Truy vấn dữ liệu chuyển động

# MỤC LỤC

[Mục lục 1](#_Toc466585982)

[Danh mục hình vẽ 2](#_Toc466585983)

[1. Mô hình hệ thống 3](#_Toc466585984)

[2. Mô hình khai thác dữ liệu chuyển động 4](#_Toc466585985)

[2.1. Mô hình khai thác dữ liệu chuyển động 4](#_Toc466585986)

[2.2. Một số mẫu truy vấn thường dùng trong xử lý bài toán giám sát hành trình xe vận tải 5](#_Toc466585987)

[2.3. Giải pháp chuyển sang cơ sở dữ liệu SECONDO 7](#_Toc466585988)

[3. Triển khai hệ thống 9](#_Toc466585989)

[3.1. Các bước triển khai hệ thống 9](#_Toc466585990)

[3.2. Cài đặt Secondo trên Ubuntu 14.04 9](#_Toc466585991)

[3.3. Cấu hình Secondo lần đầu trong Hệ điều hành Ubuntu 14.04 10](#_Toc466585992)

[3.4. Kết nối với hệ thống Secondo 11](#_Toc466585993)

[4. Kết luận 12](#_Toc466585994)

# Danh mục hình vẽ

[Hình 1. Mô hình hệ thống truy vẫn dữ liệu chuyển động 3](#_Toc466586087)

[Hình 2. Mô hình khai thác dữ liệu chuyển động 4](#_Toc466586088)

# Mô hình hệ thống



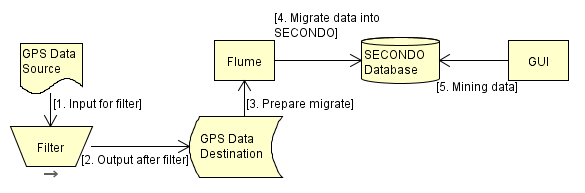
*Hình 1*. *Mô hình hệ thống truy vẫn dữ liệu chuyển động*

Hệ thống vấn CSDL chuyển động bao gồm các thành phần được mô tả như hình 1:

* Center Server: trung tâm tính toán xử lý dữ liệu chuyển động, cung cấp khả năng truy vấn, hiển thị trực quan cho người dùng
* Database Server: máy chủ lưu trữ dữ liệu chuyển động của hệ thống, các dữ liệu được lưu trữ bao gồm vị trí, thời gian, thông tin lái xe… Các dữ liệu này trước khi được đưa vào hệ thống đã được tiền xử lý từ nguồn dữ liệu thô (raw data).
* Realtime datastore: nguồn dữ liệu thời gian thực, là nguồn dữ liệu thô (raw data) được tổng hợp từ các thiết bị GPS, dữ liệu người dùng, dữ liệu hành trình thực tế…

# Mô hình khai thác dữ liệu chuyển động

## Mô hình khai thác dữ liệu chuyển động



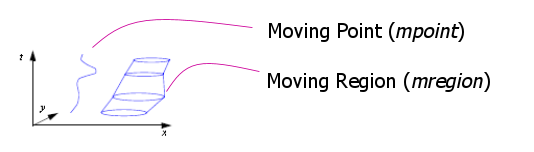
*Hình 2. Mô hình khai thác dữ liệu chuyển động*

Định nghĩa các thành phần :

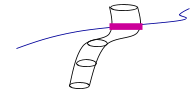
* *GPS Data Source*: Đầu vào là nguồn dữ liệu hành trình ghi lại được của xe vận tải. Nguồn dữ liệu này được lưu trữ ở dạng text file, định đạng theo từng dòng và cột dữ liệu.
* *Filter*: Bộ lọc dữ liệu. Bộ lọc dữ liệu được xây dựng trên mã nguồn JAVA, có chức năng lọc nhiễu dữ liệu. Đầu ra của bộ lọc là nguồn dữ liệu chuẩn hay còn gọi là GPS Data Destination.
* *GPS Data Destination*: Dữ liệu hành trình đã được lọc nhiễu.
* *Flume*: Bộ chuyển đổi từ dữ liệu hành trình dạng “text” phi cấu trúc sang cơ sở dữ liệu SECONDO.
* *SECONDO Database*: Hệ quản trị cơ sở dữ liệu SECONDO, đã tích hợp công cụ dùng cho khai phá dữ liệu.
* *GUI*: Một giao diện người dùng kết nối đến cơ sở dữ liệu SECONDO, thực hiện các truy vấn khai phá dữ liệu và hiển thị kết quả thu được.

