

Base de données

- ✓ optimiser
- ✓ sécuriser
- ✓ relier

1010
1010
1010



© istockphoto.com/Nishan Sothilingam

Aujourd'hui, la base de données est plus que jamais au cœur des applications. La donnée, plus que le SGBD, est devenue un enjeu central. Comment traiter une masse d'informations toujours plus grande ? Mais au-delà de la simple volumétrie des données, de nombreuses problématiques se font jour : sécurité, intégrité, optimisation des requêtes, amélioration des performances, optimisation du stockage, mapping objet / relationnel.

La notion de temps réel est illusoire

De nombreuses évolutions sont possibles et transforment déjà le monde des SGBD. Les performances et l'accès immédiat aux données constituent deux écueils importants. La notion de temps réel est illusoire, et ce à plusieurs titres : qu'appelle-t-on temps réel ? Quel temps de latence accepte-t-on ? Que faire des contraintes physiques du matériel ? N'oubliez pas que l'on ne peut pas théoriquement passer outre les Entrées / Sorties ou encore la lecture / écriture des données sur un disque dur. Nous disons bien théoriquement, car différentes techniques permettent de contourner partiellement ces contraintes : le cache ou encore la mémoire. Ce dernier cas de figure nous paraît particulièrement intéressant pour accélérer l'accès aux données. Ce n'est pas un hasard si la notion de

SGBDM ou base de données mémoire (ou en mémoire) refait surface. Aujourd'hui, les solutions hybrides SGBDR + fonctions en mémoire fleurissent. Courant 2008, IBM avait racheté SolidDB, un acteur reconnu des SGBDM.

Aujourd'hui, les données géospatiales tiennent une place de plus en plus grande, même si cela implique parfois une explosion de la volumétrie, obligeant à toujours mieux gérer les données de type Blob. La volumétrie pose d'ailleurs de réels problèmes aux développeurs et DBA : comment stocker une telle quantité de données sans exploser les tables et les bases ? A cela les éditeurs offrent deux réponses : la compression et le partitionnement. Nous verrons notamment dans notre dossier en quoi le partitionnement, sous MySQL, constitue une excellente réponse pour répartir les données d'une même base, d'une même table entre plusieurs serveurs... La compression est une autre fonction particulièrement choyée par les SGBD. Nous aborderons aussi l'éternel problème du mapping relationnel objet avec un véritable guide de survie du développeur sur Hibernate. Qui a dit que le mapping était un jeu d'enfant ?

■ François Tonic

Dans les prochains numéros nous aborderons : Hibernate, le futur des SGBD et comment utiliser Linq avec une base MySQL !



Les mécanismes de **sécurité** dans les bases de données

Le monde de la finance a toujours été un moteur pour faire évoluer la gestion et la sécurité des données. L'incendie de la salle du marché du Crédit Lyonnais en 1996 avait ainsi ouvert la voie à une révision complète des plans de secours et de reprise d'activité dans l'ensemble des services informatiques. Ce sont désormais les thèmes autour de la sécurité (accès aux données, surveillance des utilisateurs...), qui sont au cœur des préoccupations depuis les dernières affaires, notamment aux Etats Unis. Pour cela un certain nombre de directives ont vu le jour (Sarbanes Oxley, HIPAA, ...). La base de données est au cœur de cette problématique de par son rôle de stockage et de diffusion des informations critiques de l'entreprise.

Cet article a pour but de faire un état de l'art des fonctionnalités associées à la sécurité, disponibles sur les différents SGBD du marché (Oracle, SqlServer, DB2, Sybase et MySQL). Le schéma [Fig.1] symbolise les différents éléments de sécurité pouvant être mis en œuvre autour de la base de données.

