Lab Exercise: **Visual Explain Advanced**

Speaker Name: **Morten Buur Rasmussen**

Introduction:

This exercise covers SQL scenarios where you are using Visual Explain. The session is a bit more advanced and you will see examples where the advised indexes do not help you, but you will be guided though drilling down into the SQL requests and find indexes that can still help in SQL to run better.

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

You will type some SQL request in during this lab. To be able to save some of your time, you should be able to find a text document on the laptop with the SQL requests, so you can copy from there. The text file is **Visual Explain Advanced lab - SQL requests**.

Please be aware that the Visual Explain graphs can be different from the graphs in this lab hand-out. This can be caused by different configurations etc.

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:

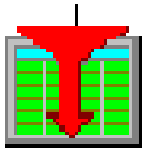
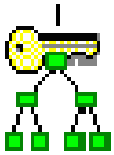
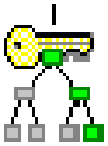


Table Scan



Index Scan



Index Probe

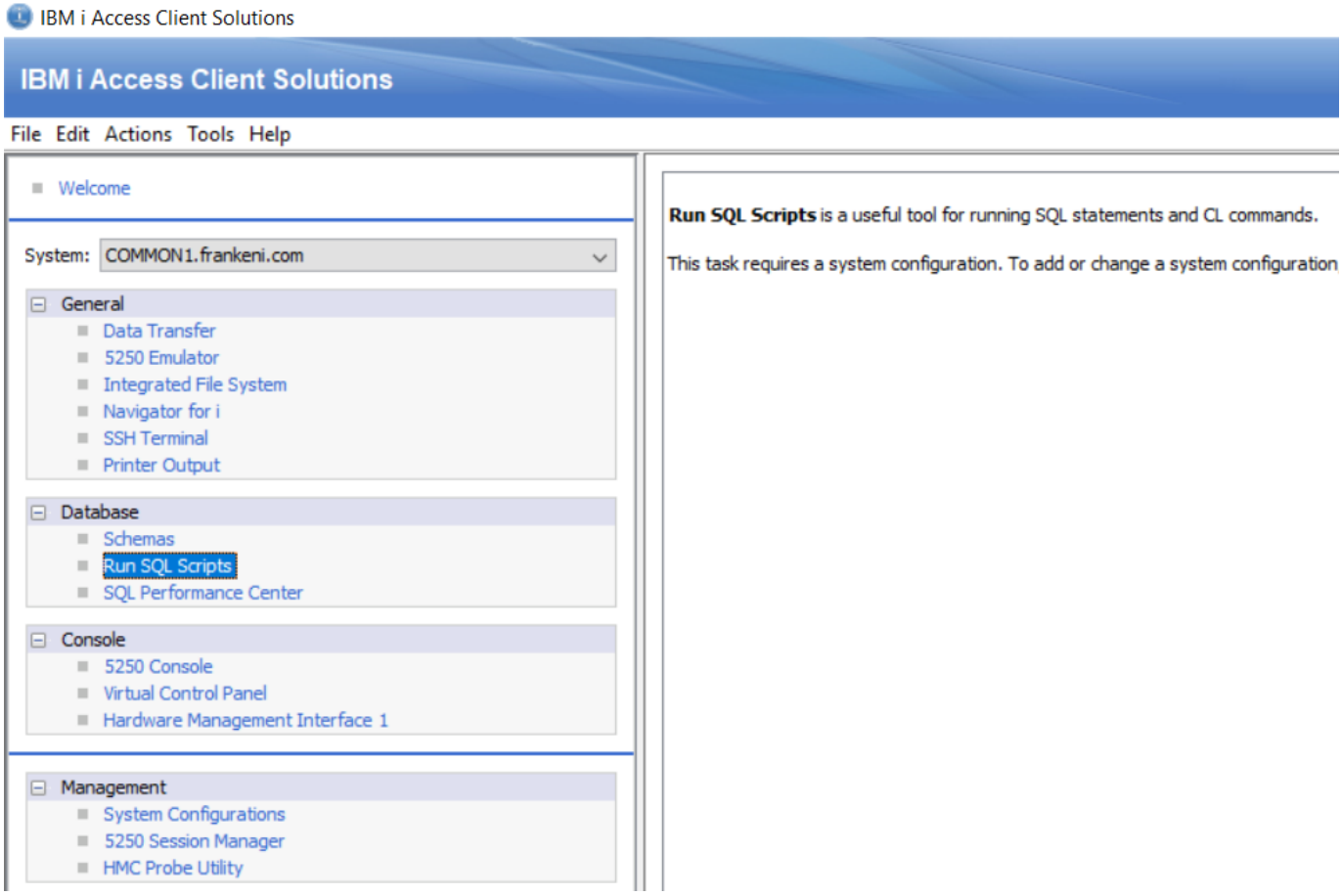
The reason for looking at the access method is that often is the table scans and index scans not very efficient, so it's better, if you can have the query use an index probe.

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 good.

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.