## Một số mẫu truy vấn thường dùng trong xử lý bài toán giám sát hành trình xe vận tải

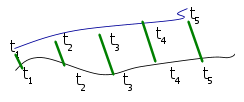
Ta có một số khái niệm sau:



* Hai kiểu dữ liệu mpoint và mregion đều là những ánh xạ từ thời gian đến không gian:
  + Mpoint = time 🡪 point
  + Mregion = time 🡪 region
* Ví dụ:
  + flight (id: string, origin: string, dest: string, route: mpoint)
  + weather (id: string, kind: string, area: mregion)
* Các kiểu dữ liệu đó sẽ mô tả các hoạt động tương ứng như sau:
  + Intersection (mpoint, mregion) → mpoint



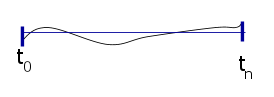
* + distance (mpoint, mpoint) → mreal



* + Trajectory (mpoint) → line



* + Deftime(mpoint) → period



* + length (line) → real
  + flight (id: string, origin: string, dest: string, route: mpoint)
* Một số mẫu truy vấn thường dùng:
  + Truy vấn 1: "Tìm tất cả các chuyến bay từ Sao Paulo dài hơn 4000 km."

SELECT \*

FROM flights

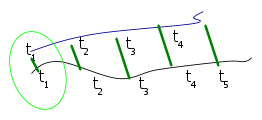
WHERE origin = ’SP’ AND lenght(trajectory(route)) > 4000

* + Query 2: "Lấy bất kỳ cặp máy bay mà trong chuyến bay của họ đến gần hơn với nhau hơn 500 mét! "

SELECT f.id, g.id

FROM flights f, flights g

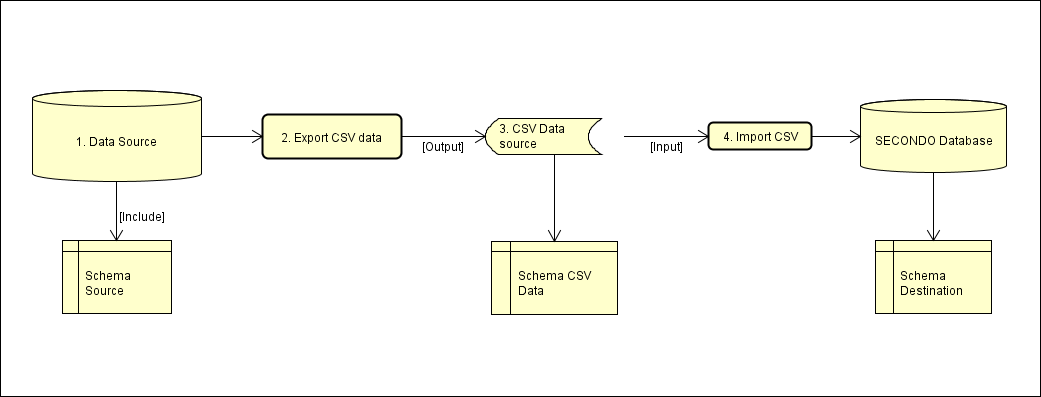
WHERE f.id <> g.id AND min(distance(f.route, g.route)) < 0.5



## Giải pháp chuyển sang cơ sở dữ liệu SECONDO

SECONDO hỗ trợ nhập dữ liệu đầu vào với định dạng file CSV phổ biến. Do vậy, đây là giải pháp hỗ trợ việc chuyển đổi từ nhiều cơ sở dữ liệu khác nhau sang cơ sở dữ liệu SECONDO một cách đơn giản. Module chuyển đổi dữ liệu này ta có thể gọi là Flume.

