Sécurité des données

Grégoire Roumache

Septembre 2020

Table des matières

Attention ! user $\neq$ login

- login = donne accès au serveur (pas aux bases de données)
- user = donne accès, à un login, à la base de données

Remarque: il n'y a pas de labo 5.

1 Théorie

1.1 Trucs du rappel de base qu'on n'avait pas vu

- Une **vue** est une table virtuelle, elle sert à donner un nom à la requête pour l'utiliser souvent.
- Une **procédure stockée** est une instruction SQL précompilée et enregistrée dans la base de données.
- un **trigger** est un méchanisme qui permet d'exécuter une fonction/procédure lorsqu'un événement survient.
- Une **injection SQL** est une méthode d'exploitation de faille de sécurité. Il s'agit d'injecter du code SQL dans une requête via une entrée de page web.
- Abréviations:

  - **DDL** = Data Definition Language (ex: create, drop)
  - **DQl** = Data Query Language (ex: select)
  - **DML** = Data Manipulation Language (ex: insert, delete)
  - **DCL** = Data Control Language (ex: grant, revoke)
- Créer une vue:

```
CREATE VIEW <schema>.<view_name> AS SELECT ...
```

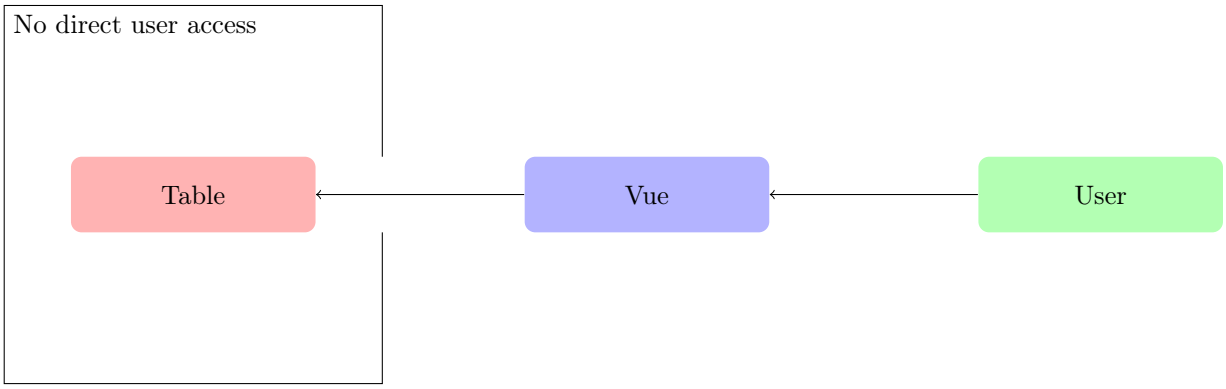
- Créer une séquence (peut-être utilisé comme IDENTITY pour créer des valeurs par défaut d'une colonne):

```
CREATE SEQUENCE dbo.InvoiceSeq AS INT
    START WITH 1
    INCREMENT BY 1;

-- Récupérer la valeur suivante dans la séquence
SELECT NEXT VALUE FOR dbo.InvoiceSeq;
```

1.2 Administration

- Autorisations:
  - Accorder une autorisation signifie que l'utilisateur peut accéder à l'objet.
  - Un refus d'autorisation l'emporte sur un accord d'autorisation.
- Authentification - Identification:
  - identification = établir son identité (ex: donner son login)
  - authentification = prouver son identité (ex: donner son mot de passe)
- Un compte SQL Server est enregistré sur le serveur SQL et n'est pas associé à Active Directory ou Windows.
- On peut créer un **compte invité** pour les utilisateurs sans compte enregistré.
- Chaînage des propriétés (en anglais: *ownership chaining*, pas: *property chaining*):
  - tous les objets (objets = tables, vues, fonction, etc.) en SQL on un propriétaire
  - on ne peut accéder à un objet que si on a les autorisations adéquates
  - quand un objet dépend d'un autre (ex: une vue qui référence une table), une chaîne de propriétés est établie
  - ça veut dire qu'un utilisateur qui a accès à la vue peut accéder aux données de la table indirectement sans avoir la permission d'accéder à la table directement (seulement si le propriétaire de la table et de la vue est le même utilisateur)



- Types de sauvegardes de base de données:
  - sauvegarde complète = toutes les données + logs
  - sauvegarde différentielle = données modifiées/ajoutées sur base de la dernière sauvegarde complète
  - sauvegarde incrémentielle = données modifiées/ajoutées depuis la dernière sauvegarde (peut être une sauvegarde complète ou incrémentielle)
- L'utilisation de sauvegardes de fichiers accélére la récupération en ne restaurant que les fichiers endommagés sans restaurer le reste de la base de données.

