

## LAB 04 – EXPLAIN FACILITY

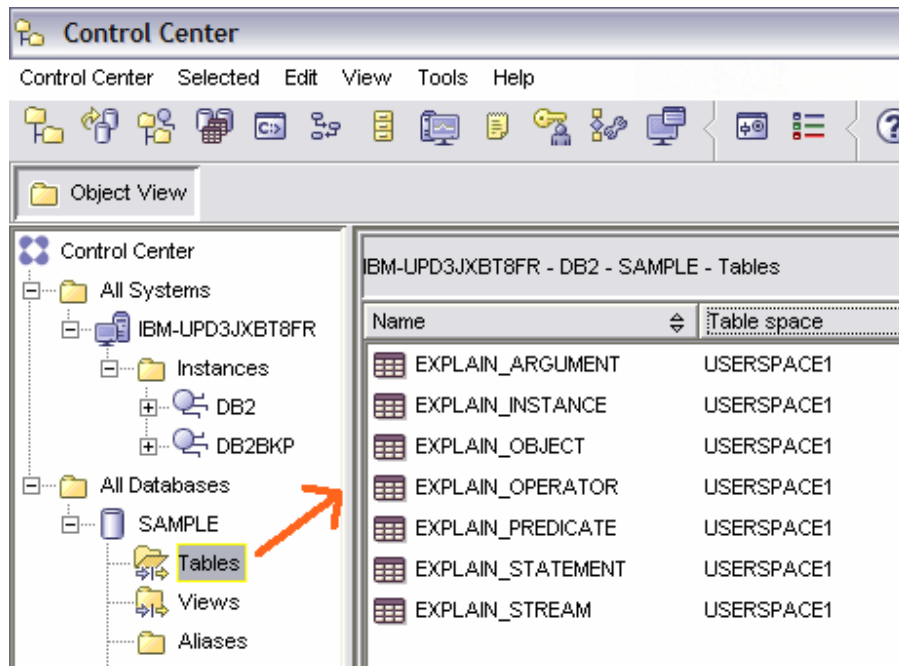
### A. CREATE EXPLAIN TABLES

All DB2 explain facility tools use the explain tables, so let's create them:

1. Make sure you position yourself in the C:/POT/04 EXPLAIN/ directory before continuing
2. To create them, review and run these scripts:

**Explain04001.CMD** which executes ⇒ **Explain04002.DDL**

3. Explore output file **Explain04001\_OUTPUT.TXT**
  - Use Control Center to verify tables were created in your schema, names like **EXPLAIN%**



## B. UNION ALL VIEW EXAMPLE SETUP

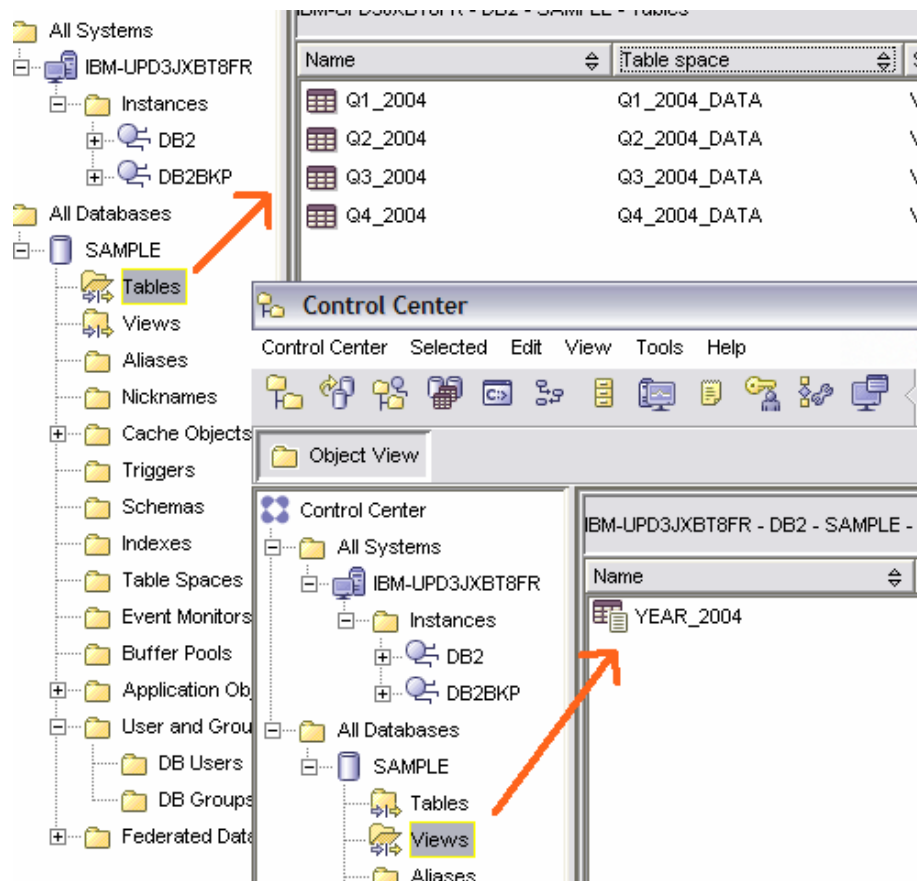
To demonstrate explain in DB2, we will use an example of DB2 UNION ALL VIEW:

1. To set up this example, review and run these scripts:

**Explain04003.CMD** which executes ⇒ **Explain04004.DDL** and **Explain04005.DDL**


2. Explore output file **Explain04003\_OUTPUT.TXT**

- The end of the output for these scripts should show a “check results” section with a Monthly and Quarterly Tax summary
- Use the Control Center to explore the 4 individual tables and 1 view along with the data
  - A filter with table names like Q% would help
  - A filter with view names like YEAR% would help
- Double click on each object to view contents
- Note: the insert to the tables was done through the view!



### C. VISUAL EXPLAIN THROUGH THE COMMAND EDITOR

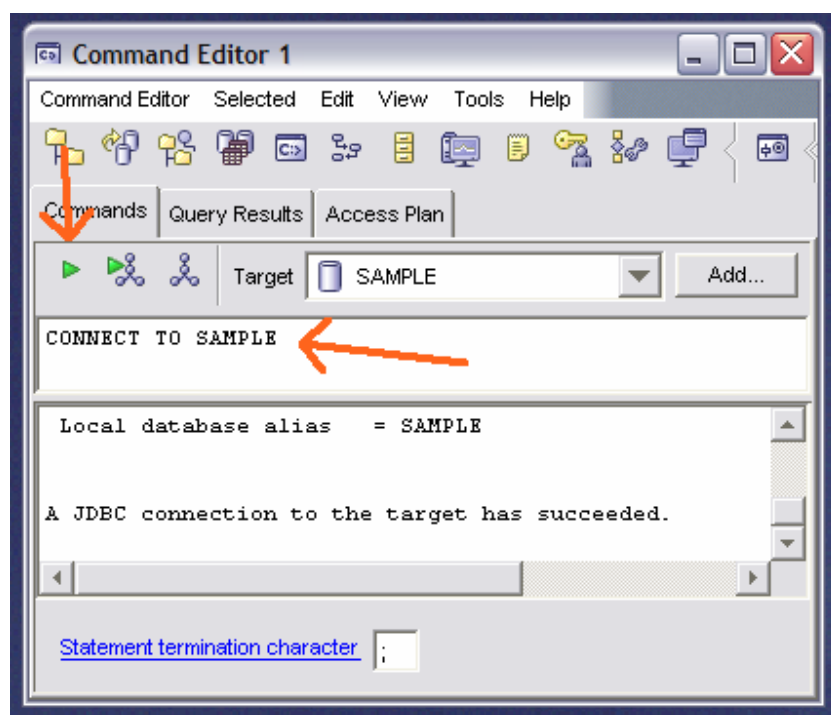
1. From the Control Center, Launch the Command Editor

- Tools ⇒ Command Editor (or use the icon on the toolbar) 

