TD2 SGBD :
Modèle de données

Marc Robison
Nils Peteil

5 mars 2025

Table des matières

1 Partie 1

1.1 Ex1 : Comparaison entre fichiers classiques et bases de données

1.1.1 Dans un système de gestion basé sur des fichiers classiques, quelles sont les principales insuffisances que vous pouvez identifier ? Justifiez votre réponse en donnant des exemples.

Les informations sont réparties sur plusieurs fichiers. Une information peut-être présente sur plusieurs fichiers. La modification de cette information implique une modification sur chaqu'un des fichiers, non seulement un seul.

Il y a donc de la redondance d'information.

Pour modifier les données on peut utiliser un programme mais le programme dépend de la structure des fichiers (de la nature de l'application).

— Données dupliquées (entre différents services, difficiles à partager, ...)
— Conflits de données (versions multiples, données non mises à jour, ...)
— Données difficiles à mettre à jour (difficile de répondre à des changements, copies à faire de toutes les données pertinentes, ...)

1.1.2 Quels sont les avantages et inconvénients d'une base de données relationnelle par rapport à un stockage sous forme de fichiers ?

Dans l'idéal il n'existe qu'un exemplaire de chaque élément de données.

Une base de données permet le partage aisé des informations et peut éliminer le problème de duplication et de maintien des ressources car toutes les données sont stockées au même endroit.

1.1.3 Complétez le tableau suivant en comparant fichiers classiques et bases de données :

Critères	Fichiers classiques	Base de Données
Redondance des données	Oui, pour lier les données	Non, pour économiser de l'espace versionnement si besoin.
Cohérence et intégrité des données	Non car on peut rentrer tout type de données dans un fichier	Oui car données typées.
Accès et manipulation des données	Accès concurentiel dépend du système d'exploitation	Accès concurentiel possible
Sécurité et contrôle d'accès	Pas de hiérarchie pour le contrôle d'accès.	Présence de hiérarchie de priorités.
Évolution et maintenance	Difficile à maintenir, pas de versionnement par défaut.	Structure plus modulable

1.1.4 Recherchez un cas réel d'entreprise ou d'organisation ayant abandonné un système basé sur des fichiers classiques au profit d'un SGBD. Expliquez les raisons de ce choix et les bénéfices obtenus.

Un cas réel d'entreprise ayant abandonné un système basé sur des fichiers classiques au profit d'un SGBD est la Société Générale, une grande banque française.

Contexte et Problème :

Avant les années 1980, comme de nombreuses banques, la Société Générale utilisait un système de gestion des comptes basé sur des fichiers plats stockés sur des bandes magnétiques et disques durs. Ce système avait plusieurs limites :

Redondance et incohérence des données :

Les mêmes informations étaient stockées dans plusieurs fichiers, entraînant des risques d'incohérence.

Difficulté d'accès aux données :

Chaque requête nécessitait une recherche séquentielle, ce qui ralentissait les opérations. Manque de flexibilité : Ajouter de nouvelles fonctionnalités ou modifier les formats de fichiers était complexe et risqué. Sécurité limitée : Le contrôle d'accès aux fichiers était rudimentaire, ce qui posait des problèmes de protection des données clients.

Migration vers un SGBD

Face à ces limites, la Société Générale a décidé dans les années 1980 de migrer vers un SGBD relationnel, notamment en adoptant IBM Db2, un des premiers systèmes de gestion de bases de données relationnelles.

Raisons du choix :

- Intégrité et cohérence des données : Le SGBD permettait d'éliminer les redondances et d'assurer l'intégrité via des contraintes (clés primaires, étrangères, etc.).
- Accès rapide et optimisé : Avec des index et des requêtes SQL, l'accès aux informations était beaucoup plus rapide que la lecture séquentielle des fichiers plats.
- Facilité de mise à jour : Modifier ou ajouter des données ne nécessitait plus de réécrire entièrement des fichiers.
- Sécurité renforcée : Un contrôle d'accès avancé permettait d'attribuer des permissions spécifiques aux employés.
- Évolutivité : La banque pouvait intégrer de nouveaux services (cartes bancaires, gestion des prêts, etc.) beaucoup plus facilement.

