BÁO CÁO ĐỒ ÁN LÝ THUYẾT AN TOÀN BẢO MẬT DỮ LIỆU TRONG HỆ THỐNG THÔNG TIN

LỚP 19HTTT1 - NHÓM 01

GIẢNG VIÊN HƯỚNG DẪN

- *TS.* Phạm Thị Bạch Huệ
- *Ths.* Lương Vĩ Minh

Mục lục

A. Thông tin chung

I. Thông tin nhóm

Nhóm 01

Số lượng thành viên: 3

Thông tin thành viên

STT	MSSV	Họ tên	Email	Ghi chú
1	19127652	Hồ Nhật Linh	19127652@student.hcmus.edu.vn	Nhóm trưởng
2	19127512	Lâm Hoàng Phúc	19127512@student.hcmus.edu.vn	
3	19127507	Nguyễn Quang Phú	19127507@student.hcmus.edu.vn	

II. Thông tin đồ án

Hệ quản trị CSDL: Oracle database 18c.

Nền tảng ứng dụng: Windows, MacOS, Linux.

Ngôn ngữ lập trình: Java.

Framework: Java swing.

API kết nối database: JDBC (Java Database Connectivity) sử dụng **công nghệ SSL** để bảo mật thông tin trên đường truyền.

B. Kết quả đồ án

I. Phân hệ 1

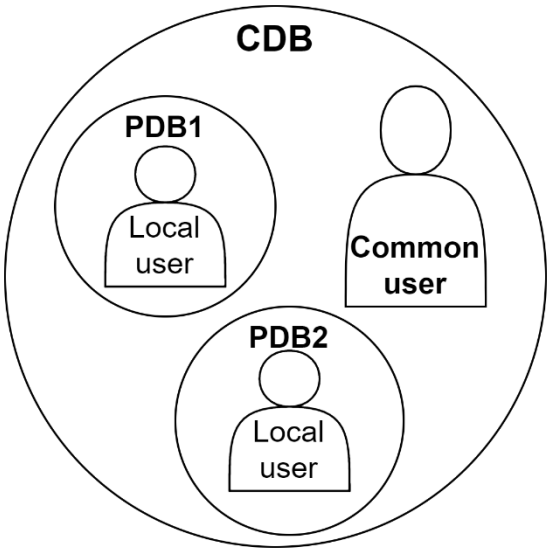
1. Lý thuyết

a. Người dùng (User)

User sở hữu 1 tài khoản trên HQTCSDL, user có thể dùng tài khoản này để đăng nhập và thực hiện các công việc (thêm, sửa, xóa dữ liệu, cấu trúc dữ liệu,…) được cho phép thực thi.

Oracle lưu trữ cơ sở dữ liệu trong 1 **CDB** (Container database hay Root) lớn để quản lý tất cả dữ liệu, trong CDB có chứa các **PDB** (Pluggable database), PDB chứa các lược đồ, dữ liệu của riêng nó.

Trong Oracle, có 2 loại user là common user và local user



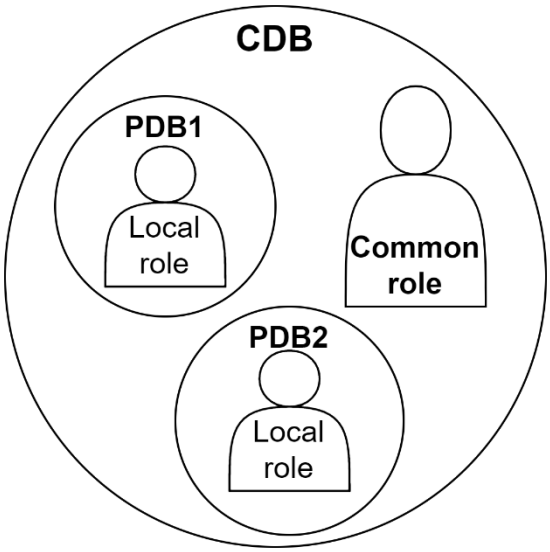
(Mô tả common user và local user trong Oracle)

- **Common user**: Có thể sử dụng tài nguyên trên toàn bộ CDB (toàn bộ CSDL), tên người dùng phải bắt đầu bằng kí tự "C##" và không được trùng nhau.
- **Local user**: Chỉ có thể sử dụng tài nguyên ở 1 PDB nó thuộc về, tên user có thể khác nhau nếu không cùng PDB.

Trong Oracle, **1 user tương ứng 1 schema**, chứa tất cả những đối tượng (table, view, procedure) của riêng schema đó, những đối tượng khác schema có thể khác tên.
Ví dụ ta có 1 user C##NHOM1 thì tuong ứng ta sẽ có 1 schema C##NHOM1, ta có thể tạo các đối tượng trên shema (sử dụng toán tử '.' để truy cập vào các đối tượng), chẳng hạn C##NHOM1.BANG_1, C##NHOM1.KHUNG_NHIN_1.

b. Vai trò (Role)

Role là 1 tập hợp những quyền liên quan với nhau. Role có thể được sử dụng để gán cho 1 user hay 1 role khác để user hay role đó kế thừa những quyền của role gán.
Tương tự như user, role cũng có 2 loại là **Common role** (có thể truy cập trên CDB, tên phải bắt đầu là "C##") và **Local role** (chỉ được truy cập PDB của mình).



(Mô tả common role và local role trong Oracle)

c. Khung nhìn (View)

Khung nhìn (View) là một bảng ảo, nó không chứa dữ liệu mà chỉ là đối tượng trung gian để tương tác với các bảng thật. View được tạo từ câu truy vấn dữ liệu
Ví dụ: Tạo view để xem thông tin giáo viên thuộc bộ môn HTTT

```
CREATE VIEW GIAO_VIEN_HTTT
AS
SELECT *
FROM GIAO_VIEN
WHERE MA_BM = 'HTTT';
```

Thực chất, View trong Oracle chỉ là những câu truy vấn đã được xác thực được lưu lại trong data dictionary.

