Lab Exercise: **Visual Explain Introduction**

Speaker Name: **Morten Buur Rasmussen**

Introduction:

This exercise covers SQL scenarios where you are using Visual Explain. The SQL requests are coming from a Plan Cache **snapshot** and introduces you to the use of Visual Explain.

You will be guided though finding indexes that could help for the performance of the SQL requests.

During the lab, you will work with the following user profile:

DBCLASSXX – The XX will be substituted by a number from 01 to 40 that you will receive from the instructor.

The password is DBCLASSXXA – So, if you get number 40 your user is DBCLASS40 and password DBCLASS40A. The password is case sensitive, so type all in upper case please.

You will connect to the following partition:

COMMON1.IDEVCLOUD.COM

Exercise instructions

In order to have you look at what is going on when a SQL query is running, you will be asked about the access methods used by the query engine. You can see the access method used by looking at the access method icon or reading the text under the icon. Here you have the most frequent access methods used in this lab:

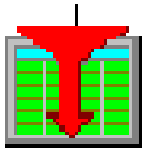
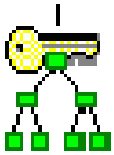
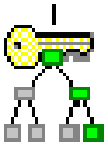


Table Scan



Index Scan



Index Probe

The reason for looking at the access method is that often is the table scans and index scans not very efficient, so it's better, if you can have the query use an index probe.

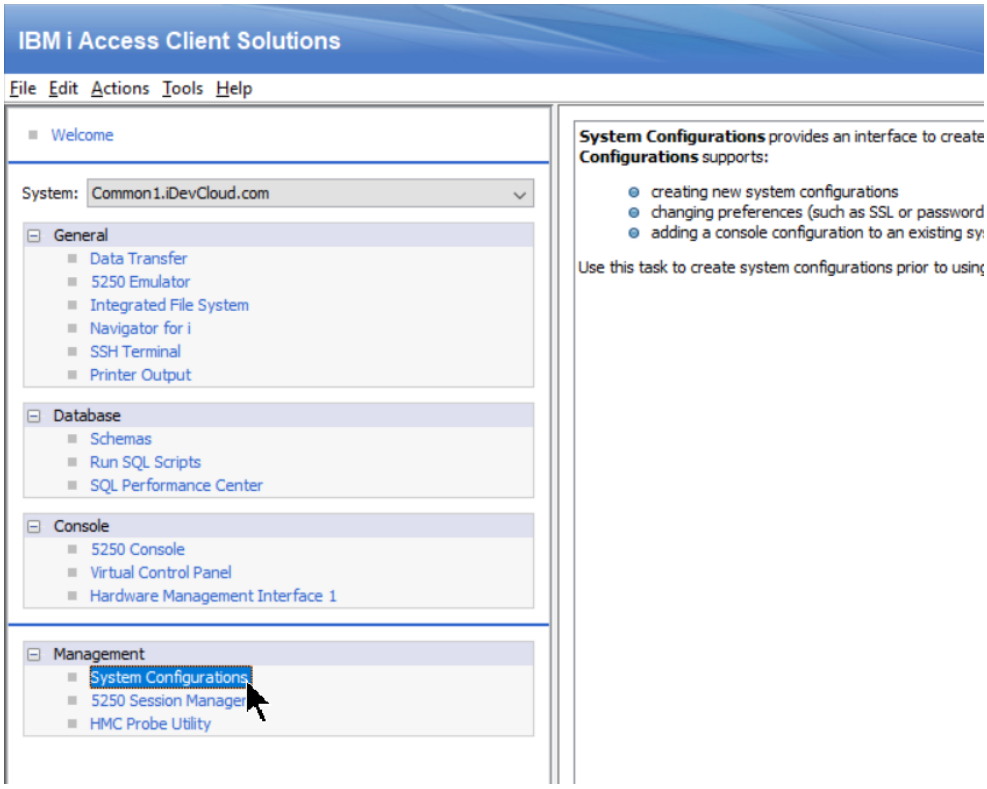
When you are working with a smaller test database, like in this lab, table scans and index scans may not take long time, so the timing do not indicate that the query is running bad. So, by looking at the icons, you can have an idea, if it's running well.

If no indexes exist, then the database can only use Table Scan.

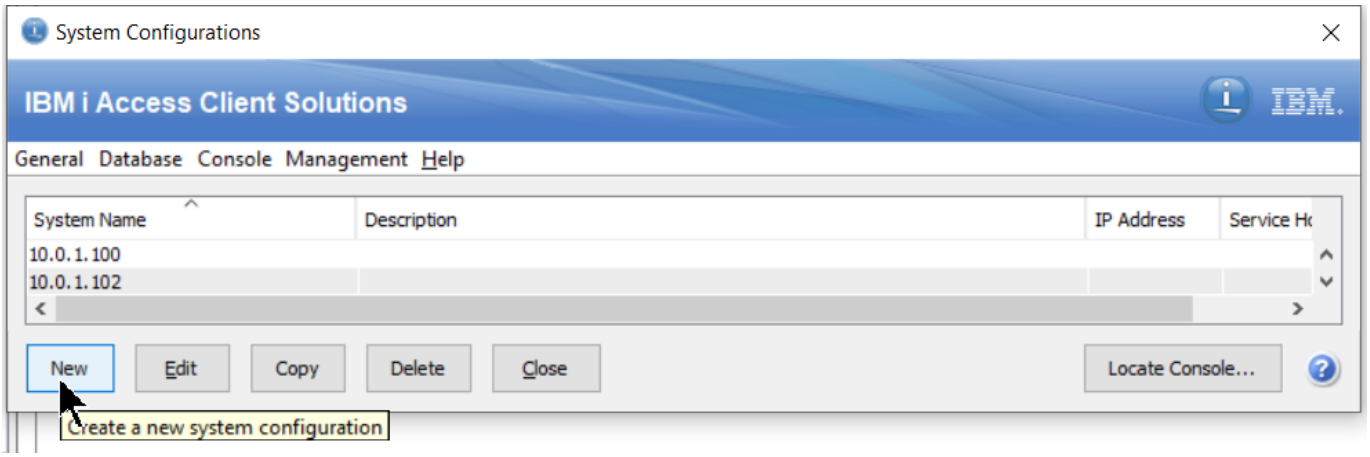
Start ACS (Access Client Solution) by double clicking on the ACS icon on the desktop.

This is the old icon and indicates you are **not** up to date:

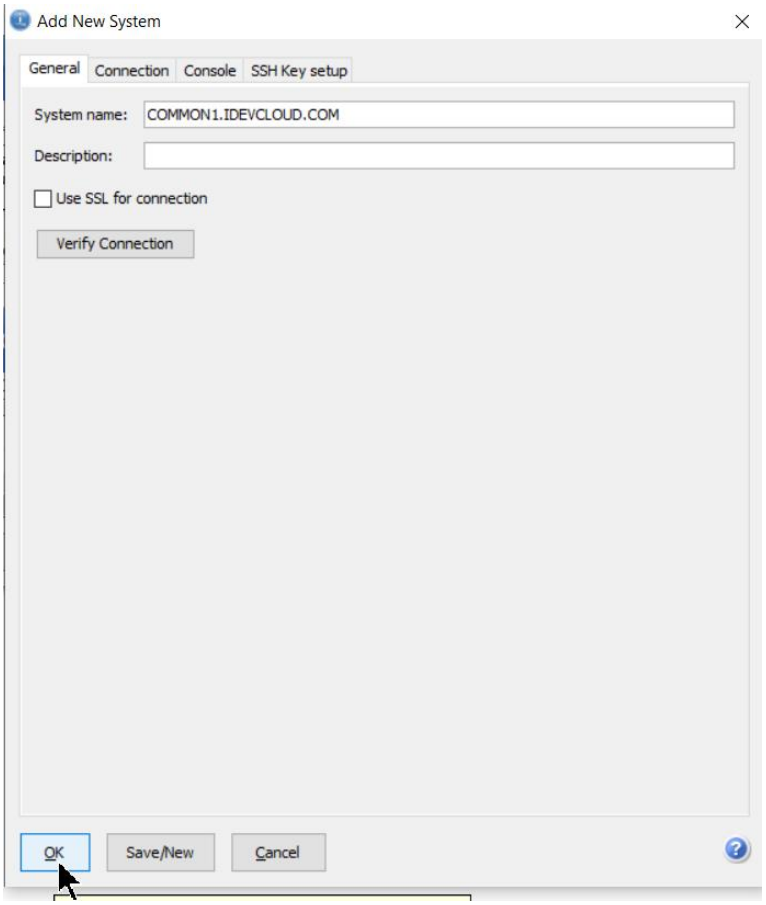
Click System Configuration:



Click the New bottom:



Type in the name of the IBM i partition and click OK:



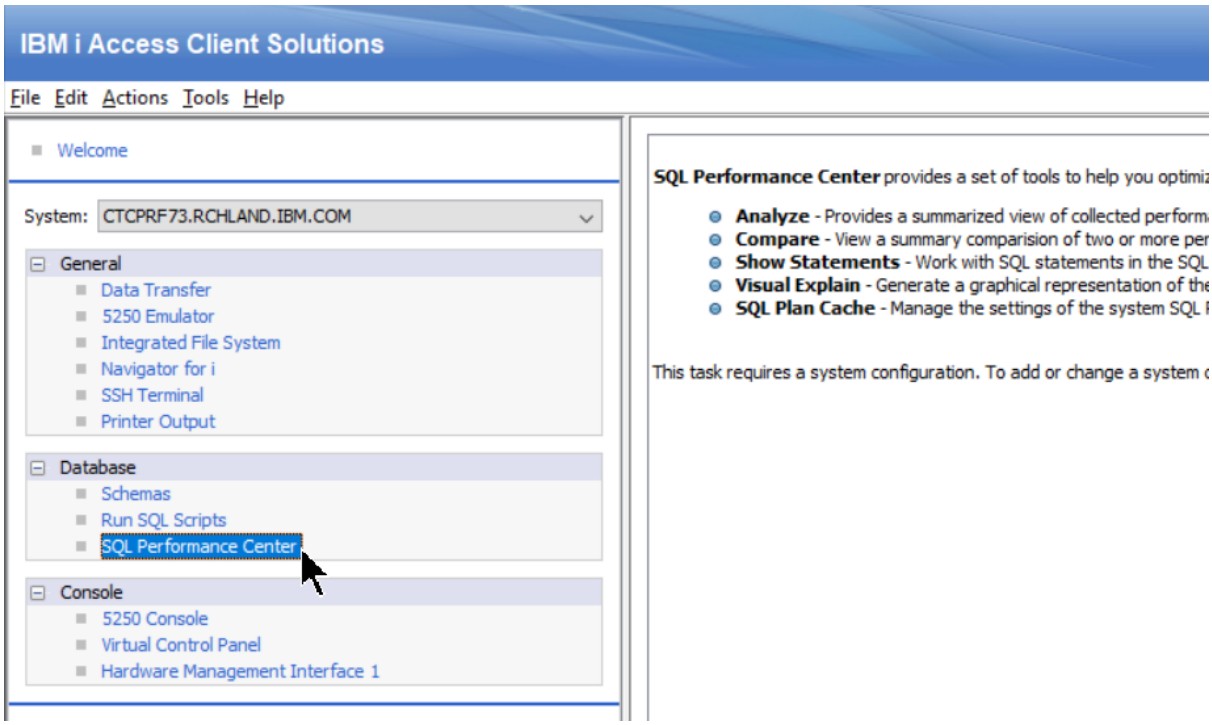
Lab 01 – Visual Explain introduction

You have been asked to look at some SQL requests that is causing the performance not to be what is expected.