Bénéfices obtenus :

- Réduction des erreurs dans les transactions bancaires grâce à la centralisation des données.
- Gain de temps significatif pour les opérations bancaires et le traitement des données clients.
- Meilleure expérience client, avec des réponses plus rapides aux demandes et une disponibilité accrue des services.
- Capacité d'analyse accrue, permettant à la banque d'exploiter les données pour la gestion des risques et le marketing.

Ce cas illustre bien comment le passage d'un système de fichiers classique à un SGBD peut transformer l'efficacité d'une organisation, en améliorant la fiabilité des données et en facilitant la gestion et l'évolution des systèmes.

1.2 Ex2 : Définitions et concepts de base

1.2.1 Donnez une définition claire et concise des termes suivants :

- **Base de données :** On définit une base de données comme un ensemble de données structuré, enregistrées sur les support accessibles par l'ordinateur, pour satisfaire simultanément plusieurs utilisateurs de façon sélective et en temps opportun.
- **Système de gestion de bases de données (SGBD) :** On définit un système de gestion de bases de données comme un logiciel qui permet à un utilisateur d'inteagir avec une telle base de données est un système de gestion de base de données (SGBD) qui permet principalement d'organiser les données sur le support périphérique (disque, disquette, bande etc.) et fournit les procédures de recherche et de sélection de ces mêmes données.
- **Modèle de données :** Un modèle de données est une représentation abstraite qui définit la structure, les relations et les règles des données dans une base de données. Il sert à organiser les informations et à décrire comment elles sont reliées entre elles.
- **Schéma d'une base de données :** c'est la structure définie de la base de données, qui décrit l'organisation des données (tables, colonnes, types de données, contraintes, relations...). Il est fixe et ne change pas fréquemment.
- **Instance d'une base de données :** c'est l'état actuel des données dans la base à un moment donné. C'est le contenu réel des tables à un instant précis, qui évolue à chaque ajout, modification ou suppression de données.

1.2.2 Associez chaque terme à sa définition correcte :

Clé primaire -> Identifiant unique d'un enregristrement dans une table
Relation -> Structure de données qui stock les informations

Attribut -> Propriété ou caractéristique d'une entité
Tuple -> Ligne d'une table

1.3 Ex3 : Architecture et composants d'un SGBD

1.3.1 Expliquez le rôle des éléments suivants dans un SGBD :

Moteur de stockage gère la manière dont les données sont stockées, organisées et récupérées sur le disque ou en mémoire.

Langage de manipulation des données (DML) manipuler et traiter les données stockées dans une base de données relationnelle.

Langage de définition des données (DDL) garantir une organisation efficace et sécuriser les données. Gestion des transactions garantit que les opérations sur la base de données sont effectuées en toute sécurité.

1.3.2 Associez les différents utilisateurs d'une base de données avec leur rôle principal :

1. Administrateur de la base de données (DBA) : Gère, sécurise et optimise la base de données. Assure la maintenance, les sauvegardes et la gestion des accès.
2. Développeur : conçoit et implémente des bases de données, écrit des requêtes SQL et développe des applications interagissant avec la BD.
3. Utilisateur final : utilise la base de données via une application ou une interface pour consulter, insérer ou modifier des données.
4. Analyste de données : exploite les données pour extraire des insights, génère des rapports et optimise la prise de décision en entreprise.

1.4 Ex4 : Méthodologie et conception d'une base de données

1.4.1 Classez les étapes suivantes dans l'ordre logique de la conception d'une base de données :

1er étape : Analyse des besoins
2 ème étape : Conception conceptuelle
3 ème étape : Conception logique
4 ème étape : Conception physique

1.5 Ex5 : Évolution des SGBD et nouvelles tendances

1.5.1 Classez les architectures suivantes selon leur évolution historique :

1. Mainframe
2. Architecture Client/Serveur
3. Architecture Client/Serveur 3-tiers
4. Bases de Données Décisionnelles
5. NoSQL

1.5.2 Donnez un exemple d'utilisation des bases de données NoSQL et expliquez pourquoi elles sont adaptées à certains contextes.

