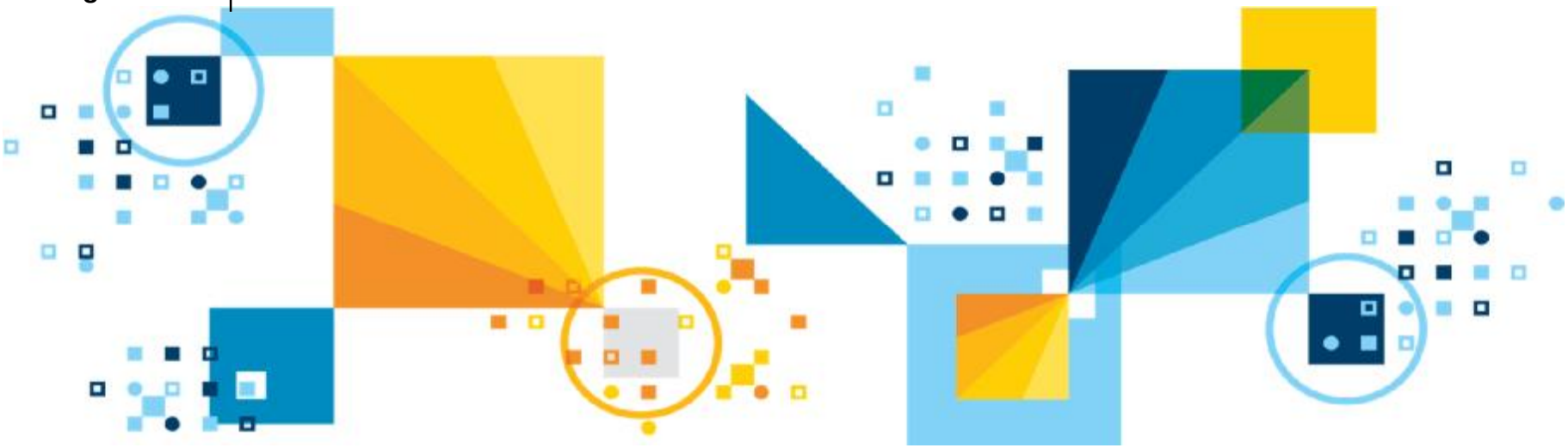
Introduction to DB2 LUW Performance

Module ID	10300
Length	1 hour + 1.5 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:
  - Modules from Curriculum, DB2 10 for Linux, Unix, and Windows
  - 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 understand
  - Overall performance tuning, db2 best practices and considerations to achieving a db2 database system.

PRESENTATION NOTES FOR PAGE 3

Agenda

- **Performance Tuning Overview**
- **DB2 Architecture Fundamentals**
- **Hardware Planning Rules of Thumb**
- **Database Design & Recommendations**

PRESENTATION NOTES FOR PAGE 4

In this short introduction to performance monitoring we consider the concept and the rational to performance tuning and monitoring. We briefly look at the DB2 architecture with the emphasis being on the importance of knowing what DB2 components, in the form of threads, are at work, what resources they consume, and how to monitor them.

Being able to correctly size your system is of the most importance to your ROI and to the system.

The right size affects your system's prospects for providing acceptable performance over it's projected "life". The primary "pieces" to a system, the CPU, Storage, and Memory are considered, sizing estimates and ratios between these pieces are conveyed.

When you create a DB2 database without regards to default parameters and settings, a less than optimized configuration for you're DB2 database can easily result. We look at default settings and discuss the performance ramifications associated with these parameters.