1.3 Protocole SNMP

- SNMP = simple network management protocol, sert à:
  - faire du network monitoring
  - changer l'état d'un équipement (ex: activer/désactiver une interface)
- 2 types d'entités snmp: les managers et les agents.
- Manager snmp:
  - serveur nms (= network management system)
  - il envoie des requêtes, et reçoit des réponses
  - il peut aussi recevoir des *traps* (quand quelque chose se passe chez l'agent)
- Agent snmp: service ou démon qui tourne sur une machine.
- SMI = structure of management information, manière standardisée d'organiser les objets gérés et leur comportement.
- MIB = management information base, base de donnée des objets gérés par un agent
- SMI – MIB:
  - SMI: ensemble de règles qui définissent la MIB
  - MIB: Base de donnée des objets
- MIB Remote Monitoring (RMON), collecte des statistiques réseau
  - La sonde de l'agent collecte les informations et construit les statistiques
  - Statistiques envoyée à intervalles réguliers ou sur demande du manager

2 Pratique

2.1 Backup et restauration en 100% SQL

Remarque: supprimer la base données avant de la restaurer.

- Créer un backup en SQL (le dossier `backup` doit exister):

```
USE master
GO
BACKUP DATABASE <nom_db> TO DISK = 'C:\backup\db.bak'
    -- ajouter l'option suivante pour encrypter
    WITH ENCRYPTION (
        ALGORITHM = AES_256,
        SERVER CERTIFICATE = MyCertificate
    )
```

- Créer un backup différentiel en SQL:

```
-- 1. Créer un backup complet
BACKUP DATABASE <nom_db> TO DISK = 'C:\backup\db_full.bak' WITH INIT;
GO

-- 2. Créer un backup différentiel
BACKUP DATABASE <nom_db> TO 'C:\backup\db_diff.bak' WITH DIFFERENTIAL;
```

- Créer une clé d'encryption en SQL (algorithme de chiffrage = AES 256):

```
-- Créer la clé de chiffrement (clé principale de chiffrement)
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '23987hxJ#KL95234nl0zBe';
```

- Créer un certificat en SQL (attention changer la date d'expiration):

```
-- Ouvrir la clé principale de chiffrement dans la session où on crée le certificat
OPEN MASTER KEY DECRYPTION BY PASSWORD = '23987hxJ#KL95234nl0zBe'

-- Créer le certificat encrypté avec la clé de chiffrement principale
CREATE CERTIFICATE MyCertificate
    WITH SUBJECT = 'Backup certificate',
    EXPIRY_DATE = '20201031';
```

- Restaurer une base de données en SQL:

```
USE master
GO
RESTORE DATABASE <nom_db> FROM DISK = 'C:\backup\db.bak'
```

- Restaurer une base de données différentielle en SQL:

```
-- 1. Restaurer la base de données complète (avec l'option NORECOVERY)
RESTORE DATABASE <nom_db> FROM DISK = 'C:\backup\db_full.bak'
   WITH NORECOVERY;
GO

-- 2. Restaurer le backup différentiel
RESTORE DATABASE <nom_db> FROM DISK = 'C:\backup\db_diff.bak'
   WITH FILE = 2,
   RECOVERY;
```

2.2 Backup et restauration en 100% UI

Remarque: supprimer la base données avant de la restaurer.