Les bases de données NoSQL sont utiliser à travers les réseaux sociaux. Elles sont adaptées dans ce cas pour la gestion des données des utilisateurs.

1.6 Ex6 : Exploration des différents types de SGBD (Recherche sur Internet + analyse critique)

1.6.1 Quelles sont les différences fondamentales entre SGBD relationnel (SQL) et SGBD non relationnel (NoSQL) ?

— Structure de données
— Langage de Requête
— Cas d'Utilisation

1.6.2 Faites une comparaison entre MySQL, PostgreSQL, MongoDB et Firebase en termes de Modèle de données, Performance, Cas d'usage privilégiés, Limitations

MySQL est adapté aux applications transactionnelles nécessitant une intégrité des données. PostgreSQL convient aux applications nécessitant des requêtes complexes. MongoDB est idéal pour des applications nécessitant une grande flexibilité dans la structure des données. Firebase est optimisé pour des applications mobiles et web en temps réel.

1.6.3 Trouvez un exemple d'application qui utilise NoSQL et justifiez pourquoi ce choix a été fait.

Un excellent exemple d'application qui utilise NoSQL de manière efficace est Netflix.

Netflix est un excellent choix car il met en valeur plusieurs avantages clés de NoSQL, notamment dans la gestion de volumes massifs de données, la fourniture de scalabilité et la gestion de types de données variés. Voici pourquoi Netflix est un bon exemple :

- **Scalabilité :** Netflix est un service de streaming mondial avec des millions d'utilisateurs qui regardent des vidéos en même temps. Il doit être capable de se scaler horizontalement, en ajoutant des serveurs pour gérer l'augmentation du trafic et des données. Les bases de données NoSQL, comme Cassandra et DynamoDB, permettent à Netflix de se scaler de manière plus efficace que les bases de données SQL traditionnelles, en prenant en charge le scaling horizontal, ce qui signifie qu'il est possible d'ajouter plus de nœuds pour gérer la charge sans avoir besoin de modifier la structure de la base de données existante.
- **Haute disponibilité :** Netflix utilise NoSQL pour maintenir une haute disponibilité, même lors de pics de trafic massifs. Les bases de données NoSQL comme Cassandra offrent une consistance éventuelle et sont conçues pour rester disponibles même si certains nœuds sont hors ligne, ce qui garantit que les utilisateurs peuvent continuer à diffuser des vidéos sans interruption.
- **Schéma flexible :** Netflix doit gérer différents types de données, comme les profils utilisateurs, l'historique de visionnage, les métadonnées du contenu et les recommandations, qui évoluent au fil du temps. Les bases de données relationnelles traditionnelles nécessitent un schéma fixe, ce qui peut être restrictif pour les données qui changent rapidement ou sont variées. Les bases de données NoSQL sont sans schéma ou possèdent un schéma flexible, ce qui facilite le stockage et l'ajustement des types de données (par exemple, les documents JSON dans MongoDB ou les paires clé-valeur dans Redis).
- **Gestion du Big Data :** Avec des millions d'utilisateurs et d'énormes quantités de données générées (comme les habitudes de visionnage, l'historique des recherches, les recommandations, etc.), Netflix a besoin d'une base de données capable de gérer le Big Data. Les bases de données NoSQL sont optimisées pour gérer des données volumineuses, non structurées ou semi-structurées, ce qui est exactement ce que Netflix nécessite. Cela permet à Netflix d'analyser et d'exploiter des ensembles de données massifs pour des fonctionnalités comme les recommandations personnalisées.
- **Performance :** La nécessité d'opérations à faible latence et à haut débit est essentielle pour garantir une expérience utilisateur fluide. Les bases de données NoSQL sont optimisées pour la performance en distribuant les données sur plusieurs nœuds, ce qui permet de réduire la latence et d'assurer des réponses plus rapides lorsque les utilisateurs interagissent avec la plateforme, comme lorsqu'ils recherchent un film ou une série.