Dưới đây là giải pháp mà Flume đã sử dụng để chuyển đổi dữ liệu:



*Hình 3. Chuyển đổi CSDL Secondo*

Mô tả các thành phần:

* *Data Source* : Nguồn dữ liệu cần chuyển đổi, có thể là các bảng dữ liệu trong cơ sở dữ liệu quan hệ hoặc các định dạng file dữ liệu có cấu trúc như CSV, JSON hay phi cấu trúc
* *Export CSV data* : Phần công hỗ trợ hoặc tự xây dựng giúp đẩy dữ liệu nguồn ra tệp file dữ liệu ngoài với định dạng CSV.
* *CSV Data souce* : Tệp dữ liệu nguồn ở định dạng CSV
* *Import CSV* : Công cụ hỗ trợ của SECONDO giúp đẩy dữ liệu từ tệp (định dạng CSV) vào một bảng dữ liệu hay còn gọi là schema
* *SECONDO Database*: Loại cơ sở dữ liệu cần chuyển đổi thành
* *Schema Souce*: Bảng dữ liệu cần chuyển đổi
* *Schema CSV Data*: Cấu trúc dữ liệu trên tệp CSV, tương ứng với cấu trúc của Schema Source
* *Schema Destination*: Cấu trúc bảng dữ liệu trong cơ sở dữ liệu SECONDO, tương ứng với cấu trúc của Schema CSV Data

Ví dụ:

* Dữ liệu CSV hành trình di chuyển của xe vận tải

|  |
| --- |
| #GPS Track File generated by KDR GPS Tracker Version 2.69  #Name:binhanh  #track param.: t=015,v=010,c=010,d=200,NAOAO  #each data set holds:  #Latitude in Deg.Decimals, (neg.=south)  #Longitude in Deg.Decimals, (neg.=west)  #UTC(hh:mm:ss),Speed(km/h)  21.109756,105.878151,00:00:08,0  21.109743,105.878151,05:22:43,5  21.109665,105.878044,05:22:48,9  21.10955,105.878021,05:22:53,7  21.109472,105.877945,05:22:58,9  21.109495,105.877808,05:23:03,9 |

* Định dạng schema tương ứng trên cơ sở dữ liệu SECONDO

|  |
| --- |
| let BinhAnhVehicle = [const rel(tuple([  Latitude: real,  Longitude: real,  UTC: string,  Speed: int  ]))  value ()] |

* Đẩy dữ liệu từ tệp CSV vào cơ sở dữ liệu SECONDO

|  |
| --- |
| let Raw = BinhAnhVehicle csvimport['../bin/binhanh.csv', 7, "#", ","] consume |

# Triển khai hệ thống

## Các bước triển khai hệ thống

* Cài đặt Apache Tomcat
* Cài đặt Secondo
  + Cài đặt Secondo
  + Cấu hình Secondo
* Kết nối ứng dụng Geospacialquery với hệ thống Secondo

## Cài đặt Secondo trên Ubuntu 14.04

* Cài Ubuntu 14.04 từ đĩa cài đặt vào đĩa cứng và bắt đầu nó.
* Tải xuống tập tin InstallSDK\_Ubuntu\_14\_04.bash và lưu trữ nó vào thư mục ***home***.
* Mở cửa sổ ứng dụng Terminal.
* Thi hành script: ***bash InstallSDK\_Ubuntu\_14\_04.bash***. Quá trình chạy sẽ yêu cầu mật khẩu của user cho việc cài đặt công cụ cần thiết.
* Chèn dòng sau đây vào cuối tập tin ***~/.bashrc***

*source $HOME/.secondorc $HOME/secondo*

* Trong Terminal gõ dòng lệnh:

*cd $HOME wget http://dna.fernuni-hagen.de/Secondo.html/files/secondo-v340-HalloweenEdition-LAT1.tar.gz tar - xzf secondo-v340-LAT1.tar.gz*

* Mở thêm cửa sổ Terminal.
* Chuyển vào thư mục ***$~/secondo*** và gõ lệnh ***make*** để biên dịch chương trình.

