Séance de cours 11

Langage de contrôle de données (LCD SQL)

Objectifs de la séance

- Implanter la politique de sécurité de la base de données, à savoir:
  - Comprendre le processus de gestion de la sécurité des données.
  - Formuler les requêtes en utilisant le langage de contrôle de données (LCD SQL).

Plan de la séance

- Sécurité et SQL
  - Identification et authentification
  - Privilèges
  - Menaces à la sécurité
  - Mécanismes de protection
- Table virtuelle (VIEW)
  - Implémentation des tables virtuelles
  - Mise à jour de tables virtuelles
  - Hiérarchie de tables virtuelles

Plan de la séance

- Sécurité et SQL
  - Identification et authentification
  - Privilèges
  - Menaces à la sécurité
  - Mécanismes de protection
- Table virtuelle (VIEW)
  - Implémentation des tables virtuelles
  - Mise à jour de tables virtuelles
  - Hiérarchie de tables virtuelles

Le langage SQL

- *Structured Query Language*
- Norme établie pour SGBD relationnel
- Partie LDD (Langage de définition de données)
  - Conceptuel : CREATE SCHEMA, TABLE,…
  - Externe : CREATE VIEW, GRANT,…
  - Interne : CREATE INDEX, CLUSTER,…
- Partie LMD (Langage de manipulation de données)
  - SELECT, INSERT, DELETE, UPDATE
- **Partie LCD (Langage de contrôle de données)**
  - **GRANT, REVOKE**

Niveau externe du schéma en SQL

- Gestion de la sécurité
  - GRANT
  - REVOKE
- Tables virtuelles
  - VIEWS

Sécurité en SQL (GRANT)
Identification et authentification

- Identification des utilisateurs
  - *authorizationID*
    - PUBLIC : tous les utilisateurs
- Authentification
  - mot de passe, ou …
- Oracle
  - Utilisateurs administrateurs créés à l'installation
    - SYS, SYSTEM, …
  - Pour créer d'autres utilisateurs
    - CREATE USER *authorizationID* …
      - https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_8003.htm

Privilèges

```
GRANT listePrivilèges ON objet TO listeAuthorizationIDs
[WITH GRANT OPTION]
```

- *privilège* :

```
SELECT |
DELETE |
INSERT [listeColonnes]|
UPDATE [listeColonnes]|
REFERENCES listeColonnes|
USAGE
```

- *objet* :

```
[TABLE] nomTable |
DOMAIN nomDomaine |
CHARACTER SET nomCharacterSet
COLLATION nomCollation
TRANSLATION nomTranslation
```

Exemples

```
GRANT SELECT ON Commande        TO commisLivraison
GRANT SELECT ON LigneCommande   TO commisLivraison
```

```
GRANT SELECT, DELETE, INSERT, UPDATE ON Commande
     TO commisAchat
GRANT SELECT, DELETE, INSERT, UPDATE ON LigneCommande
     TO commisAchat
```

```
GRANT SELECT ON Article TO PUBLIC
```

```
GRANT UPDATE(quantitéEnStock) ON Article  TO commisLivraison
```

Privilèges (suite)

- Commandes LDD
  - propriétaire du schéma
- Création d 'une VIEW sur T
  - SELECT sur T
- FOREIGN KEY sur T
  - privilège REFERENCES sur T
- SQL:1999
  - ROLE = ensemble de privilèges, Oracle
  - nouveaux privilèges
    - TRIGGER ON TABLE *nomTable*
    - EXECUTE ON PROCEDURE/FUNCTION *nomProcOuFunc*
    - UNDER ON TYPE *nomType*

Suppression de privilèges

```
REVOKE [GRANT OPTION FOR] listePrivilèges ON objet
FROM listeIdUtilisateurs [RESTRICT | CASCADE]
```

Paramétrage de la sécurité à l'installation du SGBD

- Isoler le SGBD sur une machine à part
- Sécuriser le système d'exploitation
- Support d'installation sûr !
- Éliminer les services non essentiels
- Éliminer les utilisateurs pré-définis non essentiels
- Configurer les autres utilisateurs
- Configurer les mécanismes d'audit

Établissement de l'utilisateur d'une session SQL

- Agent SQL établit connexion avec serveur SQL
  - débute une session SQL
- Contexte de session maintenu par serveur
  - identificateur d'utilisateur courant SQL et/ou de ROLE
    - pseudo-colonne CURRENT_USER (USER avec Oracle) et CURRENT ROLE
  - nom de schéma
  - nom de catalogue
  - zône de temps
  - ensemble de caractères
- Établissement de l'utilisateur SQL courant
  - à la connexion, par un mécanisme non normalisé
  - paramètres de l'opération SQL CONNECT
  - appel à une routine SQL avec privilèges de la routine
    - créateur de la routine
    - gestion par pile lors d'une cascade d'appels
- Authentification *proxy* d'Oracle pour sécurité de bout en bout multitiers
  - session lègère de utilisateur *x* sur connexion pour utilisateur *y*

