# Compte rendu du TME1_2 SAM

Anyes TAFOUGHALT

January 28, 2024

## 1 Exercice 1 : Création d'index

### 1.1 Création d'index unique pour l'attribut $a_0$

La fonction **creation_index_unique** prend en paramètre une table et établit un index unique en associant les valeurs du premier attribut de chaque tuple à un rowid constitué du numéro de la page et de la position dans la page.

Elle parcourt de manière séquentielle tous les tuples de chaque page de la table, extrait la valeur du premier attribut, et l'utilise comme clé associée à un rowid (numéro de la page et position) pour construire un dictionnaire trié à l'aide de la bibliothèque sortedcontainers.

En résultat, cette structure d'index offre un accès rapide et trié aux données de la table lorsqu'on recherche un tuple avec une valeur spécifique du premier attribut.

### 1.2 Création d'index non unique pour l'attribut $a_i$

La fonction **creation_index** prend en paramètres une table et le numéro d'un attribut. Elle crée un index non unique associant les valeurs de l'attribut spécifié à une liste de rowids, composés du numéro de page et de la position dans la page.

Le processus consiste à parcourir séquentiellement tous les tuples de chaque page de la table. Pour chaque tuple, la fonction extrait la valeur de l'attribut spécifié, et si cette valeur n'est pas déjà présente dans l'index, elle crée une nouvelle entrée avec la valeur comme clé et une liste contenant le rowid comme valeur. Si la valeur est déjà présente, la fonction ajoute simplement le rowid à la liste existante.

En fin de compte, la fonction retourne un dictionnaire trié à l'aide de la bibliothèque sortedcontainers, où chaque valeur de l'attribut est associée à une liste de rowids, permettant ainsi un accès rapide aux données de la table en fonction de cet attribut.

## 2 Exercice 2 : Accès par index

### 2.1 Accés ciblé

#### 2.1.1 Index unique scan

La fonction **acces_par_index_unique** prend un index unique, une table, et une valeur de recherche en entrée. Elle exploite l'index pour extraire le numéro de page et la position associés au tuple ayant la valeur recherchée comme premier attribut.

Ensuite, en utilisant ces informations, elle lit la page correspondante et accède directement à la position où se trouve le tuple dans la table. La fonction récupère ce tuple et le renvoie.

En résumé, elle permet un accès rapide et ciblé à un tuple spécifique de la table grâce à l'index unique.

#### 2.1.2 Index scan

La fonction **acces_par_index** prend en paramètres un index, une table, et une valeur de recherche. Elle utilise l'index pour obtenir les rowids (numéro de page et position) associés aux tuples qui ont la

valeur recherchée comme attribut. Ensuite, elle parcourt ces rowids, lit les tuples correspondants dans la table, et les stocke dans une liste. Enfin, la fonction retourne cette liste de tuples.

En résumé, elle permet d'extraire et de renvoyer tous les tuples associés à une valeur spécifique via l'index fourni.

#### 2.1.3 Accès par intervalle sur un attribut unique

La fonction **acces_intervalle_par_index_unique** prend en paramètres un index, une table, et deux bornes (borne_inf et borne_sup) définissant un intervalle sur un attribut. Elle utilise la méthode bisect_left pour déterminer l'indice de la première valeur de l'intervalle dans l'index trié. Ensuite, elle parcourt les clés de l'index à partir de cette position jusqu'à ce qu'elle atteigne une clé supérieure à la borne supérieure de l'intervalle.

À chaque itération, la fonction appelle la fonction **acces_par_index_unique** pour récupérer le tuple associé à chaque valeur de l'attribut dans cet intervalle. Le tuple obtenu est ensuite ajouté à la liste tuples. Finalement, la liste complète des tuples dans l'intervalle spécifié est renvoyée.

En résumé, cette fonction permet d'accéder à tous les tuples dont la valeur de l'attribut indexé se situe dans un intervalle donné, en utilisant un index et en exploitant la méthode **acces_par_index_unique**.

#### 2.1.4 Accès par intervalle sur un attribut non unique

Pour la fonction **acces_intervalle_par_index**, elle a le méme principe que la fonction précédentes sauf que celle ci fait appelle à la fonction **acces_par_index** au lieu de **acces_par_index_unique** afin d'obtenir la liste des tuples associés à chaque valeur de l'attribut dans cet intervalle et les ajoutés à la liste tuples q'on va retourner à la fin de cette fonction

La fonction d'accès par intervalle par index non unique **acces_intervalle_par_index** suit le même principe que la fonction précédente, à la différence qu'elle fait appel à la fonction d'accès par index plutôt qu'à la fonction d'accès par index unique. Cela a pour but d'obtenir la liste des tuples associés à chaque valeur de l'attribut dans cet intervalle, et ces tuples sont ajoutés à la liste qui sera retournée à la fin de la fonction.