2. In the command pane (top half) type: **connect to sample;**

3. Execute this by clicking the green run arrow: 

- You are now connected to the database and ready give SQL commands



4. From the tool bar choose: Selected ⇒ Open

5. Navigate to the C:\POT\04 EXPLAIN directory and choose this file:  
**Explain004006.SQL**

6. [OK] brings this script into the command window

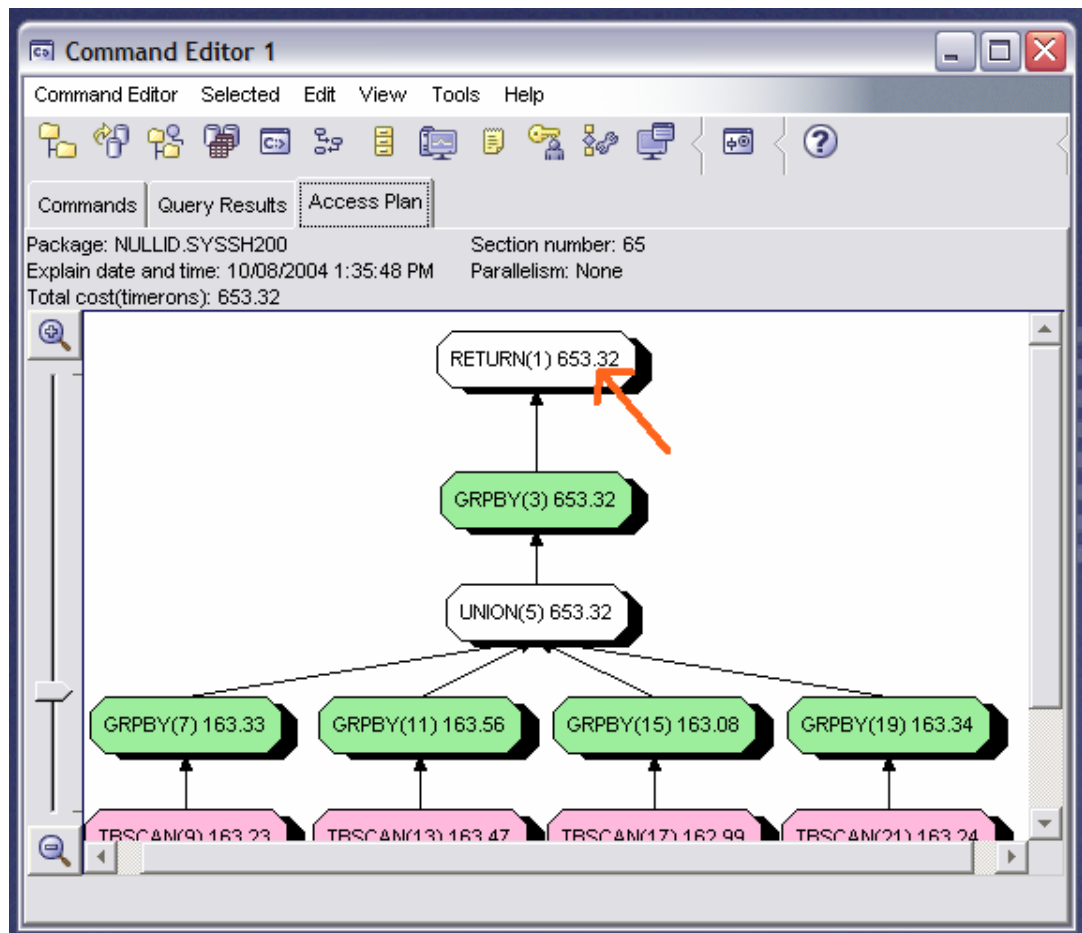
- You could also use a cut/paste method if you like

