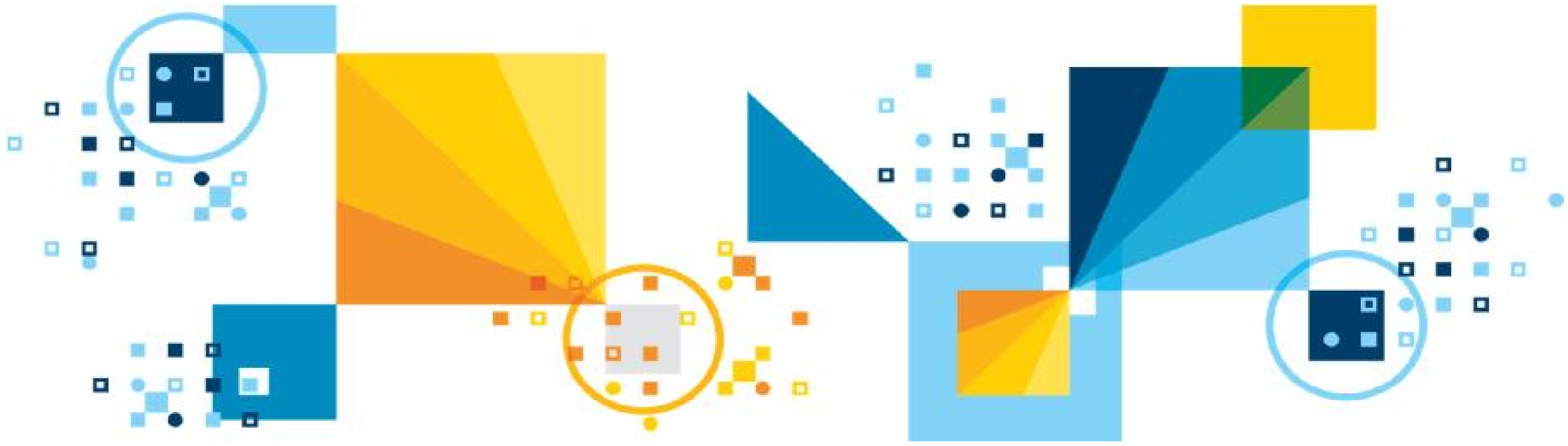
DB2 SQL Query Tuning with BLU Acceleration

Module ID	10303
Length	1 hours + 2 hour hands on lab



For questions about this presentation contact askdata@ca.ibm.com

February 9, 2015

PRESENTATION NOTES FOR PAGE 1

Disclaimer

PRESENTATION NOTES FOR PAGE 2

Module Information

- You should have completed or acquired the necessary knowledge for the following modules in order to complete this module:
  - Module DB2 Performance Monitoring Essentials
  - Module DB2 Performance Monitoring Essentials II
  - Module DB2 BLU Acceleration Fundamentals
  - Module Monitoring Essentials in DB2 BLU Acceleration
  - Module Deep Dive into DB2 BLU Acceleration Main Ideas

- After completing this module, you should be able to know
  - How to capture, analyze, and improve SQL through both better coding techniques and defining complementary objects like indexes and MQT’s for your database Optim Query tuner

PRESENTATION NOTES FOR PAGE 3

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 4

Explore How to capture, analyze, and improve SQL through both better coding techniques and defining complementary objects like indexes and MQT’s for your database Optim Query tuner

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 5

Problematic SQL

- Query optimization is the main common factor that affects application performance
- Problematic SQL statements slow down application
- Problematic SQL statements can be **detected** with:
  - Snapshot monitors
  - Event monitors
  - SQL monitoring interfaces
    - Administrative views
    - Table functions
- Problematic SQL statements can be **analyzed** with:
  - Visual Explain
  - Text Explain Facility

PRESENTATION NOTES FOR PAGE 6

One area of tuning your database server performance is application tuning. Here we focus on rewriting SQL statements and ensuring that the correct database server objects are created for executing the application optimally. This would involve that the correct indexes are available. Some times indexes by themselves do not help. You may need to implement summary tables aka materialized query tables, or reorganize your data using multi-dimensional clustering or even partition your data across multiple servers.

But how do you decide which of these is going to help improve your application performance. For this we need to dig deeper into the application and identify the slow performing SQL statements.

DB2 offers may ways to capture the problematic SQL statements. It offers statement snapshots, administrative views and table functions to capture SQL statements based on a number of criteria. The criteria could be CPU intense, I/O intense, excessive sorting, longest running, etc.

Once you've identified the problematic SQL statements you can use check it's query access plan using the Visual Explain utility or check how long it takes to run using the db2batch utility. Then you can rewrite it by following the recommended best practices or determine what DB2 object(s) can help improve the SQL statement performance using the DB2 Design advisor. The DB2 Design Advisor will analyze one or more SQL statements as a single workload. It'll consider how frequently it's going to run, the data it's going to access, the resources available on the server, etc. Based on these factors it'll provide a list of index, MQT or MDC recommendations for this workload. You can then implement these recommendations.

For dynamic SQL statements we can look at Statement Concentrator as well.

Now we'll look into each of these steps in detail. We'll start with capturing problematic SQL statements

Problematic SQL

§ Problematic SQL statements can be **improved** with:

- Design Advisor (database objects)
  - Indexes
  - Materialized Query Tables
  - Multi-dimensional Clustering
  - Database Partitioning
- Better coding
- Statement Concentrator
- DB2 Optim Query Workload Tuner
- **DB2 10/10.5**
  - Explore new performance enhancing features and take advantage of them

A number of performance improvements have been included in DB2 10
to improve the speed of many queries

NEW IN DB2 10

These improvements are automatic; there are no configuration settings or changes to the SQL statements required

PRESENTATION NOTES FOR PAGE 7

One area of tuning your database server performance is application tuning. Here we focus on rewriting SQL statements and ensuring that the correct database server objects are created for executing the application optimally. This would involve that the correct indexes are available. Some times indexes by themselves do not help. You may need to implement summary tables aka materialized query tables, or reorganize your data using multi-dimensional clustering or even partition your data across multiple servers.

But how do you decide which of these is going to help improve your application performance. For this we need to dig deeper into the application and identify the slow performing SQL statements.

DB2 offers may ways to capture the problematic SQL statements. It offers statement snapshots, administrative views and table functions to capture SQL statements based on a number of criteria. The criteria could be CPU intense, I/O intense, excessive sorting, longest running, etc.

Once you've identified the problematic SQL statements you can use check it's query access plan using the Visual Explain utility or check how long it takes to run using the db2batch utility. Then you can rewrite it by following the recommended best practices or determine what DB2 object(s) can help improve the SQL statement performance using the DB2 Design advisor. The DB2 Design Advisor will analyze one or more SQL statements as a single workload. It'll consider how frequently it's going to run, the data it's going to access, the resources available on the server, etc. Based on these factors it'll provide a list of index, MQT or MDC recommendations for this workload. You can then implement these recommendations.

For dynamic SQL statements we can look at Statement Concentrator as well.

We'll also look at best practices in writing SQL statements and what to do if no amount of tuning helps.

Now we'll look into each of these steps in detail. We'll start with capturing problematic SQL statements

A common scenario

- **New major application version**
- Expectations:
  - Increase value
  - Reduce response time
  - Reduce CPU utilization
  - Increase user capacity
- Use **`db2advis`** with **`-wlm evmonname`** or **`-w workloadname`** option to capture and advise about:
  - Index
  - MDC
  - MQT
  - DB Partitioning
- Capture additional high cost SQL for further analysis and tuning
- Further analyze and tune with Visual Explain
- STMT_CONC (DB CFG parameter)

PRESENTATION NOTES FOR PAGE 8

We will present a fairly common scenario where a new application version has been developed and is being performance tested. Expectations are such that this version will offer more value not only adding new functionality, but also by reducing performance overhead and CPU requirements. We will EXPLORE some of the tools and techniques to tune SQL on a DB2 platform.