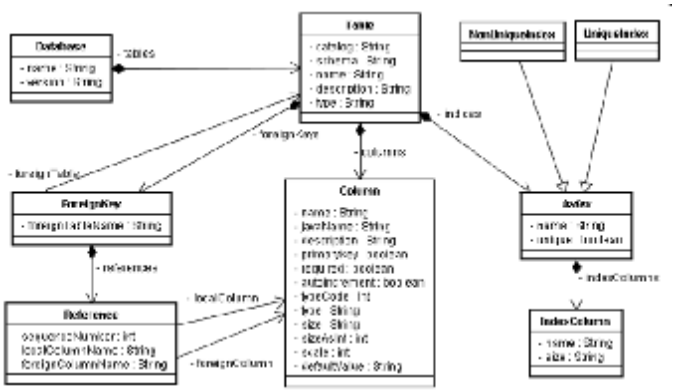
Agenda

- **Performance Tuning Overview**
- **DB2 Architecture Fundamentals**
- **Hardware Planning Rules of Thumb**
- **Database Design & Recommendations**

PRESENTATION NOTES FOR PAGE 5

Performance Tuning Begins with a Problem

§ Database or application design issues: *How should we design our application/database for the best performance?*



§ User complaint(s): *This report is taking too long*

§ System capacity: *Can the current system handle a 50% increase in the size of our database?*

PRESENTATION NOTES FOR PAGE 6

Performance tuning must begin with an issue or a problem to be solved. The problem can either be a complaint from a user, a decision about the application or database design, or a capacity planning issue.

Having DB2 Performance Issues?

§ Performance problems are trickier than functional problems
- Symptoms may provide no clues about the problem source, e.g. you observe general slowdown and excessive lock timeouts
- A performance problem can be intermittent
- Performance problems can be avoided

§ Some common reactions to performance problems (especially if you're new at this…)
- Panic
- Buy more hardware (CPU, memory, disk, etc.)
- Blame DB2…
  - or AIX / Windows / Linux…
  - or IBM / HP / Sun /…
- Take "shots in the dark" at the problem
  - Making almost random changes based on not much data

PRESENTATION NOTES FOR PAGE 7

So why have a DB2 Performance Clinic: Many people make the claim that performance tuning and troubleshooting is an Art, or for highly trained specialists However, DB2 provides a rich set of resources to Monitor, Tune and detect performance problems on your systems, with a little bit of understanding of your system combined with the knowledge of “where clues” can be found, the premise of it being an “Art form” is debunked allowing you to effectively Monitor and Tune DB2 systems and optimize performance.

In a worse case scenario, you are operating completely or near completely in the dark, under these circumstances when problem occurs, typically when you are most reliant on the system, the reaction to panic and throw hardware at the problem prevails. Of course we know that this is a costly and often unnecessary resolution in terms of hardware, maintenance and software costs.

What is Performance?

- The way a system behaves in response to a particular workload
- Measured in terms of system response time, throughput, and resource utilization
- Affected by:
  - Resources available on the system
  - How well those resources are used and shared
- Typically tuned to improve cost-benefit ratio
  - Processing larger, or more demanding workloads without increasing processing costs
  - Obtaining faster system response times, or higher *throughput*, without increasing processing costs
  - Reducing processing costs without degrading service to users

PRESENTATION NOTES FOR PAGE 8

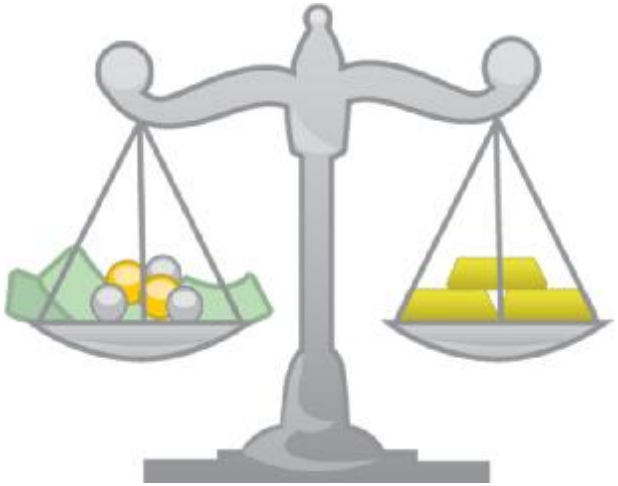
When you get right down to it, performance is represented as an assessment based on a combination of measurements of how your system responds under a particular workload.

