Chương 5: Ngôn ngữ SQL (Structured Query Language)

Giới thiệu SQL

- SQL là ngôn ngữ tiêu chuẩn của các hệ quản trị cơ sở dữ liệu quan hệ dùng để giao tiếp với cơ sở dữ liệu. Các lệnh trong SQL có thể phân làm 3 loại
  - Ngôn ngữ định nghĩa dữ liệu (Data Definition Language commands - DDL)
  - Ngôn ngữ thao tác dữ liệu (Data Manipulation Language commands -DML)
  - Ngôn ngữ điều khiển dữ liệu (Data Control Language commands -DCL)

Định nghĩa dữ liệu

- **Lệnh định nghĩa dữ liệu trong SQL: CREATE**
- Lệnh CREATE được sử dụng để tạo:
  - Lược đồ (schemas)
  - Bảng (relations)
  - Domains
  - Các cấu trúc khác như: views, assertions và triggers

Lệnh tạo cơ sở dự liệu trong SQL

- **Cú pháp:**

```
CREATE DATABASE database_name
```

- Ví dụ:
  - **CREATE DATABASE** QuanlySach

Lệnh tạo cơ sở dự liệu trong SQL

- Các tùy chọn trong lệnh tạo cơ sở dữ liệu:
  - Logic name
  - Physical name
  - Size: initial size.
  - Maximum size.
  - Growth increment

Lệnh tạo cơ sở dự liệu trong SQL

- Cú pháp đầy đủ:

```
CREATE DATABASE DatabaseName
ON (
      NAME = Logical_Dat_fileName,
      FILENAME = ‘Path\fileName.mdf',
      SIZE = Size,
      MAXSIZE = Maxsize,
      FILEGROWTH = filegrowth_increament)
LOG ON (
      NAME = Logical_Log_fileName,
      FILENAME = 'Path\fileName.ldf',
      SIZE = Size,
      MAXSIZE = Maxsize,
      FILEGROWTH = filegrowth_increament) ;
```

Lệnh tạo cơ sở dự liệu trong SQL

- Ví dụ:

```
CREATE DATABASE  QuanlySach
ON (
   NAME = QuanlySach,
   FILENAME = 'D:\data\QuanlySach.mdf',
   SIZE = 10 MB,
   MAXSIZE = 40 MB,
   FILEGROWTH = 1 MB)
LOG ON (
   NAME = SalesDB_log,
   FILENAME = 'D:\data\QuanlySach.ldf',
   SIZE = 6 MB,
   MAXSIZE = 8 MB,
   FILEGROWTH = 5% )
```

Lệnh tạo bảng (Table)

- Cú pháp:

```
CREATE TABLE table_name
    (<column name> data type)
```

  - Table name: tên bảng
  - Column name: tên các cột trong bảng
  - Data type: kiểu dữ liệu

Lệnh tạo bảng (Table)

- Ví dụ:

```
CREATE TABLE Nhanvien
(
    MaNhanvien char(5),
    HoNhanvien varchar(30),
    TenNhanvien varchar(25),
    Ngaysinh Smalldatetime
)
```

Kiểu dữ liệu trong SQL

- Kiểu dữ liệu cơ bản trong SQL gồm:
  - Numeric,
  - Character string,
  - Bit string,
  - Boolean,
  - Date, and time

Kiểu dữ liệu trong SQL

- **Numeric:**
  - **Int (Integer)** (tập hữu hạn các số nguyên, phụ thuộc vào máy)
  - **Smallint (Small integer):** tập con của loại số nguyên – phụ thuộc vào máy).
  - Một số có thể định dạng:
    - `DECIMAL(i,j)`
    - `DEC(i,j)`
    - `NUMERIC(i,j)`

    *Trong đó: **i** là số ký số, **j** là số ký số bên phải dấu thập phân*

Kiểu dữ liệu trong SQL

- **Real, double precision**: kiểu số thực dấu chấm động, độ chính xác tùy thuộc vào máy
- **float(n)**: số thực dấu chấm động, với độ chính xác của người dùng chỉ định ít nhất n chữ số.