Authentification

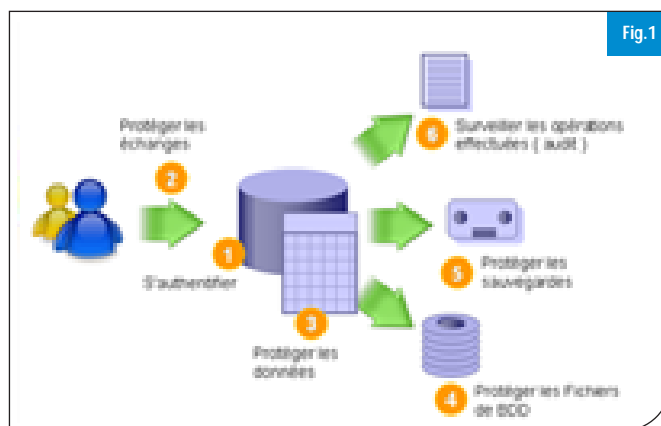
L'authentification est l'opération qui consiste à vérifier la légitimité d'une demande de connexion par un utilisateur. En général tous les SGBD permettent de déclarer les utilisateurs directement au niveau de la base à travers une notion de user, login et mot de passe. Pour certains SGBD, l'authentification se fera directement avec le compte système. Gérer les comptes utilisateurs devient complexe quand le nombre de bases et d'applications devient important. C'est pour cela que les éditeurs proposent des solutions permettant de s'interfacer à des sources extérieures de type annuaire LDAP (), Active Directory, etc. qui vont centraliser la gestion des utilisateurs. Cela permet d'en simplifier la gestion.

Le principe d'utilisation d'une source externe est sensiblement la même pour les éditeurs de SGBD, en général il faut installer un module spécifique qui très souvent demande une licence supplémentaire afin d'établir le dialogue avec l'annuaire.

Une gestion centralisée des utilisateurs permet également de mettre en place une solution d'identification unique ou SSO (Single Sign On) qui permet à un utilisateur une fois identifié d'utiliser différents services ou applications.

Le schéma [Fig.2] décrit un exemple d'architecture qui met en relation une base de données, un annuaire de type active directory et un serveur d'authentification de type Kerberos. Le schéma [Fig.2] décrit la démarche d'authentification dans ce type d'architecture.

- Les utilisateurs de la base de données sont assignés à l'Active Directory
- Le mapping des utilisateurs permet à un système d'exploitation d'accéder à une base en tant qu'utilisateur de l'Active Directory
- L'authentification va se faire avec Kerberos
- Avec Kerberos, un client Windows s'authentifie sur l'Active



Directory via le serveur Kerberos, le kdc (Microsoft Key distributor) puis il reçoit un ticket Kerberos.

- Si l'utilisateur veut se connecter au serveur de BDD, celui-ci vérifie auprès du serveur Kerberos si l'utilisateur est déjà connecté. Si c'est déjà le cas, le serveur de Base de données récupère le ticket Kerberos et l'utilisateur peut se connecter sans spécifier de mots de passe.

La difficulté de la mise en place d'une architecture SSO, réside plus sur la partie cliente que la partie serveur base de données. En effet pour qu'une application puisse fonctionner en SSO, il faut le prévoir lors du développement.

Protection des échanges

La protection des échanges consiste à sécuriser les informations échangées entre l'utilisateur et la base de données. Le principal protocole de transport pour l'échange de données entre un client et un SGBD sur système ouvert est TCP/IP. La plupart des solutions de cryptage sont transparentes pour les applications. SSL (Secure Socket Layers) et sa nouvelle dénomination TLS, est le protocole de sécurisation le plus connu, notamment pour les échanges sur internet. Cette solution qui s'appuie sur l'utilisation d'un certificat numérique peut être également mise en œuvre pour les échanges avec la base de données. Tous les éditeurs de

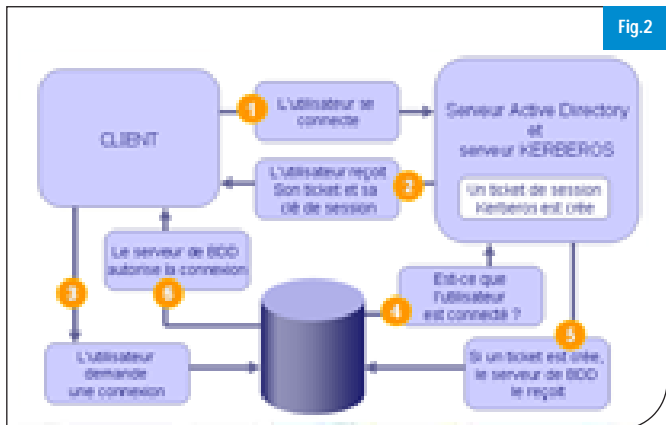


Fig.2

SGBD de notre étude, permettent la mise en œuvre directe ou à travers un module complémentaire d'une sécurisation SSL.

