

TRƯỜNG ĐẠI HỌC SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN



**TIỂU LUẬN**  
**CƠ SỞ DỮ LIỆU NÂNG CAO**

MSSV: 3120410297

Họ tên: Trần Nguyên Lộc

Lớp môn học: DCT1201

Năm học: 2022-2023, Học kì: 1

Giảng viên giảng dạy: PGS. TS .Nguyễn Tuấn Đăng

*Thành phố Hồ Chí Minh - Tháng 05/2022*

## **Mục lục**

<b>1. MỤC TIÊU ĐỀ TÀI TIỂU LUẬN.....</b>	<b>1</b>
1.1 Lí do chọn đề tài.....	1
1.2 Đối tượng nghiên cứu .....	2
1.3 Mục tiêu của việc nghiên cứu .....	2
<b>2. CƠ SỞ LÝ THUYẾT .....</b>	<b>3</b>
2.1 Lý thuyết tổng quát .....	3
2.2 Primary Index.....	4
2.3 Clustering Index .....	5
2.4 Secondary Index.....	8
2.5 Tổng kết .....	9
<b>3. CHƯƠNG TRÌNH VÀ XỬ LÝ BÀI TOÁN .....</b>	<b>10</b>
3.1 Mã nguồn và lời giải .....	10
3.2 Các lưu đồ thuật toán .....	15
<b>4. KẾT QUẢ BÀI TOÁN.....</b>	<b>18</b>
4.1 Kết quả tính toán bằng Excel .....	18
4.2 Kết quả tính toán bằng chương trình (Kiểm thử chương trình).....	18
<b>5. TÀI LIỆU THAM KHẢO .....</b>	<b>21</b>

# **1. MỤC TIÊU ĐỀ TÀI TIỂU LUẬN**




## **1.1 Lí do chọn đề tài**

Ngày nay, Cơ sở dữ liệu là một phần cốt lõi thiết yếu của hầu hết các kiến trúc phần mềm, chúng ta cần cơ sở dữ liệu để lưu trữ thông tin ứng dụng, thông tin khách hàng, thông tin cá nhân người sử dụng hay là thông tin quan trọng của công ty và việc thiết kế một cơ sở dữ liệu tốt cũng đồng nghĩa với việc sử dụng chúng sẽ tốt hơn rất nhiều, một cơ sở dữ liệu được thiết kế tốt sẽ góp phần giảm bớt thời gian truy vấn dữ liệu, hạn chế rủi ro từ việc truy vấn và góp phần tạo thêm lợi ích cho cơ sở dữ liệu đó.

Cơ sở dữ liệu được thiết kế tốt không chỉ quan trọng mà một phần của quy trình thiết kế còn liên quan đến việc đảm bảo rằng cấu trúc của hệ thống được phát triển đủ hiệu quả để đáp ứng yêu cầu của người dùng hiện tại và trong tương lai. Việc điều chỉnh cơ sở dữ liệu một cách có kỹ thuật có thể cải thiện hiệu suất, do đó trong bài báo cáo này chúng ta sẽ xem xét các vấn đề về thiết kế chỉ mục và các cách tích hợp để sử dụng chỉ mục. Các chỉ mục trong cơ sở dữ liệu đóng một vai trò tương tự như một quyển sách và trong đó các chỉ mục được sử dụng để tăng tốc độ truy cập thông tin. Các cấu trúc tệp có thể bị ảnh hưởng bởi các kỹ thuật lập chỉ mục khác nhau và chúng sẽ có ảnh hưởng đến hiệu suất cơ sở dữ liệu. Các chỉ mục có thể giúp các nhà phát triển cơ sở dữ liệu xây dựng cấu trúc tệp hiệu quả và đưa ra các phương pháp truy cập hiệu quả. Khi được sử dụng và điều chỉnh đúng cách, hiệu suất cơ sở dữ liệu có thể được cải thiện hơn nữa. Trên thực tế, các chỉ mục có lẽ là cơ chế quan trọng nhất có sẵn rõ ràng cho các nhà phát triển cơ sở dữ liệu và quản trị viên để điều chỉnh hiệu suất của cơ sở dữ liệu.

## 1.2 Đối tượng nghiên cứu

Đối tượng nghiên cứu của bài báo cáo này tập trung vào phần Single Level, trong đó bao gồm các thành phần chỉ mục sau:

-  Primary Index
-  Clustering Index
-  Secondary Index

## 1.3 Mục tiêu của việc nghiên cứu

Mục tiêu của việc nghiên cứu nhằm giúp người đọc hiểu hơn về cách các chỉ mục hoạt động như thế nào trong CSDL và cũng như việc ứng dụng của nó trong thực tế như thế nào, cũng như sẽ có ví dụ cho việc ứng dụng bằng các bài toán kĩ thuật trong bài viết.

## 2. CƠ SỞ LÝ THUYẾT

### 2.1 Lý thuyết tổng quát

Chỉ mục là các kỹ thuật được sử dụng để lưu trữ một lượng lớn dữ liệu có cấu trúc trên đĩa. Những kỹ thuật này rất quan trọng đối với các nhà phát triển thiết kế CSDL, ví dụ như: DBA – Quản trị viên CSDL, DBMS – Người triển khai CSDL

Một ứng dụng có sử dụng cơ sở dữ liệu điển hình luôn cần truy cập cơ sở dữ liệu và truy xuất một số dữ liệu để xử lý. Bất cứ khi nào cần dữ liệu nhất định, phần dữ liệu đó phải được đặt trên đĩa và được tải vào bộ nhớ chính để xử lý và sau đó được ghi trở lại đĩa nếu có sự thay đổi. Dữ liệu được lưu trữ trên đĩa được tổ chức dưới tệp của bản ghi và mỗi bản ghi là một tập hợp các dữ liệu có thể hiểu là sự thật về các thực thể (entity), thuộc tính (attribute) và mối quan hệ (relationship) của chúng. Do đó các bản ghi nên được lưu trữ trên đĩa theo cách có thể định vị, và vì thế chúng ta có thể tìm chúng một cách dễ dàng.