Using the document containing the list of SQL statements provided by the instructor, review and analyse the queries. Identify the access methods and strategies using Visual Explain. You will need a **Run SQL script** to be open, like the following:



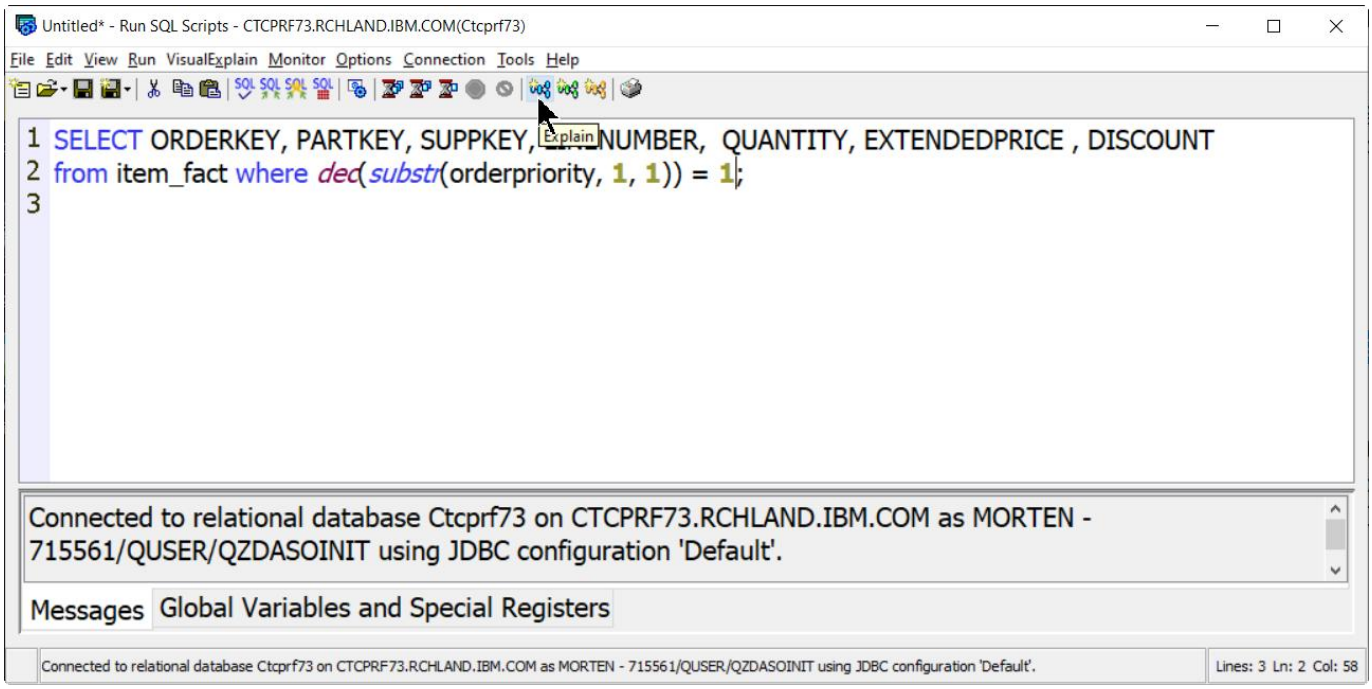
Lab 01 – Visual Explain Advanced – Example 1

Using the document containing the list of SQL statements provided by the instructor, review and analyse the following queries. Identify the access methods and strategies using Visual Explain.

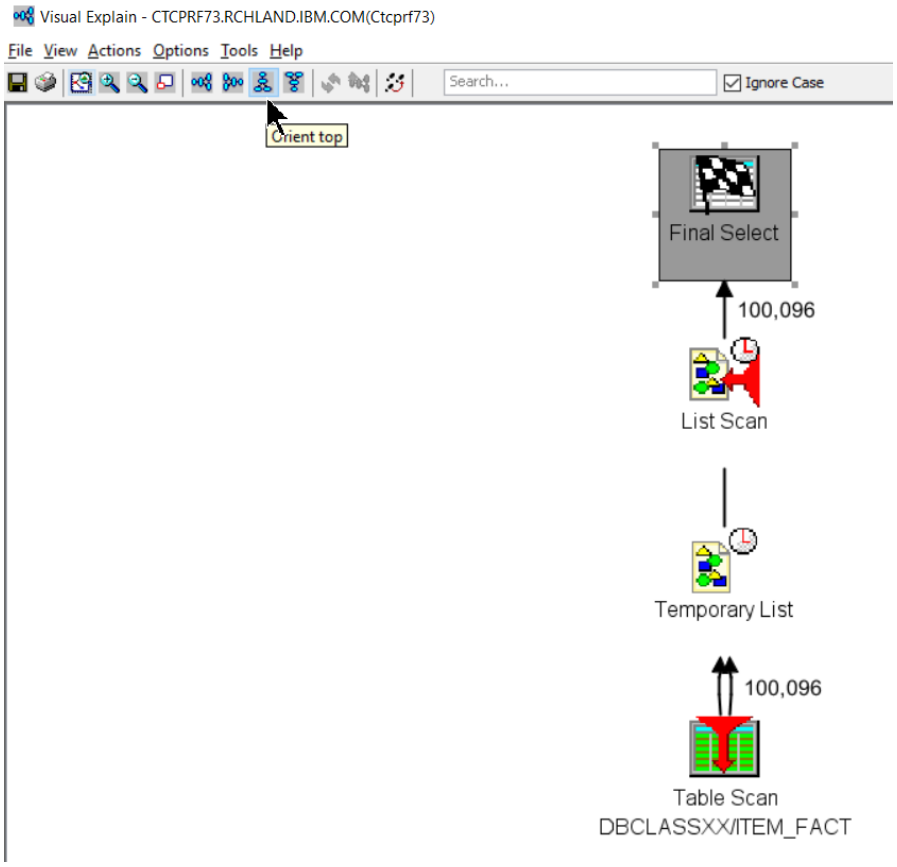
Open the Run SQL Script window if no window is currently open.

