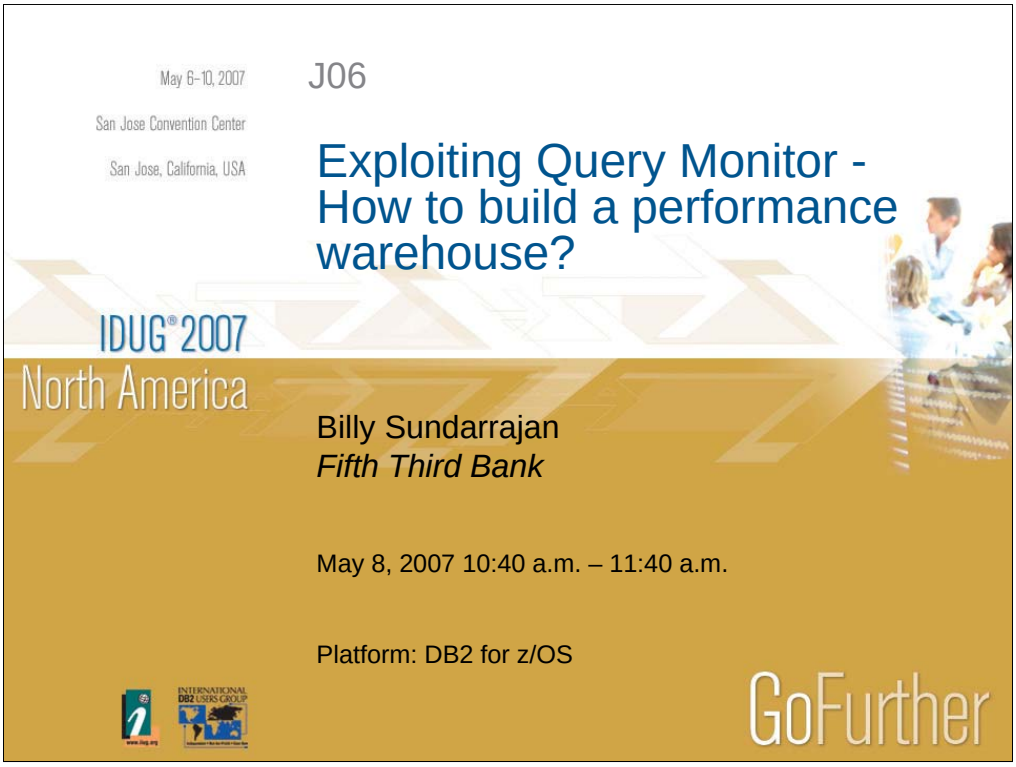


One of the significant challenges faced by most DBA teams is the ability to provide an effective data warehouse that will cater to the needs of the application programmers as well as the management team. The purpose of a performance data warehouse is usually multifold: from providing management with the ability to view things at a macro level, while providing the application team to drill down and track performance at a plan/package/SQL level. At Fifth Third, we used a combination of the Query Monitor data store, SMF101 data, and MXG processes to extract and store Plan, Package, and SQL level performance data. This presentation focuses on the challenges in designing an effective performance warehouse, and how we overcame the challenges.

Agenda

- Fifth Third Bank Environment
- Performance warehouse – Definition
- Challenges in extracting performance information
- What have we done at Fifth Third?
- Query Monitor – DB2 performance data base
- SQL performance – Summarization challenges
- Query Monitor – Best Practices
- Questions

This slide outlines the structure of the presentation.

Fifth Third Environment

- Data Sharing (2-way in Dev/Test/Load, 2-way in Prod-DW, 6-way in PROD-OLTP)
- 18TB DASD
- DB2 V8 NFM
- Data Marts, Data Warehouse, Reporting, OLTP
- Stored Procedures (SQL and Cobol)
- Majority of SQL workload is dynamic SQL (~17 Million dynamic SQLs)
- Majority of large tables populated by online LOAD

This slide describes the production database environment at Fifth Third

Main emphasis is on the fact that majority of the workload is dynamic and there are quite a few SMF records cut.

Performance warehouse – Definition

- Collection of DB2 performance information
  - Accounting information (plan, package, SQL)
    - Summary
    - Exception
  - Statistics information (SMF 100)
- Information should be
  - Detailed (useful to the application programmers, DBAs, sysprogs)
  - Easy to summarize (useful to the managers, IT departmental heads)

This slides provides the definition of a performance warehouse.

Performance warehouse is essentially a collection of DB2 performance information at various levels

Accounting (Plan, package, and SQL level), Statistics

This information is primarily derived from SMF100/101 and a SQL level monitor

Performance warehouse – Definition

- Contains derived information in addition to the DB2 accounting/statistics information.
- Extends the capabilities of the Query Monitor performance database

This slides provides the definition of a performance warehouse.