Vậy trong thực tế, chỉ mục (Indexing) là gì và có công dụng gì? Indexing là một cấu trúc giúp xác định nhanh chóng các bản ghi (record) trong bảng (table). Hiểu một cách đơn giản thì nếu không có chỉ mục (indexing) thì CSDL phải scan toàn bộ bảng (table) để tìm được các bản ghi (record) có liên quan. Dữ liệu càng lớn tốc độ truy vấn (query) càng chậm

Do đó, cấu trúc truy cập còn được gọi là chỉ mục, chúng được sử dụng để tăng tốc độ truy xuất bản ghi nếu một số yêu cầu về điều kiện tìm kiếm được đáp ứng. Chỉ mục cho tệp bản ghi hoạt động giống như danh mục chỉ mục trong thư viện. Ví dụ, trong một môi trường thư viện bình thường, nên có các danh mục như chỉ mục tác giả và chỉ mục tên sách. Người dùng có thể sử dụng một trong các chỉ mục này để nhanh chóng tìm thấy vị trí của một cuốn sách được yêu cầu, nếu họ biết (các) tác giả hoặc tên sách. Mỗi chỉ mục cung cấp một đường dẫn truy cập tới

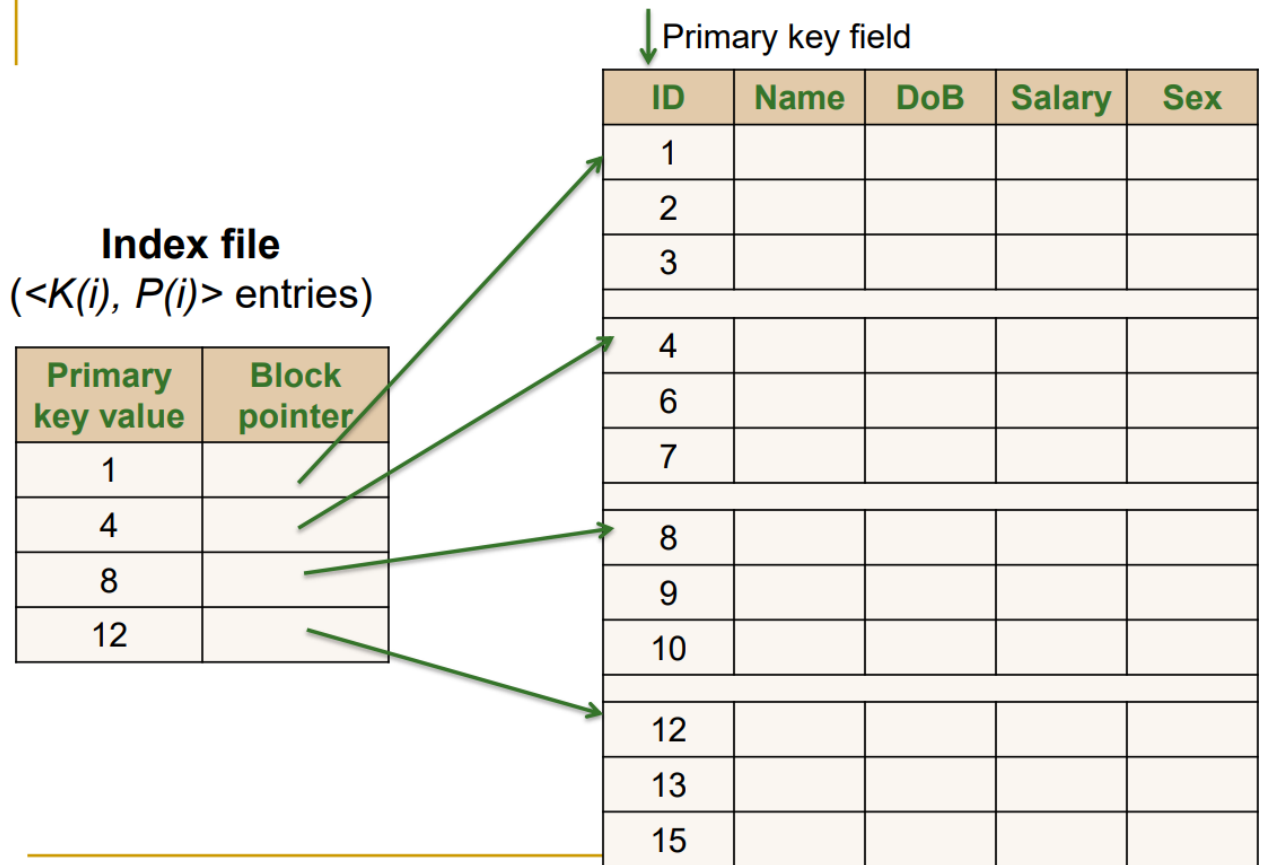
các bản ghi. Một số loại chỉ mục, được gọi là đường dẫn truy cập thứ cấp, không ảnh hưởng đến vị trí vật lý của các bản ghi trên đĩa. Thay vào đó, chúng cung cấp các đường dẫn tìm kiếm thay thế để định vị các bản ghi một cách hiệu quả dựa trên các trường lập chỉ mục. Các loại chỉ mục khác chỉ có thể được xây dựng trên một tệp với một tổ chức chính cụ thể. Trong đó 3 phương pháp lập chỉ mục chính được đề cập đến trong bài viết này là:

1. Primary Index
2. Clustering Index
3. Secondary Index

## **2.2 Primary Index**

Chỉ mục chính (primary index) được định nghĩa chủ yếu trên khóa chính (primary key) của tập tin dữ liệu (data file), trong đó tập tin dữ liệu đã được sắp xếp thứ tự dựa trên khóa chính.

Chỉ mục chính (primary index) là một tập tin có thứ tự có các bản ghi (record) có độ dài cố định với hai trường (field). Trường đầu tiên của chỉ mục sao chép (replicate) khóa chính của tập tin dữ liệu theo cách có thứ tự và trường thứ hai của chứa một con trỏ (pointer) trỏ đến khối dữ liệu (data block) nơi có bản ghi chứa khóa (key). Số khối (khối) trung bình sử dụng chỉ mục chính là  $= \log_2 B + 1$ , trong đó B là số khối của chỉ mục.