# 3 Exercice 3 : Mise à jour de données

## 3.1 Modification d'un seul tuple

La fonction **update_unique** est spécifiquement conçue pour mettre à jour de manière efficace la valeur d'un attribut d'un tuple dont le premier attribut a une valeur donnée.

Cette fonction prend en paramètres un index unique associé au premier attribut des tuples, une table, et une valeur cible. En utilisant cette valeur comme clé dans l'index, on obtient le numéro de page (num_page) et la position dans la page (position) où se trouve le tuple ayant cette valeur comme premier attribut.

Ensuite, la page contenant le tuple est ouverte en mode lecture et écriture ("r+"). Les lignes de la page sont lues pour extraire la ligne correspondant au tuple à mettre à jour. L'attribut $a_1$ est modifiée en ajoutant 10 à sa valeur actuelle.

Les modifications sont ensuite écrites dans le fichier de la page. En résumé, la fonction **update_unique** offre une approche optimisée pour localiser, modifier et enregistrer un tuple recherché à l'aide de l'index unique du premier attribut. Cette approche évite de recharger l'intégralité des pages et se base sur les informations de l'index pour accéder directement à la page et la position nécessaires.

## 3.2 Modification de plusieurs tuples

La fonction **update_plusieurs** vise à améliorer l'efficacité de la mise à jour de plusieurs tuples ayant une valeur spécifique dans l'attribut indexé $a_2$. Dans un premier temps, elle identifie les rowids correspondants (numéro de page et position) pour les tuples possédant cette valeur, en utilisant l'index associé. Ces rowids sont ensuite organisés dans un dictionnaire, où chaque clé représente un numéro de page et les valeurs sont les positions des tuples concernés dans cette page. Cette structure prévient la nécessité de lire une même page plusieurs fois, améliorant ainsi les performances.

Ensuite, la fonction parcourt chaque page concernée, ouvre le fichier associé en mode lecture et écriture, ajuste les valeurs des tuples à mettre à jour en se basant sur les positions, puis enregistre ces modifications dans le fichier de la page. Dans l'ensemble, **update_plusieurs** constitue une solution efficace et bien structurée pour gérer les mises à jour multiples. Elle exploite de manière astucieuse les informations d'index, optimisant l'accès aux pages de manière ciblée et évitant des opérations redondantes.

### 3.3 Modification de l'index en conséquence lorsque l'attribut modifié est indexé

La fonction **update_plusieurs_2** est similaire par rapport à la fonction précédente. Contrairement à la fonction précédente, cette version prend en compte non seulement l'index de l'attribut utilisé pour la recherche des tuples, mais aussi l'index de l'attribut sujet de la modification. Cela implique une étape supplémentaire dans laquelle l'index doit être mis à jour après la modification des tuples.

Pour chaque tuple mis à jour, son rowid est retiré de l'entrée de l'index, où la clé correspond à l'ancienne valeur de l'attribut modifié. Ensuite, ce rowid est ajouté à la liste des rowids de l'entrée de l'index, où la clé est la nouvelle valeur de l'attribut. Si cette nouvelle valeur n'existe pas dans l'index, une nouvelle entrée est créée avec la nouvelle valeur comme clé et le rowid comme valeur. Enfin, l'index est trié pour maintenir son ordre.

Cette adaptation de la fonction vise à garantir la cohérence de l'index après chaque mise à jour, offrant ainsi une solution complète pour la gestion des modifications multiples dans une table indexée.

## 4 Exercice 4 : Persistence

### 4.1 Stockage d'un index unique

Pour garantir la persistance de l'index, nous avons mis en place la fonction **index_unique**. Cette fonction prend en entrée l'index que l'on souhaite rendre persistant ainsi que le nombre d'entrées par page. Son objectif est de créer un fichier stockant les entrées de l'index. Pour chaque entrée, elle vérifie si la page en cours n'a pas encore atteint sa limite maximale de lignes. Si tel est le cas, elle ajoute une nouvelle ligne correspondant à la valeur de l'attribut et à son rowid. Dans le cas contraire, elle écrit la page actuelle, crée une nouvelle page, puis insère l'entrée dans cette nouvelle page. Ce processus se répète jusqu'à ce que toutes les entrées de l'index soient traitées.