Protection des données

La sécurisation des données permet de crypter ou de limiter la vision des données pour un utilisateur au niveau d'une base, d'une table, d'une colonne. La norme SQL ANSI permet déjà à travers la gestion des droits (GRANT, REVOKE) d'autoriser un utilisateur à voir ou modifier des informations sur une table, ou sur certaines colonnes. Par contre les données sont stockées en clair, non cryptées. Dans le cas de données sensibles, il est important de proposer des solutions de cryptage des données. Pour répondre à ces problématiques, les éditeurs de SGBD proposent essentiellement deux solutions :

- La première consiste à utiliser des fonctions de cryptage directement dans le code SQL, avec suivant les algorithmes employés, l'utilisation d'une clé. L'emploi des fonctions de cryptage oblige à une modification du code applicatif. Les principaux algorithmes disponibles suivant les SGBD sont DES, 3DES, AES, MD5, MD4, SHA et SHA-1.
- La deuxième consiste à mettre en place un système de cryptage connu sous le nom générique de TDE (Transparent Data Encryption). Cette solution de cryptage est transparente pour les applications (il n'y a pas besoin de modifier le code applicatif). En général, cette solution permet de protéger les fichiers de la base ainsi que les sauvegardes.

SQL Server 2005 à travers ses fonctions de cryptage propose trois modes de fonctionnement basés sur des commandes SQL et des fonctions spécifiques. Pour mieux comprendre l'utilisation des fonctions de cryptage, nous allons détailler les différentes variantes disponibles :

- Utilisation d'un certificat :

```
CREATE CERTIFICATE cert_1 WITH ...
UPDATE ... SET colonne_cryptée = EncryptByCert(Cert_ID('cert_1'), 'texte à crypter')...
SELECT ...DecryptByCert(Cert_ID('cert_1'), colonne_cryptée) ...
FROM ...
```

- Utilisation d'une clé

```
CREATE ASYMMETRIC KEY key_1 WITH ALGORITHM=RSA_2048 ...
UPDATE ... SET colonne_cryptée = EncryptByAsymKey(AsymKey_ID
```

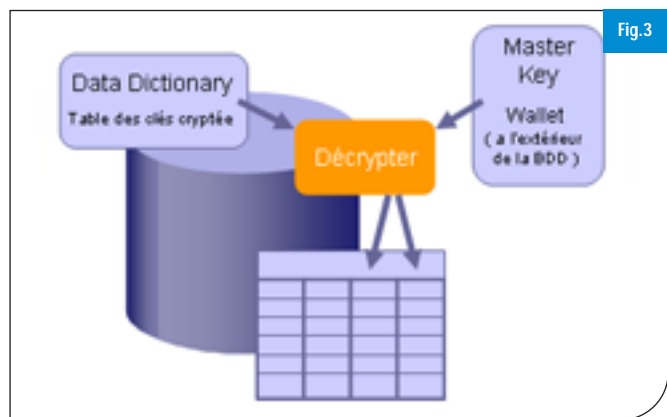


Fig.3

```
('key_1'), 'texte à crypter') ...
SELECT ... DecryptByAsymKey(AsymKey_ID('key_1'), colonne_cryptée)
... FROM ...
```

- Utilisation d'une clé et d'un certificat.

```
CREATE SYMMETRIC KEY key_2 WITH ALGORITHM AES_256
ENCRYPTION BY CERTIFICATE cert_1

OPEN SYMMETRIC KEY key_2 DECRYPTION BY CERTIFICATE cert_1
UPDATE ... SET colonne_cryptée = EncryptByKey(Key_GUID('key_2'),'texte à crypter');

SELECT ... DecryptByKey(colonne_cryptée) ... FROM ...

CLOSE SYMMETRIC KEY key_2
```

En ce qui concerne TDE (Transparent Data Encryptions), actuellement seul Oracle, SQL Server et Sybase proposent ce mode de cryptage. Sous Oracle la fonctionnalité **TDE** qui fait partie de l'option **Oracle Advanced Security**, permet d'effectuer un cryptage au niveau de la colonne ou du *tablespace*. [Fig.3]