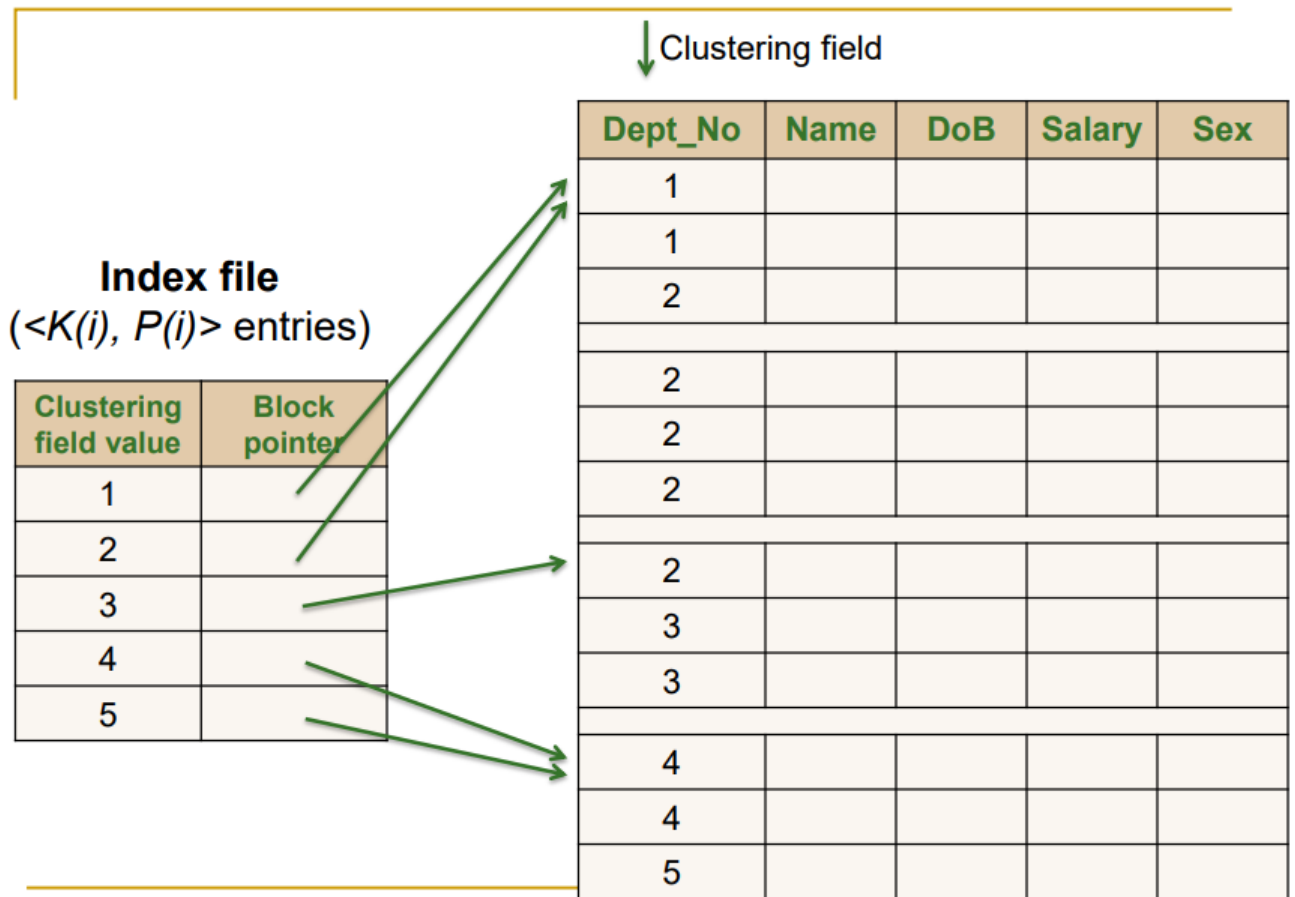
2.2.1 Hình minh họa cách hoạt động của Primary Index

## 2.3 Clustering Index

Nếu các bản ghi (record) của một tệp được sắp xếp theo thứ tự vật lý trên một trường không khóa (non-key field) có thể không có giá trị duy nhất cho mỗi bản ghi, thì trường đó được gọi là trường phân cụm (the clustering field). Dựa trên các giá trị của trường phân cụm, một chỉ mục phân cụm có thể được xây dựng để tăng tốc độ truy xuất các bản ghi có cùng giá trị cho trường phân cụm.

Chỉ mục phân cụm (clustering index) cũng là một tệp được sắp xếp gồm các bản ghi có độ dài cố định với hai trường. Trường đầu tiên cùng loại với trường phân cụm (the clustering field) của tệp dữ liệu và trường thứ hai là một con trỏ khối (a block pointer).

Ngoài ra, ta có 2 trường hợp thường thấy của Clustering index:



2.3.1 Trường hợp 1

Trường hợp 1, các bản ghi dữ liệu (record) được chứa chung trong 1 block data, điều đó khiến cho việc truy vấn dữ liệu từ index file đến data file hoạt động theo cách block pointer từ index file trở đến block có chứa dữ liệu trùng với giá trị clustering field value, và chúng luôn trở đến block anchoring ở đầu mỗi block data để tìm dữ liệu từ trên xuống.