Actual Situation

- Benchmark for new application version slower than previous application version
- DB2 is configured well
  - Large part of DB2 tuning done in previous application version
  - Autoconfigure has been run on new schema
- New SQL may need tuning
  - Complementary database objects (indexes, MDC, MQT, partitioning)
  - Coding (apply best practices)

NEW IN DB2 10

Use the AUTOCONFIGURE command to get recommendations from the configuration advisor. Although the wizard interface for the configuration advisor is discontinued in DB2 10, the configuration advisor is still available by using the AUTOCONFIGURE command

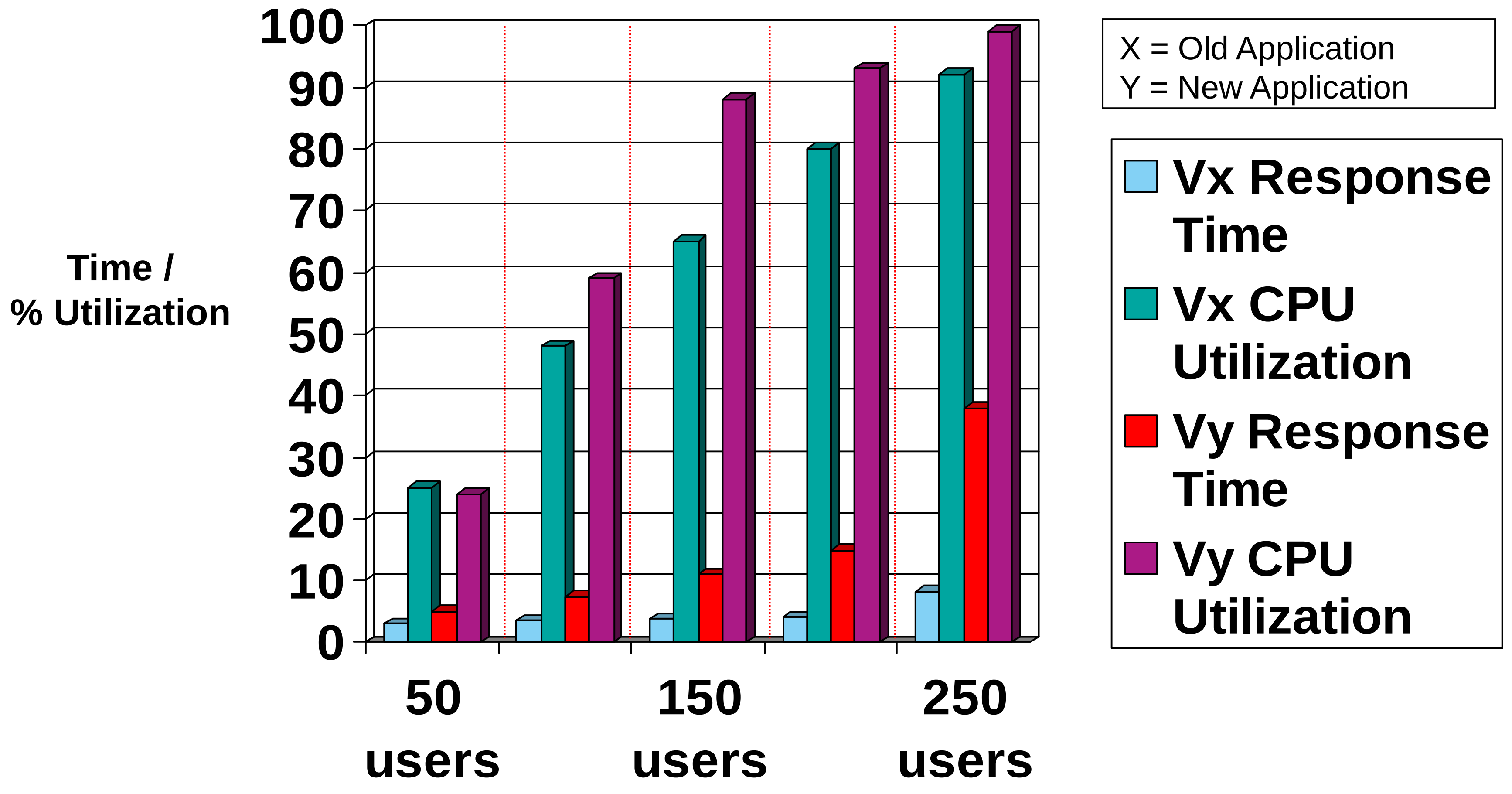
PRESENTATION NOTES FOR PAGE 9

To set up the situation:

There is a new App version with a large one with new components, including third party services, it is felt that there may be some additional tuning required, it is necessary to examine SQL from a system tuning standpoint and capture workload activity to provide insight as to whether new complimentary database objects, indexes, mdc’s…, should be included.

Original Comparative Application Results Version X vs. Version Y

- New version is slower and consumes more resources



PRESENTATION NOTES FOR PAGE 10

The main point is that the benchmark for the new version of the software yielded higher CPU utilization and response time on a per user basis. Side notes might include talking about why response time, cpu utilization, and the number of concurrent users are important metrics when performing benchmarks.

Vy represents the new version of the software.

One interesting aspect is that the general curve of performance here is the same, almost as if new version is stacked on blocks.

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 11

Use Event Monitor for Activities & db2advis To Initially Tune Workload

- Event monitor for activities captures both dynamic and static SQL
- WLM feature can focus on specific workloads and service classes
- Optionally Snapshot Monitoring or Monitoring Table Functions can be used in place of Event Monitor for Activities
- Integrates with design advisor (db2advis) for recommendations about:
  - Indexes
  - MQTs
  - MDCs
  - Database partitioning

NEW IN DB2 10

Use the db2advis command to get recommendations from the design
advisor. The wizard interface for the design advisor is discontinued in DB2 10, but the design advisor is still available using the db2advis command

PRESENTATION NOTES FOR PAGE 12

Event Monitor for Activities in combination with the db2advis command to generate recommendations with respect to whether or not new indexes, mqt’s, etc will help to improve performance.

This method will evaluate both dynamic and static SQL and when the performance optimization feature pack is licensed allows evaluation to occur on a more granular basis (workload/service class).

Steps to take for EVM for Activities & db2advis Analysis

1. Alter workload **`sysdefaultuserworkload`** (or your desired workload) to collect activity data on coordinator with details and values
2. Create event monitor *db2activities* for **`activities`**
3.

```
ALTER WORKLOAD SYSDEFAULTUSERWORKLOAD COLLECT ACTIVITY DATA
ON ALL WITH DETAILS AND VALUES;
CREATE EVENT MONITOR DB2ACTIVITIES
  FOR ACTIVITIES WRITE TO TABLE;
SET EVENT MONITOR DB2ACTIVITIES STATE 1;
```

```
db2advis –d sample –wlm db2activities –m MICP –o advise.out
```

4. workload (or your desired workload)
5. Use:

New event monitors features:

- All event monitors now support the WRITE TO TABLE t
- Existing event monitors that write to tables can be altered to capture additional logical data groups

NEW IN DB2 10

PRESENTATION NOTES FOR PAGE 13

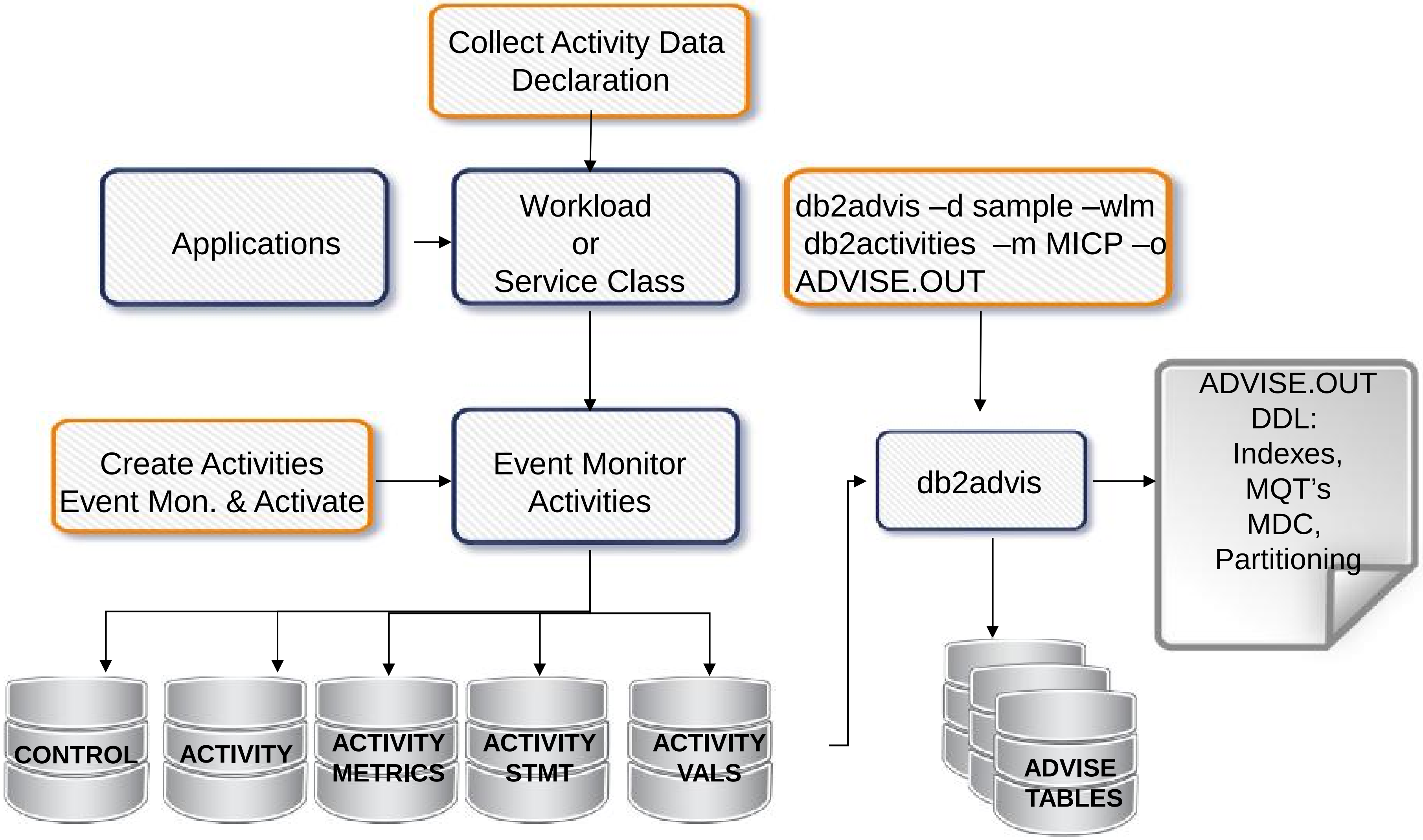
Here we take a look at how to use the event monitor for ACTIVITIES along with db2advis

