# DB Monitor Tutorial

Contents

[DB Monitor Tutorial 1](#_Toc389835649)

[Introduction 2](#_Toc389835650)

[Mirth ESB 3](#_Toc389835651)

[Calculating time taken of messages with Mirth ESB 4](#_Toc389835652)

[Using the time Taken of Mirth ESB in the DB Monitor 5](#_Toc389835653)

[How DB Monitor works 7](#_Toc389835654)

[Monitor 7](#_Toc389835655)

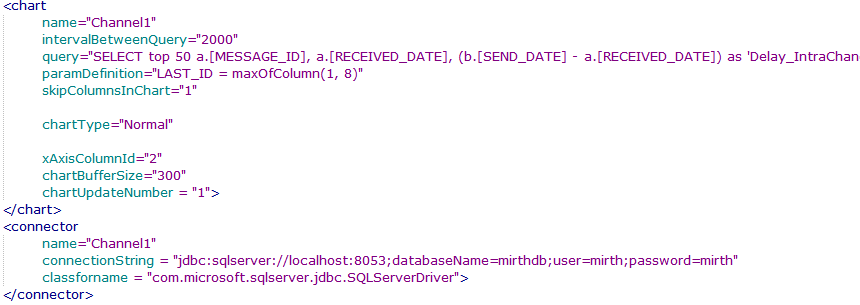
[Graficador 7](#_Toc389835656)

## Introduction

DB Monitor is a JAVA tool that queries each ‘n’ milliseconds a database and graphs the result values in a chart.



The tool has an XML configuration file that allows controlling the application.



Originally developed to monitor PostGreSQL statistical information, it was refactored and improved to allow monitoring Mirth Channels that are logged into a database.

## Mirth ESB

Mirth ESB allows logging information about the messages that arrive to each channel. Each channel creates a Table in the database that has the following content:

SELECT

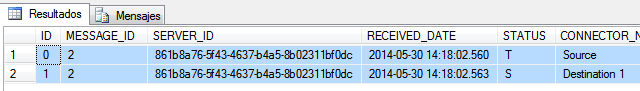
\*

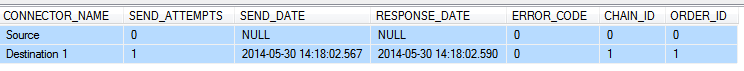
FROM

[mirthdb].[dbo].[**D\_MM1**]

WHERE

MESSAGE\_ID = 2;



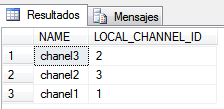


The result of the query displays two rows. It’s so, because one row is for the Source connector of the channel, and another row for the destination connector of the channel. If the channel has many destinations, the information of one message will be displayed in several rows.

As number of channels changes when you configure Mirth ESB, the same happens with the number of tables in the DB.

Executing the following query you will see the correspondence between TableIds and ChannelNames:

SELECT

 a.[NAME]

,b.[LOCAL\_CHANNEL\_ID]

FROM

[mirthdb].[dbo].[CHANNEL] a

,[mirthdb].[dbo].[D\_CHANNELS] b

WHERE

a.ID = b.CHANNEL\_ID;

The table [**D\_MM1**], corresponds to the Channel that has name: ‘chanel1’

The table [D\_MM2], corresponds to the Channel that has name: ‘chanel3’

The table [D\_MM3], corresponds to the Channel that has name: ‘chanel2’

Now you know what to do to see response times of certain channel.

Just query the right table, and check the Received\_Date, Send\_Date, Response\_Date columns.

## Calculating time taken of messages with Mirth ESB

The following query is an example for a simple channel of how to calculate time taken inside a channel and time taken outside the channel:

SELECT

a.[MESSAGE\_ID]

,b.[SEND\_DATE] - a.[RECEIVED\_DATE] as 'DelayInternal'

,b.[RESPONSE\_DATE] - b.[SEND\_DATE] as 'DelayExternal'

FROM

[mirthdb].[dbo].[D\_MM1] a

,[mirthdb].[dbo].[D\_MM1] b

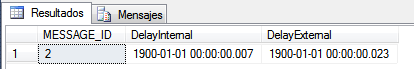
WHERE

a.MESSAGE\_ID = 2

and a.MESSAGE\_ID = b.MESSAGE\_ID

and a.[STATUS] = 'T'

and b.[STATUS] = 'S'



## Using the time Taken of Mirth ESB in the DB Monitor

As said, the DB Monitor tool has a XML configuration file to control the monitor. Now we’ll describe it. The DB monitor allows executing different queries to the database. For example, you can configure to execute one query for each Channel.

**<chart**

name="**Channel1**"

intervalBetweenQuery="2000"

query="SELECT top 200 a.[MESSAGE\_ID], a.[RECEIVED\_DATE], (b.[SEND\_DATE] - a.[RECEIVED\_DATE]) as 'Delay\_IntraChanel', (b.[RESPONSE\_DATE] - b.[SEND\_DATE]) as 'Delay\_OutOfChanel' FROM [mirthdb].[dbo].[D\_MM1] a, [mirthdb].[dbo].[D\_MM1] b WHERE a.MESSAGE\_ID **&gt;** **${LAST\_ID}** and a.MESSAGE\_ID = b.MESSAGE\_ID and a.[STATUS] = 'T' and b.[STATUS] = 'S';"

paramDefinition="LAST\_ID = maxOfColumn(1, 0)"

skipColumnsInChart="1"

chartType="Normal"

xAxisColumnId="2"

chartBufferSize="5000"

chartUpdateNumber = "1"

**/>**

**<connector**

name=" **Channel1**"

connectionString = "jdbc:sqlserver://localhost:8053;databaseName=mirthdb;user=mirth;password=mirth"

classforname = "com.microsoft.sqlserver.jdbc.SQLServerDriver"

**/ >**

With that configuration, DB Monitor will create a monitor that will run the query each 2000 milliseconds. After each query execution, the chart is going to be updated with the new values.