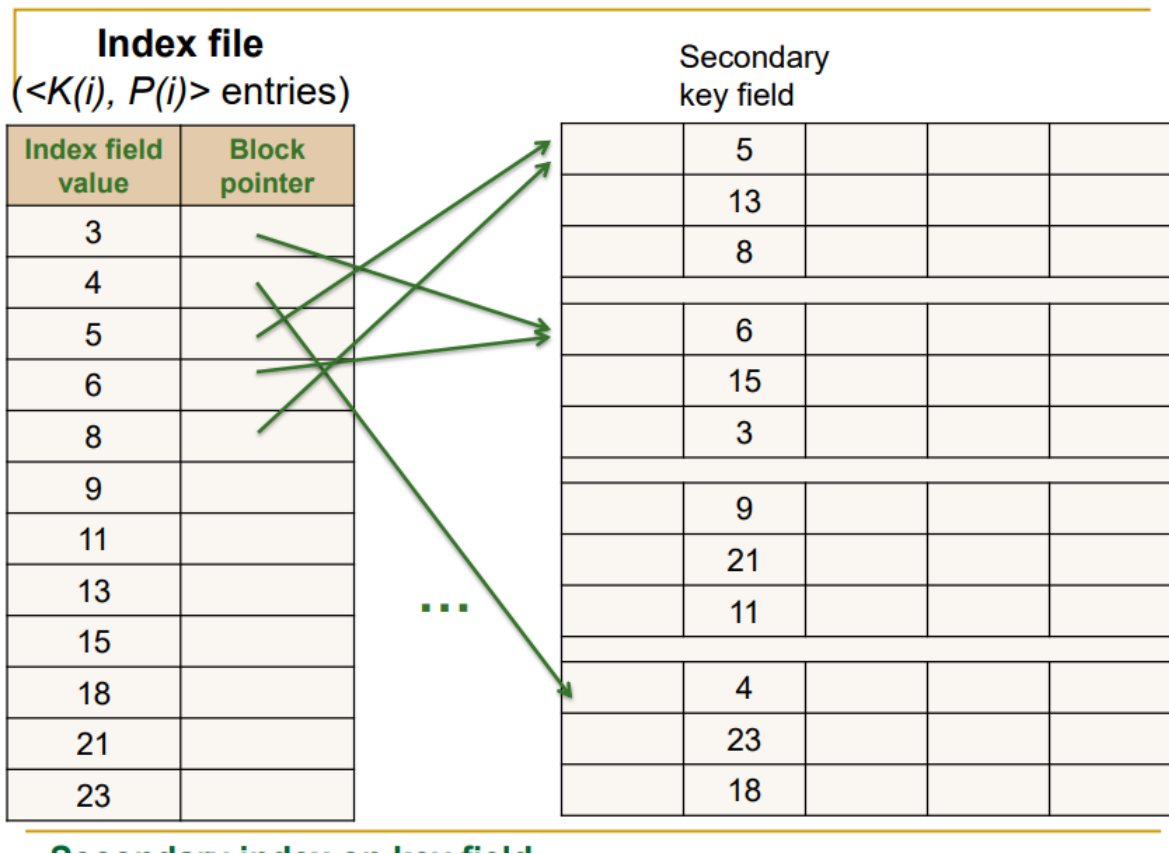
## 2.4 Secondary Index

Chỉ mục phụ (Secondary index) là một tệp bản ghi được sắp xếp (có độ dài cố định hoặc độ dài thay đổi) với hai trường. Trường đầu tiên có cùng kiểu dữ liệu với trường lập chỉ mục (tức là trường không theo thứ tự mà chỉ mục được xây dựng trên đó). Trường thứ hai là con trỏ khối hoặc con trỏ bản ghi. Một tệp có thể có nhiều hơn một chỉ mục phụ. Ngoài ra, secondary index cung cấp cho file một hướng truy cập dữ liệu khác khi file đã có vài quyền truy cập chính đã tồn tại. Đối với secondary index, dữ liệu trong bản ghi file có thể được sắp xếp hoặc không sắp xếp hoặc bị chia. Ngoài ra, secondary index có thể tạo ra các trường có tác dụng là khóa ứng viên và có giá trị duy nhất trong mỗi bản ghi, hoặc có trị trùng lặp trên trường không khóa.

Chúng ta cũng có thể tạo một secondary index trên 1 nonkey, 1 non-ordering field của file. Trong trường hợp đó, số lượng bản ghi trong file có thể sẽ có vài giá trị giống với indexing field.

Trong đó, chúng ta xem xét 3 trường hợp lựa chọn để triển khai chỉ mục phụ:

1. Trường hợp lựa chọn 1: Ta dùng các index đầu vào trùng lặp với giá trị của mỗi bản ghi. Nó có thể là dense index.
2. Trường hợp lựa chọn 2: Ta sử dụng bản ghi có độ dài thay đổi theo index đầu vào, với các trường lặp lại cho con trỏ. Trong lựa chọn 1 hoặc 2, tìm kiếm nhị phân trên index phải được sửa đổi thích hợp để tính số lượng index đầu vào cho mỗi giá trị của key index.
3. Trường hợp lựa chọn 3: Đây là lựa chọn được sử dụng phổ biến nhất trong cả 3. Đó là giữ cho index đầu vào được chỉnh sửa theo độ dài cố định và có 1 index đầu vào cho mỗi giá trị của index field.



2.4.1 Hình minh họa cách hoạt động của Secondary Index

## 2.5 Tổng kết

Kiến thức từ 3 chi mục trên chúng ta có thể tóm gọn bằng 1 bảng thống kê như sau:

Type of Index	Number of (First-level) Index Entries	Dense or Nondense (Sparse)	Block Anchoring on the Data File
Primary	Number of blocks in data file	Nondense	Yes
Clustering	Number of distinct index field values	Nondense	Yes/no <sup>a</sup>
Secondary (key)	Number of records in data file	Dense	No
Secondary (nonkey)	Number of records <sup>b</sup> or number of distinct index field values <sup>c</sup>	Dense or Nondense	No

<sup>a</sup>Yes if every distinct value of the ordering field starts a new block; no otherwise.

<sup>b</sup>For option 1.

<sup>c</sup>For options 2 and 3.

## 3. CHƯƠNG TRÌNH VÀ XỬ LÝ BÀI TOÁN

### 3.1 Mã nguồn và lời giải

```
import Feature.ClusteringIndex;
import Feature.PrimaryIndex;
import Feature.SecondaryIndex;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.zip.ZipEntry;

public class Main {
    public static void main(String[] args) {
        // Thực thi cả 3 feature: PrimaryKey, ClusteringKey, SecondaryKey
        readFileTask("Task_1.txt", 1); // PrimaryKey
        readFileTask("Task_2.txt", 2); // ClusteringKey
        readFileTask("Task_3.txt", 3); // SecondaryKey
    }
    private static void readFileTask(String fileName, int task) {
        float record = 0, blockSize = 0, recordSize = 0, Vssn = 0, blockPointer = 0, zipCode = 0;
        Scanner file = null;

        // Try-catch case: Tìm thấy/Không tìm thấy thư mục
        try {
            file = new Scanner(new File(fileName));
        } catch (FileNotFoundException e){
            System.err.println(fileName + " không tìm thấy!");
            System.exit(1);
        }

        // Đọc dữ liệu từ file
        while(file.hasNextLine()) {
            // Đọc ghi dữ liệu vào biến
            String[] terms = file.nextLine().split(":");
            switch (terms[0]) {
                case "Record_Size_(R)" -> recordSize = Integer.parseInt(terms[1]);
                case "Block_Size_(B)" -> blockSize = Integer.parseInt(terms[1]);
                case "Block_Pointer_(P)" -> blockPointer = Integer.parseInt(terms[1]);
                case "SSN_Field_(Vssn)" -> Vssn = Integer.parseInt(terms[1]);
                case "Number_of_Record_(r)" -> record = Integer.parseInt(terms[1]);
                case "Zipcode" -> zipCode = Integer.parseInt(terms[1]);
            }
        }
        file.close(); // Đóng file

        // Thực thi chương trình
        switch (task) {
            case 1 -> new PrimaryIndex(record, blockSize, recordSize, Vssn,
blockPointer).mainProgress();
            case 2 -> new ClusteringIndex(record, blockSize, recordSize, Vssn, blockPointer,
zipCode).mainProgress();
            case 3 -> new SecondaryIndex(record, blockSize, recordSize, Vssn,
blockPointer).mainProgress();
        }
    }
}
```

