---

06 - Applications - Exercices - Réponses

Raphaël P. Barazzutti - 3 mars 2024

Exercice 1 : Comprendre l’architecture 3-tiers

1. Rôles des trois tiers :
    - Tier Présentation :

      Il s’agit de l’interface utilisateur (client léger, application web, etc.) qui assure l’affichage des informations et la collecte des données auprès de l’utilisateur.

    - Tier Logique Applicative :

      Ce niveau contient la logique métier. Il effectue les traitements, calculs, validations et règles spécifiques à l’application. Généralement, il s’exécute sur un serveur d’applications.

    - Tier Données :

      Ce niveau correspond au stockage persistant des données (par exemple, une base de données). Il gère la sauvegarde, la récupération et l’organisation des données.

2. Avantages de la séparation en trois couches :
    - Scalabilité améliorée :

      Chaque couche peut être dimensionnée indépendamment selon la charge (on peut ajouter des serveurs uniquement pour la logique ou pour la présentation, par exemple).

    - Modularité et maintenance facilitée : La séparation claire des responsabilités permet de modifier ou mettre à jour une couche sans impacter les autres.

Exercice 2 : Bases de données relationnelles (SQL)

1. Principe fondamental : Une base de données relationnelle stocke les données sous forme de tables. Chaque table est constituée de lignes (enregistrements) et de colonnes (attributs). Les relations entre les tables se font grâce à des clés primaires et étrangères.

2. Exemples de SGBDR :

    - MySQL / MariaDB
    - PostgreSQL
    - Microsoft SQL Server
    - Oracle

3. ACID : ACID est un acronyme pour :

    - Atomicité : chaque transaction est réalisée entièrement ou pas du tout.
    - Cohérence : la base passe d'un état valide à un autre état valide.
    - Isolation : les transactions concurrentes ne se perturbent pas mutuellement.
    - Durabilité : une fois validée, la transaction est conservée même en cas de panne. Ces propriétés garantissent la fiabilité des transactions, ce qui est essentiel pour des applications nécessitant une forte intégrité des données (ex. finance, e-commerce).

Exercice 3 : Bases de données NoSQL

1. Définition et distinction avec SQL :

   Les bases NoSQL sont un ensemble de systèmes de gestion de bases de données qui ne suivent pas le modèle relationnel traditionnel. Elles permettent de stocker et gérer des données dans des formats variés (documents, paires clé-valeur, colonnes, graphes) et offrent une plus grande flexibilité de schéma et souvent une meilleure performance et scalabilité horizontale. Cependant, elles ne garantissent pas toujours une cohérence stricte (ACID).

2. Deux types et exemples :

    - Clé-Valeur :

      Stocke les données sous forme de paires clé/valeur. Exemple : Redis.

    - Document :

      Gère des données structurées sous forme de documents (souvent en JSON/BSON) avec un schéma flexible. Exemple : MongoDB.

3. Limites par rapport aux bases relationnelles :

    - Une cohérence parfois moindre (les transactions ne sont pas toujours ACID).
    - Moins adaptées aux transactions complexes et aux requêtes très structurées ou multi-tables.

- Les outils de visualisation et d'analyse (comme les jointures complexes) sont souvent moins développés que dans les SGBDR.

Exercice 4 : Sélection du type de base

1. Cas de prédilection pour une base NoSQL : Une base NoSQL est particulièrement adaptée lorsque l'application doit gérer de très gros volumes de données, des données semi-structurées ou non structurées, ou lorsqu'une scalabilité horizontale et des réponses ultra-rapides sont nécessaires. Exemple : Des applications web à forte charge comme les réseaux sociaux ou des systèmes de logs massifs.

2. Contexte favorable aux SGBDR (bases relationnelles) :

   Lorsque l'intégrité des données, la cohérence forte et la gestion de transactions complexes sont primordiales (ex. systèmes financiers, e-commerce), une base relationnelle est généralement préférable. Exemple : Un système de gestion de commandes dans un site e-commerce, où chaque transaction doit être irréprochable.

Exercice 5 : Le théorème CAP

1. Signification de C, A et P :

   - **C** (Consistency – Consistance) : Tous les nœuds du système voient la même donnée au même instant.
   - **A** (Availability – Disponibilité) : Chaque requête reçoit une réponse, même en cas de défaillance d'un ou plusieurs nœuds.
   - **P** (Partition Tolerance – Tolérance au partitionnement) : Le système continue de fonctionner malgré des coupures ou des latences dans le réseau entre les nœuds.

2. Pourquoi ces trois propriétés sont incompatibles simultanément :

   Dans un environnement distribué, en présence d'une partition réseau (coupure ou latence entre nœuds), il faut choisir entre maintenir la consistance (s'assurer que toutes les copies de données sont identiques) ou garantir la disponibilité (répondre à chaque requête). Il est impossible de garantir simultanément les trois propriétés car lors d'une partition, pour répondre rapidement (disponibilité), on peut être amené à renoncer à la consistance stricte.

3. Exemple de base privilégiant Disponibilité et Tolérance au partitionnement (AP) :

   Des systèmes tels que Dynamo ou Riak sont conçus pour être hautement disponibles et tolérants aux partitions, au détriment d'une consistance stricte en temps réel.