These are the steps:

Create the activity event monitior

* Activate it, even if you create an event monitor to autostart you will need to activate it immediately after creating it.
* Alter the default user workload to collect activity data.
* Work runs
* Use db2 advise to import workload from activity event monitor for evaluation.

db2advis and EVM for Activities flow


Collect Activity Data Declaration
Applications
Workload or Service Class
db2advis –d sample –wlm db2activities –m MICP –o ADVISE.OUT
Create Activities Event Mon. & Activate
Event Monitor Activities
db2advis
ADVISE.OUT DDL: Indexes, MQT's MDC, Partitioning
CONTROL
ACTIVITY
ACTIVITY METRICS
ACTIVITY STMT
ACTIVITY VALS
ADVISE TABLES

PRESENTATION NOTES FOR PAGE 14

Here is a flow chart to show what we put into effect,

The new activities event monitor is created and activated it will write activities out to a set of tables that will be created when the create event monitor for activities command is issued,

A control table – contains the xmlid which can be used when joining all tables

The activity table where you will find higher level information about the applications

The activity-statement table as the name would imply includes the SQL statements involved

The activity-values table where the transaction values are stored.

2. In order for metrics from the default workload to be collected we use the alter workload command to make that declaration upon the workload.

Here we both qualify the workload and quantify what level of data to collect.

3. The db2 advise command is issued after a genuine workload has been run. The –wlm parameter specifies that the db2activities event monitor and its corresponding tables are to be used as the source of the workload

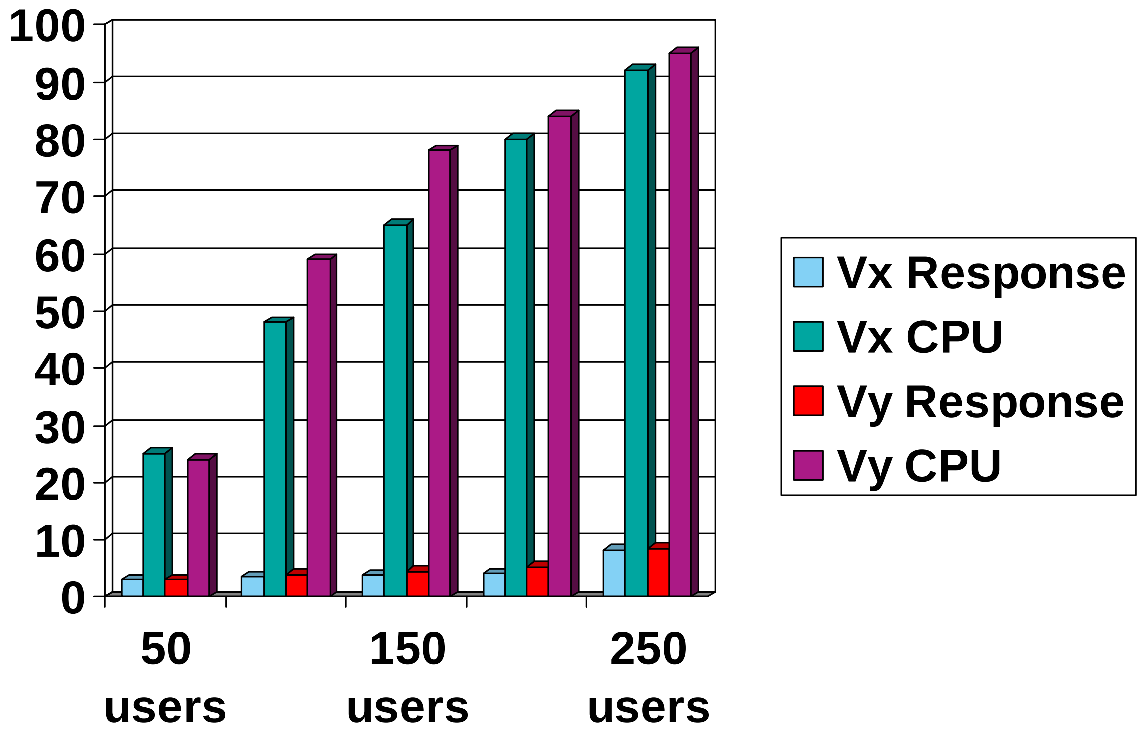
and it joins them together on ACTIVATE_TIMESTAMP, ACTIVITY_ID and ACTIVITY_SECONDARY_ID for records that have PARTIAL_RECORD = 0 (completed transactions) .

You can further qualify the workload with start and stop time stamps specification, you can also, optionally, specify specific workloads and serviceclasses to be included.

The db2 advise command saves the analysis in the advise tables and produces a formatted output of recommendations to improve performance with the inclusion (or removal) of Indexes , MDC’s and MQT’s.

After db2advis Recommendations and Implementation Comparative Application Results Version X vs. Version Y

- Now it’s better, but the new application should be much faster
- SQL costs… need further analysis, still very high



PRESENTATION NOTES FOR PAGE 15

Things are looking better after creating indexes and MQT's recommended by db2advis but there is still room for improvement. Now is necessary to dig deeper by finding specifically costly SQL.

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 16

Detect SQL costs

Snapshot Monitoring?

Event Monitoring?

SQL Interfaces?

PRESENTATION NOTES FOR PAGE 17

At this point they decide to further collect high cost SQL to see if it can be tuned further. In the next few slides we will consider some of the options he has to complete his task of finding the costly SQL.

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 18

Snapshot Monitoring

- To request snapshot information about the dynamic SQL running on SAMPLE database, issue:

```
GET SNAPSHOT FOR DYNAMIC SQL ON sample
```

- Returns a point-in-time picture of the contents of the SQL statement cache for the database
- Only for DYNAMIC SQL
- Formatted text output

⚠ **Switches must be ON**

PRESENTATION NOTES FOR PAGE 19

One way of capturing dynamic SQL statements is to get snapshot of the application while it’s executing. This is run from the command line and provides a formatted text output that needs further analysis to identify the problematic SQL statement.

To capture the information, first the Monitor switches need to be turned on.

When you use the “get snapshot for dynamic sql” command you get formatted text based output as seen here.

Note: for each of the metrics the appropriate snapshot switch must be turned on.

You can see some very important information, like the number of executions, knowing the relative number of times a statement executes is a key factor in determining whether or not it will pay to tune it.

You know you can look at preparation costs too, ideally each statement it is only compiled once else preparation time can become a concern. When parameter markers are not being used compilation time can be come a serious consumer of CPU cycles.

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 20

Event Monitors

- Event monitors are used to collect information about the database and any connected applications when specified events occur
- Filter events on APPL_ID, AUTH_ID and APPL_NAME
- Event type to capture SQL statements:
  - STATEMENTS
    - Statement start/stop time
    - CPU used
    - Dynamic and Static SQL
- **High overhead**

PRESENTATION NOTES FOR PAGE 21

Snapshot captures only the statements running at that point of time. But if you want to capture all statements, you can setup an Event Monitor.