- Créer un backup dans l'UI:
  1. click-droit sur la base de données, click sur *tasks*, click sur *back-up*
  2. choisir le type avec l'option *backup style* (*full* ou *differential*)
  3. en bas à droite, click sur *add* pour sélectionner l'emplacement du backup
  4. pour écraser le backup précédent, sélectionner *media options* dans le menu de gauche et click sur *back up to a new media set and erase all existing backup sets*
  5. pour encrypter le backup, sélectionner *backup options* dans le menu de gauche, click sur *encrypt backup*, sélectionner l'algorithme et le certificat

Remarque: pour encrypter le backup, il faut écraser les backups précédents.

- Restaurer une base de données dans l'UI:
  1. dans le menu de gauche, click-droit sur *databases*, puis click sur *restore database*
  2. sélectionner l'option *device*, puis click sur ... à droite
  3. ensuite, click sur *Add* et sélectionner le fichier de backup
  4. si vous restaurez àpd backup différentiel, ajouter le fichier de backup complet et le fichier différentiel
- Générer un script de restauration:
  1. click-droit sur la base de données, puis click sur *tasks*, puis sur *generate scripts*
  2. pour ne restaurer que les tables, dans le menu *choose objects*, sélectionner l'option *select specific database objects* et sélectionner *tables*

  Remarques:
  - supprimer la base de données avant de la restaurer avec le script
  - pour ne restaurer que les tables, supprimer uniquement les tables

2.3 Authentifications et autorisations

- Créer un login:
  - Compte windows: `CREATE LOGIN [<domain>/<login>] FROM WINDOWS`
  - Compte SQL: `CREATE LOGIN <login> WITH PASSWORD = '<password>'`
- Afficher la liste des rôles:
  - Tous les rôles: `EXEC sp_helprole`.
  - Rôles par défaut de la DB: `EXEC sp_helpsrvrole`.

2.4 Automatisation, Audit, Optimisation

- Pour avoir des jobs automatisés, il faut activer le SQL Server Agent:
  1. dans le menu de gauche, faire un clic droit sur *SQL Server Agent*
  2. cliquer sur *start*, ensuite cliquer sur *yes* dans toutes les fenêtres pop up
- Créer un job:
  1. dans le menu de gauche, faire un clic droit sur *SQL Server Agent/Jobs*
  2. cliquer sur *new job*
  3. dans *general*, donner un nom au job
  4. dans *steps*:
     (a) cliquer sur *new*
     (b) donner un nom à la step
     (c) dans *command*, mettre la query SQL à exécuter
     (d) appuyer sur *ok*
  5. dans *schedules*:
     (a) cliquer sur *new*
     (b) donner un nom au schedule
     (c) configurer le moment d'exécution
     (d) appuyer sur *ok*
  6. appuyer sur *ok*

- Ouvrir SQL Profiler : dans le menu du haut, cliquer sur *tools*, puis *SQL Profiler*.
- Analyser des requêtes:
  1. sélectionner la requête, puis faire un clic droit
  2. cliquer sur *display estimated execution plan*

2.5 Vérifications SQL

- Afficher les logins:

```
SELECT name, type, type_desc FROM master.sys.sql_logins
```

- Afficher les utilisateurs:

```
SELECT * FROM sysusers
```

- Afficher les rôles:

```
SELECT name FROM sys.database_principals WHERE type = 'R'
```

- Afficher les rôles et leurs membres:

```
SELECT DP1.name AS RoleName, isnull (DP2.name, '/') AS UserName
FROM sys.database_role_members AS DRM
RIGHT OUTER JOIN sys.database_principals AS DP1 ON DRM.role_principal_id = DP1.principal_id
LEFT OUTER JOIN sys.database_principals AS DP2 ON DRM.member_principal_id = DP2.principal_id
ORDER BY DP1.name
```

- Afficher les bases de données:

```
SELECT name FROM master.dbo.sysdatabases
```

- Afficher les permissions d'un utilisateur:

```
SELECT * FROM fn_my_permissions('<user>', 'USER')
```

A Labo 3 – Authentification & Autorisation des Utilisateurs

- Exercice 1 – Créer différents 'logins' SQL qui utiliseront plusieurs méthodes d'authentification:
  - Utiliser CMD pour localiser l'utilisateur et ses groupes locaux de sécurité.

    Dans CMD: `whomai /groups`
  - Utiliser T-SQL pour créer un login d'authentification de type 'SQL Server' nommé 'UsrSQL'.

