

DB Port Tutorial

Installation and Startup

1. Compile all classes in the dbport folders with your favourite Java compiler. Make sure that the myDB package is in the classpath.

Example: `javac -classpath <path of the myDB folder> dbport/*.java`

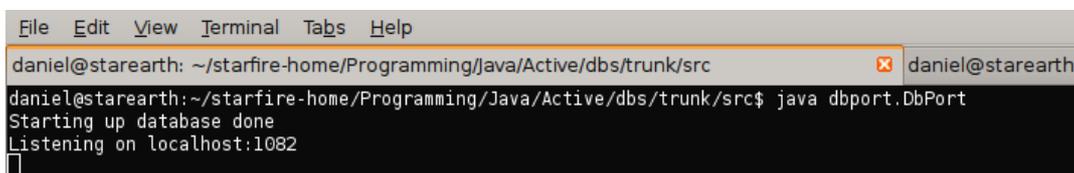
2. Create a folder “disk” in the folder that you want to run dbport in (without this folder, your database might be unable to persist its data):

`mkdir disk`

3. Run dbport.DbPort with your favorite Java virtual machine. Again, the myDB package must be in the path.

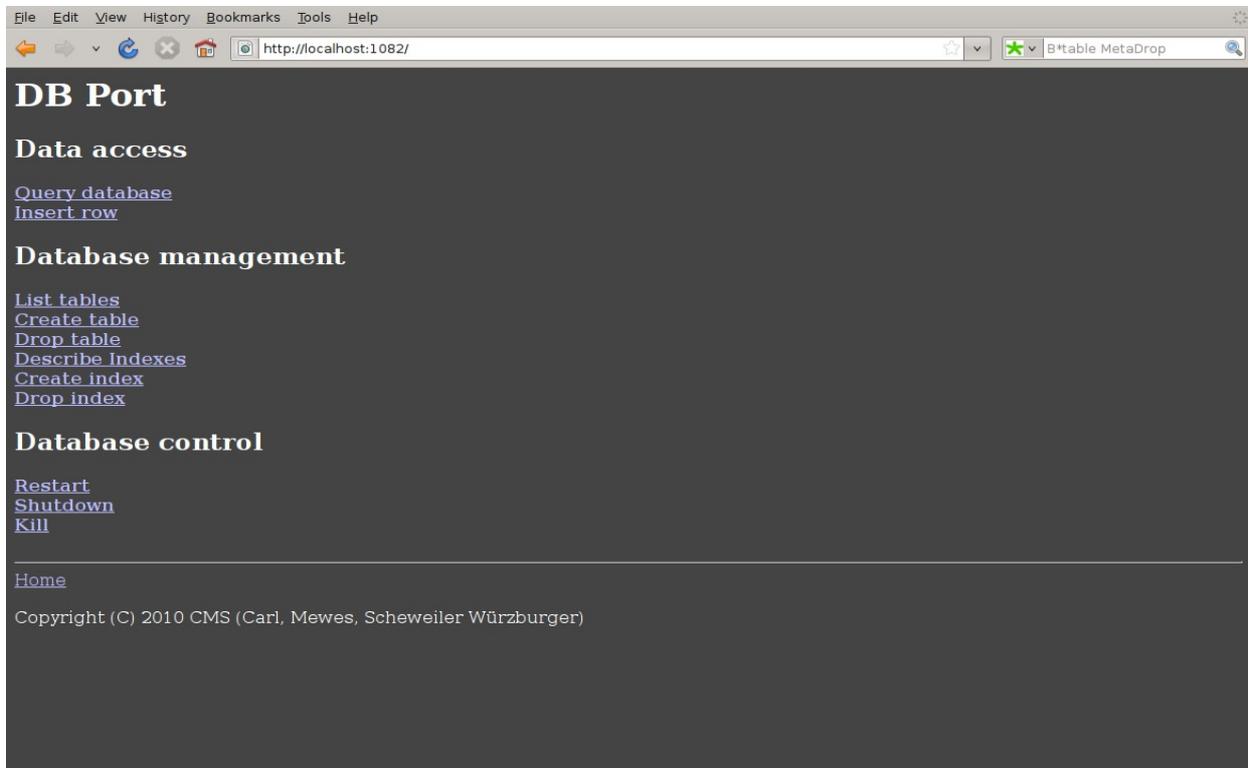
Example: `java -classpath ./:<path of the myDB folder> dbport.DbPort`