We'll see how to create one on a later slide (create event monitor statement example).

To determine if an application will generate events for a particular event monitor, the WHERE clause is evaluated:

For each active connection when an event monitor is first turned on

Subsequently for each new connection to the database at connect time

The WHERE clause is not evaluated for each event.

If no WHERE clause is specified, all events of the specified event type will be monitored.

The event-condition must not exceed 32 678 bytes in length in the database code page (SQLSTATE 22001).

APPL_ID

Specifies that the application ID of each connection should be compared with the comparison-string in order to determine if the connection should generate CONNECTION, STATEMENT or TRANSACTION events (whichever was specified).

AUTH_ID

Specifies that the authorization ID of each connection should be compared with the comparison-string in order to determine if the connection should generate CONNECTION, STATEMENT or TRANSACTION events (whichever was specified).

APPL_NAME

Specifies that the application program name of each connection should be compared with the comparison-string in order to determine if the connection should generate CONNECTION, STATEMENT or TRANSACTION events (whichever was specified).

It has a high overhead; hence specify a large buffer size and use a different tablespace than your application tablespaces.

Creating an Event Monitor for Statements

- A table event monitor streams event records to SQL tables, this makes capture, parsing, and management of event monitoring data easy
- Need SYSADM or DBADM to create a table event monitor
- Syntax:

🟥 = *REDUCE IMPACT*

```
CREATE EVENT MONITOR stmtmon
FOR STATEMENTS
WHERE APPL_NAME = 'NEWAPP' AND
        AUTH_ID = 'BBDS'
WRITE TO TABLE IN event_tblspace
   CONNHEADER(TABLE STMT_EVT_CH IN TBS_EVMON),
   STMT(TABLE STMT_EVT_STMT IN TBS_EVMON TRUNC),
   CONTROL(TABLE STMT_EVT_CTRL IN TBS_EVMON)
BUFFERSIZE 2000
NONBLOCKED
```

OR

```
WRITE TO FILE '/tmp/dlevents'
```

OR

```
WRITE TO PIPE '/home/riihi/dlevents'
```

PRESENTATION NOTES FOR PAGE 22

Here we are creating and Event Monitor for Statements, You'd have to have at least DBADM authority to do so. Notice parameters marked in red , this is where we can reduce the overhead produced by such and event monitor. We filter on only specific applications, direct tables into their own tablespaces, specify the trunc modifier on the statement table so that the statements are saved as varchar(n) where n is dependent on the pagesize of the tablespace. If you don't specify trunc then the STMT_TEXT will be written out to a CLOB. It is a good idea to use buffers and specify nonblocked this way an event agent will not ever wait for a buffer to be written to disk, so there is a small chance of data loss but that should be rare and inconsequential when looking at an entire workload.

NONBLOCKED should be used after careful consideration. In a heavily loaded system, it is common to lose data with NONBLOCKED, and this may impact the ability to diagnose a problem, where problem diagnosis is more important than performance. The solution may be to use BLOCKED, and run the trace for a short period of time.

NONBLOCKED

Specifies that each agent that generates an event should not wait for the event buffer to be written out to disk if the agent determines that both event buffers are full. NONBLOCKED event monitors do not slow down database operations to the extent of BLOCKED event monitors. However, NONBLOCKED event monitors are subject to data loss on highly active systems.

What we saw was a Table Event Monitor. Here all the information will be written to the table. This information can also be captured in file or written to pipe.

Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 23

SQL Monitoring Interfaces

- **Administrative Views**
  - Easy-to-use application programming interface
  - Execute administrative functions through SQL
- **Table functions MON_GET_...**
  - Enhanced reporting and monitoring of the database system, data objects, and the package cache
  - **Have a lower impact on the system than existing system monitor and snapshot interfaces**

PRESENTATION NOTES FOR PAGE 24

DB2 provides administrative views that provide an easy-to-use application programming interface to DB2 administrative functions through SQL.

The administrative views fall into three categories:

Views based on catalog views.

Views based on table functions with no input parameters.

Views based on table functions with one or more input parameters.

Authorized users can capture snapshots of monitor information for a DB2 instance by using snapshot administrative views or snapshot table functions. The snapshot administrative views provide a simple means of accessing data for all database partitions of the connected database. The snapshot table functions allow you to request data for a specific database partition, globally aggregated data, or data from all database partitions. Some snapshot table functions allow you to request data from all active databases.

You must have SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to capture a database snapshot. To obtain a snapshot of a remote instance, you must first connect to a local database belonging to that instance.

For all administrative views in the SYSIBMADM schema, you need SELECT privilege on the view. Some administrative views require additional authorities beyond SELECT on the view and EXECUTE on the underlying administrative table function.

There are a number of different snapshot monitor SQL administrative views available, each returning monitor data about a specific area of the database system. For example, the SYSIBMADM.SNAPBP SQL administrative view captures a snapshot of buffer pool information.

Before capturing a snapshot, consider if you need information from monitor elements that are under monitor switch control. If a particular monitor switch is off, the monitor elements under its control will not be collected.

Examples of SQL interfaces – Finding SQL Costs

- **Snapshot Administrative Views**
  - LONG_RUNNING_SQL (Time, Statement, Status)
  - QUERY_PREP_COST (High Prep Times, % of Exec)
  - TOP_DYNAMIC_SQL (Exec Time, Sorts)
  - …
- **Table Functions**
  - MON_GET_ACTIVITY_DETAILS (Executing vs. Waiting)

DB2 10 returns additional columns such as columns that report information about data tags in service class thresholds

  - MON_GET_PKG_CACHE_STMT

DB2 10 returns additional columns that report metrics about I/O server efficiency, processing time for authentication, statistics generation, statement execution, high water mark input values, and extended latch waits

PRESENTATION NOTES FOR PAGE 25

Here are some of the more useful SQL based views and table functions that can be used to detect and capture performance problems. The administrative views are aptly named APPL_PERFORMANCE can give you insight into how efficient your application is with respects to performing reads, LONG_RUNNING_SQL takes a temporal perspective looking for long running statements. QUERY_PREP_COST looks to identify queries using a lot cycles in the preparation of the query. TOP_DYNAMIC_SQL is a good generic view allowing you to sort on a variety of key metrics like execution time and sort time.

The MON_GET_ACTIVITY_DETAILS allows you to get very granular in your assessments in that it makes a lot of distinctions between the time an activity is actually executing vs waiting. The MON_GET_PKG_CACHE_STMT can be used to capture both dynamic and static costly SQL.

SQL – High CPU Time

§ Example: List top 10 SQL statements by cpu_time

```
SELECT MEMBER,
       SECTION_TYPE,
       VARCHAR(STMT_TEXT,200) AS STATEMENT,
       num_exec_with_metrics as numExec,
       TOTAL_CPU_TIME/NUM_EXEC_WITH_METRICS AS AVG_CPU_TIME,
       TOTAL_CPU_TIME
FROM   TABLE(MON_GET_PKG_CACHE_STMT('D', NULL, NULL, -2)) as T
WHERE  T.NUM_EXEC_WITH_METRICS <> 0
ORDER BY AVG_CPU_TIME desc
fetch first 10 rows only;
```

PRESENTATION NOTES FOR PAGE 26

The MON_GET_PKG_CACHE_STMT table function returns a point-in-time view of both static and dynamic SQL statements in the database package cache.

These are from the package cache and hence currently stored there.