7. Execute this new SQL, this time clicking the run/access plan button: 

- Note down the query results count here \_\_\_\_\_

8. Click on the Access Plan tab and review what this looks like.

- Note down the total timerons result here \_\_\_\_\_



9. Double click on any object in the access path visual flowchart to drill down into it

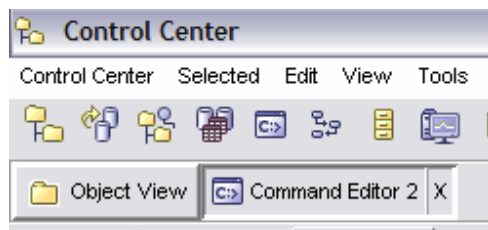
- Note you can save any output details with [Save As] button (try it if you like)
- Explore drill-down for a minute reviewing the various nodes and their detail

10. Close the Command Editor

## D. VISUAL EXPLAIN – FURTHER EXPLORATION

1. From the Control Center right click on database SAMPLE ⇒ Query
  - Command Editor launches, and since you are specifying a database location, it connects for you too
2. Find the same script we used last time (**Explain004006.SQL**) and bring it into the command window again using any method you like ⇒ Execute it with an access plan
3. On the access plan window use the slide bar on the left to adjust to a different size
4. From the tool bar choose: Selected ⇒ Statement. Explore all the options here:
  - Try not to actually print or run anything for now, just explore the possibility of what these do:
 

▪ Print Graph	▪ Show SQL Text	▪ Show Optimization Parameters
▪ Explain SQL	▪ Show Optimized SQL Text	▪ Show Statistics
5. From the tool bar choose: Selected ⇒ View ⇒ Settings
  - Note you can change many characteristics of how your Visual Explain output looks here
6. In the Control Center area just below the task bar, notice you have two buttons
  - The Command Editor (which you are currently at)
  - The Object View



7. Click on **Object View**
  - Takes you back to the Control Center without closing the Command Editor
8. Right click on database SAMPLE ⇒ Explain SQL...
  - This is how you run Visual Explain by itself
  - Cut and paste or use [Get] button to feed it an SQL command
  - Explore this a little bit and close this when you are done exploring

## E. VISUAL EXPLAIN – PACKAGES

1. From the Control Center right click on database SAMPLE ⇒ Application Objects ⇒ Packages
2. Create a filter to just see packages created by your user id
3. Right click on any package ⇒ Show Explainable Statements
  - You can only do this if the last column in this view says “Explain Snapshot” = Yes
4. Right click on any statement number and double click on it to see the explain plan
  - This is made possible because earlier, before the stored procedures were built, we set this environment variable:

**DB2\_SQLROUTINE\_PREOPTS="EXPLAIN YES EXPLSNAP YES"**

The screenshot shows the IBM DB2 Control Center interface. On the left, the 'Application Objects' tree is expanded to 'Packages'. A list of packages is shown, including P3334432, P3334348, P3274586, and DB2SAMPL. The 'Explainable Statements - P3274586' window is open, displaying a table of statements. The table has columns: Statement number, Section number, Explain snapshot, Total cost, and SQL text. The following table represents the data shown in the window:

Statement number	Section number	Explain snapshot	Total cost	SQL text
19	1	Yes	3.82E-4	VALUES(INT(RAND()*4)) INT(
23	2	Yes	3.82E-4	VALUES(INT(RAND()*100)) IF
24	3			
49	4			
50	5			
52	6			
53	7			
55	8			
56	9			
58	10			

The 'Access Plan Graph - SAMPLE' window is also open, showing a query plan. The plan includes the following nodes:

```

graph TD
    TBSCAN[TBSCAN(3) 0] --> RETURN[RETURN(1) 0]
  