2 Partie 2

2.1 Ex1 : Analyse d'un besoin et choix d'un SGBD

Besoins de l'entreprise :

- Gestion des utilisateurs avec leurs informations (nom, email, historique des achats).
- Stockage des produits avec des catégories, descriptions, prix et quantités disponibles.
- Analyse des ventes et génération de rapports.
- Évolutivité (la startup prévoit de croître rapidement).

2.1.1 Analysez les besoins et identifiez les contraintes techniques.

L'entreprise à besoins d'une base de données contenant 3 table, avec en 1er

-> la 1er table aura pour but la gestion des utilisateur, avec en paramètre (nom, email, historique d'achats)
-> la 2d table aura pour but la gestion des stockes, avec en paramètre (descriptions, prix, quantités disponibles)
-> la 3eme table aura pour but la gestion des données avec en paramètre (analyse des ventes et rentrée des profit)

l'entreprise aura besoins également d'un hébrgeur tiers avec la possibilité d'allouer beaucoups de stockage pour pallier à la croissance

2.1.2 Comparez les solutions (SQL vs NoSQL) et choisissez un type de base de données. Justifiez votre choix.

Dans ce cas de figure, nous avons : des données structurée (pas massivent), avec des relation plus ou moins complexe et avec un besoins de pouvoir mettre à l'échelle rapidement le flux de données.

la solution SQL semble donc plus approprier.

2.2 Ex2 : Décryptage des évolutions des SGBD (Recherche sur Internet)

2.2.1 Comment ont évolué les SGBD depuis les années 1970 ? Identifiez les grandes étapes (Mainframe, Client/Serveur, Cloud, NoSQL...).

Les SGBD ont connu une évolution significative depuis les années 1960, marquée par plusieurs étapes clés :

— 1965 - 1970 : Première Génération de SGBD
— Modèles hiérarchique et réseau : Les premiers SGBD étaient basés sur des structures hiérarchiques et en réseau.
— Exemples :
— IMS d'IBM : SGBD hiérarchique développé pour gérer de grandes quantités de données.
— IDS de General Electric : SGBD en réseau qui a influencé les propositions du groupe CODASYL.
— 1970 - 1985 : Deuxième Génération de SGBD
— Modèle relationnel : Introduction du modèle relationnel, offrant une abstraction plus élevée et une manipulation des données via des tables.
— Exemples :
— MRDS de Honeywell : SGBD relationnel diffusé par CII-HB.
— QBE (Query By Example) : Interface utilisateur pour formuler des requêtes en exemple.
— SQL/DS d'IBM : SGBD relationnel utilisant le langage SQL.
— INGRES de Relational Technology : SGBD relationnel académique devenu commercial.
— ORACLE de Relational Software : L'un des premiers SGBD relationnels commerciaux.
— 1985 - 2000 : Troisième Génération de SGBD
— Bases de données orientées objet : Intégration des concepts de programmation orientée objet pour gérer des données complexes.
— Bases de données réparties : Systèmes permettant la distribution des données sur plusieurs sites pour une meilleure disponibilité et performance.
— Bases de connaissances et systèmes experts : Incorporation de règles et de logique pour le raisonnement automatisé.
— 2000 - Présent : Quatrième Génération de SGBD
— Bases de données NoSQL : Conçues pour gérer des volumes massifs de données non structurées avec une scalabilité horizontale.
— Exemples : MongoDB, Cassandra, Redis.
— Cloud Computing : Déploiement de SGBD dans le cloud, offrant flexibilité, scalabilité et réduction des coûts d'infrastructure.
— Exemples : Amazon RDS, Google Cloud SQL, Azure SQL Database.

