  if (điều_kiện)
  lệnh …. | khối_lệnh
  else
  lệnh …. | khối_lệnh

- Cấu trúc lệnh If Exists

  If exists (Câu lệnh Select)
  Lệnh…| khối_lệnh
  [Else
  Lệnh…| khối_lệnh]

- khối_lệnh := begin
  lệnh … | khối_lệnh
  end

Cấu trúc lệnh (tt)

- Cấu trúc lệnh WHILE

  while (điều_kiện)
  lệnh | khối_lệnh

- Lệnh ngắt vòng lặp

  break
  continue

Cấu trúc lệnh (tt)

- Biểu thức CASE

Case biểu thức
    When giá trị 1 **Then** biểu thức 1
    [**When** giá trị 2 **Then** biểu thức 2]
    …
    [**Else** biểu thức n]
End

Cấu trúc lệnh (tt)

- Ví dụ Tính tổng số chẵn từ 1 -> 100

```
Declare @t int, @x int
Set @t = 0 ; Set @x = 1
While (@x <= 100)
begin
   if  ((@x % 2) = 0)
            set @t = @t + @x
   set @x = @x + 1
end
Print @t
```

Con trỏ

- Khai báo biến:

  Declare Tên_Biến CURSOR

  [phạm vi] [di chuyển][trạng thái][xử lý]

  For câu lệnh Select

  [For update [OF danh sách cột]]

- Trong đó:
  - Câu lệnh select: không chứa các mệnh đề Into, Compute, Compute by
  - Danh sách cột: là danh sách các cột sẽ thay đổi được

53

Con trỏ (tt)

- Phạm vi :
  - Local :chỉ sử dụng trong phạm vi khai báo (mặc định)
  - Global :sử dụng chung cho cả kết nối
- Di chuyển :
  - ForWard_Only :chỉ di chuyển một hướng từ trước ra sau(mặt định)
  - Scroll : di chuyển tùy ý

Con trỏ (tt)

- Trạng thái
  - Static: dữ liệu trên Cursor không thay đổi mặt dù dữ liệu trong bảng nguồn thay đổi(mặt định)
  - Dynamic :dữ liệu trên Cursor sẻ thay đổi mặt dù dữ liệu trong bảng nguồn thay đổi
  - KeySet :giống Dynamic nhưng chỉ thay đổi những dòng bị cập nhật
- Xử lý :
  - Read_Only :chỉ đọc (mặc định)
  - Scroll_Lock : đọc/ghi

Con trỏ (tt)

- Sử dụng

```
open   tên_biến_cursor
       ….
close  tên_biến_cursor
```

- Hủy cursor

```
deallocate  tên_biến_cursor
```

Con trỏ (tt)

- Di chuyển Cursor

  fetch *định_vị*
  from tên_biến_cursor
  into @tên_biến [,… n]

  định_vị := next | prior | last | first |
  absolute (giá_trị | biến)
  relative (giá_trị | biến)

57

Con trỏ (tt)

- NEXT: Di chuyển về sau
- PRIOR : Di chuyển về trước
- FIRST : Di chuyển về đầu
- LAST : Di chuyển về cuối
- ABSOLUTE n: nếu n>0 di chuyển đến bản ghi thứ |n| tính từ bản ghi đầu tiên, nếu n<0 : tính từ bản ghi cuối
- RELATIVE n :di chuyển đến bản ghi thứ n tính từ bản ghi hiện hành

Con trỏ

- Trạng thái Cursor
  @@fetch_status
  - =0 : Đang trong dòng dữ liệu
    (lần đi kế tiếp thành công)
  - ≠0 : Ngoài dòng dữ liệu
    (lần đi kế tiếp không thành công)

59

Con trỏ (tt)

- Chú ý: thứ tự các thao tác khi xử lý dữ liệu trên CurSor

  1. Định nghĩa biến Cursor
  2. Mở Cursor
  3. Duyệt và xử lý dữ liệu trên Cursor
  4. Đóng và giải phóng Cursor

Ví dụ

- In danh sách các sinhvien(masv char(5),tensv char(10))

```
Declare sv cursor for select * from sinhvien
Open sv
Declare @ma char(5),@ten char(10)
Fetch next from sv into @ma,@ten
While (@@fetch_status = 0)
begin
   print @ma + ' : ' + @ten
    Fetch next from sv into @ma,@ten
end
Close sv; Deallocate sv
```

61

Tránh sử dụng cursor như thế nào?