    ```
    CREATE LOGIN UsrSQL WITH PASSWORD = '<password>';
    ```
  - Utiliser SMS pour créer un login de type 'Windows Authenticated' de l'AD (etuXXXXX).

    ```
    CREATE LOGIN [<domain>\<login>] FROM WINDOWS;
    ```
  - Utiliser T-SQL pour ajouter votre login AD dans le rôle fixe du server 'setupadmin'.

    Attention ! Erreur du prof car, on ne peut ajouter que des utilisateurs à un rôle, pas des logins.

    ```
    -- EXEC sp_addrolemember @rolename = setupadmin, @membername = UserSQL;
        -- => "do not have permission"
    -- ALTER ROLE setupadmin ADD MEMBER UserSQL;
        -- => "do not have permission"
    ==> clic-droit sur <server>/security/logins/UsrSQL
    ==> properties -> server roles -> setupadmin -> ok
    ```
  - Utiliser SSMS pour ajouter le login 'UsrSQL' dans le rôle fixe du server 'bulkadmin'.

    ```
    ==> clic-droit sur <server>/security/logins/UsrSQL
    ==> properties -> server roles -> bulkadmin -> ok
    ```
  - Utiliser la bonne procédure stockée qui vérifie l'appartenance au niveau des groupes.

    ```
    EXEC sp_helprolemember <role>;
    EXEC sp_helpsrvrolemember <role>; -- rôles serveur
    ```
- Exercice 2 – Créer différents 'rôles' définis au niveau du server SQL:
  - Utiliser SMS pour créer un rôle défini nommé 'LoginManager'.

    ```
    ==>  clic-droit sur <server>/databases/<database>/security/roles/database roles
    ==> -> new database role -> role name -> ok
    ```
  - Permettre à ce rôle d'accorder la permission 'ALTER ANY LOGIN'.

    ```
    GRANT ALTER ANY LOGIN TO UserSQL;
    ```
  - Ajouter votre utilisateur AD ou local dans ce rôle défini.

    ```
    ALTER ROLE LoginManager ADD MEMBER UserSQL;
    ```
  - Utiliser T-SQL pour créer un rôle défini nommé 'CreatorManager'.

    ```
    EXEC sp_addrole @rolename = 'CreatorManager';
    -- CREATE ROLE CreatorManager;
    ```
  - Permettre à ce rôle d'accorder la permission 'CREATE ANY DATABASE'.

    ```
    USE master;
    GO
    GRANT CREATE ANY DATABASE TO CreatorManager;
    ```
  - ajouter l'utilisateur 'UsrSQL' dans ce rôle avec T-SQL.

```
EXEC sp_addrolemember @rolename = 'CreatorManager', @membername = 'UserSQL';
-- ALTER ROLE CreatorManager ADD MEMBER UserSQL;
```