A workload is made up of a collection of requests originating from an application made on a system.

So naturally the more capacity a system has, the larger the workload can be, but this is not without saying that to get the most out of your system, its available resources, storage, cpu, memory …, needs to be allocated efficiently.

Performance Tuning Limits

- How much time and money should be spent?
  - Assess the degree to which the investment will help the users
- Tuning can often help improve
  - Response times
  - Throughput problems
- More significant problems may require
  - More disk storage
  - Faster/additional CPUs
  - More memory
  - Faster network



PRESENTATION NOTES FOR PAGE 9

When considering how much time and money should be spent on improving system performance, be sure to assess the degree to which the investment of additional time and money will help the users/consumers of the system.

By measuring things like response times and throughput, thresholds for system performance can be established. Once you have a threshold you can say things like “the system is tuned to 95% capacity”. These thresholds will help you know when performance is getting “bad” OR what is more germane here is when tuning efforts can be cost effective..

To derive your thresholds, you first establish a benchmark and compare that to the hardware specifications, where is your system in respect to the capacities of these devices? like I/Os and instructions per second. Of course this kind of assessment is easier said than done since you are required to have knowledge of your workload and its base requirements and THIS IS compounded by the recursive nature of performance tuning so that when you tune your system your workload’s requirements actually change based on configurations and declarations, this is one reason why scalability in a systems architecture becomes an important feature. (Note: scalability has often been associated with “growing a system”, now the ability to “slice and dice” introduces “reducing a workload” into the equation)

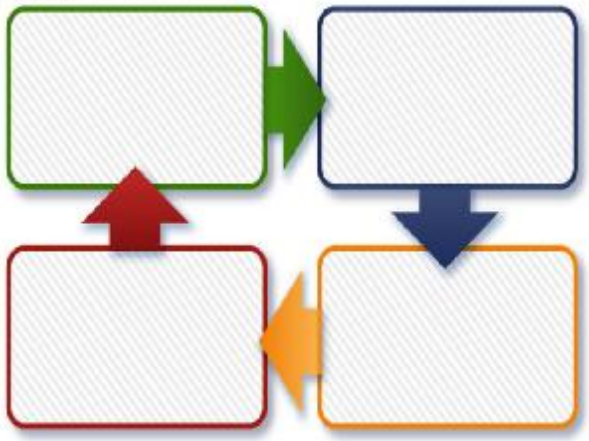
However, there is a point beyond which additional tuning cannot help. At this point, consider revising your goals and expectations. For more significant performance improvements, you might actually need to add more disk storage, faster CPUs, additional CPUs, more main memory, faster communication links, or a combination of these.

Benchmarking: A measured approach to tuning

§ Normal part of the application development life cycle
  – Involves application developers and database administrators

§ Determines current performance and can be used to improve application performance

§ Based upon controlled conditions
  – Repeatedly running SQL from your application
  – Change some of the following, for example, between iterations:
    • System configuration
    • SQL
    • Indexes
    • Table space configurations
    • Hardware configurations
  – Repeat until the application runs as efficiently as possible



§ Characteristics of good benchmarks include:
  – Repeatable tests
  – Each iteration starts in the same system state
  – No other applications are unintentionally active in the system
  – Hardware and software used match production environment

PRESENTATION NOTES FOR PAGE 10

Benchmark testing helps you to understand how the database manager responds to different conditions.