2.2.2 Recherchez les concepts OLAP et Big Data et expliquez leur rôle dans l'analyse des données.

— OLAP (Online Analytical Processing) :
— Définition : Technologie permettant l'analyse rapide et interactive de données multidimensionnelles issues de bases de données.
— Rôle : Facilite l'exploration des données sous différents angles (dimensions) pour identifier des tendances, des modèles et des insights décisionnels.
— Utilisation : Essentiel pour le reporting, les tableaux de bord et les analyses ad hoc dans les systèmes décisionnels.
— Big Data :
— Définition : Ensemble de données volumineuses, variées et générées à grande vitesse, dépassant les capacités des systèmes traditionnels de gestion de bases de données.
— Rôle : Permet l'analyse de vastes ensembles de données pour découvrir des informations cachées, des corrélations inconnues et soutenir la prise de décision stratégique.
— Utilisation : Appliqué dans divers domaines tels que le marketing, la finance, la santé et les sciences sociales pour des analyses prédictives et descriptives.

2.2.3 Quels sont les SGBD les plus utilisés aujourd'hui ? Trouvez des statistiques et des tendances actuelles.

Les SGBD les plus populaires en 2025 incluent :
— SGBD Relationnels (SQL) :
— MySQL : Apprécié pour sa simplicité et son efficacité, largement utilisé dans les applications web.
— PostgreSQL : Reconnu pour sa robustesse et sa conformité aux standards, avec une communauté active.
— Microsoft SQL Server : Utilisé principalement dans les environnements Windows pour des applications d'entreprise.
— Oracle Database : Préféré par les grandes entreprises pour sa performance et ses fonctionnalités avancées.
— SGBD NoSQL :
— MongoDB : Base de données orientée documents, populaire pour sa flexibilité et son évolutivité.
— Cassandra : Conçue pour gérer de grandes quantités de données réparties sur plusieurs serveurs sans point de défaillance unique.
— Redis : Base de données en mémoire, utilisée pour des opérations nécessitant une latence extrêmement faible.

Les tendances actuelles montrent une adoption croissante des solutions cloud pour les SGBD, offrant des avantages en termes de scalabilité, de coûts et de maintenance simplifiée. De plus, l'intégration de fonctionnalités d'analyse en temps réel et de support pour le Big Data devient un critère clé dans le choix des SGBD.

2.3 Ex3 : Conclusion et discussion finale

2.3.1 Quels ont été les points les plus intéressants à explorer ?

2.3.2 Comment choisir le bon type de base de données selon les besoins d'un projet ?

Le choix du type de base de données dépend principalement des exigences spécifiques du projet, notamment le type de données à gérer, la complexité des requêtes, les performances, la scalabilité, et la disponibilité des données. Voici quelques critères clés à prendre en compte :

Type de données : **Données structurées :** Si le projet implique des données fortement structurées, avec des relations bien définies (comme des utilisateurs, des commandes, etc.), une base de données relationnelle (SQL) comme MySQL, PostgreSQL ou Oracle peut être la meilleure option. Données non structurées ou semi-structurées : Si les données sont moins structurées (par exemple, des documents JSON, des images ou des vidéos), une base de données NoSQL (comme MongoDB pour des documents, Cassandra pour des données distribuées, ou Redis pour des données en cache) pourrait être plus appropriée.

Scalabilité : Si votre projet nécessite de scaler horizontalement (ajouter des serveurs pour gérer la charge), les bases de données NoSQL sont souvent plus adaptées. Elles sont conçues pour être distribuées, et peuvent facilement s'adapter à une augmentation importante du volume de données et du nombre d'utilisateurs. Si la scalabilité verticale (amélioration de la capacité d'un serveur unique) est suffisante, une base de données relationnelle peut être suffisante, bien qu'elle ne soit pas aussi flexible à cet égard.

Consistance vs disponibilité : Si la consistance forte est nécessaire (par exemple, garantir que toutes les copies des données sont synchronisées en temps réel), une base de données relationnelle avec des mécanismes de verrouillage est préférable. Si la disponibilité est plus critique et que l'application peut tolérer une certaine consistance éventuelle, une base de données NoSQL comme Cassandra ou DynamoDB est plus adaptée. Cela est souvent le cas dans les applications web à grande échelle où la tolérance aux pannes et la répartition des données sur plusieurs serveurs sont essentielles.