Run the following SQL statement using Visual Explain (**Explain only**):

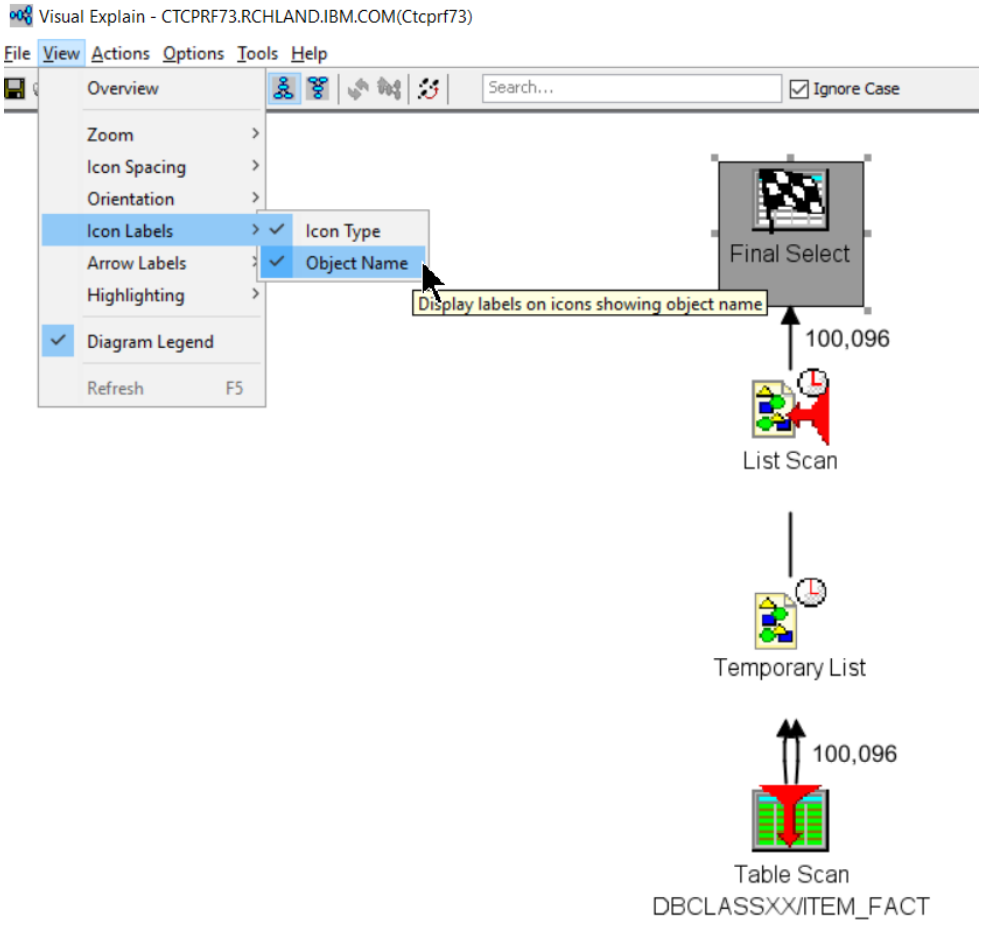
SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT
from item_fact where dec(substr(orderpriority, 1, 1)) = 1;



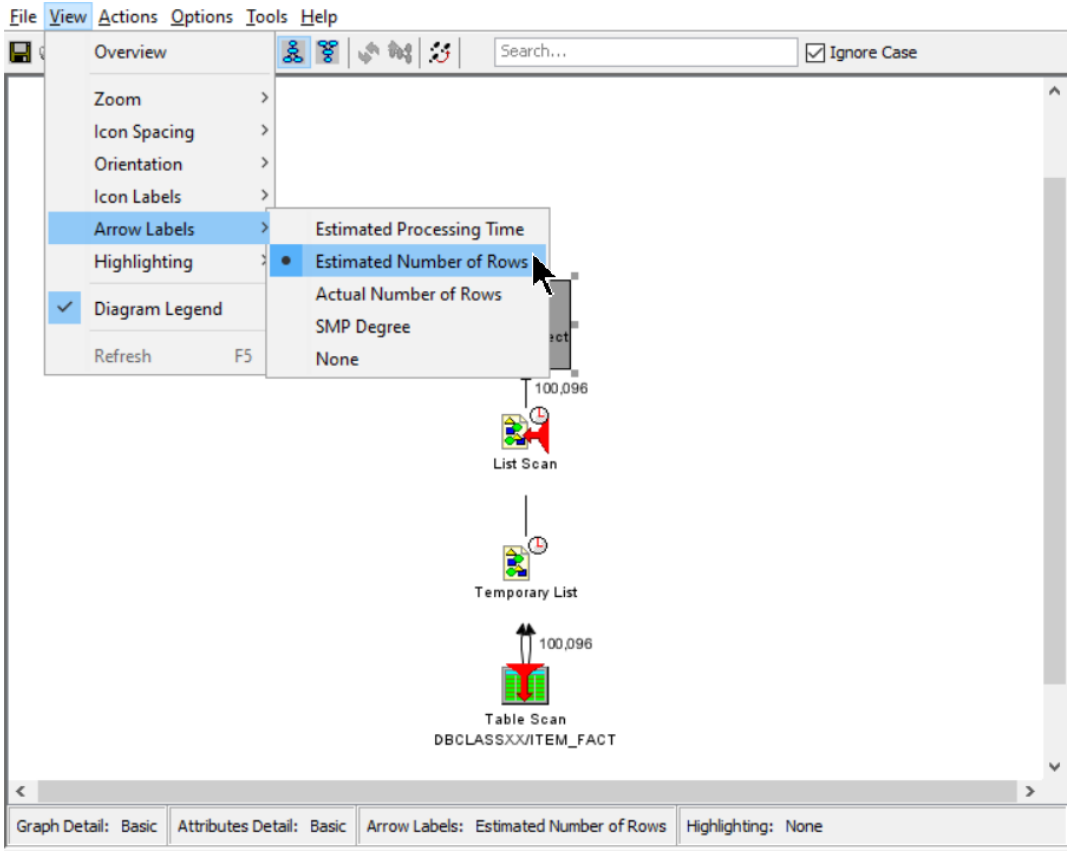
In the Visual Explain click the Orient top icon, so you get the tree structure. It may look like the following graph:



Make sure you have both the Icon Type and Object Name ticket **on** in the Icons Labels, under View:



As we are analysing the queries without running them, but using the Visual Explain feature to Explain only, you will need to make sure you are seeing the estimated number of rows. You can set this up here:



Now it is time to start working :o)

What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What indexes could be created to assist the optimizer and database engine with local selection?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

What else can be done to make the query run better?

Run the following SQL statement to create the radix indexes:

CREATE INDEX ITEM_IX3 ON ITEM_FACT (ORDERPRIORITY);

Rerun the following SQL statement using Visual Explain (explain only):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG from item_fact where dec(substr(orderpriority, 1, 1)) = 0;

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better?

Run the following SQL statement to create the radix indexes:

CREATE INDEX ITEM_IX4 ON ITEM_FACT
(DEC(SUBSTR(ORDERPRIORITY, 1, 1)) as ORDPTY);

Rerun the following SQL statement using Visual Explain (explain only):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG from item_fact where dec(substr(orderpriority, 1, 1)) = 0;

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better?

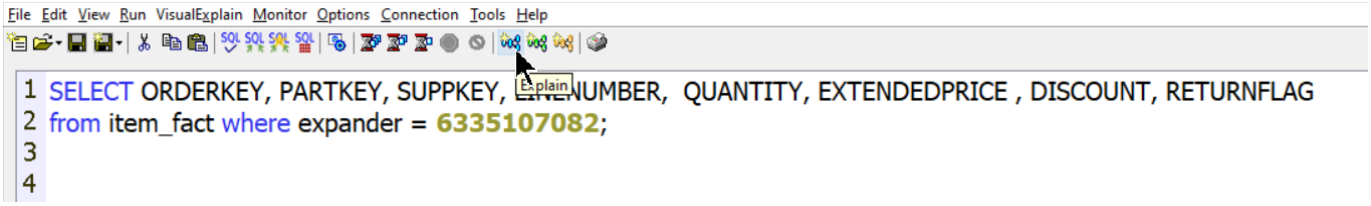
Lab 01 – Visual Explain Advanced – Example 2

Using the document containing the list of SQL statements provided by the instructor, review and analyse the following queries. Identify the access methods and strategies using Visual Explain.

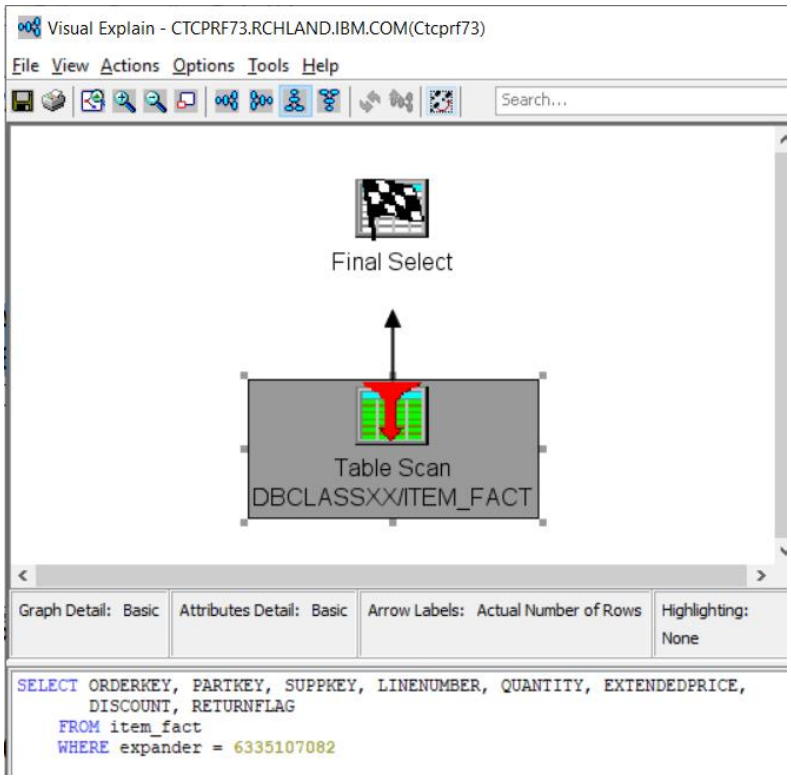
Open the Run SQL Script window if no window is currently open.

Run the following SQL statement using Visual Explain (**Explain only**):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG
from item_fact where expander = 6335107082;



The Visual Explain could show you something like this graph:



What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What indexes could be created to assist the optimizer and database engine with local selection?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

What else can be done to make the query run better?

