Architecture Projet PapayaDB

# Architecture globale du projet

Client

API Client

API Serveur

Front DataBase

Bases de données

Comme expliqué sur le schéma si dessus, grâce aux flèches qui représentent la parole, le client ne communique qu’avec l’API qui lui ait réservée. Ainsi, chaque API communiquent entres elles, jusqu’au client ou jusqu’au Front DataBase qui parle à toutes les bases de données.

Chaque API possède la même interface, ce qui nous permet de n’avoir qu’une « sorte » d’API pour plusieurs implémentations, ce qui facilite énormément l’utilisation de celles-ci.

L’API Serveur transforme les requêtes reçues par l’API Client pour qu’elles soient compréhensibles par les bases de données. L’API Client réalise quelque chose de similaire, en transformant les demandes de l’utilisateur pour l’API Serveur.

Chaque API apporte ses transformations pour que les données soient compréhensibles par toutes les entités avec lesquelles elle communique.

# Architecture des bases de données

Chaque base de données est constituée de deux objets principaux : le Reader et le Tree.

## Le Tree

Le Tree est en réalité l’optimisateur des requêtes, et le Reader va permettre de gérer les objets.

Plus spécifiquement, le Tree est un arbre à n fils. Chaque nœud est donc composé d’une requête, d’une liste d’index qui représentent les objets correspondant à la requête et d’une HashMap de Requete :Node qui représentent les fils.

L’arbre se construit facilement, nous allons prendre un exemple pour mieux comprendre.

Je demande : tous les livres qui ont pour auteur Mr X et pour prix 20€.

Mon arbre se constituera de la manière suivante :

[null :null] -> [auteur : Mr X] -> [prix : 20€]

Le nœud [null :null] est la tête, le nœud [auteur :Mr X] contiendra tous les livres dont l’auteur est Mr X. Comme ce nœud a pour fils [prix : 20€] les valeurs de ce second nœud seront parmi tous les livres dont l’auteur est Mr X tous ceux qui ont pour prix 20€.

L’intérêt de trier la liste est de pouvoir faire la requête dans l’autre sens ; si le client demande tous les livres qui coutent 20€ et dont l’auteur est Mr X, alors l’arbre recevra la requête sous la même forme que la requête précédente et il suffira de renvoyer la liste contenue dans le nœud du prix.

Si en revanche le client demande seulement les livre de Mr X, alors il suffit de renvoyer la liste du nœud [auteur : Mr X].

## Le Reader

Concernant le Reader, son élément principal est une MappedByteBuffer ainsi qu’un tableau d’index.

Le premier est un buffer permettant de lire directement dans le fichier grâce à des index. Cela permet d’aller très vite mais surtout de ne rien charger en mémoire de lourd. Les index récupérés grâce à ce buffer sont gardés dans le tableau d’index.

Chaque champ de chaque objet est indexé dans le tableau. Ainsi lorsque l’on veut la valeur d’un champ il suffit d’aller la chercher depuis grâce à son index.

Les index qui sont gardé dans l’arbre sont les index du tableau et non les index du fichier.