VIEW_NAME	TEXT
ALL_XML_SCHEMAS	select u.name, s.xmldata.schema_url, case when bitanc
ALL_XML_SCHEMAS2	select u.name, s.xmldata.schema_url, case when bitanc
V_$MAP_LIBRARY	select "LIB_IDX","LIB_NAME","VENDOR_NAME","PROTOCOL_NUM","VERS
V_$MAP_FILE	select "FILE_MAP_IDX","FILE_CFGID","FILE_STATUS","FILE_NAME","'
V_$MAP_FILE_EXTENT	select "FILE_MAP_IDX","EXT_NUM","EXT_ELEM_OFF","EXT_SIZE","EXT
V_$MAP_ELEMENT	select "ELEM_NAME","ELEM_IDX","ELEM_CFGID","ELEM_TYPE","ELEM_S
V_$MAP_EXT_ELEMENT	select "ELEM_IDX","NUM_ATTRB","ATTRB1_NAME","ATTRB1_VAL","ATTR
V_$MAP_COMP_LIST	select "ELEM_IDX","NUM_COMP","COMP1_NAME","COMP1_VAL","COMP2_N
V_$MAP_SUBELEMENT	select "CHILD_IDX","PARENT_IDX","SUB_NUM","SUB_SIZE","ELEM_OFF
V_$MAP_FILE_IO_STACK	select "FILE_MAP_IDX","DEPTH","ELEM_IDX","CU_SIZE","STRIDE","N
V_$SQL_REDIRECTION	select "ADDRESS","PARENT_HANDLE","HASH_VALUE","SQL_ID","CHILD_
V_$SQL_PLAN	select "ADDRESS","HASH_VALUE","SQL_ID","PLAN_HASH_VALUE","FULI
V_$SQL_PLAN_STATISTICS	select "ADDRESS","HASH_VALUE","SQL_ID","PLAN_HASH_VALUE","FULI
V_$SQL_PLAN_STATISTICS_ALL	select "ADDRESS","HASH_VALUE","SQL_ID","PLAN_HASH_VALUE","FULI
V_$ADVISOR_CURRENT_SQLPLAN	select "TIMESTAMP","OPERATION","OPTIONS","OBJECT_NODE","OBJECT
V_$SQL_WORKAREA	select "ADDRESS","HASH_VALUE","SQL_ID","CHILD_NUMBER","WORKARE
V_$SQL_WORKAREA_ACTIVE	select "SQL_HASH_VALUE","SQL_ID","SQL_EXEC_START","SQL_EXEC_ID
V_$SQL_WORKAREA_HISTOGRAM	select "LOW_OPTIMAL_SIZE","HIGH_OPTIMAL_SIZE","OPTIMAL_EXECUTI
V_$PGA_TARGET_ADVICE	select "PGA_TARGET_FOR_ESTIMATE","PGA_TARGET_FACTOR","ADVICE_S
V_$PGA_TARGET_ADVICE_HISTOGRAM	select "PGA_TARGET_FOR_ESTIMATE","PGA_TARGET_FACTOR","ADVICE_S
V_$PGASTAT	select "NAME","VALUE","UNIT","CON_ID" from v$pgastat
V_$SYS_OPTIMIZER_ENV	select "ID","NAME","SQL_FEATURE","ISDEFAULT","VALUE","DEFAULT_
V_$SES_OPTIMIZER_ENV	select "SID","ID","NAME","SQL_FEATURE","ISDEFAULT","VALUE","CO
V_$SQL_OPTIMIZER_ENV	select "ADDRESS","HASH_VALUE","SQL_ID","CHILD_ADDRESS","CHILD_
V_$DLM_MISC	select "STATISTIC#","NAME","VALUE","CON_ID" from v$dlm_misc

(Cách Oracle lưu view)

Bên cạnh việc tăng hiệu năng truy vấn. Thì mục đích chính của việc dùng View là để thực hiện các chính sách bảo mật trong CSDL

- **Bảo vệ dữ liệu mức dòng:** ta dùng điều kiện WHERE để loại những dòng dữ liệu cần bảo vệ khỏi View. Chẳng hạn, ta tạo View để truy vấn thông tin nhân viên kế toán (KT), những nhân viên khác sẽ không được xem
  CREATE VIEW NHAN_VIEN_KE_TOAN
  AS
  *SELECT **
  FROM NHAN_VIEN
  WHERE LOAI_NHAN_VIEN = 'KE TOAN';
- **Bảo vệ dữ liệu mức cột**: ta có thể chọn ra những cột dữ liệu cần thiết cho View và ẩn những cột còn lại đi. Chẳng hạn, ta tạo View để truy vấn thông mã nhân viên và họ tên nhân viên, các trường còn lại (ngày sinh, số CMND, lương,…) sẽ bị ẩn đi khỏi View.
  CREATE VIEW NHAN_VIEN_THONG_TIN

AS
SELECT MA_NHAN_VIEN, HO_TEN
FROM NHAN_VIEN;

d. Quyền người dùng (Privilege)

Quyền người dùng (privilege) là sự cho phép người dùng thực hiện một câu lệnh SQL cụ thể hoặc thực hiện truy cập đến các objects của các người dùng khác (hay schema khác).
Có 2 loại quyền trong Oracle:

- **Quyền hệ thống (System privilege):** Cho phép người dùng thực hiện hành động trên đối tượng (table, view, index,…) của **bất kì schema nào**. Loại quyền này chỉ nên cấp cho những user có độ tin cậy và quyền hành cao.

Privilege	Description
`ADMIN`	Enables a user to perform administrative tasks including checkpointing, backups, migration, and user creation and deletion.
`ALTER ANY CACHE GROUP`	Enables a user to alter any cache group in the database.
`ALTER ANY INDEX`	Enables a user to alter any index in the database. **Note**: There is no `ALTER INDEX` statement.
`ALTER ANY MATERIALIZED VIEW`	Enables a user to alter any materialized view in the database. **Note**: There is no `ALTER MATERIALIZED VIEW` statement.
`ALTER ANY PROCEDURE`	Enables a user to alter any PL/SQL procedure, function or package in the database.
`ALTER ANY SEQUENCE`	Enables a user to alter any sequence in the database. **Note**: There is no `ALTER SEQUENCE` statement.

(Ví dụ 1 số quyền hệ thống. Nguồn: Privileges (oracle.com))

- **Quyền đối tượng (Object privilege):** Cho phép 1 user có thể thực hiện hành động hoặc truy cập vào các **đối tượng của 1 user khác** (schema khác). **User sẽ có toàn bộ quyền đối tượng trên schema của mình**

Privilege	Object type	Description
DELETE	Table	Enables a user to delete from a table.
EXECUTE	PL/SQL package, procedure or function	Enables a user to execute a PL/SQL package, procedure or function directly.
FLUSH	Cache group	Enables a user to flush a cache group.
INDEX	Table or materialized view	Enables a user to create an index on a table or materialized view.
INSERT	Table or synonym	Enables a user to insert into a table or into the table through a synonym.
LOAD	Cache group	Enables a user to load a cache group.

(Ví dụ 1 số quyền đối tượng. Nguồn: Privileges (oracle.com))

Cú pháp câu lệnh cấp quyền:

- **GRANT <PRIVILLEGE> ON <OBJECT> TO <USER>** (với những câu lệnh có mức cột thì privilege cần có thêm thông tin cột)
- Ví dụ:
  - **GRANT SELECT ON ANOTHER.TABLE_TEST TO USER_TEST**: cấp quyền SELECT trên bảng TABLE_TEST của schema ANOTHER cho user USER_TEST.
  - **GRANT UPDATE(COLUMN_1, COLUMN_2) ON ANOTHER.TABLE_TEST TO USER_TEST**: cấp quyền UPDATE trên các cột COLUMN_1, COLUMN_2 của bảng TABLE_TEST thuộc schema ANOTHER cho user USER_TEST.

Cú pháp câu lệnh thu hồi quyền đã cấp:

- **REVOKE <PRIVILLEGE> ON <OBJECT> FROM <USER>** (khi thu hồi chỉ thu hồi toàn bộ quyền, không thu hồi trên chi tiết từng cột)
- Ví dụ:
  - **REVOKE UPDATE(COLUMN_1) ON ANOTHER.TABLE_TEST FROM USER_TEST**: Đây là **câu lệnh sai** do tiến hành thu hồi trên cột.
  - **REVOKE UPDATE ON ANOTHER.TABLE_TEST FROM USER_TEST**: Đây là câu lệnh đúng, tiến hành thu hồi quyền UPDATE trên bảng TABLE_TEST thuộc shema ANOTHER của user USER_TEST.

