Chapitre 3
Méthodes - Bien gérer les données

1. Introduction

Pour stocker puis accéder aux données (« sérialiser » les données), on dispose d'un grand choix de moteurs de bases de données que l'on utilise lorsque l'on juge utile de ne pas se cantonner aux fichiers « plats » comme le sont les fichiers .csv. Un certain nombre de bases respectent des contraintes très strictes pour assurer la cohérence des données et s'avèrent accessibles via divers dialectes, dont le plus connu est SQL (*Structured Query Language*), d'autres s'affranchissent de certaines de ces contraintes pour gagner en performance, en capacité à intégrer des informations peu ou pas structurées (textes, images, vidéos, musiques, etc.).

Vous pouvez facilement vous familiariser avec la manipulation de diverses bases en installant localement un outil générique d'accès à ces bases comme **DBeaver**.

Remarque

*Sous Windows, qui ne dispose pas naturellement d'une gestion de packages aussi efficace que Linux, vous pouvez installer de nombreux logiciels de façon simple et **réversible** en utilisant l'outil **chocolatey**. Avec cet outil, l'installation de DBeaver s'effectue en une seule ligne de commande via le terminal Windows :* `choco install dbeaver`.

Voyons maintenant les grands concepts permettant de qualifier les structures intimes de vos bases de données.

2. Les concepts CRUD, 3NF, ACID, OLTP (et pas OLAP !)

2.1 CRUD

CRUD représente les quatre opérations de base de gestion de persistance des données sur une base de données :

– **C**reate : créer ;
– **R**ead : lire ;
– **U**pdate : mettre à jour ;
– **D**elete : supprimer.

Remarque

Dans les architectures décisionnelles (BI), on évite les opérations Update et Delete. On préfère dupliquer les enregistrements en leur apposant des index d'horodatage, de la forme date/heure/min/seconde (un timestamp typiquement exprimé dans le format de date du standard POSIX) stipulant leur date de création et le cas échéant ajouter un enregistrement spécifique avec son timestamp pour signifier que les enregistrements de mêmes index ne sont plus valables. Ainsi on conserve toute la traçabilité des évolutions du système d'information et l'on peut unir sans danger plusieurs tables de même nature extraites à des dates différentes à partir des systèmes transactionnels, ceci moyennant l'élimination des doublons. La base HBase d'Hadoop est très efficace dans ce type de contexte de mise à jour incrémentale du data lake.

2.2 Les formes normales, en se limitant au niveau 3NF

2.2.1 Relations (associations)

Avant d'explorer le concept de forme normale, remarquons qu'une table représente toujours une relation entre les attributs décrits dans ses colonnes.

Dans une relation (une association) bien construite, on dispose du nom de la relation, représenté par le nom de la table en représentation relationnelle, du déterminant de la relation (une liste cohérente d'attributs qui identifient le sujet de la relation) et du déterminé de la relation (une liste cohérente d'attributs qui identifie l'objet de la relation).

On retrouve des concepts similaires dans d'autres contextes de représentation et stockage de l'information :

– Les relations/associations se rapportent à la notion de prédicat dans le Web sémantique et dans la représentation de la connaissance (*knowledge*) souvent stockée dans des triplets **RDF** (*Resource Description Framework*).
– Les relations correspondent également à des liens (arêtes) entre nœuds (sommets) dans les représentations sous forme de graphe, parfois sauvées en bases graphes comme Neo4j.

Pour préciser notre propos, observons la relation sémantique `utilise` suivante : (`individu`, `utilise`, `poste_de_travail`) :

– « utilise » est la relation (association/prédicat) ;
– « individu » est le déterminant (le sujet) ;
– « poste_de_travail » est le déterminé (l'objet de la relation).

Si l'on pense en termes d'algèbre relationnelle, on l'écrit plutôt de la façon suivante : `utilise(individu, poste_de_travail)`.

Imaginons que les individus soient complètement caractérisés par les deux attributs qui sont nom et prénom et que les postes_de_travail soient caractérisés par un étage, un numéro de pièce et un numéro d'ordre dans la pièce, on obtient la relation : *utilise(nom,prénom, étage, pièce, numéro_ordre)*. C'est une relation utile pour l'analyse de données, mais trop riche et qui ne reflète pas tout ce que nous savons.

On comprend aisément qu'il était important de ne pas oublier la sémantique des relations sous-jacentes que sont *individu(nom,prénom)* et *poste_de_travail(étage, pièce, numéro_ordre)*. C'est ce genre de dilemme que nous avons à résoudre quand nous structurons nos données : conserver le sens profond des choses ou privilégier la restitution des informations.

