ÔN TẬP CUỐI KỲ

1. **CÁC LỆNH TẠO USER, PHÂN QUYỀN, HỦY QUYỀN**
2. **GIÁM SÁT**

# Khái quát về Database Auditing

1. **Định nghĩa:**

* Auditing là hoạt động giám sát và ghi lại…. Được dựa trên các hoạt động cá nhân như thực hiện câu lệnh SQL, hay dựa trên sự kết hợp các yếu tố bao gồm tên, ứng dụng, thời gian,…Các chính sách bảo mật có thể dẫn đến việc audit khi những phần tử cụ thể trong CSDL Oracle bị truy cập hay thay thế.
* Auditing nhìn chung được sử dụng:
* Cho phép giải trình những hành động hiện tại tham gia vào một schema, bảng, dòng riêng biệt, hay một nội dung cụ thể nào đó.
* Ngăn cản user khỏi hành động không thích hợp dựa trên trách nhiệm phải giải trình đó.
* Điều tra các hoạt động đáng ngờ. Ví dụ, nếu một user không được phép đang xóa dữ liệu từ một bảng nào đó thì người quản trị bảo mật sẽ ghi lại tất cả những kết nối CDSL và tất cả những hành động xóa các dòng từ bảng trong CSDL dù thành công hay không thành công.
* Thông báo cho người giám sát rằng có user bất hợp phát đang thao tác hay xóa dữ liệu hay user có nhiều quyền hệ thống hơn sự cho phép.
* Giám sát và thu thập dữ liệu về các hoạt động CSDL cụ thể. Ví dụ, người quản trị CSDL có thể thu thập thống kê về thông tin các bảng đang được update, hay bao nhiêu users cùng trung cập vào thời điểm cực đỉnh.

1. **Các kiểu giám sát (Types of Auditing)**

* Oracle cho phép giám sát theo 2 lựa chọn tập trung hoặc mở rộng.
* Sự thực thi câu lệnh thành công, hoặc không thành công, hoặc cả hai.
* Mỗi lần thực thi câu lệnh trong mỗi session của user, hay bất kì khi nào mà câu lệnh được thực thi.
* Hoạt động của tất cả các user hay của một user cụ thể nào đó.
* Có bốn kiểu giám sát:
* **Statement auditing:** chia thành hai nhóm
  + - Câu lệnh DDL: Ví dụ AUDIT TABLE giám sát tất cả các câu lệnh CREATE và DROP TABLE.
    - Câu lệnh DML: Ví dụ AUDIT SELECT TABLE giám sát tất cả câu lệnh SELECT trên bảng và trên view
* **Privilege auditing:** Kiểm tra việc sử dụng quyền hệ thống, ví dụ AUDIT CREATE TABLE. Privilege auditing được chú trọng hơn statement auditing vì nó chỉ kiểm tra việc sử dụng một số quyền nhất định. Có thể đặt privilege auditing giám sát những user được lựa chọn hay giám sát mọi user.
* **Schema object auditing:** Kiểm tra câu lệnh cụ thể trên đối tượng schema cụ thể, ví dụ AUDIT SELECT ON employees. (Rất được chú trọng). Schema object auditing luôn áp dụng cho tất cả các user.
* **Fine-grained auditing:** Kiểm tra dữ liệu truy xuất và các hoạt động dựa trên nội dung của dữ liệu đó. Ví dụ: Sử dụng DBMS\_FGA, người quản trị bảo mật tạo ra một chính sách kiểm tra trên một bảng. Nếu bất kì dòng nào trả về từ câu lệnh DML thỏa điều kiện kiểm tra thì một mục về sự kiện kiểm tra sẽ được chèn vào trong audit trail.

1. **Audit Records và Audit Trails:**

* Những thông tin được audit sẽ được lưu trong data dictionary table, gọi là **database audit trail**, hoặc lưu trong operating system files, gọi là **operating system audit trail.**
* **Bản ghi Audit (Audit trail records)**
* Chứa những loại thông tin khác nhau, phụ thuộc vào những sự kiện được giám sát và tập các lựa chọn giám sát. Thông tin sau đây được bao gồm trong mỗi bản ghi audit:
* Database user name (DATABASE USER)
* Operating system login user name (CLIENT USER)
* Instance number (không có trong Operation System…)
* Process identifier
* Session identifier
* Terminal identifier
* Name of the schema object accessed
* Operation performed or attempted (ACTION)
* Completion code of the operation
* Date & time stamp in UTC format ( không có trong Operation System Audit Trail)
* System privileges used (PRIVILEGE)
* **Chú ý:** Audit trail không lưu thông tin về giá trị của dữ liệu dù nó liên quan đến trong câu lệnh được giám sát. Ví dụ, giá trị dữ liệu mới và giá trị dữ liệu cũ của hàng được update không được lưu lại khi câu lệnh UPDATE được giám sát. Tuy vậy, đối với phương pháp fine-grained auditing có khác.
* **Operating System Audit Trail**
* Oracle cho phép bản ghi dấu audit (audit trail records) được trực tiếp ghi vào operating system audit trail nếu hệ điều hành tạo một audit trail sẵn cho Oracle. Nếu không thì bản audit sẽ được ghi vào file bên ngoài CSDL, với định dạng tương tự như các file dấu tích Oracle (Oracle trace) khác.
* Oracle cho phép một hoạt động nào đó mà luôn bị giám sát được tiếp tục, thậm chí khi mà operating system audit trail ( hay file hệ điều hành chứa bản ghi audit) không được phép ghi lại bản ghi audit do nó bị đầy. Tuy nhiên, chú ý rằng cấu hình auditing để sử dụng **database audit trail** loại bỏ được điểm yếu này, bởi vì hệ CSDL Oracle ngăn ngừa sự kiện được audit khỏi xảy ra nếu audit trail không thể tiếp nhận bản ghi database audit cho câu lệnh đó.