2. Triển khai cài đặt

a. Xem thông tin các đối tượng

Xem các users trong hệ thống: ta sử dụng view **DBA_USERS** để lấy thông tin các user

```
SELECT USERNAME FROM DBA_USERS;
```

uery Result ×

SQL | Fetched 50 rows in 0.0

	USERNAME
1	SYS
2	SYSTEM
3	XS$NULL
4	OJVMSYS
5	LBACSYS
6	OUTLN
7	SYS$UMF
8	DBSNMP
9	APPQOSSYS
10	GGSYS
11	ANONYMOUS
12	DBSFWUSER
13	CTXSYS
14	DVSYS
15	SI_INFORMTN_SCHEMA

(Tất cả user trong hệ thống)

Xem các roles trong hệ thống: ta sử dụng view **DBA_ROLES** để lấy tất cả role

```
SELECT ROLE FROM DBA_ROLES;
```

uery Result ×

SQL | Fetched 50 rows

	ROLE
1	CONNECT
2	RESOURCE
3	DBA
4	PDB_DBA
5	AUDIT_ADMIN
6	AUDIT_VIEWER
7	SELECT_CATALOG_ROLE
8	EXECUTE_CATALOG_ROLE
9	CAPTURE_ADMIN
10	EXP_FULL_DATABASE
11	IMP_FULL_DATABASE
12	CDB_DBA
13	APPLICATION_TRACE_VIEWER
14	LOGSTDBY_ADMINISTRATOR
15	DBFS_ROLE
16	GSMUSER_ROLE
17	AQ_ADMINISTRATOR_ROLE

(Tất cả role trong hệ thống)

Xem các roles đã được cấp cho 1 user hoặc các users trong 1 role: ta sử dụng view **DBA_ROLE_PRIVS** với cột **GRANTEE** là user hoặc role được gán, và **GRANTED_ROLE** là role gán.

```
SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS;
```

Query Result

SQL | Fetched 50 rows in 0.009 seconds

	GRANTEE	GRANTED_ROLE
1	C##TEST_USER	DBA
2	APPQOSSYS	APPLICATION_TRACE_VIEWER
3	C##THUTHU4	C##HEHEHEHE
4	C##THUTHU5	C##HEHEHEHE
5	C##HEHEHEHE	DBA
6	C##GV0000	DBA
7	C##QUANLY0	OEM_MONITOR
8	C##GROUP1	DBA
9	C##DKHP_19127652	AUDIT_ADMIN
10	C##DKHP_19127652	DBA
11	C##DKHP_19127652	CTXAPP

(Liệt kê những role/user được cấp role)

b. Tạo/xóa các đối tượng

Để tạo các đối tượng ta cần lấy thông tin về tin về schema và tên của đối tượng đó.

Đầu tiên cần kiểm tra tên đối tượng đã tồn tại chưa

- **Đối với user, role** thì sử dụng view DBA_ROLES và DBA_USERS.
- **Đối với table** thì truy vấn view DBA_TABLES (lấy ra thuộc tính OWNER và TABLE_NAME) để xác nhận điều này.

Ta tiến hành tạo các đối tượng

- **Đối với role**, chỉ cần tên và schema của role đó: CREATE ROLE <schema>.<role>.
- **Đối với user**, ta cần thêm thông tin mặt khẩu của user: CREATE USER <user> IDENTIFIED BY <password>.
- **Đối với table**, cần cung cấp thông tin về các thuộc tính (kiểu dữ liệu, chiều dài dữ liệu, khóa chính, khóa ngoại (thuộc tính tham chiếu), cho phép null): Ta tạo 1 câu lệnh SQL để từ những thuộc tính đó theo mẫu (thế chuỗi):

  CREATE TABLE <SCHEMA_NAME>.<TABLE_NAME> (
  <COLUMN_NAME> <DATA_TYPE>(<DATA_LENGTH>, <DATA_PRECISION>) <NOT NULL> <UNIQUE>,
  --Nếu thuộc tính là khóa ngoại
  FOREIGN KEY (<COLUMN_NAME>) REFERENCES <REF_TABLE>(<REF_COLUMN>)
  …
  PRIMARY KEY (<COLUMN_NAME>,…));

c. Quản lý quyền truy cập

Ta đã điểm qua cú pháp cấp quyền và thu hồi quyền ở phần 1, ta chỉ cần lập trình để tạo những câu lệnh SQL dựa vào yêu cầu của DBA.

Để xem quyền của user, role trên các bảng:

- Đối với quyền không chi tiết trên cột **(INSERT, DELETE)**, ta dùng view **DBA_TAB_PRIVS** để lấy thông tin về quyền, với PRIVILEGE là hành động, GRANTABLE là chỉ định WITH GRANT OPTION:

```
SELECT PRIVILEGE, GRANTABLE FROM DBA_TAB_PRIVS;
```

ry Result ×

SQL | Fetched 50 rows in 0.51 seconds

PRIVILEGE	GRANTABLE
USE	YES
SELECT	YES
READ	NO
READ	NO
READ	NO
READ	NO
INSERT	NO
SELECT	NO
SELECT	NO
INSERT	NO
READ	NO
EXECUTE	NO
EXECUTE	NO

(Liệt kê quyền trên bảng của user)

- Đối với quyền chi tiết trên cột **(UPDATE),** ta dùng view **DBA_COL_PRIVS** để lấy thông tin về quyền trên các cột, với COLUMN_NAME là tên cột, PRIVILEGE là hành động, GRANTABLE là chỉ định WITH GRANT OPTION:

```
SELECT COLUMN_NAME, GRANTABLE FROM DBA_COL_PRIVS WHERE PRIVILEGE = 'UPDATE';
```

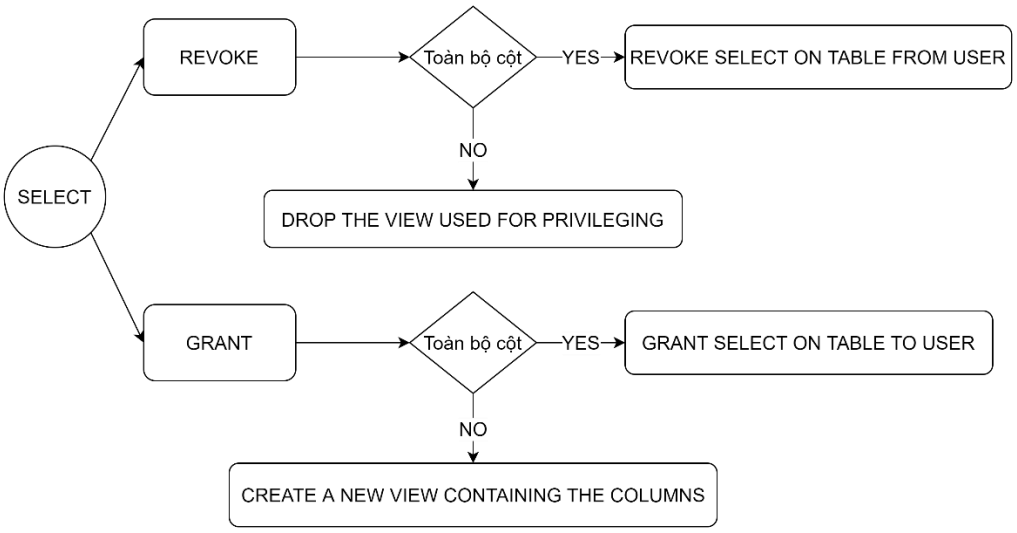
Query Result ×

SQL | All Rows Fetched: 6 in 0.284 seconds

	COLUMN_NAME	GRANTABLE
1	MA_MH	YES
2	MSSV	YES
3	A	YES
4	FIELD_3	NO
5	FIELD_2	NO
6	FIELD_1	NO

Đặc biệt chú ý, đối với quyền SELECT, do **Oracle không hỗ trợ cấp quyền SELECT chi tiết đến mức cột,** nên ta sẽ sử dụng View thay thế. Nếu DBA cấp quyền cho toàn bộ cột thì ta cấp bằng câu lệnh GRANT SELECT như bình thường, còn nếu cấp chi tiết đến cột thì ta sẽ tạo View mới có tên là <schema>.<user>_SELECT_ON_<table> với những cột đã chọn. Khi thu hồi quyền ta sẽ tiến hành thu hồi quyền SELECT trên bảng đi (nếu user có quyền

trên toàn bộ cột) hoặc xóa hẳn View dùng cho việc cấp quyền đi (nếu user có quyền trên 1 số cột).



