

CHƯƠNG 3: LÀM VIỆC VỚI CSDL

3.1 Ngôn Ngữ SQL

3.1.1 Khái niệm

Ngôn ngữ truy vấn có cấu trúc (SQL) và các hệ quản trị cơ sở dữ liệu quan hệ là một trong những nền tảng kỹ thuật quan trọng trong công nghiệp máy tính. Cho đến nay, có thể nói rằng SQL đã được xem là ngôn ngữ chuẩn trong cơ sở dữ liệu. Các hệ quản trị cơ sở dữ liệu quan hệ thương mại hiện có như Oracle, SQL Server, Informix, DB2,... đều chọn SQL làm ngôn ngữ cho sản phẩm của mình.

3.1.1.1 SQL là ngôn ngữ CSDL quan hệ

SQL, viết tắt của Structured Query Language (ngôn ngữ hỏi có cấu trúc), là công cụ sử dụng để tổ chức, quản lý và truy xuất dữ liệu được lưu trữ trong các cơ sở dữ liệu. SQL là một hệ thống ngôn ngữ bao gồm tập các câu lệnh sử dụng để tương tác với cơ sở dữ liệu quan hệ.

Tên gọi ngôn ngữ hỏi có cấu trúc phần nào làm chúng ta liên tưởng đến một công cụ (ngôn ngữ) dùng để truy xuất dữ liệu trong các cơ sở dữ liệu. Thực sự mà nói, khả năng của SQL vượt xa so với một công cụ truy xuất dữ liệu, mặc dù đây là mục đích ban đầu khi SQL được xây dựng nên và truy xuất dữ liệu vẫn còn là một trong những chức năng quan trọng của nó. SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

- ✓ **Định nghĩa dữ liệu:** SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu cũng như mối quan hệ giữa các thành phần dữ liệu.
- ✓ **Truy xuất và thao tác dữ liệu:** Với SQL, người dùng có thể dễ dàng thực hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.
- ✓ **Điều khiển truy cập:** SQL có thể được sử dụng để cấp phát và kiểm soát các thao tác của người sử dụng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu.
- ✓ **Đảm bảo toàn vẹn dữ liệu:** SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập nhật cũng như các lỗi của hệ thống.

Như vậy, có thể nói rằng SQL là một ngôn ngữ hoàn thiện được sử dụng trong các hệ thống cơ sở dữ liệu và là một thành phần không thể thiếu trong các hệ quản trị cơ sở dữ liệu. Mặc dù SQL không phải là một ngôn ngữ lập trình như C#, C++, Java,... song các câu lệnh mà SQL cung cấp có thể được nhúng vào trong các ngôn ngữ lập trình nhằm xây dựng các ứng dụng tương tác với cơ sở dữ liệu.

Khác với các ngôn ngữ lập trình quen thuộc như C#, C++, Java,... SQL là ngôn ngữ có tính khai báo. Với SQL, người dùng chỉ cần mô tả các yêu cầu cần phải thực hiện trên cơ sở dữ liệu mà không cần phải chỉ ra cách thức thực hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ dễ tiếp cận và dễ sử dụng.

3.1.1.2 Vai trò của SQL

Bản thân SQL không phải là một hệ quản trị cơ sở dữ liệu, nó không thể tồn tại độc lập. SQL thực sự là một phần của hệ quản trị cơ sở dữ liệu, nó xuất hiện trong các hệ quản trị cơ sở dữ liệu với vai trò ngôn ngữ và là công cụ giao tiếp giữa người sử dụng và hệ quản trị cơ sở dữ liệu.

Trong hầu hết các hệ quản trị cơ sở dữ liệu quan hệ, SQL có những vai trò như sau:

- ✓ **SQL là ngôn ngữ hỏi có tính tương tác:** Người sử dụng có thể dễ dàng thông qua các trình tiện ích để gửi các yêu cầu dưới dạng các câu lệnh SQL đến cơ sở dữ liệu và nhận kết quả trả về từ cơ sở dữ liệu
- ✓ **SQL là ngôn ngữ lập trình cơ sở dữ liệu:** Các lập trình viên có thể nhúng các câu lệnh SQL vào trong các ngôn ngữ lập trình để xây dựng nên các chương trình ứng dụng giao tiếp với cơ sở dữ liệu
- ✓ **SQL là ngôn ngữ quản trị cơ sở dữ liệu:** Thông qua SQL, người quản trị cơ sở dữ liệu có thể quản lý được cơ sở dữ liệu, định nghĩa các cấu trúc lưu trữ dữ liệu, điều khiển truy cập cơ sở dữ liệu,...
- ✓ **SQL là ngôn ngữ cho các hệ thống khách/chủ (client/server):** Trong các hệ thống cơ sở dữ liệu khách/chủ, SQL được sử dụng như là công cụ để giao tiếp giữa các trình ứng dụng phía máy khách với máy chủ cơ sở dữ liệu.
- ✓ **SQL là ngôn ngữ truy cập dữ liệu trên Internet:** Cho đến nay, hầu hết các máy chủ Web cũng như các máy chủ trên Internet sử dụng SQL với vai trò là ngôn ngữ để tương tác với dữ liệu trong các cơ sở dữ liệu.
- ✓ **SQL là ngôn ngữ cơ sở dữ liệu phân tán:** Đối với các hệ quản trị cơ sở dữ liệu phân tán, mỗi một hệ thống sử dụng SQL để giao tiếp với các hệ thống khác trên mạng, gửi và nhận các yêu cầu truy xuất dữ liệu với nhau.
- ✓ **SQL là ngôn ngữ sử dụng cho các cổng giao tiếp cơ sở dữ liệu:** Trong một hệ thống mạng máy tính với nhiều hệ quản trị cơ sở dữ liệu khác nhau, SQL thường được sử dụng như là một chuẩn ngôn ngữ để giao tiếp giữa các hệ quản trị cơ sở dữ liệu.

3.1.1.3 Mô hình dữ liệu quan hệ

Mô hình dữ liệu quan hệ được Codd đề xuất năm 1970 và đến nay trở thành mô hình được sử dụng phổ biến trong các hệ quản trị cơ sở dữ liệu thương mại. Nói một cách đơn giản, một cơ sở dữ liệu quan hệ là một cơ sở dữ liệu trong đó tất cả dữ liệu được tổ chức trong các bảng có mối quan hệ với nhau. Mỗi một bảng bao gồm các dòng và các cột: mỗi một dòng được gọi là một bản ghi (bộ) và mỗi một cột là một trường (thuộc tính). Hình sau minh họa cho ta thấy được 3 bảng trong một cơ sở dữ liệu

Bảng KHOA

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054822462
DHT04	Khoa Hoá học	
...	...	

Bảng LOP

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	SISO	MAKHOA
C24101	Toán K24	24	Chính quy	2000	5	DHT01
C24102	Tin K24	24	Chính quy	2000	8	DHT02
C24103	Lý K24	24	Chính quy	2000	7	DHT03
C24301	Sinh K24	24	Chính quy	2000	5	DHT05

Bảng SINHVIEN

MASV	HODEM	TEN	NGAYSINH	GIOTINH	NOISINH	MALOP
0241010001	Ngô Thị Nhật	Anh	Nov 27 1982	0	Quảng Ninh, Quảng Bình	C24101
0241010002	Nguyễn Thị Ngọc	Anh	Mar 21 1983	0	Tân Kỳ, Nghệ An	C24101
0241010003	Ngô Việt	Bắc	May 11 1982	1	Yên Khánh, Ninh Bình	C24101
0241010004	Nguyễn Đình	Bình	Oct 6 1982	1	Huế	C24101
0241010005	Hồ Đăng	Chiến	Jan 20 1982	1	Phong Điền, TTHuế	C24101
0241020001	Nguyễn Tuấn	Anh	Jul 15 1979	1	Đo Linh, Quảng Trị	C24102
0241020002	Trần Thị Kim	Anh	Nov 4 1982	0	Phong Điền, TTHuế	C24102
0241020003	Võ Đức	Ân	May 24 1982	1	Huế	C24102
0241020004	Nguyễn Công	Bình	Jun 6 1979	1	Thăng Bình, Quảng Nam	C24102
0241020005	Nguyễn Thanh	Bình	Apr 24 1982	1	Huế	C24102
...

Bảng trong một cơ sở dữ liệu

3.1.1.4 Bảng

Như đã nói ở trên, trong cơ sở dữ liệu quan hệ, bảng là đối tượng được sử dụng để tổ chức và lưu trữ dữ liệu. Một cơ sở dữ liệu bao gồm nhiều bảng và mỗi bảng được xác định duy nhất bởi tên bảng. Một bảng bao gồm một tập các dòng và các cột: mỗi một dòng trong bảng biểu diễn cho một thực thể (trong hình trên, mỗi một dòng trong bảng SINHVIEN tương ứng với một sinh viên); và mỗi một cột biểu diễn cho một tính chất của thực thể (chẳng hạn cột NGAYSINH trong bảng SINHVIEN biểu diễn cho ngày sinh của các sinh viên được lưu trữ trong bảng).

Như vậy, liên quan đến mỗi một bảng bao gồm các yếu tố sau:

- ✓ **Tên của bảng:** được sử dụng để xác định duy nhất mỗi bảng trong cơ sở dữ liệu.
- ✓ **Cấu trúc của bảng:** Tập các cột trong bảng. Mỗi một cột trong bảng được xác định bởi một tên cột và phải có một kiểu dữ liệu nào đó (chẳng hạn cột NGAYSINH trong bảng SINHVIEN ở hình trên có kiểu là DATETIME). Kiểu dữ liệu của mỗi cột qui định giá trị dữ liệu có thể được chấp nhận trên cột đó.
- ✓ **Dữ liệu của bảng:** Tập các dòng (bản ghi) hiện có trong bảng.

3.1.1.5 Khoá của bảng

Trong một cơ sở dữ liệu được thiết kế tốt, mỗi một bảng phải có một hoặc một tập các cột mà giá trị dữ liệu của nó xác định duy nhất một dòng trong một tập các dòng của bảng. Tập một hoặc nhiều cột có tính chất này được gọi là khoá của bảng.

Việc chọn khoá của bảng có vai trò quan trọng trong việc thiết kế và cài đặt các cơ sở dữ liệu quan hệ. Các dòng dữ liệu trong một bảng phải có giá trị khác nhau trên khoá. Bảng MONHOC trong hình dưới đây có khoá là cột MAMONHOC

MAMONHOC	TENMONHOC	SODVHT
HO-001	Hoá đại cương	3
TI-001	Tin học đại cương	4
TI-002	Ngôn ngữ C	5
TI-003	Lý thuyết hệ điều hành	4
TI-004	Cấu trúc dữ liệu và giải thuật	4
TO-001	Đại số tuyến tính	4
TO-002	Giải tích 1	4
TO-003	Bài tập Đại số	2
TO-004	Bài tập Giải tích 1	2
VL-001	Vật lý đại cương	3

Bảng MONHOC với khoá chính là MAMONHOC

Một bảng có thể có nhiều tập các cột khác nhau có tính chất của khoá (tức là giá trị của nó xác định duy nhất một dòng dữ liệu trong bảng). Trong trường hợp này, khoá được chọn cho bảng được gọi là khoá chính (primary key) và những khoá còn lại được gọi là khoá phụ hay là khoá dự tuyển (candidate key/unique key).

3.1.1.6 Mỗi quan hệ và khoá ngoài

Các bảng trong một cơ sở dữ liệu không tồn tại độc lập mà có mối quan hệ mật thiết với nhau về mặt dữ liệu. Mối quan hệ này được thể hiện thông qua ràng buộc giá trị dữ liệu xuất hiện ở bảng này phải có xuất hiện trước trong một bảng khác. Mối quan hệ giữa các bảng trong cơ sở dữ liệu nhằm đảm bảo được tính đúng đắn và hợp lệ của dữ liệu trong cơ sở dữ liệu.

Trong hình trên, hai bảng LOP và KHOA có mối quan hệ với nhau. Mối quan hệ này đòi hỏi giá trị cột MAKHOA của một dòng (tức là một lớp) trong bảng LOP phải được xác định từ cột MAKHOA của bảng KHOA.

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
...

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	SISO	MAKHOA
C24101	Toán K24	24	Chính quy	2000	5	DHT01
C25101	Toán K25	25	Chính quy	2001	5	DHT01
C25102	Tin K25	25	Chính quy	2001	6	DHT02
C24102	Tin K24	24	Chính quy	2000	8	DHT02
...

Bảng LOP

Mối quan hệ giữa hai bảng LOP và KHOA trong cơ sở dữ liệu

Mỗi quan hệ giữa các bảng trong một cơ sở dữ liệu thể hiện đúng mối quan hệ giữa các thực thể trong thế giới thực. Trong hình trên, mỗi quan hệ giữa hai bảng LOP và KHOA không cho phép một lớp nào đó tồn tại mà lại thuộc vào một khoa không có thật.

Khái niệm khoá ngoài (Foreign Key) trong cơ sở dữ liệu quan hệ được sử dụng để biểu diễn mối quan hệ giữa các bảng dữ liệu. Một hay một tập các cột trong một bảng mà giá trị của nó được xác định từ khóa chính của một bảng khác được gọi là khoá ngoài. Trong hình trên, cột MAKHOA của bảng LOP được gọi là khoá ngoài của bảng này, khoá ngoài này tham chiếu đến khoá chính của bảng KHOA là cột MAKHOA.

3.1.2 Ngôn ngữ định nghĩa dữ liệu

Các câu lệnh SQL đã đề cập đến trong chương 3 được sử dụng nhằm thực hiện các thao tác bổ sung, cập nhật, loại bỏ và xem dữ liệu. Nhóm các câu lệnh này được gọi là ngôn ngữ thao tác dữ liệu (DML). Trong chương này, chúng ta sẽ tìm hiểu nhóm các câu lệnh được sử dụng để định nghĩa và quản lý các đối tượng CSDL như bảng, khung nhìn, chỉ mục,... và được gọi là ngôn ngữ định nghĩa dữ liệu (DDL).

Về cơ bản, ngôn ngữ định nghĩa dữ liệu bao gồm các lệnh:

- ✓ **CREATE:** định nghĩa và tạo mới đối tượng CSDL.
- ✓ **ALTER:** thay đổi định nghĩa của đối tượng CSDL.
- ✓ **DROP:** Xóa đối tượng CSDL đã có.

3.1.2.1 Tạo bảng dữ liệu

Câu lệnh CREATE TABLE có cú pháp như sau

```
CREATE TABLE tên_bảng  
(  
    tên_cột_1    kiểu_dữ_liệu_1    các_ràng_buộc_1,  
    tên_cột_2    kiểu_dữ_liệu_2    các_ràng_buộc_2  
    ...,  
    [,các_ràng_buộc_trên_bảng]  
)
```

Ví dụ: Câu lệnh dưới đây định nghĩa bảng NHANVIEN với các trường MANV (mã nhân viên), HOTEN (họ và tên), NGAYSINH (ngày sinh của nhân viên), DIENTHOAI (điện thoại) và HSLUONG (hệ số lương)

```
CREATE TABLE NhanVien  
(  
    MaNV NVARCHAR(10) NOT NULL,  
    HoTen NVARCHAR(50) NOT NULL,  
    NgaySinh DATETIME NULL,  
    DienThoai NVARCHAR(10) NULL,  
    HSLuong DECIMAL(3,2) DEFAULT (1.92)  
)
```

Trong câu lệnh trên, trường MANV và HOTEN của bảng NHANVIEN không được NULL (tức là bắt buộc phải có dữ liệu), trường NGAYSINH và DIENTHOAI sẽ nhận giá trị NULL nếu ta

không nhập dữ liệu cho chúng còn trường HSLUONG sẽ nhận giá trị mặc định là 1.92 nếu không được nhập dữ liệu.

Nếu ta thực hiện các câu lệnh dưới đây sau khi thực hiện câu lệnh trên để bổ sung dữ liệu cho bảng NHANVIEN

```
INSERT INTO nhanvien VALUES('NV01','Le Van A','2/4/75','886963',2.14)
INSERT INTO nhanvien(manv,hoten) VALUES('NV02','Mai Thi B')
INSERT INTO nhanvien(manv,hoten,dienthoai) VALUES('NV03','Tran Thi C','849290')
```

Ta sẽ có được dữ liệu trong bảng NHANVIEN như sau:

MANV	HOTEN	NGAYSINH	DIENTHOAI	HSLUONG
NV01	Le Van A	1975-02-04 00:00:00.000	886963	2.14
NV02	Mai Thi B	NULL	NULL	1.92
NV03	Tran Thi C	NULL	849290	1.92

Kết quả sau thao tác

3.1.2.2 Kiểu dữ liệu

Chuẩn ANSI/ISO SQL cung cấp các kiểu dữ liệu khác nhau để sử dụng trong các cơ sở dữ liệu dựa trên SQL và trong ngôn ngữ SQL. Dựa trên cơ sở các kiểu dữ liệu do chuẩn ANSI/ISO SQL cung cấp, cách quản trị cơ sở dữ liệu thương mại hiện nay có thể sử dụng các dạng dữ liệu khác nhau trong sản phẩm của mình. Bảng dưới đây liệt kê một số kiểu dữ liệu thông dụng được sử dụng trong SQL.

Nhóm	Kiểu	Mô tả
Chuỗi	[N]CHAR(n)	Chuỗi có độ dài cố định, biến đổi và cực lớn. [N] chỉ định lưu unicode, (n) chỉ định số ký tự tối đa.
	[N]VARCHAR(n)	
	[N]TEXT	
Số	BIT	Số nguyên 1 bit (dùng cho kiểu logic)
	SMALLINT	Số nguyên cỡ nhỏ, vừa và lớn
	INT	
	BIGINT	
	FLOAT	Số thực
	NUMERIC	
	DECIMAL	
Ngày	MONEY	Số cực lớn, lưu tiền tệ
	DATETIME	Ngày và giờ, Ngày, giờ
	DATE	
	TIME	

Nhị phân	BINARY(n) VARBINARY(n) IMAGE	Nhị phân số byte cố định, biến đổi và cực lớn
----------	------------------------------------	---

Một số kiểu dữ liệu thông dụng trong SQL

3.1.2.3 Ràng buộc

Ràng buộc được sử dụng để qui định cho dữ liệu trên các cột khi định nghĩa hoặc sử đổi. Sau đây là một số ràng buộc thường dùng trong SQL.

Ràng buộc	Ý nghĩa	Ví dụ
NULL	Cho phép null	NgaySinh NULL
NOT NULL	Bắt buộc phải nhập	Email NOT NULL
DEFAULT	Giá trị mặc định	Diem DEFAULT 0
UNIQUE	Duy nhất	CMND UNIQUE
CHECK	Kiểm tra	CHECK(Diem >=0 AND Diem<=10)
PRIMARY KEY	Khóa chính	PRIMARY KEY (MaNV)
FOREIGN KEY	Khóa ngoại	FOREIGN KEY(MaPB) REFERENCES PhongBan(MaPB)

Ví dụ: Câu lệnh dưới đây tạo bảng DIEMTOTNGHIEP trong đó qui định giá trị của cột DIEMTOAN phải lớn hơn hoặc bằng 0 và nhỏ hơn hoặc bằng 10. Và nếu không nhập thì điểm toán sẽ chấp nhận điểm 0.

```
CREATE TABLE diemtotnghiep
(
    hoten NVARCHAR(30) NOT NULL,
    diemtoan DECIMAL(4,2) DEFAULT 0,
    CHECK(diemtoan>=0 AND diemtoan<=10)
)
```

Nếu ràng buộc chỉ liên quan đến 1 cột, bạn có thể đặt ràng buộc ngay sau cột đang định nghĩa.

Ví dụ: Câu lệnh

```
CREATE TABLE lop
(
    malop NVARCHAR(10) NOT NULL ,
    namnhaphoc INT NULL CHECK (namnhaphoc <= YEAR(GETDATE())),
    makhoa NVARCHAR(5)
)
```


Ví dụ: Câu lệnh tạo bảng sau sẽ cho chúng ta thấy cách sử dụng các ràng buộc khá đầy đủ

```
CREATE TABLE sinhvien
(
    masv NVARCHAR(10) NOT NULL PRIMARY KEY ,
    cmnd VARCHAR(10) NOT NULL UNIQUE,
    diemtb NUMERIC(4, 2) DEFAULT 0,
    FOREIGN KEY(malop) REFERENCES lop(malop)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CHECK(diemtb >= 0 AND diemtb <= 10)
)
```

Lưu ý về khóa ngoại:

- ✓ Cột được tham chiếu trong bảng tham chiếu phải là **khoá chính** (hoặc là khoá phụ).
- ✓ Cột được tham chiếu phải có **cùng kiểu** dữ liệu và độ dài với cột tương ứng trong khóa ngoại.
- ✓ Bảng tham chiếu phải được **định nghĩa trước**. Do đó, nếu các bảng có mối quan hệ vòng, ta có thể không thể định nghĩa ràng buộc FOREIGN KEY ngay trong câu lệnh CREATE TABLE mà phải định nghĩa thông qua lệnh ALTER TABLE

3.1.2.4 Sửa đổi định nghĩa bảng

Một bảng sau khi đã được định nghĩa bằng câu lệnh CREATE TABLE có thể được sửa đổi thông qua câu lệnh **ALTER TABLE**. Câu lệnh này cho phép chúng ta thực hiện được các thao tác sau:

- ✓ Bổ sung một cột vào bảng.
- ✓ Xoá một cột khỏi bảng.
- ✓ Thay đổi định nghĩa của một cột trong bảng.
- ✓ Xoá bỏ hoặc bổ sung các ràng buộc cho bảng

Cú pháp của câu lệnh **ALTER TABLE** như sau:

```
ALTER TABLE tên_bảng
    ADD định_nghĩa_cột |
    ALTER COLUMN tên_cột kiểu_dữ_liệu [NULL | NOT NULL] |
    DROP COLUMN tên_cột |
    ADD CONSTRAINT tên_ràng_buộc định_nghĩa_ràng_buộc |
    DROP CONSTRAINT tên_ràng_buộc
```

Ví dụ: Các ví dụ dưới đây minh họa cho ta cách sử dụng câu lệnh ALTER TABLE trong các trường hợp.