- Tại sao lại tránh sử dụng cursor?
- Tránh sử dụng bằng cách nào?
  - Sử dụng câu lệnh SQL chuẩn (chẳng hạn: Case)
  - Sử dụng lặp (while)
  - Sử dụng bảng tạm

62

Nội thủ tục (Stored procedure)

- Là các chương trình được lưu trữ trong CSDL. Được gọi thi hành khi có yêu cầu.
- Thường được thiết kế để thi hành các luật ràng buộc
- Ích lợi của SP:
  - Giảm thời gian biên dịch
  - Đơn giản trong bảo trì
  - Bảo mật
  - Giảm dung lượng truyền dữ liệu

Nội thủ tục (Stored procedure) (tt)

- Có 2 loại SP:
  - SP hệ thống (system sp)
  - SP người dùng (user sp)
- SP người dùng gồm 3 loại:
  - Trigger
  - User stored procedure
  - Defined function

Nội thủ tục (Stored procedure) (tt)

- SP được xây dựng từ các câu lệnh T-SQL và được lưu trú trên SQL server.
- Muốn thực hiện một SP, NSD chỉ cần thực hiện một lời gọi hàm.
- Khi SP được chạy lần đầu tiên nó sẽ được biên dịch qua 5 bước và sinh ra một mô hình truy vấn. Mô hình này sẽ được đặt trong một CSDL của SQL server, lần sau chạy lại thủ tục sẽ không phải dịch lại nữa.

Nội thủ tục (Stored procedure) (tt)

- Năm bước biên dịch thủ tục:
  - Thủ tục được phân tích ra thành nhiều phần
  - Kiểm tra sự tồn tại của các đối tượng (view, table, …) mà thủ tục tham chiếu tới.
  - Lưu trữ tên thủ tục vào bảng sysobject, lưu trữ các mã lệnh của thủ tục vào bảng syscomments.
  - Sinh ra mô hình truy vấn của thủ tục và lưu vào bảng sysprocedure
  - Khi SP được chạy lần đầu tiên, cây truy vấn sẽ được đọc và được tối ưu thành một kế hoạch thủ tục và chạy → tiết kiệm thời gian tái phân tích, biên dịch cây truy vấn mỗi khi chạy thủ tục.

Nội thủ tục (Stored procedure) (tt)

- Trong một phiên làm việc, nếu SP được thực hiện, nó sẽ được lưu trữ vào vùng nhớ đệm. Những lần sau nếu SP được gọi thực hiện lại thì nó sẽ được đọc trực tiếp ra từ vùng nhớ đệm → nâng cao hiệu suất chạy truy vấn.

Nội thủ tục (Stored procedure) (tt)

- Tạo sp:

```
CREATE PROCEDURE procedurename [parameter1
   datatype [length] [OUTPUT], parameter2...]
AS
BEGIN ... END
```

Trong đó:

+ parameter1, parameter2, .. .: là các tham số
+ datatype: kiểu dữ liệu của tham số
+ output: tham số nhận giá trị trả về
+ trong khối BEGIN … END là các lệnh SQL.

Nội thủ tục (Stored procedure) (tt)

- Thực thi thủ tục

  exec tên_thủ_tục giá_trị | @biến [output] [,…n]
  - EXEC tên_thủ_tục giá_trị_1, giá_trị_2, …
  - EXEC tên_sp @p1 = giá_trị, @p2 = giá_trị, …
- Xóa thủ tục

  Drop procedure tên_thủ_tục
- Thay đổi thủ tục

  Alter procedure tên_thủ_tục

Nội thủ tục (Stored procedure) (tt)

- Ví dụ: sp có tham số

```
CREATE PROCEDURE sp2
    @p1 int, @p2 char(100)
AS
begin
  SELECT * FROM T1
  WHERE (id = @p1) and (name = @p2)
End
```

- Gọi sp2: execute sp2 @p1 = 1, @p2 = ‘aa’

Nội thủ tục (Stored procedure) (tt)