Kiểu dữ liệu trong SQL

- **String:**
  - **char(n):** cố định chiều dài của chuỗi ký tự là n.
  - **varchar(n)**: chiều dài của chuỗi ký tự thay đổi, với số ký tự tối đa là n.
  - **nvarchar(n)**: tương tự varchar, ngoại trừ nó sử dụng Unicode và do đó tăng gấp đôi số lượng không gian cần thiết để lưu trữ các dữ liệu.
  - **Text**: lưu trữ các dữ liệu có chiều dài hơn 8.000 ký tự

Kiểu dữ liệu trong SQL

- **Datetime**:
  - **Datetime**: Kiểu ngày giờ (chính xác đến phần trăm của giây)
  - **Smalldatetime:** Kiểu ngày giờ (chính xác đến phút)

Kiểu dữ liệu trong SQL

TYPE	BASIC TYPE	SIZE	DOMAIN
Binary	Binary	8 KB	"0"…"9", "a".."f", "A".."F"
	Varbinary	8 KB	"0"…"9", "a".."f", "A".."F"
	Image	2^31 -1 bytes	2^31 –1 bytes
Character	Char	255 bytes	1..8000 ký tự
	Varchar	255 bytes	1..8000 ký tự
	Text	2147483647 bytes	2^31-1 ký tự (2147483647)
Date and Time	Datetime	8 bytes	01/01/1753->31/12/9999
	Smalldatetime	4 bytes	1/1/1900 -> 6/6/2079
Decimal	Decimal	17 bytes	-10^38-1 -> 10^38-1
	Numeric	17 bytes	-10^38-1 -> 10^38-1
Foating point	Float	8 bytes	-1.79E+308 -> 1.79E+308
	Real	4 bytes	-3.40E+38 ->3.40E+38
Integer	Bigint	8 bytes	-2^63 -> 2^63

Toàn vẹn dữ liệu (Data integrity)

- **Tính toàn vẹn dữ liệu** để đảm bảo chất lượng của dữ liệu trong cơ sở dữ liệu.
- Trong SQL, có hai cách để đảm bảo tính toàn vẹn dữ liệu:
  - *Toàn vẹn khai báo (Declarative integrity): khai báo các rang buộc khi tạo bảng* (contraint, default, rule)
  - *Toàn vẹn thủ tục: cần tạo các thủ tục để thực hiện các toàn vẹn (Procedural integrity:* trigger, stored procedure…)

Toàn vẹn dữ liệu (Data integrity)

- Các ràng buộc cơ bản có thể được xác định khi tạo bảng, bao gồm:
  - Ràng buộc khóa chính – ràng buộc toàn vẹn thực thể
  - Ràng buộc khóa ngoại – ràng buộc tham chiếu
  - Ràng buộc Default, check, not null

Toàn vẹn thực thể (Entity Integrity)

- Xác định một dòng như là một thực thể duy nhất trong một bảng cụ thể.
  - Tính toàn vẹn thực thể thể hiện bằng cột định danh hoặc **khóa chính của bảng**.
  - **Ràng buộc khóa chính (Primary Key Constraints) trong SQL**

```
CREATE TABLE table_name
(       column_name  data_type NOT NULL
        [CONSTRAINT constraintname] PRIMARY
        KEY
)
```

Toàn vẹn thực thể (Entity Integrity)

- Nếu khóa chính là một tập nhiều cột

```
CREATE TABLE  table_name
(    column_name  data_type[,...]
     [CONSTRAINT constraintname]
     PRIMARY KEY{(column1[ASC|DESC][,...columnN])}
)
```

Toàn vẹn thực thể (Entity Integrity)

- Ví dụ:

```
CREATE TABLE Ketqua
(
        masv char(10) not null,
        mamh varchar(40) not null,
        Diem float  not null,
        Primary key (masv, mamh)
);
```

Toàn vẹn tham chiếu (Referential Integrity)

- Duy trì các mối quan hệ được xác định giữa các bảng khi một record được nhập vào hoặc xóa.
- Toàn vẹn tham chiếu được dựa trên mối quan hệ giữa khóa ngoại và các khóa chính hoặc giữa khóa ngoại và khóa duy nhất (unique keys).

Toàn vẹn tham chiếu (Referential Integrity)

- Ràng buộc tham chiếu nhằm tránh các lỗi khi
  - **Thêm** một record vào một bảng quan hệ nếu không có record liên quan trong bảng chính.
  - **Cập nhật** các giá trị trong một bảng chính tạo các record mồ côi trong một bảng liên quan.
  - **Xóa** các bản ghi từ một bảng chính nếu có record trong bảng quan hệ

Toàn vẹn tham chiếu (Referential Integrity)

- Cú pháp khai báo **Referential Integrity**

```
CREATE TABLE table_name
(       colum_name datatype [,...],
        [CONSTRAINT constraint_name ]
        FOREIGN KEY [ ( column [ ,...n ] ) ]
        REFERENCES ref_table [ ( ref_column [ ,...n ])]
)
```