Bổ sung vào bảng NHANVIEN cột DIENTHOAI với ràng buộc CHECK nhằm qui định điện thoại của nhân viên là một chuỗi 6 chữ số:

```
ALTER TABLE nhanvien
    ADD dienthoai NVARCHAR(6) CHECK (dienthoai LIKE '[0-9][0-9][0-9][0-9][0-9][0-9]')
```


Định nghĩa lại kiểu dữ liệu của cột DIACHI trong bảng NHANVIEN và cho phép cột này chấp nhận giá trị NULL:

```
ALTER TABLE nhanvien  
    ALTER COLUMN diachi NVARCHAR(100) NULL
```

Xoá cột ngày sinh khỏi bảng NHANVIEN:

```
ALTER TABLE nhanvien  
    DROP COLUMN ngaysinh
```

Định nghĩa khoá chính (ràng buộc PRIMARY KEY) cho bảng NHANVIEN là cột MANV:

```
ALTER TABLE nhanvien  
    ADD PRIMARY KEY(manv)
```

Định nghĩa khoá ngoài cho bảng NHANVIEN trên cột MADV tham chiếu đến cột MADV của bảng DONVI:

```
ALTER TABLE nhanvien  
    ADD FOREIGN KEY(madv) REFERENCES donvi(madv)  
    ON DELETE CASCADE ON UPDATE CASCADE
```

Xoá bỏ ràng buộc kiểm tra số điện thoại của nhân viên

```
ALTER TABLE nhanvien  
    DROP CONSTRAINT CHK_NHANVIEN_DIENTHOAI
```

Lưu ý:

- ✓ Nếu bổ sung thêm một cột vào bảng và trong bảng đã có ít nhất một bản ghi thì cột mới cần bổ sung phải cho phép chấp nhận giá trị NULL hoặc phải có giá trị mặc định.
- ✓ Muốn xoá một cột đang được ràng buộc bởi một ràng buộc hoặc đang được tham chiếu bởi một khoá ngoài, ta phải xoá ràng buộc hoặc khoá ngoài trước sao cho trên cột không còn bất kỳ một ràng buộc và không còn được tham chiếu bởi bất kỳ khoá ngoài nào.
- ✓ Nếu bổ sung thêm ràng buộc cho một bảng đã có dữ liệu và ràng buộc cần bổ sung không được thoả mãn bởi các bản ghi đã có trong bảng thì câu lệnh ALTER TABLE không thực hiện được.

3.1.2.5 Xoá bảng

Câu lệnh có cú pháp như sau:

```
DROP TABLE tên_bảng
```

Câu lệnh DROP TABLE không thể thực hiện được nếu bảng cần xoá đang được tham chiếu bởi một ràng buộc FOREIGN KEY. Trong trường hợp này, ràng buộc FOREIGN KEY đang tham chiếu hoặc bảng đang tham chiếu đến bảng cần xoá phải được xoá trước.

Ví dụ: Giả sử cột MADV trong bảng DONVI đang được tham chiếu bởi khoá ngoài fk_nhanvien_madv trong bảng NHANVIEN. Để xoá bảng DONVI ra khỏi cơ sở dữ liệu, ta thực hiện hai câu lệnh sau:

Xoá bỏ ràng buộc fk_nhanvien_madv khỏi bảng NHANVIEN:

```
ALTER TABLE nhanvien
DROP CONSTRAINT fk_nhanvien_madv
```

Xoá bảng DONVI:

```
DROP TABLE donvi
```

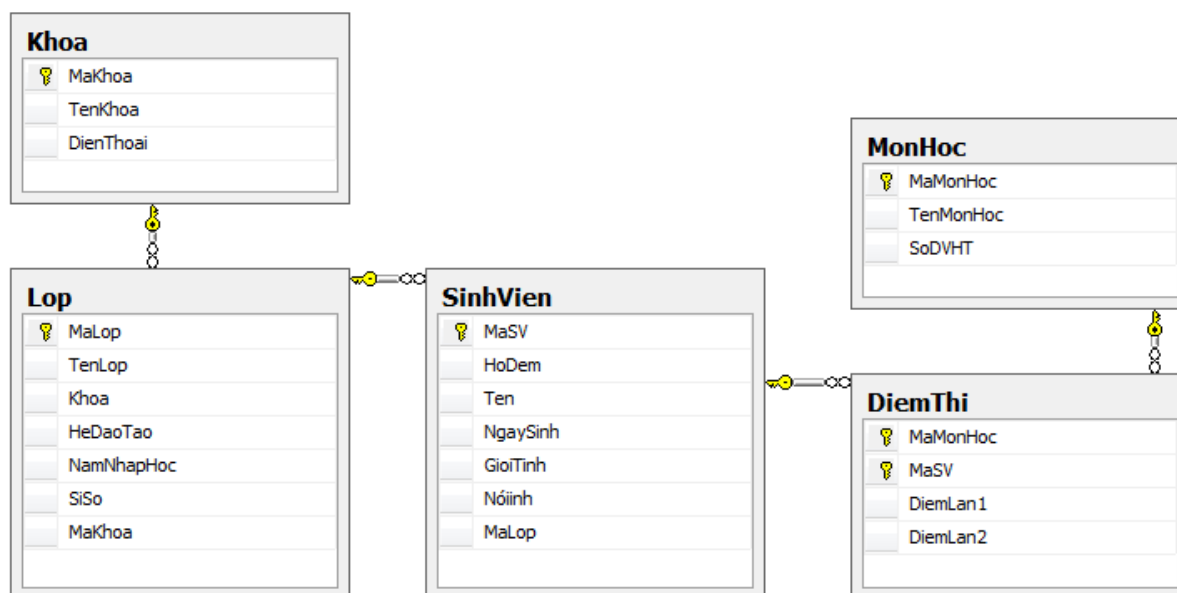
3.1.2.6 Giới thiệu CSDL mẫu

Trong toàn bộ nội dung giáo trình, hầu hết các ví dụ được dựa trên cơ sở dữ liệu mẫu được mô tả dưới đây. Cơ sở dữ liệu này được cài đặt trong hệ quản trị cơ sở dữ liệu SQL Server và được sử dụng để quản lý sinh viên và điểm thi của sinh viên trong một trường đại học. Để tiện cho việc tra cứu và kiểm chứng đối với các ví dụ, trong phần đầu của phụ lục chúng tôi giới thiệu sơ qua về cơ sở dữ liệu này.

Cơ sở dữ liệu bao gồm các bảng sau đây:

- ✓ Bảng KHOA lưu trữ dữ liệu về các khoa hiện có ở trong trường
- ✓ Bảng LOP bao gồm dữ liệu về các lớp trong trường
- ✓ Bảng SINHVIEN được sử dụng để lưu trữ dữ liệu về các sinh viên trong trường.
- ✓ Bảng MONHOC bao gồm các môn học (học phần) được giảng dạy trong trường
- ✓ Bảng DIEMTHI với dữ liệu cho biết điểm thi kết thúc môn học của các sinh viên

Mối quan hệ giữa các bảng được thể hiện qua sơ đồ dưới đây



CSDL đào tạo

Các bảng trong cơ sở dữ liệu, mối quan hệ giữa chúng và một số ràng buộc được cài đặt như sau:

```
CREATE TABLE Khoa
(
    MaKhoa NVARCHAR(5) NOT NULL,
    TenKhoa NVARCHAR(50) NOT NULL ,
    DienThoai NVARCHAR(15) NULL,
```



```

        CONSTRAINT PK_Khoa PRIMARY KEY (MaKhoa)
    )

CREATE TABLE Lop
(
    MaLop NVARCHAR(10) NOT NULL,
    TenLop NVARCHAR(30) NULL ,
    Khoa SMALLINT NULL ,
    HeDaoTao NVARCHAR(25) NULL ,
    NamNhapHoc INT NULL ,
    SiSo INT NULL ,
    MaKhoa NVARCHAR(5) NULL,
    CONSTRAINT PK_Lop PRIMARY KEY (MaLop)
)

CREATE TABLE SinhVien
(
    MaSV NVARCHAR(10) NOT NULL,
    HoDem NVARCHAR(25) NOT NULL ,
    Ten NVARCHAR(10) NOT NULL ,
    NgaySinh SMALLDATETIME NULL ,
    GioiTinh BIT NULL ,
    Nôiinh NVARCHAR(100) NULL ,
    MaLop NVARCHAR(10) NULL,
    CONSTRAINT PK_SinhVien PRIMARY KEY (MaSV)
)

CREATE TABLE MonHoc
(
    MaMonHoc NVARCHAR(10) NOT NULL,
    TenMonHoc NVARCHAR(50) NOT NULL ,
    SoDVHT SMALLINT NOT NULL,
    CONSTRAINT PK_MonHoc PRIMARY KEY (MaMonHoc)
)

CREATE TABLE DiemThi
(
    MaMonHoc NVARCHAR(10) NOT NULL ,
    MaSV NVARCHAR(10) NOT NULL ,
    DiemLan1 NUMERIC(5, 2) NULL ,
    DiemLan2 NUMERIC(5, 2) NULL,
    CONSTRAINT PK_DiemThi PRIMARY KEY(MaMonHoc, MaSV)
)

ALTER TABLE Lop    ADD
    CONSTRAINT FK_Lop_Khoa FOREIGN KEY(MaKhoa)
    REFERENCES Khoa(MaKhoa) ON DELETE CASCADE ON UPDATE CASCADE

ALTER TABLE SinhVien ADD
    CONSTRAINT FK_SinhVien_Lop FOREIGN KEY (MaLop)
    REFERENCES Lop(malop) ON DELETE CASCADE ON UPDATE CASCADE

```

```
ALTER TABLE DiemThi ADD
CONSTRAINT FK_DiemThi_MonHoc FOREIGN KEY (MaMonHoc)
REFERENCES MonHoc(MaMonHoc) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT FK_DiemThi_SinhVien FOREIGN KEY (MaSV)
REFERENCES SinhVien(MaSV) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT CHK_DiemLan1 CHECK (DiemLan1 >= 0 AND DiemLan1 <= 10),
CONSTRAINT CHK_DiemLan2 CHECK (DiemLan2 >= 0 AND DiemLan2 <= 10)

ALTER TABLE MonHoc
ADD CONSTRAINT CHK_SoDVHT CHECK (SoDVHT > 0 AND SoDVHT <= 5)
```

3.1.3 Ngôn ngữ thao tác dữ liệu

Đối với đa số người sử dụng, SQL được xem như là công cụ hữu hiệu để thực hiện các yêu cầu truy vấn và thao tác trên dữ liệu. Trong chương này, ta sẽ bàn luận đến nhóm các câu lệnh trong SQL được sử dụng cho mục đích này. Nhóm các câu lệnh này được gọi chung là ngôn ngữ thao tác dữ liệu (DML: Data Manipulation Language) bao gồm các câu lệnh sau:

- ✓ **SELECT**: Sử dụng để truy xuất dữ liệu từ một hoặc nhiều bảng.
- ✓ **INSERT**: Bổ sung dữ liệu.
- ✓ **UPDATE**: Cập nhật dữ liệu
- ✓ **DELETE**: Xoá dữ liệu

Trong số các câu lệnh này, có thể nói SELECT là câu lệnh tương đối phức tạp và được sử dụng nhiều trong cơ sở dữ liệu. Với câu lệnh này, ta không chỉ thực hiện các yêu cầu truy xuất dữ liệu đơn thuần mà còn có thể thực hiện được các yêu cầu thống kê dữ liệu phức tạp. Cũng chính vì vậy, phần đầu của chương này sẽ tập trung tương đối nhiều đến câu lệnh SELECT. Các câu lệnh INSERT, UPDATE và DELETE được bàn luận đến ở cuối chương

3.1.3.1 Bổ sung, cập nhật và xoá dữ liệu

Các câu lệnh thao tác dữ liệu trong SQL không những chỉ sử dụng để truy vấn dữ liệu mà còn để thay đổi và cập nhật dữ liệu trong cơ sở dữ liệu. So với câu lệnh SELECT, việc sử dụng các câu lệnh để bổ sung, cập nhật hay xoá dữ liệu đơn giản hơn nhiều. Trong phần còn lại của chương này sẽ đề cập đến 3 câu lệnh:

- ✓ Lệnh INSERT
- ✓ Lệnh UPDATE
- ✓ Lệnh DELETE

3.1.3.1.1 Bổ sung dữ liệu

Dữ liệu trong các bảng được thể hiện dưới dạng các dòng (bản ghi). Để bổ sung thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT. Hầu hết các hệ quản trị CSDL dựa trên SQL cung cấp các cách dưới đây để thực hiện thao tác bổ sung dữ liệu cho bảng:

- ✓ Bổ sung từng dòng dữ liệu với mỗi câu lệnh INSERT. Đây là các sử dụng thường gặp nhất trong giao tác SQL.
- ✓ Bổ sung nhiều dòng dữ liệu bằng cách truy xuất dữ liệu từ các bảng dữ liệu khác.

Bổ sung từng dòng dữ liệu với lệnh INSERT

Để bổ sung một dòng dữ liệu mới vào bảng, ta sử dụng câu lệnh INSERT với cú pháp như sau:

```
INSERT INTO tên_bảng[(danh_sách_cột)] VALUES(danh_sách_trị)
```

Trong câu lệnh INSERT, danh sách cột ngay sau tên bảng không cần thiết phải chỉ định nếu giá trị các trường của bản ghi mới được chỉ định đầy đủ trong danh sách trị. Trong trường hợp này, thứ tự các giá trị trong danh sách trị phải bằng với số lượng các trường của bảng cần bổ sung dữ liệu cũng như phải tuân theo thứ tự của các trường như khi bảng được định nghĩa.

Ví dụ: Câu lệnh dưới đây bổ sung thêm một dòng dữ liệu vào bảng KHOA

```
INSERT INTO khoa VALUES('DHT10', 'Khoa Luật', '054821135')
```

Trong trường hợp chỉ nhập giá trị cho một số cột trong bảng, ta phải chỉ định danh sách các cột cần nhập dữ liệu ngay sau tên bảng. Khi đó, các cột không được nhập dữ liệu sẽ nhận giá trị mặc định (nếu có) hoặc nhận giá trị NULL (nếu cột cho phép chấp nhận giá trị NULL). Nếu một cột không có giá trị mặc định và không chấp nhận giá trị NULL mà không được nhập dữ liệu, câu lệnh sẽ bị lỗi.

Ví dụ: Câu lệnh dưới đây bổ sung một bản ghi mới cho bảng SINHVIEN

```
INSERT INTO sinhvien(masv, hodem, ten, gioitinh, malop)  
VALUES('0241020008', 'Nguyễn Công', 'Chính', 1, 'C24102')
```

3.1.3.1.2 Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng. Câu lệnh này có cú pháp như sau:

```
UPDATE tên_bảng  
SET tên_cột1 = biểu_thức, tên_cột_2 = biểu_thức_2,...  
WHERE điều_kiện
```

Sau UPDATE là tên của bảng cần cập nhật dữ liệu. Một câu lệnh UPDATE có thể cập nhật dữ liệu cho nhiều cột bằng cách chỉ định các danh sách tên cột và biểu thức tương ứng sau từ khoá SET. Mệnh đề WHERE trong câu lệnh UPDATE thường được sử dụng để chỉ định các dòng dữ liệu chịu tác động của câu lệnh (nếu không chỉ định, phạm vi tác động của câu lệnh được hiểu là toàn bộ các dòng trong bảng)

Ví dụ: Câu lệnh dưới đây cập nhật lại số đơn vị học trình của các môn học có số đơn vị học trình nhỏ hơn 2

```
UPDATE monhoc  
SET sodvht = 3  
WHERE sodvht = 2
```

Sử dụng cấu trúc CASE trong câu lệnh UPDATE

Cấu trúc CASE có thể được sử dụng trong biểu thức khi cần phải đưa ra các quyết định khác nhau về giá trị của biểu thức

Ví dụ: Giả sử ta có bảng NHATKYPHONG sau đây

SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	
202	B	5	
101	A	2	
102	C	3	

Bảng thí nghiệm

Sau khi thực hiện câu lệnh:

```
UPDATE nhattyphong
SET tienphong=songay * (CASE WHEN loaiphong='A' THEN 100
                           WHEN loaiphong='B' THEN 70
                           ELSE 50
                           END)
```

Dữ liệu trong bảng sẽ là:

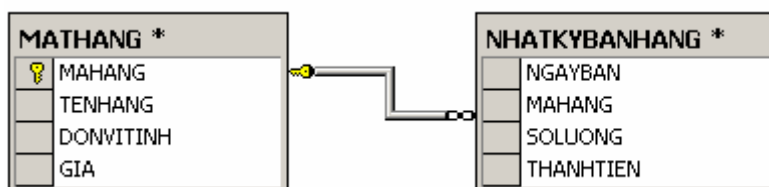
SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	500
202	B	5	350
101	A	2	200
102	C	3	150

Kết quả cập nhật

Điều kiện cập nhật dữ liệu liên quan đến nhiều bảng

Mệnh đề FROM trong câu lệnh UPDATE được sử dụng khi cần chỉ định các điều kiện liên quan đến các bảng khác với bảng cần cập nhật dữ liệu. Trong trường hợp này, trong mệnh đề WHERE thường có điều kiện nối giữa các bảng.

Ví dụ: Giả sử ta có hai bảng MATHANG và NHATKYBANHANG như sau:



Câu lệnh dưới đây sẽ cập nhật giá trị trường THANHTIEN của bảng

NHATKYBANHANG theo công thức THANHTIEN = SOLUONG × GIA

```
UPDATE nhattybanhang
SET thanhtien = soluong*gia
FROM mathang
WHERE nhattybanhang.mahang = mathang.mahang
```

3.1.3.1.3 Xóa dữ liệu

Để xóa dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE. Cú pháp của câu lệnh này như sau:

```
DELETE FROM tên_bảng [FROM danh_sách_bảng] [WHERE điều_kiện]
```

Trong câu lệnh này, tên của bảng cần xoá dữ liệu được chỉ định sau DELETE FROM. Mệnh đề WHERE trong câu lệnh được sử dụng để chỉ định điều kiện đối với các dòng dữ liệu cần xoá. Nếu câu lệnh DELETE không có mệnh đề WHERE thì toàn bộ các dòng dữ liệu trong bảng đều bị xoá.

Ví dụ: Câu lệnh dưới đây xoá khỏi bảng SINHVIEN những sinh viên sinh tại Huế

```
DELETE FROM sinhvien WHERE noisinh LIKE '%Huế%'
```

3.1.3.2 Truy xuất dữ liệu với câu lệnh SELECT

Câu lệnh SELECT được sử dụng để truy xuất dữ liệu từ các dòng và các cột của một hay nhiều bảng, khung nhìn. Câu lệnh này có thể dùng để thực hiện phép chọn (tức là truy xuất một tập con các dòng trong một hay nhiều bảng), phép chiếu (tức là truy xuất một tập con các cột trong một hay nhiều bảng) và phép nối (tức là liên kết các dòng trong hai hay nhiều bảng để truy xuất dữ liệu). Ngoài ra, câu lệnh này còn cung cấp khả năng thực hiện các thao tác truy vấn và thống kê dữ liệu phức tạp khác.

Cú pháp chung của câu lệnh SELECT có dạng:

```
SELECT [ALL | DISTINCT][TOP n] danh_sách_cột  
FROM danh_sách_bảng/khung_nhìn  
[WHERE điều_kiện]  
[GROUP BY danh_sách_cột]  
[HAVING điều_kiện]  
[ORDER BY cột_sắp_xếp]
```

Điều cần lưu ý đầu tiên đối với câu lệnh này là các thành phần trong câu lệnh SELECT nếu được sử dụng phải tuân theo đúng thứ tự như trong cú pháp. Nếu không, câu lệnh sẽ được xem là không hợp lệ.

Câu lệnh SELECT được sử dụng để tác động lên các bảng dữ liệu và kết quả của câu lệnh cũng được hiển thị dưới dạng bảng, tức là một tập hợp các dòng và các cột.

Ví dụ: Kết quả của câu lệnh sau đây cho biết mã lớp, tên lớp và hệ đào tạo của các lớp hiện có

```
SELECT malop,tenlop,hedaotao FROM lop
```

MALOP	TENLOP	HEDAOTAO
C24101	Toán K24	Chính quy
C24102	Tin K24	Chính quy
C24103	Lý K24	Chính quy
C24301	Sinh K24	Chính quy
C25101	Toán K25	Chính quy
C25102	Tin K25	Chính quy
C25103	Lý K25	Chính quy
C25301	Sinh K25	Chính quy
C26101	Toán K26	Chính quy
C26102	Tin K26	Chính quy

Kết quả truy vấn

3.1.3.2.1 Danh sách cột trong câu lệnh SELECT

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các trường, các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường, các biểu thức được chỉ định ngay sau từ khoá SELECT và phân cách nhau bởi dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

Chọn tất cả các cột trong bảng

```
SELECT * FROM lop
```

Tên cột trong danh sách chọn

```
SELECT malop,tenlop,namnhaphoc,khoa FROM lop
```

Thay đổi tiêu đề các cột

```
SELECT 'Mã lớp' = malop,tenlop 'Tên lớp',khoa AS 'Khoá' FROM lop
```

Sử dụng cấu trúc CASE trong danh sách chọn

```
SELECT masv,hodem,ten,  
       CASE gioitinh  
         WHEN 1 THEN 'Nam' ELSE 'Nữ'  
       END AS gioitinh  
FROM sinhvien
```

hoặc:

```
SELECT masv,hodem,ten,  
       CASE  
         WHEN gioitinh=1 THEN 'Nam'  
         ELSE 'Nữ'  
       END AS gioitinh  
FROM sinhvien
```

Hằng và biểu thức trong danh sách chọn

```
SELECT tenmonhoc,'Số tiết: ',sodvht*15 AS sotiet FROM monhoc
```

Loại bỏ các dòng dữ liệu trùng nhau trong kết quả truy vấn

```
SELECT DISTINCT khoa FROM lop
```

Giới hạn số lượng dòng trong kết quả truy vấn

```
SELECT TOP 5 hodem,ten,ngaysinh FROM sinhvien  
SELECT TOP 10 PERCENT hodem,ten,ngaysinh FROM sinhvien
```

3.1.3.2.2 Chỉ định điều kiện truy vấn dữ liệu

Mệnh đề **WHERE** trong câu lệnh SELECT được sử dụng nhằm xác định các điều kiện đối với việc truy xuất dữ liệu. Sau mệnh đề WHERE là một biểu thức logic và chỉ những dòng dữ liệu nào thoả mãn điều kiện được chỉ định mới được hiển thị trong kết quả truy vấn.