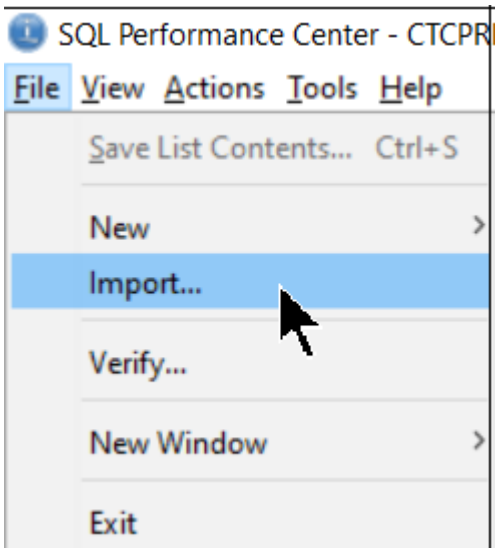
Unfortunately, you do not have access to the IBM i partition where the performance problems occur, but you have received a Plan Cache snapshot that you can import to your IBM i partition.

You will be guided through importing the Plan Cache snapshot and analyzing the SQL requests.

Click the SQL Performance Center:



Click the pull-down window for File and select Import...

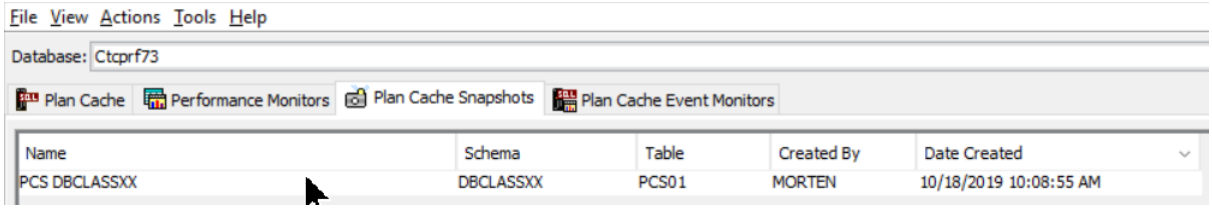


The Name is free text, but you can write PCS DBCLASSXX, where XX is your number.

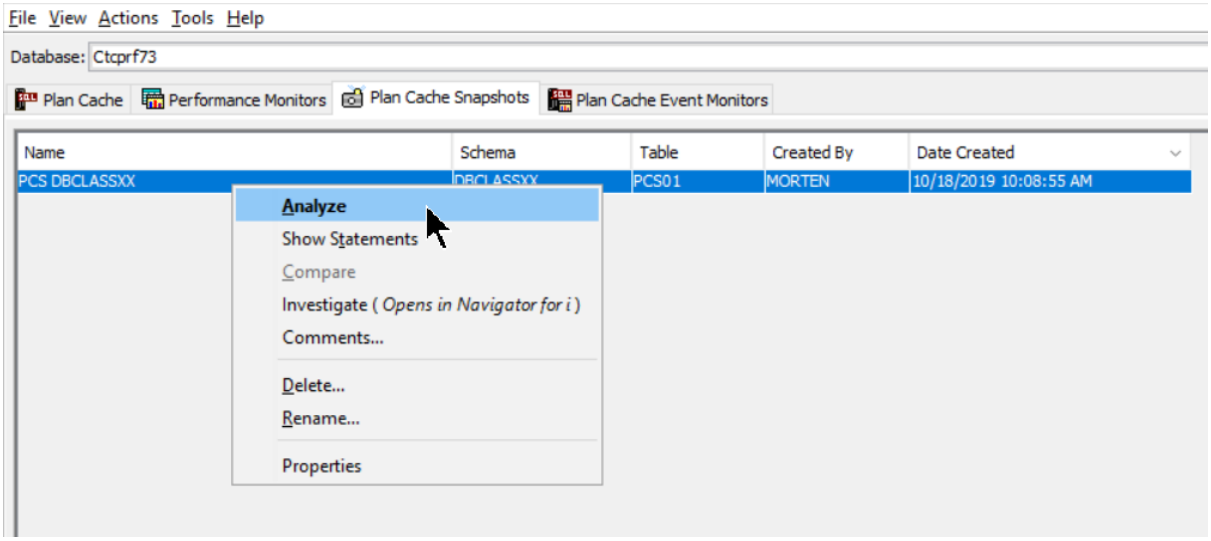
Also, for schema, you should use the one dedicated to you.

For Table, you use the pull down menu and select the PCS01 table.

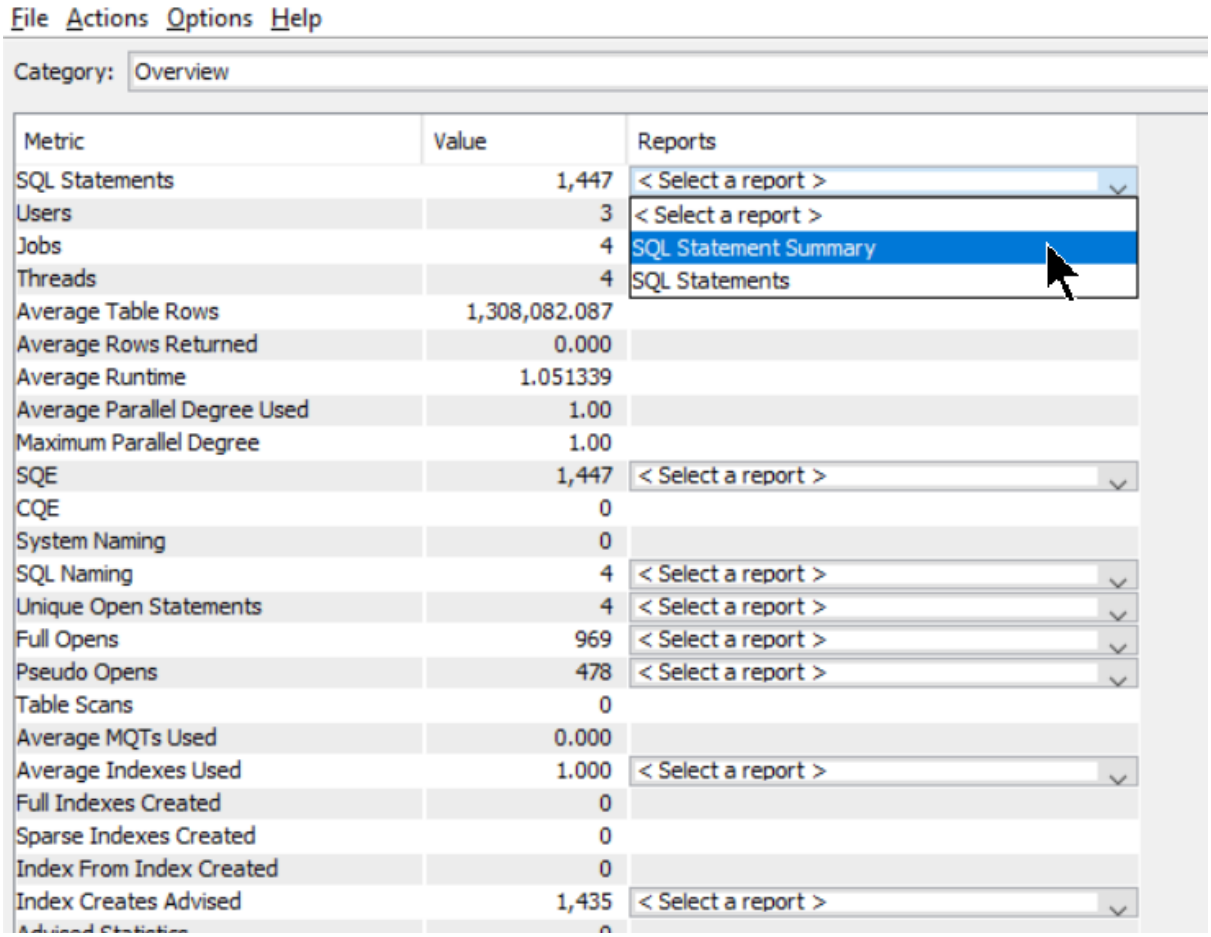
You will now, under the Plan Cache Snapshots see more snapshots, but you should only work with your own.



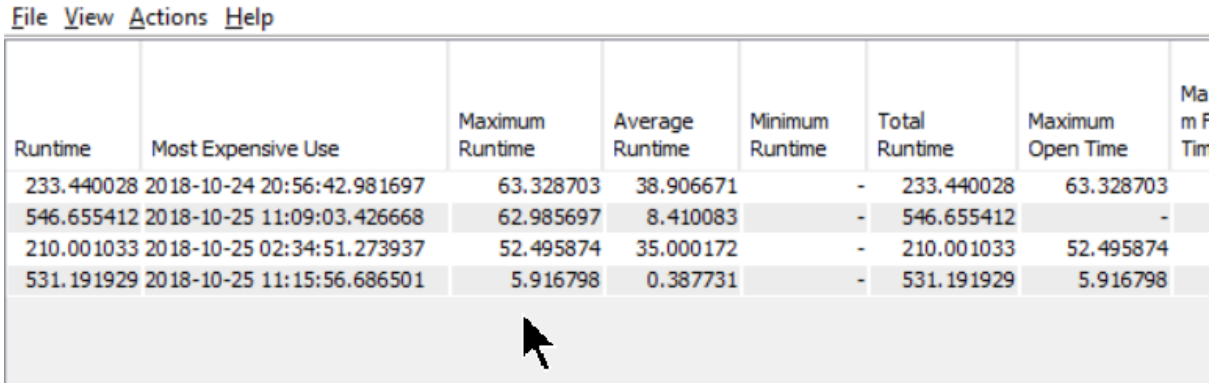
Right click and choose Analyze



Using the pull-down bar, choose the SQL Statement Summary

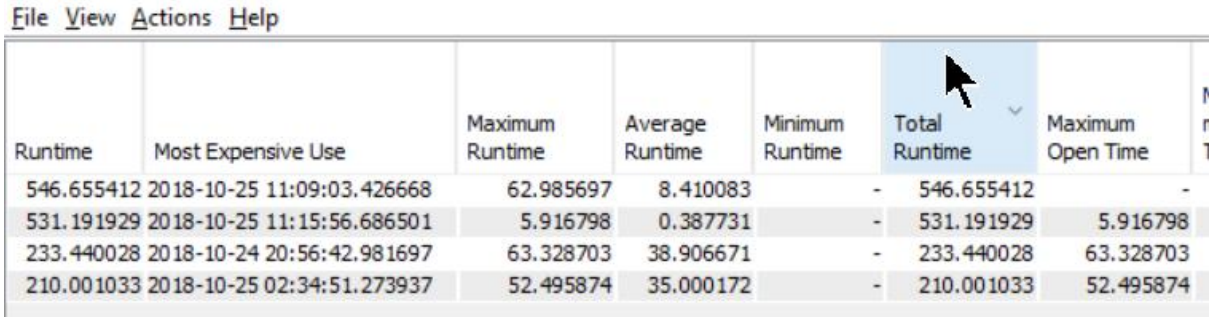


The list is by default sorted by Maximum Runtime:



It may be better for you to sort it by Total Runtime.