Mécanisme d'authentification

- Non normalisé en SQL
- Mot de passe
- Authentification biométrique (e.g. par empreinte digitale, reconnaissance de visages, …)
- Authentification par le système d'exploitation
- Certificat digital
- Authentification par la couche réseau
  - e.g. *Secure Sockets Layer* (SSL)
- Authentification centrale par un serveur d'authentification externe (*Single Sign-On* SSO)
  - e.g. serveur LDAP

Menaces à la sécurité

- Vol de mots de passe
- Menaces sur le réseau
  - interception de données
    - *sniffer* qui intercepte les paquets en transit
  - modification de données
  - usurpation d'identité
  - …

Mécanismes avancés

- *Cryptographie symétrique* (clé privée)
  - *Déchiffrer* (*cléSecrète*, *Chiffrer*(*cléSecrète* , *message*)) = *message*
  - norme *Data Encryption Standard* (DES)
- *Cryptogtraphie asymétrique* (clé publique)
  - *Déchiffrer* (*cléPrivée*, *Chiffrer*(*cléPublique* , *message*)) = *message*
  - système RSA (*Rivest, Shamir et Adleman*)
    - difficile de factoriser un grand nombre
    - *Déchiffrer*(*cléPublique*, *Chiffrer*(*cléPrivée*, *message*)) = *message*
  - employée dans *signatures digitales* et *certificats numériques*

Oracle Advanced Security (OAS)

- Plusieurs mécanismes d'authentification
- Encryptage des données
  - package PL/SQL
  - clause ENCRYPT sur colonne
- Identification/authentification unique
  - service d'annuaire *Oracle Internet Directory* (OID)
- *Oracle Virtual Private Database*
  - règles de sécurité complexes par API PL/SQL
- *Oracle Label Security*
  - niveaux de sécurité assigné à des lignes individuelles et aux utilisateurs
    - ex. public, secret, top secret, …

Plan de la séance

- Sécurité et SQL
  - Identification et authentification
  - Privilèges
  - Menaces à la sécurité
  - Mécanismes de protection
- Table virtuelle (VIEW)
  - Implémentation des tables virtuelles
  - Mise à jour de tables virtuelles
  - Hiérarchie de tables virtuelles

Table virtuelle (VIEW)

Table *Article*			
noArticle	**description**	**prixUnitaire**	**quantitéEnStock**
10	Cèdre en boule	10.99	10
20	Sapin	12.99	10
40	Epinette bleue	25.99	10
50	Chêne	22.99	10
60	Erable argenté	15.99	10
70	Herbe à puce	10.99	10
80	Poirier	26.99	10
81	Catalpa	25.99	10
90	Pommier	25.99	10
95	Génévrier	15.99	10

```
CREATE VIEW ArticlePrixModique AS
     SELECT    noArticle, description, prixUnitaire
     FROM      Article
     WHERE     prixUnitaire < 15
```

```
SELECT     *
FROM       ArticlePrixModique
```

VIEW *ArticlePrixModique*		
noArticle	**description**	**prixUnitaire**
10	Cèdre en boule	10.99
20	Sapin	12.99
70	Herbe à puce	10.99

Implémentation des tables virtuelles

- ***Résolution des vues par modification de requête***

```
CREATE VIEW ArticlePrixModique AS
     SELECT    noArticle, description, prixUnitaire
     FROM      Article
     WHERE     prixUnitaire < 15
```

```
     SELECT    *
     FROM      ArticlePrixModique
```

```
SELECT *
FROM (
     SELECT    noArticle, description, prixUnitaire
     FROM      Article
     WHERE     prixUnitaire < 15)
```

Résolution des vues par matérialisation

- Table stockée
- Redondance
- Maintenance de la cohérence
- Meilleure performance du SELECT
- Moins bonne performance des mises à jour
- Entrepôts de données

Mise à jour de tables virtuelles

- SQL2
  - une seule table
  - sans DISTINCT
  - colonnes simples
  - pas de SELECT imbriqué

```
SELECT nomColonne, [nomColonne] FROM T WHERE conditionSQL
```

- SQL:1999
  - spécification très complexe

Exemple de mise à jour par modification de requête

VIEW *ArticlePrixModique*		
noArticle	**description**	**prixUnitaire**
10	Cèdre en boule	10.99
20	Sapin	12.99
70	Herbe à puce	10.99

```
DELETE    FROM ArticlePrixModique
WHERE     noArticle = 20
```

```
DELETE    FROM Article
WHERE     noArticle = 20 AND prixUnitaire < 15
```