You should see a message similar to this one:



```
File Edit View Terminal Tabs Help
daniel@starearth: ~/starfire-home/Programming/Java/Active/dbs/trunk/src
daniel@starearth:~/starfire-home/Programming/Java/Active/dbs/trunk/src$ java dbport.DbPort
Starting up database done
Listening on localhost:1082
```

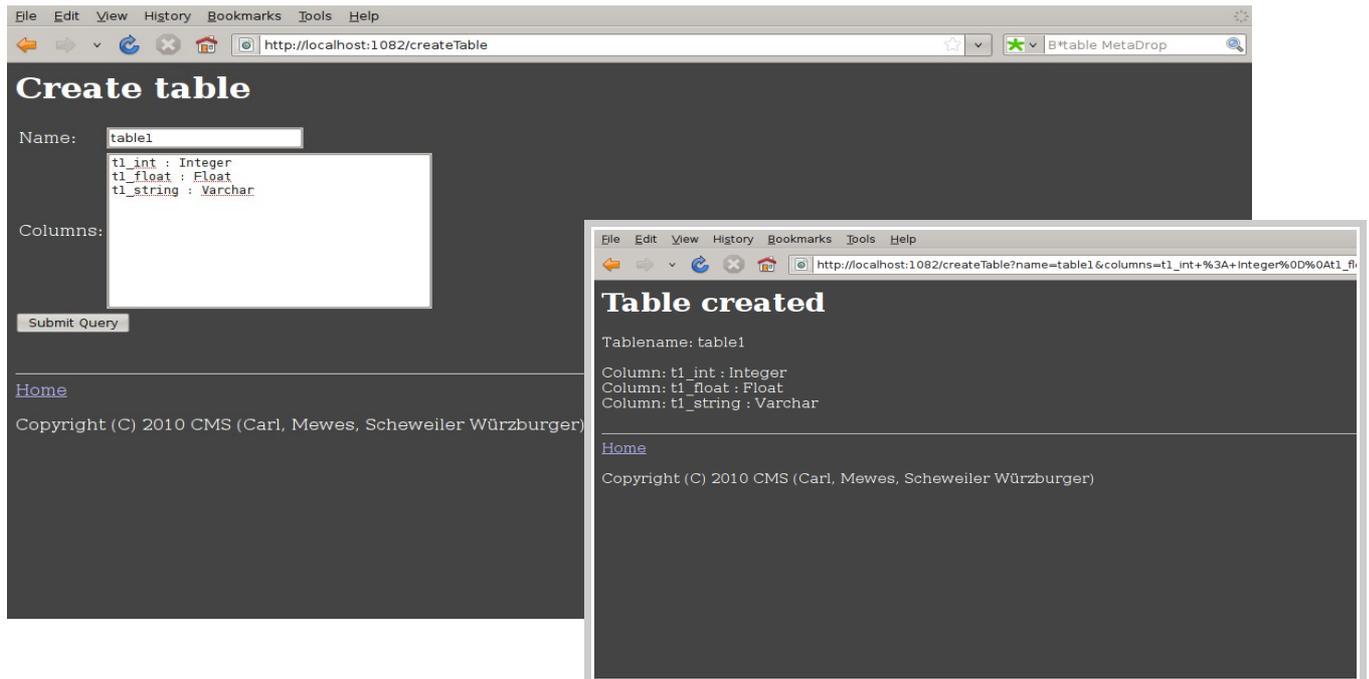
4. Per default, DB Port binds to localhost on port 1082 (check DbPort.java if you want to change it). Therefore, DB Port is accessible from you local host only. Open up your favorite web browser and navigate to <http://localhost:1082>



Database Management

We now want to create two tables and an index.