When these conditions are changed from maybe a single configuration change to maybe a new release of an application you can quantify the affect it/they have on performance. (provided you have a benchmark)

You can create benchmark scenarios that test the entire application or specific workload attributes like deadlock handling, utility performance, different methods of loading data, read efficiencies … typically on a per user or per transaction basis. You might begin by running the application in a conventional fashion where all application components are run, as expected. As you narrow down any sources to performance problem, you can develop specialized test cases that limit the scope of the function that you are testing. The specialized test cases need not emulate an entire application to obtain valuable information. Start with simple measurements, and increase the complexity only if necessary.

Benchmark tests require that a repeatable environment can be achieved, so that the test runs are made under the same conditions by doing so you will have results that can be legitimately compared.

How to avoid performance problems

- Know your environment
  - OS
  - Hardware
  - Processing
  - Memory
  - Storage
- Understand how DB2 works and how to work DB2
  - Architecture
  - Requirements
- Start with a strong foundation
  - IBM Optim – Integrated Data Management over the life of your system

    http://www-01.ibm.com/software/data/optim
  - Best Practices

    http://www.ibm.com/developerworks/data/bestpractices
  - Autonomics
  - Establish monitoring early on

PRESENTATION NOTES FOR PAGE 11

There are a few key things that you can do to help avoid performance problems from occurring. It starts with knowing your system, both it's operating system (requirements and configurations) AND the hardware that it runs on.

Strive to configure your DB2 system to perform well from the beginning of the systems life cycle, start with a strong foundation and continue to monitor and tune over the lifetime of the application.

Because Data Management issues such as performance don't exist in a silo and constantly change over the life of a system, IBM now offers IBM Optim Integrated Data Management software to :

enable organizations to more efficiently and effectively in response to emergent, data-intensive business opportunities, meet service level agreements for data-driven applications, comply with data privacy and data retention regulations and grow the business while driving down total cost of ownership.

Take advantage of published best practices and built in performance enhancing components like STMM and Autoconfigure utilities.

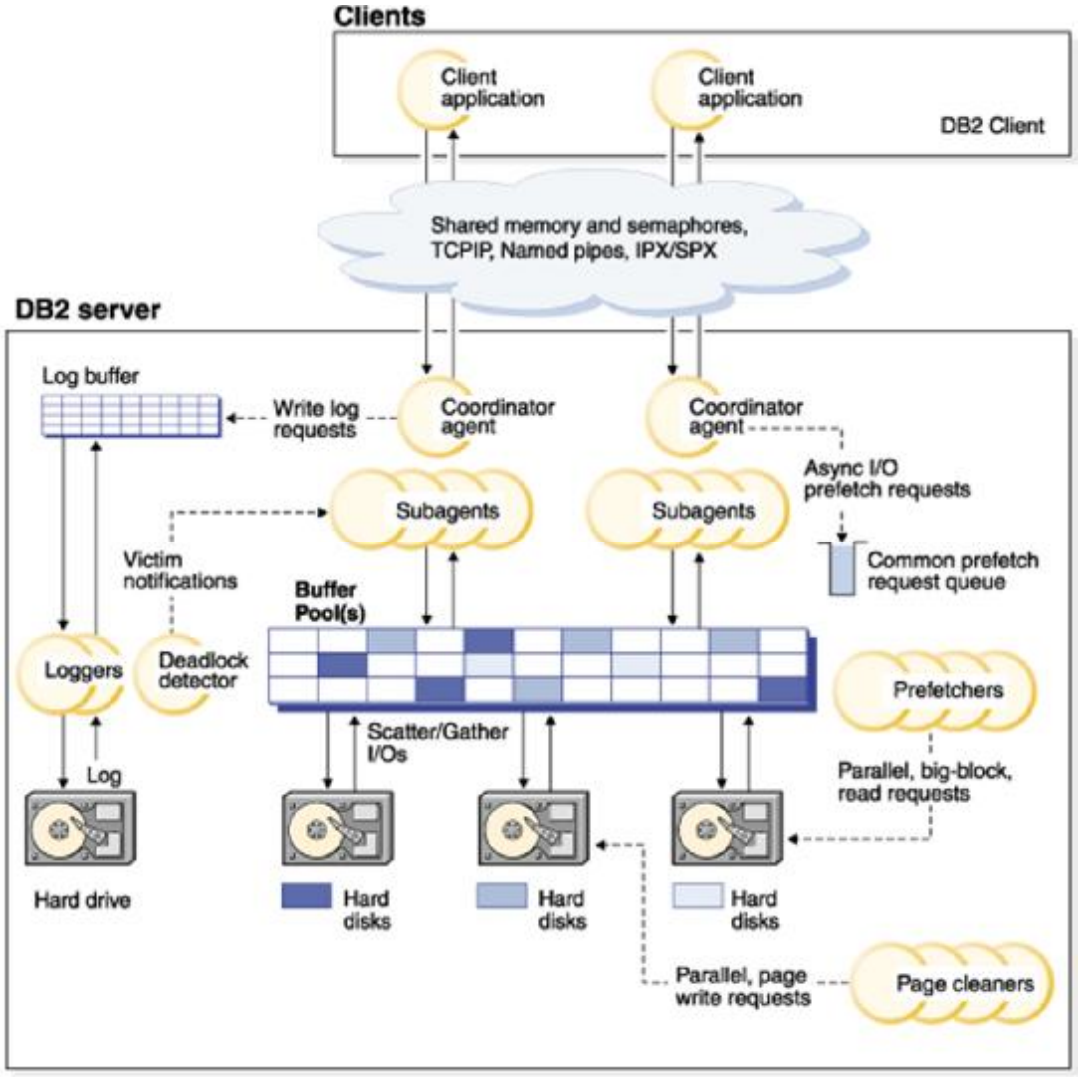
If there is one site that best complements this clinic, it is the db2 best practices portal on developerworks, this site is highly recommended.

