ĐẠI NAM
UNIVERSITY

CHƯƠNG 7

BẢO MẬT CƠ SỞ DỮ LIỆU

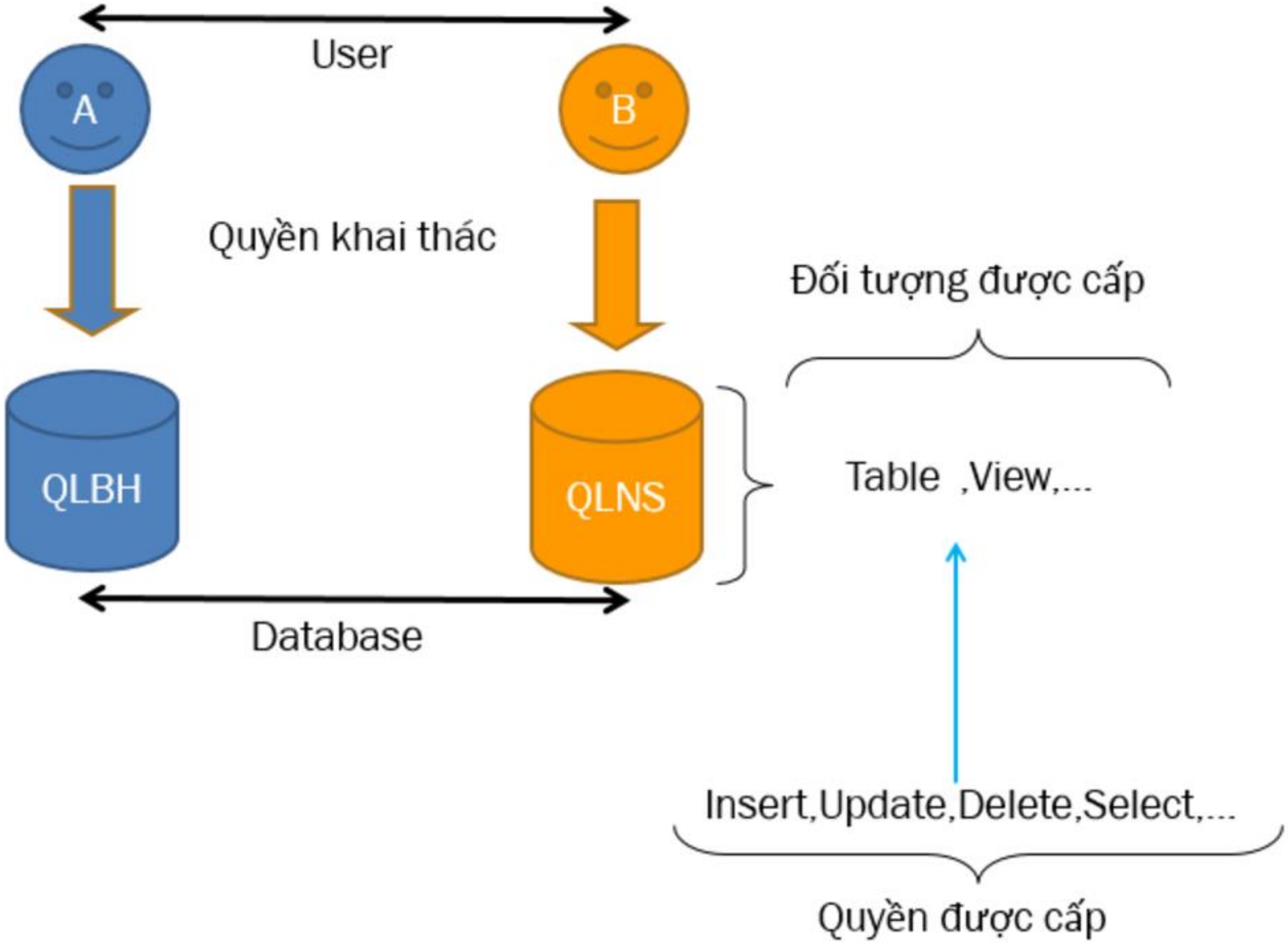
Giảng viên

ThS. Trần Thị Thanh Nhàn

NỘI DUNG

- Mã hóa dữ liệu
- Xác thực truy cập
- Lược đồ
- Vai trò và Ủy quyền
- Theo dõi thay đổi
- Kiểm soát đồng thời

BẢO MẬT TRONG SQL



MÃ HÓA DỮ LIỆU TRONG SQL SERVER

- **Mã hóa dữ liệu:** đảm bảo dữ liệu quan trọng không bị lộ khi bị truy cập trái phép.
- **Các kỹ thuật mã hóa trong SQL Server:**
  - ✓**Transparent Data Encryption:** Bảo vệ dữ liệu khi được lưu trữ trong cơ sở dữ liệu.
  - ✓**Column Encryption:** Bảo vệ dữ liệu nhạy cảm như số thẻ tín dụng, số bảo hiểm xã hội.
  - ✓**Always Encrypted:** Mã hóa dữ liệu mà chỉ có ứng dụng mới có thể giải mã.