1. On the DB Port home page, click “Create table”.
2. Enter the table name “table1” as well as the schema. We want to have three columns in table1: “t1_int” of type Integer, “t1_float” of type Float and “t1_string” of type Varchar. To enter the schema, write one column per line in the format “<column name> : <type>”. Please note that the type is case sensitive and must be one of “Integer”, “Long”, “Float”, “Double”, “Date”, “Varchar” and “Char(<length>)”. Enter the data and click the submit button.



3. Create a second table with the name “table2” and two columns: “t2_int” of type Integer and “t2_date” of type Date.

The image shows two browser windows from the DB Port application. The left window is titled "Create table" and shows a form where the table name is "table2" and the columns are "t2_int : Integer" and "t2_date : Date". A "Submit Query" button is visible. The right window is titled "Table created" and displays the confirmation: "Tablename: table2", "Column: t2_int : Integer", and "Column: t2_date : Date".

4. Now we want to add an index for table2's date column. Therefore, return to DB Port's home page and click “Create index”. Enter the index name (e.g. “t2_date_index”), the table (“table2”) and the attribute (“t2_date”). Also check the “Range index” option, to enable the creation of an index with range query support. Enter all data and submit.

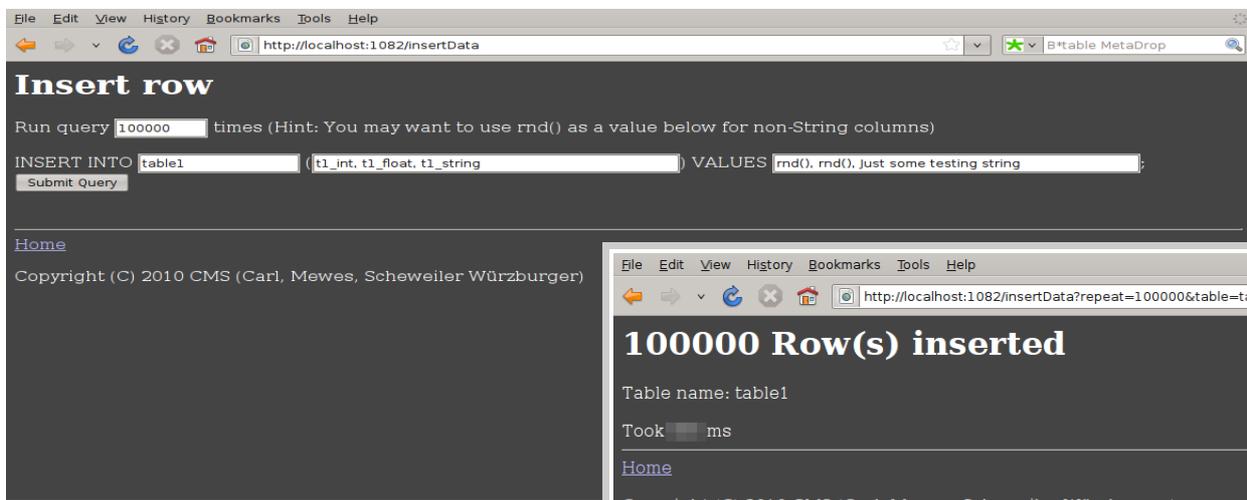
The image shows two browser windows from the DB Port application. The left window is titled "Create index" and shows a form where the index name is "t2_date_index", the table is "table2", and the attribute is "t2_date". The "Range Index" checkbox is checked. A "Submit Query" button is visible. The right window is titled "Range index created" and displays the confirmation: "Index name: t2_date_index", "Table name: table2", and "Attribute: t2_date".