Ví dụ: Câu lệnh dưới đây hiển thị danh sách các môn học có số đơn vị học trình lớn hơn 3

```
SELECT * FROM monhoc WHERE sodvht > 3
```

Trong mệnh đề WHERE thường sử dụng:

- ✓ Các toán tử quan hệ (AND, OR)
- ✓ Các toán tử so sánh
- ✓ Kiểm tra giới hạn của dữ liệu (BETWEEN/ NOT BETWEEN)
- ✓ Tập hợp IN (v1,v2,...)
- ✓ Kiểm tra khuôn dạng dữ liệu.
- ✓ Các giá trị NULL

❑ Các toán tử so sánh

Toán tử	ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ: Câu lệnh:

```
SELECT masv,hodem,ten,ngaysinh
FROM sinhvien
WHERE (ten='Anh') AND (YEAR(GETDATE())-YEAR(ngaysinh)<=20)
```

❑ Kiểm tra giới hạn của dữ liệu

Để kiểm tra xem giá trị dữ liệu nằm trong (ngoài) một khoảng nào đó, ta sử dụng toán tử BETWEEN (NOT BETWEEN) như sau:

Cách sử dụng	Ý nghĩa
giá_trị BETWEEN a AND b	$a \leq \text{giá_trị} \leq b$
giá_trị NOT BETWEEN a AND b	$(\text{giá_trị} < a) \text{ AND } (\text{giá_trị} > b)$

Ví dụ: Câu lệnh dưới đây cho biết họ tên và tuổi của các sinh viên có tên là Bình và có tuổi nằm trong khoảng từ 20 đến 22

```
SELECT hodem,ten,year(getdate())-year(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten='Bình' AND YEAR(GETDATE())-YEAR(ngaysinh) BETWEEN 20 AND 22
```

❑ Danh sách (IN và NOT IN)

Từ khoá IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu lệnh SELECT là một danh sách các giá trị. Sau IN (hoặc NOT IN) có thể là một danh sách các giá trị hoặc là một câu lệnh SELECT khác.

Ví dụ: Để biết danh sách các môn học có số đơn vị học trình là 2, 4 hoặc 5, thay vì sử dụng câu lệnh

```
SELECT * FROM monhoc WHERE sodvht IN (2,4,5)
```

❑ Toán tử LIKE và các ký tự đại diện

Từ khoá LIKE (NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm kiếm. Chúng thường được kết hợp với các ký tự đại diện sau đây:

Ký tự đại diện	Ý nghĩa
%	Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự
_	Ký tự đơn bất kỳ
[]	Ký tự đơn bất kỳ trong giới hạn được chỉ định (ví dụ [a-f]) hay một tập (ví dụ [abcdef])
[^]	Ký tự đơn bất kỳ không nằm trong giới hạn được chỉ định (ví dụ [^a-f] hay một tập (ví dụ [^abcdef])).

Ví dụ: Câu lệnh dưới đây

```
SELECT hodem,ten FROM sinhvien WHERE hodem LIKE 'Lê%'
```

Câu lệnh:

```
SELECT hodem,ten FROM sinhvien WHERE hodem LIKE 'Lê%' AND ten LIKE '[AB]%'
```

❑ Giá trị NULL

Dữ liệu trong một cột cho phép NULL sẽ nhận giá trị NULL trong các trường hợp sau:

Nếu không có dữ liệu được nhập cho cột và không có mặc định cho cột hay kiểu dữ liệu trên cột đó.

Người sử dụng trực tiếp đưa giá trị NULL vào cho cột đó.

Một cột có kiểu dữ liệu là kiểu số sẽ chứa giá trị NULL nếu giá trị được chỉ định gây tràn số.

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không, ta sử dụng cách viết:

WHERE tên_cột **IS NULL**

hoặc:

WHERE tên_cột **IS NOT NULL**

3.1.3.2.3 Sắp xếp kết quả truy vấn

Mặc định, các dòng dữ liệu trong kết quả của câu truy vấn tuân theo thứ tự của chúng trong bảng dữ liệu hoặc được sắp xếp theo chỉ mục (nếu trên bảng có chỉ mục). Trong trường hợp muốn dữ liệu được sắp xếp theo chiều tăng hoặc giảm của giá trị của một hoặc nhiều trường, ta sử dụng thêm mệnh đề ORDER BY trong câu lệnh SELECT; Sau **ORDER BY** là danh sách các cột cần sắp xếp (tối đa là 16 cột). Dữ liệu được sắp xếp có thể theo chiều tăng (**ASC**) hoặc giảm (**DESC**), mặc định là sắp xếp theo chiều tăng.

Ví dụ: Câu lệnh dưới đây hiển thị danh sách các môn học và sắp xếp theo chiều giảm dần của số đơn vị học trình

SELECT * FROM monhoc ORDER BY sodvht DESC

Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo thứ tự từ trái qua phải.

Ví dụ: Câu lệnh

SELECT hodem,ten,gioitinh, **YEAR(GETDATE())-YEAR(ngaysinh)** **AS** tuoi
FROM sinhvien
WHERE ten='Bình'
ORDER BY gioitinh,tuoi

3.1.3.3 Phép nối

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép nối. Một câu lệnh nối kết hợp các dòng dữ liệu trong các bảng khác nhau lại theo một hoặc nhiều điều kiện nào đó và hiển thị chúng trong kết quả truy vấn.

Xét hai bảng sau đây:

Bảng KHOA

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
DHT04	Khoa Hoá học	054823951
DHT05	Khoa Sinh học	054822934
DHT06	Khoa Địa lý - Địa chất	054823837

Bảng LOP

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHT01
C24102	Tin K24	24	Chính quy	2000	DHT02
C24103	Lý K24	24	Chính quy	2000	DHT03
C24301	Sinh K24	24	Chính quy	2000	DHT05
C25101	Toán K25	25	Chính quy	2001	DHT01
C25102	Tin K25	25	Chính quy	2001	DHT02
C25103	Lý K25	25	Chính quy	2001	DHT03
C25301	Sinh K25	25	Chính quy	2001	DHT05
C26101	Toán K26	26	Chính quy	2002	DHT01
C26102	Tin K26	26	Chính quy	2002	DHT02

Các bảng chuẩn bị ghép nối

Giả sử ta cần biết mã lớp và tên lớp của các lớp thuộc Khoa Công nghệ Thông tin, ta phải làm như sau:

- ✓ Chọn ra dòng trong bảng KHOA có tên khoa là Khoa Công nghệ Thông tin, từ đó xác định được mã khoa (MAKHOA) là DHT02.
- ✓ Tìm kiếm trong bảng LOP những dòng có giá trị trường MAKHOA là DHT02 (tức là bằng MAKHOA tương ứng trong bảng KHOA) và đưa những dòng này vào kết quả truy vấn

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
DHT04	Khoa Hóa học	054823951
DHT05	Khoa Sinh học	054822934
DHT06	Khoa Địa lý - Địa chất	054823837

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHT01
C24102	Tin K24	24	Chính quy	2000	DHT02
C24103	Lý K24	24	Chính quy	2000	DHT03
C24301	Sinh K24	24	Chính quy	2000	DHT05
C25101	Toán K25	25	Chính quy	2001	DHT01
C25102	Tin K25	25	Chính quy	2001	DHT02
C25103	Lý K25	25	Chính quy	2001	DHT03
C25301	Sinh K25	25	Chính quy	2001	DHT05
C26101	Toán K26	26	Chính quy	2002	DHT01
C26102	Tin K26	26	Chính quy	2002	DHT02

MALOP	TENLOP
C24102	Tin K24
C25102	Tin K25
C26102	Tin K26

Mô tả dữ liệu kết nối

Như vậy, để thực hiện được yêu cầu truy vấn dữ liệu trên, ta phải thực hiện phép nối giữa hai bảng KHOA và LOP với điều kiện nối là MAKHOA của KHOA bằng với MAKHOA của LOP.

❑ INNER JOIN

Điều kiện để thực hiện phép nối trong được chỉ định trong mệnh đề FROM theo cú pháp như sau:

tên_bảng_1 **[INNER] JOIN** tên_bảng_2 **ON** điều_kiện_nối

Ví dụ: Hiển thị họ tên và ngày sinh của các sinh viên lớp Tin K24:

```
SELECT hodem,ten,ngaysinh
FROM sinhvien INNER JOIN lop ON sinhvien.malop=lop.malop
WHERE tenlop='Tin K24'
```

❑ OUTER JOIN

SQL cung cấp các phép nối ngoài sau đây:

- ✓ Phép nối ngoài trái (LEFT OUTER JOIN)
- ✓ Phép nối ngoài phải (RIGHT OUTER JOIN)
- ✓ Phép nối ngoài đầy đủ (FULL OUTER JOIN)

Cũng tương tự như phép nối trong, điều kiện của phép nối ngoài cũng được chỉ định ngay trong mệnh đề FROM theo cú pháp:

tên_bảng_1 **LEFT | RIGHT | FULL [OUTER] JOIN** tên_bảng_2 **ON** điều_kiện_nối

Ví dụ: Giả sử ta có hai bảng dữ liệu như sau:

Bảng DONVI		Bảng NHANVIEN	
MADV	TENDV	HOTEN	MADV
1	Doi ngoai	Thanh	1
2	Hanh chinh	Hoa	2
3	Ke toan	Nam	2
4	Kinh doanh	Vinh	1
		Hung	5
		Phuong	NULL

Phép nối ngoài trái giữa hai bảng NHANVIEN và DONVI được biểu diễn bởi câu lệnh:

SELECT * FROM nhanvien **LEFT OUTER JOIN** donvi **ON** nhanvien.madv=donvi.madv

có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL

Dữ liệu kết nối ngoài, trái

Câu lệnh:

SELECT * FROM nhanvien **RIGHT OUTER JOIN** donvi **ON** nhanvien.madv=donvi.madv

thực hiện phép nối ngoài phải giữa hai bảng NHANVIEN và DONVI, và có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Vinh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
NULL	NULL	3	Ke toan
NULL	NULL	4	Kinh doanh

Dữ liệu kết nối ngoài, phải

Nếu phép nối ngoài trái (tương ứng phải) hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của bảng bên trái (tương ứng phải) trong phép nối thì phép nối ngoài đầy đủ hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của cả hai bảng tham gia vào phép nối.

Ví dụ: Với hai bảng NHANVIEN và DONVI như ở trên, câu lệnh

SELECT * FROM nhanvien **FULL OUTER JOIN** donvi **ON** nhanvien.madv=donvi.madv

cho kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL
NULL	NULL	4	Kinh doanh
NULL	NULL	3	Ke toan

Dữ liệu kết nối ngoài, đầy đủ

3.1.3.4 Thống kê dữ liệu với GROUP BY

Ngoài khả năng thực hiện các yêu cầu truy vấn dữ liệu thông thường (chiều, chọn, nối,...) như đã đề cập như ở các phần trước, câu lệnh SELECT còn cho phép thực hiện các thao tác truy vấn và tính toán thống kê trên dữ liệu như: cho biết tổng số tiết dạy của mỗi giáo viên, điểm trung bình các môn học của mỗi sinh viên,...

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu, và trên mỗi nhóm dữ liệu thực hiện tính toán các giá trị thống kê như tính tổng, tính giá trị trung bình,...

Các hàm gộp được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như là các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được phép xuất hiện trong mệnh đề WHERE

SQL cung cấp các hàm gộp dưới đây:

Hàm gộp	Chức năng
SUM([ALL DISTINCT] biểu_thức)	Tính tổng các giá trị.
AVG([ALL DISTINCT] biểu_thức)	Tính trung bình của các giá trị
COUNT([ALL DISTINCT] biểu_thức)	Đếm số các giá trị trong biểu thức.
COUNT(*)	Đếm số các dòng được chọn.
MAX(biểu_thức)	Tính giá trị lớn nhất
MIN(biểu_thức)	Tính giá trị nhỏ nhất

Trong đó:

- ✓ Hàm SUM và AVG chỉ làm việc với các biểu thức số.

- ✓ Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.
- ✓ Hàm COUNT(*) không bỏ qua các giá trị NULL.

Mặc định, các hàm gộp thực hiện tính toán thống kê trên toàn bộ dữ liệu. Trong trường hợp cần loại bỏ bớt các giá trị trùng nhau (chỉ giữ lại một giá trị), ta chỉ định thêm từ khóa DISTINCT ở trước biểu thức là đối số của hàm.

Thống kê trên toàn bộ dữ liệu

Khi cần tính toán giá trị thống kê trên toàn bộ dữ liệu, ta sử dụng các hàm gộp trong danh sách chọn của câu lệnh SELECT. Trong trường hợp này, trong danh sách chọn không được sử dụng bất kỳ một tên cột hay biểu thức nào ngoài các hàm gộp.

Ví dụ: Để thống kê trung bình điểm lần 1 của tất cả các môn học, ta sử dụng câu lệnh như sau:

```
SELECT AVG(diemlan1) FROM diemthi
```

còn câu lệnh dưới đây cho biết tuổi lớn nhất, tuổi nhỏ nhất và độ tuổi trung bình của tất cả các sinh viên sinh tại Huế:

```
SELECT MAX(YEAR(GETDATE())-YEAR(ngaysinh)),  
MIN(YEAR(GETDATE())-YEAR(ngaysinh)),  
AVG(YEAR(GETDATE())-YEAR(ngaysinh))  
FROM sinhvien  
WHERE noisinh='Huế'
```

Thống kê dữ liệu trên các nhóm

Trong trường hợp cần thực hiện tính toán các giá trị thống kê trên các nhóm dữ liệu, ta sử dụng mệnh đề GROUP BY để phân hoạch dữ liệu vào trong các nhóm. Các hàm gộp được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo các nhóm dữ liệu.

Ví dụ: Câu lệnh dưới đây cho biết sĩ số (số lượng sinh viên) của mỗi lớp

```
SELECT lop.malop, tenlop, COUNT(masv) AS siso  
FROM lop, sinhvien  
WHERE lop.malop=sinhvien.malop  
GROUP BY lop.malop, tenlop
```

và có kết quả là

MALOP	TENLOP	SISO
C24101	Toán K24	5
C24102	Tin K24	8
C24103	Lý K24	7
C24301	Sinh K24	5
C25101	Toán K25	5
C25102	Tin K25	6
C25103	Lý K25	6
C25301	Sinh K25	8
C26101	Toán K26	5
C26102	Tin K26	5

Kết quả thống kê – đếm

còn câu lệnh:

```
SELECT sinhvien.masv,hodem,ten, sum(diemlan1*sodvht)/sum(sodvht)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY sinhvien.masv,hodem,ten
```

cho biết trung bình điểm thi lần 1 các môn học của các sinh viên

Lưu ý: Trong trường hợp danh sách chọn của câu lệnh SELECT có cả các hàm gộp và những biểu thức không phải là hàm gộp thì những biểu thức này phải có mặt đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Ví dụ: Dưới đây là một câu lệnh sai

```
SELECT lop.malop,tenlop, COUNT(masv) FROM lop,sinhvien
WHERE lop.malop=sinhvien.malop
GROUP BY lop.malop
```

do thiếu trường TENLOP sau mệnh đề GROUP BY.

Chỉ định điều kiện đối với hàm gộp

Mệnh đề HAVING được sử dụng nhằm chỉ định điều kiện đối với các giá trị thống kê được sản sinh từ các hàm gộp tương tự như cách thức mệnh đề WHERE thiết lập các điều kiện cho câu lệnh SELECT. Mệnh đề HAVING thường không thực sự có nghĩa nếu như không sử dụng kết hợp với mệnh đề GROUP BY. Một điểm khác biệt giữa HAVING và WHERE là trong điều kiện của WHERE không được có các hàm gộp trong khi HAVING lại cho phép sử dụng các hàm gộp trong điều kiện của mình.

Ví dụ: Để biết trung bình điểm thi lần 1 của các sinh viên có điểm trung bình lớn hơn hoặc bằng 5, ta sử dụng câu lệnh như sau:

```
SELECT sinhvien.masv,hodem,ten, SUM(diemlan1*sodvht)/sum(sodvht)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND diemthi.mamonhoc=monhoc.mamonhoc
```

GROUP BY sinhvien.masv,hodem,ten

HAVING sum(diemlan1*sodvht)/sum(sodvht)>=5

3.1.4 Các hàm thường dùng

Mặc dù trong SQL chuẩn không cung cấp cụ thể các nhưng trong các hệ quản trị cơ sở dữ liệu luôn cung cấp cho người sử dụng các hàm cài sẵn (hay còn gọi là các hàm của hệ thống). Trong phần này, chúng tôi cung cấp một số hàm thường được sử dụng trong SQL Server để tiện cho việc tra cứu và sử dụng trong thực hành

3.1.4.1 Hàm xử lý chuỗi

Hàm	Mô tả
ASCII(string)	Hàm trả về mã ASCII của ký tự đầu tiên bên trái của chuỗi đối số
CHAR(ascii_code)	Hàm trả về ký tự có mã ASCII tương ứng với đối số
CHARINDEX(string1,string2[,start])	Hàm trả về vị trí đầu tiên tính từ vị trí start tại đó chuỗi string1 xuất hiện trong chuỗi string2.
LEFT(string,number)	Hàm trích ra number ký tự từ chuỗi string tính từ phía bên trái
LEN(string)	Hàm trả về độ dài của chuỗi string.
LOWER(string)	Hàm có chức năng chuyển chuỗi string thành chữ thường, kết quả được trả về cho hàm
LTRIM(string)	Cắt bỏ các khoảng trắng thừa bên trái chuỗi string
NCHAR(code_number)	Hàm trả về ký tự UNICODE có mã được chỉ định
REPLACE(string1,string2,string3)	Hàm trả về một chuỗi có được bằng cách thay thế các chuỗi string2 trong chuỗi string1 bởi chuỗi string3.
REVERSE(string)	Hàm trả về chuỗi đảo ngược của chuỗi string.
RIGHT(string, number)	Hàm trích ra number ký tự từ chuỗi string tính từ phía bên phải.
RTRIM(string)	Cắt bỏ các khoảng trắng thừa bên phải của chuỗi string.
SPACE(number)	Hàm trả về một chuỗi với number khoảng trắng.
STR(number [,length [,decimal]])	Chuyển giá trị kiểu số number thành chuỗi
SUBSTRING(string, m, n)	Trích ra từ n ký tự từ chuỗi string bắt đầu từ ký tự thứ m.
UNICODE(UnicodeString)	Hàm trả về mã UNICODE của ký tự đầu tiên bên trái của chuỗi UnicodeString.

UPPER(string)	Chuyển chuỗi string thành chữ hoa
---------------	-----------------------------------

3.1.4.2 Hàm xử lý ngày giờ

Hàm	Mô tả
DATEADD(datepart, number, date)	Hàm trả về một giá trị kiểu DateTime bằng cách cộng thêm một khoảng giá trị là number vào date.
DATEDIFF(datepart, startdate, enddate)	Hàm trả về khoảng thời gian giữa hai giá trị kiểu này được chỉ định tùy thuộc vào tham số datepart
DATEPART(datepart, date)	Hàm trả về một số nguyên được trích ra từ thành phần (được chỉ định bởi tham số partdate) trong giá trị kiểu ngày được chỉ định.
GETDATE()	Hàm trả về ngày hiện tại
DAY(date) MONTH(date) YEAR(date) HOUR(date) MINUTE(date) SECOND(date)	Hàm trả về giá trị ngày (tháng hoặc năm) của giá trị kiểu ngày được chỉ định.

Trong đó, datepart là tham số chỉ định thành phần sẽ được cộng đối với giá trị date bao gồm:

Datepart	Viết tắt	Ý nghĩa
Year	yy, yyyy	Năm 2, 4 chữ số
Quarter	q, qq	Quý 1, 2 chữ số
Month	m, mm	Tháng 1, 2 chữ số
Day	d, dd	Ngày 1, 2 chữ số
Week	wk, ww	Tuần 1, 2 chữ số
Hour	hh	Giờ 2 chữ số
Minute	mi, n	Phút 1, 2 chữ số
Second	s, ss	Giây 1, 2 chữ số
Millisecond	ms	Milli giây

3.1.4.3 Hàm chuyển kiểu

Hàm	Mô tả
CAST (biểu_thức AS kiểu_dữ_liệu)	Chuyển đổi giá trị của biểu thức sang kiểu được chỉ định
CONVERT(kiểu_dữ_liệu, biểu_thức)	Hàm có chức năng chuyển đổi giá trị của biểu thức sang kiểu dữ liệu được chỉ định.

3.1.5 SQL nâng cao

Trong phần nâng cao, bạn sẽ được biết thêm cách định nghĩa và sử dụng các thành phần quan trọng khác trong CSDL quan hệ:

- ✓ View
- ✓ Stored Procedure
- ✓ Function
- ✓ Trigger
- ✓ Transaction

3.1.5.1 View

Nhiều câu lệnh SELECT rất dài và phức tạp khi sử dụng. Để đơn giản hóa vấn đề này người ta đặt cho câu lệnh SELECT một cái tên để được tái sử dụng nhiều lần. Khi sử dụng, người dùng chỉ việc sử dụng cái tên đơn giản ấy đại diện cho một khối câu lệnh SELECT phức tạp.