- Ví du Viết thủ tục xóa các sinh viên theo thành phố
  sinhvien (masv char(5), tp char(5))

```
create procedure xoasinhvien
      @tp nchar(50)
as
begin
      delete from T1 where tp = @tp
end

exec  xoasinhvien  'HCM'
```


Nội thủ tục (Stored procedure) (tt)

- **Ví dụ: sp có tham số đầu ra**

```
CREATE PROCEDURE sp3 @p1 int,@p2 char(100) output
AS
begin
   select * into tmp_t1 from T1 where id1 = @p1
   select @p2 = name1 from tmp_t1
   drop table tmp_t1
end

CREATE PROCEDURE sp4
AS
begin
   declare @t1 char(100)
   execute sp3 @p1 = 1, @p2 = @t1 output
   print @t1
end
```

Nội thủ tục (Stored procedure) (tt)

- Ví du Viết thủ tục đếm xem có bao nhiêu sinh viên theo thành phố.

```
create procedure dem @tp char(5), @t int output  as
begin
      select @t = count(*) from sinhvien
      where tp = @tp
end

declare @tong int
exec  dem  'HCM' , @tong output
print @tong
```

73

Nội thủ tục (Stored procedure) (tt)

- **Ví dụ: sp có tham số đầu ra kiểu trỏ**

```
CREATE PROCEDURE sp5 @p cursor varying output
AS
begin
    set @p = cursor forward_only static for
    select * from T1
    open @p
end
CREATE PROCEDURE sp6
AS
begin
    declare @mycursor cursor, @myid int, @myname char(100)
    execute sp5 @p= @mycursor output
    fetch next from @mycursor into @myid, @myname
    while (@@FETCH_STATUS = 0)
    begin
      print convert(varchar(100),@myid) + '     ' + @myname
      fetch next from @mycursor into @myid, @myname
    end
    close @mycursor
    deallocate @mycursor
end
```

Nội thủ tục (Stored procedure) (tt)

- **Ví dụ: sp có giá trị trả về**

```
CREATE PROCEDURE  sp7
AS
begin
   declare @x char(100)
   select @x = 'Hello world'
   return @x
End

declare @v char(100)
exec @v = sp7
print @v
```

Kiểm soát lỗi với TRY … CATCH

- Thực hiện các lệnh trong khối Try, nếu gặp lỗi sẽ chuyển qua xử lý bằng các lệnh trong khối Catch
- Cú pháp:

```
BEGIN TRY
    {các câu lệnh}
END TRY
BEGIN CATCH
    {các câu lệnh}
END CATCH
```

Kiểm soát lỗi cới TRY … CATCH (tt)

- Lưu ý:
  - Try và Catch phải cùng lô xử lý
  - Sau khối Try phải là khối Catch
  - Có thể lồng nhiều cấp

```
BEGIN TRY
    -- Generate some error.
    Declare @str varchar(20);
    Set @str = 'SQL SERVER!';
    print convert(datetime, @str);
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

Tạo một câu lệnh SQL phát sinh ra lỗi

Đọc thông tin về lỗi

Kiểm soát lỗi với TRY … CATCH (tt)

- Một số hàm cung cấp thông tin về lỗi vừa phát sinh:

STT	Tên hàm	Chức năng
1	Error_number()	Trả lại mã lỗi (dưới dạng số)
2	Error_severity()	Trả lại mức độ nghiêm trọng của lỗi
3	Error_state()	Trả lại trạng thái lỗi (dưới dạng số)
4	Error_procedure()	Trả lại tên thủ tục hoặc Trigger phát sinh lỗi
5	Error_line()	Trả lại vị trí dòng lệnh phát sinh lỗi
6	Error_message()	Trả lại thông báo lỗi dưới hình thức văn bản (text)

Kiểm soát lỗi cới TRY … CATCH (tt)

- Ví dụ kiểm soát lỗi TRY...CATCH với thủ tục:

```
ALTER PROCEDURE deleteA
    @A1 int
AS
BEGIN TRY
    DELETE FROM A WHERE A1 = @A1
END TRY
BEGIN CATCH
    PRINT 'ERROR on delete record: ' + CONVERT(varchar,@A1)
    RETURN 1001 -- return user-defined error code
END CATCH
GO
```

Sử dụng TRY ... CATCH để bắt và xử lý lỗi; Hai khối lệnh TRY và CATCH phải nằm liền kề nhau.

79

Hàm (function)

- Tạo lập hàm: Scalar Functions

