Systèmes de gestion de bases de données

BARBIER J.M. - LGT Dumezil - 1NSI

26 janvier 2026

1 Introduction

1.1 Programme

Notions : Système de gestion de bases de données relationnelles.

Capacités attendues : Identifier les services rendus par un système de gestion de bases de données relationnelles : persistence des données, gestion des accès concurrents, efficacité de traitement des requètes, sécurisation des accès.

Remarques : Il s'agit de comprendre le rôle et les enjeux des différents services sans en détailler le fonctionnement.

2 Qu'est-ce qu'un SGBD ?

2.1 Définition

Un **Système de Gestion de Bases de Données** (SGBD) est un logiciel qui permet de :

- créer et manipuler des bases de données
- interroger et modifier les données
- garantir l'intégrité et la cohérence des données
- gérer les accès multiples et simultanés

Un **SGBDR** (Système de Gestion de Bases de Données Relationnelles) est un SGBD qui implémente le modèle relationnel.

2.2 Exemples de SGBDR

Quelques SGBDR largement utilisés :

- **PostgreSQL** : open source, très complet et performant
- **SQLite** : léger, embarqué, sans serveur
- **MariaDB** : open source, très populaire sur le web, "fork" de **MySQL** qui gère l'open source à la sauce Oracle

- **Oracle Database** : propriétaire, utilisé en entreprise, payer cher rassure les DSI
- **Microsoft SQL Server** : propriétaire, écosystème Microsoft, idem ci-dessus

Chacun a ses spécificités, mais tous respectent les principes du modèle relationnel et offrent des services similaires.

2.3 Exemples de SDGB non relationnels

Non-relationnel = on libère une partie des contraintes du modèle relationnel pour gagner en performance ou en flexibilité.

Quelques SGBD non relationnels (NoSQL) :

- **MongoDB** : base de données orientée documents
- **Cassandra** : base de données distribuée pour les grandes volumétries
- **Redis** : base de données en mémoire, clé-valeur

3 Les services rendus par un SGBDR

3.1 Vue d'ensemble

Un SGBDR rend quatre services essentiels :

1. **Persistance des données**
2. **Gestion des accès concurrents**
3. **Efficacité de traitement des requêtes**
4. **Sécurisation des accès**

4 Persistance des données

4.1 Persistance

Le SGBDR assure la **persistance** des données en les stockant sur un support durable (disque dur, SSD).

Garanties offertes :

- les données survivent à l'arrêt du programme ou du système
- les modifications validées ne sont jamais perdues
- en cas de panne, le système peut récupérer un état cohérent

4.2 Transactions et atomicité

Le SGBDR garantit que les modifications puissent être effectuées de manière **atomique** : soit elles réussissent toutes, soit aucune n'est appliquée.

Exemple : transfert d'argent entre deux comptes bancaires

- débiter le compte A de 100€
- créditer le compte B de 100€

Si une panne survient entre les deux opérations, on pourrait perdre 100€ ! Le SGBDR garantit que soit les deux opérations sont effectuées, soit aucune ne l'est.

Cela se fait via le concept de **transaction** :

```
DÉBUTER_TRANSACTION
  débiter(compte_A, 100)
  créditer(compte_B, 100)
VALIDER_TRANSACTION
```

Si une erreur survient, on peut annuler toutes les modifications en **annulant la transaction**.

5 Gestion des accès concurrents

5.1 Le problème

Dans un système réel, plusieurs utilisateurs ou processus accèdent **simultanément** à la base de données.

Exemple : bibliothèque avec un système de réservation en ligne

- Alice veut réserver le dernier exemplaire d'un livre
- Bob veut réserver le même livre en même temps
- Sans gestion appropriée, le système pourrait permettre deux réservations du même exemplaire !

Autre exemple : compteur de vues sur un site web

```
1. Lire le compteur : 1000
2. Incrémenter : 1001
3. Écrire la nouvelle valeur : 1001
```

Si deux utilisateurs font cela simultanément, on risque de perdre un incrément [1].

1. Les deux lisent 1000, les deux écrivent 1001, total 1001 au lieu de 1002 : une vue est "perdue"

5.2 Solution : contrôle de concurrence

Le SGBDR implémente des mécanismes de **contrôle de concurrence** qui permettent de gérer les accès simultanés aux données.