Pour cela, Oracle utilise une clé externe (master key) qui peut être stockée dans un Oracle wallet (protégé par mot de passe). Si un utilisateur accède à la base, les données seront décryptées automatiquement.

```
// Création du répertoire devant contenir le 'Wallet'

ALTER SYSTEM SET ENCRYPTION KEY AUTHENTICATED BY "MotDePasse";

CREATE TABLE test (
    id NUMBER,
    num_cb VARCHAR2(12) ENCRYPT
);

// Au démarrage et à l'arrêt de l'instance on ouvre et on ferme l'accès au 'Wallet'. La fermeture du Wallet interdit la lecture des données cryptées.

ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY "MotDePasse";
```



```
ALTER SYSTEM SET WALLET CLOSE;
```

Sous Sybase la fonction TDE se fera au niveau de la table ou de la colonne, l'utilisateur n'aura accès aux données que si le DBO ou le SSO lui a attribué un droit spécifique.

```
CREATE ENCRYPTION KEY key_1 AS DEFAULT FOR AES

ALTER TABLE achat MODIFY num_cb NULL ENCRYPT WITH key_1

GRANT DECRYPT ON achat TO user2
```

La fonction TDE disponible sous SQL Server 2008 permet de crypter une base complète. Les données seront décryptées automatiquement aux utilisateurs qui se connectent à la base, il n'y a pas besoin de droits spécifiques.

```
USE master
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MotDePasse'
go
CREATE CERTIFICATE Cert_1 WITH SUBJECT = 'Mon certificat'
go
USE MaBase
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE Cert_1
GO
ALTER DATABASE MaBase
SET ENCRYPTION ON
GO
```

Protection des fichiers

La protection des fichiers de la base de données, évite qu'une copie de fichier puisse être utilisée par un autre serveur de base de données. Cette solution consiste souvent en un cryptage complet à travers une solution de cryptage de type TDE. Certains éditeurs n'ayant pas de solutions réelles, proposent d'utiliser le cryptage assuré par le système d'exploitation, au niveau des File System. Cette solution est malgré tout souvent déconseillée pour des problèmes de performances.

Sécurisation des fichiers de sauvegardes

La sécurisation des fichiers de sauvegardes évite de pouvoir recharger facilement ces fichiers sur une nouvelle instance. En général une protection par mot de passe est proposée ou si une solution de cryptage complet de la base a été mise en place (TDE), la sauvegarde sera elle-même cryptée.

Audit

La surveillance de l'utilisation de la base consiste à consigner toutes les opérations qui sont effectuées sur la base. Des directives de type Sarbanes Oxley conseillent de consigner les opérations effectuées au niveau des systèmes d'information et notamment au niveau des bases de données. Des mécanismes d'audit sont implémentés dans tous les SGBD afin de tracer l'origine, l'heure et le but de chaque action, permettant ainsi de s'assu-

rer de la responsabilité des actions utilisateurs. En général, un paramétrage permet de filtrer les informations recherchées avant de les stocker dans une table. Suivant les solutions implémentées et le paramétrage choisi, l'impact sur les performances sera plus ou moins important.

- Oracle audite l'activité de la base de données par ordre SQL, par privilège système de l'utilisateur, par objet, ou par utilisateur. L'enregistrement de l'audit peut être stocké dans une table de la base de données. Pour les responsables DBA, Oracle peut auditer toutes les opérations SYS sur le système d'opération, afin de réaliser une revue complète de la sécurité administrateur et son audit. Il est également possible d'utiliser un audit plus fin avec l'option " **Fine-Grained Auditing** ", ce qui permet de générer un événement d'audit lorsqu'une table est par exemple accédée durant une tranche horaire ou sur une colonne spécifique.

Avec **Audit Vault**, Oracle automatise la collection d'informations d'audit et le processus d'analyse des informations consolidées. En plus des bases Oracle, cet outil supporte Microsoft SQL Server 2000 and 2005. IBM DB2 UDB 8.2 ,9.5, Sybase ASE 12.5 and 15.2 .