```
CREATE FUNCTION [ schema_name. ] function_name
    ( [ { @parameter_name data_type [ = default ] [ READONLY ] } [ ,...n ] ] )
    RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
```

80

Hàm (function)…

- Tạo lập hàm: Table-Valued Functions

```
CREATE FUNCTION [ schema_name. ] function_name
    ( [ { @parameter_name parameter_datatype [ = default ]
    [ READONLY ] } [ ,...n ] ]
    RETURNS TABLE
    [ WITH <function_option> [ ,...n] ]
    [ AS ]
    RETURN [ (] select_stmt[ )]
```


Hàm (function)…

```
CREATE FUNCTION [ schema_name. ] function_name
    ( [ { @parameter_name data_type [ = default ] [ READONLY ] } [ ,...n
    ] ] )
RETURNS @table_name TABLE
          (
           columns          data_type
          )
 AS
BEGIN
          function_body
RETURN
 END
```

Hàm (function)

- Thực thi hàm
  - = tên_hàm (giá_trị | @biến [,…n])
  - dùng select tên_hàm hoặc select…from tên_hàm
- Xóa hàm

  Drop function tên_hàm
- Thay đổi hàm

  Alter function tên_hàm

  …….

Hàm (function)

- Ví dụ

  Viết hàm sinh ra mã sinh viên tự động theo quy tắc

  - Mã sinh viên có dạng: BA0001

  'BA' : quy định (luôn có)

  0001 : là số

  VD:

  Hiện tại sinh viên có mã cao nhất là BA0024

  Thì sinh mã mới là BA0025

Hàm (function)

```
Create function sinhkhoa () returns char(6) As
Begin
   declare @max int
   select
        @max = max(cast(substring(masv,3,4) as int)) + 1
        from sinhvien
   declare @s char(8)
   set @s = '000' + rtrim(cast(@max as char(4)))
   set @s = 'BA' + right(@s,4)
   return @s
end
```


Hàm (function)

- Ví dụ với Table Function

```
create function laydssv (@malop char(5))
returns TABLE
as
     return (
          select masv,tensv from sinhvien
          where malop = @malop
          )

select * from laydssv('QT1')
```

Hàm (function)

- Ví dụ với Table Function

```
create function laydssv1 (@malop char(5))
      returns  @btam table(masv char(5),tensv char(20))
as
begin
      insert into @btam  select masv,tensv from sinhvien
            where malop = @malop
      return
end

select * from laydssv1('QT1')
```

87

HÀM CỦA NSD
(USER DEFINED FUNCTIONS-UDFs)

– UDFs giống như SP nhưng khác ở các điểm sau:

UDF	SP
- Giá trị các tham số không được truyền ra ngoài.	- Có thể đưa giá trị của tham số ra ngoài bằng thuộc tính OUTPUT
- Có thể trả về một giá trị vô hướng hoặc một bảng dữ liệu.	- Chỉ trả về kiểu DL giá trị kiểu vô hướng

Bẫy sự kiện (trigger)

- Trigger được phát sinh sau những hành vi thêm mới hay thay đổi, xóa trên bảng.
  - Có thể hủy các cập nhập trên dữ liệu
- Trigger được phát sinh để thay thế những hành vi thêm, đổi, xóa.
- Trigger lưu giữ tách rời giá trị mới được đưa vào và giá trị cũ được xóa bỏ.
  - Dùng bảng tạm **Inserted** và **Deleted**
- Trigger còn áp dụng cho *Login*.

Trigger (tt)

- Là một loại sp đặc biệt gắn với 1 table / view
- Tự động thi hành khi có một sự kiện xảy ra với bảng, view như: insert, update, delete 1 / 1 số các row
- Có 2 loại trigger
  - After trigger
  - Instead of trigger
- Trigger ko trả về giá trị, không có tham số

Trigger (tt)

- Trigger thường dùng trong các trường hợp:
  - Ràng buộc toàn vẹn dữ liệu
  - Kiểm soát dữ liệu hiện tại khi có thay đổi đến giá trị trong mẩu tin của bảng
  - Kiểm tra dữ liệu nhập vào phù hợp với mối liên hệ dữ liệu giữa các bảng với nhau
  - Kiểm chứng khi xóa mẩu tin trong bảng

91

Tạo trigger

- CREATE TRIGGER name ON table | view

[FOR | AFTER | INSTEAD OF]

[INSERT, UPDATE, DELETE]

AS

BEGIN ... END

- Xóa và thay đổi

  Alter | Drop trigger tên_trigger

Trigger (tt)