Click the Total Runtime to have it sorted in that way.

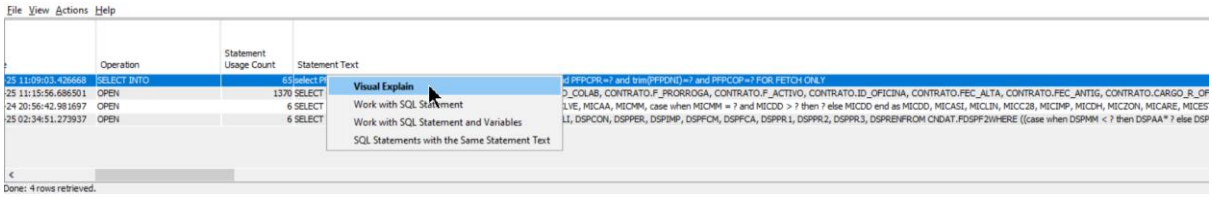


Often you will have many SQL statements, so it is advised to work with the SQL requests taking up most time totally.

In this lab you will work with all 4 SQL statements.

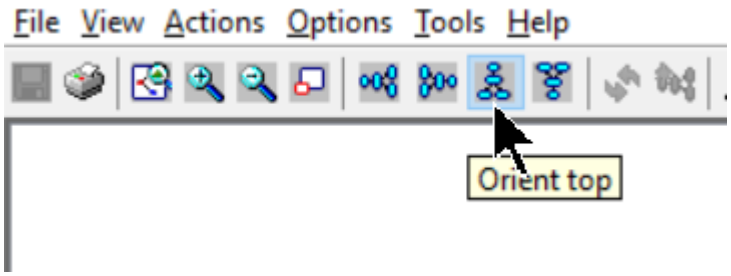
You can use the scroll bar at the bottom to scroll to the right to see more information:

Right click the first SQL statement and select Visual Explain



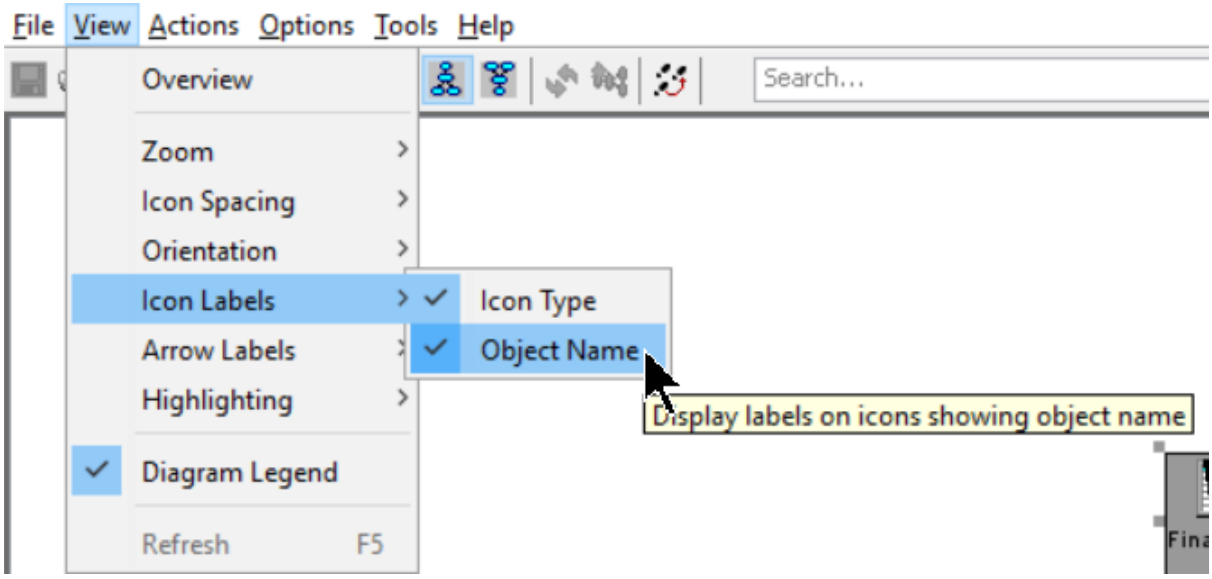
In the Visual Explain, its important that you are doing some settings. Those settings will be kept for the Visual Explain you are using.

First Click the Orient top bottom.

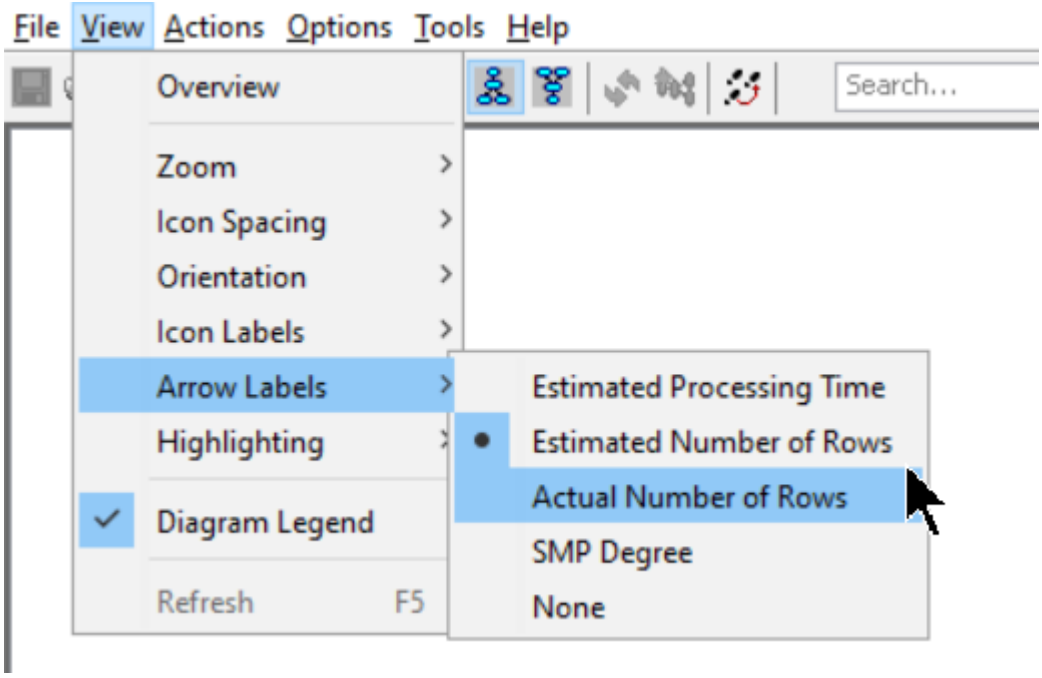


This will make sure we are all looking at the query graph in the same way.

Next, click the View pull down bar and choose Icon Labels and make sure that the Object Name is ticked on:

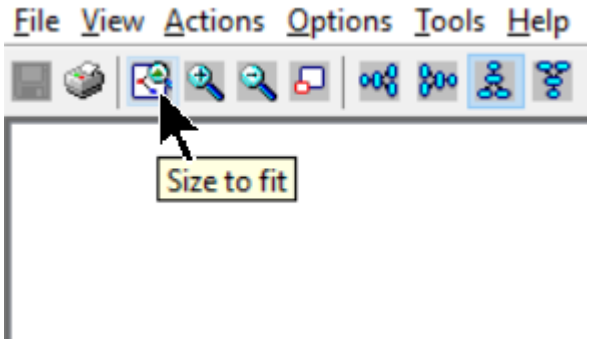


You should also change the Arrow Labels from Estimated Number of Rows to Actual Number of Rows:



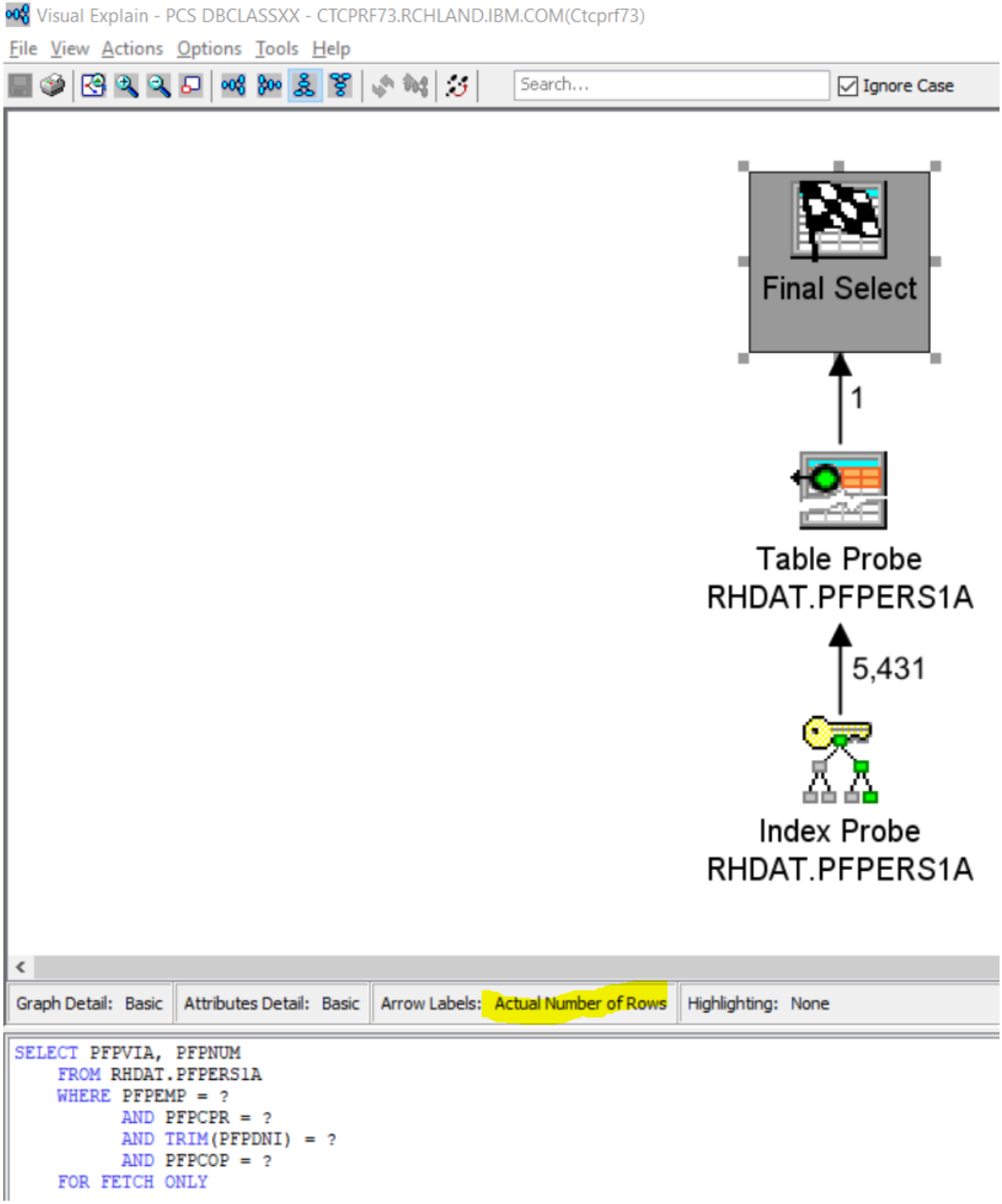
This is very useful when you have SQL requests already run.