Agenda

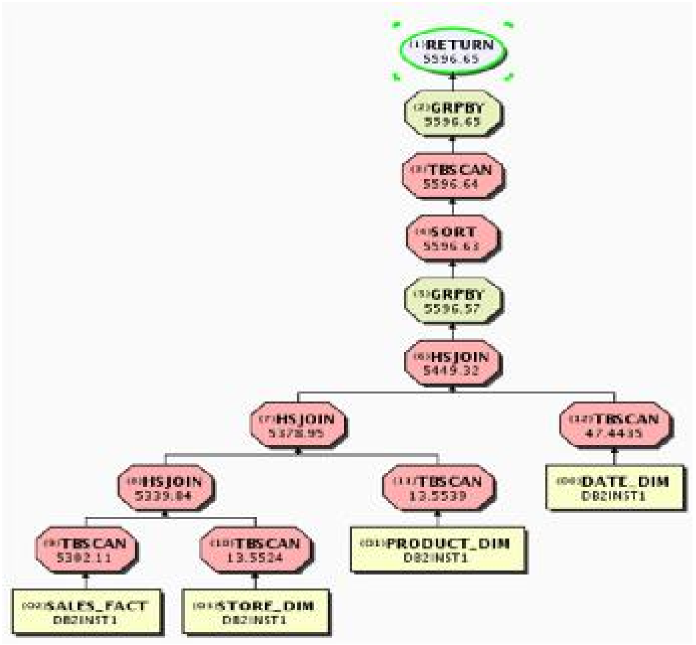
- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

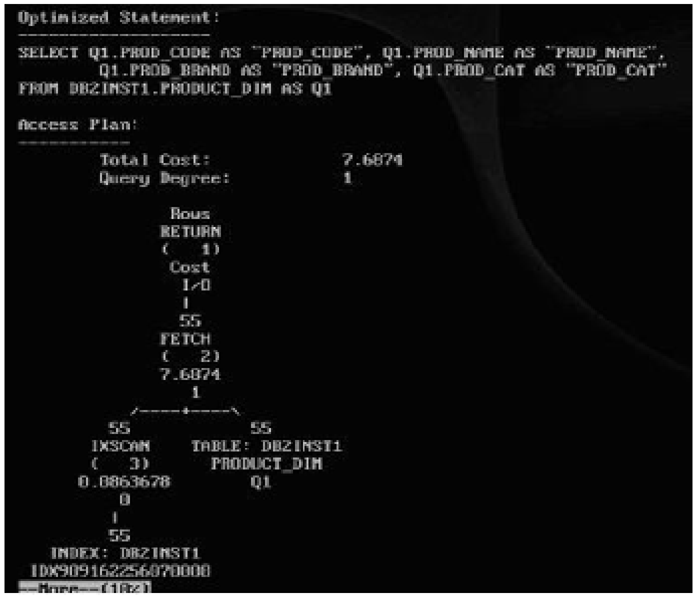
PRESENTATION NOTES FOR PAGE 27

SQL Explain Tools

- **Graphical**
  - Easy to quickly spot the problem
  - Provides drill down functionality
  - Multiple images can be stored for comparison
- **Text Based**
  - Can be used with any interface
  - All the information is contained on a single screen
  - Available on all platforms
  - Format output with db2exfmt or db2expln

In DB2 10 the db2exfmt command output now shows the table space attributes value for each table space containing a partitioned table





```
Optimized Statement:
-------------------
SELECT Q1.PROD_CODE AS "PROD_CODE", Q1.PROD_NAME AS "PROD_NAME",
        Q1.PROD_BRAND AS "PROD_BRAND", Q1.PROD_CAT AS "PROD_CAT"
FROM DB2INST1.PRODUCT_DIM AS Q1

Access Plan:
-----------
        Total Cost:             7.6874
        Query Degree:           1

              Rows
             RETURN
             (   1)
              Cost
               I/O
               |
               55
             FETCH
             (   2)
             7.6874
                1
          /-----+-----\
        55            55
      IXSCAN    TABLE: DB2INST1
      (   3)       PRODUCT_DIM
    0.0863678          Q1
         0
         |
        55
   INDEX: DB2INST1
  IDX909162256070000
--More--(18%)
```

PRESENTATION NOTES FOR PAGE 28

So far we’ve seen tools that provide resource usage information that enables us to capture the problematic SQL. We’ll now find out how the optimizer is running the query.

We do this by examining the query access plans.

There’s both a graphical and a character based interface to DB2’s Explain facility.

Each has it’s pro and cons….. See slide

Why use Explain?

- **To seek performance tuning opportunities**
  - How are tables being accessed?
  - How useful are additional indexes?
  - Does rewriting the query help?

- **Comparisons: To understand changes in query performance due to:**
  - Changes in the data model
  - Changes in the data
  - Changes in configuration parameters

- **View statistics used at time of optimization and current performance**

NEW IN DB2 10

In general Control Center and related wizards and advisors have been discontinued in DB2 10. These have been replaced by a new set of GUI tools: IBM Data Studio and IBM InfoSphere Optim tools

PRESENTATION NOTES FOR PAGE 29

Using the query access plan will help us determine if the query is being executed by DB2 as we think it should be executed. Are tables being access in the same way as we think? Is the correct index being used? Are table statistics up to date? When we make changes to a query, or add an index or change optimization levels we can compare and quantify the differences in the access plans.

The Explain Tools offer clues as to why the optimizer has made particular decisions.

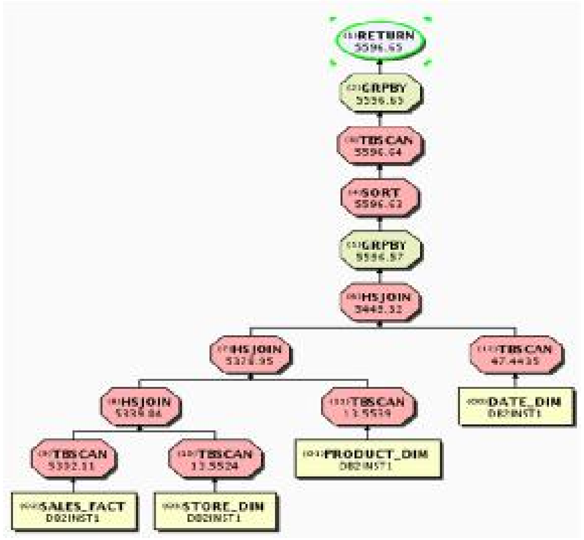
It allows the user to maintain a history of problem query access plans during key transition periods like:

- New index additions
- Large data updates/additions
- RUNSTATS changes
- Release to Release migration
- Significant DB or DBM configuration changes

With this information problem determination is easier, and often faster with a reference plan to compare against

How to use Visual Explain

- **Invoke from**
  - IBM Data Studio
  - IBM Optim Workload Query Tuner
- **Enter SQL to be analyzed**
  - Trap the poorly-running SQL statement from your program or from performance monitors, or create a brand new statement
  - The text can then just be typed or copied into the input box
  - Click the Visual Explain button
- **Output**
  - Explain Information stored in Explain Tables
    - Detailed information
    - Manipulate explain information using SQL
  - Access plan graph
- **For dynamic and static SQL statements**



PRESENTATION NOTES FOR PAGE 30

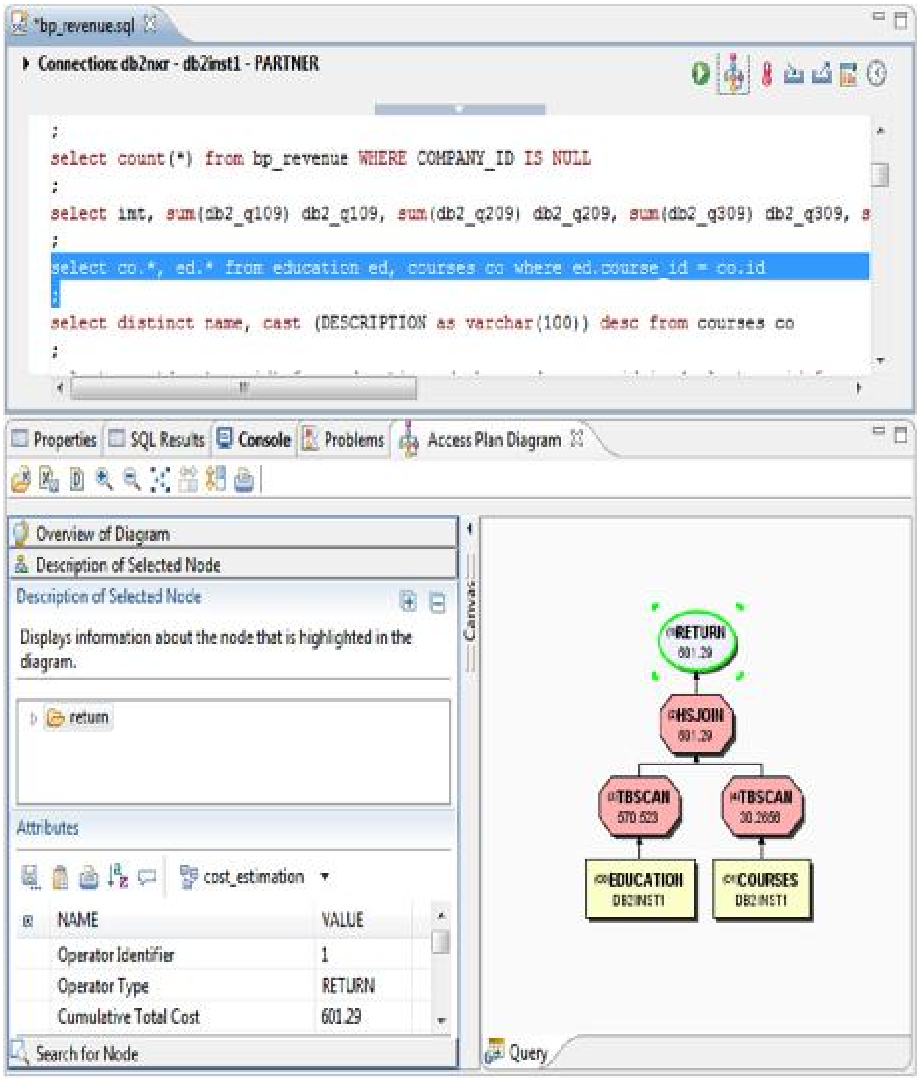
You can get to the visual explain interface from either the control center, or the data studio toolset. All you really need to provide is the SQL statement to analyze. The output is stored in the explain tables where detailed information is stored.