Inserting Data

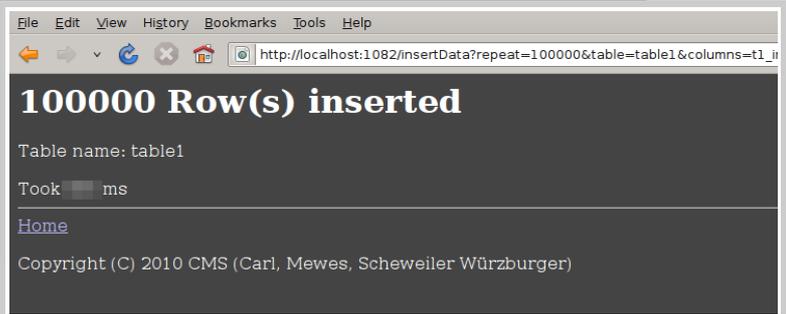
In the last steps, we have created two tables in our database. Now we want to fill them with some data.

1. Click the “Insert row” link on DB Port's home page.
2. We want to fill table1 with 100,000 rows. DB Port comes with two features, which greatly simplify the creation of test data. First, it has a repetition field that you can use to let DB port perform a certain action (inserting rows or querying the database) multiple times. The second feature is the insertion of random values (not supported for Varchar and Char type columns currently). We now use both features to create some test data.

For this, enter “100000” into the repetition field. Make the insertion query read like “INSERT INTO table1 (t1_int, t1_float, t1_string) VALUES rnd(), rnd(), just some testing string;”. As you can see, we assign random values to t1_int and t1_float and the constant value “just some testing string” to t1_string (additional information: string values must not contain commata, enter null as a value to assign null). Click the submit button to insert the rows.

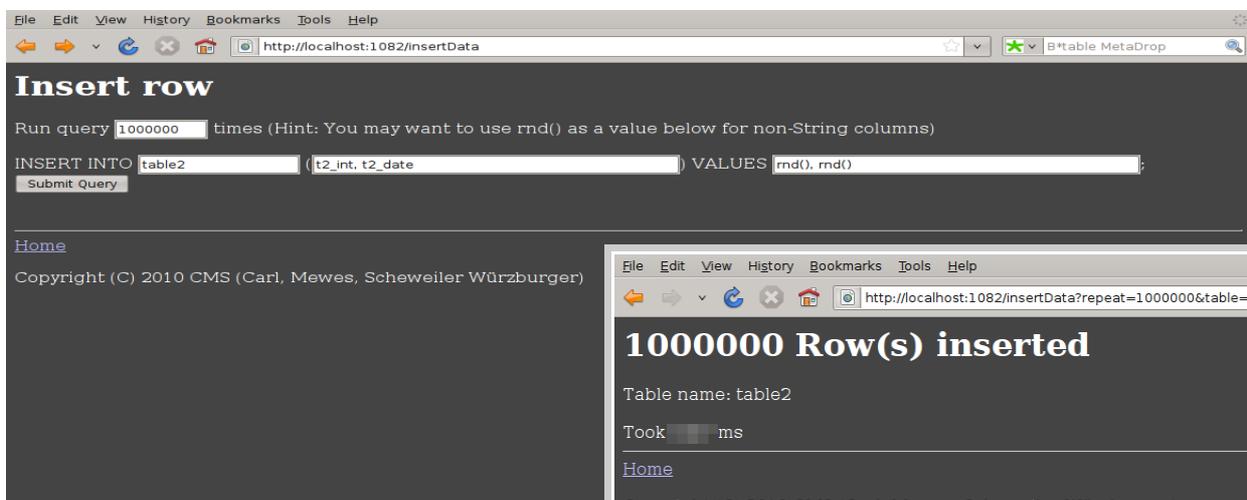


The screenshot shows the 'Insert row' interface in a browser. The URL is `http://localhost:1082/insertData`. The form includes a 'Run query' field with the value '100000'. The 'INSERT INTO' field contains 'table1', the columns field contains 't1_int, t1_float, t1_string', and the 'VALUES' field contains 'rnd(), rnd(), just some testing string'. A 'Submit Query' button is visible.