If the SQL statement is not including too many icons, it's often fast to click the Size to fit icon:

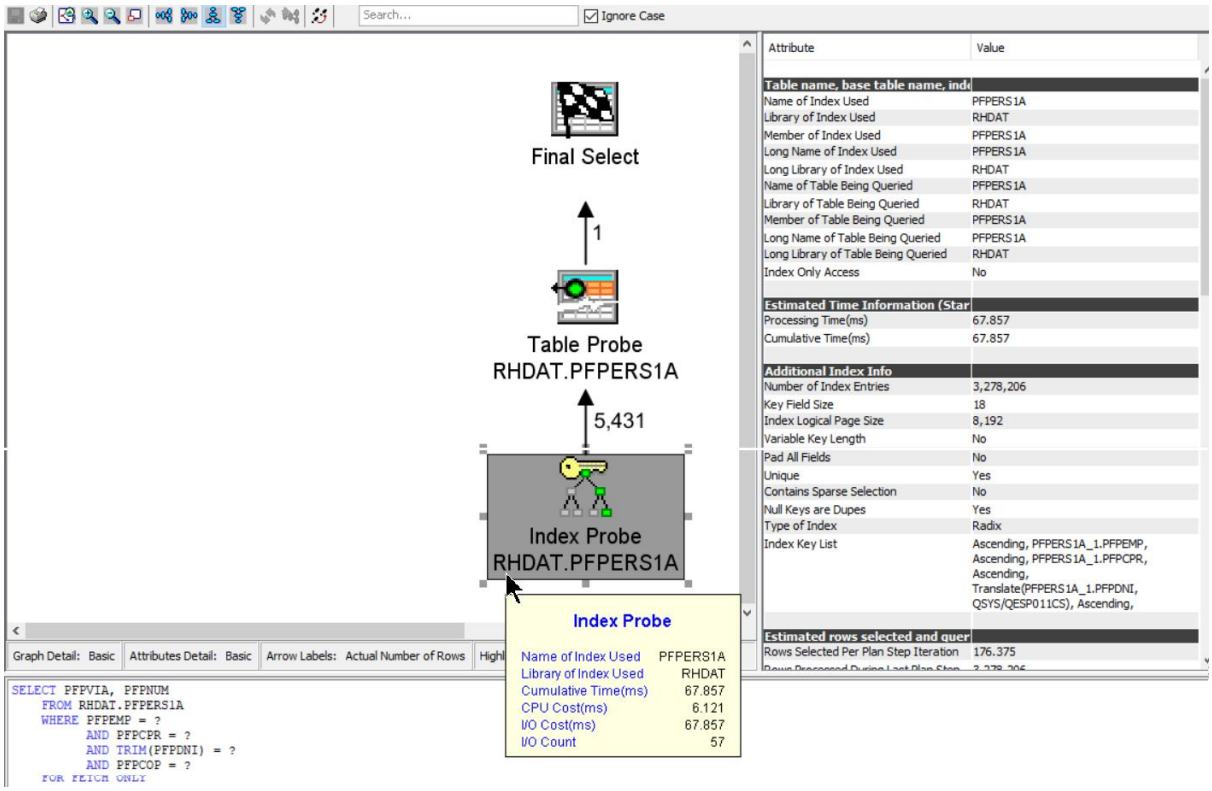


You can also use the Zoom in and Zoom out icons to fit the graph best possible for you.

After the preparation, you will see a Visual Explain graph like the following:

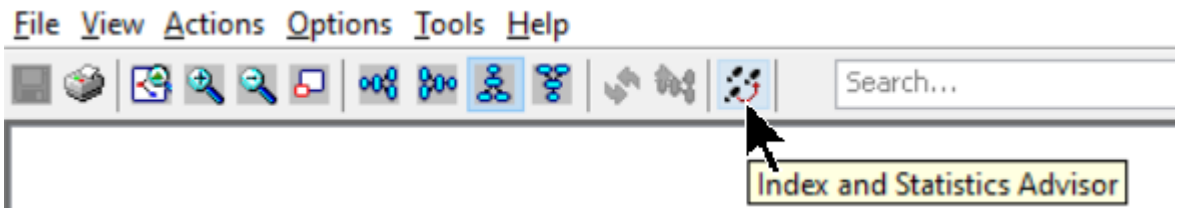


By clicking the icon for the Index Probe, the text on the right side informs about the index.

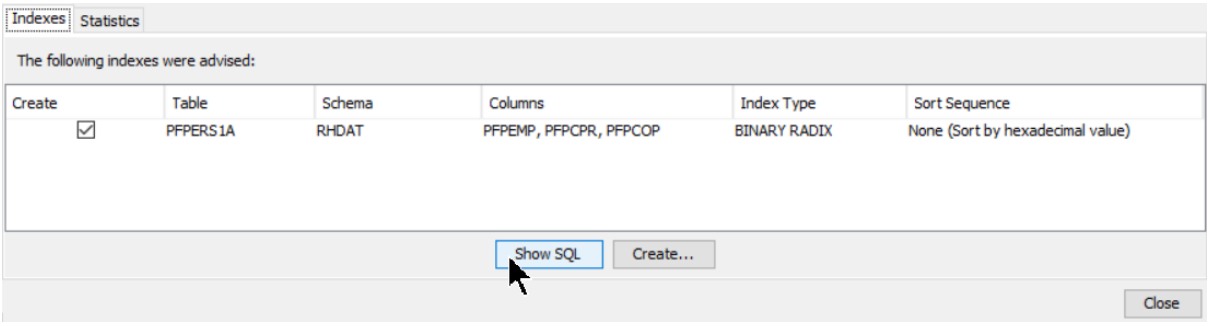


By hovering over the icon, you will see information about run time.

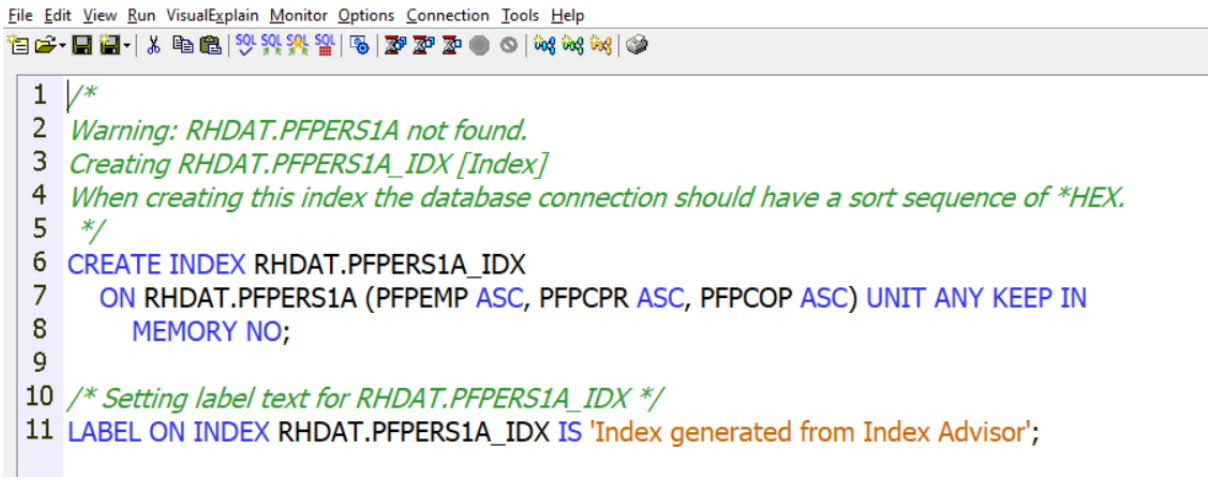
If you click the food steps icon, you will see the index advised:



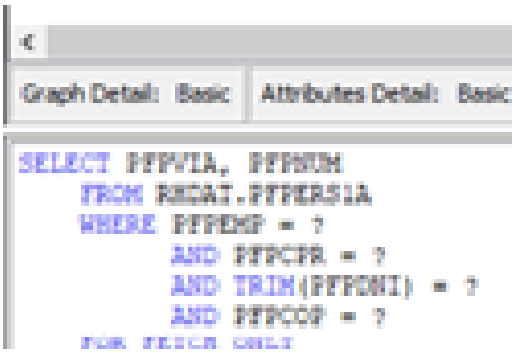
This SQL has one index recommended. You can see the SQL to create the index by clicking the Show SQL bottom:



Here you have the text for the index recommended. This is often a great documentation for you and what you will suggest creating.



You can also see that the recommended index has 3 columns. That is the same 3 columns that you can see after the WHERE in the SQL request having an equal (=):



No column in the index is suggested for the TRIM(PFPDNI), but only for the equals.

Be aware that you are not analyzing the SQL request on the partition where it was running, and the library are also not available.