This is how we use the Visual Explain utility

Evaluate Index usage, access type, join methods

Visual Explain Interface

- **Every object in the Visual Explain interface can be drilled down for additional information**
- **Cost**
  - The estimated total resource usage necessary to execute the access plan for a statement. The unit of cost is the timeron
  - **Timeron**
    - A combination of CPU cost (in number of instructions) and I/O (in numbers of seeks and page transfers)
    - In general if you have a larger number of timerons your query will run slower
- **All of the run times of the individual components are cumulative and are measured in timerons**



```
;
select count(*) from bp_revenue WHERE COMPANY_ID IS NULL
;
select int, sum(db2_q109) db2_q109, sum(db2_q209) db2_q209, sum(db2_q309) db2_q309, s
;
select co.*, ed.* from education ed, courses co where ed.course_id = co.id
;
select distinct name, cast (DESCRIPTION as varchar(100)) desc from courses co
;
```

Displays information about the node that is highlighted in the diagram.

NAME	VALUE
Operator Identifier	1
Operator Type	RETURN
Cumulative Total Cost	601.29

PRESENTATION NOTES FOR PAGE 31

The overall cost of a query is represented in timerons. A timeron does not directly equate to any actual elapsed time, but gives a rough relative estimate of the resources (cost) required by the database manager to execute two plans for the same query.

Generally the higher the timeron the more expensive the query.

This is the default view for an Access Plan Diagram generated with Visual Explain from the Optim Database Administrator.

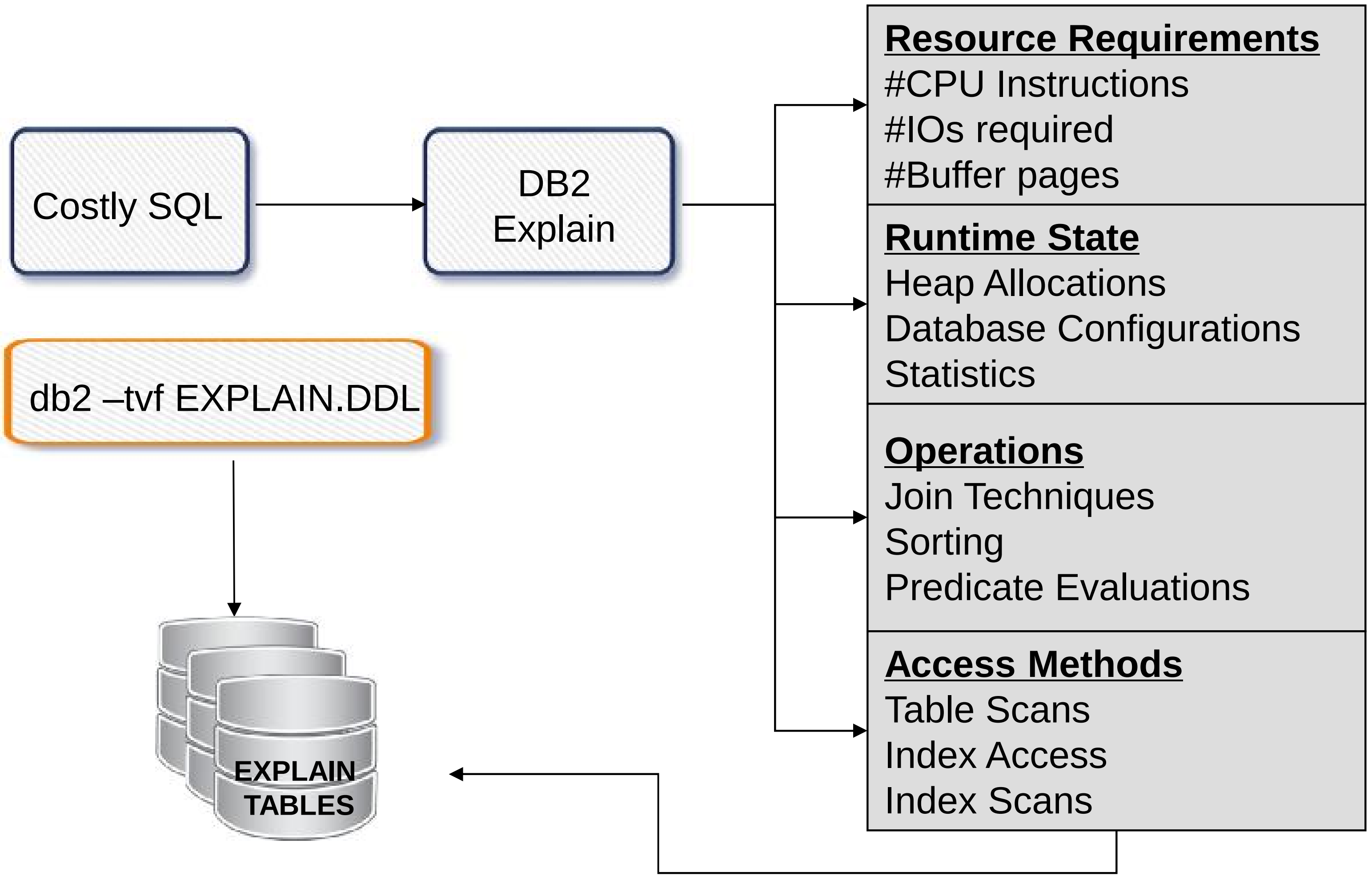
Use the diagram overview to navigate to specific sections of a graph to focus on specific part of the access plan.

Use the control buttons to save the access plan diagram or open a new one from a file.

You can customize your preferences for Visual explain by clicking on the Window and selecting Preferences, there

You can set up color schemes, fonts, shape settings. There are a number of Query explain settings that can be set for the various platforms supported by Optim's incarnation of Visual Explain (DB2 z/OS, DB2 LUW, IDS, and Oracle).

Overview: Costly SQL à Explain



PRESENTATION NOTES FOR PAGE 32

The SQL in question is fed into the DB2 Explain facility. The explain facility produces various sets of information like Resource Requirements, Operations performed, this information is stored in a what is known as “The Explain Tables”. You must create explain tables in your database by executing the EXPLAIN.DDL file which comes as part of your DB2 installation.

Capturing and accessing section actuals

- **Section actuals are runtime statistics collected during the execution of the section for an access plan**
- **The section actuals values can then be compared with the estimated access plan values generated by the optimizer to assess the validity of the access plan**
- **To capture a section with actuals, you need to enable section actuals:**
  - Enable section actuals for the entire database using the section_actuals database configuration parameter

```
db2 update database configuration using section_actuals base
```

  - Enable section actuals for a specific application using the WLM_SET_CONN_ENV procedure

```
CALL WLM_SET_CONN_ENV(…)
```

PRESENTATION NOTES FOR PAGE 33

Capturing and accessing section actuals

Section actuals are runtime statistics collected during the execution of the section for an access plan. To capture a section with actuals, you use the activity event monitor. To access the section actuals, you perform a section explain using the EXPLAIN_FROM_ACTIVITY stored procedure.

To be able to view section actuals, you must perform a section explain on a section for which section actuals were captured (that is, both the section and the section actuals are the inputs to the explain facility). Information about enabling, capturing, and accessing section actuals is provided here.

Enabling section actuals

Section actuals will only be updated at runtime if they have been enabled. Enable section actuals for the entire database using the section_actuals database configuration parameter or for a specific application using the WLM_SET_CONN_ENV procedure.

Section actuals will only be updated at runtime if they have been enabled. Enable section actuals using the section_actuals database configuration parameter. To enable section actuals, set the parameter to BASE (the default value is NONE). For example:db2 update database configuration using section_actuals base

To enable section actuals for a specific application, use the WLM_SET_CONN_ENV procedure and specify BASE for the section_actuals element. For example:CALL WLM_SET_CONN_ENV(NULL, '<collectactdata>WITH DETAILS, SECTION</collectactdata> <collectsectionactuals>BASE</collectsectionactuals> ')

Note:

The setting of the section_actuals database configuration parameter that was in effect at the start of the unit of work is applied to all statements in that unit of work. When the section_actuals database configuration parameter is changed dynamically, the new value will not be seen by an application until the next unit of work.