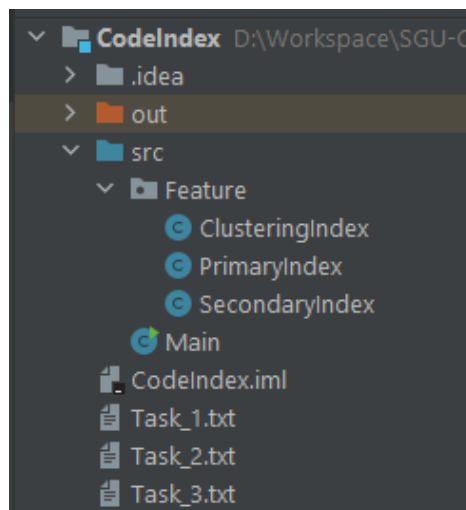
3.1.1 Source code hàm main của chương trình

Như hình 3.1.1 bên trên ta có thể thấy đoạn code đó là nền móng cho cả chương trình thực hiện. Cách thức hoạt động vô cùng đơn giản, khi ta chạy chương trình thì nó sẽ tiến hành đọc dữ liệu input từ file được viết sẵn cho từng trường hợp Index khác nhau và dữ liệu từ file input đó sẽ được truyền cho các class mang nhiệm vụ giải quyết các trường hợp trong bài tập.

```
1 Record_Size_(R):40
2 Block_Size_(B):4096
3 Block_Pointer_(P):6
4 SSN_Field_(Vssn):9
5 Number_of_Record (r):600000
```

3.1.2 Hình ảnh lấy từ mã nguồn cho biết các dữ liệu input như thế nào

Và như thế ta sẽ có 3 file input dữ liệu đầu vào ứng với 3 task (3 câu hỏi có trong phụ lục ), ta có thể quan sát trên hình 3.1.3. Và cũng như theo đoạn code trên thì em cũng có chia package của chương trình và đặt chúng như sau:



3.1.3 Hình ảnh các file .txt được sử dụng để làm input cho chương trình

Đối với từng các bài tập trong phụ lục, đưa vào dạng câu hỏi của 3 câu em xin phép chia chúng ra làm 3 task khác nhau (hình 3.1.3), đặt ở package Feature

và đặt tên class của chúng theo dạng index của bài tập đó. VD: PrimaryIndex = Bài tập 1, ClusteringIndex = Bài tập 2

### ❖ Bài 1

Đối với bài tập 1 trong phụ lục, ta có mã nguồn như sau:

```
package Feature;

public class PrimaryIndex {
    private float record, blockSize, recordSize, Vssn, blockPointer; // input value
    private float bfr, b, Ri, bfri, ri, bi, binaraySearchDataFile, binarySearchIndexFile,
    binraySearchWithSupportIndex, linearSearchDataFile; // output value

    public PrimaryIndex(float record, float blockSize, float recordSize, float vssn, float
    blockPointer) {
        this.record = record;
        this.blockSize = blockSize;
        this.recordSize = recordSize;
        this.Vssn = vssn;
        this.blockPointer = blockPointer;
    }

    // main progress
    public void mainProgress() {
        bfr = (float) Math.floor(blockSize/recordSize); //Blocking factor
        b = (float) Math.ceil(record/bfr); // Number of Block needed for the file
        Ri = Vssn + blockPointer; // The size of each index Entry
        bfri = (float) Math.floor(blockSize/Ri); // The blocking factor for the index
        ri = b; // The total number of index entries
        bi = (float) Math.ceil(ri/bfri); // The number of index blocks is hence
        binaraySearchDataFile = (float) Math.ceil(Math.log10(b)/Math.log10(2)); // A binary search
on the data file
        linearSearchDataFile = (float) Math.ceil(b/2); // A Linear search on the data file
        binarySearchIndexFile = (float) Math.ceil(Math.log10(bi)/Math.log10(2)); // A binary search
on the index file
        binraySearchWithSupportIndex = binarySearchIndexFile + 1;

        System.out.println("Đáp án task 1!");
        System.out.printf("1. What is the blocking factor value of the data file? Kết quả = %.0f\n",
bfr);
        System.out.printf("2. How many data blocks are there for the data file? Kết quả = %.0f\n",
b);
        System.out.printf("3. How many block accesses for a linear search on the data file? Kết quả
= %.0f\n", linearSearchDataFile);
        System.out.printf("4. How many block accesses for a binary search on the data file? Kết quả
= %.0f\n", binaraySearchDataFile);
        System.out.printf("5. How many block accesses for a binary search on the index file? Kết quả
= %.0f\n", binarySearchIndexFile);
        System.out.printf("6. How many block accesses for a binary search with the support of
Primary Index? Kết quả = %.0f\n\n", binraySearchWithSupportIndex);
    }
}
```

3.1.4 Class PrimaryIndex

Dữ liệu mà ta nhận được sẽ được truyền vào trong class PrimaryIndex này, và trong đó hàm **mainProgress()** sẽ đóng vai trò là hàm xử lý chính cho cả chương trình, nó nhận dữ liệu được truyền vào và xử lý, sau khi xử lý xong kết quả sẽ được xuất ra màn hình ngay lập tức. Ngoài ra, công thức của Primary Index được xử lý đúng theo dạng bài tập của nó.