```
CREATE TRIGGER tg4 ON [dbo].[T1]
FOR UPDATE
AS
begin
   declare @id_del int
   select @id_del = id1 from deleted
   raiserror( 'ban da xoa ban ghi co  id = %d', 16, 1,
   @id_del)
end
```

Trigger (tt)

Sự kiện lồng nhau được tối đa 32

- Thay đổi thông số cho phép lồng nhau

```
alter database tendatabase
        set  recursive_triggers  { on | off }
```

- Thiết lập giới hạn lồng nhau

```
exec sp_configure 'Nested Triggers' n
```

Trigger (tt)

- Tạo trigger cho bảng sinhvien (masv, tensv, malop) thỏa mãn điều kiện một lớp không quá 20 người.

```
Create trigger tssv on sinhvien for insert,update As
Begin
   declare @malop char(5), @ts int
   select @malop = malop from inserted
   select @ts = count(*) from sinhvien
            where malop=@malop
   if (@ts > 20)
            rollback transaction
end
```


Trigger (tt)

- Tạo trigger cho bảng sinhvien (masv, tensv, trangthai) thỏa mãn điều kiện khi xóa một sinh viên tức thay đổi trạng thái từ 0 thành 1.

```
Create trigger tssv on sinhvien instead of delete As
Begin
   update sinhvien set trangthai = 1
   where        masv in
                (select masv from deleted)
end
```

Trigger (tt)

- Giả sử có hai bảng HoaDon(SoHD, NgayHD) và ChiTietHD (SoHD, MaH, SLBan, DonGia). Tạo Trigger cho bảng ChiTietHD khi them vào bản ghi thì SoHD đó đã thêm vào bản HoaDon hay chưa

```
CREATE TRIGGER trgIns
ON ChiTietHD
FOR INSERT
AS
        IF NOT EXISTS (SELECT "True" FROM INSERTED WHERE
                Inserted.SoHD=HoaDon.SoHD)
BEGIN
        RAISERROR (60000, 16,1,'SoHD', 'ChiTietHD', 'SoHD',
                'HoaDon')
        ROLLBACK TRAN
END
```

97

Transaction

- Nhóm nhỏ các phát biểu: hoặc thực thi toàn bộ, hoặc không làm gì cả
- Begin: bắt đầu một tracsaction
  - **Begin tran**saction [<tên transaction|@biến transaction>]
- Commit: xác định kết thúc hay hoàn tất
  - **commit tran**saction [<tên transaction|@biến transaction>]

98

Transaction (tt)

- Rollback: các thao tác bị hủy bỏ từ begin hoặc điểm đánh dấu
  - **Rollback transaction** [<tên transaction| điểm đánh dấu| @biến transaction>]
- Save transaction
  - **Save transaction** [<điểm đánh dấu>]
- Biến @@trancount: cho biết số transaction hiện đang thực hiện (chưa được kết thúc với rollback hay commit) trong connection hiện hành

99

Transaction (tt)

```
SET XACT_ABORT ON
BEGIN TRAN
BEGIN TRY
-- lệnh 1
-- lệnh 2
-- ...
COMMIT
END TRY
BEGIN CATCH
     ROLLBACK
     DECLARE @ErrorMessage VARCHAR(2000)
     SELECT @ErrorMessage = 'Lỗi: '
          +ERROR_MESSAGE()
     RAISERROR(@ErrorMessage, 16, 1)
END CATCH
```

100

Transaction (tt)-ví dụ