Run the following SQL statement to create the radix indexes:

CREATE INDEX ITEM_IX31 ON ITEM_FACT (EXPANDER);

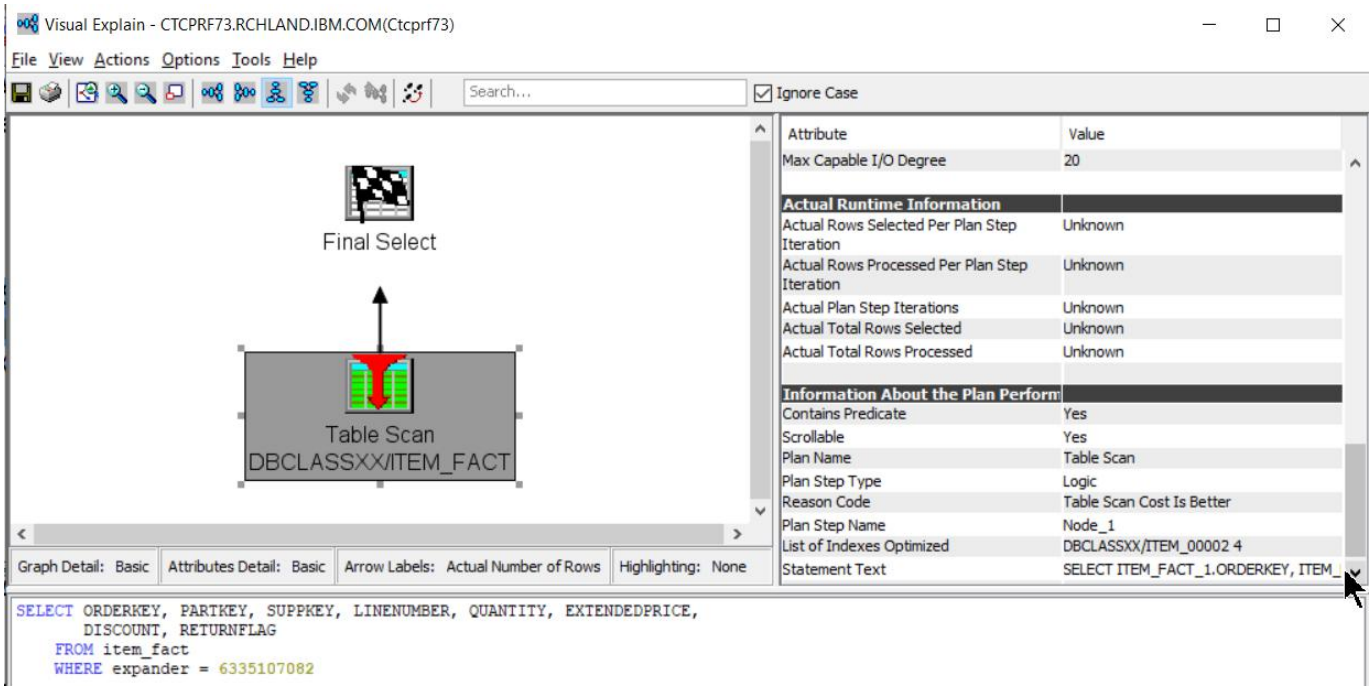
Rerun the following SQL statement again, and still using Visual Explain (explain only):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG
from item_fact where expander = 6335107082;

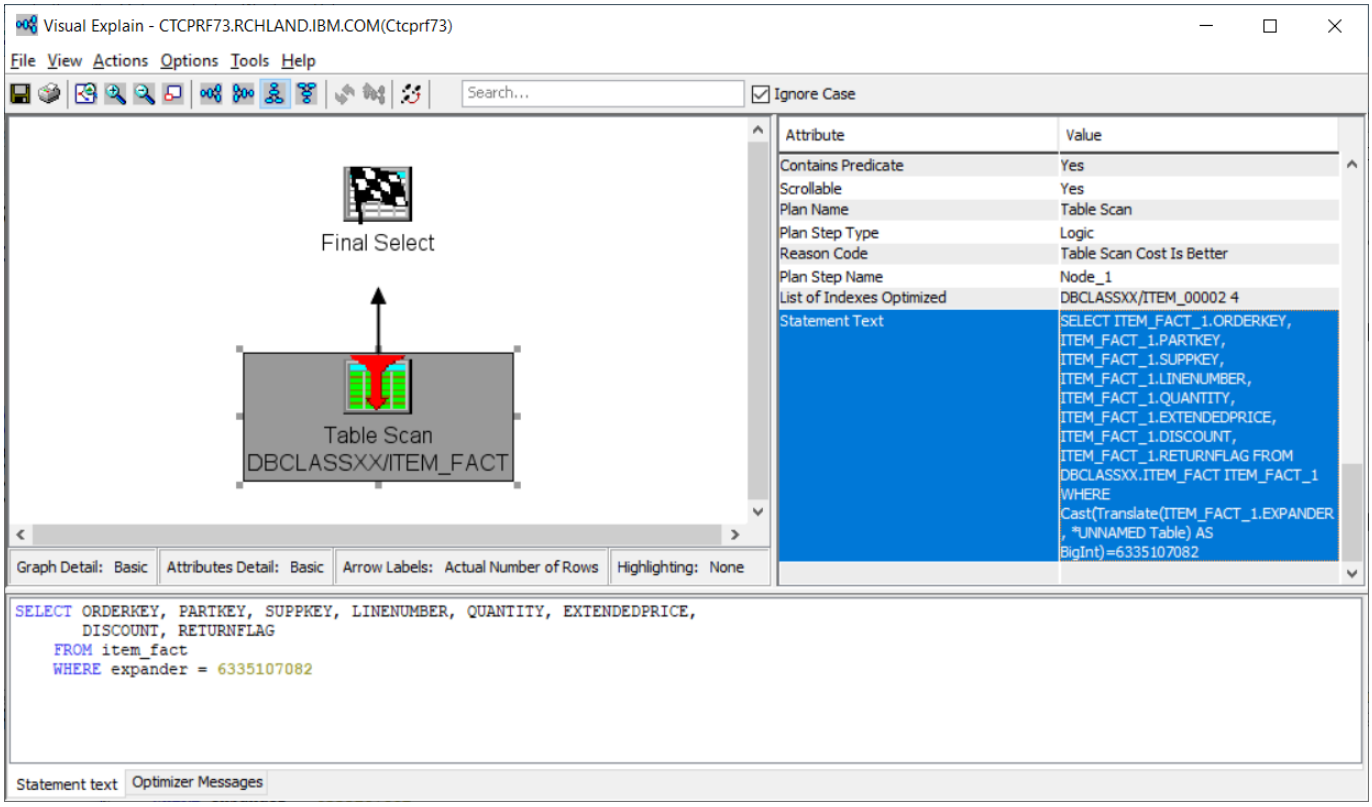
From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better?