## ❖ Bài 2

Đối với bài tập 2 trong phụ lục, ta có mã nguồn như sau:

```
package Feature;

public class ClusteringIndex {
    private float record, blockSize, recordSize, Vssn, blockPointer, zipCode;
    private float bfr, b, Ri, bfri, ri, bi, binaraySearchDataFile, binarySearchIndexFile,
    binraySearchWithSupportIndex, linearSearchDataFile; // output value

    public ClusteringIndex(float record, float blockSize, float recordSize, float vssn, float
    blockPointer, float zipCode) {
        this.record = record;
        this.blockSize = blockSize;
        this.recordSize = recordSize;
        this.Vssn = vssn;
        this.blockPointer = blockPointer;
        this.zipCode = zipCode;
    }

    // main progress
    public void mainProgress() {
        bfr = (float) Math.floor(blockSize/recordSize); //Blocking factor
        b = (float) Math.ceil(record/bfr); // Number of Block needed for the file
        Ri = Vssn + blockPointer; // The size of each index Entry
        bfri = (float) Math.floor(blockSize/Ri); // The blocking factor for the index
        ri = zipCode; // The total number of index entries
        bi = (float) Math.ceil(ri/bfri); // The number of index blocks is hence
        binaraySearchDataFile = (float) Math.ceil(Math.log10(b)/Math.log10(2)); // A binary search
on the data file
        linearSearchDataFile = (float) Math.ceil(b/2); // A Linear search on the data file
        binarySearchIndexFile = (float) Math.ceil(Math.log10(bi)/Math.log10(2)); // A binary search
on the index file
        binraySearchWithSupportIndex = binarySearchIndexFile + 1;

        System.out.println("Đáp án task 2!");
        System.out.printf("1. What is the blocking factor value of the data file? Kết quả = %.0f\n",
bfr);
        System.out.printf("2. How many data blocks are there for the data file? Kết quả = %.0f\n",
b);
        System.out.printf("3. How many block accesses for a linear search on the data file? Kết quả
= %.0f\n", linearSearchDataFile);
        System.out.printf("4. How many block accesses for a binary search on the data file? Kết quả
= %.0f\n", binaraySearchDataFile);
        System.out.printf("5. How many block accesses for a binary search on the index file? Kết quả
= %.0f\n", binarySearchIndexFile);
        System.out.printf("6. How many block accesses for a binary search with the support of
```

```
Clustering Index? Kết quả = %.0f\n\n", binraySearchWithSupportIndex);
    }
}
```

### 3.1.5 Class ClusteringIndex

Đối với trường hợp của ClusteringIndex, ta thấy đối số bây giờ có thêm tham số **zipCode** và công thức tính toán đã thay đổi của **Total index entries** từ  $ri = b$  sang  $ri = \text{zipCode}$ .

## ❖ Bài 3

Đối với bài tập 3 trong phụ lục, ta có mã nguồn như sau:

```
package Feature;

public class SecondaryIndex {
    private float record, blockSize, recordSize, Vssn, blockPointer;
    private float bfr, b, Ri, bfri, ri, bi, binaraySearchDataFile, binarySearchIndexFile,
    binraySearchWithSupportIndex, linearSearchDataFile; // output value

    public SecondaryIndex(float record, float blockSize, float recordSize, float vssn, float
    blockPointer) {
        this.record = record;
        this.blockSize = blockSize;
        this.recordSize = recordSize;
        this.Vssn = vssn;
        this.blockPointer = blockPointer;
    }

    // main progress
    public void mainProgress() {
        bfr = (float) Math.floor(blockSize/recordSize); //Blocking factor
        b = (float) Math.ceil(record/bfr); // Number of Block needed for the file
        Ri = Vssn + blockPointer; // The size of each index Entry
        bfri = (float) Math.floor(blockSize/Ri); // The blocking factor for the index
        ri = record; // The total number of index entries
        bi = (float) Math.ceil(ri/bfri); // The number of index blocks is hence
        binaraySearchDataFile = (float) Math.ceil(Math.log10(b)/Math.log10(2)); // A binary search
on the data file
        linearSearchDataFile = (float) Math.ceil(b/2); // A Linear search on the data file
        binarySearchIndexFile = (float) Math.ceil(Math.log10(bi)/Math.log10(2)); // A binary search
on the index file
        binraySearchWithSupportIndex = binarySearchIndexFile + 1;

        System.out.println("Đáp án task 3!");
        System.out.printf("1. What is the blocking factor value of the data file? Kết quả = %.0f\n",
bfr);
        System.out.printf("2. How many data blocks are there for the data file? Kết quả = %.0f\n",
b);
        System.out.printf("3. How many block accesses for a linear search on the data file? Kết quả
= %.0f\n", linearSearchDataFile);
        System.out.printf("4. How many block accesses for a binary search on the data file? Kết quả
= %.0f\n", binaraySearchDataFile);
        System.out.printf("5. How many block accesses for a binary search on the index file? Kết quả
= %.0f\n", binarySearchIndexFile);
    }
}
```



```

        System.out.printf("6. How many block accesses for a binary search with the support of
Secondary Index? Kết quả = %.0f\n\n", binraySearchWithSupportIndex);
    }
}

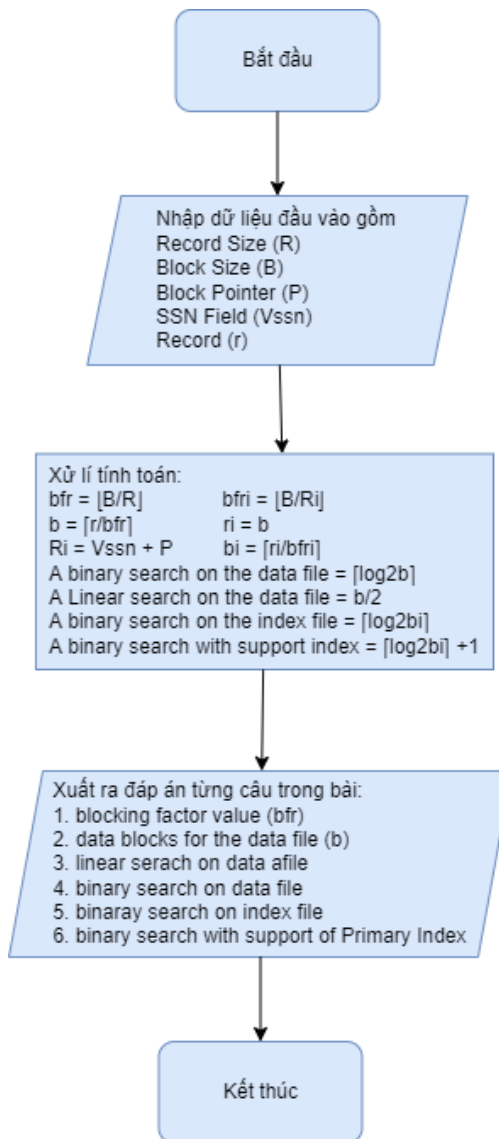
```

### 3.1.6 Class SecondaryIndex

Đối với trường hợp của SecondaryIndex thì đối số không khác gì với PrimaryIndex, nhưng lại có sự thay đổi công thức tính toán của **Total index entries** từ  $ri = b$  sang  $ri = r$  (record).