❑ Tạo View

Câu lệnh CREATE VIEW được sử dụng để tạo ra khung nhìn và có cú pháp như sau:

```
CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)]
AS
câu_lệnh_SELECT
```

Ví dụ: Câu lệnh dưới đây tạo khung nhìn có tên DSSV từ câu lệnh SELECT truy vấn dữ liệu từ hai bảng SINHVIEN và LOP

```
CREATE VIEW dssv
AS
SELECT masv,hodem,ten,
DATEDIFF(YY,ngaysinh,GETDATE()) AS tuoi,tenlop
FROM sinhvien,lop
WHERE sinhvien.malop=lop.malop
```

❑ Sử dụng View

```
SELECT * FROM dssv
```

❑ Sửa khung nhìn

```
ALTER VIEW dssv
```

AS

<câu lệnh SELECT mới>

❑ Xóa khung nhìn

DROP VIEW dssv

3.1.5.2 Stored procedure

Việc mô-đun hóa để tái sử dụng là rất quan trọng. Trong SQL chúng ta cũng thực hiện việc tạo ra các thủ tục nhận vào một số tham số, thực hiện công việc và trả về các kết quả như trong các ngôn ngữ lập trình. Thông thường stored procedure được tạo ra để tận dụng một số đặc điểm sau:

- ✓ Đơn giản hóa việc sử dụng câu lệnh
- ✓ Tái sử dụng nhiều lần
- ✓ Tăng tốc đồng xử lý vì khối lệnh được viết và chạy trong lòng CSDL
- ✓ Tận dụng khả năng lập trình của SQL (IF, WHILE, FETCH...)

❑ Tạo Stored Procedure

Thủ tục lưu trữ được tạo bởi câu lệnh CREATE PROCEDURE với cú pháp như sau:

CREATE PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]

AS

Các_câu_lệnh_của_thủ_tục

Ví dụ: Giả sử ta cần thực hiện một chuỗi các thao tác như sau trên cơ sở dữ liệu

1. Bổ sung thêm môn học cơ sở dữ liệu có mã TI-005 và số đơn vị học trình là 5 vào bảng MONHOC
2. Lên danh sách nhập điểm thi môn cơ sở dữ liệu cho các sinh viên học lớp có mã C24102(tức là bổ sung thêm vào bảng DIEMTHI các bản ghi với cột MAMONHOC nhận giá trị TI-005, cột MASV nhận giá trị lần lượt là mã các sinh viên học lớp có mã C24105 và các cột điểm là NULL).

Định nghĩa stored procedure

CREATE PROC sp_LenDanhSachDiem

(

@mamonhoc NVARCHAR(10),
@tenmonhoc NVARCHAR(50),
@sodvht SMALLINT,
@malop NVARCHAR(10)

)

AS

BEGIN

INSERT INTO monhoc

VALUES(@mamonhoc,@tenmonhoc,@sodvht)

INSERT INTO diemthi(mamonhoc,masv)

SELECT @mamonhoc,masv FROM sinhvien WHERE malop=@malop

END

Sau đây là ví dụ về 1 sp phân trang dữ liệu thường được sử dụng để hạn chế truy xuất nhiều gây nghẽn bộ nhớ

```
CREATE PROC spPhanTrang
(
    @Start INT,
    @Length INT
)
AS
BEGIN
    WITH Temp AS
    (
        SELECT *, ROW_NUMBER() OVER (ORDER BY MaHH) AS STT FROM
        HangHoa
    )
    SELECT * FROM Temp WHERE STT BETWEEN @Start AND @Start+@Length-1
END
```

❑ Sử dụng stored procedure

EXECUTE tên_thủ_tục [danh_sách_các_đối_số]

Thứ tự của các đối số được truyền cho thủ tục có thể không cần phải tuân theo thứ tự của các tham số như khi định nghĩa thủ tục nếu tất cả các đối số được viết dưới dạng:

@tên_tham_số = giá_trị

Ví dụ: Lời gọi thủ tục ở ví dụ trên có thể viết như sau:

```
EXEC sp_LenDanhSachDiem
    @malop='C24102',
    @tenmonhoc='Cơ sở dữ liệu',
    @mamonhoc='TI-005',
    @sodvht=5
```

❑ Biến trong stored procedure

Ngoài những tham số được truyền cho thủ tục, bên trong thủ tục còn có thể sử dụng các biến nhằm lưu giữ các giá trị tính toán được hoặc truy xuất được từ cơ sở dữ liệu. Các biến trong thủ tục được khai báo bằng từ khoá DECLARE theo cú pháp như sau:

DECLARE @tên_biến kiểu_dữ_liệu

Tên biến phải bắt đầu bởi ký tự @ và tuân theo qui tắc về định danh. Ví dụ dưới

đây minh họa việc sử dụng biến trong thủ tục

Ví dụ: Trong định nghĩa của thủ tục dưới đây sử dụng các biến chứa các giá trị truy xuất được từ cơ sở dữ liệu.

```
CREATE PROCEDURE sp_Vidu
(
    @malop1 NVARCHAR(10),
    @malop2 NVARCHAR(10)
)
AS
BEGIN
```




```

DECLARE @tenlop1 NVARCHAR(30)
DECLARE @namnhaphoc1 INT
DECLARE @tenlop2 NVARCHAR(30)
DECLARE @namnhaphoc2 INT

SELECT @tenlop1=tenlop,@namnhaphoc1=namnhaphoc
FROM lop WHERE malop=@malop1
SELECT @tenlop2=tenlop,@namnhaphoc2=namnhaphoc
FROM lop WHERE malop=@malop2
PRINT @tenlop1+' nhập học nam '+str(@namnhaphoc1)
print @tenlop2+' nhập học nam '+str(@namnhaphoc2)
IF @namnhaphoc1=@namnhaphoc2
    PRINT 'Hai lớp nhập học cùng năm'
ELSE
    PRINT 'Hai lớp nhập học khác năm'

END

```

❑ Tham số OUTPUT của stored procedure

Trong các ví dụ trước, nếu đổi số truyền cho thủ tục khi có lời gọi đến thủ tục là biến, những thay đổi giá trị của biến trong thủ tục sẽ không được giữ lại khi kết thúc quá trình thực hiện thủ tục.

Ví dụ: Xét câu lệnh sau đây

```

CREATE PROCEDURE sp_Conghaiso(@a INT,@b INT, @c INT OUTPUT)
AS
BEGIN
    SELECT @c=@a+@b
END

```

Nếu sau khi đã tạo thủ tục với câu lệnh trên, ta thực thi một tập các câu lệnh như sau:

```

DECLARE @tong INT
EXECUTE sp_Conghaiso 100, 200, @tong OUTPUT
SELECT @tong

```

thì câu lệnh **"SELECT @tong"** sẽ cho kết quả là: 300

❑ Tham số với giá trị mặc định

Các tham số được khai báo trong thủ tục có thể nhận các giá trị mặc định. Giá trị mặc định sẽ được gán cho tham số trong trường hợp không truyền đổi số cho tham số khi có lời gọi đến thủ tục.

Tham số với giá trị mặc định được khai báo theo cú pháp như sau:

```
@tên_tham_số kiểu_dữ_liệu = giá_trị_mặc_định
```

Ví dụ: Trong câu lệnh dưới đây:

```

CREATE PROC sp_TestDefault
(
    @tenlop NVARCHAR(30)=NULL,
    @noisinh NVARCHAR(100)='Huê'
)

```

```

AS
BEGIN
    IF @tenlop IS NULL
        SELECT hodem,ten
        FROM sinhvien INNER JOIN lop ON sinhvien.malop=lop.malop
        WHERE noisinh=@noisinh
    ELSE
        SELECT hodem,ten
        FROM sinhvien INNER JOIN lop ON sinhvien.malop=lop.malop
        WHERE noisinh=@noisinh AND tenlop=@tenlop
END

```

thủ tục sp_TestDefault được định nghĩa với tham số @tenlop có giá trị mặc định là NULL và tham số @noisinh có giá trị mặc định là Huế. Với thủ tục được định nghĩa như trên, ta có thể thực hiện các lời gọi với các mục đích khác nhau như sau:

- ✓ Cho biết họ tên của các sinh viên sinh tại Huế:

```
EXEC sp_testdefault
```

- ✓ Cho biết họ tên của các sinh viên lớp Tin K24 sinh tại Huế:

```
EXEC sp_testdefault @tenlop='Tin K24'
```

❑ Sửa đổi stored procedure

Khi một thủ tục đã được tạo ra, ta có thể tiến hành định nghĩa lại thủ tục đó bằng câu lệnh ALTER PROCEDURE có cú pháp như sau:

```

ALTER PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]
AS
Các_câu_lệnh_Của_thủ_tục

```

Câu lệnh này sử dụng tương tự như câu lệnh CREATE PROCEDURE. Việc sửa đổi lại một thủ tục đã có không làm thay đổi đến các quyền đã cấp phát trên thủ tục cũng như không tác động đến các thủ tục khác hay trigger phụ thuộc vào thủ tục này.

❑ Xoá thủ tục

Để xoá một thủ tục đã có, ta sử dụng câu lệnh DROP PROCEDURE với cú pháp như sau:

```
DROP PROCEDURE tên_thủ_tục
```

Khi xoá một thủ tục, tất cả các quyền đã cấp cho người sử dụng trên thủ tục đó cũng đồng thời bị xoá bỏ. Do đó, nếu tạo lại thủ tục, ta phải tiến hành cấp phát lại các quyền trên thủ tục đó

3.1.5.3 User-defined function

Hàm là đối tượng cơ sở dữ liệu tương tự như thủ tục. Điểm khác biệt giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm còn thủ tục thì không. Điều này cho phép ta sử dụng hàm như là một thành phần của một biểu thức (chẳng hạn trong danh sách chọn của câu lệnh SELECT).

Ngoài những hàm do hệ quản trị cơ sở dữ liệu cung cấp sẵn, người sử dụng có thể định nghĩa thêm các hàm nhằm phục vụ cho mục đích riêng của mình.

❑ Tạo Function

Hàm được định nghĩa thông qua câu lệnh CREATE FUNCTION với cú pháp như sau:

```
CREATE FUNCTION tên_hàm ([danh_sách_tham_số])
RETURNS (kiểu_trả_về_của_hàm)
AS BEGIN
    các_câu_lệnh_của_hàm
END
```

Ví dụ: Câu lệnh dưới đây định nghĩa hàm tính ngày trong tuần (thứ trong tuần) của một giá trị kiểu ngày

```
CREATE FUNCTION thu(@ngay DATETIME)
RETURNS NVARCHAR(10)
AS
BEGIN
    DECLARE @st NVARCHAR(10)
    SELECT @st=CASE DATEPART(DW, @ngay)
        WHEN 1 THEN 'Chu nhật'
        WHEN 2 THEN 'Thứ hai'
        WHEN 3 THEN 'Thứ ba'
        WHEN 4 THEN 'Thứ tư'
        WHEN 5 THEN 'Thứ năm'
        WHEN 6 THEN 'Thứ sáu'
        ELSE 'Thứ bảy'
    END
    RETURN (@st) /* Trị trả về của hàm */
END
```

❑ Sử dụng hàm

Một hàm khi đã được định nghĩa có thể được sử dụng như các hàm do hệ quản trị cơ sở dữ liệu cung cấp (thông thường trước tên hàm ta phải chỉ định thêm tên của người sở hữu hàm)

Ví dụ: Câu lệnh SELECT dưới đây sử dụng hàm đã được định nghĩa ở ví dụ trước:

```
SELECT masv, hodem, ten, dbo.thu(ngaysinh), daysinh
FROM sinhvien
WHERE malop='C24102'
```

3.1.5.4 Trigger

Trigger là một thủ tục đặc biệt được định nghĩa gắn liền với một bảng và được thực thi một cách tự động theo các sự kiện thao tác dữ liệu như INSERT, UPDATE và DELETE. Sử dụng trigger một cách hợp lý trong cơ sở dữ liệu sẽ có tác động rất lớn trong việc tăng hiệu năng của cơ sở dữ liệu. Các trigger thực sự hữu dụng với những khả năng sau:

- ✓ Một trigger có thể nhận biết, ngăn chặn và huỷ bỏ được những thao tác làm thay đổi trái phép dữ liệu trong cơ sở dữ liệu.

- ✓ Các thao tác trên dữ liệu (xoá, cập nhật và bổ sung) có thể được trigger phát hiện ra và tự động thực hiện một loạt các thao tác khác trên cơ sở dữ liệu nhằm đảm bảo tính hợp lệ của dữ liệu.
- ✓ Thông qua trigger, ta có thể tạo và kiểm tra được những mối quan hệ phức tạp hơn giữa các bảng trong cơ sở dữ liệu mà bản thân các ràng buộc không thể thực hiện được.

❑ Tạo trigger

Một trigger là một đối tượng gắn liền với một bảng và được tự động kích hoạt khi xảy ra những giao tác làm thay đổi dữ liệu trong bảng. Định nghĩa một trigger bao gồm các yếu tố sau:

- ✓ Trigger sẽ được áp dụng đối với bảng nào?
- ✓ Trigger được kích hoạt khi câu lệnh nào được thực thi trên bảng: INSERT, UPDATE, DELETE?
- ✓ Trigger sẽ làm gì khi được kích hoạt?

Câu lệnh CREATE TRIGGER được sử dụng để định nghĩa trigger và có cú pháp như sau:

```
CREATE TRIGGER tên_trigger
ON tên_bảng
FOR {[INSERT][,][UPDATE][,][DELETE]}
AS
[IF UPDATE(tên_cột) [AND UPDATE(tên_cột) OR UPDATE(tên_cột)]...]
các_câu_lệnh_của_trigger
```

Ví dụ: Ta định nghĩa các bảng như sau:

Bảng MATHANG lưu trữ dữ liệu về các mặt hàng:

```
CREATE TABLE mathang
(
    mahang NVARCHAR(5) PRIMARY KEY, /*mã hàng*/
    tenhang NVARCHAR(50) NOT NULL, /*tên hàng*/
    soluong INT, /*số lượng hàng hiện có*/
)
```

Bảng NHATKYBANHANG lưu trữ thông tin về các lần bán hàng

```
CREATE TABLE nhatkysanhang
(
    stt INT IDENTITY PRIMARY KEY,
    ngay DATETIME, /*ngày bán hàng*/
    nguoi mua NVARCHAR(30), /*tên người mua hàng*/
    mahang NVARCHAR(5) /*mã mặt hàng được bán*/
    FOREIGN KEY REFERENCES mathang(mahang),
    soluong INT, /*giá bán hàng*/
    giamban MONEY /*số lượng hàng được bán*/
)
```

Câu lệnh dưới đây định nghĩa trigger trg_nhatkysanhang_insert. Trigger này có chức năng tự động giảm số lượng hàng hiện có khi một mặt hàng nào đó được bán (tức là khi câu lệnh INSERT được thực thi trên bảng NHATKYBANHANG).



```
CREATE TRIGGER trg_nhatkybanhang_insert ON nhatkybanhang
FOR INSERT
AS
BEGIN
    UPDATE mathang
    SET mathang.soluong=mathang.soluong-inserted.soluong
    FROM mathang INNER JOIN inserted ON mathang.mahang=inserted.mahang
END
```

Với trigger vừa tạo ở trên, nếu dữ liệu trong bảng MATHANG là:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	20
H2	Kem đánh răng	45

Dữ liệu trước khi thao tác

thì sau khi ta thực hiện câu lệnh:

```
INSERT INTO nhatkybanhang (ngay,nguoiimua,mahang,soluong,giaban)
VALUES('5/5/2004','Tran Ngoc Thanh','H1',10,5200)
```

dữ liệu trong bảng MATHANG sẽ như sau:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	30
H2	Kem đánh răng	45

Dữ liệu bị cập nhật bởi trigger

❑ Các bảng đặc biệt trong trigger

Chuẩn SQL định nghĩa hai bảng logic **INSERTED** (chứa các bản ghi mới được chèn vào) và **DELETED** (chứa các bản ghi sẽ bị xóa) để sử dụng trong các trigger. Cấu trúc của hai bảng này tương tự như cấu trúc của bảng mà trigger tác động. Dữ liệu trong hai bảng này tùy thuộc vào câu lệnh tác động lên bảng làm kích hoạt trigger.

❑ Hàm UPDATE(cột) trong trigger

Hàm này cho biết một cột nào đó có bị cập nhật hay không.

Ví dụ 2.56: Xét lại ví dụ với hai bảng MATHANG và NHATKYBANHANG, trigger dưới đây được kích hoạt khi ta tiến hành cập nhật cột SOLUONG cho một bản ghi của bảng NHATKYBANHANG (lưu ý là chỉ cập nhật đúng một bản ghi)

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong ON nhatkybanhang
FOR UPDATE
AS
BEGIN
    IF UPDATE(soluong)
        UPDATE mathang
        SET mathang.soluong = mathang.soluong - (inserted.soluong-
        deleted.soluong)
        FROM (deleted INNER JOIN inserted ON deleted.stt = inserted.stt)
        INNER JOIN mathang ON mathang.mahang = deleted.mahang
END
```

❑ ROLLBACK TRANSACTION và trigger

Chúng ta có thể hủy thao tác dữ liệu của người dùng bằng cách gọi lệnh

ROLLBACK TRANSACTION

Ví dụ: Nếu trên bảng MATHANG, ta tạo một trigger như sau:

```
CREATE TRIGGER trg_mathang_delete ON mathang
FOR DELETE
AS
BEGIN
    ROLLBACK TRANSACTION
END
```

3.1.5.5 Transaction

Khi bạn làm việc với CSDL trong tình huống bạn muốn tất cả các câu lệnh của bạn phải được thành công thì mới chấp nhận kết quả thực hiện của cả một khối lệnh còn trong trường hợp ngược lại chỉ cần một trong số các câu lệnh trên có sự cố thì bạn sẽ hủy toàn bộ những lệnh đã thực hiện trước đó. Bạn sẽ làm thế nào để đạt được kết quả đó ? Không phải làm gì cả, chỉ theo dõi thật kỹ bài học này.

