SECONDO

A Database System for Moving Objects[*]

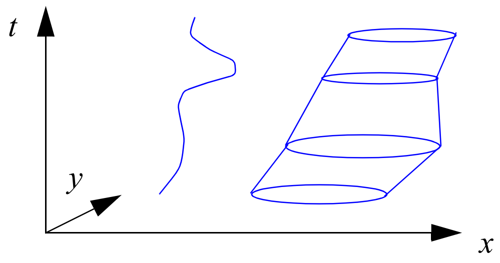
August 17, 2011, revised October 14, 2016

SECONDO was built by the *SECONDO team*.

Outline:

1 Overview

SECONDO is one of the first database system prototypes that can handle moving objects, that is, continuously changing time-dependent geometries. More precisely, complete histories of movement can be represented in a database and queried. The basic idea is to use spatio-temporal data types such as *moving point* and *moving region*. For example, a complete trip of a vehicle can be represented as a single value of the type *moving point* (*mpoint* for short), and the development of a forest fire over time can be represented as a single value ot the type *moving region* (*mregion*). Essentially, a moving point is a function from time into point values and a moving region a function from time into region values; they can be visualized as shown below.



[*] The development of SECONDO, in particular the part concerning the management of moving objects, was supported by grant Gu 293/8-2 from the Deutsche Forschungsgemeinschaft (DFG), project “Datenbanken für bewegte Objekte” (Databases for Moving Objects).

Together with the data types, suitable operations for querying and data manipulation are provided. For example:

operation	signature	meaning
trajectory	$mpoint \rightarrow line$	compute the projection of a moving point into the 2D plane
distance	$mpoint \times point \rightarrow mreal$	compute the time dependent distance between a moving point and a static point.. Result is of data type moving real (*mreal*)
passes	$mpoint \times region \rightarrow bool$	check whether a moving point passes a given area in the plane
inside	$mpoint \times mregion \rightarrow mbool$	Returns a time dependent boolean value (type *mbool*) indicating when the moving point is inside the moving region.

This approach to modeling and querying histories of moving objects is described in papers [1-4] and in the text book [5]. SECONDO implements a considerable part of the design described in those papers. See the *Publications* page to find more information about SECONDO.

2 Quick Start - Trying It

In the sequel we describe how you can perform relatively quickly a “do it yourself demo” of the moving objects DBMS without reading all the manuals and papers first.

2.1 Installation

If you do not yet have a SECONDO installation, go to the *Installation Instructions* page and select an installation suitable for your platform (Linux, Mac-OS X).

If necessary, get from the *Sources* page the latest SECONDO version and build it according to the instructions. The current version is SECONDO 4.0. This step is not needed for the virtual machine appliance or the Ubuntu Binary Installations.

2.2 Running SECONDO

2.2.1 Restoring Databases

1. Now a runnable SECONDO system should be available. Open a shell window. Change to the directory secondo/bin:

```
cd secondo/bin
```

In the Ubuntu Binary Installation, this step is omitted as one can start the SECONDO user interfaces from any directory.

2. Type

```
SecondoTTYBDB
```

3. This starts a single user version of SECONDO. We use it to restore some example databases:

```
restore database opt from opt;
close database;

restore database berlintest from berlintest;
close database;
q
```

In the Ubuntu Binary Installation, the files from which databases are restored are located in the directory /opt/secondo/bin, therefore the respective commands need to be:

```
restore database opt from '/opt/secondo/bin/opt';
restore database berlintest from '/opt/secondo/bin/berlintest';
```

Note that at this user interface a command is terminated either by a semicolon or an empty line. The last command quits SecondoTTYBDB.

2.2.2 Using Kernel and GUI

4. Still in the same directory, type

```
SecondoMonitor -s
```

This starts a process listening for user interfaces to register with the kernel.

5. Open a new shell. Change to the directory secondo/Javagui (not needed in Ubuntu Binary Installation)

6. Type

```
sgui
```

This starts the graphical user interface of SECONDO. A red message appears “Error: error in enabling optimizer”. Don’t worry, SECONDO works fine with or without query optimizer. For the moment we use it without optimizer.

A popup window appears: “Errors in loading favoured queries.” This happens only on using the GUI for the first time, because a file is not yet available.

7. At the command window in the GUI (upper left), type

```
open database berlintest
```

As you have guessed, this opens the test database “berlintest”.

8. From the drop-down menu “Viewers”, select “Hoese-Viewer”.

9. From the “File” menu, select “Load Categories”. In the pop-up window, select “BerlinU.cat” and open it. The viewer responds “categories loaded”. Categories are display styles for geometric attributes.

10. In the command window (upper left), type

    ```
    query UBahn
    ```

    A pop-up window appears asking for a display style for the attribute “geoData”. From the menu “View category”, select (style) “UBahn” and click “ok”. The underground network of the city of Berlin, Germany, appears. We will use it as a background for displaying moving underground trains.

    These are available in a relation “Trains” in the database with schema

    ```
    Trains(Id: int, Line: int, Up: bool, Trip: mpoint)
    ```

    There are 562 trains moving on a specific day 11/20/2003.

11. First, we use a single object “train7”, a value of type *mpoint*. This is the *mpoint* attribute of one of the trains, stored as a separate SECONDO object. Type

    ```
    query train7
    ```

    In the pop-up window, select style “QueryMPoint” and confirm. Train7 appears as a red square. Note the time displayed: 2003-11-20-06:06:00.000. Hence on November 20, 2006, it is six minutes after 6am, 0 seconds, 0 milliseconds.