Performance warehouse is essentially a collection of DB2 performance information at various levels

Accounting (Plan, package, and SQL level), Statistics

This information is primarily derived from SMF100/101 and a SQL level monitor

Performance warehouse – Source of information

- Plan/package level information
  - SMF 101 (IFCID 3, IFCID 239)
  - Processed by MXG and/or Omegamon, Rexx
- Statistics
  - SMF100
  - Processed by MXG and/or Omegamon, Rexx
- SQL level performance information
  - Extracted from Query Monitor VSAM Datasets (data store)
  - Extracted from Query Monitor performance database

This slide describes the source of DB2 performance information.

Plan and package level information are obtained from SMF101 records. This information should be used as the primary filter when identifying resource intensive packages.

SQL level performance information is derived from the Query Monitor VSAM Datastore/Performance database.

Performance warehouse – why do we need one?

- Answer questions such as:
  - Is my dynamic SQL using this index?
  - Which program consumes the highest amount of CPU?
  - Which package performs the highest number of GETPAGES
  - Can I get a list of all SQLs that took more than 5 minutes to execute?
  - Tell me all the SQLs that ended with a sqlcode –905
  - Which dynamic SQL consumed the most CPU last month?

This slide addresses the need for a performance warehouse.

As the amount of data managed by a DBA (I.e., terabytes or transactions/DBA) increase, it has become imperative that automated processes exist to collect and store performance information. This information allows the DBA to answer a lot of complex questions that are posed in real-life situations.

Performance warehouse – why do we need one?

- Increasingly important to have current and historical performance data to project estimates
- Ability to measure impact to CPU utilization from changes
- Information easily available to DBAs, developers, and managers

This slide addresses the need for a performance warehouse.

As the amount of data managed by a DBA (I.e., terabytes or transactions/DBA) increase, it has become imperative that automated processes exist to collect and store performance information. This information allows the DBA to answer a lot of complex questions that are posed in real-life situations.

Challenges in extracting information

- Volume of data
  - # of SMF records
  - Establishing the summarization level
  - Storage for holding the performance data
- Complexity of SQL level information
  - SQL statements can be up to 2 MB
  - CCSID issues
  - Performance data collection overhead
  - Parameter markers versus literals
  - Establishing a trend/pattern for dynamic/ static SQL

The primary challenges in extracting performance data are

(a) Volume of data to be processed

(b) Challenges in obtaining SQL level performance info

(c) Handling dynamic SQL Performance information

What have we done at Fifth Third?

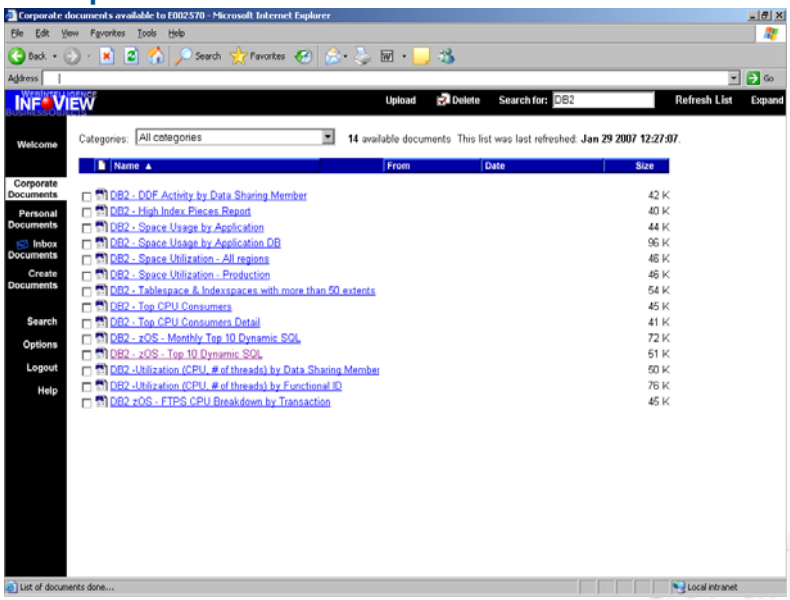
- Designed an extensive Performance warehouse - Contains
  - Plan/Package Information
    - Extracted from SMF101 records
    - Processed by MXG/Rexx
  - SQL level performance Information
    - Extensive use of Query Monitor Data store
    - Custom built tables to store
      - Top 100 Dynamic SQL (DAILY)
      - Top 100 Dynamic SQL (MONTHLY)
      - Exception SQL (DAILY)
      - SQL utilization by work-station correlation variables

This slide describes the performance warehouse at Fifth Third and the key reports that are generated out of the performance warehouse.