```
BEGIN TRAN TRS_KHOA;
BEGIN TRY
    INSERT INTO Khoa(MaKhoa, TenKhoa, DienThoai) VALUES('K0001', 'Kinh Te',
'0913745789');
    INSERT INTO Khoa(MaKhoa, TenKhoa, DienThoai) VALUES('K0002', 'CNTT',
'0918355888');
    INSERT INTO Khoa(MaKhoa, TenKhoa, DienThoai) VALUES('K0003', 'Dien Tu',
'0917060606');
    INSERT INTO Khoa(MaKhoa, TenKhoa, DienThoai) VALUES('K0001', 'Hoa', '0918696969');
    INSERT INTO Khoa(MaKhoa, TenKhoa, DienThoai) VALUES('K0005', 'Nano',
'0987456789');
    COMMIT TRAN TRS_KHOA;
END TRY
BEGIN CATCH
    ROLLBACK TRAN TRS_KHOA;
END CATCH
```


Đoạn mã trên thực hiện 5 lệnh bổ sung dữ liệu vào bảng Khoa. Bạn cũng thấy rõ, khi chạy thì lệnh thứ tư sẽ gặp lỗi do trùng khóa. Nhưng trong trường hợp này cả 3 lệnh trước đó bị hủy bỏ hoàn toàn vì chúng ta áp dụng transaction ở đây.


- ✓ Dòng: "**BEGIN TRAN** TRS_KHOA;" sẽ đánh dấu bắt đầu 1 transaction.
- ✓ Khối lệnh đặt giữa "**BEGIN TRY**" và "**END TRY**" sẽ được kiểm tra lỗi.
 - Nếu cả 5 lệnh INSERT thực hiện thành công thì chương trình sẽ gặp lệnh "**COMMIT TRAN** TRS_KHOA;" lệnh này sẽ ra lệnh chấp nhận toàn bộ các lệnh đã thực hiện trước đó.
- ✓ Nếu một trong các lệnh trên sai, thì chương trình chuyển hướng xuống khối lệnh đặt giữ "**BEGIN CATCH**" và "**END CATCH**" ở đó sẽ gặp lệnh "**ROLLBACK TRAN** TRS_KHOA;" có nghĩa là hủy bỏ các thao tác trước đó.


Và vì vậy với cấu trúc lệnh trên, bạn đã thực hiện được điều mà bạn muốn là **"ĐƯỢC ĂN CÀ, NGẤ VỀ KHÔNG."**

3.1.6 Bài tập

1. Tạo CSDL tên ThucHanh có các bảng với kiểu và ràng buộc như hình sau.

Courses					
Column Name	Condensed Type	Nullable	Identity	Description	
 Id	int	No	<input checked="" type="checkbox"/>	Mã khóa học	
Name	nvarchar(50)	No	<input type="checkbox"/>	Tên khóa học	
Schoolfee	float	No	<input type="checkbox"/>	Học phí	
NoOfLearners	int	No	<input type="checkbox"/>	Số học viên	
StartDate	datetime	No	<input type="checkbox"/>	Ngày khai giảng	
Finished	bit	No	<input type="checkbox"/>	Kết thúc?	

Categories					
Column Name	Condensed Type	Nullable	Identity	Description	
 Id	char(3)	No	<input type="checkbox"/>	Mã loại	
Name	nvarchar(50)	No	<input type="checkbox"/>	Tên loại	
Description	nvarchar(255)	Yes	<input type="checkbox"/>	Mô tả	

Products					
Column Name	Condensed Type	Nullable	Identity	Description	
 Id	int	No	<input checked="" type="checkbox"/>	Mã hàng hóa	
Name	nvarchar(50)	No	<input type="checkbox"/>	Tên hàng hóa	
UnitPrice	float	No	<input type="checkbox"/>	Đơn giá	
ProductDate	datetime	No	<input type="checkbox"/>	Ngày sản xuất	
Image	nvarchar(50)	No	<input type="checkbox"/>	Hình ảnh	
Description	nvarchar(4000)	Yes	<input type="checkbox"/>	Mô tả	
CategoryId	char(3)	No	<input type="checkbox"/>	Mã loại	
Available	bit	No	<input type="checkbox"/>	Vẫn còn hàng	

Các ràng buộc:

- ✓ Course Name phải duy nhất
- ✓ Tham chiếu khóa ngoại Products(CategoryId)->Categories(Id) là ON DELETE CASCADE ON UPDATE CASCADE.

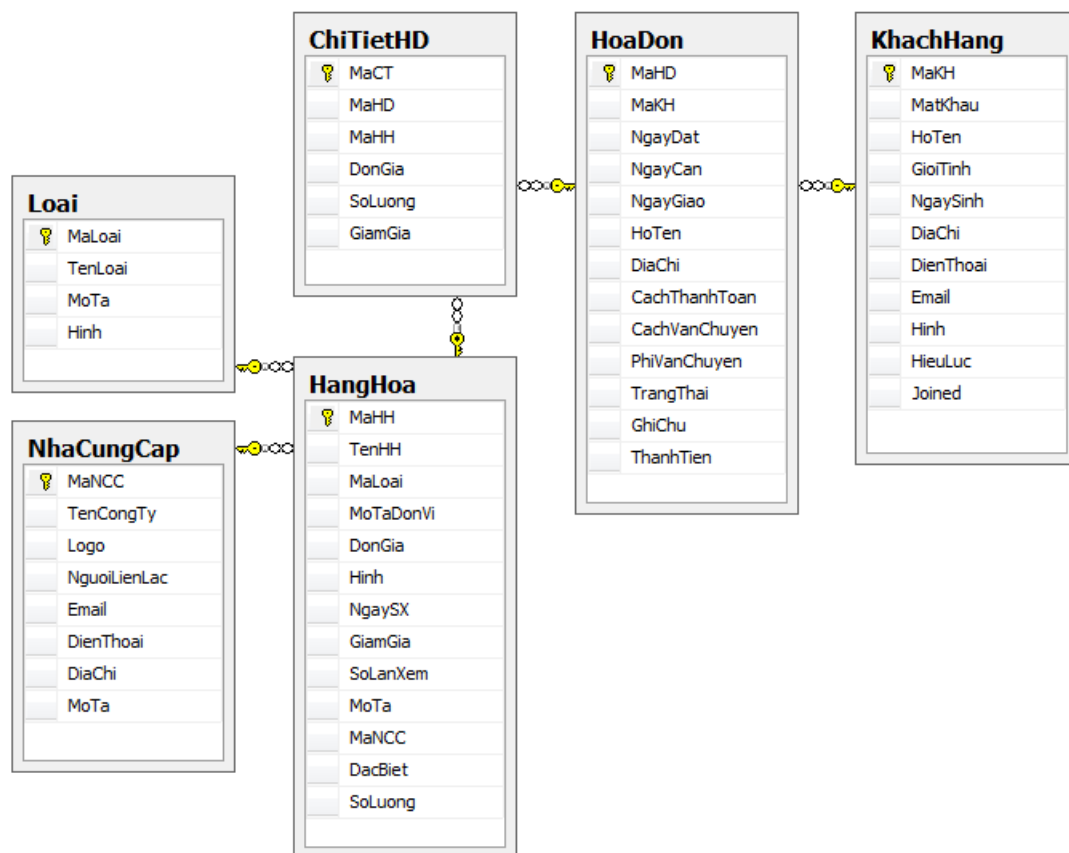
2. Viết các câu lệnh SQL theo yêu cầu sau

- a. Chèn 5 khóa học vào bảng Courses
- b. Hiển thị các khóa học có tên chứa chữ "Java"
- c. Hiển thị các khóa học có học phí từ 10 đến 100
- d. Hiển thị các khóa học khai giảng tháng 10 năm 2012
- e. Hiển thị các khóa học chưa kết thúc và có sĩ số trên 20
- f. Nâng học phí của các lớp có tên chứa PHP chưa kết thúc lên 5%
- g. Xóa các khóa đã kết thúc cách đây 1 năm
- h. Thống kê tổng học phí và số lượng học viên theo từng tháng (tháng, tổng học phí, tổng học viên).
- i. Hiển thị các khóa học từ vị trí thứ 5 đến 12


3. Viết các câu lệnh SQL theo yêu cầu

- a. Chèn 3 loại vào bảng Categories và 5 hàng hóa vào bảng Products
- b. Cập nhật mã loại từ NOK thành MOT và quan sát sự thay đổi của mã loại trên bảng Products
- c. Hiển thị tên loại, tên hàng hóa và đơn giá

- d. Thống kê số lượng các mặt hàng theo từng loại (tên loại và số hàng)
 - e. Hiển thị các mặt hàng có tên loại chứa chữ "Nokia"
 - f. Hiển thị các loại chưa có hàng hóa
 - g. Hiển thị các hàng hóa có tên loại không chứa chữ "Nokia" đang còn hàng
4. Tạo CSDL bán hàng đơn giản như mô tả sau đây



❑ **Bảng HangHoa: lưu thông tin hàng hóa**

HangHoa						
	Column Name	Condensed Type	Nullable	Identity	Default Value	Description
	MaHH	int	No	<input checked="" type="checkbox"/>		Mã hàng hóa
	TenHH	nvarchar(40)	No	<input type="checkbox"/>		Tên hàng hóa
	MaLoai	int	No	<input type="checkbox"/>		Mã loại, FK
	MoTaDonVi	nvarchar(50)	Yes	<input type="checkbox"/>		Mô tả đơn vị tính
	DonGia	float	Yes	<input type="checkbox"/>	((0))	Đơn giá
	Hinh	nvarchar(50)	Yes	<input type="checkbox"/>		Hình ảnh
	NgaySX	datetime	No	<input type="checkbox"/>	(getdate())	Ngày sản xuất
	GiamGia	float	No	<input type="checkbox"/>	((0))	Giảm giá
	SoLanXem	int	No	<input type="checkbox"/>	((0))	Số lượt xem
	MoTa	nvarchar(MAX)	Yes	<input type="checkbox"/>		Mô tả hàng hóa
	MaNCC	nvarchar(50)	No	<input type="checkbox"/>		Mã nhà cung cấp, FK
	DacBiet	bit	No	<input type="checkbox"/>	((0))	Hàng hóa đặc biệt?
	SoLuong	int	No	<input type="checkbox"/>	((100))	Số lượng tồn

❑ **Bảng Loại: lưu thông tin chủng loại hàng hóa**

Loại						
	Column Name	Condensed Type	Nullable	Identity	Default Value	Description
🔑	MaLoai	int	No	<input checked="" type="checkbox"/>		Mã loại
	TenLoai	nvarchar(50)	No	<input type="checkbox"/>		Tên chủng loại
	MoTa	nvarchar(MAX)	Yes	<input type="checkbox"/>		Mô tả chủng loại
	Hinh	nvarchar(50)	Yes	<input type="checkbox"/>		Hình đại diện
				<input type="checkbox"/>		


❑ **Bảng NhaCungCap: lưu thông tin các nhà cung cấp hàng hóa**

NhaCungCap						
	Column Name	Condensed Type	Nullable	Identity	Default Value	Description
🔑	MaNCC	nvarchar(50)	No	<input type="checkbox"/>		Mã nhà cung cấp
	TenCongTy	nvarchar(50)	No	<input type="checkbox"/>		Tên công ty
	Logo	nvarchar(50)	No	<input type="checkbox"/>		Hình logo
	NguoiLienLac	nvarchar(50)	Yes	<input type="checkbox"/>		Người đại diện
	Email	nvarchar(50)	No	<input type="checkbox"/>		Email
	DienThoai	nvarchar(50)	Yes	<input type="checkbox"/>		Điện thoại
	DiaChi	nvarchar(50)	Yes	<input type="checkbox"/>		Địa chỉ
	MoTa	nvarchar(MAX)	Yes	<input type="checkbox"/>		Mô tả thêm
				<input type="checkbox"/>		


❑ **Bảng KháchHang: lưu thông tin khách hàng đăng ký trực tuyến**

KhachHang						
	Column Name	Condensed Type	Nullable	Identity	Default Value	Description
🔑	MaKH	nvarchar(20)	No	<input type="checkbox"/>		Mã khách hàng
	MatKhau	nvarchar(50)	Yes	<input type="checkbox"/>		Mật khẩu đăng nhập
	HoTen	nvarchar(50)	No	<input type="checkbox"/>		Họ và tên
	GioiTinh	bit	No	<input type="checkbox"/>	((0))	Giới tính
	NgaySinh	datetime	No	<input type="checkbox"/>	(getdate())	Ngày sinh
	DiaChi	nvarchar(60)	Yes	<input type="checkbox"/>		Địa chỉ
	DienThoai	nvarchar(24)	Yes	<input type="checkbox"/>		Điện thoại
	Email	nvarchar(50)	No	<input type="checkbox"/>		Email
	Hinh	nvarchar(50)	Yes	<input type="checkbox"/>	(N'Photo.gif')	Hình
	HieuLuc	bit	No	<input type="checkbox"/>	((0))	Đã kích hoạt
				<input type="checkbox"/>		

❑ **Bảng HoaDon: lưu thông tin đơn đặt hàng của khách hàng**

HoaDon						
	Column Name	Condensed Type	Nullable	Identity	Default Value	Description
	MaHD	int	No	<input checked="" type="checkbox"/>		Mã hóa đơn
	MaKH	nvarchar(20)	No	<input type="checkbox"/>		Mã khách hàng, FK
	NgayDat	datetime	No	<input type="checkbox"/>	(getdate())	Ngày đặt hàng
	NgayCan	datetime	No	<input type="checkbox"/>	(getdate())	Ngày căn có hàng
	NgayGiao	datetime	Yes	<input type="checkbox"/>	((1)/(1))/(...	Ngày giao hàng
	HoTen	nvarchar(50)	Yes	<input type="checkbox"/>		Họ tên người nhận
	DiaChi	nvarchar(60)	No	<input type="checkbox"/>		Địa chỉ nhận
	CachThanhToan	nvarchar(50)	No	<input type="checkbox"/>	(N'Cash')	Phương thức thanh toán
	CachVanChuyen	nvarchar(50)	No	<input type="checkbox"/>	(N'Airline')	Phương thức vận chuyển
	PhiVanChuyen	float	No	<input type="checkbox"/>	((0))	Phí vận chuyển
	TrangThai	int	Yes	<input type="checkbox"/>	((0))	Trạng thái hóa đơn
	GhiChu	nvarchar(50)	Yes	<input type="checkbox"/>		Ghi chú thêm
	ThanhTien	float	Yes	<input type="checkbox"/>	((0))	Tổng tiền đơn hàng
				<input type="checkbox"/>		

- **Bảng ChiTietHD:** lưu thông tin chi tiết hóa đơn (các mặt hàng trên từng hóa đơn)

ChiTietHD						
	Column Name	Condensed Type	Nullable	Identity	Default Value	Description
	MaCT	int	No	<input checked="" type="checkbox"/>		Mã tự tăng
	MaHD	int	No	<input type="checkbox"/>		Mã hóa đơn, FK
	MaHH	int	No	<input type="checkbox"/>		Mã hàng hóa, FK
	DonGia	float	No	<input type="checkbox"/>	((0))	Đơn giá bán
	SoLuong	int	No	<input type="checkbox"/>	((1))	Số lượng mua
	GiamGia	float	No	<input type="checkbox"/>	((0))	Giảm giá
				<input type="checkbox"/>		

5. Viết câu lệnh truy vấn

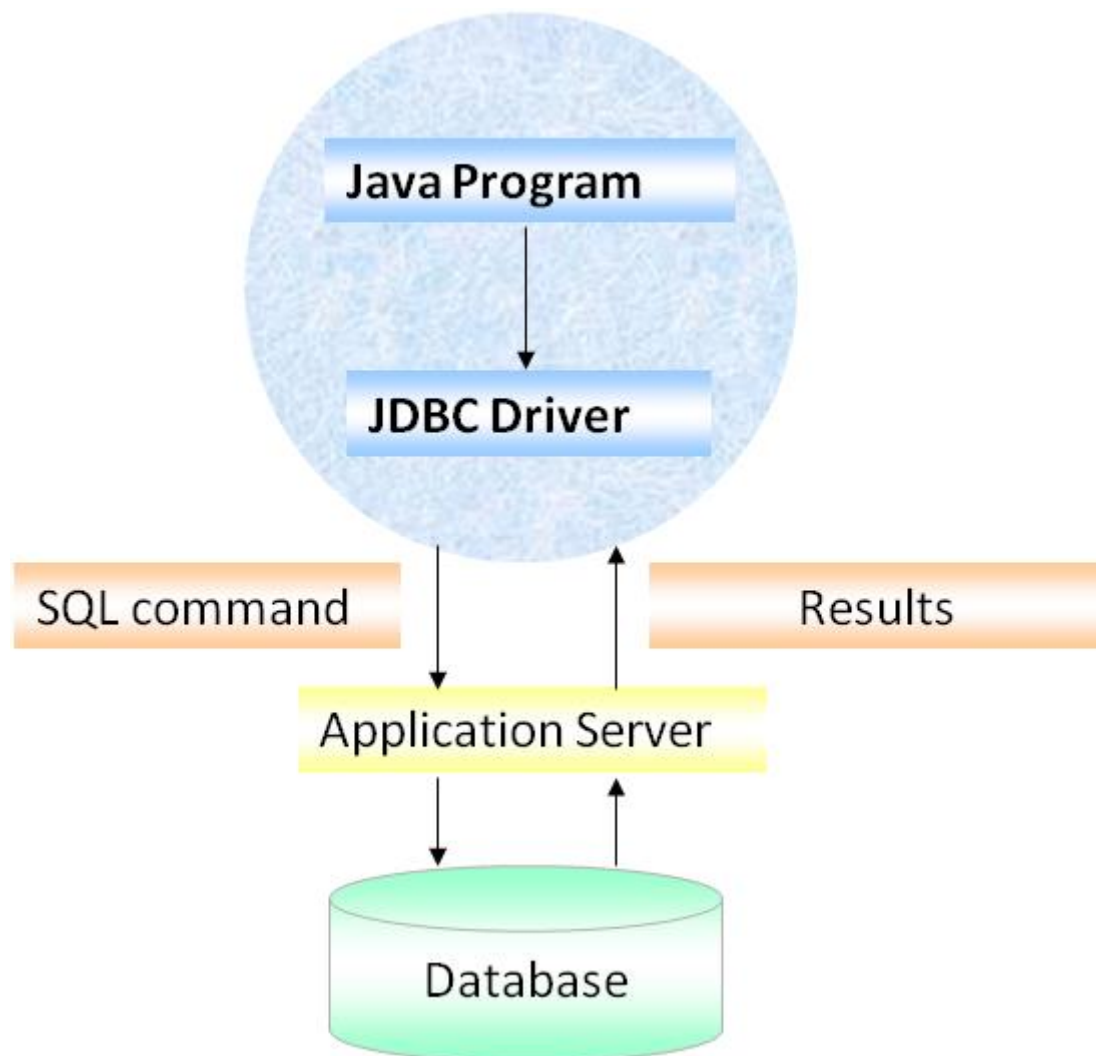
- Hiển thị tên loại, tên công ty cung cấp, tên hàng hóa và đơn giá
- Hiển thị loại có mã số 1001 và các hàng hóa thuộc loại đó (2 SQL)
- Hiển thị khách hàng có mã nngkiem và các hóa đơn của khách hàng (2 SQL)
- Hiển thị hóa đơn có mã số 10249 và các mặt hàng trên hóa đơn đó (2 SQL)
- Thống kê tổng giá trị hàng hóa theo loại (tên loại, số hàng, tổng giá trị)
- Thống kê doanh số bán hàng theo hàng hóa (tên hàng, doanh số)
- Thống kê doanh số bán hàng theo chủng loại (tên loại, doanh số)

- h. Thống kê doanh số bán hàng theo khách hàng(tên khách hàng, doanh số)
- i. Thống kê doanh số bán hàng theo quý (tên quý, doanh số)
- j. Thống kê doanh số bán hàng theo tháng (tên hàng, doanh số) trong năm 2012

3.2 JDBC

Sau khi kết thúc phần NGÔN NGỮ SQL, bạn đã có khả năng làm chủ CSDL từ việc định nghĩa cho đến thao tác dữ liệu. Trong phần này bạn sẽ sử dụng JDBC trong Java để làm việc với CSDL của bạn.

3.2.1 Mô hình ứng JDBC



Mô hình lập trình JDBC

- ✓ Giao tiếp lập trình JDBC gồm 2 layer:
 - **Application Layer** – Lập trình viên tạo ra các lời gọi đến CSDL thông qua SQL và nhận kết quả truy vấn.

- **Driver layer** – Điều khiển tất cả sự truyền thông với sự cài đặt các driver cụ thể. Đây là nhiệm vụ của việc viết code để các hoạt động như giao tiếp với CSDL và hỗ trợ các lời gọi mắc ứng dụng JDBC.
- ✓ Chi tiết 4 interface chính mỗi driver layer phải cài đặt là:
 - **Driver:** phần điều khiển
 - **Connection:** tham chiếu đến đối tượng kết nối CSDL
 - **Statement:** chứa câu lệnh SQL
 - **ResultSet:** chứa kết quả truy vấn dữ liệu

3.2.2 Ví dụ đơn giản

Ví dụ 1: bổ sung một khóa học vào bảng Courses của CSDL ThucHanh

```
try{
    /*
     * Các thông số kết nối CSDL
     */
    String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    String dburl = "jdbc:sqlserver://localhost:1433;DatabaseName=ThucHanh";
    String user = "sa";
    String pass = "songlong";

    //1. Driver - Tải trình điều khiển
    Class.forName(driver);

    //2. Connection - Mở kết nối
    Connection connection = DriverManager.getConnection(dburl, user, pass);

    //3. Statement - Tạo Statement để thực thi câu lệnh SQL
    Statement statement = connection.createStatement();

    //4. Thực thi câu lệnh SQL (INSERT, UPDATE, DELETE)
    String sql = "INSERT INTO Courses(Name, Schoolfee, NoOfLearners," +
        " StartDate, Finished) VALUES('Struts & Hibernate', " +
        " 100, 25, '12/31/2012', 0)";
    int rows = statement.executeUpdate(sql);
    System.out.printf("Số bản ghi bị ảnh hưởng: %s\r\n", rows);

    //5. Đóng kết nối
    connection.close();
}
catch (Exception e) {
    throw new RuntimeException(e);
}
```

Ví dụ 2: truy vấn các khóa học trong bảng Courses của CSDL ThucHanh có học phí từ 10 đến 99 và hiển thị kết quả truy vấn trên màn hình Console.

```
try{
    /*
```

```
* Các thông số kết nối CSDL
*/
String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
String dburl = "jdbc:sqlserver://localhost:1433;DatabaseName=ThucHanh";
String user = "sa";
String pass = "songlong";

//1. Driver - Tải trình điều khiển
Class.forName(driver);

//2. Connection - Mở kết nối
Connection connection = DriverManager.getConnection(dburl, user, pass);

//3. Statement - Tạo Statement để thực thi câu lệnh SQL
Statement statement = connection.createStatement();

//4. ResultSet - Truy vấn dữ liệu
String sql = "SELECT * FROM Courses WHERE Schoolfee BETWEEN 10 AND 99";
ResultSet rs = statement.executeQuery(sql);
//-- Xử lý kết quả truy vấn
while(rs.next()){
    //--đọc dữ liệu tại các cột
    int id = rs.getInt("Id");
    String name = rs.getString("Name");
    double schoolfee = rs.getDouble("Schoolfee");
    int noOfLearners = rs.getInt("NoOfLearners");
    Date startDate = rs.getDate("StartDate");
    boolean finished = rs.getBoolean("Finished");
    //--xuất dữ liệu ra màn hình
    System.out.printf("name=%s, schoolfee=%s\r\n", name, schoolfee);
}