Agenda

- Performance Tuning Overview
- **DB2 Architecture Fundamentals**
- Hardware Planning Rules of Thumb
- Database Design & Recommendations

PRESENTATION NOTES FOR PAGE 12

Understand DB2's Architecture



PRESENTATION NOTES FOR PAGE 13

(Here we want to emphasize the importance of knowing the DB2 architecture, you might want to give an architectural overview but this could be very brief too..)

There are a lot of components that make up a database system, which ones, how, and how much they are used can of course affect performance but being aware and knowledgeable about what these components need and do, will help you to ensure better performance and have a better chance at achieving timely resolutions when something suddenly happens.

##

In order to better understand factors that can affect performance, an understanding of DB2's architecture is important. On the client side, local or remote applications are linked with the DB2 client library. Local clients communicate using shared memory and semaphores; remote clients use a protocol, such as named pipes (NPIPE) or TCP/IP. On the server side, activity is controlled by engine dispatchable units (EDUs), which are implemented as threads and shown as circles or groups of circles. Using threads provides several performance advantages, such as better use of memory, fewer operating system resources, and faster context switches than processes .

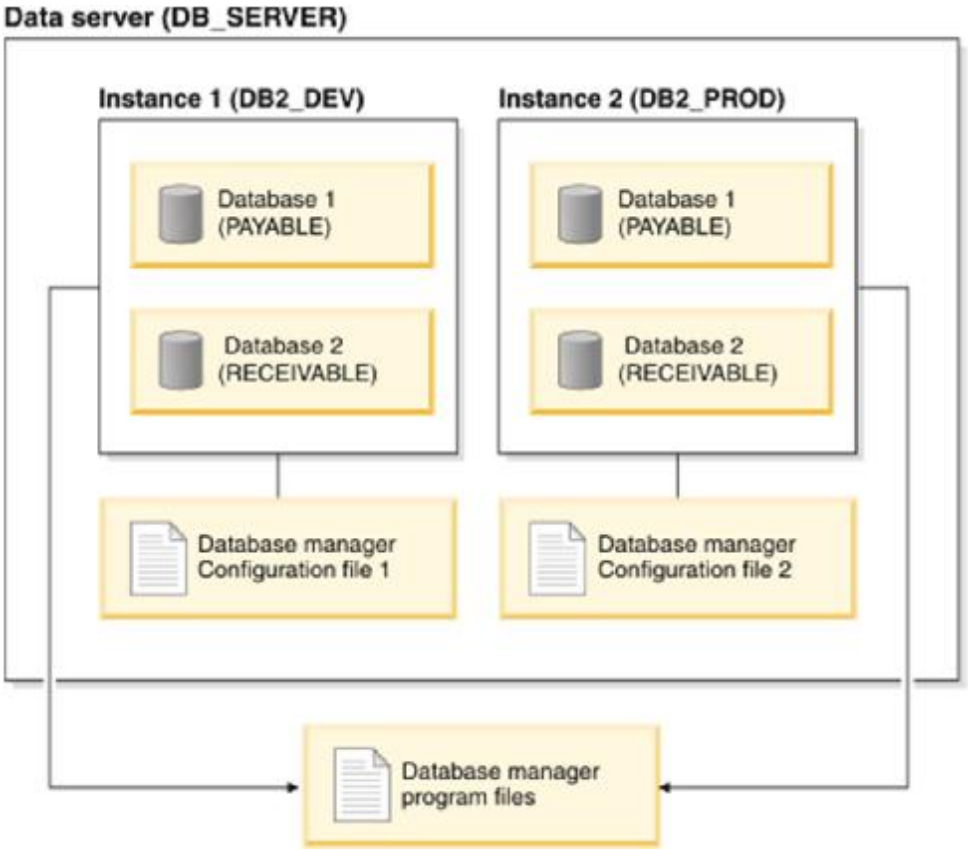
DB2 agents, the most common type of EDU, are the agents that perform most of the SQL and XQuery processing on behalf of applications. Prefetchers and page cleaners are other common EDUs. Subagents may also assist in processing requests if the machine on which the server resides has multiple processors or is part of a partitioned database environment. All agents and subagents are managed by a pooling algorithm that minimizes the creation and destruction of EDUs.