Transparent Data Encryption TDE

- **TDE** mã hóa toàn bộ cơ sở dữ liệu khi lưu trữ trên đĩa.
- Dữ liệu được mã hóa khi lưu trữ và giải mã khi truy cập mà không cần thay đổi ứng dụng.

Ví dụ: Tạo chứng chỉ và bật mã hóa TDE cho cơ sở dữ liệu "MyEncryptedDB".

```
-- Tạo chứng chỉ
CREATE CERTIFICATE MyCertificate
WITH SUBJECT = 'TDE Certificate';

-- Tạo cơ sở dữ liệu mã hóa
CREATE DATABASE MyEncryptedDB;
GO

-- Bật mã hóa TDE trên cơ sở dữ liệu
USE MyEncryptedDB;
CREATE DATABASE ENCRYPTION KEY;
ALTER DATABASE MyEncryptedDB SET ENCRYPTION ON;
```

Column Encryption

- **Column Encryption** cho phép mã hóa dữ liệu ở cấp độ cột trong bảng, bảo vệ thông tin nhạy cảm như số thẻ tín dụng, số BHXH, …
- **Cú pháp**:

VD: Mã hóa cột "CreditCardNumber" trong bảng "Customers" sử dụng chứng chỉ và khóa đối xứng.

```
-- Tạo chứng chỉ
CREATE CERTIFICATE MyColumnCert
WITH SUBJECT = 'Column Encryption Certificate';

-- Tạo khóa bảo vệ
CREATE SYMMETRIC KEY MySymmetricKey
WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE MyColumnCert;

-- Mã hóa dữ liệu trong cột
OPEN SYMMETRIC KEY MySymmetricKey DECRYPTION BY CERTIFICATE MyColumnCert;
UPDATE Customers
SET CreditCardNumber = ENCRYPTBYKEY(KEY_GUID('MySymmetricKey'), CreditCardNumber)
WHERE CustomerID = 1;
```

Always Encrypted

- **Always Encrypted** mã hóa dữ liệu ở phía khách hàng.
- Phương pháp bảo mật mạnh mẽ, ngăn ngừa SQL Server và bất kỳ người quản trị cơ sở dữ liệu nào truy cập dữ liệu bảo mật.
- **Cú pháp:**

VD: Mã hóa cột "SSN" trong bảng "Employees" bằng phương pháp Always Encrypted.

```
-- Tạo chứng chỉ để mã hóa
CREATE CERTIFICATE MyAlwaysEncryptedCert
WITH SUBJECT = 'Always Encrypted Certificate';


-- Tạo cột với Always Encrypted
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name NVARCHAR(100),
    SSN NVARCHAR(20) ENCRYPTED WITH (COLUMN_ENCRYPTION_KEY = MyAlwaysEncryptedCert)
);
```

XÁC THỰC TRUY CẬP

- **Xác thực truy cập** là quá trình kiểm tra danh tính của người dùng trước khi cho phép truy cập vào CSDL.
- **Hai phương pháp xác thực trong SQL Server:**
  - ✓ Xác thực Windows Authentication
  - ✓ Xác thực SQL Server Authentication

Windows Authentication

- Sử dụng tài khoản Windows để xác thực người dùng.
- Quản lý quyền truy cập thông qua tài khoản người dùng Windows, phù hợp với các tổ chức sử dụng Active Directory.
- Cú pháp:

```
-- Tạo login sử dụng tài khoản Windows
CREATE LOGIN [Domain\Username] FROM WINDOWS;

-- Tạo user trong cơ sở dữ liệu
USE MyDatabase;
CREATE USER [Domain\Username] FOR LOGIN [Domain\Username];

-- Cấp quyền cho người dùng
ALTER ROLE db_datareader ADD MEMBER [Domain\Username];
```

SQL Server Authentication

- Sử dụng tài khoản SQL Server độc lập, với tên đăng nhập và mật khẩu được lưu trữ trong cơ sở dữ liệu SQL Server.
- **Cú pháp:**

```
-- Tạo login với xác thực SQL Server
CREATE LOGIN mySqlLogin WITH PASSWORD = 'StrongPassword123';

-- Tạo user trong cơ sở dữ liệu
USE MyDatabase;
CREATE USER mySqlLogin FOR LOGIN mySqlLogin;

-- Cấp quyền cho người dùng
ALTER ROLE db_datareader ADD MEMBER mySqlLogin;
```