```

## F. VISUAL EXPLAIN BASICS ON OPTIMIZING A QUERY

1. Return to your open Command Editor screen ⇒ Commands tab
2. Bring in **Explain04007.DDL** and execute it
  - You just built 4 indexes for your quarterly tables
3. Highlight the entire create indexes DDL and delete it
  - You can use Edit ⇒ Select All
4. Choose: Selected ⇒ History
5. Find the previous query (before the create indexes DDL) bring it in and run it again with access plan
6. Check out the visual explain again and note the timerons here \_\_\_\_\_
  - Index scans resulted in less timerons than table scans on this query right?
  - We made looking in every table of the union all joined view more efficient, but... how can we get DB2 to only consider just the one table in the view? (Answer is in next steps)
7. Bring in **Explain004008.SQL** and execute it with access plan. Check results
  - Note query results count here \_\_\_\_\_ (it should still be the same as before)
  - Note total timerons count here \_\_\_\_\_
  - Does this query give the same results as the one before and if so, why is it better?
8. Choose: Selected ⇒ Statement ⇒ Explain SQL
  - Try a high optimization class and a lower optimization class to see if this affects the query timeron count
9. Choose: Selected ⇒ Statement ⇒ Show Optimization Parameters
  - Check out Bind Options in particular
  - What was the best performing scenario you came up with?

### Section Answers:

- F7. Results are the same, but the query is entirely different as it uses the table constraints on the underlying tables of union all view tables in the query. The optimizer knows this data is only in one table in the view.
- F9. Query optimization class 5 works the same as 3 or 7 for this particular scenario. However, there are many situations where changing an optimizer class **can** affect your timeron count positively or negatively. See query optimization in the documentation for details on how these classes work.

## G. DB2EXFMT

1. To demonstrate how db2exfmt works, review and run these scripts:  
**Explain04009.CMD** which executes ⇒ **Explain04010.DML**
  - Hit [return] to accept all defaults when prompted by the utility
2. Explore output file **Explain04009\_OUTPUT\_PLAN.TXT**
3. Check the explain tables; anything there?
4. Review **Explain DB2EXFMT\_Command.txt** for full command syntax

### Section Answers:

- G3. Yes. Script Explain04010.DML used the EXPLAIN ALL FOR... syntax which is what puts data into the EXPLAIN tables. db2exfmt formats this data and displays it for you.

## H. EXTRA EXERCISES - DB2EXPLN

### Static SQL method

Earlier in script **Explain04005.DDL**, we created a stored procedure called: **UNION\_DATA**. Let's find the DB2 package that contains the static SQL in that stored procedure so we can explain that package:

1. Review and run this script any way you like: **Explain04011.SQL**
  - Find the package name for the stored procedure and note it here \_\_\_\_\_
2. To demonstrate how this utility works in its static mode, review and run this script:  
**Explain04012.CMD [package\_name]**
  - Don't forget to use the parameter when executing this script!
3. Explore output file **Explain04012\_OUTPUT\_PLAN.TXT**
  - Notice packages are broken into sections

### Dynamic SQL method

4. To demonstrate how this utility works in its dynamic mode, review and run these scripts:  
**Explain04013.CMD** which executes ⇒ **Explain04014.SQL**
5. Explore output file **Explain04013\_OUTPUT\_PLAN.TXT**
  - Check the explain tables
6. Review **Explain DB2EXPLN\_Command.txt** for full command syntax

## I. OTHER EXTRA EXERCISES

### 1. Visual Explain **Explain04011.SQL**

- Why are there three tables when the SELECT had two in the FROM clause?

### 2. Rebind package for UNION\_DATA stored procedure using: **Explain04015.DB2**

- Why would you think you would ever need to do this?

### 3. Single out the first record in table Q1\_2004 and note **TX\_NUMBER** here \_\_\_\_\_

- Through the union all view YEAR\_2004 update that record with the date '2004-12-31'
- What happens to the record?

#### Section Answers:

- I1. Remember that SYSCAT.\* are views. If you right click on them and then Show Related, you'll see many of them are made up of more than one system table. (Try SYSCAT.ROUTINES for example)
- I2. For stored procedures, the package is bound when it is created. Statistics and the data model can change though affecting the access path. A rebind can make sure this path is up-to-date.
- I3. Sample SQL is below - fill in TX\_NUMBER and run it.

What happens is that the data moves tables, from Q1\_2004 to Q4\_2004! Note that this is NOT the DB2 default behavior. When the view was created, we used a WITH ROW MOVEMENT predicate to allow this to occur. DELETES, INSERTS and SELECTS also work as expected through the union all view.

```
UPDATE YEAR_2004
SET     TX_DATE = '2004-12-31'
WHERE  TX_NUMBER = ?;
```