Toàn vẹn tham chiếu (Referential Integrity)

- Ví dụ:

```
Create table Phongban
(
        Mapb int not null primary key
        Tenpb varchar(30)
)
Create table nhanvien
(
        manv int not null primary key
        Hoten varchar(40),
        Mapb int,
        Contraint mapb_Fk foreign key(mapb) references
        phongban(mapb)
 )
```

Toàn vẹn tham chiếu (Referential Integrity)

- Ràng buộc toàn vẹn tham chiếu có thể bị vi phạm khi có một bộ dữ liệu được chèn hoặc bị xóa, hoặc có sự hiệu chỉnh khóa chính hoặc khóa ngoại.
- Mặc định là SQL từ chối các hoạt động Insert, update và delete.
- Tuy nhiên có thể thực hiện được bằng cách gán các thuộc tính SET NULL, CASCADE và SET DEFAULT vào ràng buộc khóa ngoại

Toàn vẹn tham chiếu (Referential Integrity)

- Thuộc tính của khai báo toàn vẹn tham chiếu
  - RESTRICT, CASCADE, SET NULL or SET DEFAULT

```
CREATE TABLE table_name
(       colum_name datatype [,...],
        [CONSTRAINT constraint_name ]
        FOREIGN KEY [ ( column [ ,...n ] ) ]
        REFERENCES ref_table [ ( ref_column [ ,...n ])]
        ON DELETE SET DEFAULT ON UPDATE CASCADE
)
```

Toàn vẹn tham chiếu (Referential Integrity)

- **CASCADE**: Cho phép xoá hoặc cập nhật các record đang tham chiếu đến record cần xoá hoặc cập nhật.
- **SET NULL**: thiết lập tất cả các giá trị đang tham chiếu đến record cần xoá thành NULL
- **SET DEFAULT**: thiết lập tất cả các giá trị đang tham chiếu đến record cần xoá có giá trị mặc định.
- **RESTRICT**: không được phép xóa hoặc cập nhật những record được tham chiếu.

Toàn vẹn tham chiếu (Referential Integrity)

- Ví dụ:

```
CREATE TABLE Nhanvien
(
    SoCM varchar(20) not null PRIMARY KEY
    MaPB int not null DEFAULT 1,
    MaNguoiQL varchar(20),
    Constraint FK_MaPB Foreign Key (MaPB) References PhongBan(MaPB)
    On Delete Set Default On Update Cascade
);
```

Toàn vẹn miền giá trị (Domain Integrity)

- Dữ liệu nhập vào các cột phải thỏa một điều kiện được chỉ định
- Kiểm tra toàn vẹn miền giá trị dựa vào
  - Kiểu dữ liệu
  - Thông qua **CHECK** constraints và **Rules**
  - Phạm vi giá trị thông qua **FOREIGN KEY** constraints, **CHECK** constraints, **DEFAULT**, **NOT NULL**, rules

Toàn vẹn miền giá trị (Domain Integrity)

- **Check Constraints**: kiểm tra các dữ liệu được chèn vào một cột trước khi chấp nhận. Có thể có nhiều kiểm tra ràng buộc trong một cột
- Cú pháp

```
CREATE TABLE table_name
    (column_name data_type
    [CONSTRAINT constraint_name]
    CHECK (logical expression)
```

Toàn vẹn miền giá trị (Domain Integrity)

- **Ví dụ**:

```
CREATE TABLE  nhanvien
   (
     manv  smallint Primary Key,
     tennv varchar(50) Not Null ,
     tuoimin int Not Null Check (tuoimin >= 18),
     tuoimax int Not Null Check (tuoimax <= 40)
   )
```

Toàn vẹn miền giá trị (Domain Integrity)

- **Default constraint:** Gán giá trị mặc định cho một cột.
- Cú pháp

```
CREATE TABLE Table_name
    (Column_name Datatype [NULL| NOT NULL]
     [CONSTRAINT Constraint_name] DEFAULT
     expression[…])
```

Toàn vẹn miền giá trị (Domain Integrity)

- Ví dụ:

```
CREATE  TABLE events
(     EventID int Indentity(1, 1) Not Null,
      EventType nvarchar(10) Not Null,
      EventTitle nvarchar(100) Null,
      EventDate SmallDatetime Null Default Getdate()
)
 ALTER TABLE events ADD DEFAULT ‘party’ for EventType
```

Toàn vẹn miền giá trị (Domain Integrity)

- **Unique Constraints**: để duy trì các giá trị riêng biệt trong một cột hay tập hợp các cột không tham gia vào khóa chính.
  - Có thể chỉ định nhiều Unique constraint trên một bảng
  - Có thể chỉ định Unique constraint trên một hoặc nhiều cột chấp nhận giá trị NULL.
  - Tuy nhiên nếu chỉ định Unique constraint trên một cột thì cột đó chỉ chấp nhận một giá trị NULL.