# Quản lí Standard Audit Trail

1. **Kích hoạt Standard Auditing**

* Bất cứ database user hợp pháp nào cũng có thể thiết lập lựa chọn giám sát đối với câu lệnh, quyền và đối tượng bất cứ khi nào. Tuy nhiên hệ CSDL Oracle không sinh thông tin audit cho Standart database audit trail trừ khi CSDL giám sát được kích hoạt. Người quản trị bảo mật thường có trách nhiệm điều khiển việc giám sát này.
* Auditing là chức năng mặc định trong Oracle server. Các tham số khởi tạo ban đầu ảnh hưởng đến hành vi của nó có thể được xem bằng cách sử dụng câu lệnh.

SQL> SHOW PARAMETER AUDIT

NAME TYPE VALUE

-------------------- -------- ----------------------

audit\_file\_dest string C:\ORACLEXE\APP\ORACLE\

ADMIN\XE\ADUMP

audit\_sys\_operations boolean FALSE

audit\_trail string NONE

* Chức năng Audit mặc định bị bất hoạt, nhưng có thể kích hoạt nó bằng cách thiết lập giá trị cho tham số AUDIT\_TRAIL

AUDIT\_TRAIL = { none | os | db | db,extended | xml | xml,extended }

Trong đó:

* none or false – chức năng giám sát bị bất hoạt.
* db or true – Bật chức năng giám sát và các bản ghi giám sát được sẽ được lưu trong database audit trail (SYS.AUD$).
* xml- Bật chức năng giám sát với các bản ghi giám sát được sẽ được lưu như file OS có định dạng XML.
* os- Bật chức năng giám sát với các bản ghi giám sát được ghi vào operating system audit trail.
* Tham số AUDIT\_SYS\_OPERATIONS dùng để kích hoạt hay bất hoạt giám sát các hoạt động của những user kết nối vào hệ thống với quyền SYSDBA hay SYSOPER, bao gồm user SYS. Khi đó tất cả các bản ghi giám sát được ghi vào OS audit trail.
* Tham số AUDIT\_FILE\_DEST đặc tả đường dẫn cho audit trail khi các lựa chọn OS, XML, EXTENDED được sử dụng. Nó cũng là vị trí lưu các bản ghi giám sát khi tham số AUDIT\_SYS\_OPERATIONS = true
* Để kích hoạt chức năng giám sát, làm theo các bước sau

SQL> ALTER SYSTEM SET audit\_trail=db SCOPE=SPFILE;

System altered.

SQL> SHUTDOWN

Database closed.

Database dismounted.

ORACLE instance shut down.

SQL> STARTUP

ORACLE instance started.

Database mounted.

Database opened.

SQL>

1. **Kích hoạt lựa chọn Standard Auditing**

* Để sử dụng lệnh AUDIT thiết lập lựa chọn về câu lệnh và quyền thì bạn nhất thiết phải có quyền AUDIT SYSTEM. Còn để thiết lập các lựa chọn giám sát đối tượng bạn phải làm chủ đối tượng bị giám sát hay có quyền AUDIT ANY.
* Lệnh AUDIT thiết lập lựa chọn giám sát câu lệnh và quyền có thể bao gồm mệnh đề BY để cụ thể danh sách những user hay application proxy để giới hạn tầm vực của câu lệnh và lựa chọn giám sát quyền.
  + - BY SESSION/ BY ACCESS
* BY SESSION:

Ghi một bản ghi đơn cho tất cả các câu lệnh SQL cùng loại thực thi trên cùng những đối tượng schema trong cùng một session. Tuy nhiên nếu sử dụng operating system trail cho theo dấu giám sát (khi đó tham số AUDIT\_FILE\_DEST được gán giá trị OS), thì CSDL sẽ viết nhiều mục vào file theo dấu ngay cả khi bạn sử dụng mệnh đề BY SESSION.

* BY ACCESS:

Ghi một bản ghi cho mỗi lần truy xuất. Nếu đặc tả lựa chọn câu lệnh hay quyền hệ thống mà giám sát câu lệnh DDL thì CSDL sẽ tự động giám sát theo BY ACCESS không quan tâm bạn sử dụng mệnh đề BY SESSION hay BY ACCESS. Ngoài ra thì nếu không đặc tả thì BY SESSION là mặc định.

* + - WHENEVER SUCCESSFUL/ WHENEVER NOT SUCCESSFUL
* WHENEVER SUCCESSFUL giám sát những lệnh thành công.
* WHENEVER NOT SUCCESSFUL giám sát những lệnh thất bại hay kết quả lỗi. Nếu loại bỏ hai mệnh đề này thì Oracle cũng sẽ ghi lại sự giám sát không quan tâm câu lệnh có thành công hay không.

1. **Bất hoạt lựa chọn Standard Auditing**

* Câu lệnh NOAUDIT để tắt các lựa chọn giám sát của Oracle 10G.
* Mệnh đề WHENEVER để tắt các giám sát đối với các câu lệnh được thực hiện thành công hay không thành công. Nếu không sử dụng mệnh đề đó thì chức năng giám sát sẽ tắt cả đối với trường hợp thành công hay thất bại.
* Mệnh đề BY SESSION/BY ACCESS không được hỗ trợ trong câu lệnh NOAUDIT.

