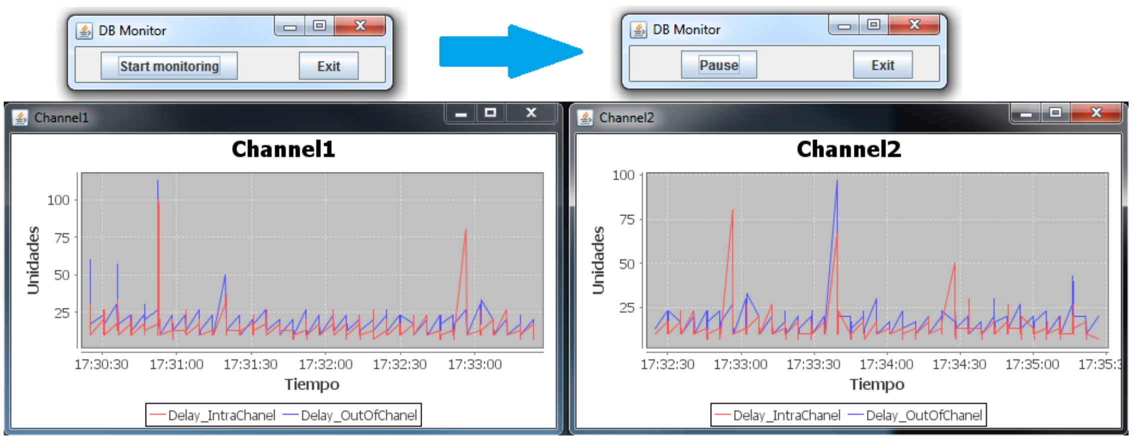
DB Monitor Tutorial

Contents

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.

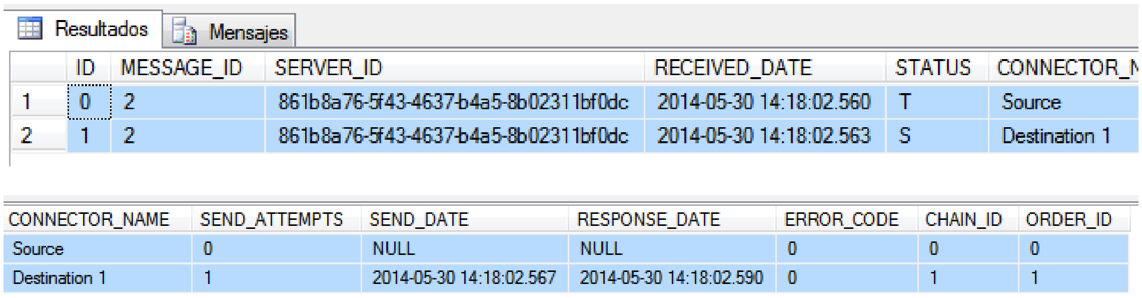
```
<chart
        name="Channel1"
        intervalBetweenQuery="2000"
        query="SELECT top 50 a.[MESSAGE_ID], a.[RECEIVED_DATE], (b.[SEND_DATE] - a.[RECEIVED_DATE]) as 'Delay_IntraChan
        paramDefinition="LAST_ID = maxOfColumn(1, 8)"
        skipColumnsInChart="1"

        chartType="Normal"

        xAxisColumnId="2"
        chartBufferSize="300"
        chartUpdateNumber = "1">
</chart>
<connector
        name="Channel1"
        connectionString = "jdbc:sqlserver://localhost:8053;databaseName=mirthdb;user=mirth;password=mirth"
        classforname = "com.microsoft.sqlserver.jdbc.SQLServerDriver">
</connector>
```

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;
```



Resultados | Mensajes

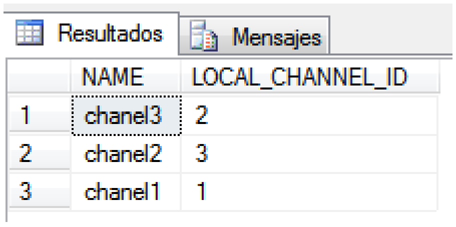
	ID	MESSAGE_ID	SERVER_ID	RECEIVED_DATE	STATUS	CONNECTOR_N
1	0	2	861b8a76-5f43-4637-b4a5-8b02311bf0dc	2014-05-30 14:18:02.560	T	Source
2	1	2	861b8a76-5f43-4637-b4a5-8b02311bf0dc	2014-05-30 14:18:02.563	S	Destination 1

CONNECTOR_NAME	SEND_ATTEMPTS	SEND_DATE	RESPONSE_DATE	ERROR_CODE	CHAIN_ID	ORDER_ID
Source	0	NULL	NULL	0	0	0
Destination 1	1	2014-05-30 14:18:02.567	2014-05-30 14:18:02.590	0	1	1

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;
```