Une instance d'une telle relation pourrait se concrétiser dans **un enregistrement** (les data scientists diraient **une observation**) ayant la forme de la **ligne** suivante dans une table nommée « utilise » : *(Eva, Laude, 3, 21, 4)*. Ce qui signifie : *L'individu Eva Laude utilise le poste de travail de l'étage 3, pièce 21, emplacement 4*.

2.2.2 Les trois formes importantes : 1FN, 2FN, 3FN

La notion de « forme normale » vous donne des critères pour identifier si la représentation de telles relations est de qualité. On dénombre huit formes normales, mais habituellement on ne s'attache qu'à être conforme aux trois premières :

– 1FN, première forme normale, atomicité : les attributs doivent être sémantiquement atomiques, c'est-à-dire que l'on ne peut pas découper un tel attribut en autres attributs, par exemple dans notre cas une codification comme 3-21-4 portée par une seule colonne de la table n'aurait pas été conforme si l'on imagine d'utiliser un jour l'information comme quoi Eva Laude est au troisième étage.

– 2FN, deuxième forme normale, non-redondance déterminant/déterminé : il faut qu'elle soit 1FN et qu'aucun des attributs déterminés ne puisse se déduire d'un attribut partiel du déterminant. Par exemple, si le troisième étage est l'étage appartenant à la famille Laude, il y aurait redondance entre l'attribut étage et l'attribut nom. C'est une situation très dangereuse car cela pourrait créer des incohérences difficiles à gérer, typiquement lors de l'évolution ou la destruction des déterminants et/ou des déterminés.

– 3FN, troisième forme normale, non-redondance au sein du déterminé : il faut qu'elle soit en en 2FN et que l'on ne puisse pas déduire un des attributs du déterminé d'un ou plusieurs autres attributs du déterminé. Dans notre exemple, si le numéro d'emplacement 4 implique que l'on est au troisième étage alors nous ne sommes pas en 3FN.

Remarque

La définition des index et/ou des clés primaires et secondaires dans un schéma relationnel peut parfois laisser penser, à tort ou pas, que l'on ne respecte pas les formes normales quand on remarque la présence d'identifiants qui semblent redondants. Si l'on veut éviter le moindre doute, il faut créer physiquement les tables correspondant aux déterminants et aux déterminés et ne reporter que leurs clés dans la relation.

2.3 Les propriétés ACID

Les propriétés ACID sont les caractéristiques d'une transaction fiable effectuée sur une base de données :

– **Atomicité** : tous les changements de la transaction sont effectués ou aucun. Autrement dit, on revient à l'état initial (*rollback*) si l'on n'a pas atteint la fin de transaction en effectuant son commit, y compris en cas de panne matérielle.
– **Cohérence** : chaque transaction amène la base de données d'un état cohérent à un autre état cohérent en termes de contraintes d'intégrité fonctionnelle (CIF), de gestion de cascades d'opérations *rollback* ou de toute contrainte de cohérence exprimée pendant la conception.
– **Isolation** : chaque transaction est indépendante et ne dépend donc pas de la réalisation d'une autre.
– **Durabilité** : les transactions ayant fait l'objet d'un commit sont enregistrées, même en cas de panne matérielle.

Les transactions basiques sur des bases de données relationnelles SQL, dont les schémas sont bien pensés, sont naturellement ACID (c'est-à-dire : écrire, mettre à jour ou supprimer un enregistrement dans une table en relation avec d'autres tables).

Dans la plupart des cas, il n'y a aucun sens à penser « ACID » pour les actions Read (cRud).

On remarque que les développeurs négligent parfois de gérer les propriétés ACID pour les actions Update et Delete (crUD), ce qui a des conséquences néfastes.

En tout état de cause, il est de bonne pratique de concevoir des tests unitaires de qualité et automatisés pour toutes les actions CRUD et les propriétés ACID. En effet, une évolution du schéma de base de données (ou d'un UX, *User eXperience* = parcours utilisateur) peut avoir des conséquences importantes sur ces aspects et il faut donc assurer l'automatisation de ces tests pour chaque évolution de la portée des transactions ou des schémas de base de données.

2.4 L'OLTP

Les logiciels de classe OLTP (*Online Transaction Processing*) savent gérer les transactions ACID sur Internet (ou sur un intranet), y compris quand la transaction concerne plusieurs serveurs de base de données et transite par plusieurs serveurs HTTP. Évidemment, un système redondant, créé pour éviter les *Single Point of Failure* et augmenter la rapidité des transactions est sujet à des problèmes concernant les critères ACID. C'est donc là que se trouve le challenge des logiciels de classe OLTP.

Dans la plupart des cas, les caractéristiques OLTP sont portées par des améliorations posées sur des moteurs de base de données très aboutis et le plus souvent relationnels (SQL Server de Microsoft dans sa version où toute la base est en mémoire vive (*in memory*), Autonomous Transaction Processing Database d'Oracle, etc.) ou sont portées par des serveurs d'application comme CICS Transaction server d'IBM, qui est une évolution de l'ancêtre de l'OLTP phare des années 2000 : « le bon vieux CICS ».