What have we done at Fifth Third?

- Performance warehouse - Used for monitoring
  - Top 10 SQL
  - CPU consumption trends
  - Exception information
  - Impact of database changes/new release of software
- Performance data is stored in custom designed tables
- Used Query Monitor's DB2 performance database tables as a storage area for DBA analysis

List of reports



This slide describes the various reports that are being produced and distributed on a automated basis at Fifth Third.

Query Monitor Data Performance database

- Query Monitor Performance data base – definition
- Population of Query monitor performance database
- Design of Performance warehouse tables
- ETL process for populating the performance warehouse
- Summary monitoring versus Exception monitoring

This slides provides the basic definition of a Query monitor performance database.

Query Monitor Data Performance database

- What is Query Monitor Performance data base?
  - Group of DB2 tables designed to hold information extracted from Query monitor VSAM Data store
    - Summary
    - Exception
    - Adverse SQL codes
    - Commands
  - Tables begin with CQM22 or CQM21
  - LOBs for storing SQL text

This slides provides the basic definition of a Query monitor performance database.

Query Monitor Performance database

- How do I populate the Query Monitor Performance Database?
  - Flexible utilities to extract data from VSAM data store - CQM@WDB2
  - Data can be easily loaded using DB2LOAD
  - SQL text loaded using CQM@ITXT program
    - New parameter to perform COMMITs

Describe the programs used to populate the query monitor performance database

The programs use the flat files produced by CQM@WDB2 as the input, and use either DB2 LOAD or CQM@ITXT.

Query Monitor Performance Database

- Summary information (level of summary can be controlled through monitoring profiles)
  - CQM22_SUMM_METRICS
  - CQM22_SUMM_TEXT
  - CQM22_SUMM_OBJECTS
- Information about SQL Codes
  - CQM22_SQLCODE_DET
  - CQM22_SQLCODES
  - CQM22_SQLCODE_TEXT

Provides an overview of the various tables in the query monitor performance database.

Five main categories:

Summary Metrics

Exception SQLCODES

Commands

Intervals

Exceptions

Query Monitor Performance Database

- Information about collections intervals
  - CQM22_INTERVALS
- Information about DB2 commands issues
  - CQM22_DB2_COMMANDS
- Information about exceptions triggered
  - CQM22_EXCEPTIONS
  - CQM22_EXCP_CALLS
  - CQM22_EXCP_TEXT
  - CQM22_EXCP_HOSTV
  - CQM22_EXCP_OBJS

Provides an overview of the various tables in the query monitor performance database.

Five main categories:

Summary Metrics

Exception SQLCODES

Commands

Intervals

Exceptions

Query Monitor Performance Database

- How is SQL Text stored?
  - SQL Text information is stored as LOB
  - User defined function available for data stored in version 2.1 to convert SQL from various CCSIDs
    - EBCDIC – 37
    - ASCII (Solaris) – 819
    - Unicode - 1208
    - ASCII – 1252
  - TEXT_TOKEN Uniquely identifies a piece of SQL text – It is similar to a hash token

This slide describes how SQL statements are stored in the Query Monitor Database Depending on the version of query monitor data is either stored as BLOB or CLOB

Designing tables for Performance warehouse

- Custom tables
  - Ability to augment information collected in DB2
  - User specific information
  - Customized summaries
  - Ability to present information without exposing the Query Monitor tables (in our installation approx 50 GB/day)

This tables describes how “custom tables” were designed to store a subset of the performance data from the Query Monitor datastore. The custom tables augment the rich capability provided by Query Monitor.

Designing tables for Performance warehouse

- Categories of performance data
  - Plan level performance data
  - Package level performance data
  - SQL level performance data
    - Summary
    - Exception

Three main categories of performance data: Plan, Package, SQL

SQL has two main groupings: Summary, Exception.

Table Design – Plan level

- PLAN level accounting information
  - Granularity – Date, Subsystem/member ID, Plan name, Authorization ID/Job
  - Key fields to be captured – Same as the ones in the accounting short report
  - Captured by a Rexx routine which process the MXG summary report
  - Rexx can be easily modified to handle DB2 PE/third part accounting report summaries

This slide describes the table design for storing Plan level performance data.

The information is stored at a Extraction Date, Subsystem/Member, Plan, Auth ID level.

Information stored at a Member level provides significant insight into the workload balancing and effect of location on the SYSPLEX.