The section_actuals setting specified by the WLM_SET_CONN_ENV procedure for an application takes effect immediately. Section actuals will be collected for the next statement issued by the application.

Section actuals cannot be enabled if automatic statistics profile generation is enabled (SQLCODE -5153).

Capturing section actuals

The mechanism for capturing a section, with section actuals, is the activity event monitor. An activity event monitor writes out details of an activity when the activity completes execution, if collection of activity information is enabled. Activity information collection is enabled using the COLLECT ACTIVITY DATA clause on a workload, service class, threshold, or work action. To specify collection of a section and actuals (if the latter is enabled), the SECTION option of the COLLECT ACTIVITY DATA clause is used. For example, the following statement indicates that any SQL statement, issued by a connection associated with the WL1 workload, will have information (including section and actuals) collected by any active activity event monitor when the statement completes:ALTER WORKLOAD WL1 COLLECT ACTIVITY DATA WITH DETAILS,SECTIONIn a partitioned database environment, section actuals are captured by an activity event monitor on all partitions where the activity was executed, if the statement being executed has a COLLECT ACTIVITY DATA clause applied to it and the COLLECT ACTIVITY DATA clause specifies both the SECTION keyword and the ON ALL DATABASE PARTITIONS clause. If the ON ALL DATABASE PARTITIONS clause is not specified, then actuals are captured on only the coordinator partition. In addition, besides the COLLECT ACTIVITY DATA clause on a workload, service class, threshold, or work action, activity collection can be enabled (for an individual application) using the WLM_SET_CONN_ENV procedure with a second argument that includes the collectactdata tag with a value of "WITH DETAILS, SECTION".

Limitations

The limitations, with respect to the capture of section actuals, are the following:

Section actuals will not be captured when the WLM_CAPTURE_ACTIVITY_IN_PROGRESS stored procedure is used to send information about a currently executing activity to an activity event monitor. Any activity event monitor record generated by the WLM_CAPTURE_ACTIVITY_IN_PROGRESS stored procedure will have a value of 1 in its partial_record column.

When a reactive threshold has been violated, section actuals will be captured on only the coordinator partition.

Explain tables must be migrated to DB2® Version 9.7 Fix Pack 1, or later, before section actuals can be accessed using a section explain. If the explain tables have not been migrated, the section explain will work, but section actuals information will not be populated in the explain tables. In this case, an entry will be written to the EXPLAIN_DIAGNOSTIC table.

Existing DB2 V9.7 activity event monitor tables (in particular, the activity table) must be recreated before section actuals data can be captured by the activity event monitor. If the activity logical group does not contain the SECTION_ACTUALS column, a section explain may still be performed using a section captured by the activity event monitor, but the explain will not contain any section actuals data.

Accessing section actuals

Section actuals can be accessed using the EXPLAIN_FROM_ACTIVITY procedure. When you perform a section explain on an activity for which section actuals were captured, the EXPLAIN_ACTUALS explain table will be populated with the actuals information. Note: Section actuals are only available when a section explain is performed using the EXPLAIN_FROM_ACTIVITY procedure.

The EXPLAIN_ACTUALS table is the child table of the existing EXPLAIN_OPERATOR explain table. When EXPLAIN_FROM_ACTIVITY is invoked, if the section actuals are available, the EXPLAIN_ACTUALS table will be populated with the actuals data. If the section actuals are collected on multiple database partitions, there is one row per database partition for each operator in the EXPLAIN_ACTUALS table.

URL: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.perf.doc/doc/c0056362.html

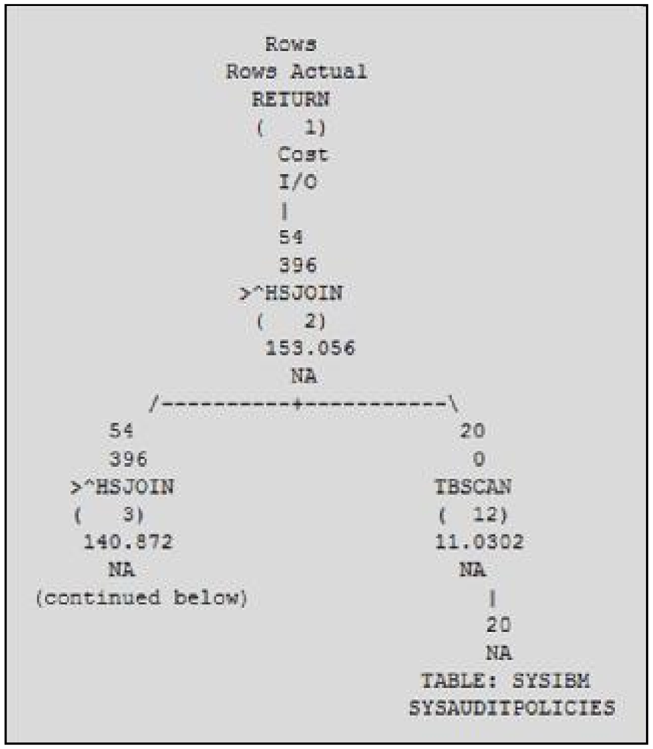
Capturing and accessing section actuals

- **Section actuals can be accessed using the EXPLAIN_FROM_ACTIVITY procedure**

```
CALL EXPLAIN_FROM_ACTIVITY (…)
```

- **You can format the explain data using the db2exfmt command and specifying, as input, the explain instance key that was returned as output from the EXPLAIN_FROM_ACTIVITY procedure**

There is graphical interface where you can collect and analyze actuals. This interface is called Optim Query Workload Tuner (OQWT)



```
                 Rows
              Rows Actual
                RETURN
                (   1)
                 Cost
                 I/O
                  |
                  54
                  396
               >^HSJOIN
                (   2)
                153.056
                  NA
        /----------+-----------\
      54                        20
      396                        0
   >^HSJOIN                   TBSCAN
   (   3)                     (  12)
    140.872                   11.0302
      NA                        NA
 (continued below)               |
                                 20
                                 NA
                          TABLE: SYSIBM
                         SYSAUDITPOLICIES
```

PRESENTATION NOTES FOR PAGE 34

URL: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.perf.doc/doc/c0056362.html

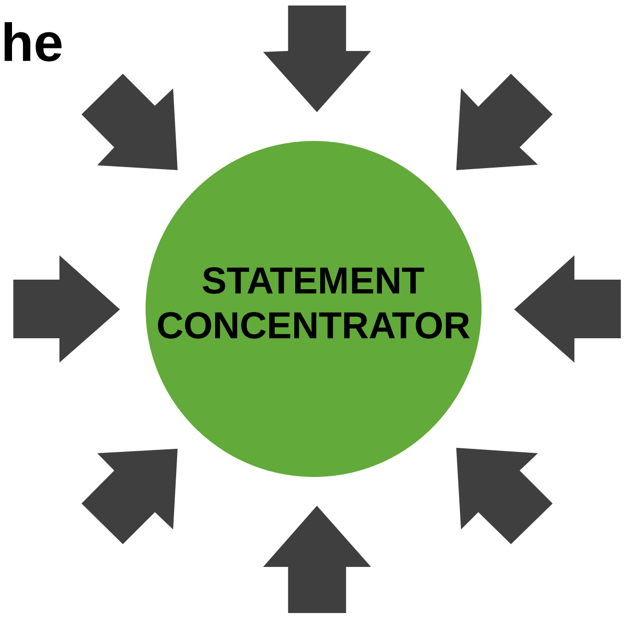
Agenda

- **Problematic SQL & Situation**
- **Response time solution**
- **SQL costs solution**
  - Snapshot Monitoring
  - Event Monitoring
  - SQL Monitoring Interfaces
  - Analyzing SQL
    - Explain Tools
    - Visual Explain
  - Statement Concentrator
- **Making Performance Improvements**
  - Database Objects
  - Better Coding
  - Design Advisor
  - DB2 Optim Query Workload Tuner
  - Other Considerations

PRESENTATION NOTES FOR PAGE 35

Statement Concentrator

- **Specifies whether dynamic statements that contain literal values use the statement cache**
- **Database configuration parameter**
  - stmt_conc OFF | LITERALS
- **CLI/ODBC configuration keyword**
  - StmtConcentrator = OFF | WITHLITERALS
- **Environment or connection attribute**
  - SQL_ATTR_STMT_CONCENTRATOR
    - SQL_STMT_CONCENTRATOR_OFF
    - SQL_STMT_CONCENTRATOR_WITH_LITERALS
- **The default setting for the configuration keyword or environment attribute is the one that's specified for statement concentration on the server**



PRESENTATION NOTES FOR PAGE 36

Statement concentrator is a feature that can also be utilized to improve the dynamic SQL query performance. It's mainly for statements that are same but differ only in the variables used in the predicate clause.