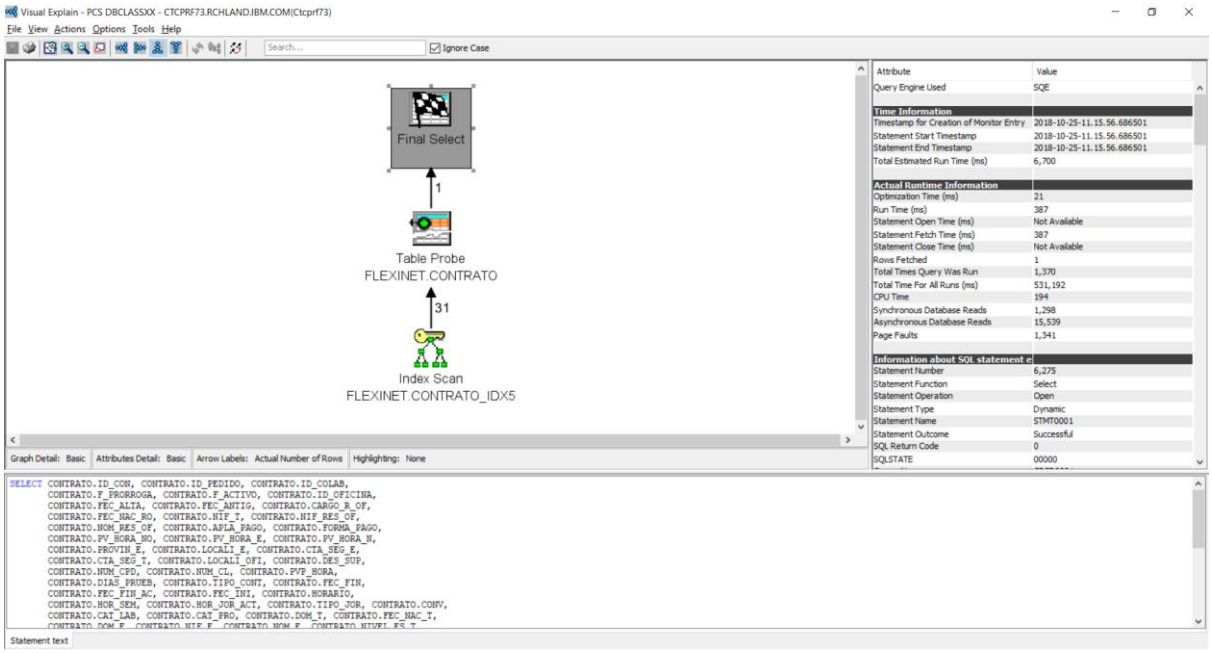
So, for the first SQL you have analyzed, you have found the index that most likely will help this query run much better. You cannot test this, but you can advise it.

This was the analysis of your first SQL request. You can close the Run SQL Scripts window and the Visual Explain and continue to the next SQL request on next page.

2. SQL

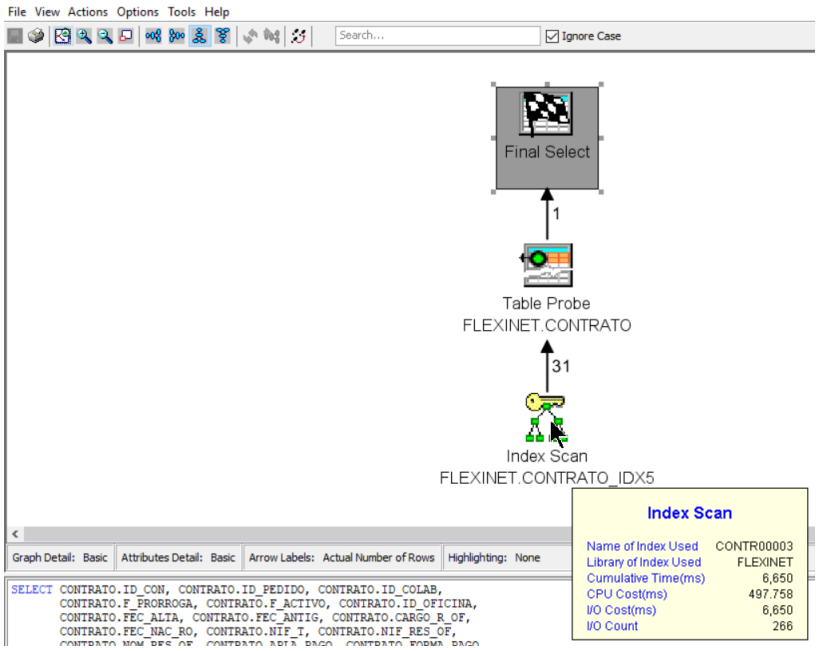
Right click the second SQL in the list and choose Visual Explain.

Click the icon for size to fit  .This should give you graph like the following:

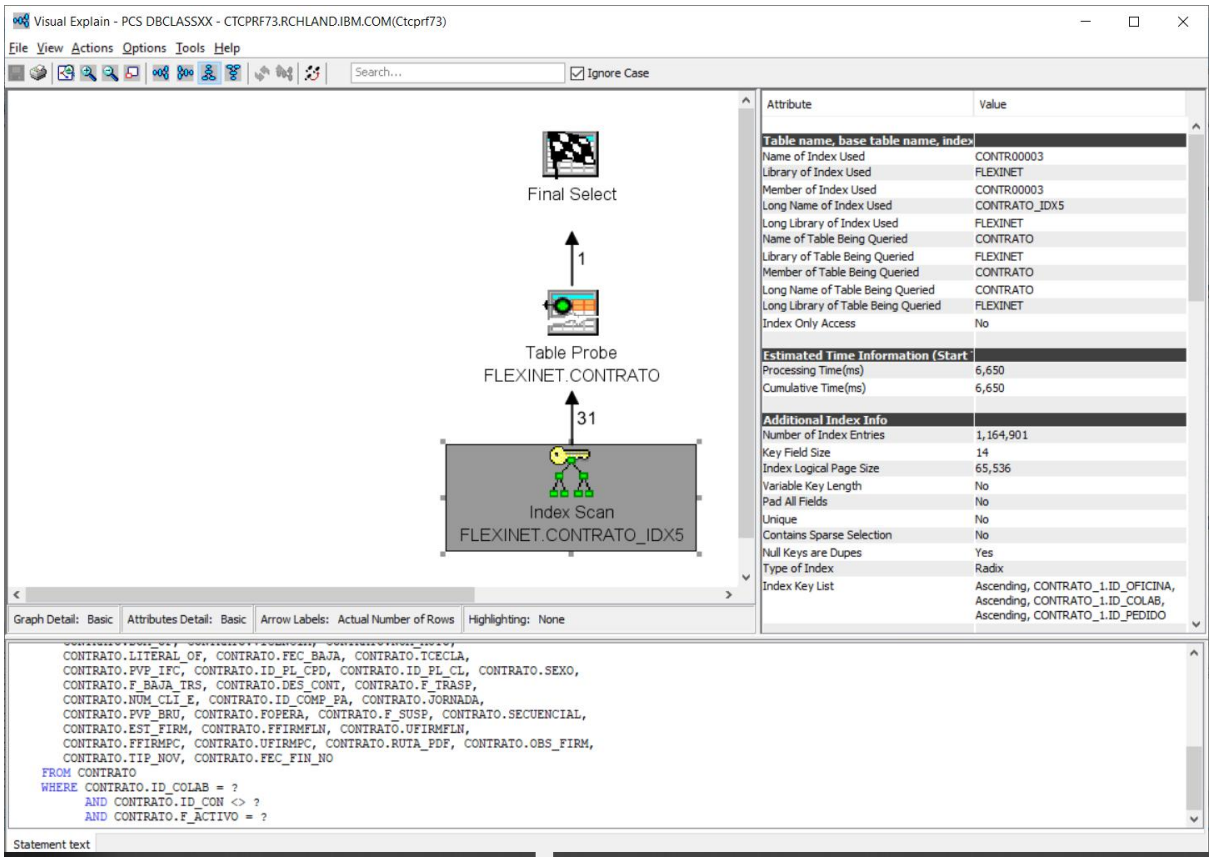


The graph may look fine. An index is used, but its an index scan, so it is going through the entire index during the run of the SQL request.

If you hover over the index icon, you will see information about the I/O use and CPU use:

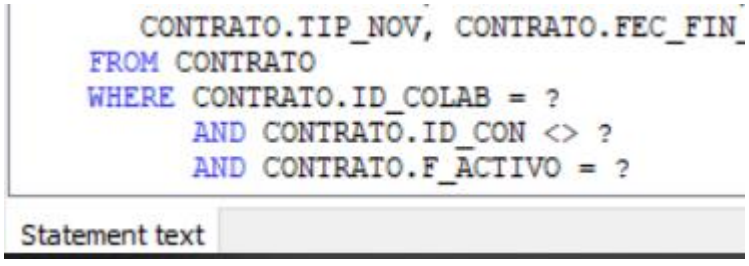


Click the index icon, then you get interesting information on the right side. You can see that the number of index entries is 1.164.901, so good information because you now see that it not efficient to scan through over 1 million entries to get 31 entries out and then have to read the table for the remaining information.

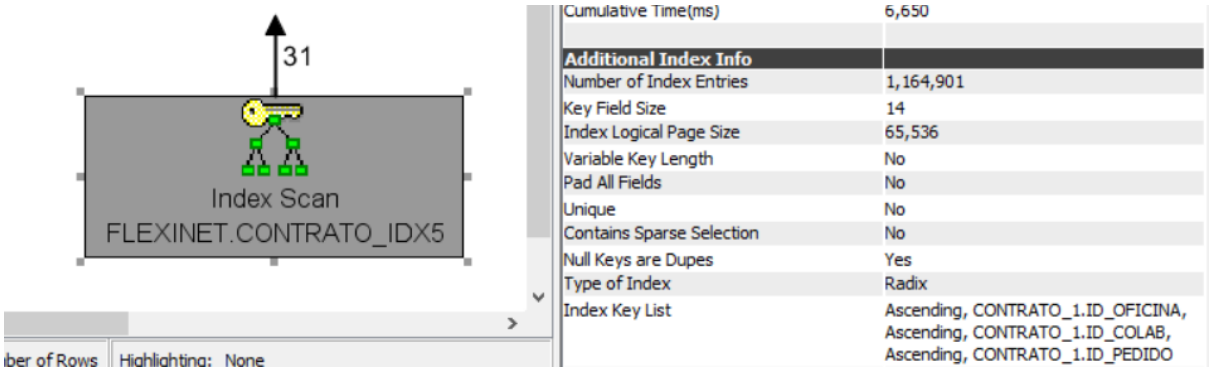


If you adjust the right window between Attributes and Value, you can get a nice list of the Index Key List. This is important to understand the index used.