- Exercice 3 – Créer différents 'utilisateurs' définis au niveau des bases de données basés sur les 'logins' créés précédemment:
  - Utiliser SSMS pour ajouter votre login AD dans le rôle 'db_datareader' pour la DB 'AdventureWorks'.
    ```
    ==> clic-droit sur <serveur>/security/logins/<login> -> properties
    ==> -> user mapping -> AdventureWorks -> db_datareader -> ok
    ```
  - Utiliser T-SQL pour créer un role au niveau de la DB nommé 'TableAdmin' pour la DB 'AdventureWorks'.
    ```
    USE AdventureWorks;
    GO
    EXEC sp_addrole @rolename = 'TableAdmin';
    -- CREATE ROLE TableAdmin;
    ```
  - Donner les permissions 'CREATE TABLE' et 'CREATE SCHEMA' au rôle 'TableAdmin'
    ```
    GRANT CREATE TABLE, CREATE SCHEMA TO TableAdmin;
    ```
- Exercice 4 – Configurer les permissions au niveau des bases de données, schémas et rôles.
  - S'identifier sur l'instance par défaut avec l'utilisateur 'KimPossible'.
    1. Créer l'utilisateur *KimPossible*:
       `CREATE LOGIN KimPossible WITH PASSWORD = '<password>';`
    2. Se connecter avec *KimPossible*:
       `SETUSER KimPossible;`
       Si ça ne marche pas, il faut se connecter avec SSMS
       `==> clic-droit sur <serveur> -> connect.`
  - Créer une nouvelle base de données nommée 'Middleton'.
    ```
    CREATE DATABASE MiddleTon;
    ```
  - Créer un nouveau rôle nommé 'TeamImpossible'.
    ```
    CREATE ROLE TeamImpossible;
    ```
  - Créer 3 tables nommés par défaut 'Website', 'Babysitting', 'LowMowning'.
    ```
    CREATE TABLE Website (Id INT);
    CREATE TABLE Babysitting (Id INT);
    CREATE TABLE LowMowning (Id INT);
    ```
  - Créer un nouveau schéma nommé 'KimServices'.
    ```
    CREATE SCHEMA KimServices;
    ```
  - Octroyer au rôle 'TeamImpossible' les privilèges suivants 'SELECT' 'REFERENCES' 'INSERT' 'UPDATE' 'DELETE' et 'VIEW DEFINITION' sur le schéma 'KimServices'
    ```
    GRANT SELECT, REFERENCES, INSERT, UPDATE, DELETE, VIEW DEFINITION
    TO KimServices;
    ```
- Commandes pour annuler tout ce qui a été modifié dans ce labo:

```
    -- Exercice 1
DROP LOGIN UserSQL;
DROP LOGIN [<domain>\<login>];
    -- Exercice 2
DROP ROLE LoginManager;
DROP ROLE CreatorManager;
```

```
    -- Exercice 3
USE AdventureWorks;
GO
DROP ROLE TableAdmin;
    -- Exercice 4
DROP DATABASE Middleton;
```

B Labo 4 – Gestion des Accès

- À partir d'un besoin client :
  - Établir la liste des utilisateurs devant avoir accès aux bases de données.
  - Regrouper des objets de bases de données en schéma.
  - Définir les permissions des utilisateurs pour chaque objet de bases de données.
  - Établir une matrice des rôles qui fait le lien entre les utilisateurs et les objets de bases de données.
- Exercice:
  1. Lister les utilisateurs et les comptes pour chaque base de données.

     Utilisateurs:
     - Patrick, président
     - André, administrateur système
     - Pierre, responsable du refuge des champs
     - Paul, responsable du refuge des prés
     - Jacques, responsable du refuge des plages
     - Jean, responsable du refuge des plaines
     - 1 responsable des inscriptions, dans chaque refuge
     - 1 médecin, dans l'association
     - 1 soigneur, dans l'association
     - les utilisateurs externes
  2. Attribuer les rôles serveur et de base de données pour chaque utilisateur.

     Rôles:
     - président
     - administrateur système
     - responsable du refuge
     - responsable des inscriptions
     - médecin
     - soigneur
     - utilisateurs externes
  3. Créer la matrice des rôles (pour chaque base de données) en respectant le canevas.

     Matrice des **autorisations**:

	select	insert	update	delete
président	*	*	*	*
administrateur système	*	*	*	*
responsable du refuge	*	*	*	*
responsable des inscriptions	A,R	A	A	/
médecin	A,R,S	S	A,S	S
soigneur	A,R,S	S	S	/
utilisateurs externes	A,R	/	/	/

* = tout, A = animal, L = localisation, S = soin

4. Créer les 4 bases de données ainsi que les utilisateurs en respectant la matrice des rôles.

   – Restaurer la base de données 4 fois:

```
USE master
GO
RESTORE DATABASE HENACCUEIL1 FROM DISK = 'C:\backup\db.bak'

-- Remarque: Si on essaie de restaurer une 2ème fois la même base de données,
-- ça va donner des erreurs et pointer vers les fichiers logiques à modifier,
-- ici: 'Examen', 'Examen_log'

RESTORE DATABASE HENACCUEIL2 FROM DISK = 'E:\HENACCUEIL.bak' WITH
    MOVE 'Examen' TO 'E:\HENACCUEIL2',
    MOVE 'Examen_log' TO 'E:\HENACCUEIL2_log'
GO

RESTORE DATABASE HENACCUEIL3 FROM DISK = 'E:\HENACCUEIL.bak' WITH
    MOVE 'Examen' TO 'E:\HENACCUEIL3',
    MOVE 'Examen_log' TO 'E:\HENACCUEIL3_log'
GO

RESTORE DATABASE HENACCUEIL4 FROM DISK = 'E:\HENACCUEIL.bak' WITH
    MOVE 'Examen' TO 'E:\HENACCUEIL4',
    MOVE 'Examen_log' TO 'E:\HENACCUEIL4_log'
GO
```