Table Design – Package level

- PACKAGE level accounting information
  - Granularity – Date, Subsystem/member ID, Plan name, Authorization ID/Job, Package
  - Key fields to be captured – Key fields from the accounting summary long report
  - Captured by a Rexx routine which process the MXG summary report
  - Rexx can be easily modified to handle DB2 PE/third part accounting report summaries

This slide describes the table design for storing Plan level performance data.

The information is stored at a Extraction Date, Subsystem/Member, Plan, Package, Auth ID level.

Information stored at a Member level provides significant insight into the workload balancing and effect of location on the SYSPLEX.

Table Design – SQL level information

- Tables for storing SQL performance
  - Top 100 dynamic SQL summary level information (daily)
  - Top 100 static SQL summary level information (daily)
  - Exception SQL (daily)
  - Top 100 dynamic SQL (monthly)
  - SQL activity by Transaction – for dynamic SQL with literals

Multiple table were designed for storing SQL level performance information. Grouped by static vs dynamic, summary vs exception

Table Design – SQL level information

- Tables for storing SQL performance
  - Key fields
    - TEXT_TOKEN – Allows the ability to track performance over time
    - RANK – pre-calculated
    - AUTHID, Plan, Extraction Date
    - CPU, Elapsed time, GETPAGES, Synch I/Os, Wait time
    - SQLTEXT
    - #executions

This slide addresses the key information needed for storing SQL performance data. TEXT_TOKEN provides the ability to track performance over a period of time.

Extracting SQL Performance data

- Extraction of SQL level performance data consists of three main steps:
  - Extract data from the Query Monitor VSAM datasets
  - Load data into Query Monitor Data performance database
  - Use custom stored procedures to populate performance warehouse

Describes the various steps in extracting SQL performance data

1. Extract data from Query Monitor VSAM data store
2. Load data to the Query Monitor performance database
3. Use custom stored procedures to populate the performance warehouse

SQL Performance data –Extract from VSAM datasets

- Extract data from the VSAM datasets
  - Use Query Monitor Unload program – CQM@WDB2 to extract data for the previous 24 hours
  - If working on multiple data sharing group(s), make sure data is extracted from all Query Monitor subsystems
  - Extract at least METRICS, SQLTEXT, EXCEPTIONS
  - Query Monitor uses TEXT_TOKEN to uniquely identify a dynamic SQL

This slide describes the steps involved in extracting data from the query monitor VSAM data store.

Key things to remember:

-Extract atleast METRRICS, SQLTEXT, EXCEPTIONS

-Extract from all QM subsystems

SQL Performance data – Load perf. database

- Load data to Query Monitor performance database
  - Use DB2LOAD to load non LOB tables
  - Use CQM@ITXT to load the Query Monitor DB2 tables which contain SQL text (introduced a new COMMIT frequency option)
  - Data load replaced on a daily basis – Approx 50 GB of data is loaded in our installation.
  - Query Monitor performance database used for analysis by DBAs

Describes the parameters and the program used by Query Monitor for unloading data from the VSAM data store.

SQL Performance data – Load perf. warehouse

- Extracting summary data
  - Use Query Monitor Data performance database as a working storage/repository
  - Custom stored procedure to extract the
    - Top 100 SQLs (static, dynamic)
    - Exceptions
    - Summaries by transaction (for dynamic SQL using literals)
  - Store performance data in custom tables
  - Custom tables insulate the Query Monitor database from complex queries (due to the size)

This slide describes the steps involved in extracting the summary data from the Query Monitor performance database

Exception monitoring vs Summary monitoring

- Exception monitoring
  - Identifies if an individual instance of an SQL does not perform within pre-set expectations
  - Can be based on CPU time, elapsed time, GETPAGES, wait time
  - Limit can be set on the number of exceptions
  - Why is it important?
    - When using parameter markers, an single instance of a bad performer may be hidden
    - Averages can be sometimes misleading

This slide describes the key aspects of Exception monitoring.

Exception monitoring can highlight significant issues which can be hidden when looking at averages.

Summary monitoring

- Summary monitoring
  - Identifies if an SQL is one of the top consumers of CPU
  - Usually based on total CPU usage
  - When individual SQLs cannot be consolidated due to literals/complex variations, use the correlation ID or workstation variables.
  - Client accounting variables can be set by using JDBC methods or SQLESETI API

This slide addresses the key aspects of summary monitoring.

When de-sensitizing parameter markers does not provide granularity, client accounting variables provide a pragmatic alternative.

Dynamic SQL summarization challenges

- Query monitor uses a hash token for uniquely identifying a dynamic SQL
- SQL using literals will appear as separate SQL
- Query Monitor Datastore used to identify SQLs with literals and re-written with parameter markers
- Enhancement request SENT to IBM to allow the option for a hash token without parameter markers

  (in collector as well as post-collector)

This slide addresses the SQL summarization challenges, key being the ability to track the performance of a dynamic SQL across time periods, and de-sensitizing the literals.