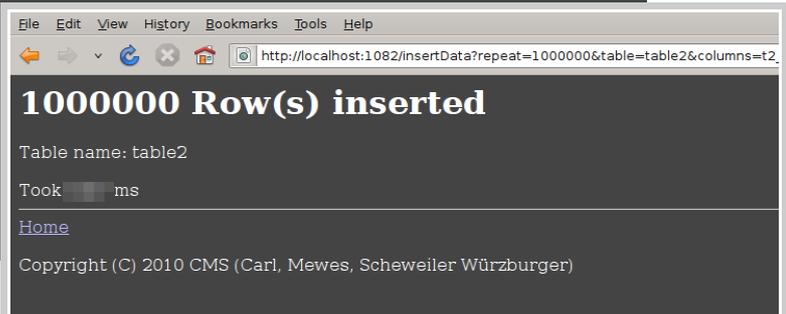


The success message displays: **100000 Row(s) inserted**. Below this, it shows 'Table name: table1' and 'Took [] ms'. A 'Home' link and copyright notice are also present.

3. Repeat the procedure for table2 in order to insert a million random rows:



The screenshot shows the 'Insert row' interface in a browser. The URL is `http://localhost:1082/insertData`. The form includes a 'Run query' field with the value '1000000'. The 'INSERT INTO' field contains 'table2', the columns field contains 't2_int, t2_date', and the 'VALUES' field contains 'rnd(), rnd()'. A 'Submit Query' button is visible.



The success message displays: **1000000 Row(s) inserted**. Below this, it shows 'Table name: table2' and 'Took [] ms'. A 'Home' link and copyright notice are also present.

Querying the Database

We have just inserted some rows into our tables. We now want to formulate a query to the database.

1. Navigate to “Query database” from DB Port's home page.
2. Formulating queries in DB Port works by
 1. defining a set of input, join and cross product nodes
 2. selecting a result node (like the root of a relational algebra expression)
 3. specifying predicate nodes which are applied as a selection on top of the result node
 4. specifying which columns the result should be projected on
3. We want to formulate the following query (in SQL): “SELECT t1_string, t2_date FROM table1 JOIN table2 ON t1_int = t2_int WHERE t2_date >= '2010-01-01' AND t2_date < '2011-01-01'; ” (i.e. select all dates from table2 which are in 2010 and also get the corresponding – with regard to t1_int=t2_int – t1_string values from table1).

Therefore, we first specify two input nodes. Each node has a node number. At node 1, enter “table1” as the table name. At node 2, enter “table2”.

4. To join both inputs, we use a join node. At node 8, enter “1” as the left node number (=table1) and “2” as the right node number (=table2). Also specify the join condition as “t1_int” = “t2_int”. This essentially specifies a tree with the join node as the root and the two input nodes as its children / as the leaves.

File Edit View History Bookmarks Tools Help

http://localhost:1082/queryData

Query database

Run query times

Input nodes:

Input 1:	table1
Input 2:	table2
Input 3:	table name
Input 4:	table name
Input 5:	table name
Input 6:	table name
Input 7:	table name

Join nodes:

Join 8:	JOIN	1	WITH	2	ON	t1_int	=	t2_int
Join 9:	JOIN	left node number	WITH	right node number	ON	left attribute	=	right attribute
Join 10:	JOIN	left node number	WITH	right node number	ON	left attribute	=	right attribute
Join 11:	JOIN	left node number	WITH	right node number	ON	left attribute	=	right attribute
Join 12:	JOIN	left node number	WITH	right node number	ON	left attribute	=	right attribute
Join 13:	JOIN	left node number	WITH	right node number	ON	left attribute	=	right attribute

Cross product nodes:

Cross product 14:	node number	X	node number
Cross product 15:	node number	X	node number
Cross product 16:	node number	X	node number
Cross product 17:	node number	X	node number
Cross product 18:	node number	X	node number
Cross product 19:	node number	X	node number

Take result from node...

Predicate root: AND

Predicate node 1:

Predicate node 2:

Predicate node 3:

Predicate node 4:

Predicate node 5:

Predicate node 6:

Predicate node 7:

Predicate node 8:

Predicate node 9:

...apply predicate... Predicate node 10:

Predicate node 11:

Done

5. The root of our tree is join node 8. Enter “8” as the result node.
6. To specify the selection (“WHERE” clause in SQL), first set predicate node 1 to “t2_date >= 2010-01-01” and predicate node 2 to “t2_date < 2011-01-01”. To “AND” both predicates together and specify the result of the logical and as the predicate's tree root, set the predicate root to “1 AND 2”.
7. Enter “t1_string, t2_string” into the projection field. (Hint: to not apply a projection at all, simply empty the projection field).