    The small window below the graphics display shows the time interval when this object is defined. Remove this window (click into the arrow) to see the graphics better.

12. Use the animation buttons (forward) to see the object moving. Click on the double arrows to double or halve the speed of animation.

13. Now we wish to compute the path that train7 has taken. Type

    ```
    query trajectory(train7)
    ```

    and select display style “QueryLine”. Click in the graphics window to deselect. The path taken by train7 appears in red.

14. Let’s see a moving region. There are two objects called “mrain” and “msnow” of type *mregion*. Type, for example,

    ```
    query msnow
    ```

    to see one of them. Select an appropriate display style like “QueryRegionTransp” or simply the default style. Animation can also be done pulling the slider.

15. Compute the time dependent distance between train7 and a point (an underground station) called “mehringdamm”. First, type

    ```
    query mehringdamm
    ```

    A point appears in the display (style “QueryPoint” suggested). Now type

```
query distance(train7, mehringdamm)
```

A textual display “mreal : TA(MReal)” appears on the left. Double-click into this. A pop-up window appears showing the development of the distance over time, a value of type *moving real* (*mreal*). Since train7 passes through mehringdamm, the distance decreases to 0 and then increases again.

16. In the next query, we want to find all trains passing through a certain static region. The region is called “tiergarten”. First, let us see it:

    ```
    query tiergarten
    ```

    Then, say

    ```
    query Trains count
    ```

    This confirms that there are 562 trains in the Trains relation.

    ```
    query Trains feed filter[.Trip passes tiergarten] count
    ```

    There are 80 trains passing through tiergarten. Let us display them at the user interface:

    ```
    query Trains feed filter[.Trip passes tiergarten] consume
    ```

2.2.3 Using the Query Optimizer

17. Now let us add the query optimizer to the running system. Open a new shell window. Change to directory secondo/Optimizer (not needed in Ubuntu Binary Installation). Type

    ```
    StartOptServer
    ```

    The optimizer process starts and some messages appear.

18. At the GUI, from the “Optimizer” menu, select “Enable”. A message should appear “optimizer enabled”.

19. In the command window of the GUI, type

    ```
    select count(*) from Trains where Trip passes mehringdamm
    ```

    Because the optimizer is used for the first time, some initializations are done. After some time, we get a result of 122. By saying

    ```
    select * from Trains where Trip passes mehringdamm
    ```

    we can also see these trains.

20. The upper right window of the GUI contains query results. Click the “Clear” button to remove all the previous objects from the display. Reload the “UBahn” background as in step 10. Also reload “mehringdamm” as in step 15.

21. Now let us find all trains that exist at time 7:05 and that pass through mehringdamm, and let’s display their positions at this time. First we introduce an object to represent the instant of time 7:05 on this day.

```
let seven05 = theInstant(2003,11,20,7,5)
```

Then we formulate the query:

```
select [Id, Line, Up, val(Trip atinstant seven05) as Pos]
from Trains where [Trip passes mehringdamm, Trip present seven05]
```

22. Finally, let us find all trains getting into the moving snow area "msnow" and for each of them compute the part when it is inside msnow. First, empty the display and then reload UBahn and msnow. Select "QueryRegionTransp" as a display style for msnow.

```
select *
from Trains where [not(isempty(deftime(intersection(Trip, msnow))))]
```

This returns the complete movement of all trains passing through msnow. Select "QueryM-Point" as a display style.

```
select [Id, intersection(Trip, msnow) as Insnow]
from Trains where [not(isempty(deftime(intersection(Trip, msnow))))]
```

This returns only the respective part when the train is inside msnow. Select "QueryMPoint2" as a display style. As a result, the symbols representing the trains appear to change colour from red to blue when they enter the region msnow. This is because the blue square appears in front of the red square. Reduce the animation speed to see precisely what happens. One can also select the msnow object on the left and then pull a rectangle around it (holding the right mouse button) to zoom in. Because it is selected, the focus will follow the moving msnow object.

2.2.4 Final Hints

23. This concludes the demo. To get more information about the system and its algebras, types and operations, the following commands are helpful that you can type at the command interface in the GUI:

```
list algebras
list algebra TemporalAlgebra
```

You can also query to find a particular operator, for example, operator **atinstant**:

```
query SEC2OPERATORINFO feed filter[.Name contains "atinstant"] consume
```

24. Terminate SECONDO by `Program -> Exit` in the GUI, `quit` in the optimizer shell, and `quit` followed by `yes` to stop the SecondoMonitor (in the secondo/bin shell).

3 Papers, Book

1. M. Erwig, R.H. Güting, M. Schneider, and M. Vazirgiannis, Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. GeoInformatica 3:3 (1999), 269-296.

2. R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jensen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis, A Foundation for Representing and Querying Moving Objects. ACM Transactions on Database Systems 25:1 (2000), 1-42. *Paper*.
3. L. Forlizzi, R.H. Güting, E. Nardelli, and M. Schneider, A Data Model and Data Structures for Moving Objects Databases. Proc. ACM SIGMOD Conf. (Dallas, Texas) May 2000, 319-330. *Paper*.
4. J.A. Cotelo Lema, L. Forlizzi, R.H. Güting, E. Nardelli, and M. Schneider, Algorithms for Moving Objects Databases. The Computer Journal 46:6 (2003), 680-712. *Paper*.
5. R.H. Güting and M. Schneider, Moving Objects Databases. Morgan Kaufmann Publishers, 2005. *Book*, *Book Web Site*.