1. **Điều khiển sự phát triển và kích cỡ của Standard Audit Trail**

* Nếu audit trail đầy dẫn tới không một bản ghi giám sát nào được ghi thêm vào thì những câu lệnh AUDIT không thể thực thi thành công cho tới khi audit trail trống trở lại. Do đó, người quản trị bảo mật phải điều khiển sự phát triển và kích cỡ của audit trail.
* Khi chức năng giám sát được kích hoạt và các bản ghi giám sát được sinh ra thì dung lượng của audit trail phụ thuộc hai yếu tố:
* Số lựa chọn giám sát được sử dụng.
* Tần số thực hiện các câu lệnh được giám sát.
* Để điều khiển sự phát triển của audit trail, bạn có thể sử dụng phương pháp:
* Kích hoạt và bất hoạt giám sát CSDL. Nếu nó được kích hoạt thì các bản ghi giám sát được sinh ra và lưu trữ trong audit trail. Nếu nó bất hoạt thì các bản ghi sẽ không được sinh ra.
* Chọn lọc kĩ những lựa chọn giám sát được kích hoạt. Nếu nhiều lựa chọn giám sát được kích hoạt thì những bản ghi giám sát không cần thiết có thể làm đầy audit trail.
* Quản lí chặt khả năng giám sát đối tượng. Điều đó có thể được thực hiện bằng hai cách khác nhau:
  + - Nhà quản trị bảo mật làm chủ tất cả các đối tượng và quyền hệ thống AUDIT ANY không bị cấp cho bất kì một user nào khác.
    - Tất cả các đối tượng chứa trong những schema mà không tương ứng với database user thực sự (user đó không được cấp quyền CREATE SESSION) và người quản trị bảo mật là user duy nhất có quyền AUDIT ANY.
* Xóa một số bản ghi trong audit trail để vừa giải phóng vùng nhớ vừa làm thuận tiện cho việc quản lí audit trail.

Ví dụ:

Xóa toàn bộ bản ghi trong audit trail:

DELETE FROM SYS.AUD$;

Xóa toàn bộ bản ghi được sinh ra do kết quả của việc giám sát bảng emp:

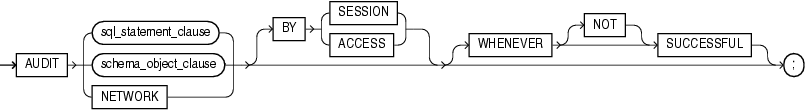
DELETE FROM SYS.AUD$

WHERE obj$name = ‘EMP’;

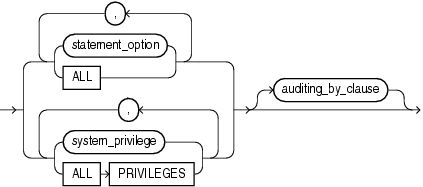
Chú ý: Chỉ có user SYS (user có quyền DELETE ANY TABLE), hay user đượcSYS gán quyền DELETE trên SYS.AUD$ mới có thể thực hiện những câu lệnh trên.

# Cú pháp

1. **Lý thuyết**
2. audit::=

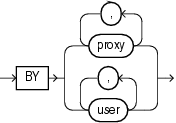


1. sql\_statement\_clause::=



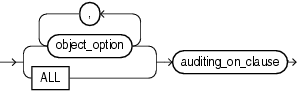
Chú ý: Oracle khuyên rằng nên đặc tả lựa chọn quyền hệ thống và câu lệnh để giám sát hơn là nêu chung qua các role hoặc shortcuts.

1. auditing\_by\_clause::=



Giám sát chỉ những câu lệnh SQL gọi bởi những user cụ thể. Nếu không sử dụng mệnh đề này thì Oracle sẽ giám sát câu lệnh của tất cả user.

1. schema\_object\_clause::=

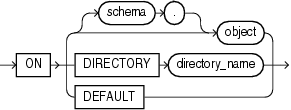


Giám sát các hoạt động trên những đối tượng schema.

Object\_option: đặc tả hoạt động cụ thể cho việc giám sát. Ví dụ như ALTER, COMMENT, AUDIT, DELETE, EXECUTE, GRANT, INSERT, READ,…

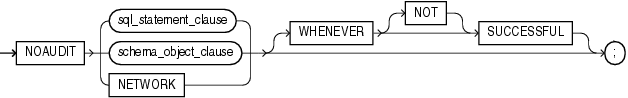
ALL: là một shortcut tương ứng với tất cả các object options cho các loại đối tượng.

1. auditing\_on\_clause::=



* auditing\_on\_clause: đặc tả đối tượng schema riêng biệt được giám sát.
* ON DEFAULT: thiết lập những lựa chọn đối tượng cụ thể trở thành những lựa chọn đối tượng mặc định cho những đối tượng được tạo ra sau đó. Sau khi thiết lập những lựa chọn giám sát mặc định, bất kì đối tượng nào được tạo ra sau đó thì được giám sát tự động với những lựa chọn đó. Những lựa chọn giám sát mặc định cho một view luôn là sự kết hợp của lựa chọn giám sát cho bảng cơ sở của view đó. Có thể xem lựa chọn giám sát mặc định hiện tại bằng cách truy vấn ALL\_DEF\_AUDIT\_OPTS.
* Khi thay đổi lựa chọn giám sát mặc định, lựa chọn giám sát cho những đối tượng được tạo lúc trước vẫn giữ nguyên. Bạn có thể thay đổi lựa chọn giám sát cho đối tượng đã tồn tại chỉ bằng đặt tả đối tượng trong mệnh đề ON của câu lệnh AUDIT.