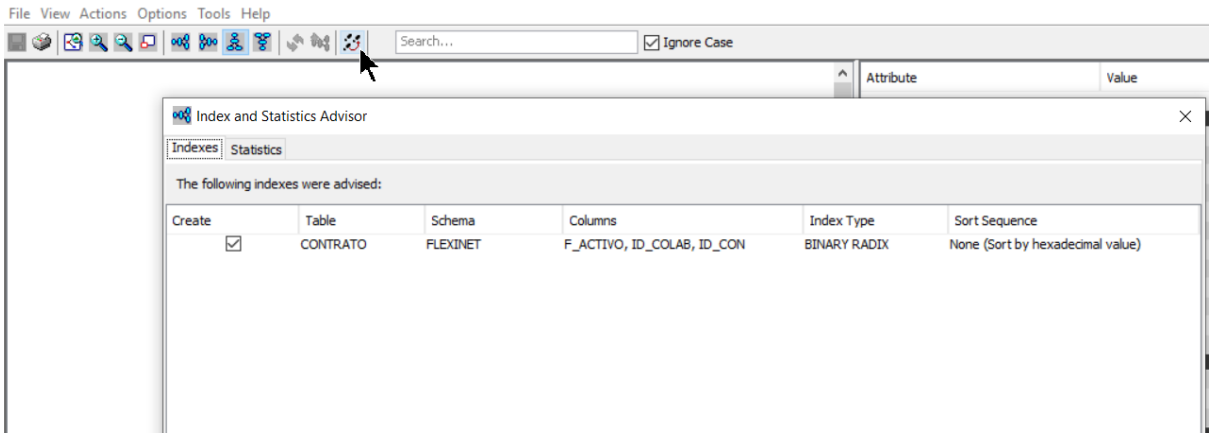
The first column in the index is ID_OFICINA, but if you look at the local selection (it is what comes after the WHERE in the SQL request), you do not see the ID_OFICINA:

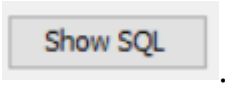


This is causing the index scan.



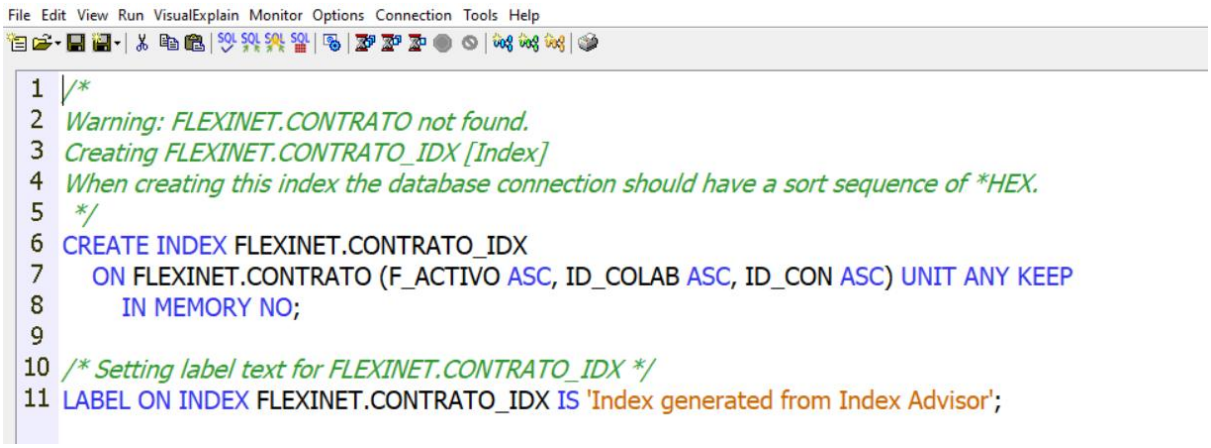
If you click the food steps, you will see the index advised:



You can generate the SQL request to create the index at this button 

.

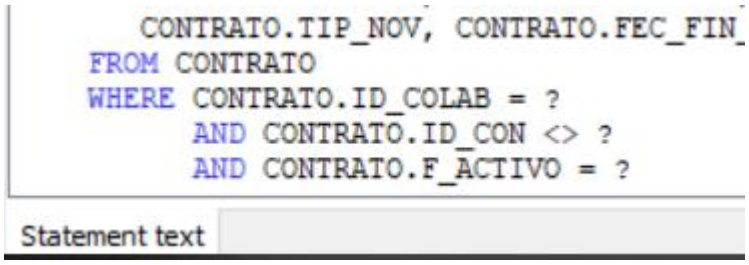
Here you have the SQL to create the recommended index:



The recommended index has the following columns:

F_ACTIVO, ID_COLAB, ID_CON

The first two columns are coming from the equal (=) and the last column is coming from the not equal <>):

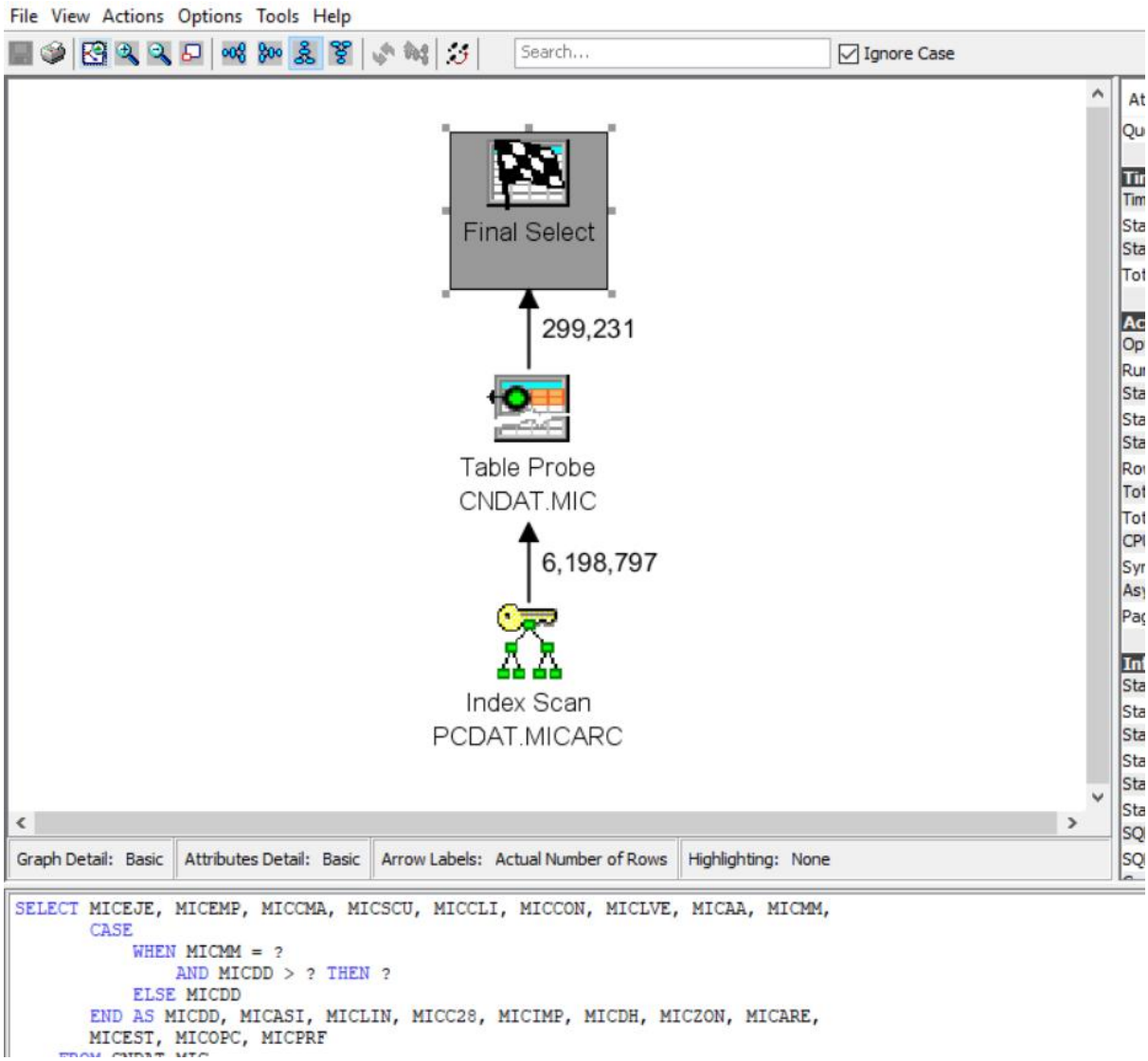


So, the recommended index will most likely help this SQL to run much more efficient.

You can close the Visual Explain and continue to the next SQL request on the list.