LƯU Ý

- SQL Server cho phép chuyển đổi giữa Windows Authentication và SQL Server Authentication.
  Cú pháp:

```
ALTER LOGIN mySqlLogin WITH WINDOWS = 'Domain\Username’;
```

- SQL Server có thể hoạt động trong chế độ Mixed Mode, cho phép sử dụng cả xác thực Windows và SQL Server.
  Cú pháp:

```
EXEC sp_configure 'authentication mode’, 2;
```

LƯỢC ĐỒ CƠ SỞ DỮ LIỆU

- Lược đồ cơ sở dữ liệu (Database Schema) là cách thức tổ chức các đối tượng trong cơ sở dữ liệu.
- Lược đồ phân tách các đối tượng cơ sở dữ liệu theo nhóm, từ đó dễ dàng quản lý quyền truy cập của người dùng.
- **Lược đồ chứa các đối tượng như**: tables, view, stored procedures, functions, indexes,constraints,…

TẠO LƯỢC ĐỒ

- Lược đồ là một cách tổ chức các đối tượng CSDL.
- Mỗi lược đồ có thể chứa nhiều đối tượng khác nhau.
- Cú pháp:

```
-- Tạo lược đồ mới
CREATE SCHEMA Sales;

-- Tạo bảng trong lược đồ Sales
CREATE TABLE Sales.Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATETIME,
    CustomerID INT
);
```

```
-- Xem các lược đồ trong cơ sở dữ liệu
SELECT name FROM sys.schemas;
```

GÁN ĐỐI TƯỢNG VÀO LƯỢC ĐỒ

- Khi tạo bảng, view hoặc thủ tục, có thể chỉ định lược đồ mà chúng thuộc về.
- Nếu không chỉ định, đối tượng mặc định sẽ được tạo trong lược đồ dbo.
- Cú pháp:

```sql
-- Tạo bảng trong lược đồ Sales
CREATE TABLE Sales.Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName NVARCHAR(100)
);

-- Tạo view trong lược đồ Sales
CREATE VIEW Sales.CustomerOrders AS
SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Sales.Orders o
JOIN Sales.Customers c ON o.CustomerID = c.CustomerID;
```

THÊM VÀ XÓA LƯỢC ĐỒ

- Có thể thêm hoặc xóa lược đồ trong SQL Server khi cần.
- Cú pháp:

```
-- Xóa một lược đồ, yêu cầu lược đồ không có đối tượng bên trong
DROP SCHEMA Sales;
```

PHÂN QUYỀN VÀ BẢO MẬT VỚI LƯỢC ĐỒ

- Có thể phân quyền truy cập cho người dùng trên một lược đồ.
- Giúp kiểm soát truy cập và làm việc với các đối tượng trong một lược đồ.
- Cú pháp:

```
-- Cấp quyền SELECT cho người dùng trên toàn bộ lược đồ Sales
GRANT SELECT ON SCHEMA::Sales TO myUser;
```

VAI TRÒ VÀ ỦY QUYỀN

- **Vai trò** là một tập hợp các quyền truy cập được cấp cho người dùng hoặc nhóm người dùng, quản lý quyền một cách hiệu quả.
- **Ủy quyền** (Delegation) là việc cấp quyền cho một người dùng hoặc nhóm người dùng để họ có thể cấp quyền cho người dùng khác, giúp tiết kiệm thời gian và dễ dàng quản lý quyền truy cập.

CÁC LOẠI VAI TRÒ TRONG SQL SERVER

- **Vai trò cấp hệ thống**: Là các vai trò được định nghĩa sẵn trong SQL Server để quản lý các quyền truy cập trên toàn bộ hệ thống cơ sở dữ liệu.

VD: sysadmin, db_owner, db_datareader, db_datawriter.

- **Vai trò cấp cơ sở dữ liệu**: Được tạo ra để quản lý quyền truy cập trong một CSDL cụ thể.

VD: db_securityadmin, db_accessadmin, db_backupoperator.