(Mô tả việc cấp/thu hồi quyền SELECT)

II. Phân hệ 2

1. Tóm tắt lý thuyết

a. DAC

Mô hình DAC (Discretionary Access Control - Điều khiển truy cập tùy quyền) thực hiện phân quyền dựa trên mỗi người dùng, xem xét người dùng U có quyền P gì trên đối tượng O của cơ sở dữ liệu. DAC cho phép người dùng có thể cấp quyền truy cập vào những đối tượng mà họ có quyền cho người dùng khác.

Ví dụ, ta cấp quyền cho user GROUP_1 để đọc trên bảng NHAN_VIEN, và cho phép GROUP_1 có thể cấp quyền đọc cho bất kì user nào khác trên bảng NHAN_VIEN

GRANT SELECT ON NHAN_VIEN TO GROUP_1 WITH GRANT OPTION

Sau đó, GROUP_1 cấp quyền đọc cho GROUP_2, tuy nhiên không muốn cho GROUP_2 cấp quyền này cho ai

GRANT SELECT ON NHAN_VIEN TO GROUP_2;

Khi quyền đọc trên NHAN_VIEN của GROUP_1 bị thu hồi, thì GROUP_2 cũng bị mất quyền này đi trong trường hợp là Oracle database, tuy nhiên sự thu hồi này có thể khác đi ở những RDBMS khác (có thể là GROUP_2 vẫn còn giữ được quyền của mình).

REVOKE SELECT ON NHAN_VIEN FROM GROUP_1

- Có thể dẫn đến: *REVOKE SELECT ON NHAN_VIEN FROM GROUP_2*

Mô hình DAC tập trung vào cá nhân hóa của mỗi người dùng, thường là cho phép họ truy cập vào những dòng dữ liệu có liên quan đến chính bản thân mình. DAC thường sử dụng View (đã đề cập ở phần I) như cơ chế bảo mật để ép thỏa.

Ví dụ, ta có 1 View

CREATE VIEW NHAN_VIEN_KE_TOAN
AS
*SELECT **

FROM NHAN_VIEN

WHERE MA_NHAN_VIEN = USER;

View này được sử dụng để mỗi nhân viên chỉ có thể truy cập vào dòng dữ liệu của chính mình.

Nhận xét về DAC, thì đây là mô hình cấp quyền rất linh hoạt, mỗi người dùng có thể có những quyền đa dạng tùy vào vai trò của họ trong hệ thống. Tuy nhiên, mô hình này sẽ trở nên cồng kềnh và khó quản lý khi mà số lượng người dùng trong hệ thống quá lớn, việc cấp những quyền thích hợp cho mỗi người dùng là rất tốn chi phí và thời gian, dẫn đến sự ra đời của mô hình có tính thực tế hơn là RBAC.

b. RBAC

Để nói về RBAC, đầu tiên ta cần phải hiểu về role (vai trò). Như đã đề cập ở phần I, role là một tập những quyền trong hệ thống, những người dùng ở trong role bất kì sẽ được thực hiện tất cả những quyền của role đó. Chẳng hạn, ta có role NHAN_VIEN_ROLE

CREATE ROLE NHAN_VIEN

Ta cấp quyền SELECT, UPDATE cho role NHAN_VIEN_ROLE trên bảng NHAN_VIEN

GRANT SELECT, UPDATE ON NHAN_VIEN TO NHAN_VIEN_ROLE

Lúc này, những người dùng nằm trong role NHAN_VIEN sẽ có quyền SELECT, UPDATE trên bảng NHAN_VIEN. Ví dụ, ta thêm người dùng GROUP_1 vào role NHAN_VIEN_ROLE

GRANT NHAN_VIEN_ROLE TO GROUP_1

RBAC cho thấy tính hiệu quả của nó trong những hệ thống lớn, khi mà số lượng người dùng lên đến 5, 6 con số (thậm chí là lớn hơn), thì tất cả những gì ta cần làm là đưa họ vào những role thích hợp.

c. VPD

Ở phần trên, khi đề cập đến DAC ta có điểm qua cách sử dụng View để bảo mật dữ liệu cấp dòng (chỉ cấp quyền truy cập trên những dòng dữ liệu thích hợp với mỗi người dụng cụ thể, hay còn gọi là Row-Level Security).

Tuy nhiên, ta có thể nhìn ra điểm yếu của việc ép thỏa bằng View, đó chính là số lượng View sẽ "bùng nổ" khi mà ta có quá nhiều vai trò khác nhau trong hệ thống, chẳng hạn như trong đồ án này, ta có các người dùng "Thanh tra", "Nghiên cứu", "Cơ sở y tế", "Bác sĩ" khi xem thông tin trên bảng HSBA (Hồ sơ bệnh án) sẽ ra những dòng khác nhau, điều đó đồng nghĩa với việc ta phải 4 View khác nhau và cấp quyền cho từng role trên những View thích hợp. Việc này sẽ trở nên khó khăn hơn trong việc quản lý và gây tốn chi phí, thử tưởng tượng ta có thêm 10 hay 20 vai trò trong hệ thống thì lại tạo thêm 10 hay 20 View, và khi yêu cầu nghiệp vụ thay đổi thì ta phải đi chỉnh sửa từng View cho phù hợp. Khái niệm VPD ra đời nhằm giải quyết tình huống này.

VPD là cơ chế bảo mật (thường dùng cho các chính sách bảo mật dữ liệu cấp dòng), nó hoạt động bằng cách tự động thêm các vị từ trong truy vấn của người dùng.

Vị từ đơn giản là những câu điều kiện nằm sau mệnh đề WHERE trong câu truy vấn, chẳng hạn ta có câu truy vấn

*SELECT * FROM NHAN_VIEN WHERE MANV != 10*

Thì vị tự ở đây là 'MANV != 10'

Vậy, rất đơn giản, ta sẽ tạo ra các vị từ cho VPD để nó tự động thêm vào trong các câu truy vấn (thêm điều kiện WHERE hoặc AND với những điều kiện đã có trong câu)

Để tạo vị từ, ta sẽ sử dụng các hàm (Function) để trả về vị từ, chẳng hạn ta có 1 hàm để trả về toàn bộ nhân viên nếu người đọc là DBA, hoặc trả về các nhân viên không phải là 'Giám đốc' nếu người đọc là người khác.

```
CREATE OR REPLACE FUNCTION LAY_VI_TU (
    schema_name     IN VARCHAR2 DEFAULT NULL,
    object_name     IN VARCHAR2 DEFAULT NULL)
AS
BEGIN
        IF USER = 'SYS'
        THEN
                RETURN '1 = 1'; --Trả ra tất cả các dòng
        END IF;
        RETURN 'VAITRO != ''GIAMDOC''';
END;
```

Ta tiến hành áp dụng VPD trên bảng NHAN_VIEN

```
DBMS_RLS.ADD_POLICY(
    object_name     => 'NHAN_VIEN',
    policy_name     => 'SELECT_NV',
    policy_function => 'LAY_VI_TU');
```