1. noaudit::=



1. **Thực hành**
2. **Giám sát câu lệnh SQL liên quan đến ROLE**

* Giám sát tất cả các câu lệnh SQL liên quan đến ROLE (create, alter, drop, set) không quan tâm câu lệnh được thực hiện thành công hay không:

AUDIT ROLE;

* Giám sát tất cả câu lệnh liên quan đến ROLE mà thực hiện thành công:

AUDIT ROLE WHENEVER SUCCESSFUL;

1. **Giám sát truy vấn và update câu lệnh SQL**

* Giám sát cho bất cứ câu lệnh nào truy vấn hay update bất kì bảng nào:

AUDIT SELECT TABLE, UPDATE TABLE;

* Giám sát câu lệnh phát ra bởi user hr và oe mà truy vấn hay update một bảng hoặc view nào đó:

AUDIT SELECT TABLE, UPDATE TABLE BY hr, oe;

1. **Giám sát xóa**

* Giám sát câu lệnh sử dụng quyền hệ thống DELETE ANY TABLE:

AUDIT DELETE ANY TABLE;

1. **Giám sát câu lệnh liên quan tới Directories**

* Giám sát câu lệnh sử dụng quyền hệ thống CREATE ANY DIRECTORY:

AUDIT CREATE ANY DIRECTORY;

* Giám sát lệnh CREATE DIRECTORY (và DROP DIRECTORY) mà không sử dụng quyền hệ thống CREATE DIRECTORY:

AUDIT DIRECTORY;

* Giám sát mỗi câu lệnh đọc file từ đường dẫn bfile\_dir:

AUDIT READ ON DIRECTORY bfile\_dir;

1. **Giám sát truy vấn trên bảng**

* Giám sát cho mỗi câu SQL truy vấn tới bảng employees trong schema hr:

AUDIT SELECT ON hr.employees;

* Giám sát cho mỗi câu truy vấn tới bảng employees trong schema hr và kết quả đó trong CSDL Oracle bị lỗi:

AUDIT SELECT ON hr.employees

WHENEVER NOT SUCCESSFUL;

1. **Giám sát chèn và update trên bảng**

* Giám sát cho mỗi câu lệnh chèn và update một hàng trong bảng customers của schema oe:

AUDIT INSERT, UPDATE

ON oe.customers;

1. **Thiết lập mặc định cho lựa chọn giám sát**

* Đặc tả lựa chọn giám sát mặc định cho các đối tượng được tạo ra trong tương lai:

AUDIT ALTER, GRANT, INSERT, UPDATE, DELETE

ON DEFAULT;

* Bất kì đối tượng nào được tạo ra sau đó sẽ tự động bị giám sát với đặc tả được lựa chọn đó (trong trường hợp chức năng giám sát được kích hoạt)
* Nếu tạo ra một bảng thì Oracle tự động giám sát các câu lệnh ALTER, GRANT, INSERT, UPDATE, DELETE liên quan đến bảng này.
* Nếu tạo ra một view thì Oracle sẽ tự động giám sát các câu lệnh GRANT, INSERT, UPDATE, DELETE liên quan đến view này.
* Nếu tạo ra một procedure, package, hay function thì Oracle sẽ tự động giám sát các câu lệnh ALTER hay GRANT liên quan đến nó.

1. **Tắt giám sát**

* Tắt giám sát trên câu lệnh:

NOAUDIT ALL;

* Tắt giám sát trên quyền:

NOAUDIT ALL PRIVILEGES;

Chú ý: để bất hoạt giám sát câu lệnh và quyền thì phải có quyền hệ thống AUDIT SYSTEM.

* Tắt giám sát trên đối tượng:

NOAUDIT DELETE

ON emp;

NOAUDIT SELECT, INSERT, DELETE

ON jward.dept;

* Để tắt giám sát trên đối tượng cụ thể thì bạn phải là chủ của đối tượng đó. Để tắt giám sát trên đối tượng của một đối tượng thuộc schema của user khác hay tắt giám sát mặc định trên đối tượng thì bạn phải có quyền hệ thống AUDIT ANY.

1. **Các view của Audit Trail**

* Các bản ghi giám sát được lưu trong bảng AUD$ trong schema của SYS. Nội dung của nó có thể được xem trực tiếp hoặc qua các view.

SELECT view\_name

FROM dba\_views

WHERE view\_name LIKE 'DBA%AUDIT%'

ORDER BY view\_name;

VIEW\_NAME

------------------------------

DBA\_AUDIT\_EXISTS

DBA\_AUDIT\_OBJECT

DBA\_AUDIT\_POLICIES

DBA\_AUDIT\_POLICY\_COLUMNS

DBA\_AUDIT\_SESSION

DBA\_AUDIT\_STATEMENT

DBA\_AUDIT\_TRAIL

DBA\_COMMON\_AUDIT\_TRAIL

DBA\_FGA\_AUDIT\_TRAIL

DBA\_OBJ\_AUDIT\_OPTS

DBA\_PRIV\_AUDIT\_OPTS

DBA\_REPAUDIT\_ATTRIBUTE

DBA\_REPAUDIT\_COLUMN

DBA\_STMT\_AUDIT\_OPTS

14 rows selected.