Resultados | Mensajes

	NAME	LOCAL_CHANNEL_ID
1	chanel3	2
2	chanel2	3
3	chanel1	1

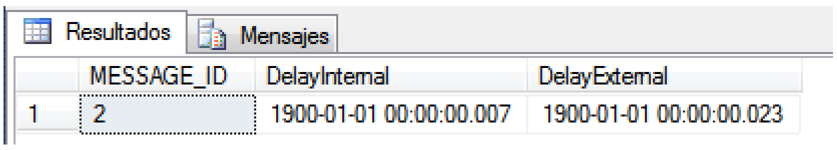
The table [D_MM1], corresponds to the Channel that has name: 'chanel1'
The table [D_MM2], corresponds to the Channel that has name: 'chanel3'
The table [D_MM3], corresponds to the Channel that has name: 'chanel2'

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'
```



	MESSAGE_ID	DelayInternal	DelayExternal
1	2	1900-01-01 00:00:00.007	1900-01-01 00:00:00.023

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';
```

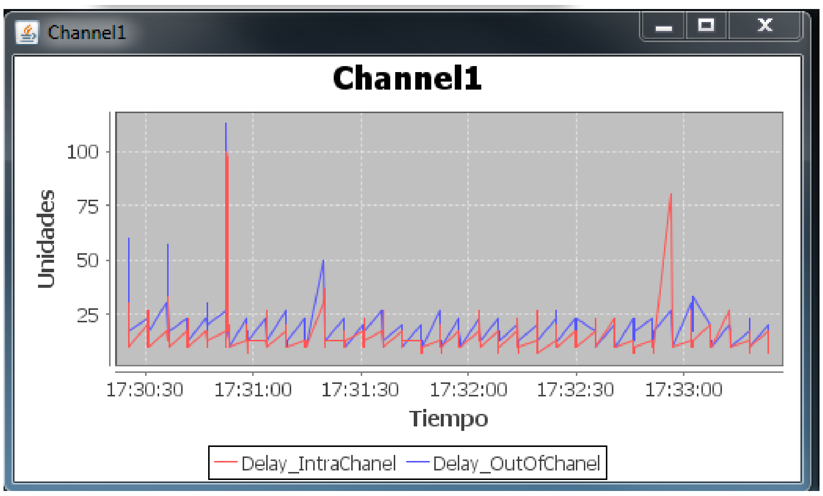
Second time:

```
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:

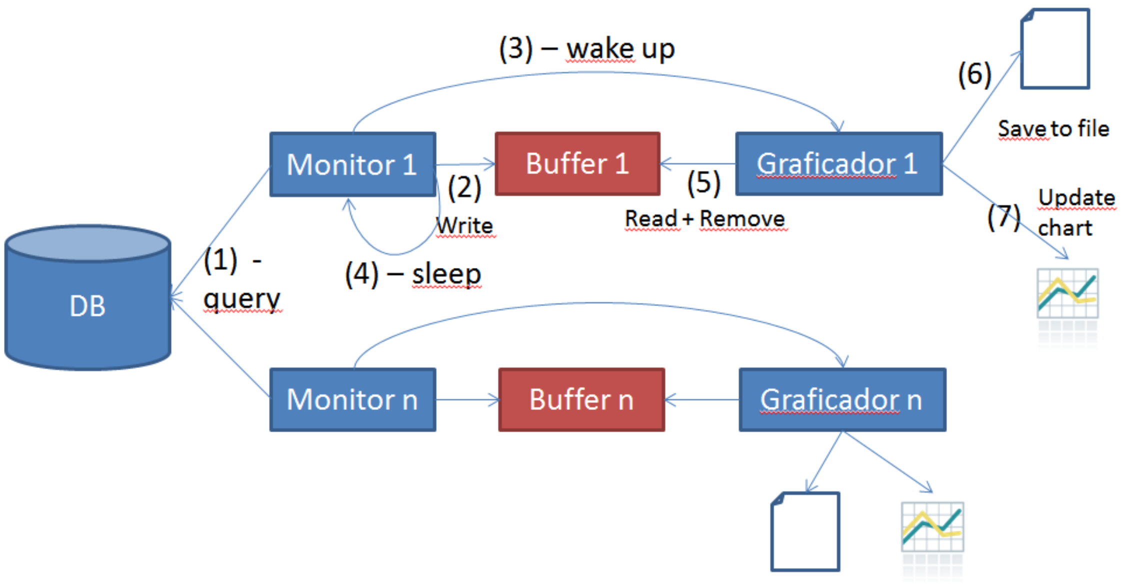
```
SELECT top 200 a.[MESSAGE_ID], a.[RECEIVED_DATE], (b.[SEND_DATE] - a.[RECEIVED_DATE]) as 'Delay_IntraChanel', (b.[RESPONSE_DATE] - b.[SEND_DATE]) as 'Delay_OutOfChanel' FROM [mirthdb].[dbo].[D_MM1] a, [mirthdb].[dbo].[D_MM1] b WHERE a.MESSAGE_ID > 358 and a.MESSAGE_ID = b.MESSAGE_ID and a.[STATUS] = '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()
}
```