Khi đó, ví dụ ta truy vấn

```
SELECT * FROM NHAN_VIEN
```

Oracle sẽ tự động thêm điều kiện cho câu truy vấn

Nếu người dùng hiện tại là DBA thì câu truy vấn thực sự sẽ trở thành

```
SELECT * FROM NHAN_VIEN WHERE 1 = 1 --Trả về mọi dòng
```

Nếu người dùng hiện tại là người nào khác ngoài DBA, thì câu truy vấn thực sự sẽ trở thành

```
SELECT * FROM NHAN_VIEN WHERE VAITRO != 'GIAMDOC'
```

d. OLS

Ngoại trừ View và VPD, ta vẫn còn 1 cơ chế bảo mật dữ liệu cấp độ dòng rất hiệu quả là **OLS (Oracle label security).**

OLS hoạt động bằng cách sử dụng các khái niệm level, compartment, group của người dùng, tiến hành đánh nhãn dữ liệu (thành phần của nhãn bao gồm level, compartment và group đó), khi người dùng truy cập dữ liệu, Oracle sẽ đánh giá so sánh nhãn của dữ liệu và nhãn của người dùng dựa vào thuật toán để xác định có cho phép truy cập hay không.

Level (cấp độ): dùng để đánh giá cấp bậc của người dùng (biểu thị mức độ quan trọng của họ trong hệ thống), ví dụ ta có các level 'Nhân viên tập sự', 'Nhân viên chính thức', 'Quản lý', level là tiêu chí quan trọng đầu tiên quan trọng nhất để xem xét khả năng truy cập của người dùng.

Compartment: đề cập tiêu chí lĩnh vực của người dùng trong hệ thống, chẳng hạn ta có các lĩnh vực trong đồ án là 'Khoa nội', 'Khoa ngoại' và 'Chuyên sâu'. Compartment là tiêu chí không bắt buộc khi gán nhãn dữ liệu.

Group: đề cập tiêu chí khu vực hoạt động của người dùng trong hệ thống, chẳng hạn ta có các khu vực trong đồ án là 'Cận trung tâm', 'Ngoại thành' và 'Cận thành'. Group là tiêu chí không bắt buộc khi gán nhãn dữ liệu.

Nhãn được ghi như sau:

'Level:Compartment_1,Compartment_2,…,Compartment_n:Group_1,Group_2,…,Group_n'

Thuật toán để Oracle xem xét cho phép truy cập dữ liệu (đọc dữ liệu)

- Level của người dùng phải lớn hơn hoặc bằng level của dữ liệu
- Compartment của người dùng phải bao gồm toàn bộ compartment của dữ liệu
- Group của người dùng phải bao gồm ít nhất 1 group của dữ liệu

Chẳng hạn, ta có các level YBS < GDS; compartment KN, CS; Group CTT, NT

Nếu một dữ liệu có nhãn 'YBS:KN:CTT' thì yêu cầu người dùng phải có ít nhất level YBS (YBS hay GDS), compartment phải có KN, group phải có ít nhất CTT

Nếu một dữ liệu có nhãn 'GDS:KN,CS:CTT,NT' thì yêu cầu người dùng phải có ít nhất level GDS (hiện tại chỉ có GDS), compartment phải có KN, CS; group phải có CTT hoặc NT (1 trong 2).

Ta có thể thấy, so với VPD, OLS dễ thực hiện hơn rất nhiều, cách hoạt động cũng rất đơn giản và dễ hiểu, không cần phải code dài dòng và tính toán nhiều trường hợp như VPD. Tuy nhiên cũng có hạn chế, ví dụ ta phải xác định rõ ràng và chặt chẽ các level, compartment và group của từng người dùng trong hệ thống, đôi khi có những trường hợp đặc biệt ta muốn người dùng level thấp đọc những dữ liệu có level cao thì cũng khó khăn để thực hiện.

e. Mã hóa

Mã hóa là quá trình ẩn dữ liệu đi, biến nó thành dạng không thể đọc được (thường là trở thành chuỗi byte)

Mã hóa là lá chắn cuối cùng trong quá trình bảo mật cơ sở dữ liệu, dù cho kẻ xấu có đi qua những bước cấp quyền bên trên thì cũng không thể khai thác dữ liệu của hệ thống

Để thực hiện mã hóa, ta cần có khóa, tùy vào cơ chế mã hóa mã có thể xuất hiện thêm nhiều khóa. Có 2 cách mã hóa cơ bản nhất; Đầu tiên là mã hóa sử dụng khóa đối xứng, nghĩa là ta sử dụng cùng 1 khóa cho quá trình mã hóa và giải mã dữ liệu, cách làm này giúp đẩy mạnh hiệu năng của quá trình mã hóa nhờ sử dụng thuật toán đơn giản và khóa ngắn, ta cũng phải đối diện với sự khó khăn của việc phân phối khóa; Cách thứ 2 là mã hóa bằng khóa bất đối xứng, nghĩa là ta sử dụng 1 khóa (gọi là public key) cho quá trình mã hóa và sử dụng 1 khóa khác cho quá trình giải mã (gọi là private key), cách làm này giúp giải quyết vấn đề phân phối tuy nhiên lại gây ảnh hưởng về mặt hiệu năng do thuật toán phức tạp.

Vấn đề rất quan trọng trong mã hóa, đó là quản lý khóa (lưu trữ, phân phối, thay đổi khóa, làm mới khóa), điều này phụ thuộc phần lớn vào nơi chúng ta mã hóa (mức ứng dụng, mức hệ điều hành hay mức CSDL), 1 trong những cách mã hóa phổ bên nhất là mã hóa mức CSDL, việc mã hóa ở mức CSDL đem lại cho ta nhiều lợi ích: bảo mật dữ liệu khỏi những kẻ xâm nhập, kể cả DBA, có thể chia sẻ dữ liệu giữa nhiều ứng dụng với nhau; Đương nhiên, phương pháp này vẫn có những nhược điểm nhất định, như là làm chậm hoạt động của DB Server, không thể bảo vệ được tấn công ở mức ứng dụng.

f. Audit

Audit không phải là một cơ chế dùng để thực hiện cấp quyền truy cập cho người dùng.

Audit là cơ chế bảo mật hoạt động bằng cách ghi lại các hoạt động đã xảy ra trong cơ sở dữ liệu (tương tự như một nhật kí).

Mục đích chính của audit là để phát hiện ra những hoạt động bất thường trong hệ thống (phát hiện ra những lỗ hỏng bảo mật) và ai là người gây ra sự bất thường này (tránh sự thoái thác trách nhiệm), audit còn dùng để truy vết những lần cập nhật dữ liệu.