Static SQL- summarization challenges

- Query monitor uses the statement# and the package/DBRM to identify a SQL
- Same SQL in multiple DBRMs cannot be combined
- Enhancement request sent to IBM to have a has token created for static SQLs.
- Gives the flexibility to summarize across plans

This slide address the key challenges of summarizing static SQL key being the ability to track the same statement across multiple DBRMs/packages.

Query Monitor – Best practices

- Obtaining SQL level statistics is not free. There is a overhead associated with it.
- Guidelines for reducing overhead
  - Understanding the workload is key
  - Exclude packages/plan which require the highest performance
  - Use multiple monitoring profiles – Use multiple profiles to decide the level of monitoring to be done at different times of the day
  - Level of summarization affects the amount of overhead – OPTKEYS CALLS has the highest overhead.

This slide details the best practices for reducing overhead of using Query monitor to track SQL level performance information.

Query Monitor – Best practices

- Guidelines for reducing overhead
  - Do not use multiple OPTKEYS (Beware of the multiplier effect)
  - Specify limits
    - MAX_SQLCODES
    - MAX_SQLCODES_DETAIL
    - ALERT_LIMIT

This slide details the performance impact associated with using multiple OPTKEYS.

Query Monitor – Best practices

- Guidelines for reducing overhead
  - Use exception profiles
    - Set your exception thresholds high and work downwards
  - One size fits all may not be the best approach
  - Exclude SQL codes which you do not want monitored (May want to exclude –514)
  - CATALOG_OBJECTS set to NO if not needed (useful for dynamic SQL)

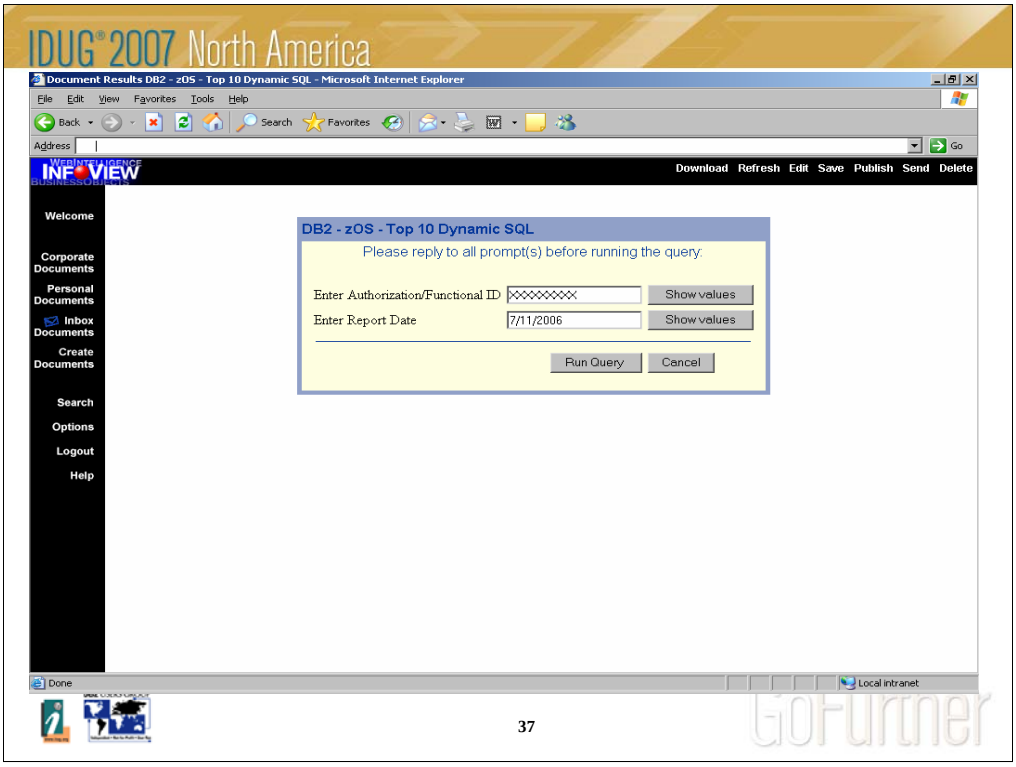
This slide addresses the guidelines on the usage of exception profiles.