The screenshot shows a web browser window with the URL `http://localhost:1082/queryData`. The interface is a query builder with the following sections:

- Cross product nodes:** A list of five rows, each with a label (Cross product 15-19) and two columns labeled 'node number' and 'X node number'.
- Take result from node...:** A text input field containing the number '8'.
- Predicate root:** A dropdown menu showing '1 AND 2'.
- Predicate nodes 1-9:** A list of nodes with columns for 'node number / column', an operator, and 'node number / value'.

Predicate node 1:	t2_date	>=	2010-01-01
Predicate node 2:	t2_date	<	2011-01-01
Predicate node 3:	node number / column	-	node number / value
Predicate node 4:	node number / column	-	node number / value
Predicate node 5:	node number / column	-	node number / value
Predicate node 6:	node number / column	-	node number / value
Predicate node 7:	node number / column	-	node number / value
Predicate node 8:	node number / column	-	node number / value
Predicate node 9:	node number / column	-	node number / value
- ...apply predicate...:** A list of predicate nodes 10-20, each with columns for 'node number / column', an operator, and 'node number / value'. All operators are set to '-'.

Predicate node 10:	node number / column	-	node number / value
Predicate node 11:	node number / column	-	node number / value
Predicate node 12:	node number / column	-	node number / value
Predicate node 13:	node number / column	-	node number / value
Predicate node 14:	node number / column	-	node number / value
Predicate node 15:	node number / column	-	node number / value
Predicate node 16:	node number / column	-	node number / value
Predicate node 17:	node number / column	-	node number / value
Predicate node 18:	node number / column	-	node number / value
Predicate node 19:	node number / column	-	node number / value
Predicate node 20:	node number / column	-	node number / value
- ...and project to:** A text input field containing 't1_string, t2_date'.
- Submit Query:** A button with a warning icon and the text 'Delete selected rows (only supported for single-input queries without projection; slow implementation! do not use for benchmarking)'. Below it is a 'Submit Query' button.
- Footer:** Includes a 'Home' link, copyright information 'Copyright (C) 2010 CMS (Carl, Mewes, Schweiler Würzburger)', and the word 'Done' at the bottom left.

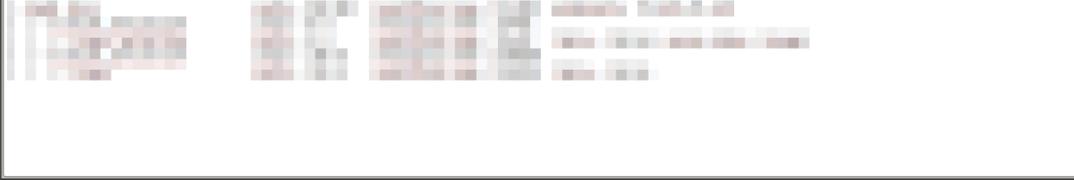
8. Submit the query! DB Port displays the query plan as provided by the database's explainQuery() method and also shows the result of the query.

File Edit View History Bookmarks Tools Help

http://localhost:1082/queryData?repeat=1&input1Table=table1&input2Table=table2&input3Table=t B*table MetaDrop

Query database

Query plan



Query result

t2_date	t1_string
Sat Jan 30 08:25:14 CET 2010	Just some testing string
Wed Dec 01 03:41:55 CET 2010	Just some testing string

Got 2 rows

Took 0 ms

[Home](#)

Copyright (C) 2010 CMS (Carl, Mewes, Scheweiler Würzburger)

(sorry, we do not give away our ingenious query plans! ;-)

Finally...

... you should shutdown the database by using the “Shutdown” link on DB Port's home page. This ensures that the database gets properly persisted to disk. To not persist any changes you have made, you may use the “Kill” function instead.