- SQL Server 2005 supporte trois types d'audit, le premier est basé sur la connexion, le deuxième, sur des triggers de modification de structure (DDL triggers), et le dernier sur des événements. L'audit basé sur les événements a un fonctionnement complètement asynchrone (utilisation du bus de messages XML Service Broker), ce qui lui permet d'avoir un impact très faible sur les traitements, contrairement à l'audit basé sur les triggers. La version 2008 apporte des mécanismes supplémentaires à l'audit, notamment une historisation dans de nouvelles tables systèmes (Change Data Capture, Change Tracking).
- Sybase propose un système d'audit performant, qui permet de consigner les opérations effectuées sur la base suivant le paramétrage qui a été mis en œuvre. Les informations sont stockées dans une série de tables (SYSAUDITS_01, SYSAUDITS_02 ...), ce qui évite la perte d'information d'auditing et facilite l'archivage.
- DB2 fournit un service d'**audit** qui permet d'enregistrer les opérations réalisées par les applications, les accès utilisateurs individuels et les administrateurs de base de données. L'administrateur de la base utilise l'outil db2audit pour configurer l'audit au niveau d'une instance ou au niveau de chacune des bases de l'instance. Cet outil permet également d'archiver les fichiers d'audit ou bien d'extraire des informations des fichiers archivés.
- Sous MySQL, il n'y a pas à ce jour de solution réellement aboutie. Le log général permet de tracer l'activité des utilisateurs, dans un fichier ou dans une table depuis la version 5.1, mais au détriment des performances. L'outil MySQL Proxy, pour l'instant en version Alpha, propose de se glisser entre le serveur et les clients afin d'intervenir à chaque niveau de communication. Il est alors possible, entre autres, d'auditer tout ce qui passe entre le serveur et les clients.

Limiter les droits des administrateurs (DBA)

On considère qu'actuellement les administrateurs de base de données ont trop de pouvoir sur la base de données. Les éditeurs de base de données ont développé de nouvelles fonctionnalités qui permettent de réduire la vision des DBA.

Sybase ASE avec sa version 15, apporte cette nouvelle fonctionnalité, à travers la notion de 'key custodian' qui permet à des utili-



sateurs de privatiser et de partager des données sans que l'administrateur puisse y accéder. Pour cela, un système de mot de passe et de déclaration permet d'échanger des clés entre divers individus.

```
// user1
CREATE ENCRYPTION KEY key_1 WITH PASSWD 'MotDePasse'
GRANT SELECT ON KEY key_1 TO dbo

// DBO
CREATE TABLE test (
  id NUMBER,
  num_cb VARCHAR2(12) ENCRYPT WITH user1.key_1 )

GRANT SELECT ON test TO user2

// user1
GRANT DESCRIPT ON test (num_cb) TO user2
ALTER ENCRYPTION KEY key_1 WITH PASSWD 'MotDePasse'
ADD ENCRYPTION WITH PASSWD 'user2_ps' FOR USER user2

//user2
SET ENCRYPTION PASSWD 'user2_ps' FOR KEY user1.key_1
SELECT * FROM test ...
```

Oracle avec son module Oracle Database Vault permet également de restreindre les privilèges des administrateurs (DBA) et de vérifier la conformité à certaines réglementations comme SOX (Sarbanes Oaxley)

Informations complémentaires

Oracle, avec sa base de données Privée Virtuelle (**Virtual Private Database : VPD**) permet, au sein même de la base de données, d'avoir l'assurance de la séparation des données pour les accès aux données par utilisateur ou par groupe d'utilisateurs.

En ajoutant dynamiquement le prédicat aux ordres SQL (derrière la clause WHERE), la VPD limite les accès aux données au niveau ligne, et elle attache la politique de sécurité à la table (à la vue, ou au synonyme) elle-même. D'autres options et outils complémentaires sont également disponibles dans l'offre Oracle :

- **Oracle Database Vault** : Permet de restreindre les privilèges des super utilisateurs (dba..) et de vérifier la conformité à certaines réglementations ex SOX (Sarbane Oaxley)
- **Oracle Label Security** : Permet de mettre un niveau de sécurité à chaque ligne d'une table.
- **Data masking** : Permet de masquer les données de production (duplication de base, export)

La plupart de ces modules d'Oracle supposent d'être en version entreprise et demandent une licence supplémentaire :

La plupart des fonctionnalités de sécurités associées à Sybase ASE sont présentes dans 2 modules supplémentaires :

- 'Security & Dir services' : permet de gérer l'authentification externe (Kerberos, PAM), ainsi que le cryptage SSL des connexions.
- 'Sybase ASE Encryption' : permet de gérer les différentes solutions de cryptage des données.

■ Yves **MOULIN**

Directeur Technique - Cap DATA Consulting