Distribute Performance information

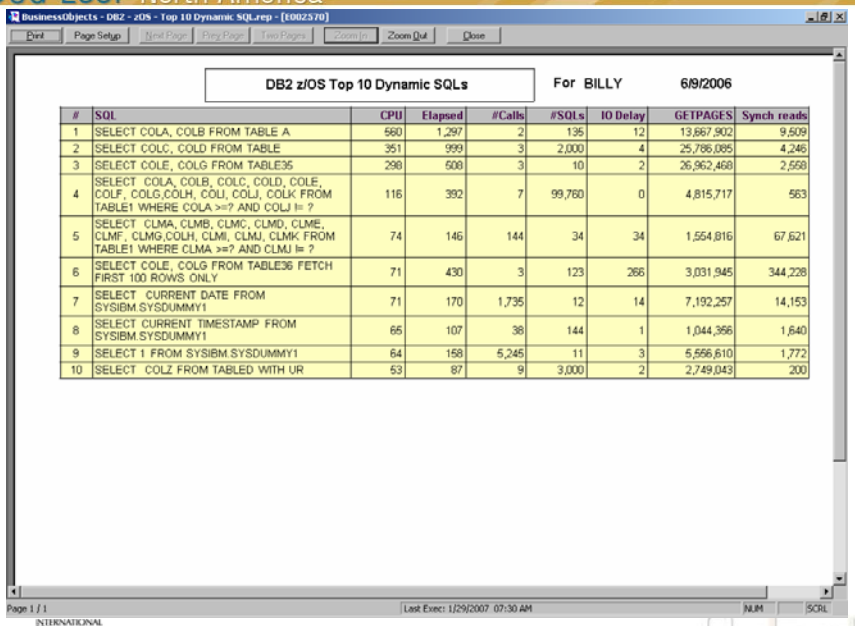
- Use Business Objects as a distribution tool
  - Top 100 Dynamic SQL – daily
    - Allows the user to enter their Authorization ID and Date as filters
  - Top 100 Dynamic SQL – monthly
    - Allows the users to enter their Authorization ID and Month as filters
  - Top 100 Exception SQL
    - Allows the users to enter their Authorization ID and Date as filters
  - SQL utilization by Transaction
    - Used where literals have caused summarization challenges

Describe how the performance information is disseminated at Fifth third.

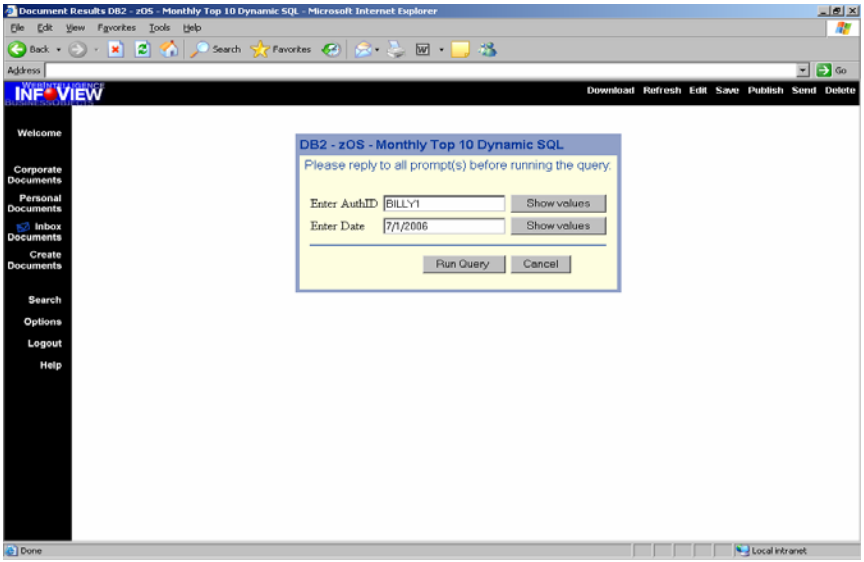
Business objects is used as the primary vehicle for distribution of information.



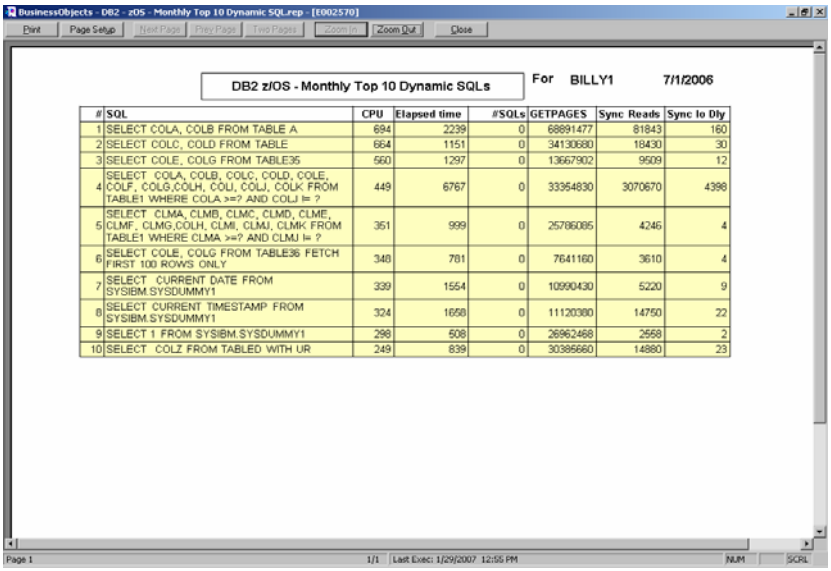
Report lists the Top 10 Dynamic SQL for a given Auth ID