   – Créer les logins:

```
CREATE LOGIN Patrick WITH PASSWORD = 'password'
CREATE LOGIN Andre   WITH PASSWORD = 'password'
CREATE LOGIN Pierre  WITH PASSWORD = 'password'
CREATE LOGIN Paul    WITH PASSWORD = 'password'
CREATE LOGIN Jacques WITH PASSWORD = 'password'
CREATE LOGIN Jean    WITH PASSWORD = 'password'
```

   – Créer les utilisateurs:

```
CREATE USER Patrick
CREATE USER <responsable_du_refuge>
CREATE USER Jean
```

   – Créer les rôles:

```
CREATE ROLE President
CREATE ROLE AdministrateurSysteme
CREATE ROLE ResponsableRefuge
```

```
CREATE ROLE ResponsableInscriptions
CREATE ROLE Medecin
CREATE ROLE Soigneur
-- utilisateurs externes = pas de rôle
```

  – Ajouter les utilisateurs aux rôles:

```
ALTER ROLE President             ADD MEMBER Patrick
ALTER ROLE AdministrateurSysteme ADD MEMBER André
ALTER ROLE ResponsableRefuge     ADD MEMBER <responsable_du_refuge>
```

  – Donner les permissions aux rôles:

```
-- par défaut: DENY ALL ON ALL TO PUBLIC
GRANT ALL ON ALL TO President, AdministrateurSysteme, ResponsableRefuge

GRANT SELECT, INSERT, UPDATE ON ANIMAL TO ResponsableInscriptions
GRANT SELECT ON REFUGE TO ResponsableInscriptions

GRANT SELECT, UPDATE ON ANIMAL TO Medecin
GRANT SELECT ON REFUGE TO Medecin
GRANT ALL ON SOIN TO Medecin

GRANT SELECT ON ANIMAL, REFUGE TO Soigneur
GRANT SELECT, INSERT, UPDATE ON SOIN TO Soigneur

GRANT SELECT ON ANIMAL, REFUGE TO PUBLIC
```

- Commandes pour annuler tout ce qui a été modifié dans ce labo:

```
DROP LOGIN Patrick
DROP LOGIN Andre
DROP LOGIN Pierre
DROP LOGIN Paul
DROP LOGIN Jacques
DROP LOGIN Jean
-- montrer tous les logins
SELECT name, type, type_desc FROM master.sys.sql_logins

DROP ROLE President
DROP ROLE AdministrateurSysteme
DROP ROLE ResponsableRefuge
DROP ROLE ResponsableInscriptions
DROP ROLE Medecin
DROP ROLE Soigneur
-- montrer tous les rôles et les utilisateurs membres
SELECT DP1.name AS RoleName, isnull (DP2.name, '/') AS UserName
    FROM sys.database_role_members AS DRM
    RIGHT OUTER JOIN sys.database_principals AS DP1
        ON DRM.role_principal_id = DP1.principal_id
    LEFT OUTER JOIN sys.database_principals AS DP2
        ON DRM.member_principal_id = DP2.principal_id
    ORDER BY DP1.name
-- montrer tous les rôles
SELECT name FROM sys.database_principals WHERE type = 'R' -- R = rôle

DROP DATABASE HENACCUEIL1
DROP DATABASE HENACCUEIL2
```

```
DROP DATABASE HENACCUEIL3
DROP DATABASE HENACCUEIL4
-- montrer toutes les bases de données
SELECT * FROM master.dbo.sysdatabases
```

C Labo 6 – Index

- Restaurer la base de données.

```
USE master
GO
RESTORE DATABASE BigDB FROM DISK = 'E:\databases\bigDB_Light.bak'
```