1. **MÃ HÓA**

**Encryption**

USE master

GO

CREATE DATABASE EncryptTest

go

USE EncryptTest

GO

CREATE TABLE TestTable (FirstCol INT, SecondCol VARBINARY(256))

go

/\* Create Database Master Key \*/

CREATE MASTER KEY ENCRYPTION

BY PASSWORD = 'SQLAuthority'

GO

/\* Create Encryption Certificate \*/

CREATE CERTIFICATE EncryptTestCert

WITH SUBJECT = 'SQLAuthority'

GO

/\* Create Symmetric Key \*/

CREATE SYMMETRIC KEY TestTableKey

WITH ALGORITHM = TRIPLE\_DES ENCRYPTION

BY CERTIFICATE EncryptTestCert

GO

OPEN SYMMETRIC KEY TestTableKey DECRYPTION BY CERTIFICATE EncryptTestCert

GO

--UPDATE TestTable

--SET EncryptSecondCol = ENCRYPTBYKEY(KEY\_GUID('TestTableKey'),SecondCol)

--GO

INSERT INTO TestTable values(1,ENCRYPTBYKEY(KEY\_GUID('TestTableKey'),'Hello'))

INSERT INTO TestTable values(2,ENCRYPTBYKEY(KEY\_GUID('TestTableKey'),'123456'))

INSERT INTO TestTable values(3,ENCRYPTBYKEY(KEY\_GUID('TestTableKey'),'gogogo'))

go

SELECT \* FROM TestTable

GO

/\* Decrypt the data of the SecondCol  \*/

OPEN SYMMETRIC KEY TestTableKey DECRYPTION BY CERTIFICATE EncryptTestCert

SELECT CONVERT(VARCHAR(50),DECRYPTBYKEY(SecondCol)) AS DecryptSecondCol

FROM TestTable

GO

CLOSE SYMMETRIC KEY TestTableKey

GO

[APRIL 28, 2009](http://blog.sqlauthority.com/2009/04/28/sql-server-introduction-to-sql-server-encryption-and-symmetric-key-encryption-tutorial-with-script/) BY [PINAL DAVE](http://blog.sqlauthority.com/author/pinaldave/)

# SQL SERVER – Introduction to SQL Server Encryption and Symmetric Key Encryption Tutorial with Script

SQL Server 2005 and SQL Server 2008 provide encryption as a new feature to protect data against hackers’ attacks. Hackers might be able to penetrate the database or tables, but owing to encryption they would not be able to understand the data or make use of it. Nowadays, it has become imperative to encrypt crucial security-related data while storing in the database as well as during transmission across a network between the client and the server.

Encryption hierarchy is marked by three-level security. These three levels provide different mechanisms for securing data across networks and local servers. Different levels of hierarchies allow multiple instances of services (e.g., SQL Server Services) to run on one physical server.

* **Windows Level** – Highest Level – Uses Windows DP API for encryption
* **SQL Server Level**– Moderate Level – Uses Services Master Key for encryption
* **Database Level** – Lower Level – Uses Database Master Key for encryption

There are two  kinds of keys used in encryption:

* **Symmetric Key** – In Symmetric cryptography system, the sender and the receiver of a message share a single, common key that is used to encrypt and decrypt the message. This is relatively easy to implement, and both the sender and the receiver can encrypt or decrypt the messages.
* **Asymmetric Key** – Asymmetric cryptography, also known as Public-key cryptography, is a system in which the sender and the receiver of a message have a pair of cryptographic keys – a public key and a private key – to encrypt and decrypt the message. This is a relatively complex system where the sender can use his key to encrypt the message but he cannot decrypt it. The receiver, on the other hand, can use his key to decrypt the message but he cannot encrypt it. This intricacy has turned it into a resource-intensive process.

Yet another way to encrypt data is through certificates. A public key certificate is a digitally signed statement that binds the value of a public key to the identity of the person, device, or service that holds the corresponding private key. A Certification Authority (CA) issues and signs certifications. [**Download complete script here**](http://www.pinaldave.com/sql-script-bank/script-downloads-1/sql-server-introduction-to-sql-server-encryption-and-symmetric-key-encryption-tutorial-with-script/).

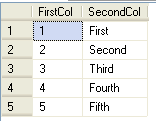
Please create a sample database that we will be use for testing Encryption. There are two different kinds of encryptions available in SQL Server:

* **Database Level** – This level secures all the data in a database. However, every time data is written or read from database, the whole database needs to be decrypted. This is a very resource-intensive process and not a practical solution.
* **Column (or Row) Level** – This level of encryption is the most preferred method. Here, only columns containing important data should be encrypted; this will result in lower CPU load compared with the whole database level encryption. If a column is used as a primary key or used in comparison clauses (WHERE clauses, JOIN conditions) the database will have to decrypt the whole column to perform operations involving those columns.

Let’s go over a simple instance that demonstrates the encryption and the decryption process executed with Symmetric Key and Triple DES encryption algorithm.  
 **/\* Create Database  \*/  
USE master  
GO  
CREATE DATABASE EncryptTest  
ON PRIMARY ( NAME = N'EncryptTest', FILENAME = N'C:\EncryptTest.mdf')  
LOG ON ( NAME = N'EncryptTest\_log', FILENAME =N'C:\EncryptTest\_log.ldf')  
GO**

First, let’s create a sample table and then populate it with sample data. We will now encrypt one of the two columns of the table.  
 **/\* Create table and insert data in the table \*/  
USE EncryptTest  
GO  
CREATE TABLE TestTable (FirstCol INT, SecondCol VARCHAR(50))  
GO  
INSERT INTO TestTable (FirstCol, SecondCol)  
SELECT 1,'First'  
UNION ALL  
SELECT 2,'Second'  
UNION ALL  
SELECT 3,'Third'  
UNION ALL  
SELECT 4,'Fourth'  
UNION ALL  
SELECT 5,'Fifth'  
GO  
/\* Check the content of the TestTable \*/  
USE EncryptTest  
GO  
SELECT \*  
FROM TestTable  
GO**

The preceding code will return the result depicted in the subsequent figure.