STMT_CONC db cfg:

The new stmt_conc database parameter enables statement concentration for dynamic statements. The setting in the database configuration is used only when the client does not explicitly enable or disable statement concentrator.

When enabled, statement concentrator modifies dynamic statements to allow increased sharing of package cache entries.

Statement concentrator is disabled when the configuration parameter is set to OFF. When the configuration parameter is set to LITERALS, statement concentrator is enabled. When statement concentrator is enabled, SQL statement that are identical, except for the values of literals in the statements, may share package cache entries.

- The statement concentrator with literal behavior is enabled for situations that are supported by the server. For example, the statement concentrator is not enabled if the statement has parameter markers, named parameter markers, or a mix of literals, parameter markers, and named parameter markers.

* Because statement concentration alters the statement text, it has an impact on access plan selection. The statement concentrator should be used when similar statements in the package cache have similar access plans. If different literal values in a statement result in significantly different access plans, the statement concentrator should not be enabled for that statement.

Statement Concentrator Example

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE
    WHERE EMPNO='000020'
```

and

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE
    WHERE EMPNO='000070'
```

Share the same entry in the package cache

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE
    WHERE EMPNO=:L0
```

Package cache will use the statement

§ DB2 will provide the value for :L0 (either '000020' or '000070') based on the literal used in the original statements

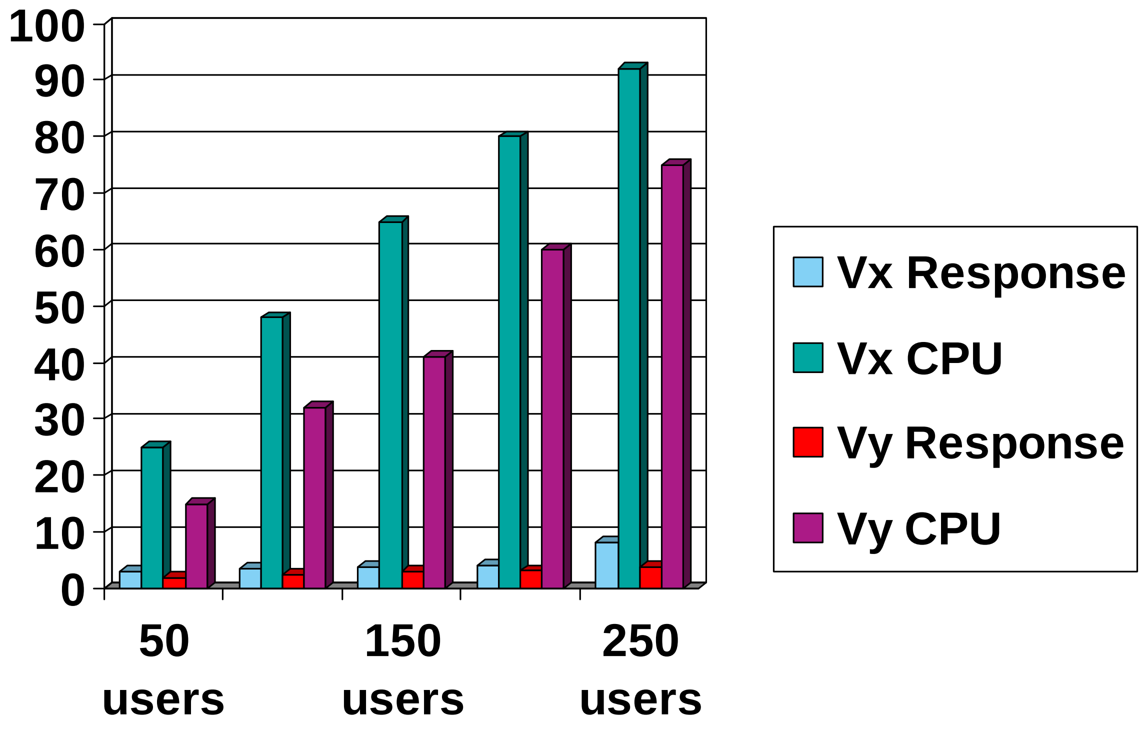
PRESENTATION NOTES FOR PAGE 37

For SQL statements that need to be executed many times, where the only difference in the statement are the variables used as predicates, it is often beneficial to prepare the SQL statement once, and reuse the query plan by using parameter markers to substitute the predicate values during runtime.

HOST VARIABLE SUBSTITUTION

After Statement Concentrator
Comparative Application Results Version X vs. Version Y

- Now it’s much faster
- SQL costs look much better!



PRESENTATION NOTES FOR PAGE 38

Now, performance objectives have been reached with the new application version of their software solution with reductions in both CPU usage and response time allowing greater capacity, giving added value to their customers.

Summary

§ In this Module you learned about:
  – Explore How to capture, analyze, and improve SQL through both better coding techniques and defining complementary objects like indexes and MQT's for your database Optim Query tuner

PRESENTATION NOTES FOR PAGE 39

The next steps…

PRESENTATION NOTES FOR PAGE 40

The Next Steps…

- Complete the Hands on Lab for this module
  - Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
  - Find the module
  - Download the workbook 10303_WB1_DB2_SQLQueryTuningwithBLUAcceleration.pdf and the virtual machine image 10300_VM1_DB2_PerformanceMonitoringAndTuning_10.5.part#.rar
  - Follow the instructions in the workbook to complete the lab
- Complete the online quiz for this module
  - Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
  - Find the module and select the quiz
- Provide feedback on the module
  - Log onto SKI, go to “My Learning” page
  - Find the module and select the “Leave Feedback” button to leave your comments

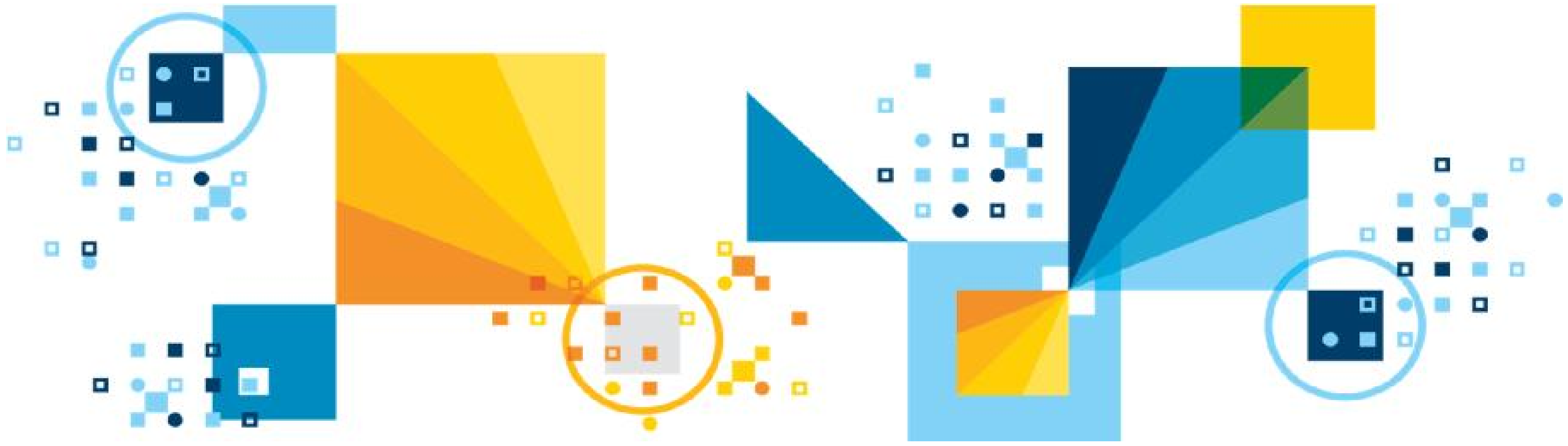
PRESENTATION NOTES FOR PAGE 41

The Next Steps…

§ The next set of modules to consider :
  – Module DB2 SQL Query Tuning with BLU Acceleration II

PRESENTATION NOTES FOR PAGE 42

Questions?
askdata@ca.ibm.com



PRESENTATION NOTES FOR PAGE 43