```
BEGIN
DECLARE @Account_Id_A integer = 1;
DECLARE @Account_Id_B integer = 2;
DECLARE @Amount float = 10;
 BEGIN TRAN;
  BEGIN TRY
      UPDATE Account SET AVAIL_BALANCE = AVAIL_BALANCE -
       @Amount WHERE Account_Id = @Account_Id_A;
      UPDATE Account SET AVAIL_BALANCE = AVAIL_BALANCE +
       @Amount WHERE Account_Id = @Account_Id_B;
      COMMIT TRAN;
  END TRY
  BEGIN CATCH
       PRINT 'Error: ' + ERROR_MESSAGE();
       ROLLBACK TRAN;
  END CATCH;
END;
```


LOCK

- Shared Locks (khóa chia sẻ): cho phép đọc dữ liệu, không cho phép sự thay đổi nào của tài nguyên
- Exclusive Locks (Khóa độc quyền): ngăn hai người cùng thực hiện đọc, cập nhật, xóa, thêm bản ghi trong cùng một thời gian
- Update Lock: khi chưa cần cập nhật thì ở chế độ Shared Locks. Khi lênh Update thực sự thực thi thì ở chế độ Exclusive Locks
- Intent Locks: Thông báo về việc sẽ khóa dữ liệu

Lock (tt) – Các ví dụ

- **Shared Locks**

```
BEGIN TRAN
USE AdventureWorks

SELECT * FROM Person.Address WITH (HOLDLOCK)
WHERE AddressId = 2

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks
WHERE resource_type <> 'DATABASE'

ROLLBACK
```

103

Lock (tt) – Các ví dụ

- **Update Locks**

```
BEGIN TRAN
USE AdventureWorks

SELECT * FROM Person.Address WITH (UPDLOCK)
WHERE AddressId < 2

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks
WHERE resource_type <> 'DATABASE'

ROLLBACK
```

104

Lock (tt) – Các ví dụ

- **Exclusive locks (X)**

```
BEGIN TRAN
USE AdventureWorks

UPDATE Person.Address SET AddressLine2 = 'Test
Address 2' WHERE AddressId = 5

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks WHERE
resource_type <> 'DATABASE'

ROLLBACK
```

105

Lock (tt) – Các ví dụ

- **Intent locks (I)**

```
BEGIN TRAN
USE AdventureWorks

UPDATE TOP(5) Person.Address SET
AddressLine2 = 'Test Address 2' WHERE
PostalCode = '98011'

SELECT resource_type, request_mode,
resource_description FROM sys.dm_tran_locks
WHERE resource_type <> 'DATABASE'

ROLLBACK
```

106

Lock (tt)- sys.dm_tran_locks

```
SELECT dm_tran_locks.request_session_id,
    dm_tran_locks.resource_database_id,
    DB_NAME(dm_tran_locks.resource_database_id) AS dbname,
    CASE WHEN resource_type = 'OBJECT'
        THEN OBJECT_NAME(dm_tran_locks.resource_associated_entity_id)
        ELSE OBJECT_NAME(partitions.OBJECT_ID)
    END AS ObjectName,
    partitions.index_id, indexes.name AS index_name,
    dm_tran_locks.resource_type,
    dm_tran_locks.resource_description,
    dm_tran_locks.resource_associated_entity_id,
    dm_tran_locks.request_mode,
    dm_tran_locks.request_status
FROM sys.dm_tran_locks
LEFT JOIN sys.partitions ON partitions.hobt_id =
        dm_tran_locks.resource_associated_entity_id
LEFT JOIN sys.indexes ON indexes.OBJECT_ID = partitions.OBJECT_ID
        AND indexes.index_id = partitions.index_id
WHERE resource_associated_entity_id > 0
        AND resource_database_id = DB_ID()
ORDER BY request_session_id, resource_associated_entity_id
```


Thực thi lệnh SQL

- Các câu lệnh có thể được thực thi như là các câu lệnh đơn hoặc như một lô
- Xử lý lô
  - Một lô là một tập hợp của một hoặc nhiều câu lệnh SQL được gửi cùng một thời điểm từ một ứng dụng đến SQL Server để thực thi
  - Các câu lệnh này được biên dịch thành một đơn vị thực thi và được gọi là “execution plan”
  - Các câu lệnh trong “execution plan” được thực hiện cùng một lúc
- Scripts
  - Các câu lệnh SQL có thể được thực thi trong script bằng cách lưu trên tập tin. Phần mở rộng của file thường lưu dưới dạng *.sql. Tập tin sẽ được đọc khi được yêu cầu để thực thi.

Cảm ơn các em đã chú ý lắng nghe

Những em chưa chú ý vẫn được cảm ơn bình thường

109