TẠO VÀ QUẢN LÝ VAI TRÒ

- Có thể tạo vai trò mới trong cơ sở dữ liệu và cấp quyền truy cập cho vai trò đó.
- Cú pháp:

```
-- Tạo vai trò mới trong cơ sở dữ liệu
CREATE ROLE MyRole;

-- Cấp quyền SELECT cho vai trò MyRole trên bảng Customers
GRANT SELECT ON dbo.Customers TO MyRole;

-- Gán người dùng user1 vào vai trò MyRole
EXEC sp_addrolemember 'MyRole', 'user1';
```

ỦY QUYỀN TRONG SQL SERVER

- **Ủy quyền** (GRANT) trong SQL Server cho phép người dùng cấp quyền cho người khác để họ có thể thực hiện các thao tác truy cập dữ liệu.
- Một người dùng được cấp quyền WITH GRANT OPTION để ủy quyền quyền truy cập cho người khác.
- Cú pháp:

```
-- Cấp quyền SELECT cho user2 và cho phép user2 ủy quyền quyền này
GRANT SELECT ON dbo.Customers TO user2 WITH GRANT OPTION;


-- Kiểm tra quyền của người dùng user2
SELECT * FROM fn_my_permissions('dbo.Customers', 'OBJECT');
```

THU HỒI QUYỀN VÀ XÓA VAI TRÒ

- Có thể thu hồi quyền từ người dùng hoặc vai trò nếu không còn cần thiết.
- Nếu vai trò không còn sử dụng, có thể xóa vai trò đó.
- Cú pháp:

```
-- Thu hồi quyền SELECT từ vai trò MyRole
REVOKE SELECT ON dbo.Customers FROM MyRole;

-- Xóa vai trò MyRole
DROP ROLE MyRole;
```

KIỂM TRA QUYỀN VÀ VAI TRÒ

- SQL Server cung cấp các chức năng để kiểm tra các quyền mà người dùng có đối với các đối tượng CSDL và vai trò mà họ tham gia.
- Cú pháp:

```
-- Kiểm tra các quyền mà người dùng 'user1' có trên bảng Customers
SELECT * FROM fn_my_permissions('dbo.Customers', 'OBJECT');


-- Kiểm tra vai trò mà người dùng 'user1' tham gia
EXEC sp_helpuser 'user1';
```

THEO DÕI THAY ĐỔI

- **Theo dõi thay đổi (Change Tracking)** là quá trình ghi nhận các thay đổi dữ liệu hoặc cấu trúc CSDL, như thêm mới, sửa đổi, xóa dữ liệu.
- Các phương pháp chính để theo dõi thay đổi:
  - ✓ **Change Data Capture (CDC)**
  - ✓ **Change Tracking (CT)**
  - ✓ **Triggers**
  - ✓ **SQL Server Auditing**

Change Data Capture (CDC)

CDC là một tính năng trong SQL Server giúp ghi nhận và theo dõi các thay đổi trong dữ liệu (INSERT, UPDATE, DELETE) mà không ảnh hưởng đến hiệu suất.

```
-- Bật tính năng CDC cho cơ sở dữ liệu
EXEC sys.sp_cdc_enable_db;

-- Bật CDC cho bảng Customers
EXEC sys.sp_cdc_enable_table
    @source_schema = N'dbo',
    @source_name = N'Customers',
    @role_name = NULL;
```

Change Data Capture (CDC)

- CDC tạo ra các bảng phụ có tên theo mẫu cdc.<table_name>_CT để lưu trữ các thay đổi dữ liệu.
- Cú pháp

```
-- Truy vấn bảng thay đổi của CDC
SELECT *
FROM cdc.dbo_Customers_CT;
```

Change Tracking (CT)

- **CT** là một tính năng nhẹ hơn CDC, chỉ theo dõi xem có thay đổi nào trong dữ liệu mà không lưu trữ chi tiết các giá trị đã thay đổi.
- **Cú pháp:**

```
-- Bật Change Tracking cho cơ sở dữ liệu
ALTER DATABASE MyDatabase
SET CHANGE_TRACKING = ON
(AUTO_CLEANUP = ON, CLEANUP_INTERVAL = 720);

-- Bật Change Tracking cho bảng Customers
ALTER TABLE dbo.Customers
ENABLE CHANGE_TRACKING;
```

Change Tracking

- **Hàm CHANGETABLE** để truy vấn các thay đổi trong bảng đã được theo dõi bằng Change Tracking.
- **Cú pháp**

```
-- Truy vấn các thay đổi trong bảng Customers
SELECT *
FROM CHANGETABLE (CHANGES dbo.Customers, 0) AS CT
WHERE CT.SYS_CHANGE_VERSION > 100;
```

