

**Projet de recherche sur les SGBD du marché**  
*Dans le cadre du cours FSGBD*  
*(M1 MIAGE UCA NICE)*  
**2022 - 2023**

## Préambule

Le travail ici proposé fait suite aux enseignements dispensés autour du cours Fonctionnement des SGBD. Ce travail est structuré en deux parties :

- Partie 1 : Cette partie concerne l'évaluation d'un SGBD relationnel du marché

Cette partie comptera pour 25% de la note finale. Vous devez choisir un SGBD à évaluer dans la liste ci-dessous ou proposer un SGDB relationnel qui vous convient.

- Partie 2 : Cette partie concernent l'implémentation d'algorithme d'indexation et recherche

Cette partie comptera pour 25% de la note finale.

Vous devez Former des groupes de 4 à 5 personnes avant le 10 Novembre 2022 Voir le lien disponible dans le fichier : LienInscriptionGroupe\_4\_a\_5Personnes\_Max.txt dans le même dossier.

**A rendre au plus tard le 30 Janvier 2022**

# Partie 1 : Evaluation d'un SGBD relationnel du marché

Choix du SGBD à évaluer. Jusqu'à 2 groupes peuvent choisir un même SGBD. Toute recopie entre groupe sera sanctionnée par un zéro.

Voici la liste des SGBD pouvant être choisis :

- 1) MySQL
- 2) Microsoft SQL Server
- 3) IBM DB2
- 4) PostgreSQL
- 5) Sybase de SAP
- 6) Teradata
- 7) Informix
- 8) HIVE
- 9) SAP SQL Anywhere
- 10) Google BigQuery
- 11) FileMaker
- 12) Spark SQL

Vous devez avoir

## 1. Identification du SGBD

(nom, date parution, créateur, propriétaire, modèles de données supportées,

## 2. Installation et prise en main du SGBD

Vous devez installer le SGBD sur votre poste de travail.

Vous devez donner des exemples pratiques lors de réalisation des points 3 à 11.

## 3. Architecture fonctionnelle du SGBD choisi

Dessins d'architecture

Les caches mémoires et leurs rôles

Les processus gravitant autour du cache mémoire et leur rôle

Gestion des connexions

Processus d'exécution des requêtes

Illustrer avec des exemples et démos concrets.

#### 4. Le dictionnaire de données du moteur choisi

Décrire le dictionnaire

Donner la liste de ces tables et leurs rôles

Illustrer avec des exemples et démos concrets.

#### 5. Organisation physique du SGBD

Les différents types de fichiers qui composent ce moteur et leur rôle

Visualiser les informations sur ces fichiers en consultant le dictionnaire ;

Illustrer avec des exemples et démos concrets.

#### 6. Organisation logique des données

Comment sont organisées physiquement les données des tables

Comment sont organisées physiquement les données des index

Comment sont organisées physiquement les données d'annulation

Comment sont organisées physiquement les données temporaires

Visualiser les informations sur les tables, index, etc. en consultant le dictionnaire ;

Illustrer avec des exemples et démos concrets.

#### 7. Gestion de la concurrence d'accès

Notion de transaction

Support des propriétés ACID

Technique de gestion de la concurrence d'accès

La sérialisation

...

Illustrer avec des exemples et démos concrets.

## 8. Gestion des transactions distribuées

Architecture

Positionnement par rapport aux 12 règles de commit

Méthode pour garantir

Traitement des pannes

Illustrer avec des exemples et démos concrets.

## 9. Gestion de la reprise sur panne

Les techniques d'annulation

La journalisation

Les sauvegardes

Gestion de la reprise à chaud

Gestion de la reprise à froid

Illustrer avec des exemples et démos concrets.

## 10. Techniques d'indexation

Les différentes techniques d'indexation avec des exemples pratiques

Illustrer avec des exemples et démos concrets.

## 11. Optimisation de requêtes

Les techniques d'optimisation avec des exemples

Illustrer avec des exemples et démos concrets.

Montrer les plans d'exécution et la façon de les lire.

# Partie 2 : Projet de développement de techniques d'indexations

Pour cette partie, vous travaillerez sur un projet Java de simulation de B+ Tree. Le projet est basique et construit sur une base existante dont les fonctionnalités ont été étendues et dont le code a été commenté pour faciliter la compréhension des algorithmes mis en oeuvre.

En l'état, le projet permet de créer un B+ Tree avec une interface graphique, vous pouvez y ajouter un ensemble de clefs d'un coup, des clefs unes à unes, effacer des clefs, sauvegarder et charger un arbre.

Pour une question de simplicité, nous partirons du principe que les clefs sont uniques. Vous trouverez le projet de base à cette adresse : <https://gitlab.com/fsgbd/b-tree/>

## 1. Indexation - Objectifs

Etendre les fonctionnalités de cette application afin que l'arbre puisse être construit à partir d'un fichier, ce fichier que vous pourrez générer vous-même contiendra 10000 **lignes**.

Chaque ligne sera composée d'un entier (numéro de sécurité sociale par exemple) et d'informations diverses comme par exemple, un nom, un prénom etc...

Vous pourrez indexer votre fichier à partir de l'entier ou d'une autre colonne. La **colonne** sur laquelle votre **index** sera basé déterminera la **clef** de votre arbre, le **pointeur** associé à chaque clef contiendra la **position de la ligne** (index du début de la ligne dans le fichier) en question **dans le fichier**. Recherche - Objectifs

Avant de travailler sur le chargement de fichier, vous devrez modifier l'application afin que les clefs soient associées à des pointeurs car ce n'est pas le cas à l'heure actuelle.

Afin de rendre l'arbre B+ complètement fonctionnel, vous devrez aussi ajouter une fonctionnalité absente. Dans un arbre B+, chaque feuille de l'arbre doit être capable de référencer la feuille suivante afin qu'il soit possible d'effectuer un parcours séquentiel de toutes les feuilles de l'arbre. Vous devrez ajouter cette fonctionnalité.

## 2. Recherche - Objectifs

Finalement les indexes sont utilisés pour proposer une solution efficace pour rechercher une donnée. Leurs avantages, au-delà du fait qu'ils aident à avoir des temps de recherche constant est qu'ils accélèrent grandement la recherche par rapport à une recherche séquentielle classique.

Vous effectuerez pour cette partie une petite étude qui consistera à rechercher des lignes dans votre fichier en vous basant sur l'index que vous aurez précédemment créé et vous comparerez les temps de chargement avec une recherche séquentielle classique.

Pour avoir une base solide de comparaison, vous vous baserez sur 100 recherches différentes.

Produire des statistiques (temps minimum / maximum / moyen) sur les temps de recherche et comparer les deux approches :

- Recherche séquentielle dans le fichier
- Recherche depuis l'index