## Cấu hình Secondo lần đầu trong Hệ điều hành Ubuntu 14.04

Trong Ubuntu 14.04 chứa Secondo 3.40:

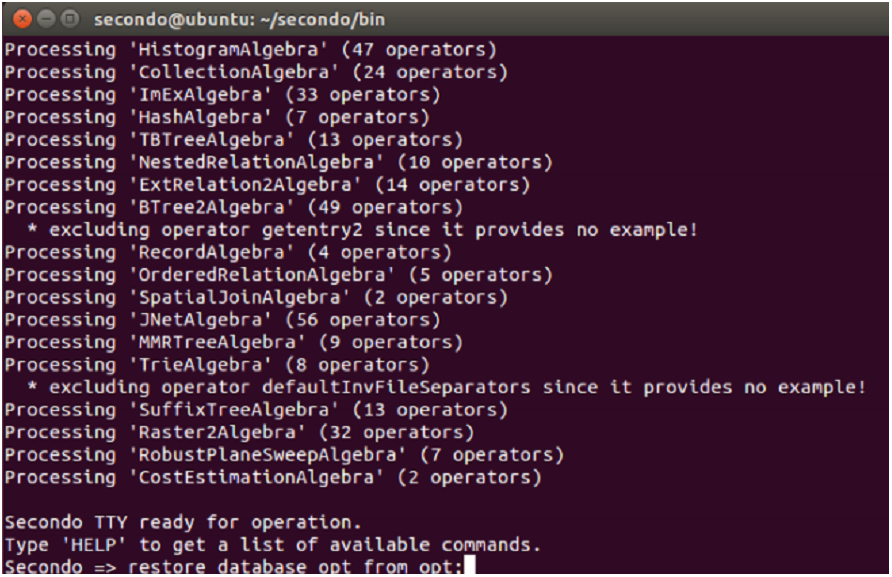
* Mở terminal (Ctrl+Alt+T), thực hiện các lệnh :

*cd secondo*

*make*

*cd bin*

*SecondoTTYBDB*



*Hình 4. Một số lệnh khởi tạo hệ thống Secondo*

## *Kết nối với hệ* thống *Secondo*

SECONDO hỗ trợ giao thức JDBC cho các nhà phát triển ứng dụng phần mềm có thể kết nối và sử dựng truy vấn dữ liệu. Dưới đây là từng bước để thiết lập kết nối bằng giao thức (driver) JDBC đến cơ sở dữ liệu SECONDO:

* Cấu hình kết nối dữ liệu

|  |
| --- |
| Driver-Name: DriverSet.Driver  JDBC-URL: jdbc:secondo:  IP-Address: 127.0.0.1  Secondo-Port: 1234  Optimizer-Port: 1235  Database: test\_queries\_secondo |

* Thông số kết nối đến cơ sở dữ liệu được lưu tại file cấu hình hệ thống ứng dụng
* Thiết lập kết nối

|  |
| --- |
| SepDriverName = parametersToConnect.indexOf('|');  DriverName=parametersToConnect.substring(0, SepDriverName);  parametersToConnect=parametersToConnect.substring(SepDriverName+1);  con=DriverManager.getConnection(parametersToConnect); |

# Kết luận

Hệ thống đã đáp ứng được yêu cầu đặt ra hỗ trợ truy vấn dữ liệu chuyển động của các đối tượng trong thời gian thực. Cùng với việc hệ thống Secondo là một hệ thống mở, cho phép nâng cấp, đưa thêm các tính năng theo yêu cầu của người dùng, hệ thống có thể mở rộng cho các yêu cầu thực tiễn sử dụng hay các yêu cầu về nghiên cứu chuyên sâu.

# Refference

**SecondoConnector.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package com.tungvu;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

import java.io.File;

import java.io.IOException;

import java.io.OutputStream;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.Vector;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

/\*\*

\*

\* @author TungVu

\*/

public class SecondoConnector {

// This field sets the name of the file where Secondo() method must store the

// result when the commandAsText parameter is set to true. It is predefined

// to be "SecondoResult".

private static final String SECONDO\_RESULT\_FILE = "SecondoResult";

// Flag indicating if a connection with a Secondo server exists.

protected static boolean initialized;

// Socket connected to the Secondo Server

protected static Socket serverSocket;

// flag for binary reading lists from Server

private static boolean binaryLists = false;

// Connection's input reader (from secondoServerSocket).

private static MyDataInputStream inSocketStream;

// Connection's output writer (to secondoServerSocket).

private static MyBufferedOutputStream outSocketStream;

public static boolean initialize(String user, String pswd,

String host, int port) {

String secHost = host;

int secPort = port;

String line;

if (!initialized) {

Reporter.writeInfo("Initializing the Secondo system ...");

// Connect with server, needed host and port

if (secHost.length() > 0 && secPort > 0) {

Reporter.writeInfo("SecondoInterface: Connecting with Secondo server '"

+ secHost + "' on port " + secPort + " ...");

try {

serverSocket = new Socket(secHost, secPort);

try {

// Creates the input/output streams.

inSocketStream = new MyDataInputStream(new BufferedInputStream(serverSocket.getInputStream()));

outSocketStream = new MyBufferedOutputStream((serverSocket.getOutputStream()));

} catch (IOException e) {

serverSocket = null; // To help detecting internal errors.