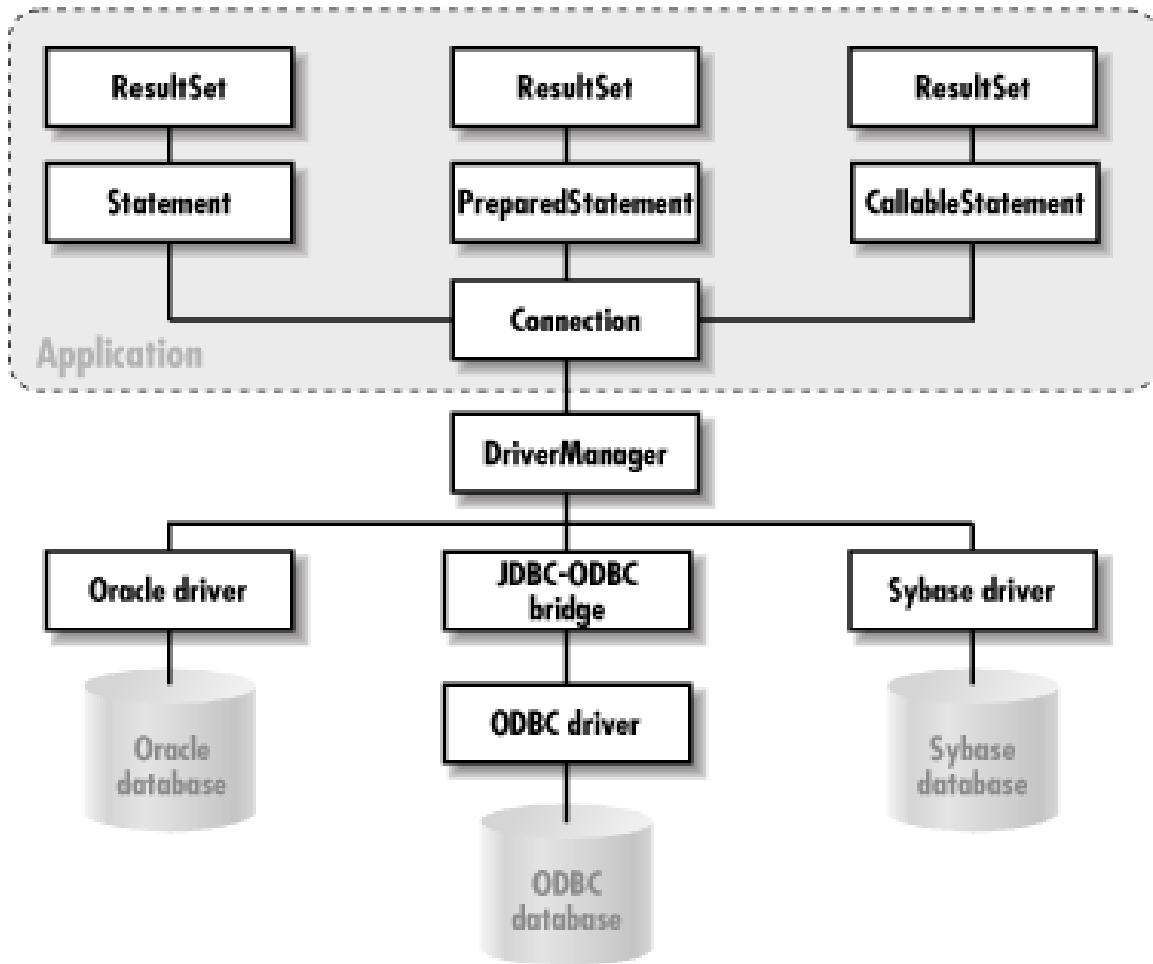
//5. Đóng kết nối
connection.close();
}
catch (Exception e) {
    // xử lý lỗi
    throw new RuntimeException(e);
}
```

Qua hai ví dụ (bổ sung và truy vấn) trên bạn rút ra được gì?

- ✓ Cần có thư viện JDBC sqljdbc5.jar để làm việc với SQL Server 2000/2005/2008
- ✓ Các đối tượng DriverManager, Connection, Statement, ResultSet thuộc gói java.sql
- ✓ Các bước thực hiện chỉ khác nhau ở bước 4.
 - Với SELECT -> phải sử dụng executeQuery() và trích rút dữ liệu trả về từ ResultSet
 - Với INSERT -> số bản ghi insert được là bao nhiêu

3.2.3 Kiến trúc tổ chức của JDBC

- ✓ JDBC là bộ giao tiếp lập trình ứng dụng (JDBC API) được sử dụng để kết nối và làm việc với CSDL trong Java.
- ✓ Nó định nghĩa một tập các class và interface chứa các phương thức sử dụng để giao tiếp với CSDL
- ✓ JDBC không thể thiếu đối với các ứng dụng Java truy xuất đến các nguồn lưu trữ bên ngoài như SQL Server, Oracle, MySQL, MS Access...



Kiến trúc ứng dụng JDBC

Ở 2 ví dụ trước bạn chỉ mới được làm quen với DriverManager, Connection, Statement, ResultSet còn PreparedStatement và CallableStatement sẽ được tìm hiểu sau.

3.2.4 Làm việc với ResultSet

- ✓ Một câu lệnh truy vấn (SELECT) trả kết quả dưới dạng bảng.
- ✓ ResultSet duy trì con trỏ để theo dõi hàng hiện tại đang làm việc.
- ✓ ResultSet hoàn toàn phụ thuộc vào các đối tượng Statement và Connection. Nếu đóng Connection hay Statement thì không thể làm việc được với ResultSet.
- ✓ Đối tượng ResultSet sẽ tự đóng khi đối tượng Statement liên quan bị đóng lại.
- ✓ Hình sau đây minh họa vị trí con trỏ xác định hàng hiện tại bên trong một ResultSet

1	Name	Age	Place		
2	Harry	34	Florida		
3	Samson	19	London		
4	Johny	25	Ottawa		
5	Carol	45	Auckland		
6	Christina	23	Sydney		
7	Mary	9	Rome		
8					
9					



Mô hình ResultSet và việc điều hướng

Các chức năng đơn giản của ResultSet

- ✓ Khi một ResultSet được mở ra thì con trỏ được trỏ vào vị trí trước bản ghi đầu tiên (gọi là before first).
- ✓ Phương thức boolean ResultSet.next() sẽ chuyển vị trí con trỏ đến bản ghi tiếp theo. Nếu con trỏ chỉ vào hàng cuối thì kết quả trả về của phương thức next() là false
- ✓ Phương thức Int ResultSet.getRow() sẽ cho vị trí của hàng hiện tại (tính từ 1)
- ✓ Các phương thức get<Type>(int columnIndex) hay get<Type>(int columnName) sẽ lấy dữ liệu của cột tại vị trí columnIndex hay tên cột columnName ở hàng hiện tại.
 - getString(2): lấy dữ liệu chuỗi của cột thứ 2 ở hàng hiện tại. Vị trí được tính từ 1.
 - getInt("Age"): lấy dữ liệu số nguyên của cột có tên "Age" ở hàng hiện tại

3.2.5 PreparedStatement

Giao tiếp PreparedStatement kế thừa từ giao tiếp Statement và làm việc với câu lệnh SQL có tham số.

- ✓ Statement sẽ dịch và thực thi câu lệnh SQL trước khi thực thi câu lệnh đó. Trong khi đó câu lệnh SQL được dịch trước trong PreparedStatement do vậy sẽ tiết kiệm được thời gian dịch nếu câu lệnh đó được thực hiện nhiều lần.
- ✓ Tránh được lỗi dấu nháy đơn của giá trị
- ✓ Tránh được hacker sử dụng kỹ thuật SQL injection.
- ✓ Cho phép chèn và cập nhật dữ liệu nhị phân
- ✓ Mã nguồn rõ ràng, sáng sủa, tránh được rắc rối trong trường hợp bảng có quá nhiều cột
- ✓ Câu lệnh SQL trong PreparedStatement có thể không hoặc có nhiều tham số vào IN.
- ✓ Sử dụng phương thức set<Type>(index, value) để cấp giá trị cho tham số tại vị trí index (tính từ 1). Trong đó:
 - <Type> là tên kiểu dữ liệu (Int, Long, Float, Double, Boolean, Date...)
 - Index là vị trí đánh dấu giữ chỗ, tính từ 1.
 - value là giá trị cấp cho tham số IN đặt tại vị trí index
- ✓ Một lần gọi phương thức set<Type>() chỉ cấp 1 giá trị cho 1 tham số. Bạn phải gọi nhiều lần khác nhau để cấp cho các tham số ở vị trí khác.

Để tạo một đối tượng PreparedStatement, bạn phải viết đoạn mã nguồn như sau:

```
String sql = "UPDATE course SET hours=? WHERE coursetitle=?";
PreparedStatement pstmt = con.prepareStatement(sql);
```

Trong đó con là đối tượng kết nối đến CSDL

Ví dụ:

```
try{
    /*
     * Thông số kết nối CSDL
     */
    String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    String dburl = "jdbc:sqlserver://localhost:1433;DatabaseName=eStore";
    String uid = "sa";
    String pwd = "songlong";

    //1. Driver
    Class.forName(driver);

    //2. Connection
    Connection connection = DriverManager.getConnection(dburl, uid, pwd);

    //3. PreparedStatement
    String updateSql = "UPDATE Loai SET Hinh=? WHERE TenLoai LIKE ?";
    PreparedStatement pstmt = connection.prepareStatement(updateSql);
    //--cung cấp giá trị cho các tham số
    pstmt.setString(1, "Nokia.gif");
    pstmt.setString(2, "%Nokia%");

    //4. Thực thi câu lệnh
    pstmt.executeUpdate();
    System.out.println("Cập nhật thành công !");

    //3. PreparedStatement
    String selectSql = "SELECT * FROM Loai WHERE TenLoai LIKE ?";
    PreparedStatement sstmt = connection.prepareStatement(selectSql);
    //--cấp giá trị cho tham số
    sstmt.setString(1, "%Nokia%");

    //4. ResultSet
    ResultSet rs = sstmt.executeQuery();
    //--duyet tập kết quả và xử lý
    while(rs.next()){
        System.out.println("MaLoai: "+rs.getInt("MaLoai")+" ");
        System.out.println("TenLoai: "+rs.getString("TenLoai")+" ");
        System.out.println("Hinh: "+rs.getString("Hinh"));
        System.out.println("-----");
    }

    //5. Đóng kết nối
    connection.close();
}
catch(ClassNotFoundException ex){
```

```
System.out.println("Lỗi tải driver: " + ex.getMessage());
}
catch(SQLException ex){
    System.out.println("Lỗi SQL: " + ex.getMessage());
}
```

3.2.6 CallableStatement

Stored procedure (PROC) là thủ tục lưu được viết trong lòng CSDL để thực hiện một công việc cụ thể nào đó (như bạn đã biết trong phần ngôn ngữ SQL). Để thực thi PROC trong Java, chúng ta sử dụng CallableStatement.

Cú pháp lời gọi PROC

✓ {call tên_proc(?, ?)}

CallableStatement kế thừa từ PreparedStatement. Mã Java tạo đối tượng CallableStatement như sau

```
String sql = "{call getData(?, ?)}";
CallableStatement cstm = con.prepareCall(sql);
```

Trong đó

✓ getData() là tên của PROC

✓ ?, ? là vị trí của các tham số

Ví dụ

```
try{
    /*
     * Thông số kết nối CSDL
     */
    String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    String dburl = "jdbc:sqlserver://localhost:1433;DatabaseName=eStore";
    String uid = "sa";
    String pwd = "songlong";

    //1. Driver
    Class.forName(driver);

    //2. Connection
    Connection connection = DriverManager.getConnection(dburl, uid, pwd);

    //3. CallableStatement
    String sql = "{call spPhanTrang (?, ?)}";
    CallableStatement cstmt = connection.prepareCall(sql);
    //--truyền tham số cho PROC
    cstmt.setInt(1, 5); // bản ghi bắt đầu
    cstmt.setInt(2, 10); // số bản ghi cần lấy

    //4. ResultSet
    ResultSet rs = cstmt.executeQuery();
    //--Xử lý tập kết quả
```

```
if(rs.next()) {
    String TenHH = rs.getString("TenHH");
    double DonGia = rs.getDouble("DonGia");
    System.out.printf("Name: %s, Price: %.3f", TenHH, DonGia);
}

//5. Close
connection.close();
}
catch(ClassNotFoundException ex){
    System.out.println("Lỗi tải driver: " + ex.getMessage());
}
catch(SQLException ex){
    System.out.println("Lỗi SQL: " + ex.getMessage());
}
```

3.2.7 Khối lệnh (Statement Batch)

Trong ứng dụng đôi khi chúng ta thực hiện nhiều câu lệnh cùng một lúc. Chẳng hạn như tạo một đơn hàng, khi đó chúng ta phải thêm một bản ghi vào bản hóa đơn và nhiều bản ghi vào bảng chi tiết hóa đơn. Để tránh gửi các câu lệnh nhiều lần đến CSDL để thực thi, JDBC hỗ trợ cơ chế thực thi khối câu lệnh, nghĩa là chúng ta đóng các câu lệnh thành một lô (batch) và gửi đến CSDL để thực thi.

Ưu điểm của phương pháp này là: giảm số lần truyền thông và cải thiện tốc độ thực hiện.

Ví dụ:

```
try{
    /*
     * Thông số kết nối CSDL
     */
    String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    String dburl = "jdbc:sqlserver://localhost:1433;DatabaseName=eStore";
    String uid = "sa";
    String pwd = "songlong";

    //1. Driver
    Class.forName(driver);

    //2. Connection
    Connection connection = DriverManager.getConnection(dburl, uid, pwd);

    //3. Statement
    Statement stmt = connection.createStatement();
    stmt.addBatch("INSERT INTO Person(Name, Age) VALUES('A', 50)");
    stmt.addBatch("INSERT INTO Person(Name, Age) VALUES('B', 30)");
    stmt.addBatch("DELETE FROM Person WHERE Age NOT BETWEEN 15 AND 70");

    //4 Thực thi khối lệnh
    stmt.executeBatch();

    //5. Close
    connection.close();
}
```

```
catch(ClassNotFoundException ex){  
    System.out.println("Lỗi tải driver: " + ex.getMessage());  
}  
catch(SQLException ex){  
    System.out.println("Lỗi SQL: " + ex.getMessage());  
}  
}
```

3.2.8 Sử dụng Transactions

“ĐƯỢC ĂN CẢ, NGÃ VỀ KHÔNG” là cơ chế làm việc của transaction nghĩa là xem một khối lệnh như một lệnh đơn. Khi có một lệnh lỗi thì toàn bộ khối lệnh sẽ bị hủy.

Một kết nối CSDL mặc định ở chế độ chuyển giao tự động (auto-commit), tức là khi một lệnh được thực hiện bởi các phương thức executeXYZ() thì ngay lập tức được gửi đến CSDL để thực thi. Như vậy một câu lệnh được đối xử như một transaction (hai câu lệnh khác nhau được xem như 2 giao dịch khác nhau).

Chế độ auto-commit phải được hủy bỏ để có thể thực hiện hai hay nhiều hơn các câu lệnh như một nhóm lệnh thống nhất được đối xử như một transaction. Một khi chế độ auto-commit được hủy, không có một câu lệnh nào được chuyển giao tự động mà phải đợi cho đến khi một lời gọi phương thức commit() được thực hiện thì toàn bộ khối lệnh đó mới thực sự được chuyển giao. Hoặc lời gọi phương thức rollback() được thực hiện để toàn bộ các câu lệnh đã thực hiện trước đó sẽ bị hủy bỏ.

Để điều khiển transaction chúng ta sử dụng cấu trúc lệnh sau

```
Connection connection = ...  
//hủy chế độ auto-commit  
connection.setAutoCommit(false);  
try  
{  
    Thực thi câu lệnh 1  
    Thực thi câu lệnh 2  
    ...  
    Thực thi câu lệnh N  
  
    //chấp nhận chuyển giao  
    connection.commit();  
}  
catch(SQLException ex){  
    //hủy các lệnh đã thực hiện  
    connection.rollback();  
}  
//đặt lại chế độ auto-commit  
connection.setAutoCommit(true);
```

Ví dụ:

```
try{  
    /*  
     * Thông số kết nối CSDL  
     */  
    String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";  
    String dburl = "jdbc:sqlserver://localhost:1433;DatabaseName=eStore";
```

```
String uid = "sa";
String pwd = "songlong";

//1. Driver
Class.forName(driver);

//2. Connection
Connection connection = DriverManager.getConnection(dburl, uid, pwd);

try {
    // hủy chế độ auto-commit
    connection.setAutoCommit(false);

    // thực thi câu lệnh 1
    String sql1 = "Delete Student where Rollno = 3";
    PreparedStatement delStud = connection.prepareStatement(sql1);
    delStud.executeUpdate();

    // thực thi câu lệnh 2
    String sql2 = "Delete Results where Rollno = 3";
    PreparedStatement delResult = connection.prepareStatement(sql2);
    delResult.executeUpdate();

    // thực hiện chuyển giao
    connection.commit();
}
catch(SQLException ex) {
    // hủy bỏ các câu lệnh đã thực hiện
    connection.rollback();
}
// đặt lại chế độ auto-commit
connection.setAutoCommit(true);
}
catch(ClassNotFoundException ex){
    System.out.println("Lỗi tải driver: " + ex.getMessage());
}
catch(SQLException ex){
    System.out.println("Lỗi SQL: " + ex.getMessage());
}
}
```

Ở ví dụ trên, 2 thao tác xóa dữ liệu được đặt trong một transaction. Điều đó có nghĩa là chỉ cần một trong 2 thao tác thất bại thì cả 2 lệnh sẽ bị hủy bỏ.

3.2.9 Bài tập

1. Tạo CSDL DaoTao gồm 2 bảng SinhVien, LopHoc và một thủ tục lưu như sau:

```
CREATE LopHoc
(
    MaLH INT IDENTITY(1000, 1) PRIMARY KEY,
    TenLH NVARCHAR(50) NOT NULL,
    NgayKG DATETIME NOT NULL,
    NgayBG DATETIME NULL
)
```

```
);  
  
CREATE SinhVien  
(  
    MaSV NVARCHAR(20) PRIMARY KEY,  
    HoTen NVARCHAR(50) NOT NULL,  
    NgaySinh DATETIME NOT NULL,  
    GioiTinh BIT DEFAULT 1 NOT NULL,  
    Diem FLOAT DEFAULT 0,  
    GhiChu NVARCHAR(255),  
    MaLH INT NOT NULL FOREIGN KEY REFERENCES LopHoc(MaLH) ON DELETE  
CASCADE  
);  
  
CREATE PROC sp_BoSungLH  
(  
    @TenLH NVARCHAR(50),  
    @NgayKG DATETIME,  
)  
AS  
INSERT INTO LopHoc(TenLH, NgayKG) VALUES(@TenLH, @NgayKG)
```

- Viết chương trình Java cho phép thực hiện chèn mỗi bảng 5 bản ghi sử dụng Statement.
- Viết chương trình Java cho phép thực hiện chèn mỗi bảng 2 bản ghi sử dụng PreparedStatement và CallableStatement đối với bảng LopHoc.
- Viết chương trình Java cho phép thực hiện cập nhật điểm (tùy thích) của sinh viên có mã số là SV002.
- Viết chương trình Java cho phép thực hiện hiển thị tất cả các sinh viên nam và sinh năm chẵn.
- Viết chương trình Java cho phép thực hiện hiển thị tất cả các sinh viên nhập học năm 2010 và đã kết thúc.
- Viết chương trình Java cho phép thực hiện hiển thị tất cả các sinh viên mang họ nguyên và điểm từ 6.5 đến 7.5.
- Viết chương trình Java cho phép thực hiện hiển thị tất cả các sinh viên có họ tên hoặc tên lớp học chứa chuỗi "Java".
- Viết chương trình Java cho phép thực hiện thống kê số lượng sinh viên nam, nữ theo lớp khai giảng năm 2011.
- Viết chương trình Java cho phép thực hiện thống kê điểm trung bình, điểm thấp nhất, điểm cao nhất của mỗi lớp.
- Viết chương trình Java cho phép thực hiện thống kê số lượng sinh viên theo lớp của các lớp có ít nhất 10 sinh viên.
- Viết chương trình Java cho phép hiển thị tất cả các bảng và thông tin các cột trong CSDL DaoTao.

3.3 Hibernate

3.3.1 Hibernate cơ bản

3.3.1.1 Giới thiệu

- ✓ Hibernate là một giải pháp ánh xạ đối tượng quan hệ trong Java và nó được xem như một framework quản lý lưu trữ mở.
- ✓ Hibernate ánh xạ các lớp Java vào các bảng của CSDL quan hệ và kiểu dữ liệu Java với kiểu dữ liệu SQL.
- ✓ Hibernate đứng ở giữa các đối tượng Java truyền thống và server CSDL để điều hành các công việc quản lý lưu trữ trạng thái của các đối tượng đó dựa trên cơ chế ánh xạ thích hợp.



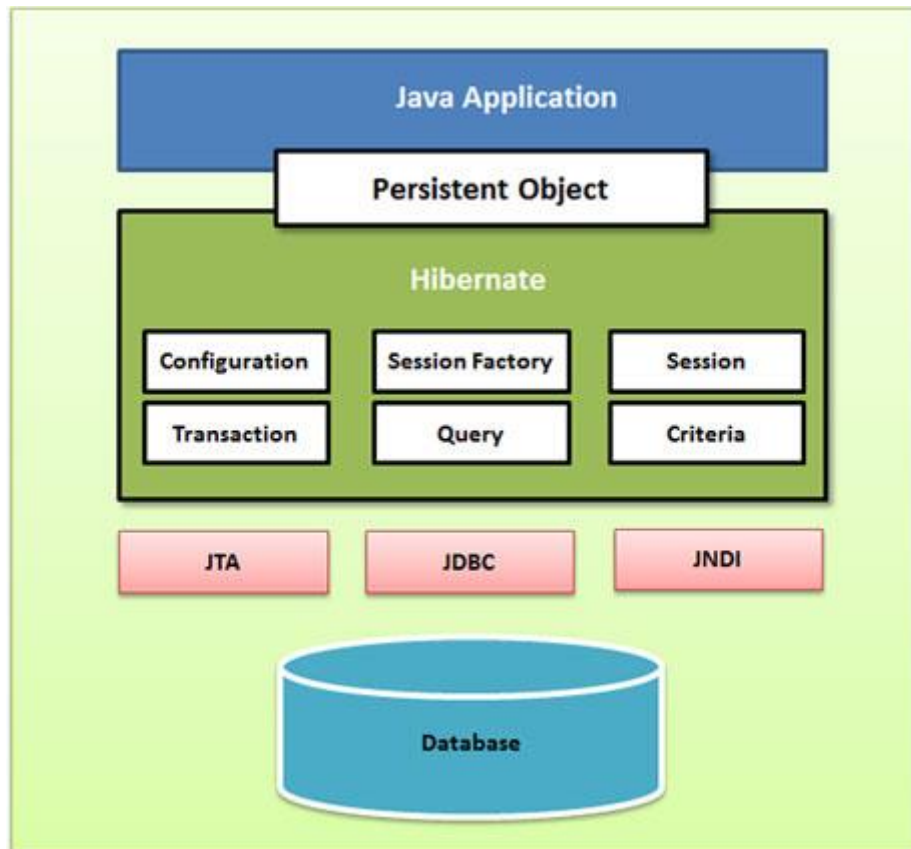
3.3.1.2 Ưu điểm của hibernate

- ✓ Hibernate tập trung vào việc ánh xạ các lớp Java vào các bảng trong CSDL bằng XML hoặc annotation mà không cần phải viết mã Java.
- ✓ Hibernate cung cấp API đơn giản để lưu trữ và truy xuất các đối tượng Java trực tiếp từ cơ sở dữ liệu.
- ✓ Nếu có sự thay đổi trong cơ sở dữ liệu hoặc trong bất kỳ bảng nào, chúng ta chỉ cần thay đổi các thuộc tính trong tập tin XML hoặc annotation.
- ✓ Trong suốt với SQL: chúng ta chỉ làm việc với các đối tượng Java quen thuộc.
- ✓ Thao tác các đối tượng kết hợp một cách dễ dàng thông qua mối quan hệ giữa các thực thể.
- ✓ Giảm thiểu sự truy cập cơ sở dữ liệu nhờ cơ chế xử lý thông minh.
- ✓ Cung cấp kỹ thuật truy vấn dữ liệu đơn giản.
- ✓ Công việc này sẽ giảm bớt 95% nhiệm vụ của người lập trình lưu trữ dữ liệu.
- ✓ Hibernate hỗ trợ hầu hết các hệ quản trị CSDL quan hệ chính hiện tại.
 - HSQL Database Engine
 - DB2/NT
 - MySQL
 - PostgreSQL
 - FrontBase
 - Oracle

- Microsoft SQL Server Database
- Sybase SQL Server
- Informix Dynamic Server

3.3.1.3 Kiến trúc

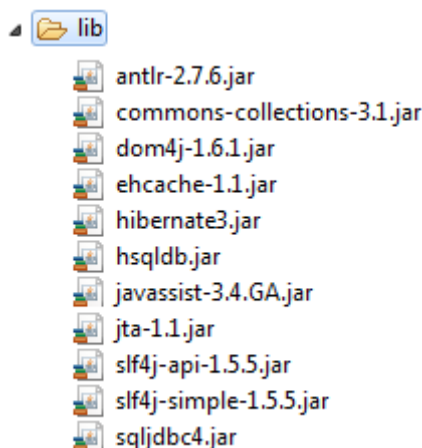
- ✓ Kiến trúc Hibernate được phân tầng để bạn không cần phải quan tâm về API phía dưới. Hibernate tạo ra CSDL và dữ liệu cấu hình để cung cấp cho các dịch vụ quản lý lưu trữ trạng thái đối với ứng dụng.



- ✓ Tổng quan về các đối tượng của Hibernate:
 - **Configuration:** được sử dụng để quản lý thông tin cấu hình kết nối đến CSDL và ánh xạ thực thể vào CSDL.
 - **SessionFactory:** đối tượng này cho phép sản sinh ra nhiều phiên làm (session) việc khác nhau.
 - **Session:** sử dụng để tạo một phiên làm việc với CSDL thông qua Hibernate.
 - **Transaction:** sử dụng để điều khiển transaction (none or all)
 - **Query:** Đối tượng này được sử dụng để thực hiện truy vấn dữ liệu
 - **Criteria:** Đối tượng này được sử dụng để xây dựng câu lệnh truy vấn bằng lập trình thay cho truy vấn bằng câu lệnh HQL hay SQL.

3.3.1.4 Cài đặt môi trường

Để có thể lập trình và chạy được ứng dụng với Hibernate, bạn chỉ cần bổ sung các thư viện sau đây vào ứng dụng của bạn là đủ.



Trong số các thư viện trên, sqljdbc4.jar được sử dụng để làm việc với SQL Server 2000/2005/2008 của Microsoft.

Nếu bạn muốn làm việc với một CSDL cụ thể nào đó (Oracle, MySQL, DB2...) thì phải bổ sung thư viện JDBC tương ứng với CSDL đó vào ứng dụng của bạn.

3.3.1.5 Cấu hình

Cấu hình là phần công việc đầu tiên và không thể thiếu khi làm việc với CSDL. Các thông số kết nối CSDL và các thực thể ánh xạ phải được khai báo để Hibernate quản lý.

3.3.1.5.1 CSDL mẫu

Để tiện lợi cho việc trình bày thống nhất giữa các phần trong tài liệu, chúng tôi giới thiệu một CSDL mẫu gồm 3 bảng hội đủ các đặc điểm kỹ thuật có lợi cho việc minh họa với Hibernate.

- ✓ Courses: quản lý thông tin khóa học (độc lập không liên kết với bảng nào khác)
- ✓ Categories: quản lý thông tin chủng loại hàng hóa
- ✓ Products: quản lý thông tin hàng hóa
- ✓ Bảng Categories và Products có quan hệ một nhiều

Courses					
	Column Name	Condensed Type	Nullable	Identity	Description
🔑	Id	int	No	<input checked="" type="checkbox"/>	Mã khóa học
	Name	nvarchar(50)	No	<input type="checkbox"/>	Tên khóa học
	Schoolfee	float	No	<input type="checkbox"/>	Học phí
	NoOfLearners	int	No	<input type="checkbox"/>	Số học viên
	StartDate	datetime	No	<input type="checkbox"/>	Ngày khai giảng
	Finished	bit	No	<input type="checkbox"/>	Kết thúc?
				<input type="checkbox"/>	

Categories					
	Column Name	Condensed Type	Nullable	Identity	Description
🔑	Id	char(3)	No	<input type="checkbox"/>	Mã loại
	Name	nvarchar(50)	No	<input type="checkbox"/>	Tên loại
	Description	nvarchar(255)	Yes	<input type="checkbox"/>	Mô tả
				<input type="checkbox"/>	

Products					
	Column Name	Condensed Type	Nullable	Identity	Description
🔑	Id	int	No	<input checked="" type="checkbox"/>	Mã hàng hóa
	Name	nvarchar(50)	No	<input type="checkbox"/>	Tên hàng hóa
	UnitPrice	float	No	<input type="checkbox"/>	Đơn giá
	ProductDate	datetime	No	<input type="checkbox"/>	Ngày sản xuất
	Image	nvarchar(50)	No	<input type="checkbox"/>	Hình ảnh
	Description	nvarchar(4000)	Yes	<input type="checkbox"/>	Mô tả
	CategoryId	char(3)	No	<input type="checkbox"/>	Mã loại
	Available	bit	No	<input type="checkbox"/>	Vẫn còn hàng
				<input type="checkbox"/>	

3.3.1.5.2 Tập tin cấu hình

Tập tin cấu hình hibernate.cfg.xml cung cấp các thông số kết nối đến CSDL bạn muốn làm việc và khai báo các thực thể ánh xạ với các bảng trong CSDL.

Sau đây là tập tin cấu hình dùng để làm việc với CSDL Java ở phần trên.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
    <property name="hibernate.connection.driver_class">
        com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
    <property name="hibernate.connection.url">
        jdbc:sqlserver://localhost:1433;DatabaseName=Java;</property>
    <property name="hibernate.connection.username">sa</property>
    <property name="connection.password">songlong</property>
    <property name="connection.pool_size">100</property>
    <property name="show_sql">>false</property>
    <property name="hbm2ddl.auto">none</property>
    <property name="hibernate.dialect">
        org.hibernate.dialect.SQLServerDialect</property>

    <!-- Mapping files -->
    <mapping resource="Course.hbm.xml"/>
    <mapping resource="Category.hbm.xml"/>
    <mapping resource="Product.hbm.xml"/>
</session-factory>
```

```
</hibernate-configuration>
```