Toàn vẹn miền giá trị (Domain Integrity)

- **Cú pháp khai báo unique constraint**

```
CREATE TABLE  table_name
(       column_name  data_type
        [CONSTRAINT constraint_name]  UNIQUE
)
```

- **Ví dụ:**

Create Table Hoadon(

OrderID int Not Null Constraint PK_MaHD PRIMARY KEY,

OrderNumber int Null Constraint UQ_ORD_NUM **UNIQUE**)

Hiệu chỉnh cấu trúc bảng

- **Thêm thuộc tính (thêm cột)**
  - **Cú pháp**:

```
ALTER TABLE  <Tên Bảng>
ADD <Tên Cột> <Kiểu Dữ Liệu> [NOT NULL]
[CONSTRAINT…]
```

Hiệu chỉnh cấu trúc bảng

- **Xóa thuộc tính (xóa cột)**
  - Cú pháp:

```
ALTER TABLE <Tên bảng>
DROP COLUMN <tên cột>
[CONSTRAINT…]
```

Hiệu chỉnh cấu trúc bảng

- **Thay đổi kiểu dữ liệu của thuộc tính**
  - Cú pháp:

```
ALTER TABLE <Tên bảng>
ALTER COLUMN <Thuộc tính> <Kiểu dữ liệu>
[CONSTRAINT…]
```

Hiệu chỉnh cấu trúc bảng

- **Thêm ràng buộc**
  - Cú pháp:

```
ALTER TABLE <Tên bảng>
ADD CONSTRAINT…
```

Hiệu chỉnh cấu trúc bảng

- **Xóa ràng buộc**
  - Cú pháp:

```
ALTER TABLE  <Tên bảng>
        DROP CONSTRAINT Constraint_name
```

Xóa bảng

- Xóa bảng và toàn bộ dữ liệu trong bảng
  - Cú pháp:

```
DROP TABLE <Tên bảng>
```

  - Ví dụ:

```
Drop Table Department
```

Xóa bảng (Truncate table)

- Lệnh **TRUNCATE TABLE** xóa tất cả các dòng trong bảng đồng thời giải phóng không gian lưu trữ bảng.
  - Cú pháp:

```
TRUNCATE TABLE Tên bảng
```

Ngôn ngữ thao tác dữ liệu

- **Thêm một record mới vào bảng**:
  - Cú pháp:

```
INSERT INTO tablename
VALUES ('value1', 'value2',. . . ,'valueN')
```

  - Ví dụ:

Insert into sinhvien values (’01’,’Le van A',‘CDTH1A')

Ngôn ngữ thao tác dữ liệu

- **Cập nhật dữ liệu trong bảng**:
  - Cú pháp:

```
UPDATE table name
SET attribute value=new value
WHERE condition
```

  - Ví dụ:

```
update SinhVien
set MaLop='CDTH1A'
where MaSV='A01'
```

Ngôn ngữ thao tác dữ liệu

- **Xóa các record trong bảng:**
  - Cú pháp:

```
DELETE FROM table name
WHERE condition
```

- Ví dụ:

```
Delete from Sinhvien
Where Malop='CDTH1A'
```

Câu lệnh Select

- Cú pháp

```
SELECT [DISTINCT] select_list
[INTO new_table ]
FROM table_source
[WHERE search_condition ]
[GROUP BY group_by_expression ]
[HAVING search_condition]
[ORDER BY order_expression [ASC|DESC ] ]
```

Câu lệnh Select

- [DISTINCT]: loại bỏ tất cả các hàng trùng lặp từ kết quả truy vấn.
- Ví dụ:

```
SELECT DISTINCT MaKH, TenKH
FROM Khachhang AS kh
INNER JOIN HoaDon AS hd
ON kh.MaKH= hd.MaKH
ORDER BY kh.MaKH
```

Câu lệnh Select

- **[WHERE]:** Biểu thức điều kiện kiểu Boolean, xác định các record trong kết quả của truy vấn
- Ví dụ:

```
SELECT HoNV, TenNV, Diachi
FROM Nhanvien, PhongBan
WHERE MaPB='Research‘
```

Câu lệnh Select

- **[GROUP BY]:** tổng hợp các nhóm con của các record trong một quan hệ, mà trong đó các phân nhóm được dựa trên một số giá trị thuộc tính
- Ví dụ:

```
SELECT DNa, COUNT (*), AVG (SALARY)
FROM EMPLOYEE
GROUP BY DNa;
```

Câu lệnh Select

- Kết quả của câu lệnh:

FNAME	MINIT	LNAME	SSN	. . .	SALARY	SUPERSSN	DNO
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	. . .	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	null	1

DNO	COUNT (*)	AVG (SALARY)
5	4	33250
4	3	31000
1	1	55000

Grouping EMPLOYEE tuples by the value of DNO.