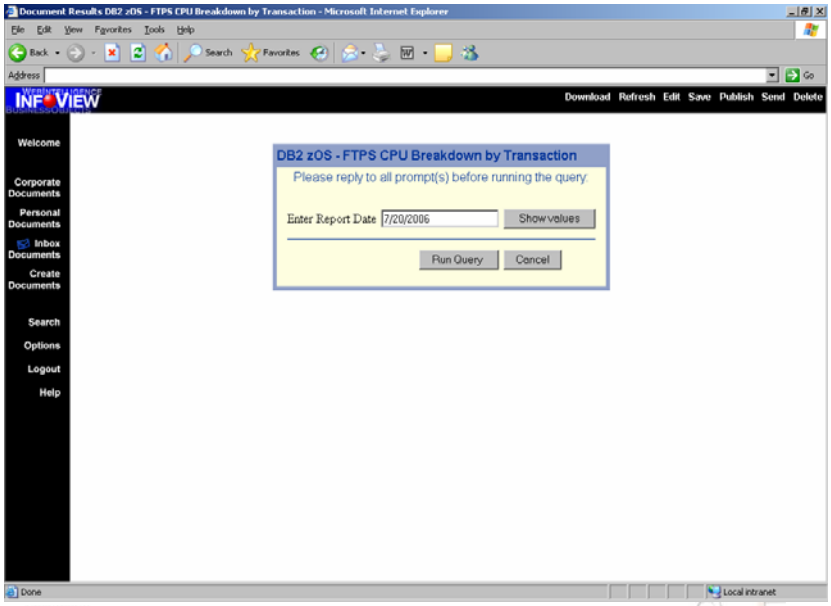
#	SQL	CPU	Elapsed	#Calls	#SQLs	IO Delay	GETPAGES	Synch reads
1	SELECT COLA, COLB FROM TABLE A	660	1,297	2	135	12	13,667,902	9,509
2	SELECT COLC, COLD FROM TABLE	351	999	3	2,000	4	25,786,085	4,246
3	SELECT COLE, COLG FROM TABLE35	298	508	3	10	2	26,962,468	2,558
4	SELECT COLA, COLB, COLC, COLD, COLE, COLF, COLG,COLH, COLI, COLJ, COLK FROM TABLE1 WHERE COLA >=? AND COLJ != ?	116	392	7	99,760	0	4,815,717	563
5	SELECT CLMA, CLMB, CLMC, CLMD, CLME, CLMF, CLMG,COLH, CLMI, CLMJ, CLMK FROM TABLE1 WHERE CLMA >=? AND CLMJ != ?	74	146	144	34	34	1,554,816	67,621
6	SELECT COLE, COLG FROM TABLE36 FETCH FIRST 100 ROWS ONLY	71	430	3	123	266	3,031,945	344,228
7	SELECT CURRENT DATE FROM SYSIBM.SYSDUMMY1	71	170	1,735	12	14	7,192,257	14,153
8	SELECT CURRENT TIMESTAMP FROM SYSIBM.SYSDUMMY1	65	107	38	144	1	1,044,366	1,640
9	SELECT 1 FROM SYSIBM.SYSDUMMY1	64	158	5,245	11	3	5,556,610	1,772
10	SELECT COLZ FROM TABLED WITH UR	53	87	9	3,000	2	2,749,043	200