Trong `<session-factory>` chứa nhiều `<property>` và `<mapping>`. Mỗi `<property>` được sử dụng để khai báo một thông số khác nhau tùy vào giá trị của thuộc tính `@name`. Mỗi `<mapping>` khai báo một tập tinh ánh xạ thực thể (tập tin này sẽ được trình bày ở phần sau - Ánh xạ).

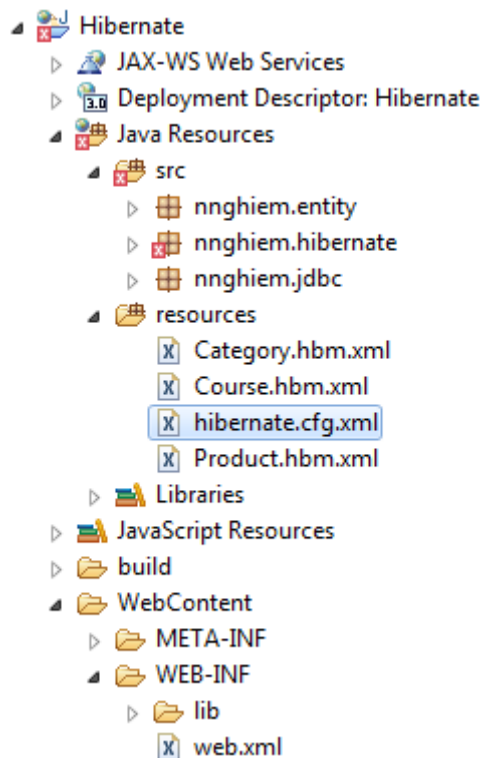
❖ Giá trị của thuộc tính `@name`

@name	Mô tả
hibernate.connection.driver_class	JDBC driver
hibernate.connection.url	Chuỗi kết nối đến CSDL
hibernate.connection.username	Mã đăng nhập CSDL
connection.password	Mật khẩu đăng nhập CSDL
connection.pool_size	Số kết nối tối đa trong hồ
show_sql	Xuất câu lệnh SQL khi truy vấn hoặc thao tác dữ liệu
hbm2ddl.auto	Chỉ ra cơ chế tự định nghĩa CSDL, thường dùng <ul style="list-style-type: none"> ✓ create-drop: tự tạo CSDL ✓ none: làm việc với CSDL đã tồn tại
hibernate.dialect	Chỉ ra phương pháp làm việc với CSDL của Hibernate (xem bảng bên dưới để khai báo cho đúng)

Database	Dialect Property
DB2	org.hibernate.dialect.DB2Dialect
HSQLDB	org.hibernate.dialect.HSQLDialect
HypersonicSQL	org.hibernate.dialect.HSQLDialect
Informix	org.hibernate.dialect.InformixDialect
Ingres	org.hibernate.dialect.IngresDialect
Interbase	org.hibernate.dialect.InterbaseDialect
Microsoft SQL Server 2000	org.hibernate.dialect.SQLServerDialect
Microsoft SQL Server 2005	org.hibernate.dialect.SQLServer2005Dialect

Microsoft SQL Server 2008	org.hibernate.dialect.SQLServer2008Dialect
MySQL	org.hibernate.dialect.MySQLDialect
Oracle (any version)	org.hibernate.dialect.OracleDialect
Oracle 11g	org.hibernate.dialect.Oracle10gDialect
Oracle 10g	org.hibernate.dialect.Oracle10gDialect
Oracle 9i	org.hibernate.dialect.Oracle9iDialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
Progress	org.hibernate.dialect.ProgressDialect
SAP DB	org.hibernate.dialect.SAPDBDialect
Sybase	org.hibernate.dialect.SybaseDialect
Sybase Anywhere	org.hibernate.dialect.SybaseAnywhereDialect

Tập tin cấu hình hibernate.cfg.xml được đặt tại thư mục gốc nơi đặt mã nguồn Java.



3.3.1.6 Ánh xạ

3.3.1.6.1 Lớp thực thể

Lớp thực thể (hay còn gọi là persistent class) dùng để mô tả một thực thể và được ánh xạ với bảng trong CSDL. Lớp này phải được định nghĩa tuân thủ các quy ước của JavaBean, nghĩa là phải có phương thức khởi dựng (Constructor) mặc định và các thuộc tính (Property) phải có dạng getXyz() và setXyz(). Nếu thuộc tính Xyz có kiểu là boolean thì qui bạn có thể sử dụng getXyz() hoặc isXyz().

Sau đây là lớp thực thể mô tả khóa học dùng để ánh xạ vào bảng Courses trong CSDL.

- ✓ Lớp này không định nghĩa constructor thức sử dụng constructor mặc định.
- ✓ Các thuộc tính
 - getId()/setId()
 - getName()/setName()
 - getSchoolfee()/setSchoolfee()
 - getNoOfLearners()/setNoOfLearners()
 - getStartDate()/setStartDate()
 - isFinished()/setFinished()

```
package nnghiem.entity;  
  
import java.util.Date;  
  
public class Course {  
  
    protected int id;  
    protected String name;  
    protected double schoolfee;  
    protected int noOfLearners;  
    protected Date startDate;  
    protected boolean finished;  
  
    getters/setters  
  
}
```

3.3.1.6.2 Tập tin ánh xạ

Tập tin này dùng để mô tả sự ánh xạ giữa lớp thực thể vào một bảng trong CSDL. Sau đây là tập tin mô tả ánh xạ giữa lớp thực thể Course và bảng Courses trong CSDL Java.

```
<?xml version="1.0"?>  
<!DOCTYPE hibernate-mapping PUBLIC  
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"  
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">  
  
<hibernate-mapping>  
    <class name="nnghiem.entity.Course" table="Courses">  
        <id name="id" type="int" column="Id">  
            <generator class="native" />  
        </id>  
        <property name="name" type="string" column="Name" />  
    </class>  
</hibernate-mapping>
```

```
<property name="schoolfee" type="double" column="Schoolfee" />
<property name="noOfLearners" type="int" column="NoOfLearners" />
<property name="startDate" type="date" column="StartDate" />
<property name="finished" type="boolean" column="Finished" />
</class>
</hibernate-mapping>
```

Sau đây là mô tả các thẻ sử dụng trong tập tin này

- ✓ **<hibernate-mapping>**: là thẻ gốc, chứa nhiều thẻ <class>
- ✓ **<class>**: ánh xạ một lớp thực thể và một bảng
 - @**name**: chỉ ra tên lớp thực thể
 - @**table**: chỉ ra tên bảng
- ✓ **<id>**: ánh xạ thuộc tính của lớp thực thể với cột khóa chính
 - @**name**: tên thuộc tính
 - @**type**: kiểu của thuộc tính
 - @**column**: cột được ánh xạ trong bảng
 - **<generator>**: chỉ ra cách sinh giá trị tự tăng
- ✓ **<property>**: ánh xạ thuộc tính của lớp thực thể với cột thông thường
 - @**name**: tên thuộc tính
 - @**type**: kiểu của thuộc tính
 - @**column**: cột được ánh xạ trong bảng

3.3.1.6.3 Kiểu dữ liệu ánh xạ

Sau đây là qui ước về kiểu dữ liệu ánh xạ giữa kiểu của thuộc tính thực thể và kiểu SQL trong CSDL.

- ✓ Ánh xạ với kiểu nguyên thủy

Mapping type	Java type	ANSI SQL Type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte or java.lang.Byte	TINYINT
boolean	boolean or java.lang.Boolean	BIT
yes/no	boolean or java.lang.Boolean	CHAR(1) ('Y' or 'N')
true/false	boolean or java.lang.Boolean	CHAR(1) ('T' or 'F')

- ✓ Ánh xạ với kiểu thời gian

Mapping type	Java type	ANSI SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

- ✓ Ánh xạ kiểu dữ liệu lớn

Mapping type	Java type	ANSI SQL Type
binary	byte[]	VARBINARY (or BLOB)
text	java.lang.String	CLOB
serializable	any Java class that implements java.io.Serializable	VARBINARY (or BLOB)
clob	java.sql.Clob	CLOB
blob	java.sql.Blob	BLOB

- ✓ Ánh xạ với một số kiểu dữ liệu khác

Mapping type	Java type	ANSI SQL Type
class	java.lang.Class	VARCHAR
locale	java.util.Locale	VARCHAR
timezone	java.util.TimeZone	VARCHAR
currency	java.util.Currency	VARCHAR

3.3.1.7 Ứng dụng java

Sau khi chuẩn bị xong các khâu cần thiết (cài đặt môi trường, cấu hình, ánh xạ thực thể), bạn có thể viết mã ứng dụng Java để làm việc với CSDL.

3.3.1.7.1 Khởi đầu

Bước đầu tiên trong ứng dụng Hibernate là tải tập tin cấu hình. Sau đó mở ra một phiên làm việc với CSDL.

Đoạn mã sau đây thực hiện công việc nói trên:

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();
```

Sau khi đã mở ra một session bạn có thể truy vấn hoặc thao tác dữ liệu.

3.3.1.7.2 Truy vấn

Có rất nhiều phương pháp truy vấn dữ liệu hay trong Hibernate sẽ được trình bày trong phần Hibernate nâng cao. Trong phần cơ bản, chúng tôi chỉ trình bày 2 cách đơn giản nhất là

- ✓ Truy vấn tất cả các thực thể
- ✓ Truy vấn một thực thể theo khóa chính

3.3.1.7.2.1 Truy vấn tất cả các thực thể

Sau đây là đoạn mã cho phép truy vấn tất cả các bản ghi trong bảng Courses của CSDL Java sau đó chuyển đổi tự động thành danh sách các thực thể Course (List<Course>). Chúng ta chỉ việc duyệt qua danh sách để xử lý (trong trường hợp này chúng ta chỉ xuất tên và học phí ra màn hình) dữ liệu nhận được

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

String hql = "FROM Course";
Query query = session.createQuery(hql);
List<Course> courses = query.list();

for(Course course : courses){
    String name = course.getName();
    double fee = course.getSchoolfee();
    System.out.println("+ Name of course: " + name);
    System.out.println("+ Schoolfee of course: " + fee);
}
```

3.3.1.7.2.2 Truy vấn một thực thể theo khóa chính

Để truy vấn một bản ghi theo khóa và chuyển đổi thành thực thể, bạn sử dụng phương thức load của session.

Sau đây là đoạn mã cho phép tải thực thể có mã là 1

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

Course course = (Course) session.load(Course.class, 1);
String name = course.getName();
double fee = course.getSchoolfee();
System.out.println("+ Name of course: " + name);
System.out.println("+ Schoolfee of course: " + fee);
```

3.3.1.7.3 Thao tác dữ liệu

Thêm, xóa, sửa dữ liệu được thực hiện khá đơn giản với Hibernate nhờ các phương thức của session. Các thao tác này thường đi kèm với transaction.

3.3.1.7.3.1 Cấu trúc mã thao tác dữ liệu

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();
```



```
Transaction transaction = session.beginTransaction();
try{
    ...mã thao tác thực thể...

    /*
     * Chấp nhận kết quả
     */
    transaction.commit();
}
catch (Exception e) {
    /*
     * Hủy bỏ kết quả
     */
    transaction.rollback();
}
```

3.3.1.7.3.2 Thêm mới

Sau đây là đoạn mã lệnh cho phép thêm mới một bản ghi vào bảng Courses. Đoạn mã được viết với sự điều khiển của transaction.

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

Transaction transaction = session.beginTransaction();
try{
    /*
     * Tạo một thực thể mới
     */
    Course entity = new Course();
    entity.setName("Clocks 2012");
    entity.setSchoolfee(123.9);
    entity.setNoOfLearners(25);
    entity.setStartDate(new Date());
    entity.setFinished(false);
    /*
     * Thêm vào CSDL
     */
    session.save(entity);
    /*
     * Chấp nhận kết quả
     */
    transaction.commit();
}
catch (Exception e) {
    /*
     * Hủy bỏ kết quả
     */
    transaction.rollback();
}
```

3.3.1.7.3.3 Thêm mới hoặc cập nhật

Sau đây là đoạn mã lệnh cho phép thêm mới một bản ghi vào bảng Courses nếu trong CSDL chưa có bản ghi có mã là 1. Nếu đã tồn tại bản ghi với mã là 1 thì bản ghi đó được sửa đổi theo thông tin mới.

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

Transaction transaction = session.beginTransaction();
try{
    /*
     * Tạo một thực thể mới có id
     */
    Course entity = new Course();
    entity.setId(1);
    entity.setName("Clocks 2012");
    entity.setSchoolfee(123.9);
    entity.setNoOfLearners(25);
    entity.setStartDate(new Date());
    entity.setFinished(false);
    /*
     * Thêm mới nếu chưa có thực thể nào có
     * id là 1, ngược lại thì cập nhật
     */
    session.saveOrUpdate(entity);
    /*
     * Chấp nhận kết quả
     */
    transaction.commit();
}
catch (Exception e) {
    /*
     * Hủy bỏ kết quả
     */
    transaction.rollback();
}
```

3.3.1.7.3.4 Cập nhật

Sau đây là đoạn mã lệnh cho phép cập nhật bản ghi đã tồn tại trong bảng Courses với mã số là 1.

- ✓ Bước 1: Trước hết là tìm và tải thực thể vào
- ✓ Bước 2: Thay đổi thông tin cần thiết
- ✓ Bước 3: Lưu thực thể trở lại với thông tin đã cập nhật

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

Transaction transaction = session.beginTransaction();
try{
    /*
     * Tải thực thể có mã số là 1 và
     * thay đổi số học viên và ngày khai giảng thực thể
     */
}
```



```

    */
    Course entity = (Course) session.load(Course.class, 1);
    entity.setNoOfLearners(38);
    entity.setStartDate(new Date());
    /*
     * Cập nhật thực thể
     */
    session.update(entity);
    /*
     * Chấp nhận kết quả
     */
    transaction.commit();
}
catch (Exception e) {
    /*
     * Hủy bỏ kết quả
     */
    transaction.rollback();
}

```

3.3.1.7.3.5 Xóa

Sau đây là đoạn mã lệnh cho phép xóa bớt bản ghi có mã số là 1 khỏi bảng Courses.

```

Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

Transaction transaction = session.beginTransaction();
try{
    /*
     * Tạo thực thể có id là 1
     */
    Course entity = new Course();
    entity.setId(1);
    /*
     * Xóa thực thể
     */
    session.delete(entity);
    /*
     * Chấp nhận kết quả
     */
    transaction.commit();
}
catch (Exception e) {
    /*
     * Hủy bỏ kết quả
     */
    transaction.rollback();
}

```

3.3.1.7.3.6 Tải lại (làm tươi) một thực thể

Sau đây là đoạn mã lệnh cho phép tải lại thông tin của một thực thể từ CSDL. Thao tác này rất cần trong các hành động Cancel hoặc tải một thực thể từ CSDL.

```

Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();

```

```
Session session = factory.openSession();

/*
 * Tạo thực thể có id là 1
 */
Course entity = new Course();
entity.setId(1);
/*
 * Tải lại các thuộc tính thực thể từ CSDL
 */

session.refresh(entity);
```

3.3.2 Hibernate nâng cao

3.3.2.1 Truy vấn nâng cao

Bên cạnh kỹ thuật truy vấn đơn giản được trình bày ở phần cơ bản, Hibernate còn cung cấp thêm nhiều kỹ thuật mạnh giúp việc truy vấn dữ liệu thuận tiện hơn.

Trong phần nâng cao này chúng tôi sẽ trình bày thêm các kỹ thuật truy vấn sau:

- ✓ Truy vấn một số thuộc tính
- ✓ Truy vấn một giá trị đơn
- ✓ Truy vấn có tham số
- ✓ Phân trang dữ liệu kết quả truy vấn
- ✓ Kỹ thuật truy vấn đặc thù SQL

3.3.2.1.1 Truy vấn một số thuộc tính

Đôi khi chúng ta chỉ truy vấn một số thuộc tính của thực thể mà không cần truy vấn toàn bộ thực thể. Khi đó kết quả nhận được không phải là danh sách các thực thể mà là danh sách các mảng, mỗi mảng chứa danh sách các thuộc tính.

Đoạn mã sau sẽ giúp bạn thực hiện điều đó

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

/*
 * Truy vấn một số thuộc tính của thực thể
 */
String hql = "SELECT name, schoolfee FROM Course";
Query query = session.createQuery(hql);
List<Object[]> arrays = query.list();

for(Object[] array : arrays){
    String name = (String)array[0];
    double fee = (Double)array[1];
    System.out.println("+ Name of course: " + name);
    System.out.println("+ Schoolfee of course: " + fee);
}
```

3.3.2.1.2 Truy vấn một giá trị đơn

Thỉnh thoảng chúng ta cần một số liệu tổng hợp cụ thể nào đó từ CSDL. Kết quả truy vấn đó chỉ cho một giá trị đơn. Để truy vấn một giá trị đơn chúng ta sử dụng đoạn mã sau đây:

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

/*
 * Truy vấn một giá trị đơn
 */
String hql = "SELECT AVG(schoolfee) FROM Course";
Query query = session.createQuery(hql);
double average = (Double)query.uniqueResult();

System.out.println("+ Average schoolfee of course: " + average);
```

3.3.2.1.3 Truy vấn có tham số

Câu lệnh truy vấn HQL thường xuyên chứa các tham số. Các tham số này sẽ được cấp dữ liệu tại thời điểm chạy chương trình.

- ✓ Để tạo một tham số, bạn sử dụng dấu hai chấm và theo sau là tên tham số (<tên tham số>).
- ✓ Để cung cấp giá trị cho tham số, bạn sử dụng phương thức `setParameter("<tên tham số>", <giá trị tham số>)`. Chú ý giá trị của tham số phải đúng kiểu của thuộc tính nhận tham số.

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();
/*
 * Truy vấn với tham số
 */
String hql = "FROM Course WHERE name LIKE :name "
            + " AND schoolfee BETWEEN :min AND :max";
Query query = session.createQuery(hql);
query.setParameter("name", "%asp%");
query.setParameter("min", 5.5);
query.setParameter("max", 200);
List<Course> courses = query.list();

for(Course course : courses){
    String name = course.getName();
    double fee = course.getSchoolfee();
    System.out.println("+ Name of course: " + name);
    System.out.println("+ Schoolfee of course: " + fee);
}
```

3.3.2.1.4 Truy vấn phân trang

Đôi khi dữ liệu quá lớn chúng ta không thể truy vấn hết ra. Làm như vậy rất nguy hiểm (sẽ làm tràn bộ nhớ). Chúng ta nên chia nhỏ để lấy từng đoạn dữ liệu từ vị trí bắt đầu và một số lượng bản ghi cụ thể nào đó.