Result of the SQL query

Every database can have one master key. Database master key is a symmetric key used to protect the private keys of certificates and asymmetric keys present in the database. It uses Triple DES algorithm together with user-provided password to encrypt the keys.  
 **/\* Create Database Master Key \*/  
USE EncryptTest  
GO  
CREATE MASTER KEY ENCRYPTION  
BY PASSWORD = 'SQLAuthority'  
GO**

Certificates are used to safeguard encryption keys, which are used to encrypt data in the database. SQL Server 2005 has the capability to generate self-signed X.509 certificates.  
 **/\* Create Encryption Certificate \*/  
USE EncryptTest  
GO  
CREATE CERTIFICATE EncryptTestCert  
WITH SUBJECT = 'SQLAuthority'  
GO**

The symmetric key can be encrypted by using various options such as certificate, password, symmetric key, and asymmetric key. A number of different algorithms can be employed for encrypting key. The supported algorithms are DES, TRIPLE\_DES, RC2, RC4, RC4\_128, DESX, AES\_128, AES\_192, and AES\_256.  
 **/\* Create Symmetric Key \*/  
USE EncryptTest  
GO  
CREATE SYMMETRIC KEY TestTableKey  
WITH ALGORITHM = TRIPLE\_DES ENCRYPTION  
BY CERTIFICATE EncryptTestCert  
GO**

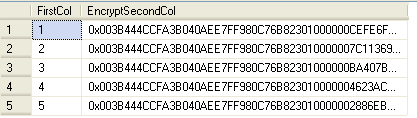
Now add a column of type varbinary to the original table, which will store the encrypted value for the SecondCol.  
 **/\*  Encrypt Data using Key and Certificate  
Add Columns which will hold the encrypted data in binary \*/  
USE EncryptTest  
GO  
ALTER TABLE TestTable  
ADD EncryptSecondCol VARBINARY(256)  
GO**

Before the key is used, it needs to be decrypted using the same method that was used for encrypting it. In our example, we have used a certificate for encrypting the key. Because of the same reason, we are using the same certificate for opening the key and making it available for use. Subsequent to opening it and making it available for use, we can use the encryptkey function and store the encrypted values in the database, in the EncryptSecondCol column.  
 **/\* Update binary column with encrypted data created by certificate and key \*/  
USE EncryptTest  
GO  
OPEN SYMMETRIC KEY TestTableKey DECRYPTION  
BY CERTIFICATE EncryptTestCert  
UPDATE TestTable  
SET EncryptSecondCol =ENCRYPTBYKEY(KEY\_GUID('TestTableKey'),SecondCol)  
GO**

We can drop the original SecondCol column, which we have now encrypted in the EncryptSecondCol column. If you do not want to drop the column, you can keep it for future comparison of the data when we decrypt the column.  
 **/\* DROP original column which was encrypted for protect the data \*/  
USE EncryptTest  
GO  
ALTER TABLE TestTable  
DROP COLUMN SecondCol  
GO**

We can run a SELECT query on our database and verify if our data in the table is well protected and hackers will not be able to make use of it even if they somehow manage to reach the data.

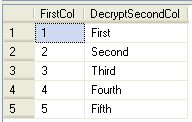
**/\* Check the content of the TestTable \*/  
USE EncryptTest  
GO  
SELECT \*  
FROM TestTable  
GO**



Result of the previous SQL query

Authorized user can use the decryptbykey function to retrieve the original data from the encrypted column. If Symmetric key is not open for decryption, it has to be decrypted using the same certificate that was used to encrypt it. An important point to bear in mind here is that the original column and the decrypted column should have the same data types. If their data types differ, incorrect values could be reproduced. In our case, we have used a VARCHAR data type for SecondCol and EncryptSecondCol.

**/\* Decrypt the data of the SecondCol  \*/  
USE EncryptTest  
GO  
OPEN SYMMETRIC KEY TestTableKey DECRYPTION  
BY CERTIFICATE EncryptTestCert  
SELECT CONVERT(VARCHAR(50),DECRYPTBYKEY(EncryptSecondCol)) ASDecryptSecondCol  
FROM TestTable  
GO**



If you drop the database after the entire processing is complete, you do not have to worry about cleaning up the database. However, in real world on production servers, the database is not dropped. It is a good practice for developers to close the key after using it. If keys and certificates are used only once or their use is over, they can be dropped as well. Dropping a database will drop everything it contains – table, keys, certificates, all the data, to name a few.

**/\* Clean up database  \*/  
USE EncryptTest  
GO  
CLOSE SYMMETRIC KEY TestTableKey  
GO  
DROP SYMMETRIC KEY TestTableKey  
GO  
DROP CERTIFICATE EncryptTestCert  
GO  
DROP MASTER KEY  
GO  
USE [master]  
GO  
DROP DATABASE [EncryptTest]  
GO**

**Summary**

Encryption is a very important security feature of SQL Server 2005. Long keys and asymmetric keys create unassailable, stronger encryption and stronger encryption uses lots of CPU to encrypt data. Stronger encryption is slower to process. When there is a huge amount of data to encrypt, it is suggested to encrypt it using a symmetric key. The same symmetric key can be encrypted further with an asymmetric key for additional protection, thereby adding the advantage of a stronger encryption. It is also recommended to compress data before encryption, as encrypted data cannot be compressed.