Performance et latence : Pour des requêtes complexes, impliquant des jointures multiples et une forte normalisation des données, une base de données relationnelle sera plus performante. Pour des requêtes simples à haut débit, comme des lectures ou écritures rapides dans des applications web ou mobiles, une base de données NoSQL est souvent plus rapide et plus efficace.

Budget et gestion : Si votre projet nécessite un coût faible et une gestion simplifiée, des bases de données NoSQL peuvent offrir plus de flexibilité en termes de coût d'exploitation. Certaines bases de données NoSQL sont également gérées dans le cloud (comme MongoDB Atlas ou DynamoDB), ce qui réduit les coûts et les efforts de gestion. Si une gestion stricte des données et des transactions est essentielle (comme dans les systèmes bancaires), alors une base de données relationnelle avec des fonctionnalités de gestion de transactions (ACID) est plus appropriée.

2.3.3 Discussion sur les tendances futures (IA, bases de données distribuées, etc.).

Les tendances futures des bases de données sont marquées par l'évolution des besoins technologiques, notamment avec l'avènement de l'intelligence artificielle (IA), des bases de données distribuées, et de la gestion de données massives. Voici quelques évolutions importantes à surveiller :

Intelligence Artificielle et bases de données : **Optimisation des requêtes par IA :** L'IA pourrait jouer un rôle clé dans l'optimisation des performances des bases de données. Par exemple, des algorithmes d'apprentissage automatique pourraient être utilisés pour optimiser les requêtes en fonction du comportement des utilisateurs, ou pour prédire et ajuster dynamiquement les configurations des bases de données pour améliorer la performance. **Bases de données dédiées à l'IA :** De nouvelles bases de données pourraient être spécialement conçues pour stocker et gérer des données massives liées à l'IA, comme des données d'entraînement de modèles de machine learning. Ces bases de données pourraient être capables d'intégrer nativement des capacités de traitement de données massives, de gestion de modèles, et d'optimisation en temps réel.

Bases de données distribuées : Les bases de données distribuées deviennent de plus en plus populaires avec la croissance des applications mondiales nécessitant une répartition des données sur plusieurs serveurs et régions géographiques. Des bases comme Cassandra, CockroachDB, ou Google Spanner continuent à émerger pour répondre à la demande de scalabilité horizontale tout en garantissant des performances de haute qualité. Ces bases sont essentielles pour des applications de grande envergure qui nécessitent une disponibilité continue et une résilience face aux pannes. Les entreprises, par exemple dans le secteur du e-commerce ou des réseaux sociaux, en bénéficient particulièrement.

Bases de données multi-modèles : Une autre tendance est l'émergence de bases de données multi-modèles, qui permettent de gérer plusieurs types de données (relationnelles, documentaires, graphiques, etc.) au sein de la même plateforme. Cela permet aux entreprises de ne pas se limiter à un seul type de base de données et de choisir le modèle le plus adapté à chaque cas d'utilisation. Par exemple, ArangoDB ou Couchbase permettent de gérer différents modèles de données dans une même base.

Bases de données en cloud et serverless : Les bases de données cloud comme AWS DynamoDB, Google Firebase ou Azure Cosmos DB connaissent une adoption croissante, offrant une gestion simplifiée et évolutive sans avoir à se soucier de l'infrastructure. Cela permet aux développeurs de se concentrer sur le développement d'applications plutôt que sur la gestion des bases de données. Le modèle serverless pourrait se développer davantage, permettant aux entreprises d'utiliser des bases de données sans avoir à gérer de serveurs dédiés. Cela offre de l'élasticité et une facturation à la demande, ce qui est idéal pour les applications avec une charge variable.

Blockchain et bases de données décentralisées : Avec la montée en puissance de la blockchain, les bases de données décentralisées pourraient devenir une tendance clé. Ces bases de données sont conçues pour offrir des transactions transparentes, sécurisées et immuables. Elles sont particulièrement pertinentes pour les industries financières, les chaînes d'approvisionnement, ou encore la gestion des identités.