Đoạn mã sau giúp bạn thực hiện được điều này (lấy 10 bản ghi bắt đầu từ bản ghi thứ 5):

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

/*
 * Truy vấn có phân trang dữ liệu
 */
String hql = "FROM Course";
Query query = session.createQuery(hql);
query.setFirstResult(5); // vị trí bắt đầu
query.setMaxResults(10); // số lượng thực thể
List<Course> courses = query.list();

for(Course course : courses){
    String name = course.getName();
    double fee = course.getSchoolfee();
    System.out.println("+ Name of course: " + name);
    System.out.println("+ Schoolfee of course: " + fee);
}
```

3.3.2.1.5 Truy vấn với SQL đặc thù

Bạn có thể truy vấn đối tượng bằng câu lệnh SQL thông thường mà không cần sử dụng HQL. Cách truy vấn như vậy người ta gọi là truy vấn đặc thù SQL.

Để truy vấn đối tượng với câu lệnh SQL đặc thù bạn sử dụng phương thức `createSQLQuery()` của `session` thay vì `createQuery()` khi sử dụng HQL. Sử dụng kết hợp với các phương thức **`addEntity()`**, **`addJoin()`** hay **`addScalar()`** để nhận được thực thể, sự kết nối hay một giá trị đơn.

Đoạn mã sau truy vấn danh sách khóa học (Course) bằng câu lệnh SQL

```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

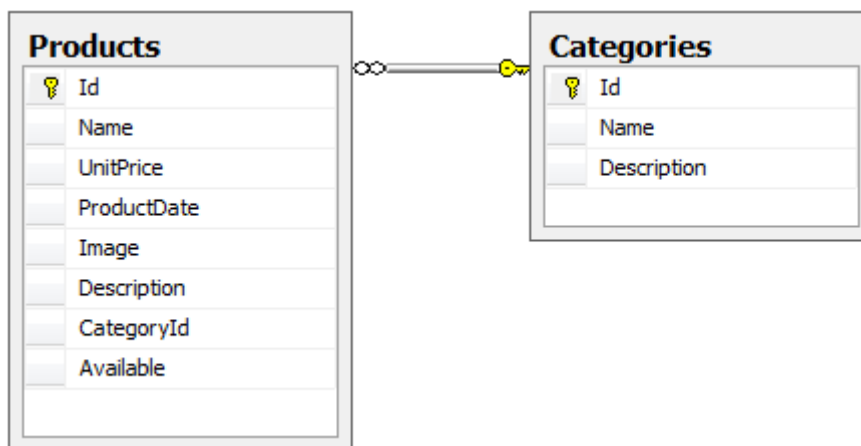
/*
 * Truy vấn theo đặc thù ngôn ngữ SQL
 */
String sql = "SELECT * FROM Courses ORDER BY NewID()";
SQLQuery query = session.createSQLQuery(sql);
query.addEntity(Course.class);
List<Course> courses = query.list();

for(Course course : courses){
    String name = course.getName();
    double fee = course.getSchoolfee();
    System.out.println("+ Name of course: " + name);
    System.out.println("+ Schoolfee of course: " + fee);
}
```

3.3.2.2 Thực thể kết hợp

Dựa vào một thực thể chúng ta có thể truy vấn các thực thể liên quan nhờ vào quan hệ giữa các thực thể đó.

Ví dụ với CSDL Java được giới thiệu trong phần cơ bản, khi biết một hàng hóa chúng ta có thể truy ra chủng loại kết hợp với hàng hóa đó. Hoặc ngược lại, khi biết một chủng loại hàng hóa, chúng ta có thể truy ra các mặt hàng thuộc loại đó.



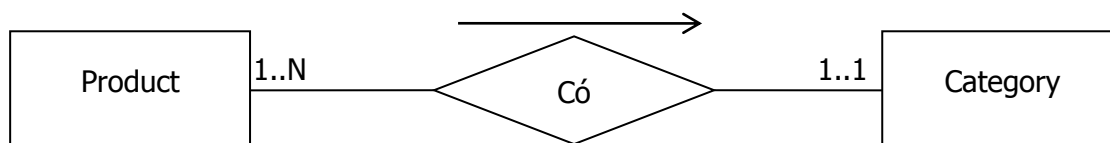
Người ta gọi quan hệ giữa chủng loại (Category) và hàng hóa (Product) là quan hệ 1-nhiều (One-To-Many). Và ngược lại quan hệ giữa hàng hóa và chủng loại là quan hệ nhiều-1 (Many-To-One). Đây là 2 dạng quan hệ được dùng chủ yếu.

Trong thực tế còn tồn tại 2 dạng quan hệ khác là 1-1 (One-To-One) và nhiều-nhiều (Many-To-Many). Cả 2 dạng quan hệ này đều có thể thay thế bằng 2 dạng ở trên.

Vì vậy ở phần này chúng ta chỉ khai thác quan hệ Many2One và One2Many sau đó suy ra 2 dạng quan hệ còn lại.

3.3.2.2.1 Many2One

Trong phần này chúng ta học cách ánh xạ quan hệ nhiều-1 trong Hibernate. Sau đây là mô hình ánh xạ nhiều một trong UML.



Thực thể ánh xạ Category

```
public class Category {
    String id, name, description;

    Getters/setters
}
```

Tập tin ánh xạ thực thể Category.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="nnghiem.entity.Category" table="Categories">
        <id name="id" type="string" column="id"/>
    </class>
</hibernate-mapping>
```



```

        <property name="name" type="string" column="Name" />
        <property name="description" type="string" column="Description" />
    </class>
</hibernate-mapping>

```

Thực thể ánh xạ Product

```

public class Product {
    int id;
    String name, image, description;
    double unitPrice;
    Date productDate;
    boolean available;
    Category category; // thực thể kết hợp

    Getters/setters
}

```

Tập tin ánh xạ thực thể Product.hbm.xml

```

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="nnghiem.entity.Product" table="Products">
        <id name="id" type="int" column="id">
            <generator class="native" />
        </id>
        <property name="name" type="string" column="Name" />
        <property name="unitPrice" type="double" column="UnitPrice" />
        <property name="productDate" type="date" column="ProductDate" />
        <property name="image" type="string" column="Image" />
        <property name="description" type="string" column="Description" />
        <property name="available" type="boolean" column="Available" />
        <many-to-one name="category" column="CategoryId"
            class="nnghiem.entity.Category" cascade="all" not-null="true"/>
    </class>
</hibernate-mapping>

```

Khai báo trong tập tin cấu hình hibernate.cfg.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
    <property name="hibernate.connection.driver_class">
        com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
    <property name="hibernate.connection.url">
        jdbc:sqlserver://localhost:1433;DatabaseName=Java;</property>
    <property name="hibernate.connection.username">sa</property>
    <property name="connection.password">songlong</property>
    <property name="connection.pool_size">100</property>
    <property name="show_sql">>false</property>
    <property name="hbm2ddl.auto">none</property>
    <property name="hibernate.dialect">

```



```
org.hibernate.dialect.SQLServerDialect</property>

<!-- Mapping files -->
<mapping resource="Course.hbm.xml"/>
<mapping resource="Category.hbm.xml"/>
<mapping resource="Product.hbm.xml"/>
</session-factory>

</hibernate-configuration>
```

Trên đây là thực thể ánh xạ và tập tin ánh xạ của cả Category và Product. Bạn chú ý phần được bôi nền vàng trong Product và Product.hbm.xml sẽ thấy cột khóa ngoại (CategoryId) trong Product được khai báo với kiểu là Category chứ không phải là String (kiểu của cột CategoryId). Làm như vậy thì Hibernate sẽ giúp bạn kết hợp Category với Product.

```
...
Category category;
public Category getCategory() {
    return category;
}
public void setCategory(Category category) {
    this.category = category;
}
...
```

```
...
<many-to-one name="category" column="CategoryId"
    class="nnghiem.entity.Category" cascade="all" not-null="true"/>
...
```

Sau đây là ví dụ truy vấn các thực thể Product và hiển thị tên của hàng hóa và tên của chúng loại hàng hóa.

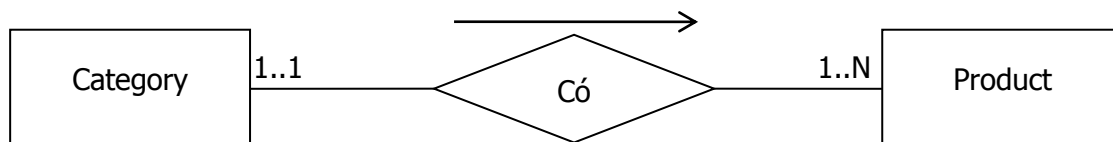
```
Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

String hql = "FROM Product";
Query query = session.createQuery(hql);
List<Product> products = query.list();

for(Product product : products){
    String productName = product.getName();
    String categoryName = product.getCategory().getName();
    System.out.println("+ Name of product: " + productName);
    System.out.println("+ Name of category: " + categoryName);
}
```

3.3.2.2.2 One2Many

Trong phần này chúng ta học cách ánh xạ quan hệ 1-nhiều trong Hibernate. Sau đây là mô hình ánh xạ 1-nhiều trong UML.



Thực thể ánh xạ Category

```

public class Category {
    String id, name, description;

    Collection<Product> products; // các thực thể kết hợp

    Getters/setters
}
  
```

Tập tin ánh xạ thực thể Category.hbm.xml

```

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="nnghiem.entity.Category" table="Categories">
        <id name="id" type="string" column="id"/>
        <property name="name" type="string" column="Name" />
        <property name="description" type="string" column="Description" />
        <set name="products" cascade="all">
            <key column="CategoryId"/>
            <one-to-many class="nnghiem.entity.Product"/>
        </set>
    </class>
</hibernate-mapping>
  
```

Thực thể ánh xạ Product

```

public class Product {
    int id;
    String name, image, description;
    double unitPrice;
    Date productDate;
    boolean available;
    Category category; // thực thể kết hợp

    Getters/setters
}
  
```

Tập tin ánh xạ thực thể Product.hbm.xml

```

<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="nnghiem.entity.Product" table="Products">
        <id name="id" type="int" column="id">
            <generator class="native" />
        </id>
    </class>
</hibernate-mapping>
  
```

```

<property name="name" type="string" column="Name" />
<property name="unitPrice" type="double" column="UnitPrice" />
<property name="productDate" type="date" column="ProductDate" />
<property name="image" type="string" column="Image" />
<property name="description" type="string" column="Description" />
<property name="available" type="boolean" column="Available" />
<many-to-one name="category" column="CategoryId"
              class="nnghiem.entity.Category" cascade="all" not-null="true"/>
</class>
</hibernate-mapping>

```

Trên đây là thực thể ánh xạ và tập tin ánh xạ của cả Category và Product. Bạn chú ý phần được bôi nền vàng trong Category và Category.hbm.xml sẽ thấy một thuộc tính mới products được định nghĩa trong có kiểu Collection<Product>. Làm như vậy thì Hibernate sẽ giúp bạn kết hợp Collection<Product> với Category.

```

...
Collection<Product> products;

public Collection<Product> getProducts() {
    return products;
}
public void setProducts(Collection<Product> products) {
    this.products = products;
}
...

```

```

...
<set name="products" cascade="all">
    <key column="CategoryId"/>
    <one-to-many class="nnghiem.entity.Product"/>
</set>
...

```

Sau đây là ví dụ truy vấn các thực thể Product kết hợp với một thực thể Category.

```

Configuration config = new Configuration().configure();
SessionFactory factory = config.buildSessionFactory();
Session session = factory.openSession();

Category category = session.load(Category.class, 1);
Collection<Product> products = category.getProducts();

```

3.3.2.3 Ánh xạ bằng annotations

Nhưng gì chúng ta đã làm từ trước đến nay, để ánh xạ 1 lớp thực thể với một bảng thì phải

- ✓ Định nghĩa lớp thực thể (Course.java)
- ✓ Khai báo tập tin ánh xạ (Course.hbm.xml)
- ✓ Khai báo vào tập tin cấu hình (hibernate.cfg.xml)

Với Annotation bạn chỉ phải làm 2 việc là Course.java và hibernate.cfg.xml vì việc ánh xạ được thực hiện chung với Course.java. Như vậy sẽ không còn các tập tin ánh xạ xml nữa.

3.3.2.3.1 Thư viện bổ sung

Để sử dụng được với Annotations bạn cần bổ sung các thư viện sau đây

- ✓ hibernate-annotations.jar
- ✓ hibernate-comons-annotations.jar
- ✓ ejb3-persistence.jar

Download phiên bản mới nhất ở đây

- ✓ <http://sourceforge.net/projects/hibernate/files/latest/download?source=files>

3.3.2.3.2 Các JPA annotations

Hibernate sử dụng các annotation của JPA để ánh xạ thực thể. Sau đây là các annotation thường dùng để ánh xạ thực thể:

Annotation	Mô tả	Ví dụ
@Entity	Chỉ ra class là thực thể	@Entity @Table(name="Courses") public class Course{...}
@Table	Ánh xạ lớp thực thể với bảng	
@Id	Chỉ ra cột khóa chính	@Id @GeneratedValue int id;
@GeneratedValue	Chỉ ra cột tự tăng	
@Column	Ánh xạ thuộc tính với cột	@Column(name = "Name") String name
@Temporal	Chỉ ra kiểu dữ liệu chuyển đổi	@Temporal(TemporalType.DATE) Date startDate
@ManyToOne	Ánh xạ quan hệ nhiều-1	@ManyToOne @JoinColumn(name="CategoryId") Category category;
@JoinColumn	Chỉ ra cột kết nối	
@OneToMany	Ánh xạ quan hệ 1-nhiều	@OneToMany(mappedBy="category") Collection<Product> products;

3.3.2.3.3 Thực thể ánh xạ

Sau đây là các lớp thực thể Course, Category và Product được ánh xạ bằng annotation. Bạn chú ý vào những điểm được đánh dấu bằng cách bôi nền vàng.

3.3.2.3.3.1 Course

```

@Entity
@Table(name="Courses")
public class Course {

    @Id
    @GeneratedValue
    @Column(name="Id")
    protected int id;

    @Column(name="Name")

```



```

protected String name;

@Column(name="Schoolfee")
protected double schoolfee;

@Column(name="NoOfLearners")
protected int noOfLearners;

@Temporal(TemporalType.DATE)
@Column(name="StartDate")
protected Date startDate;

@Column(name="Finished")
protected boolean finished;

getters/setters
}

```

3.3.2.3.3.1 Category

```

@Entity
@Table(name="Categories")
public class Category {
    @Id
    @Column(name="Id")
    String id;

    @Column(name="Name")
    String name;

    @Column(name="Description")
    String description;

    @OneToMany(mappedBy="category")
    Collection<Product> products;

    Getters/setters
}

```

3.3.2.3.3.2 Product

```

@Entity
@Table(name="Products")
public class Product {
    @Id
    @GeneratedValue
    @Column(name="Id")
    int id;

    @Column(name="Name")
    String name;

    @Column(name="Image")
    String image;

    @Column(name="Description")
    String description;

    @Column(name="UnitPrice")
    double unitPrice;
}

```



```

@Temporal(TemporalType.DATE)
@Column(name="ProductDate")
Date productDate;

@Column(name="Available")
boolean available;

@ManyToOne
@JoinColumn(name="CategoryId")
Category category;

Getters/setters
}

```

3.3.2.3.3 Cấu hình

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
    <property name="hibernate.connection.driver_class">
        com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
    <property name="hibernate.connection.url">
        jdbc:sqlserver://localhost:1433;DatabaseName=Java;</property>
    <property name="hibernate.connection.username">sa</property>
    <property name="connection.password">songlong</property>
    <property name="connection.pool_size">100</property>
    <property name="show_sql">>false</property>
    <property name="hbm2ddl.auto">none</property>
    <property name="hibernate.dialect">
        org.hibernate.dialect.SQLServerDialect</property>

    <!-- Mapping class -->
    <mapping class = "nnghiem.entity.Course"/>
    <mapping class = "nnghiem.entity.Category"/>
    <mapping class = "nnghiem.entity.Product"/>
</session-factory>
</hibernate-configuration>

```

3.3.2.3.4 Ghi nhớ

- ✓ Nếu bảng và lớp thực thể cùng tên, bạn có thể bỏ annotation @Table
- ✓ Nếu thuộc tính cùng tên với cột của bảng ánh xạ, bạn có thể bỏ annotation @Column

3.3.2.4 Ngôn ngữ truy vấn đối tượng HQL

3.3.2.4.1 Mệnh đề FROM

Truy vấn đơn giản nhất với mệnh đề FROM để lấy tất cả các thuộc tính của thực thể

- ✓ FROM Course
- ✓ FROM Course as c
- ✓ FROM Course c

3.3.2.4.2 Mệnh đề SELECT

Chỉ sử dụng mệnh đề SELECT khi bạn muốn lấy một số thuộc tính của thực thể

- ✓ SELECT name, schoolfee FROM Course
- ✓ SELECT c.name, c.schoolfee FROM Course c

3.3.2.4.3 Hàm tổng hợp số liệu

HQL hỗ trợ các hàm tổng hợp số liệu sau đây

- ✓ avg(...), sum(...), min(...), max(...)
- ✓ count(*)
- ✓ count(...), count(distinct ...), count(all...)

Ví dụ:

```
SELECT avg(schoolfee), max(startDate), min(noOfLearners) FROM Course
```

3.3.2.4.4 Mệnh đề WHERE

Dùng để lọc các thực thể thỏa mãn điều kiện

- ✓ FROM Course WHERE name LIKE 'Nguyễn %'
- ✓ FROM Course WHERE schoolfee BETWEEN 100 AND 250
- ✓ FROM Product WHERE description IS NOT NULL
- ✓ FROM Product p WHERE p.category.id IN (1, 3, 5, 7)

3.3.2.4.5 Hàm và toán tử

3.3.2.4.5.1 Toán tử

- ✓ Số học: +, -, *, /
- ✓ So sánh: =, >=, <=, <>, !=
- ✓ Logic: AND, OR, NOT
- ✓ Đặc biệt: [NOT] IN, [NOT] BETWEEN, IS [NOT] NULL, [NOT] LIKE, IS [NOT] EMPTY, [NOT] MEMBER OF

3.3.2.4.5.2 Hàm

- ✓ concat(..., ...): ghép chuỗi
- ✓ current_date(): lấy ngày, tháng năm
- ✓ current_time(): lấy giờ, phút và giây
- ✓ current_timestamp(): lấy ngày giờ
- ✓ second(...): lấy giây
- ✓ minute(...): lấy phút
- ✓ hour(...): lấy giờ trong ngày
- ✓ day(...): lấy ngày trong tháng
- ✓ month(...): lấy tháng

- ✓ year(...): lấy năm
- ✓ substring(): lấy chuỗi con
- ✓ trim(): cắt bỏ ký tự trắng 2 đầu chuỗi
- ✓ lower(): chuyển in thường
- ✓ upper(): chuyển in hoa
- ✓ length(): lấy độ dài chuỗi
- ✓ locate(): tìm vị trí chuỗi con
- ✓ abs(): lấy giá trị tuyệt đối
- ✓ sqrt(): tính căn bậc 2
- ✓ str(): chuyển số/ngày sang chuỗi
- ✓ cast(... as ...): ép kiểu

3.3.2.4.6 Mệnh đề ORDER BY

Dùng để sắp xếp các thực thể kết quả truy vấn

- ✓ FROM Course ORDER BY startDate DESC, schoolfee

3.3.2.4.7 Mệnh đề GROUP BY...HAVING

Sử dụng để thống kê dữ liệu

- ✓ SELECT AVG(unitPrice), MAX(discount) FROM Product p GROUP BY p.category
- ✓ SELECT AVG(unitPrice), MAX(discount) FROM Product p GROUP BY p.category
HAVING AVG(unitPrice) > 100
- ✓ SELECT AVG(unitPrice), MAX(discount) FROM Product p GROUP BY p.category
HAVING p.category.name LIKE '%Nokia%'

3.3.3 Bài tập

Bài 1: Truy vấn và hiển thị tất cả các khóa học dưới dạng bảng

Bài 2: Tạo form nhập cho phép thêm mới 1 khóa học

Bài 3: Tạo ứng dụng CRUD quản lý khóa học

Bài 4: Tìm kiếm khóa học theo học phí

Bài 5: Phân trang kết quả tìm kiếm

Bài 6: Truy vấn và hiển thị thông tin hàng hóa có tên chủng loại

Bài 7: Hiển thị hàng hóa theo loại

Bài 8: Tạo ứng dụng CRUD quản lý hàng hóa