Trong Oracle ta có 2 loại audit chính là **Standard audit** và **Fine-grained audit.**
Standard là audit chuẩn của Oracle, sẽ ghi nhận lại toàn bộ hoạt động mà ta đã chỉ định, ta có thể thực hiện audit trên table, view, procedure,…
Chẳng hạn ta thực hiện audit cho hành động SELECT trên bảng NHAN_VIEN khi không thành công (BY ACCESS là chỉ định ghi trên mỗi lần truy cập, nếu dùng BY SESSION thì chỉ ghi trên mỗi session dù có truy cập bao nhiêu lần)
AUDIT SELECT ON NHAN_VIEN BY ACCESS WHENEVER NOT SUCCESSFUL
Fine-grained audit hỗ trợ nhiều hơn so với Standard audit, ta sẽ nghĩ đến những trường hợp khi mà chỉ cần audit ở 1 điều kiện cụ thể, hay chỉ audit khi truy cập lên những cột nào đó.
Chẳng hạn ta thực hiện audit khi người đọc bảng NHAN_VIEN không phải là chính họ

```
dbms_fga.add_policy(object_schema   => 'C##QLKCB',
        object_name     => 'NHANVIEN',
        policy_name     => 'audit_nhanvien_another_pol',
        audit_condition => 'USERNAME != USER',
        statement_types => 'SELECT');
```

Thực hiện audit khi có cập nhật trên trường lương của nhân viên

```
dbms_fga.add_policy(object_schema   => 'C##QLKCB',
        object_name     => 'NHANVIEN',
        policy_name     => 'audit_nhanvien_salary_pol',
        audit_column    => 'LUONG'
        statement_types => 'UPDATE');
```

2. Các chính sách bảo mật trong đồ án

2.1 Kết nối dòng dữ liệu với tài khoản người dùng

Ta cần kết nối dòng dữ liệu với tài khoản người dùng, ta có 2 bảng NHANVIEN và BENHNHAN cần thực hiện điều này.
Ta có thể thực hiện điều này bằng cách cho trường mã (nhân viên/bệnh nhân) chính là username của người dùng. Tuy nhiên, điều này có thể làm lộ username của người dùng khi mã của họ bị lộ, nên ta sẽ dùng cách thứ 2. Ta sẽ chèn thêm 1 trường USERNAME cho 2 bảng NHANVIEN và BENHNHAN (tất nhiên là thuộc tính UNIQUE)

BENHNHAN
MABN
.....
USERNAME

NHANVIEN
MANV
.....
USERNAME

2.2 Ứng dụng DAC + RBAC

Xem thông tin bệnh nhân: ta có yêu cầu nghiệp vụ

- Thanh tra có thể xem thông tin toàn bộ bệnh nhân trong hệ thống
- Bệnh nhân có thể xem thông tin của chính mình
- Bác sĩ có thể xem thông tin bệnh nhân mà mình đã chữa trị

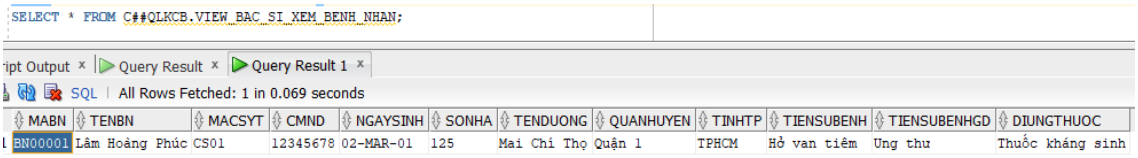
Xem thông tin nhân viên:

- Nhân viên có thể xem thông tin của chính mình
- Thanh tra có thể xem thông tin của toàn bộ nhân viên trong hệ thống

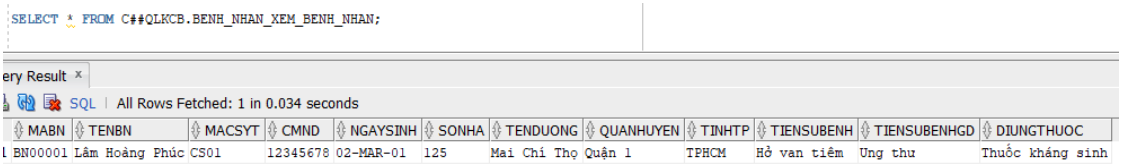
Ta có thể thấy trước mặt, ta cần tạo ra 2 View để xem thông tin bệnh nhân và 1 View để xem thông tin nhân viên, xét thấy số lượng này là không lớn, hơn nữa cũng không yêu cầu thuật toán quá khó khăn nên ta vẫn sẽ áp dụng cơ chế View để ép thỏa các chính sách này thay vì dùng VPD.

Ngoài Thanh tra viên có thể đọc thẳng trên bảng gốc (xem toàn bộ), thì với nhân viên thường, bác sĩ và bệnh nhân cần phải tạo 3 View

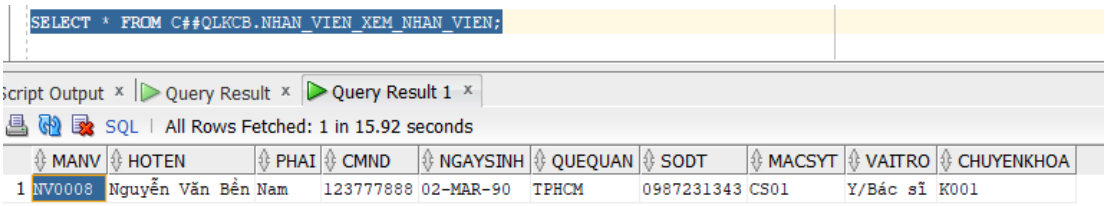
- View cho bác sĩ xem thông tin bệnh nhân: chỉ lấy những bệnh nhân có hồ sơ bệnh án có mã bác sĩ trùng với bác sĩ đang truy cập. Ví dụ khi bác sĩ có mã 'NV0008' xem thông tin bệnh nhân



- View cho bệnh nhân xem thông tin cá nhân: chỉ lấy bệnh nhân có USERNAME trùng với USER hiện tại đang truy cập. Ví dụ khi bệnh nhân 'BN00001' xem thông tin cá nhân



- View cho nhân viên xem thông tin cá nhân: chỉ lấy nhân viên có USERNAME trùng với USER hiện tại đang truy cập. Ví dụ khi nhân viên 'NV0008' xem thông tin cá nhân



Số lượng người dùng trong hệ thống là rất lớn nên ta cần tạo các role cho họ. Ta có các role như sau

- BENHNHAN: cho toàn bộ người dùng bệnh nhân
- NHANVIEN: cho toàn bộ người dùng nhân viên
- THANHTRA: cho các nhân viên thanh tra
- BACSI: cho các nhân viên bác sĩ
- CSYT: cho các nhân viên cơ sở y tế
- NGHIENCUU: cho các nhân viên nghiên cứu

2.3 Ứng dụng VPD

Xem hồ sơ bệnh án

- Nhân viên Thanh tra và DBA có thể xem thông tin toàn bộ hồ sơ bệnh án
- Nhân viên Cơ sở y tế có thể đọc, thêm, xóa hồ sơ bệnh án thuộc cơ sở y tế của họ trong vòng từ ngày 5 đến ngày 27 tháng này

- Nhân viên Y/Bác sĩ có thể đọc hồ sơ bệnh án mà họ chữa trị (có mã bác sĩ trùng với mã nhân viên)
- Nhân viên Nghiên cứu có thể đọc hồ sơ bệnh án có thuộc cơ sở y tế và chuyên khoa của họ

Xem hồ sơ bệnh án dịch vụ

- Người dùng chỉ có thể đọc, thêm, xóa các dòng dịch vụ trong hồ sơ bệnh án nếu có quyền tương tự trên hồ sơ bệnh án của dòng dữ liệu này (nghĩa là có chính sách tương tự như hồ sơ bệnh án)

Phân tích vấn đề này, ta thấy rằng có thể tiếp cận được bằng cơ chế View, tuy nhiên ta sẽ phải tạo ít nhất 3 View cho bảng HSBA và 3 View cho bảng HSBA_DV nếu muốn ép thỏa các chính sách một cách chặt chẽ. Xét thấy số lượng view này là khá lớn, hơn nữa các view bảng HSBA_DV sẽ phải sử dụng các phép truy vấn để JOIN với bảng HSBA khá phức tạp, nên ta sẽ sử dụng 1 giải pháp hợp lí hơn là VPD.