- Supprimer le cache de SQL Server.

```
CHECKPOINT
GO
DBCC DROPCLEANBUFFERS
```

- Activer la mesure du temps d'exécution des requêtes.

```
SET STATISTICS TIME ON
```

- Exercice 1:
  - Exécuter la requête suivante : `SELECT COUNT(*) FROM users WHERE age < 50` (4956 ms)
  - Créer un index pour accélérer la requête et mesurer le temps d'exécution.

```
-- essayer avec l'age
-- essayer avec l'id
-- essayer avec l'id + age
-- essayer avec l'id + la condition
-- essayer avec l'age + la condition
CREATE INDEX i1 ON users(age)                -- build = 247116 ms, select =  389 ms
CREATE INDEX i2 ON users(id)                 -- build =  75506 ms, select = 4821 ms
CREATE INDEX i3 ON users(id, age)            -- build =  99161 ms, select = 1566 ms
CREATE INDEX i4 ON users(id) WHERE age < 50  -- build =  59025 ms, select =  459 ms
CREATE INDEX i5 ON users(age) WHERE age < 50 -- build = 137070 ms, select =  387 ms
                                             -- sans index,        select = 4956 ms
                                             -- sans index, cache, select = 1641 ms
```

- Exercice 2:
  - Exécuter la requête suivante : `SELECT * FROM users WHERE age > 99`
  - Désactiver l'index de l'exercice 1.

```
ALTER INDEX i1 ON users DISABLE
```

  - Créez un index plus petit que celui de l'exercice 1 et qui donnera les mêmes performances.

```
CREATE INDEX i6 ON users(age) WHERE age > 99
SELECT * FROM users WHERE age > 99      -- build = 1977 ms, select =   58 ms
                                        -- sans index,      select = 4944 ms
```

  - Utiliser l'instruction suivante pour voir la taille des index de la db:

```
SELECT
    i.[name] AS IndexName,
    SUM(s.[used_page_count]) * 8 AS IndexSizeKB
FROM
```

```
    sys.dm_db_partition_stats AS s
INNER JOIN
    sys.indexes AS i
ON
    s.[object_id] = i.[object_id]
    AND s.[index_id] = i.[index_id]
GROUP BY i.[name]
ORDER BY i.[name]
GO
```

- Comparer la taille des index des exercices 1 et 2.

```
- i1 = 1'387'672 ko
- i2 =   990'616 ko
- i3 = 1'387'680 ko
- i4 =   490'432 ko
- i5 =   686'952 ko
- i6 =     6'968 ko
```

- Exercice 3:
  - Exécutez la requête suivante : `SELECT TOP 100 * FROM users ORDER BY nom DESC, prenom DESC, age ASC`
  - Créez des index sur nom, prénom et sur âge. Exécutez, ensuite, à nouveau la requête.

```
CREATE INDEX i7 ON users(nom)
CREATE INDEX i8 ON users(prenom)
CREATE INDEX i1 ON users(age)       -- existe déjà
```

  - Créez un seul nouvel index qui sera vraiment efficace (penser à un annuaire).

```
CREATE INDEX i9 ON users(nom DESC, prenom DESC, age ASC)
```

- Exercice 4:
  - Regarder la place que les index prennent sur le disque dur.
    1. Clic droit sur la base de données.
    2. Cliquer sur *Reports*.
    3. Cliquer sur *Standard Reports*.
    4. Cliquer sur *Disk Usage by Top Tables*.
  - Comparez l'espace occupé par les index avec l'espace pris par la table.

```
- user = 6'199'284 ko
-   i1 = 1'387'672 ko (index le plus large)
```

- Commandes pour annuler tout ce qui a été modifié dans ce labo:

```
DROP INDEX i1 ON users
DROP INDEX i2 ON users
DROP INDEX i3 ON users
DROP INDEX i4 ON users
DROP INDEX i5 ON users
DROP INDEX i6 ON users
DROP INDEX i7 ON users
DROP INDEX i8 ON users
DROP INDEX i9 ON users
SET STATISTICS TIME OFF
USE master
```

```
DROP DATABASE BigDB
```