On implémente souvent des composants OLTP pour garantir la conformité ACID des :

– sites de commerce électronique haut de gamme ;
– transactions financières ;
– CRM (*Customer Relationship Management*) comme Salesforce ou Hubspot ;
– ERP (*Enterprise Ressource Planning*, progiciel de gestion intégré) comme SAP et Odoo ;
– MRP (*Manufacturing/Material Resources Planning*) : un MRP gère la production et la logistique de l'entreprise, et est appelé logiciel de GPAO, en français, pour « Gestion de la production assistée par ordinateur ».

Les performances d'un système OLTP sont mesurées par le TPC (*Transaction Processing Performance Council*) en transactions par minute (tpm). TPC-C est le test standard le plus répandu ; on mesure couramment des tpmC atteignant 100 000 tpm dans de nombreuses configurations OLTP (et de plusieurs centaines de millions sur certaines architectures extrêmement rares et coûteuses !).

3. Les concepts liés aux databases de la BI et du BigData

La BI, pour *Business Intelligence*, est en fait un terme marketing qui recouvre les techniques de *reporting*, interactives ou pas. On la dénomme parfois *analytics* ou « décisionnel » pour mettre en évidence le fait que de bonnes statistiques et de bons rapports président à de bonnes décisions.

Les entrepôts de données (***data warehouse***) étaient au centre des architectures BI. Depuis que les volumes de données ont explosé, il a fallu adjoindre à ces entrepôts de données structurés et centralisés des *clusters* de données massivement parallèles sur de nombreux *nodes*. Quand ces données sont collectées sans a priori sur leur usage futur, on est souvent amené à les stocker sous une forme un peu générique que l'on nomme *datalake*.

Sur ces entrefaites, l'arrivée du *Machine Learning*, une branche de l'intelligence artificielle qui permet d'inférer (prévoir) sur des données massives ou de trouver des similitudes entre ces données (*clustering*), a complexifié les architectures.

Cette accélération continue aujourd'hui, puisque les modèles actuels évoluent rapidement en puissance et en consommation de ressources, en témoigne l'essor des réseaux neuronaux profonds (*deep learning*).

Enfin l'IA générative, comme ChatGPT, complète aujourd'hui le paysage, sans pour autant que celle-ci ait une très grande influence sur l'architecture technique sous-jacente à nos systèmes d'information (à ce jour !).

Quand les données sont issues de systèmes relationnels, on aime les agréger sous une forme puissante et centralisée s'appuyant sur des *data warehouses* bien nettoyés et structurés qui ne concernent que des données comportant une grande part de données numériques et qui porte le doux nom de « hypercube OLAP ».

3.1 OLAP

Il ne faut pas confondre OLTP et OLAP. Cette confusion est entretenue par le fait que certaines architectures de stockage très modernes gèrent aussi bien l'un que l'autre (comme les machines Exadata d'Oracle). Pour mémoire, l'OLAP (pour *Online Analytic Processing*) est un type de présentation des données sous forme d'hypercube, très utilisé dans l'informatique décisionnelle (*Business Intelligence*).

Chaque cellule de l'hypercube comporte une donnée élémentaire indexée par toutes les dimensions de l'hypercube, un peu comme dans un `array` du package Numpy très utilisé en Python ou un `array` du langage R ou comme dans les tenseurs de Tensorflow ou Pytorch qui permettent de gérer d'immenses tableaux multidimensionnels.

Remarque

Nous vous invitons d'ailleurs à lire nos autres ouvrages aux Éditions ENI pour comprendre l'usage et la puissance de ces outils (array, tenseur, Tensorflow, etc.), qui sont aujourd'hui les structures manipulées au quotidien dans l'IA, le traitement d'image et les calculs industriels.

L'interface homme-machine d'un OLAP ressemble à celle d'un tableur, mais les données sont accessibles dans toutes les dimensions et à différents niveaux de profondeur. La grille de donnée manipulable par l'utilisateur change instantanément en fonction de ses volontés (exemples : ventes par région, marge par produit sur une région donnée, répartition des ventes par type de produit dans un secteur commercial donné, etc.).

On peut agréger ces données sur plusieurs axes et analyser les résultats de ces agrégations via :

– le ***slicing*** : une requête proposant un résultat sur deux dimensions et agrégée sur des intervalles de valeurs filtrant les autres dimensions (ex. : tableau des ventes *pays par années* sur les produits de tel type et de telle région) ;

– le ***drill down*** : « creuser » une cellule pour comprendre de quoi est constituée la valeur d' un agrégat de données (interactivement : on clique sur une cellule et cela ouvre un écran détaillant son contenu) ;