Click the table scan icon and drag the scrollbar to the bottom on the right side with the text information. Like this:



If the Statement Text contains one line, then double click the line and you will have a picture like the following:



The value on the Statement Text is not SQL, but a pseudo language like SQL. It gives us some insight to what is going on.
So, we can see what is going on during the table scan. We can see on the local selection what is tested for:

WHERE Cast(Translate(ITEM_FACT_1.EXPANDER, *UNNAMED Table) AS BigInt)=6335107082

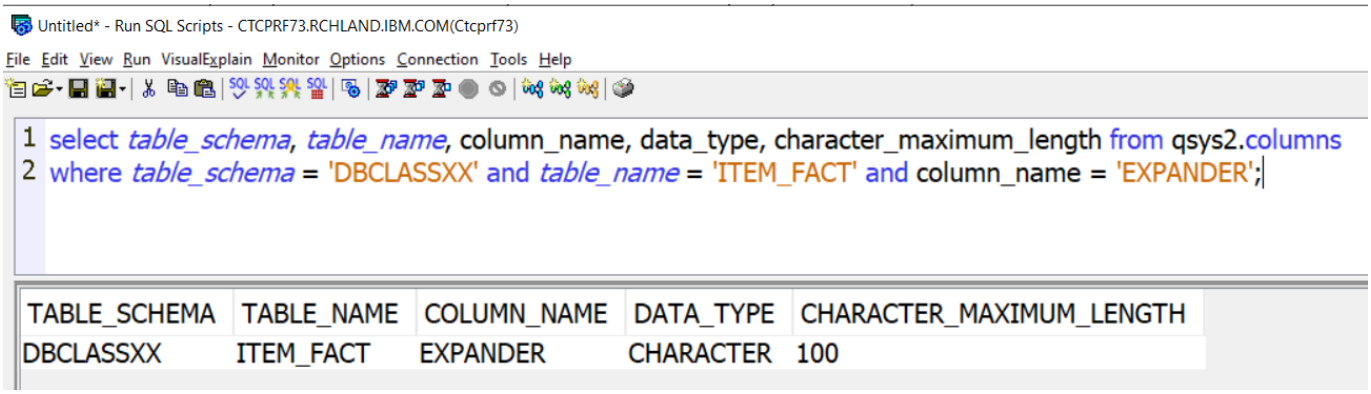
So, it takes all rows in the ITEM_FACT table and translate the EXPANDER column to BigInt to be able to compare to the value in the SQL request.

This is obviously not efficient, but why is this happening?

Substitute the XX to your number in the library DBCLASSXX and run the following query:

```
select table_schema, table_name, column_name, data_type,
character_maximum_length from qsys2.columns
where table_schema = 'DBCLASSXX' and table_name = 'ITEM_FACT'
and column_name = 'EXPANDER';
```

You should get a similar result as you can see here:



What does the data type tell you?

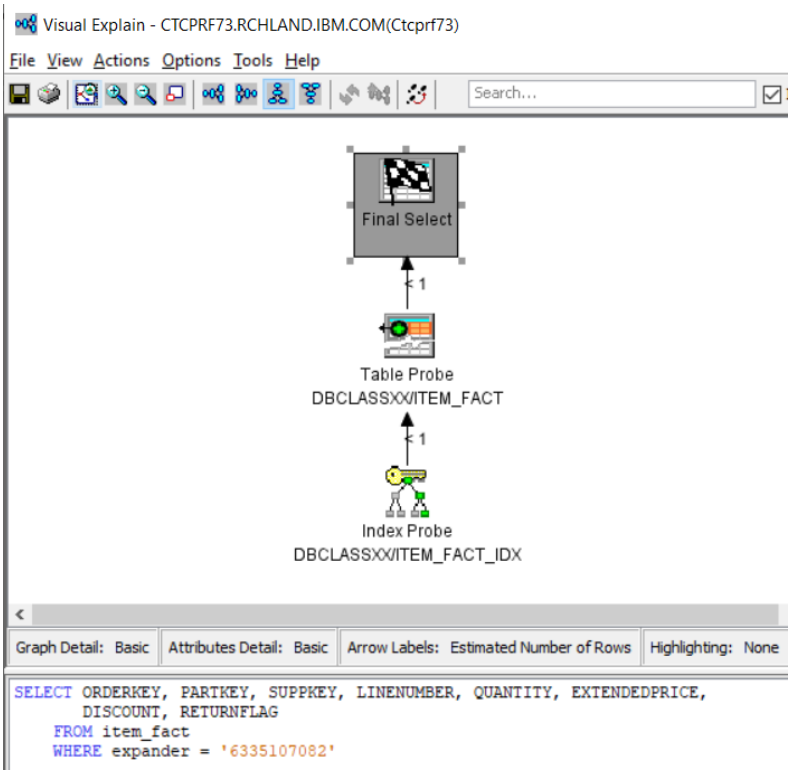
This issue here is often seen in application programs assuming that the data has a numeric column, but it’s actually a character.
You can easily prove this by running the query with a character string.