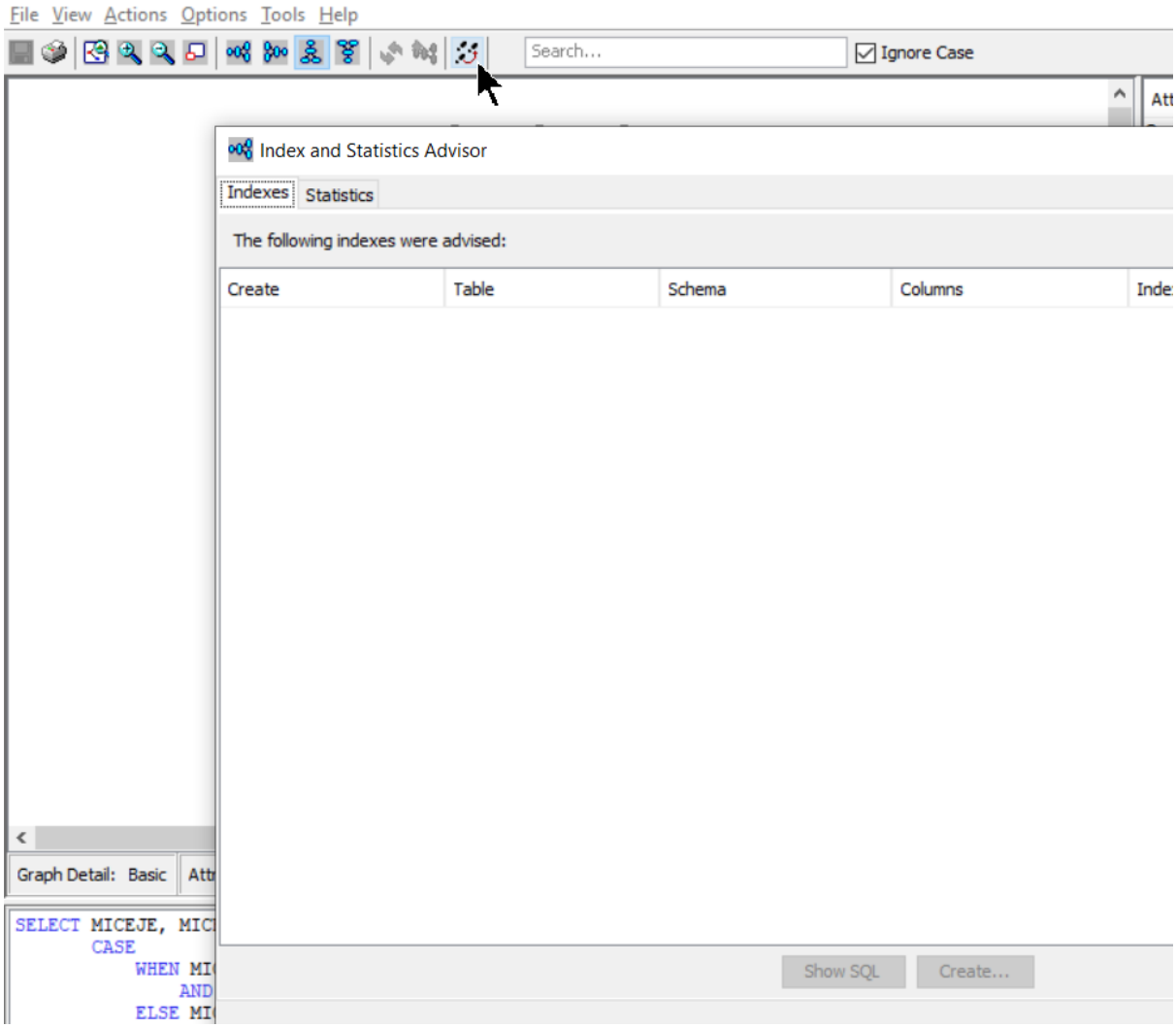
3. SQL

Again, you should right the SQL statement and choose Visual Explain. This will give you a graph like the following:



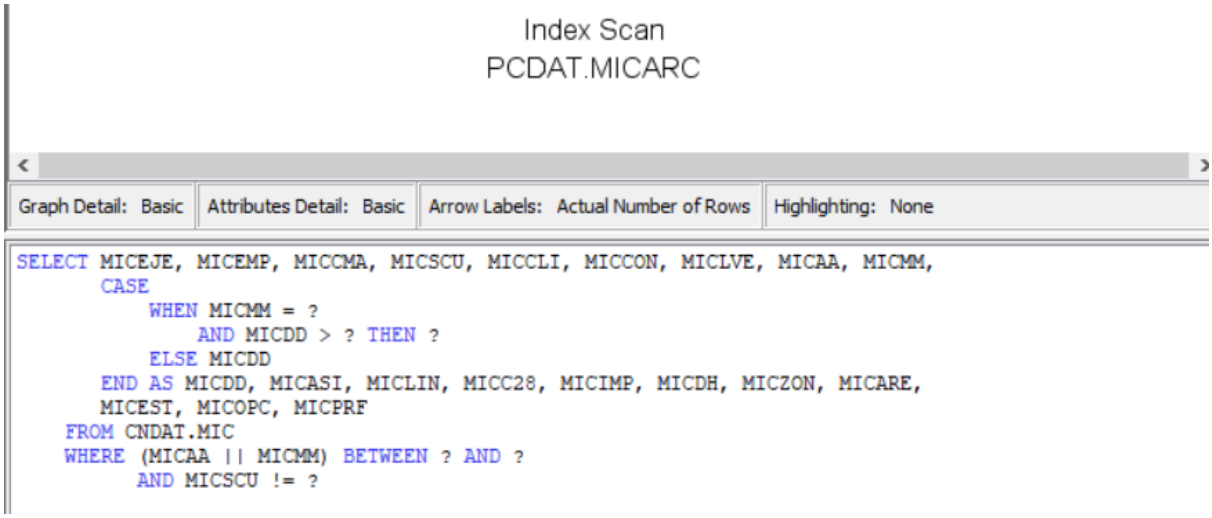
Here you see an index scan and that is seldom the best.

Click the food steps icon to check for index advised:

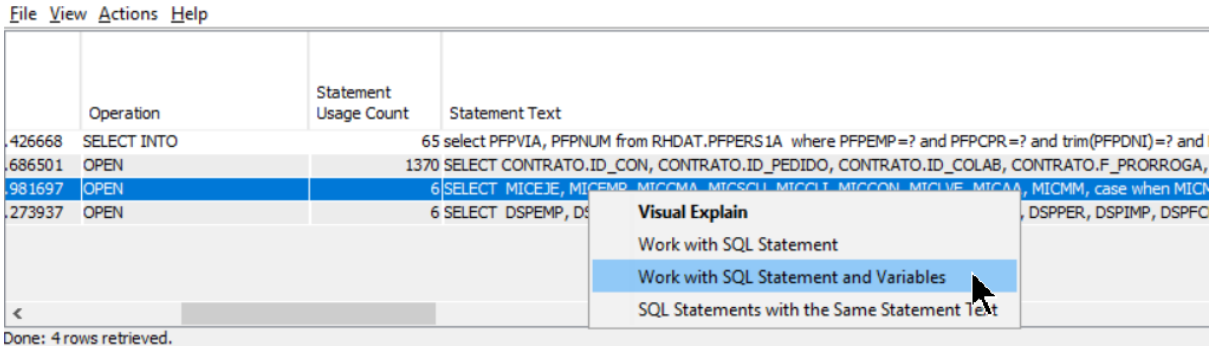


Unfortunately, you have no index recommendations, so now it begins to be more advanced, but hold on, you can do it...

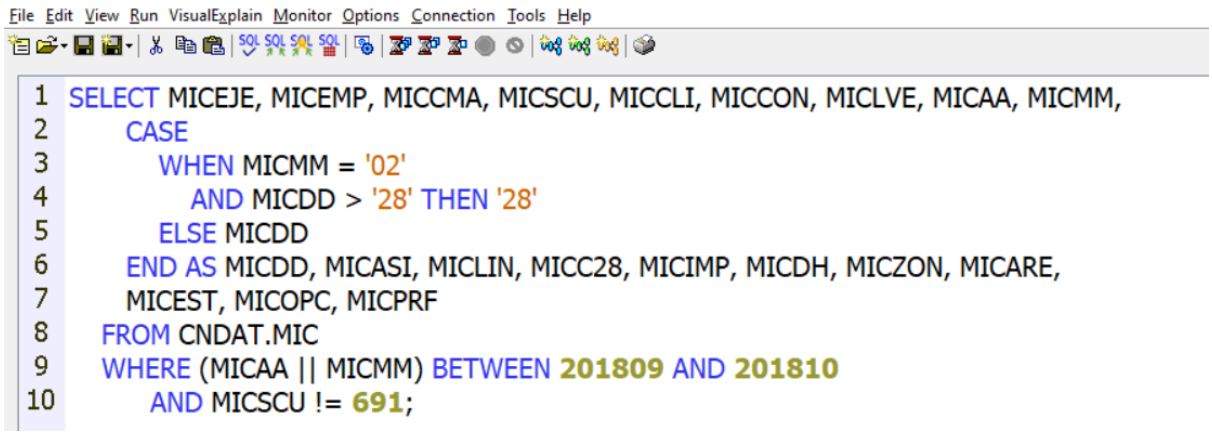
It can be difficult to fully understand what is going on, when you see all the host variables, also called parameter markers:



A good help is to see the actual variables. You can right click the list of SQL statements and this time choose the **Work with SQL Statement and Variables**:



This is much better to read:



Here you have the SQL request in text:

```
SELECT MICEJE, MICEMP, MICCMA, MICSCU, MICCLI, MICCON, MICLVE, MICAA, MICMM,
    CASE
      WHEN MICMM = '02'
        AND MICDD > '28' THEN '28'
      ELSE MICDD
    END AS MICDD, MICASI, MICLIN, MICC28, MICIMP, MICDH, MICZON, MICARE,
    MICEST, MICOPC, MICPRF
  FROM CNDAT.MIC
  WHERE (MICAA || MICMM) BETWEEN 201809 AND 201810
     AND MICSCU != 691;
```

You are looking at the SQL and realize that MICMM is month and MICAA is year. The query selects the information in a way where the DB can NOT do selection via an index

So, the query wants all rows from September and October in 2018. So, it would be possible to make a selection, where an index could be used.

As this was a SQL request that should run faster, you should have a talk with the developer and suggest that he/she tries to access data that exist in the DB and can be indexed.

He could adjust the program and request September and October in 2018, the following way:

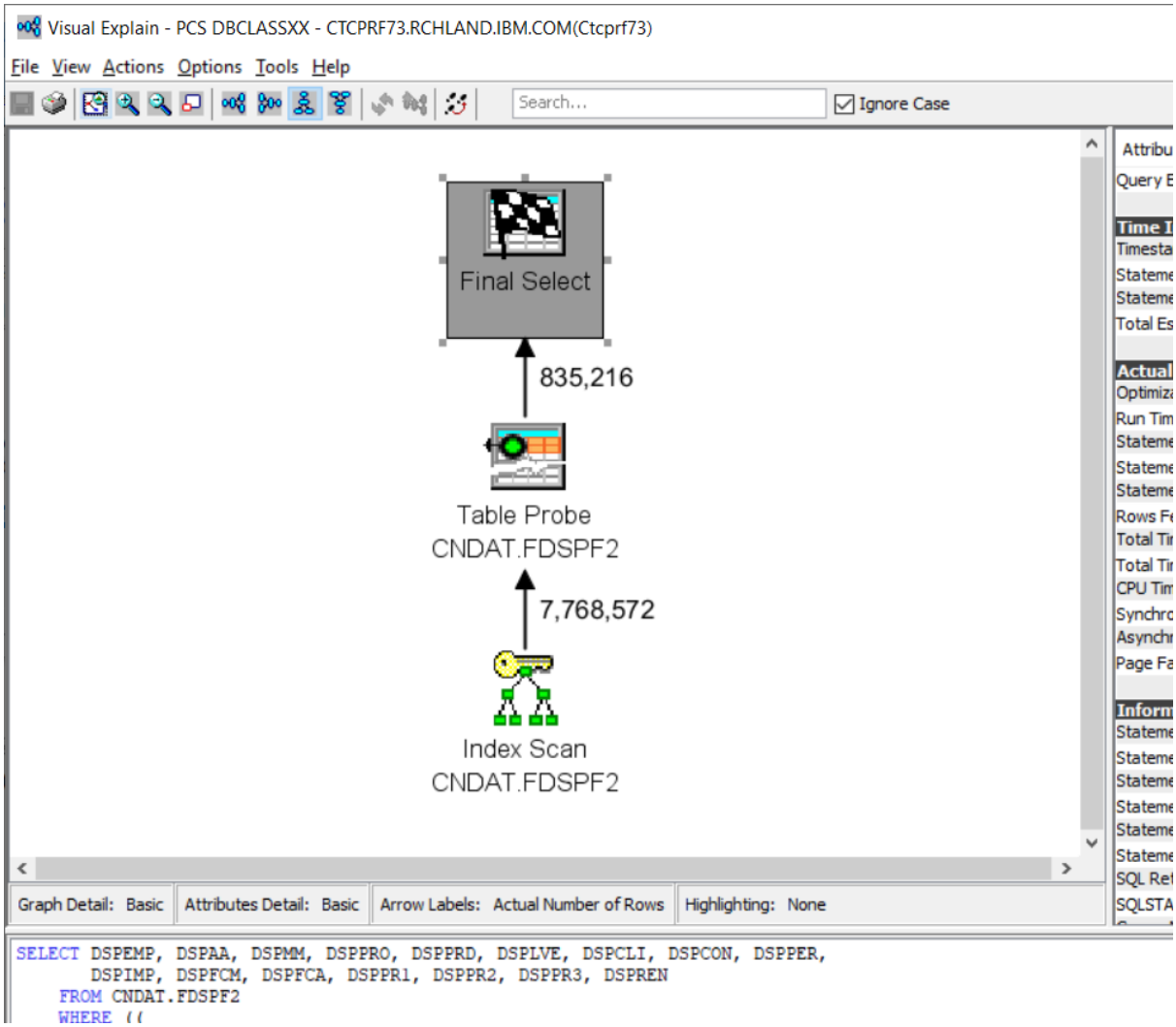
```
SELECT MICEJE, MICEMP, MICCMA, MICSCU, MICCLI, MICCON, MICLVE, MICAA, MICMM,
    CASE
      WHEN MICMM = '02'
        AND MICDD > '28' THEN '28'
      ELSE MICDD
    END AS MICDD, MICASI, MICLIN, MICC28, MICIMP, MICDH, MICZON, MICARE,
    MICEST, MICOPC, MICPRF
  FROM CNDAT.MIC
  WHERE MICAA = 2018 and MICMM in (09, 10)
     AND MICSCU != 691;
```