Principaux mécanismes :

- **Verrouillage** : bloquer temporairement l'accès à certaines données pour éviter les conflits
- **Transactions** : regrouper plusieurs opérations en une seule unité atomique

5.3 Responsabilité du développeur

Attention : le SGBDR ne peut garantir la cohérence que si le développeur conçoit correctement son application !

Exemple du compteur de vues :

```
# ⊠ INCORRECT : le SGBDR ne peut rien faire
compteur = lire_compteur()      # 1. Lire : 1000
compteur = compteur + 1         # 2. Incrémenter en Python
écrire_compteur(compteur)       # 3. Écrire : 1001
# Si deux clients font ça en même temps, une vue est perdue !

# ✓ CORRECT : opération atomique dans la base
incrémenter_compteur_dans_la_base()
# Le SGBDR garantit que les deux incréments sont bien pris en compte
```

Le développeur doit utiliser les mécanismes du SGBDR (transactions, opérations atomiques) pour que celui-ci puisse assurer la cohérence.

6 Efficacité de traitement des requêtes

6.1 Le problème

Une base de données peut contenir des **millions** voire des **milliards** de n-uplets.

Sans optimisation, rechercher une information pourrait nécessiter de parcourir **toute la base** : inacceptable pour des applications réactives.

Exemple : rechercher un livre par son ISBN dans une base de 100 000 livres. Sans optimisation, il faudrait examiner jusqu'à 100 000 livres !

6.2 Solution : optimisation des requêtes

Le SGBDR implémente de nombreuses techniques pour accélérer les requêtes :

- **Index** : structures de données permettant de retrouver rapidement un n-uplet
- **Optimiseur de requêtes** : choisit automatiquement la meilleure stratégie d'exécution
- **Cache** : garde en mémoire les données fréquemment consultées
- **Parallélisation** : exploite plusieurs cœurs de processeur

Avec un index sur l'ISBN, la recherche d'un livre se fait en temps logarithmique $\log_2(100\,000) \approx 17$ comparaisons au lieu de 100 000 !

Le développeur peut créer des index sur les attributs fréquemment recherchés :

```
créer_index(relation: Livres, attribut: isbn)
```

L'optimiseur de requêtes analyse chaque requête et choisit automatiquement la meilleure stratégie parmi plusieurs possibles.

6.3 Statistiques et analyse

Le SGBDR maintient des **statistiques** sur les données :

- nombre de n-uplets dans chaque relation
- distribution des valeurs pour chaque attribut
- taille moyenne des données

Ces statistiques permettent à l'optimiseur de prendre les bonnes décisions, comme choisir entre plusieurs index possibles ou décider de l'ordre des jointures.

7 Sécurisation des accès

7.1 Le problème

Une base de données contient souvent des informations sensibles :

- données personnelles (RGPD)
- informations financières
- secrets industriels

Il faut pouvoir contrôler **qui** peut accéder à **quelles** données et faire **quelles** opérations. Exemple : un service qui calcule des statistiques de visites sur un site n'a pas besoin d'accéder aux informations de paiement des clients !

7.2 Solution : gestion des droits

Le SGBDR implémente un système de **gestion des droits d'accès** :

- **Authentification** : vérifier l'identité de l'utilisateur
- **Autorisation** : définir ce que chaque utilisateur peut faire
- **Audit** : tracer les accès et modifications pour détecter les intrusions

Les droits peuvent être définis de manière très fine :

- par utilisateur ou par groupe
- par relation, par attribut, voire par n-uplet
- pour chaque type d'opération (lecture, écriture, suppression, etc.)

8 Exercices

8.1 Rappel

Service	Rôle
Efficacité	Optimiser les temps de réponse
Persistance	Conserver les données durablement
Sécurisation	Contrôler les accès
Accès concurrents	Gérer les accès simultanés

8.2 Exercice 1

Pour chacune des situations suivantes, identifier quel(s) service(s) du SGBDR est (sont) principalement sollicité(s) :

1. Un utilisateur consulte son historique d'achats sur un site e-commerce
2. 1000 personnes réservent simultanément des places pour un concert
3. Une application mobile fonctionne hors ligne puis synchronise ses données
4. Un programme génère des statistiques de fréquentation d'un site web à partir de 40 millions d'enregistrements de visites

8.3 Exercice 2

Expliquer pourquoi chacun des services suivants est indispensable pour une application bancaire :

- persistance des données
- gestion des accès concurrents
- efficacité de traitement
- sécurisation des accès

Donner un exemple de problème qui pourrait survenir si l'un de ces services était défaillant.