Run the following SQL statement using Visual Explain (**Explain only**):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG
from item_fact where expander = '6335107082';

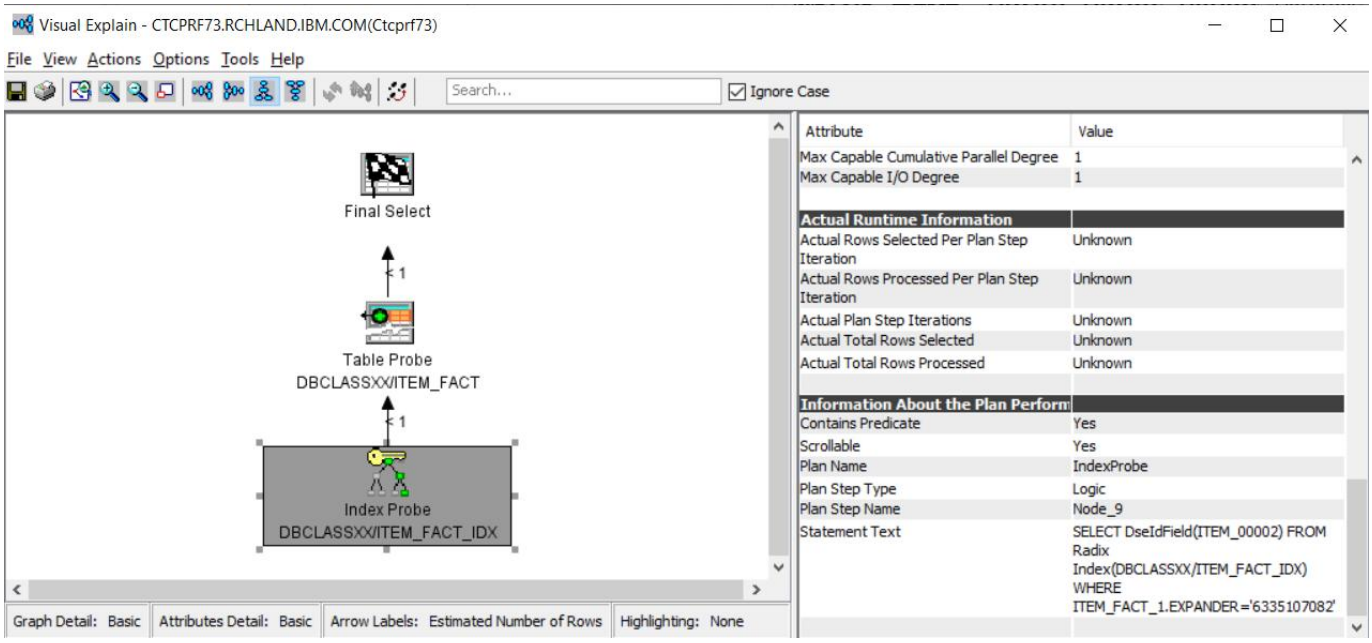
Do you like the outcome?

You probably got a graph like the following:



Please double check that no translation is taking place.

Do this by clicking the Index Probe icon and drag the scrollbar to the bottom on the right side with the text information. Like this:



Lab 01 – Visual Explain Advanced – Example 3

Run the following SQL statement using Visual Explain (explain only):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG from item_fact where TRIM(orderpriority) = '0-VIP';

What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What indexes should be created to assist the optimizer and database engine with local selection?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

What else can be done to make the query run better?

Run the following SQL statement to create the radix indexes:

CREATE INDEX ITEM_IX5 ON ITEM_FACT (TRIM(ORDERPRIORITY) as ORDPTYTRIM);

Rerun the following SQL statement using Visual Explain (explain only):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG from item_fact where TRIM(orderpriority) = '0-VIP';

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better?

Lab 01 – Visual Explain Advanced – Example 4

Run the following SQL statement using Visual Explain (explain only):

```
SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER,  QUANTITY,
EXTENDEDPRICE , DISCOUNT
from item_fact where substr(orderpriority, 3, 3) = 'VIP';
```

What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What indexes should be created to assist the optimizer and database engine with local selection?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

What else can be done to make the query run better?

Run the following SQL statement to create the radix indexes:

CREATE INDEX ITEM_IX6 ON ITEM_FACT ((substr(orderpriority, 3, 3)) as ORDPTYshrt);

Rerun the following SQL statement using Visual Explain (explain only):

SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER, QUANTITY, EXTENDEDPRICE , DISCOUNT
from item_fact where substr(orderpriority, 3, 3) = 'VIP';

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better?

Close the visual explain window and do not save the data.

Lab 01 – Visual Explain Advanced – Example 5

Run the following SQL statement using Visual Explain (explain only):

```
SELECT count( * ) FROM ITEM_FACT
WHERE IFNULL ( SHIPPRIORITY , 0 ) = 1;
```

What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What indexes should be created to assist the optimizer and database engine with local selection?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

What else can be done to make the query run better?

Run the following SQL statement to create the radix index:

CREATE INDEX ITEM_IX7 ON ITEM_FACT (SHIPPRIORITY);

Rerun the following SQL statement using Visual Explain (explain only):

SELECT count(*) FROM ITEM_FACT WHERE IFNULL (SHIPPRIORITY , 0) = 1;

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better and why?