As the query has a parameter, it will be preprocessed to instantiate that parameter.

If the table has 358 messageIds, DB Monitor will execute the following queries:

First time:

SELECT **top 200** a.[MESSAGE\_ID], a.[RECEIVED\_DATE], (b.[SEND\_DATE] - a.[RECEIVED\_DATE]) as 'Delay\_IntraChanel', (b.[RESPONSE\_DATE] - b.[SEND\_DATE]) as 'Delay\_OutOfChanel' FROM [mirthdb].[dbo].[D\_MM1] a, [mirthdb].[dbo].[D\_MM1] b WHERE a.MESSAGE\_ID **>** **0** and a.MESSAGE\_ID = b.MESSAGE\_ID and a.[STATUS] = 'T' and b.[STATUS] = 'S';

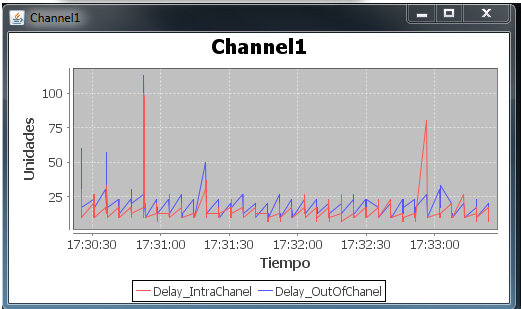
Second time:

SELECT **top 200** a.[MESSAGE\_ID], a.[RECEIVED\_DATE], (b.[SEND\_DATE] - a.[RECEIVED\_DATE]) as 'Delay\_IntraChanel', (b.[RESPONSE\_DATE] - b.[SEND\_DATE]) as 'Delay\_OutOfChanel' FROM [mirthdb].[dbo].[D\_MM1] a, [mirthdb].[dbo].[D\_MM1] b WHERE a.MESSAGE\_ID **>** **200** and a.MESSAGE\_ID = b.MESSAGE\_ID and a.[STATUS] = 'T' and b.[STATUS] = 'S';

Third time:

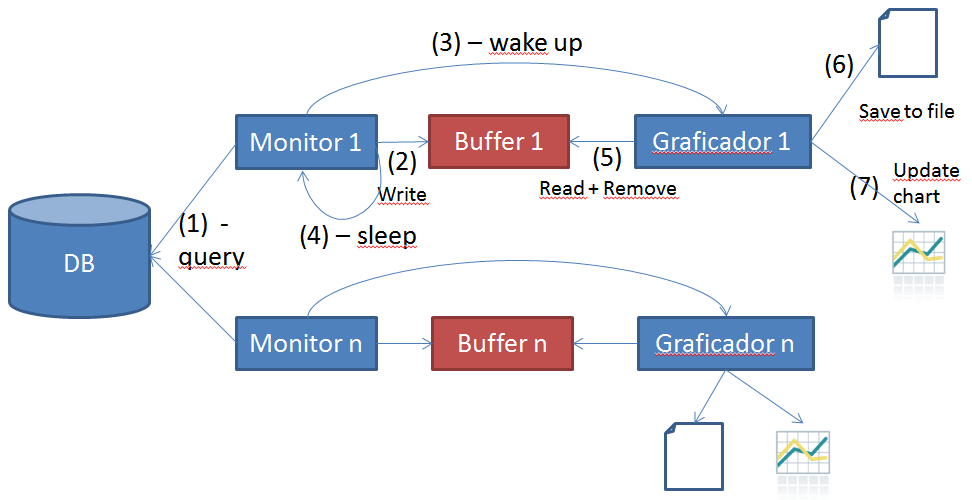
SELECT **top 200** a.[MESSAGE\_ID], a.[RECEIVED\_DATE], (b.[SEND\_DATE] - a.[RECEIVED\_DATE]) as 'Delay\_IntraChanel', (b.[RESPONSE\_DATE] - b.[SEND\_DATE]) as 'Delay\_OutOfChanel' FROM [mirthdb].[dbo].[D\_MM1] a, [mirthdb].[dbo].[D\_MM1] b WHERE a.MESSAGE\_ID **>** **358** and a.MESSAGE\_ID = b.MESSAGE\_ID and a.[STATUS] = 'T' and b.[STATUS] = 'S';

DB Monitor will display the following chart:



## How DB Monitor works

Each chart tag in the config.xml file will create two threads: Monitor and Graficador. The communication between those threads is made using the buffer and a semaphore to synchronize.



### Monitor

The following algorithm describes what Monitor thread does:

*Connect to DB()*

*While (true) {*

*preProcessQuery()*

*executeQuery()*

*processResults()*

*saveToBuffer()*

*wakeUpGraficador()*

*sleep(time)*

*}*

### Graficador

The following algorithm describes what Graficador thread does:

*While (true) {*

*waitUntilWakeUp()*

*saveSharedBufferToFile()*

*updateOwnBuffer()*

*clearSharedBuffer()*

*updateChart()*

*}*