Table *Article*			
noArticle	**description**	**prixUnitaire**	**quantitéEnStock**
10	Cèdre en boule	10.99	10
20	Sapin	12.99	10
40	Epinette bleue	25.99	10
50	Chêne	22.99	10
60	Erable argenté	15.99	10
70	Herbe à puce	10.99	10
80	Poirier	26.99	10
81	Catalpa	25.99	10
90	Pommier	25.99	10
95	Génévrier	15.99	10

Problèmes de mise à jour d'une table virtuelle

```
INSERT INTO ArticlePrixModique VALUES(200,'Viagra',50.99)
```

```
INSERT INTO Article VALUES(200,'Viagra',50.99,0)
```

Table *Article*			
noArticle	**description**	**prixUnitaire**	**quantitéEnStock**
10	Cèdre en boule	10.99	10
20	Sapin	12.99	10
40	Epinette bleue	25.99	10
50	Chêne	22.99	10
60	Erable argenté	15.99	10
70	Herbe à puce	10.99	10
80	Poirier	26.99	10
81	Catalpa	25.99	10
90	Pommier	25.99	10
95	Génévrier	15.99	10
200	Viagra	50.99	0

VIEW *ArticlePrixModique*		
noArticle	**description**	**prixUnitaire**
10	Cèdre en boule	10.99
20	Sapin	12.99
70	Herbe à puce	10.99

- Sémantique incohérente…

Rejet de mise à jour incohérente avec WITH CHECK OPTION

```
CREATE VIEW ArticlePrixModique AS
     SELECT    noArticle, description, prixUnitaire
     FROM      Article
     WHERE     prixUnitaire < 15
WITH CHECK OPTION
```

```
INSERT INTO ArticlePrixModique VALUES(200,'Viagra',50.99)
                              {Insertion rejetée}
```

```
UPDATE ArticlePrixModique
SET prixUnitaire = 20.99
WHERE noArticle = 10 {Modification rejetée}
```

Hiérarchie de tables virtuelles

```
CREATE VIEW ArticlePrixModique AS
     SELECT    noArticle, description, prixUnitaire
     FROM      Article
     WHERE     prixUnitaire < 15
```

```
CREATE VIEW ArticlePrixMoyen AS
SELECT    noArticle, description, prixUnitaire
FROM      ArticlePrixModique
WHERE     prixUnitaire > 12
WITH CASCADED CHECK OPTION
```

```
INSERT INTO ArticlePrixMoyen VALUES(200,'Viagra',50.99)
                             {Insertion rejetée}
```

```
CREATE VIEW ArticlePrixMoyen AS
SELECT    noArticle, description, prixUnitaire
FROM      ArticlePrixModique
WHERE     prixUnitaire > 12
WITH LOCAL CHECK OPTION
```

```
INSERT INTO ArticlePrixMoyen VALUES(200,'Viagra',50.99)
                             {Insertion acceptée}
```

Renommer les colonnes d'une VIEW

```
CREATE VIEW  TotalCommande (noCommande, totalCommande, totalPlusTaxe) AS
       SELECT noCommande, SUM(quantité*prixUnitaire),
              SUM(prixUnitaire*quantité*1.15)
       FROM        LigneCommande AS L, Article AS A
       WHERE       L.noArticle = A.noArticle
       GROUP BY    noCommande
```

N.B. Modification interdite

VIEW *TotalCommande*		
noCommande	**totalCommande**	**totalPlusTaxe**
1	190.84	219.47
2	99.95	114.94
3	12.99	14.94
4	48.98	56.33
5	152.87	175.80
6	190.84	219.47
7	54.97	63.22
8	38.97	44.82

Indépendance logique des données et encapsulation par les tables virtuelles

- *Catalogue* (*noArticle*, *description*, *prixUnitaire*)
- *Inventaire* (*noArticle*, *quantitéEnStock*)

```
CREATE VIEW Article (noArticle, description, prixUnitaire, quantitéEnStock)AS
       SELECT     C.noArticle, description, prixUnitaire, quantitéEnStock
       FROM       Catalogue AS C, Inventaire AS I
       WHERE      C.noArticle = I.noArticle
```

Sécurité par les tables virtuelles

```
CREATE VIEW User_Tables AS
     SELECT *
     FROM Tables
     WHERE Table_Owner = CURRENT_USER
```

```
GRANT SELECT ON User_Tables TO PUBLIC
```

Schéma interne

- Non standardisé
  - organisation primaire de la table
  - organisations secondaires (INDEX)

```
CREATE INDEX indexNoComNoArtDétLiv
ON DétailLivraison (noCommande, noArticle)
```

Référence bibliographique

- Godin, R. (2012). *Systèmes de gestion de bases de données par l'exemple. 3ième édition, Montréal,* Canada: Loze-Dion, Chapitre 4.

Prochain cours

- Théorie de la normalisation