Run the following SQL statement to create the encoded vector index:

CREATE encoded vector INDEX ITEM_IX71 ON ITEM_FACT (SHIPPRIORITY);

Rerun the following SQL statement using Visual Explain (explain only):

SELECT count(*) FROM ITEM_FACT WHERE IFNULL (SHIPPRIORITY , 0) = 1;

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better and why?

Close the visual explain window and do not save the data.

Lab 01 – Visual Explain Advanced – Example 6

Run the following SQL statement using Visual Explain (explain only):

```
SELECT YEAR, SUM(EXTENDEDPRICE)
FROM   item_fact
WHERE SHIPPRIORITY IN (1,2)
GROUP BY YEAR
ORDER BY YEAR DESC;
```

What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What indexes should be created to assist the optimizer and database engine with local selection?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

What else can be done to make the query run better?

Run the following SQL statement to create advised Indexes:

```
CREATE INDEX ITEM_IX8 ON ITEM_FACT (SHIPPRIORITY);
CREATE INDEX ITEM_IX9 ON ITEM_FACT (YEAR);
```

Rerun the following SQL statement using Visual Explain (explain only):

```
SELECT YEAR, SUM(EXTENDEDPRICE)
FROM   item_fact
WHERE SHIPPRIORITY IN (1,2)
GROUP BY YEAR
ORDER BY YEAR;
```

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better and why?

Run the following SQL statement to create an Encoded Vector Index (EVI).

```
CREATE ENCODED VECTOR INDEX item_evi1
ON item_fact (YEAR, SHIPPRIORITY)
INCLUDE(SUM(EXTENDEDPRICE))
```

Rerun the following SQL statement using Visual Explain (explain only):

```
SELECT YEAR, SUM(EXTENDEDPRICE)
FROM   item_fact
WHERE SHIPPRIORITY IN (1,2)
GROUP BY YEAR
ORDER BY YEAR;
```

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better and why?

Close the visual explain window and do not save the data.

Lab 01 – Visual Explain Advanced – Example 7

Run the following SQL statement using Visual Explain (explain only):

```
SELECT year, orderkey, shippriority
FROM item_fact
WHERE year = 2004
AND orderkey = 100;
```

What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What indexes should be created to assist the optimizer and database engine?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

What else can be done to make the query run better?

Create the recommended index (could be the following):

CREATE INDEX ITEM_IX10 ON ITEM_FACT (YEAR, ORDERKEY);

Rerun the following SQL statement using Visual Explain (explain only):

```
SELECT year, orderkey, shippriority
FROM item_fact
WHERE year = 2004
AND orderkey = 100;
```

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better and why?

Run the following SQL statement to create the radix index:

CREATE INDEX ITEM_IX11 ON ITEM_FACT (YEAR, ORDERKEY, SHIPPRIORITY);

Rerun the following SQL statement using Visual Explain (explain only):

```
SELECT year, orderkey, shippriority
FROM item_fact
WHERE year = 2004
AND orderkey = 100;
```

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Does the query run better and why?

Close the visual explain window and do not save the data.

Optional exercises - Example 8

Using the document containing the list of SQL statements provided by the instructor, review and analyse the following queries. Identify the access methods and strategies using Visual Explain.

Open the Run SQL Script window if no window is currently open.

Run the following SQL statement using Visual Explain (explain only):

```
SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER,
QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG
from ITEM_FACT
where orderdate >= (select checkdate from rbdates);
```

In the Visual Explain click the Orient top icon, so you get the tree structure.

What are the data access methods used by the Optimizer for table ITEM_FACT?

What is the reason that the Optimizer used this data access method?

What is the estimated number of records selected?

What indexes should be created to assist the optimizer and database engine with local selection?

Is the Visual Explain index advisor advising these indexes? If not, are other indexes advised?

Run the following SQL statement to create the radix indexes:

CREATE INDEX ITEM_IX1 ON ITEM_FACT (ORDERDATE);

Rerun the following SQL statement using Visual Explain (explain only):

```
select ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER,
QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG
from ITEM_FACT
where orderdate >= (select checkdate from rbdates);
```


From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

What are the data access methods used by the Optimizer for table ITEM_FACT and RBDATES?

Which indexes were used and what where they used for?

Add the OPTIMIZE FOR 1 ROW, and Rerun the following SQL statement using Visual Explain (explain only):

```
SELECT ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER,
QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG
from ITEM_FACT
where orderdate >= (select checkdate from rbdates)
OPTIMIZE FOR 1 ROW;
```

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

What are the data access methods used by the Optimizer for table ITEM_FACT?

Which indexes were used and what where they used for?

What is the estimated number of records selected?

What is the selectivity for the ITEM_FACT table?

Check the number of actual rows returned by running the following SQL:

select count(*) from ITEM_FACT where orderdate >= (select checkdate from rbdates);

What does the count return?

Is this similar to any of the estimated rows returned when running the statement before?

If you see a difference, what is the reason?

Check the value of CHECKDATE in the RBDATES table by running the following SQL:

select checkdate from rbdates;

What does the SELECT return?

Modify the SQL request to have the hard-coded value, and Rerun the following SQL statement using Visual Explain (explain only):

```
select ORDERKEY, PARTKEY, SUPPKEY, LINENUMBER,
QUANTITY, EXTENDEDPRICE , DISCOUNT, RETURNFLAG
from ITEM_FACT
where orderdate>= '2007-04-04';
```

From the Visual Explain graph and window, click the various icons and from the Attributes and Values window answer the following questions:

Which indexes were used and what where they used for?

What is the estimated number of records selected?

What is the selectivity for the ITEM_FACT table?

Is this run different? If yes, can you explain why this run is different?

Close the visual explain window and do not save the data.

End of exercise

Congratulations – You are a finisher!