With an index over MICAA and MICMM, plus the MICSCU for the not equal, you will be able to have a SQL running much better.

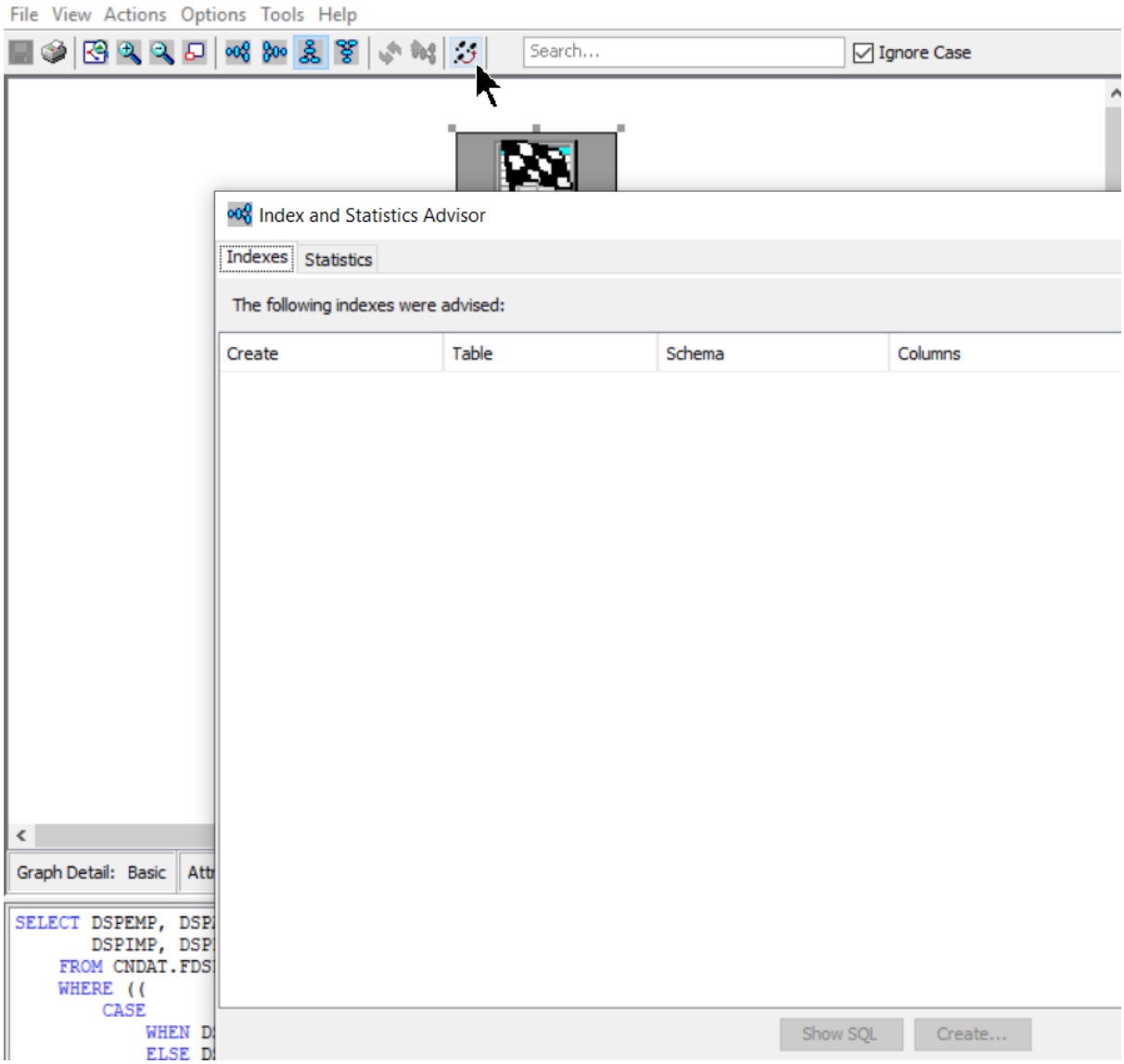
Close the Visual Explain and the Run SQL window and continue to the next SQL on the list.

4. SQL

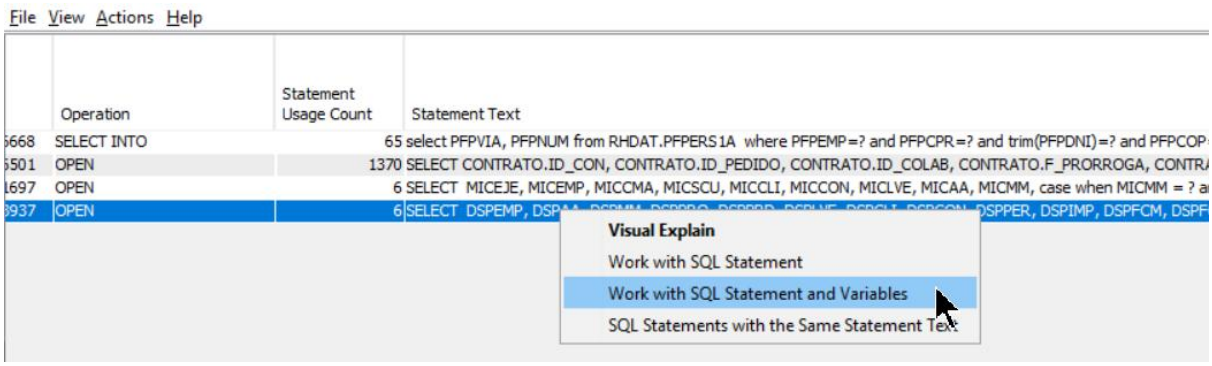
Again, you should begin by right click and choose Visual Explain. You will get a graph like the following:



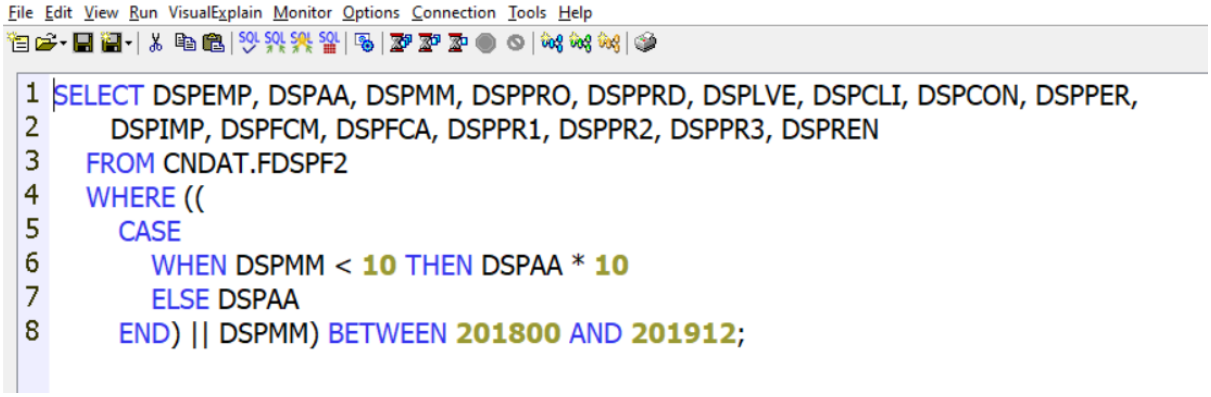
You check for recommended indexes by clicking the food steps icon:



Unfortunately, no indexes are recommended. To be able to have a better view, look at SQL statement and variables:



You can observe that the SQL request is a bit fancy having the CASE WHEN, so you will need to try to understand what is going on.




```
SELECT DSPEMP, DSPAA, DSPMM, DSPPRO, DSPPRD, DSPLVE, DSPCLI, DSPCON, DSPPER,
    DSPIMP, DSPFCM, DSPFCA, DSPPR1, DSPPR2, DSPPR3, DSPREN
  FROM CNDAT.FDSPF2
  WHERE ((
    CASE
      WHEN DSPMM < 10 THEN DSPAA * 10
      ELSE DSPAA
    END) || DSPMM) BETWEEN 201800 AND 201912;
```

You are looking at the SQL and realize that DSPMM is month and DSPAA is year. The query selects the information in a way where the DB cannot do selection via an index

So, the query wants all rows from the years 2018 and 2019. So, it would be possible to make a selection where an index could be used.

You could make a selection without the month, like the following example:

```
SELECT DSPEMP, DSPAA, DSPMM, DSPPRO, DSPPRD, DSPLVE, DSPCLI, DSPCON, DSPPER,
    DSPIMP, DSPFCM, DSPFCA, DSPPR1, DSPPR2, DSPPR3, DSPREN
  FROM CNDAT.FDSPF2
  WHERE DSPAA IN (2018,2019);
```

Again, you have some input for the discussion with the developer. You can explain that no indexes can be used, so its important to re-write the SQL request, so indexes can be used.

Close the visual explain window and Run SQL script window. Do not save the data.

End of exercise - Congratulations!