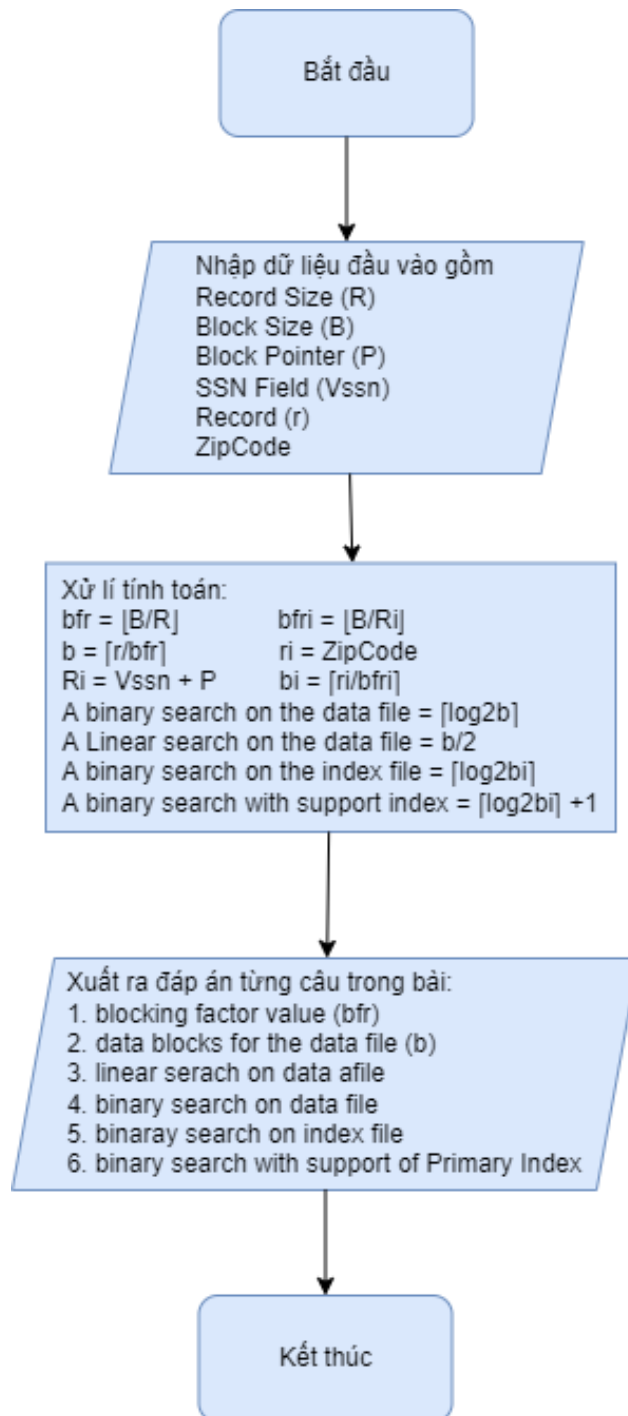
## 3.2 Các lưu đồ thuật toán

### ❖ Bài 1



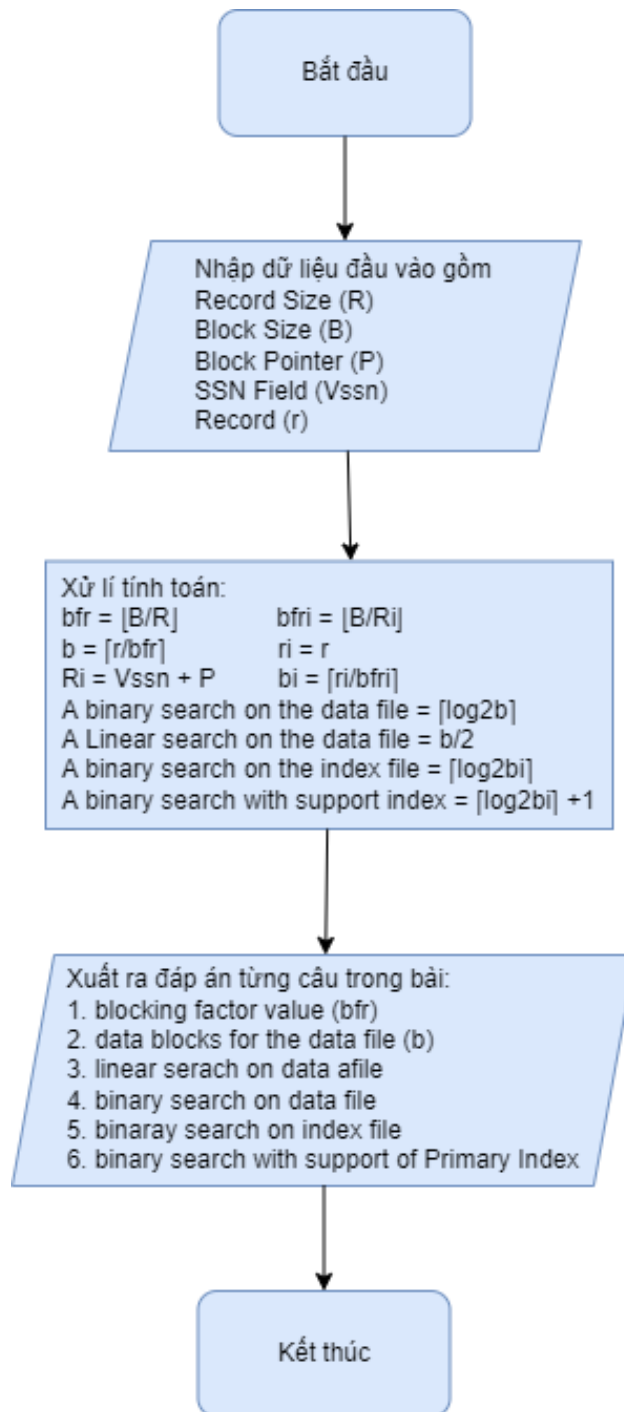
#### 3.2.1 Lưu đồ thuật toán xử lý PrimaryIndex