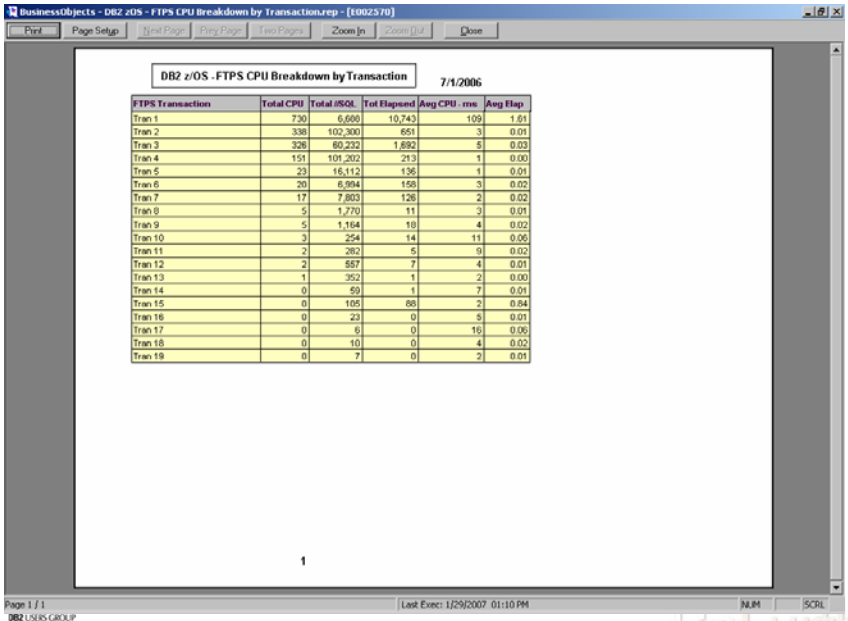
Report lists the Top 10 Monthly Dynamic SQL for a given Auth ID



#	SQL	CPU	Elapsed time	#SQLs	GETPAGES	Sync Reads	Sync Io Dly
1	SELECT COLA, COLB FROM TABLE A	694	2239	0	68891477	81843	160
2	SELECT COLC, COLD FROM TABLE	664	1151	0	34130680	18430	30
3	SELECT COLE, COLG FROM TABLE35	560	1297	0	13667902	9509	12
4	SELECT COLA, COLB, COLC, COLD, COLE, COLF, COLG,COLH, COLI, COLJ, COLK FROM TABLE1 WHERE COLA >=? AND COLJ != ?	449	6767	0	33354830	3070670	4398
5	SELECT CLMA, CLMB, CLMC, CLMD, CLME, CLMF, CLMG,COLH, CLMI, CLMJ, CLMK FROM TABLE1 WHERE CLMA >=? AND CLMJ != ?	351	999	0	25786085	4246	4
6	SELECT COLE, COLG FROM TABLE36 FETCH FIRST 100 ROWS ONLY	348	781	0	7641160	3610	4
7	SELECT CURRENT DATE FROM SYSIBM.SYSDUMMY1	339	1554	0	10990430	5220	9
8	SELECT CURRENT TIMESTAMP FROM SYSIBM.SYSDUMMY1	324	1658	0	11120380	14750	22
9	SELECT 1 FROM SYSIBM.SYSDUMMY1	298	508	0	26962468	2558	2
10	SELECT COLZ FROM TABLED WITH UR	249	839	0	30386680	14880	23



This slide addresses the FTPS CPU Breakdown by Transaction – Used for situations where dynamic SQLs cannot be grouped due to the multitude of combinations.



FTPS Transaction	Total CPU	Total #SQL	Tot Elapsed	Avg CPU - ms	Avg Elap
Tran 1	730	6,666	10,743	109	1.61
Tran 2	338	102,300	651	3	0.01
Tran 3	326	60,232	1,692	5	0.03
Tran 4	151	101,202	213	1	0.00
Tran 5	23	16,112	136	1	0.01
Tran 6	20	6,994	158	3	0.02
Tran 7	17	7,803	126	2	0.02
Tran 8	5	1,770	11	3	0.01
Tran 9	5	1,164	18	4	0.02
Tran 10	3	254	14	11	0.06
Tran 11	2	262	5	9	0.02
Tran 12	2	557	7	4	0.01
Tran 13	1	352	1	2	0.00
Tran 14	0	59	1	7	0.01
Tran 15	0	105	88	2	0.84
Tran 16	0	23	0	5	0.01
Tran 17	0	6	0	16	0.06
Tran 18	0	10	0	4	0.02
Tran 19	0	7	0	2	0.01

Enhancement requests sent to IBM

- Allow for Sampling
- Expand OPTKEYS to turn on/off collection of object/buffer pool/lock information at the monitoring profile level
- Hash token for static SQLs
- De-sensitize literals in SQLs (collector/post-collector)
- Ability to load SQL text using IBM Load utility
- Externalize # of executions

This slide addresses the key enhancement request sent to IBM.

Allowing for sampling, and de-sensitizing literals in dynamic SQLs are some of the key requests.

Session: J08

Session Title: Exploiting Query Monitor - How to build a performance warehouse?

Thanikachalam “Billy” Sundarrajan

billysundar@yahoo.com

Session: J08

Session Title: Exploiting Query Monitor - How to build a performance warehouse?

Questions ?

GoFurther