Ta cần 2 hàm trả ra vị từ cho 2 bảng HSBA và HSBA_DV

- **Hàm trả ra vị từ cho bảng HSBA**
  - *Nếu người dùng là DBA hoặc Thanh tra viên thì trả ra '1 = 1' (toàn bộ dòng)*
  - *Nếu người dùng là Cơ sở y tế thì trả ra 'MACSYT = MACSYT_CUA_NGUOI_DUNG AND NGAY thuộc vào tháng này (ngày 5 – 27)'*
  - *Nếu người dùng là Y/Bác sĩ thì trả ra 'MABS = MANV'*
  - *Nếu người dùng là Cơ sở y tế thì trả ra 'MACSYT = MACSYT_CUA_NGUOI_DUNG AND MAKHOA = MAKHOA_CUA_NGUOI_DUNG'*
- **Hàm trả ra vị từ cho bảng HSBA_DV**: ta sẽ dùng điều kiện EXISTS để tìm dòng HSBA kết nối với nó và dùng điều kiện tương tự như hàm vị từ của HSBA đối với HSBA vừa tìm được

Khi Thanh tra đọc bảng HSBA

```
SELECT * FROM C##QLKCB.HSBA;
```

Query Result

SQL | All Rows Fetched: 5 in 16.39 seconds

	MAHSBA	MABN	NGAY	CHANDOAN	MABS	MAKHOA	MACSYT	KETLUAN
1	HS000001	BN00001	12-MAY-22	HIV/AIDS	NV0008	K001	CS01	Âm tính
2	HS000002	BN00002	12-MAY-22	Bệnh viêm mũi	NV0009	K001	CS01	Bệnh nhân bị viêm mũi nặng
3	HS000003	BN00003	12-MAY-22	Đau nửa đầu	NV0010	K004	CS02	Ung thư não
4	HS000004	BN00004	12-MAY-22	Bệnh gout	NV0011	K003	CS02	Bị gout nhẹ
5	HS000005	BN00001	12-MAY-22	Tiểu đường	NV0012	K002	CS02	Tiểu đường trong thời kì mang thai