## ❖ Bài 2



3.2.2 Lưu đồ thuật toán xử lý ClusteringIndex

### ❖ Bài 3



3.2.3 Lưu đồ thuật toán xử lý SecondaryIndex

## 4. KẾT QUẢ BÀI TOÁN

### 4.1 Kết quả tính toán bằng Excel

Để minh họa kết quả của cách chương trình trên xử lý như đã được trình bày trên, ta có một bảng tính Excel xử lý các trường hợp và kết quả của từng trường hợp đó:

#### ❖ Bảng kết quả đầu vào input của tất cả bài toán

Record Size (R)	Block Size (B)	Block Pointer (P)	
40	4096	6	Primary Index
40	4096	6	Clustering Index
40	4096	6	Secondary Index
SSN Field (Vssn)	Record (r)	Zipcode	
9	600000	0	Primary Index
9	600000	500	Clustering Index
9	600000	0	Secondary Index

#### 4.1.1 Lưu đồ thuật toán xử lý SecondaryIndex

#### ❖ Bảng kết quả đầu ra output của cả tất cả bài toán

	Blocking factor (bfr = $\lceil B/R \rceil$ )	Number of Block needed for the file (b = $\lceil r/bfr \rceil$ )		
Primary Index	102	5883		
Clustering Index	102	5883		
Secondary Index	102	5883		
	The size of each index Entry ( $R_i = V_{ssn} + P$ )	The blocking factor for the index ( $bfr_i = \lceil B/R_i \rceil$ )	The total number of index entries ( $ri = b$ )	The number of index blocks is hence ( $bi = \lceil ri/bfr_i \rceil$ )
Primary Index	15	273	5883	22
Clustering Index	15	273	500	2
Secondary Index	15	273	600000	2198
	A Linear search on the data file = $b/2$	A binary search on the data file = $\lceil \log_2 b \rceil + 1$	A binary search on the index file = $\lceil \log_2 bi \rceil + 1$	A binary search with support index = $\lceil \log_2 bi \rceil + 1$
Primary Index	2942	13	5	6
Clustering Index	2942	13	1	2
Secondary Index	2942	13	12	13

#### 4.1.2 Lưu đồ thuật toán xử lý SecondaryIndex

### 4.2 Kết quả tính toán bằng chương trình (Kiểm thử chương trình)

Sau khi hoàn tất quá trình giải quyết bài toán bằng lời giải lý thuyết và kết quả thu được từ việc tính toán từ bảng excel, ta tiếp tục kiểm thử chương trình và thực thi các bài toán trong phần phụ lục.

Quá trình và kết quả thực thi được biểu diễn như sau:

#### ❖ Bài tập 1

Ta nhập vào input chương trình như sau:

```
Record_Size_(R):40
Block_Size_(B):4096
Block_Pointer_(P):6
SSN_Field_(Vssn):9
Number_of_Record (r):600000
```

#### 4.1 Đổi số đầu vào của bài tập 1

Sau khi chương trình thực hiện và ta được kết quả như sau:

```
Đáp án task 1!
1. What is the blocking factor value of the data file? Kết quả = 102
2. How many data blocks are there for the data file? Kết quả = 5883
3. How many block accesses for a linear search on the data file? Kết quả = 2942
4. How many block accesses for a binary search on the data file? Kết quả = 13
5. How many block accesses for a binary search on the index file? Kết quả = 5
6. How many block accesses for a binary search with the support of Primary Index? Kết quả = 6
```

#### 4.2 Kết quả đáp án bài tập 1

### ❖ Bài tập 2

Ta nhập vào input chương trình như sau:

```
Record_Size_(R):40
Block_Size_(B):4096
Block_Pointer_(P):6
SSN_Field_(Vssn):9
Number_of_Record (r):600000
Zipcode:500
```

#### 4.3 Đổi số đầu vào của bài tập 2

Sau khi chương trình thực hiện, ta nhận được kết quả như sau:

Đáp án task 2!

1. What is the blocking factor value of the data file? Kết quả = 102
2. How many data blocks are there for the data file? Kết quả = 5883
3. How many block accesses for a linear search on the data file? Kết quả = 2942
4. How many block accesses for a binary search on the data file? Kết quả = 13
5. How many block accesses for a binary search on the index file? Kết quả = 1
6. How many block accesses for a binary search with the support of Clustering Index? Kết quả = 2

#### 4.4 Kết quả đáp án bài tập 2

### ❖ Bài tập 3

Ta nhập vào input chương trình như sau:

```
Record_Size_(R):40
Block_Size_(B):4096
Block_Pointer_(P):6
SSN_Field_(Vssn):9
Number_of_Record (r):600000
```

#### 4.5 Đối số đầu vào của bài tập 3

Sau khi chương trình thực hiện và ta được kết quả như sau:

Đáp án task 3!

1. What is the blocking factor value of the data file? Kết quả = 102
2. How many data blocks are there for the data file? Kết quả = 5883
3. How many block accesses for a linear search on the data file? Kết quả = 2942
4. How many block accesses for a binary search on the data file? Kết quả = 13
5. How many block accesses for a binary search on the index file? Kết quả = 12
6. How many block accesses for a binary search with the support of Secondary Index? Kết quả = 13

#### 4.6 Kết quả đáp án bài tập 3

## 5. TÀI LIỆU THAM KHẢO

- Sách, Slide bài giảng:

- ❖ Chapter 18. Indexing Structures for Files
- ❖ Chapter 2. Indexing Structures for Files (Slide thầy Đăng)

- Website:

- ❖ Chapter 11. File Organisation and Indexes.

[https://www.cs.uct.ac.za/mit\\_notes/database/htmls/chp11.html?fbclid=IwAR1n1ebfQa9HH3IIU241CaApp8ggPQb4nnwqt\\_e-PVyIQMNC7gwCumCzlBg#primary-indexes](https://www.cs.uct.ac.za/mit_notes/database/htmls/chp11.html?fbclid=IwAR1n1ebfQa9HH3IIU241CaApp8ggPQb4nnwqt_e-PVyIQMNC7gwCumCzlBg#primary-indexes)

- ❖ Primary Index là gì

[Primary index là gì? - Từ điển CNTT \(dictionary4it.com\)](#)

- ❖ VTI TechBlog – Sử dụng index trong MySQL

[Sử dụng Index trong MySQL: Phần 1- Các loại index và cách đánh index - VTI TechBlog!](#)