D Labo 7 – Audit et journalisation

- Restaurez la base de données présente sur Moodle.

```
USE master
RESTORE DATABASE SecuData FROM DISK = 'E:\databases\DB_SecuData_Audit.bak'
```

- Créer les utilisateurs:
  - Créez un utilisateur avec un mot de passe respectant la stratégie évoquée plutôt, en lui donnant le rôle le plus approprié et l'accès en lecture sur la table Personne.
  - Créez un autre utilisateur qui aura les droits propriétaires sur la base de données.

```
USE SecuData
CREATE LOGIN User1 WITH PASSWORD = '??Password007??'
CREATE LOGIN User2 WITH PASSWORD = '??Password007??'
GO
CREATE USER User1
CREATE USER User2
GO
GRANT SELECT ON Personne TO User1
GRANT ALL ON DATABASE::SecuData TO User2
GO
```

Remarque:

  - le prof avait déjà créé les utilisateurs *user1*, et *user2*
  - il faut les supprimer puis les rajouter

- Créez un Audit pour chacun des utilisateurs.
  - Pour le premier : le type d'action ciblera les Select et les Insert.
  - Pour le second : le type d'action ciblera les Execute et les Delete.

```
USE master
CREATE SERVER AUDIT AuditManip1
TO FILE (FILEPATH = 'E:\databases\')
ALTER SERVER AUDIT AuditManip1 WITH (STATE = ON);
CREATE SERVER AUDIT AuditManip2
TO FILE (FILEPATH = 'E:\databases\')
ALTER SERVER AUDIT AuditManip2 WITH (STATE = ON);

USE SecuData
CREATE DATABASE AUDIT SPECIFICATION AuditManip1Specification
FOR SERVER AUDIT AuditManip1
ADD (SELECT ON DATABASE::SecuData BY User1),
ADD (INSERT ON DATABASE::SecuData BY User1)
WITH (STATE = ON);
CREATE DATABASE AUDIT SPECIFICATION AuditManip2Specification
FOR SERVER AUDIT AuditManip2
ADD (EXECUTE ON DATABASE::SecuData BY User2),
ADD (DELETE  ON DATABASE::SecuData BY User2)
WITH (STATE = ON);
```

- Affichez les noms et les prénoms des personnages ayant 16 ans avec le premier utilisateur.

```
EXECUTE AS USER = 'User1'
SELECT nom, prenom FROM personnages WHERE age = 16
REVERT
```

- Ajoutez une ligne dans la table Personne avec le second utilisateur et supprimez la ligne avec l'ID = 1.

```
EXECUTE AS USER = 'User2'
INSERT INTO Personne (nom, prenom, age) VALUES ('Grégoire', 'Roumache', 21)
DELETE FROM Personne WHERE id = 1
REVERT
```

- Consultez les journaux d'audit.

1. Dans SSMS, faire un clic droit sur *<serveur>/security/audits/<audit>*.
2. Cliquer sur *view audit logs* (*afficher les journaux d'audit*).

- Commandes pour annuler tout ce qui a été modifié dans ce labo:

```
USE SecuData
ALTER DATABASE AUDIT SPECIFICATION AuditManip1Specification WITH (STATE = OFF)
ALTER DATABASE AUDIT SPECIFICATION AuditManip2Specification WITH (STATE = OFF)
DROP DATABASE AUDIT SPECIFICATION AuditManip1Specification
DROP DATABASE AUDIT SPECIFICATION AuditManip2Specification
USE master
ALTER SERVER AUDIT AuditManip1 WITH (STATE = OFF);
ALTER SERVER AUDIT AuditManip2 WITH (STATE = OFF);
DROP SERVER AUDIT AuditManip1
DROP SERVER AUDIT AuditManip2
DROP LOGIN User1
DROP LOGIN User2
GO
DROP DATABASE SecuData
```

E Labo 8

Si vous avez ce problème sous Debian : "Bad operator (INTEGER): At line 73 in /usr/share/snmp/mibs/ietf/SNMPv2-PDU"

Voici la commande qui peut réparer l'erreur : "wget http://pastebin.com/raw.php?i=p3QyuXzZ -O /usr/share/snmp/mibs/ietf/SNMPv2-PDU"