Reporter.writeError("SecondoInterface: IOError creating input/output streams for serverSocket..");

}

} catch (UnknownHostException e) {

Reporter.writeError("SecondoInterface: Network error: Unknown host '" + secHost + "'.");

} catch (IOException e) {

Reporter.writeError("SecondoInterface: Network error: Unable to connect with the Secondo Server.");

}

if (serverSocket != null) {

try {

line = inSocketStream.readLine();

if (line == null) {

Reporter.writeError("Could not read data from server.");

initialized = false;

return false;

}

if (line.equals("<SecondoOk/>")) {

outSocketStream.write("<Connect>\n"

+ user + "\n"

+ pswd + "\n"

+ "</Connect>\n");

outSocketStream.flush();

line = inSocketStream.readLine();

if (line == null) {

initialized = false;

Reporter.writeError("Could not read data from server.");

return false;

}

if (line.equals("<SecondoIntro>")) {

do {

line = inSocketStream.readLine();

if (line == null) {

initialized = false;

Reporter.writeError("Could not read data from server.");

return false;

}

if (!line.equals("</SecondoIntro>")) {

Reporter.writeInfo(line);

}

} while (!line.equals("</SecondoIntro>"));

initialized = true;

} else if (line.equals("<SecondoError>")) {

initialized = false;

do {

line = inSocketStream.readLine();

if (line == null) {

initialized = false;

Reporter.writeError("SecondoError, connection lost");

}

Reporter.writeError("Server-Error: " + line);

} while (line != null && !line.equals("</SecondoError>"));

} else {

Reporter.writeError("Unidentifiable response from server: " + line);

}

} else {

Reporter.writeError("Unidentifiable response from server: " + line);

}

} catch (IOException e) {

Reporter.debug(e);

initialized = false;

}

} else {

Reporter.writeError("Connection to SecondoServer could not be established.");

initialized = false;

}

if (!initialized && serverSocket != null) {

serverSocket = null;

}

} else {

Reporter.writeError("\*\*\* SecondoInterface: Invalid or missing host (" + secHost

+ ") and/or port (" + secPort + ").");

}

}

return (initialized);

}

private static class MyBufferedOutputStream extends BufferedOutputStream {

public MyBufferedOutputStream(OutputStream out) {

super(out);

}

public void write(String s) throws IOException {

if (Environment.TRACE\_SERVER\_COMMANDS) {

Reporter.write("Send to Server: \"" + s + "\"");

}

if (Environment.ENCODING != null) {

try {

super.write(s.getBytes(Environment.ENCODING));

} catch (Exception e) {

System.err.println("invalid encoding found, check environment");

System.exit(0);

}

} else {

super.write(s.getBytes());

}

flush();

}

}

public static void secondo(String command,

ListExpr resultList,

IntByReference errorCode,

IntByReference errorPos,

StringBuffer errorMessage) {

// write command to console if desired

if (Environment.SHOW\_COMMAND) {

Reporter.write(command);

}

// clean the errormessage

errorMessage.setLength(0);

errorCode.value = 0;

resultList.setValueTo(ListExpr.theEmptyList());

// check for existing connection

if (serverSocket == null) {

errorCode.value = 80;

return;

}

// check for restore command

command = command.trim();

try {

// not a special command

int commandLevel = command.startsWith("(") ? 0 : 1;

outSocketStream.write("<Secondo>\n");

outSocketStream.write(commandLevel + "\n");

outSocketStream.write(command);

outSocketStream.write("\n</Secondo>\n");

outSocketStream.flush();

receiveResponse(resultList, errorCode, errorPos, errorMessage);

} catch (IOException e) {

errorCode.value = 81;

return;

}

}

}