1. **CHÍNH SÁCH**

**Bài thực hành số 11**

**ORACLE LABEL SECURITY (4)**

* ***Tóm tắt nội dung:***
* Che dấu cột thông tin chính sách
* Sử dụng hàm gán nhãn
* Các view của OLS

# Một số kỹ thuật nâng cao trong OLS

## Lý thuyết

### Che dấu cột thông tin nhãn dữ liệu

* Để tránh việc hiển thị cột chứa thông tin chính sách, người quản trị có thể thiết lập tùy chọn HIDE khi gán chính sách cho bảng.
* Một khi chính sách đã được áp dụng, trạng thái Ẩn/Không Ẩn của cột không thể được thay đổi trừ khi ta remove chính sách khỏi bảng với tham số DROP\_COLUMN bằng TRUE. Sau đó chính sách có thể được áp dụng lại với trạng thái mới.
* Khi người dùng INSERT dữ liệu vào bảng có trạng thái ẩn cột chính sách, giá trị của cột chứa nhãn sẽ không bị yêu cầu phải insert.
* Câu lệnh SELECT \* sẽ không tự động trả về giá trị của cột ẩn, trừ khi nó được truy xuất trực tiếp.
* Câu lệnh DESCRIBE cũng sẽ không hiển thị thông tin cột ẩn.

### Hàm gán nhãn

* Có những bảng dữ liệu lớn, ta không thể ngồi gán nhãn cho từng trường hợp. OLS cung cấp cho ta một cách gán nhãn khác. Đó là sử dụng một hàm (function) do mình hiện thực để OLS sẽ tự động gán nhãn mỗi khi có hàng mới được insert vào bảng dữ liệu được bảo vệ. Xem phần thực hành để hiểu rõ hơn về cách thức làm việc này.
* Hàm gán nhãn sẽ override 2 tùy chọn LABEL\_DEFAULT và LABEL\_UPDATE.

## Thực hành

### Che dấu cột thông tin chính sách

* Do trong bài lab trước, ta đã áp dụng chính sách cho bảng mà không có tùy chọn HIDE nên trong bải lab này ta phải remove chính sách (xóa luôn cột thông tin), thực hiện lại đoạn code gán nhãn trong bài lab trước và gán lại chính sách.

CONN sec\_admin/secadmin;

BEGIN

sa\_policy\_admin.remove\_table\_policy

(policy\_name => 'ACCESS\_LOCATIONS',

schema\_name => 'HR',

table\_name => 'LOCATIONS',

drop\_column => true);

END;

/

SELECT \* FROM hr.locations;

* Ta nhận thấy lúc này cột OLS\_COLUMN vẫn chưa bị xóa dù Oracle báo thực hiện thành công thủ tục. Lý do là khi remove chính sách, ta sẽ xóa cột OLS\_COLUMN, tức là đã thay đổi bảng Locations. Do vậy ta cần cấp thêm quyền ALTER trên bảng Locations cho sec\_admin để cột OLS\_COLUMN thật sự bị xóa. Bây giờ ta gán lại chính sách trên để thấy sự thay đổi sau khi sec\_admin được gán quyền.

CONN sec\_admin/secadmin;

BEGIN

sa\_policy\_admin.apply\_table\_policy

(policy\_name => 'ACCESS\_LOCATIONS',

schema\_name => 'HR',

table\_name => 'LOCATIONS',

table\_options => 'NO\_CONTROL');

END;

/

CONN system/system;

GRANT alter ON hr.locations TO sec\_admin;

CONN sec\_admin/secadmin;

BEGIN

sa\_policy\_admin.remove\_table\_policy

(policy\_name => 'ACCESS\_LOCATIONS',

schema\_name => 'HR',

table\_name => 'LOCATIONS',

drop\_column => true);

END;

/

SELECT \* FROM hr.locations;

Ta nhận thấy bây giờ cột OLS\_COLUMN đã thật sự được xóa.

* Gán lại chính sách cho bảng với NO\_CONTROL và HIDE:

CONN sec\_admin/secadmin;

BEGIN

sa\_policy\_admin.apply\_table\_policy

(policy\_name => 'ACCESS\_LOCATIONS',

schema\_name => 'HR',

table\_name => 'LOCATIONS',

table\_options => 'HIDE,NO\_CONTROL');

END;

/

* Gán lại nhãn cho dữ liệu trong bảng (do lúc remove đã xóa mất cột chứa thông tin chính sách):

CONN sec\_admin/secadmin;

UPDATE hr.locations SET ols\_column = char\_to\_label ('ACCESS\_LOCATIONS', 'CONF');

UPDATE hr.locations SET ols\_column = char\_to\_label ('ACCESS\_LOCATIONS', 'CONF::US')

WHERE country\_id = 'US';

UPDATE hr.locations SET ols\_column = char\_to\_label ('ACCESS\_LOCATIONS', 'CONF::UK')

WHERE country\_id = 'UK';

UPDATE hr.locations SET ols\_column = char\_to\_label ('ACCESS\_LOCATIONS', 'CONF::CA')

WHERE country\_id = 'CA';

UPDATE hr.locations SET ols\_column = char\_to\_label ('ACCESS\_LOCATIONS', 'CONF:SM:UK,CA')

WHERE (country\_id = 'CA' and city = 'Toronto')

or (country\_id = 'UK' and city = 'Oxford');

UPDATE hr.locations SET ols\_column = char\_to\_label ('ACCESS\_LOCATIONS', 'CONF:HR:UK')

WHERE country\_id = 'UK' and city = 'London';

UPDATE hr.locations SET ols\_column = char\_to\_label ('ACCESS\_LOCATIONS', 'SENS:HR,SM,FIN:CORP')

WHERE country\_id = 'CH' and city = 'Geneva';

COMMIT ;

* Tiếp theo ta cần gán lại chính sách với tùy chọn HIDE và READ\_CONTROL:

CONN sec\_admin/secadmin;

BEGIN

sa\_policy\_admin.remove\_table\_policy

(policy\_name => 'ACCESS\_LOCATIONS',

schema\_name => 'HR',

table\_name => 'LOCATIONS');

sa\_policy\_admin.apply\_table\_policy

(policy\_name => 'ACCESS\_LOCATIONS',

schema\_name => 'HR',

table\_name => 'LOCATIONS',

table\_options => 'HIDE,READ\_CONTROL,WRITE\_CONTROL,CHECK\_CONTROL');