Trigger

Trigger được sử dụng để tự động thực thi khi có các thay đổi (INSERT, UPDATE, DELETE) trên bảng hoặc view.

```
-- Tạo Trigger để ghi nhận các thay đổi (INSERT, UPDATE, DELETE)
CREATE TRIGGER trg_Audit_Customers
ON dbo.Customers
FOR INSERT, UPDATE, DELETE
AS
BEGIN
    -- Ghi nhận thay đổi vào bảng Audit
    INSERT INTO dbo.Audit_Customers
    SELECT * FROM inserted;  -- Dữ liệu mới sau thay đổi
    INSERT INTO dbo.Audit_Customers
    SELECT * FROM deleted;   -- Dữ liệu cũ trước thay đổi
END;
```

SQL Server Auditing

- **SQL Server Auditing** là tính năng cho phép theo dõi và ghi lại các sự kiện truy cập và thay đổi trong cơ sở dữ liệu, đặc biệt hữu ích trong môi trường yêu cầu bảo mật cao.
- **Cú pháp:**

```
-- Tạo một Audit mới
CREATE SERVER AUDIT MyAudit
    TO FILE (FILEPATH = 'C:\AuditLogs\');

-- Tạo một Audit Specification để theo dõi các truy cập
CREATE SERVER AUDIT SPECIFICATION MyAuditSpec
    FOR SERVER AUDIT MyAudit
    ADD (SUCCESSFUL_LOGIN_GROUP),
    ADD (FAILED_LOGIN_GROUP);
```

KIỂM SOÁT ĐỒNG THỜI

- **Kiểm soát đồng thời** trong SQL Server là phương pháp quản lý truy cập đồng thời đến cơ sở dữ liệu, giúp tránh các xung đột và đảm bảo tính nhất quán của dữ liệu.

- **Các phương pháp chính**:
  - ✓ **Locks**
  - ✓ **Isolation Levels**
  - ✓ **Tranzaction Management**
  - ✓ **Deadlock Handling**

Locks

- **Khóa** giúp ngăn chặn các giao dịch khác truy cập dữ liệu khi một giao dịch đang thực hiện thao tác.
- **Các loại khóa trong SQL Server**:
  - ✓Shared Lock (S)
  - ✓Exclusive Lock (X)
  - ✓Update Lock (U)
- **Cú pháp:**

```
BEGIN TRANSACTION;

-- Khóa Exclusive (X) trên bảng Customers
SELECT * FROM dbo.Customers WITH (XLOCK);

-- Cập nhật dữ liệu
UPDATE dbo.Customers
SET LastName = 'Nguyen'
WHERE CustomerID = 1;

COMMIT;
```

Isolation Levels

- **Isolation Level** xác định mức độ cách ly giữa các giao dịch đồng thời, giúp kiểm soát cách thức các giao dịch truy cập dữ liệu và ảnh hưởng đến nhau.
- **Các cấp độ Isolation phổ biến trong SQL Server:**
  - ✓ Read Uncommitted
  - ✓ Read Committed
  - ✓ Repeatable Read
  - ✓ Serializable

```
-- Cấu hình Isolation Level là READ COMMITTED
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

-- Cấu hình Isolation Level là SERIALIZABLE
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Transaction Management

Quản lý giao dịch đảm bảo tính toàn vẹn của dữ liệu trong môi trường đa giao dịch. Giao dịch trong SQL Server bắt đầu bằng lệnh BEGIN TRANSACTION, và có thể được kết thúc bằng COMMIT hoặc ROLLBACK tùy thuộc vào kết quả của giao dịch.

```
BEGIN TRANSACTION;


-- Thực hiện các thao tác SQL
UPDATE dbo.Customers
SET LastName = 'Nguyen'
WHERE CustomerID = 1;


COMMIT; -- Hoặc ROLLBACK nếu có lỗi
```

XỬ LÝ DEADLOCK

- **Deadlock** xảy ra khi hai hoặc nhiều giao dịch chờ nhau giải phóng khóa mà cần để tiếp tục, dẫn đến tình trạng treo mà không thể tiếp tục.
- **Cú pháp:**

```
BEGIN TRY
    BEGIN TRANSACTION;

    -- Thực hiện các thao tác có thể gây deadlock
    UPDATE dbo.Customers SET LastName = 'Tran' WHERE CustomerID = 1;

    COMMIT;
END TRY
BEGIN CATCH
    -- Xử lý lỗi deadlock
    IF ERROR_NUMBER() = 1205
    BEGIN
        PRINT 'Deadlock detected. Retrying the transaction...';
        ROLLBACK;
        -- Thực hiện lại giao dịch nếu cần
    END
END CATCH;
```

THỰC HÀNH


Q&A

BÀI TẬP VỀ NHÀ

- ✓ Câu hỏi trắc nghiệm
- ✓ Bài tập