Khi Bác sĩ 'NV0008' đọc bảng HSBA

```
SELECT * FROM C##QLKCB.HSBA;
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 1 in 0.271 seconds

	MAHSBA	MABN	NGAY	CHANDOAN	MABS	MAKHOA	MACSYT	KETLUAN
1	HS000001	BN00001	12-MAY-22	HIV/AIDS	NV0008	K001	CS01	Âm tính

2.4 Ứng dụng OLS

Ta xét yêu cầu nghiệp vụ xem thông báo của người dùng. Có các cấp bậc người dùng: Y bác sĩ, Giám đốc cơ sở y tế, Giám đốc sở; Có các lĩnh vực Ngoại trú, Nội trú, Chuyên khoa; Có các khu vực Ngoại thành, Cận trung tâm, Trung tâm. Khi mà các yếu tố này đã được xác định rõ ràng, thì OLS là một lựa chọn tối ưu cho chính sách này, mặc dù VPD vẫn có thể được ứng dụng, tuy nhiên OLS sẽ dễ triển khai hơn rất nhiều, hơn nữa nếu thêm các vai trò này vào trong dữ liệu ban đầu có thể khiến ta phải chỉnh sửa các hàm vị từ VPD đã tạo.

Đầu tiên, ta có các level trong hệ thống

- YBS: Y bác sĩ, 7000
- GDCSYT: Giám đốc cơ sở y tế, 8000
- GDS: Giám đốc sở, 9000

Ta có các compartments:

- Ngoai: Ngoại trú
- Noi: Nội trú
- Sau: Điều trị chuyên sâu

Ta có các group:

- NT: Ngoại thành
- CTT: Cận trung tâm
- TT: Trung tâm

Ta tạo 1 số nhãn cho dữ liệu trong bảng THONGBAO

- (1) YBS: Toàn bộ user level ít nhất là YBS
- (2) YBS:noi,ngoai:ctt,tt: Toàn bộ user level ít nhất là YBS; bao gồm compartments Noi,Ngoai; ít nhất phải có group ctt hoặc tt
- (3) YBS:noi,sau:tt: Toàn bộ user level ít nhất là YBS; bao gồm compartments Noi,Sau; ít nhất phải có group tt
- (4) GDCSYT: Toàn bộ user có level ít nhất là GDCSYT
- (5) GDCSYT:noi,ngoai:tt: Toàn bộ user level ít nhất là GDSYT; bao gồm compartments Noi,Ngoai; ít nhất phải có group tt
- (6) GDS: Toàn bộ user có level ít nhất là GDS

Ta thiết lập 1 số loại người dùng trong hệ thống

- YBS: có thể đọc (1)
- YBS:noi,ngoai:ctt: có thể đọc (1), (2)
- YBS:noi,ngoai,sau:tt: có thể đọc (1), (2), (3)
- GDCSYT:noi:tt: có thể đọc (1), (4)

- GDCSYT:noi,ngoai:nt: có thể đọc (1), (4)
- GDCSYT:noi,ngoai,sau:ctt,tt: có thể đọc (1), (2), (3), (4)
- GDS::nt: có thể đọc (1), (4), (6)
- GDS: ngoai,noi,sau:tt: có thể đọc toàn bộ dữ liệu

2.5 Ứng dụng mã hóa

Trong đồ án này, ta sẽ tiến hành **mã hóa dữ liệu CMND của bệnh nhân và nhân viên**.
Do ứng dụng của chúng ta không có 1 web server cố định, mà sử dụng Database như một server chính quản lý các hoạt động, nên ta sẽ tiến hành mã hóa ở mức CSDL.
Đầu tiên ta cần đề cập đến cách lưu trữ khóa. Do mã hóa mức CSDL, nên tốt nhất ta sẽ lưu trong cơ sở dữ liệu (bảng chứa khóa nằm trong schema khác) để tăng hiệu năng và dễ dàng trong việc giải mã. Trong hệ thống, mỗi nhân viên hay bệnh nhân có 1 khóa riêng, được tạo bằng hàm tạo byte ngẫu nhiên của Oracle, mỗi dòng trong bảng chứa khóa tương ứng với 1 khóa, gồm 2 thuộc tính là mã nhân viên/bệnh nhân và giá trị của khóa.
Bảng chứa khóa cũng là một yếu tố đáng quan tâm, nếu dữ liệu của bảng này bị truy cập bất hợp lệ thì coi như việc mã hóa thành "công cốc". Do đó, ta sẽ chồng thêm 1 lớp bảo mật nữa, bằng cách mã hóa bảng chứ khóa, mỗi dòng khóa (thuộc về 1 bệnh nhân hay nhân viên) được mã hóa bằng key là 'mããm' (ví dụ BN001 có khóa của dòng khóa là 'BN0001100NB'), key này là do chúng ta tự suy diễn ra (sẽ viết thành thuật toán trong các function, procedure giải mã). Chú ý rằng, chúng ta không chỉ mã hóa trường khóa, mà còn mã hóa cả trường mã nhân viên/bệnh nhân trong bảng khóa, ta làm điều này là do nếu trường mã không bị ẩn đi, kẻ tấn công có thể dùng cột mã để suy diễn ra thuật toán mã hóa trong bảng chứa khóa.

	D3B75866A3178	B261E86AE5213
1	5478037C7A74E1146C4FE57DE3C4E5ED	359C7A6BA5B67194331CE2C1132994B0C0A3543ADD31975FFF3C34EE47A945DEEA8B798597C9939FC3DA35E4ADC63549
2	660F3E0D8234FD474FF9104A2AD7C838	4B2F428B3D8A85BF723A2BB113A09FF78169AEFF3A50619A5FE7728347D100D7BF17BD8A1C988B11CB3340EDC10476AC
3	3F2641DB55CFE27630BFD6D1CBA9B526	59B908805F2546112E2A70B95B5B182E37A39C23A84D96CF1B0B5A1830D224DA81F59148AACEDCA01BAB6F94C9A712F7
4	0D4BDBACB5FC2A59F780A3276B6E264A	E76BA109CBE9F2866F69DEB801A5667BB1C79AE915A6C24F667C996DB654B30DC32C14451666668B928BAED5A681534C
5	1A7916055B867B7A24490D9B12ED35B4	958AB88EAC782E96CEBC6DA2691AA66CEFEC67492536B88853D8A7D9AB98A2E3E22CF4E6353322FB9759086F230E415A
6	42998F9B2EF676844AD832FB91E7A1ED	89B4061DDDA1EDC8551A0E695758D6CFA648079F228079786550F199AA1655AE864C80C86845D3F70E0CA911DB49E3A0
7	460D2D3AD5C415D12BBD5F3ACD2D6F77	D2575D669662E8A0BF172B56CDCC517362C8D81504004A3A05ED0A612765617BC0BA68AED849A293F297F1A3982FA07E
8	DD89EA5E7760850E6226420ED550A42E	420GA3C5F80C373F1C4FEBD9B5E02DE76908AC8FC0C704D0E1ED59C1371929EA82B86D3435BBA16F4C95E4D2CBA269DA
9	525635AF669C1465123D0443491F3E3F	8A50BEA2ED7F009ADEA4BF9A77FF588558094B48A5F4A36B5BC7BF863B3DBBFCD2B275385E558DFA9F403537B571CCDF
10	276CAF1C782A49CB4980CB1E109C6952	5733FADE6A68F7F13D978E95AEE3676B25985A68DE5B9AF0B409F3BCCA348641241C17B23C95E423C17F5629F9CAF666
11	52A5B3565E058F4888DD05572F4AC901	5E6BB34EB4023CADEFC267603351552 9AA61AD68D92B5FD6B2A017BCB31A2315117C7A7D71C96D3AEBD8FFF2319340F1
12	4808DF21FE1F62A28F8E03E324689998	021C4B058F00EDB4DD350B21FB87E13697B6F83964D999D43AB44A65346195ECB1596982B24C3F10DCC7991CDCCF9E91
13	41C17C42587026 93F30CC79900408587	5037AFADB427936 1008DE6D3F6A08A388657026752BCD98FA5D39D2F811E240BB8B24BCA24362FF69997DC743AA4BFC2
14	D0C7A7A2CBEFAF9A9E043DB10EE382AE	4B3D73D3C27353D97A964013FA308B0DCA4001356FAB546D264604E2F1E6A39E2A8C65DC1E5EE0C3DB9444B909EE3FE1
15	37858C3D5897D641AFDFA4A94D1832C8	D6ABF9AE807864C0A8480FD589A8074B4BE92111EED3089C2543DB5C3A9AD356A53498E6D0D69B84FA51539E116B5A80
16	1454303D7307056D11EEA02A1706135D	D33C2909CBB93AD3B2B3390EAF9437E85594A34E9881A882DE72D96C9904FEE8D73BBDBB3B1EE83B65B80986F8F8F0B5

(Hình ảnh bảng chứa khóa)

Mỗi khi cần đọc, thêm, cập nhật CMND bảng nhân viên, bệnh nhân cần giải mã khóa trong bảng khóa rồi mới dùng khóa đó giải mã tìm ra CMND, có thể thấy cách mã hóa của chúng ta dù an toàn nhưng đã làm tốn chi phí của hệ thống khá nhiều.
Do khóa được lưu trong cơ sở dữ liệu, ta không cần bận quá tâm quá nhiều đến việc phân phối khóa, mất khóa, tuy nhiên phải bảo mật thật cẩn thận bảng chứa khóa, vì mất khóa thì coi như dữ liệu cũng "đi" theo.
Để thay đổi khóa cần cần giải mã ra CMND, xóa dòng trong bảng khóa, tạo khóa mới ngẫu nhiên, mã hóa lại CMND và insert khóa mới vào lại bảng chứa khóa.

2.6 Ứng dụng Audit

a. Standard audit

Bênh nhân và nhân viên là những người dùng trực tiếp trong hệ thông. Việc thêm, xóa nhân viên hay bênh nhân cần phải được ghi nhận lại

➔ Audit INSERT, DELETE trên bảng BENHNHAN và NHANVIEN

Các hoạt động khám chữa bệnh hàng ngày cần phải được ghi nhận lại (HSBA, HSBA_DV), tránh những cập nhật không hợp lệ làm ảnh hưởng hoạt động của hệ thống, nếu có xảy ra thì cũng phải xem xét trách nhiệm của người đó

➔ Audit INSERT, DELETE, UPDATE trên bảng HSBA và HSBA_DV

Bảng chứa khóa rất quan trọng, việc mất khóa hay sửa khóa có thể làm mất hẳn dữ liệu đã mã hóa, cần ghi nhận lại các hành động thêm, sửa, xóa trên bảng này

➔ Audit INSERT, DELETE, UPDATE trên bảng khóa

Hành động đọc không thành công trên bảng chứa khóa có thể là dấu hiệu cho kẻ xấu cố tình truy cập, cần ghi nhân lại.

➔ Audit hành động SELECT không thành công trên bảng khóa

Chúng ta có sử dụng khóa tự suy diễn (thuật toán được ghi lại trong các procedure và function), do đó cần ghi nhận lại khi có người cố tình truy cập xem mã nguồn của các procedure và function

➔ Audit SELECT trên view mà hệ thống cho xem thông tin các procedure (ALL_SOURCE)

b. Fine-grained audit

Cần ghi nhận lại khi có người dùng bảng nhân viên hay bệnh nhân mà không phải là chính họ

➔ Fine-grained audit trên bảng nhân viên và bệnh nhân khi người truy xuất không phải chính họ (USER != USERNAME trong bảng)

Cần ghi nhận lại khi thông tin VAITRO, USERNAME, CMND của nhân viên được cập nhật, đây là những thông tin quan trọng, có thể làm ảnh hưởng đến quá trình hoạt động của hệ thống, chẳng hạn chỉnh sửa VAITRO có thể khiến cho nhân viên có thể xem mọi thông tin họ muốn nếu chỉnh thành Thanh tra viên.

➔ Fine-grained audit trên bảng nhân viên cho hành động UPDATE trên các trường VAITRO, USERNAME, CMND

Cần ghi nhận lại khi thông tin USERNAME, CMND của bệnh nhân được cập nhật, đây là những thông tin quan trọng nhất của bệnh nhân, chỉ có thể bị chỉnh sửa bởi nhân viên, nếu bị chỉnh sửa không hợp lệ cần truy ra ai là người thực hiện

➔ Fine-grained audit trên bảng bệnh nhân cho hành động UPDATE trên các trường USERNAME, CMND

C. Tham khảo

McGraw-Hill Osborne - Effective Oracle Database 10g Security by Design
Slide môn học của thầy Lương Vĩ Minh và cô Phạm Thị Bạch Huệ