END;

/

* Bây giờ ta thử truy xuất bảng Locations:

CONN sec\_admin/secadmin;

SELECT \* FROM hr.locations;

DESCRIBE hr.locations;

* Kết quả của dòng code SELECT là “no rows selected”. Chỉ có câu lệnh DESCRIBE có trả về kết quả. Nguyên nhân là do bây giờ bảng này đã được bảo vệ, chỉ những người được cấp quyền OLS cụ thể mới có thể truy xuất. Ta log in lại bằng user SKING:

CONN sking/sking;

SELECT \* FROM hr.locations;

SELECT label\_to\_char (ols\_column) as label, locations.\*

FROM hr.locations;

* Ta thấy trong câu lệnh SELECT thứ 2, ta có chỉ định rõ cột ols\_column nên cột này mới xuất hiện. Trong kết quả truy vấn của câu SELECT thứ nhất không có cột thông tin chính sách này.

### Dùng hàm gán nhãn

* Trong phần thực hành này, ta sẽ dùng bảng Employees của schem HR để minh họa.
* Cấp các quyền cần thiết cho sec\_admin trên bảng Employees:

CONN system/system;

GRANT select, insert, update, alter

ON hr.employees TO sec\_admin;

GRANT create procedure TO sec\_admin;

CONN lbacsys/lbacsys

GRANT execute ON to\_lbac\_data\_label

TO sec\_admin WITH GRANT OPTION;

* Tiếp theo ta viết một hàm gán nhãn dựa trên điều kiện của thông tin nhân viên:

CONN sec\_admin/secadmin;

CREATE OR REPLACE FUNCTION sec\_admin.gen\_emp\_label

(Job varchar2, Depto number, Sal number)

RETURN LBACSYS.LBAC\_LABEL

AS

i\_label varchar2(80);

BEGIN

/\*\*\*\*\*\*\*\*\*\*\*\*\* Xác định level \*\*\*\*\*\*\*\*\*\*\*\*\*/

IF Sal > 17000 THEN

i\_label := 'SENS:';

ELSIF Sal > 10000 THEN

i\_label := 'CONF:';

ELSE

i\_label := 'PUB:';

END IF;

/\*\*\*\*\*\*\*\*\*\*\*\*\* Xác định compartment \*\*\*\*\*\*\*\*\*\*\*\*\*/

IF Job LIKE '%HR%' THEN

i\_label := i\_label||'HR:';

ELSIF (Job LIKE '%MK%') OR (Job LIKE '%SA%') THEN

i\_label := i\_label||'SM:';

ELSIF Job LIKE '%FI%' THEN

i\_label := i\_label||'FIN:';

ELSE

i\_label := i\_label||':';

END IF;

/\*\*\*\*\*\*\*\*\*\*\*\*\* Xác định groups \*\*\*\*\*\*\*\*\*\*\*\*\*/

i\_label := i\_label||'CORP';

RETURN TO\_LBAC\_DATA\_LABEL('ACCESS\_LOCATIONS',i\_label);

END;

/

* Ta cần gán cho LBACSYS quyền thực thi trên hàm gán nhãn vừa được tạo:

CONN sec\_admin/secadmin;

GRANT execute ON sec\_admin.gen\_emp\_label TO lbacsys;

* Ta chỉ định thủ tục vừa hiện thực làm hàm gán nhãn cho bảng Employees:

CONN sec\_admin/secadmin;

BEGIN

SA\_POLICY\_ADMIN.APPLY\_TABLE\_POLICY (

policy\_name => 'ACCESS\_LOCATIONS',

schema\_name => 'HR',

table\_name => 'EMPLOYEES',

table\_options => 'READ\_CONTROL,WRITE\_CONTROL,CHECK\_CONTROL',

label\_function => 'sec\_admin.gen\_emp\_label

(:new.job\_id,:new.department\_id,:new.salary)');

END;

/

# Các view thông tin của OLS

* Các thông tin về các chính sách của OLS được lưu trong data dictionary. Ta có thể xem các thông tin này thông qua các view của OLS.
* View DBA\_SA\_USERS: hiển thị thông tin về tất cả các chính sách có trong CSDL.
* DBA\_SA\_USER\_LEVELS: hiển thị thông tin level của mọi người dùng.
* DBA\_SA\_USER\_COMPARTMENTS: hiển thị thông tin compartment của mọi người dùng.
* DBA\_SA\_USER\_GROUPS: hiển thị thông tin group của mọi người dùng.
* Để xem tất cả các view trên cần log in vào tài khoản LBACSYS hoặc được cấp quyền SELECT từ LBACSYS.

conn lbacsys/lbacsys;

select \* from DBA\_SA\_USERS;

select \* from DBA\_SA\_USER\_LEVELS;

select \* from DBA\_SA\_USER\_COMPARTMENTS;

select \* from DBA\_SA\_USER\_GROUPS;

# Bài tập

1. Viết hàm gán nhãn GET\_CUSTOMER\_LABEL cho các khách hàng trong bảng CUSTOMERS đã tạo ở bài lab 9 theo điều kiện sau:

* Credit > 2000: level 3; 500 < credit <= 2000: level 2; còn lại level 1
* Cust\_type = ‘Platinum’ thì compartment là Manager, còn lại là Employee
* Group gán theo region.

1. Thực hiện các câu lệnh cần thiết để bảng trên được gán nhãn và được áp dụng chính sách REGION\_POLICY đã tạo trong bài lab 8.
2. Thực hiện một số câu lệnh để kiểm tra tác dụng của chính sách.