Câu lệnh Select

- **[HAVING]:** để hạn chế điều kiện đầu ra của một câu lệnh SQL có sử dụng chức năng tổng hợp.
- Ví dụ:

```
SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT (*) > 2;
```

Câu lệnh Select

- Kết quả:

PNAME	PNUMBER		ESSN	PNO	HOURS
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10	• • •	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	null
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

PNAME	COUNT (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

(PNUMBER not shown).

After applying the HAVING clause condition.

Câu lệnh Select

- **[ORDER BY]:** Sắp xếp dữ liệu trong kết quả

Ví dụ:

```
SELECT c.CustomerID,
COUNT(o.orderID) Total Orders',
SUM (od.UnitPrice * od.Quantity) 'Total Sales'
FROM Customers c, Orders o, [Order Details] od
WHERE c.CustomerID = o.CustomerID
AND o.OrderID = od.OrderID
GROUP BY c.CustomerID ORDER BY c.CustomerID
```

Gán nhãn (Alias)

- Nếu các thuộc tính ở các bảng khác nhau có cùng tên, thì khi thực hiện câu lệnh SQL cần phải chỉ rõ thuộc tính đó thuộc bảng nào để tránh sự nhập nhằng.
- Bí danh (Alias) của một cột trong SQL được sử dụng cho kết quả của câu truy vấn dễ đọc và rõ nghĩa hơn.

Gán nhãn (Alias)

- Ví dụ:

```
Select E.Fname, E.Lname, S.Fname, S.Lname
From Employee As E, Employee As S
Where E.Superssn=S.Ssn
```

Các loại liên kết (JOIN)

- **INNER JOIN:** Trả về những dòng mà giá trị trong cột liên kết giữa hai bảng là bằng nhau.
  - **Cú pháp:**

```
SELECT col_name(s)
FROM table1
INNER JOIN table2
ON table1.col_name=table2.col_name
```

Các loại liên kết (JOIN)

- Ví dụ:

```
SELECT KhachHang.MaKH, TenKH, MaHD, NgayHD
FROM KhachHang INNER JOIN HoaDon
ON Khachhang.MaKH = HoaDon.MaHD
```

Các loại liên kết (JOIN)

- **OUTER JOIN:**
  - **LEFT OUTER JOIN**: trả về kết quả là những dòng trong bảng bên trái (Bảng cha), ngay cả những dòng không so trùng với bảng bên phải (Bảng con).
  - **RIGHT OUTER JOIN:** trả về tất cả những dòng của bảng bên phải (bảng con), ngay cả những dòng không so trùng với bảng bên trái (bảng cha).

Các loại liên kết (JOIN)

- **Cú pháp**:

```
SELECT colname(s)
FROM table1 {LEFT OUTER JOIN |
             RIGHT OUTER JOIN} table2
ON table1.colname=table2.colname
```

Các loại liên kết (JOIN)

- Ví dụ: *liệt kê danh sách các nhân viên không lập hóa đơn*

```
select n.manv, TenNV, h.MaNV
from nhanvien n left outer join hoadon h
on n.MaNV=h.MaNV
where h.MaNV is null
```

Các loại liên kết (JOIN)

- **CROSS JOIN:** trả về tất cả các record mà mỗi dòng trong bảng thứ nhất kết hợp với tất cả các dòng trong bảng thứ hai, CROSS JOIN tương đương với tích descartes của các record trong hai bảng tham gia.
  - Cú pháp:

```
SELECT column_list
FROM table1 CROSS JOIN table2
```

Các loại liên kết (JOIN)

- **CROSS JOIN:** trả về tất cả các record mà mỗi dòng trong bảng thứ nhất kết hợp với tất cả các dòng trong bảng thứ hai, CROSS JOIN tương đương với tích descartes của các record trong hai bảng tham gia.
  - Cú pháp:

```
SELECT column_list
FROM table1 CROSS JOIN table2
```