### 4.2 Stockage d'un index non unique

La fonction **index_non_unique** a pour objectif de générer de manière persistante un index non unique. Elle prend en entrée l'index à traiter, l'attribut (`att`) sur lequel repose l'index, le nom de la table, et le nombre maximal d'entrées par page (`nb_index_par_page`, dont la valeur par défaut est de 10 000). La fonction trie les clés de l'index et itère sur ces clés. À chaque clé, elle évalue si l'ajout des rowids associés dépasserait le nombre maximal d'entrées par page. Si cette limite est atteinte, la fonction crée une nouvelle page où elle inscrit la clé (représentant la valeur de l'attribut) ainsi que tous les rowids correspondants, étant donné que l'index est non unique (une ligne par rowid). Ensuite, elle commence à ajouter les nouvelles entrées à cette page créée. Ce processus se répète jusqu'à ce que toutes les entrées de l'index aient été traitées.

### 4.3 index non dense

En se référant aux pages des index, qu'il s'agisse d'index uniques ou non, on établit un index non dense pour les pages trié. Étant donné que ces pages sont triées, on crée l'index qui stockent la première valeur de l'attribut présente dans la page des index comme clé, tandis que son rowid qui s'agit du numéro de cette page est enregistré comme valeur. Cette approche est conçue pour simplifier la recherche et l'accès aux données et pour retrouver le rowid associé à une valeur donnée sans parcourir toutes les pages d'index.

### 4.4 Mise à jour de données

La fonction **modification_bis** suit le même principe que la fonction **update_plusieurs_2**, mais elle se concentre sur la mise à jour des pages d'index associées aux valeurs de l'attribut 3. Pour obtenir les rowids des tuples à modifier, elle s'appuie sur l'index associé à l'attribut 1, permettant ainsi de trouver les pages et les positions des tuples ayant la valeur recherchée en tant que premier attribut. Les rowids récupérés sont ensuite regroupés dans un dictionnaire qui classe, pour chaque page, toutes les positions des tuples à modifier (afin d'éviter la relecture multiple de la même page). Ensuite, chaque page à modifier est parcourue. Pour chaque page, la fonction lit la page et explore toutes les positions des tuples à modifier, puis écrit la page après la mise à jour.

Pour chaque tuple modifié, les pages de l'index du troisième attribut sont également modifiées. Cela implique l'utilisation de l'index non dense de ces pages pour déterminer quelle page lire, évitant ainsi le parcours de l'ensemble des pages. La fonction utilise la méthode bisect sur l'index non dense pour récupérer l'indice de la première clé supérieure ou égale à l'ancienne valeur de l'attribut 3 du tuple modifié. Si la clé à cet indice dans l'index est égale à la valeur, la page associée est lue, sinon, la page précédente est lue. Une fois la page lue, chaque ligne est parcourue pour localiser l'entrée à supprimer. De manière similaire, en utilisant le même principe, la fonction détermine la page et la position pour insérer la nouvelle valeur du troisième attribut du tuple modifié avec son rowid dans les pages d'index.

## 5 Exercice 5 :Index bitmap

Pour résoudre cet exercice, on peut créer des matrices bitmap pour les attributs $a_5$ et $a_6$ de la manière suivante :

- Identifier toutes les valeurs distinctes de l'attribut $a_5$.
- Créer une matrice bitmap où chaque ligne correspond à une valeur distincte de $a_5$, et chaque colonne représente un rowid (numéro de page et position) associé à un tuple de la table.
- Remplir la matrice bitmap en attribuant un "1" si le tuple se trouvant dans le rowid de la colonne a comme 5éme attribut la valeur correspondant à la ligne, et un "0" sinon.
- Répéter le même processus pour créer la matrice bitmap de l'attribut $a_6$.

Pour rechercher les tuples ayant $a_5 = v_1$ et $a_6 = v_2$, on va utiliser les matrices bitmap de $a_5$ et $a_6$. En examinant la ligne de la matrice bitmap de $a_5$ correspondant à $a_5 = v_1$ et la ligne de la matrice bitmap de $a_6$ correspondant à $a_6 = v_2$, les rowids marqués par "1" dans les deux lignes indiquent les tuples satisfaisant les conditions. Ces rowids peuvent ensuite être utilisés pour lire les pages correspondantes et récupérer les tuples en utilisant les positions.