Buffer pools are also a very important part of the DB2 architecture. These are areas of database server memory where pages of user data, index data, and catalog data are temporarily moved and can be modified. They are key to database performance as data can be accessed much faster from memory than from disk. The configuration of buffer pools, as well as prefetcher and page cleaner EDUs, controls how quickly data can be accessed by applications.

Prefetchers retrieve data from disk and move it into a buffer pool before applications need the data. Agents of the application send asynchronous read-ahead requests to a common prefetch queue. As prefetchers become available, they implement those requests by using big-block or scatter-read input operations to bring the requested pages from disk into the buffer pool. If you have multiple disks for data storage, the data can be striped across those disks. Striping enables the prefetchers to use multiple disks to retrieve data simultaneously. Page cleaners move data from a buffer pool back to disk. Page cleaners are background EDUs that are independent of the application agents. They look for pages that have been modified, and write those changed pages out to disk. Page cleaners ensure that there is room in the buffer pool for pages that are being retrieved by prefetchers. Without the independent prefetchers and page cleaner EDUs, the application agents would have to do all of the reading and writing of data between a buffer pool and disk storage.

DB2 Instances

- Stand-alone DB2 environment
- Can have multiple instances per data server/OS instance
- All instances share the same executable binary files
- Each instance has its own configuration



PRESENTATION NOTES FOR PAGE 14

All DB2 constructs are organized and associated with an Instance.

An instance is an endpoint for establishing things like resource allocation, security and communications for your databases.

Each OS instance (with respect to system partitioning) may have multiple DB2 instances running and each instance can have multiple databases. We will look at the constructs in a database on the next slide.

DB2 Storage

§ **Instance** ⟷ **Database** ⟷ **Schema** ⟷ **Table** ⟷ **Row/Cell**

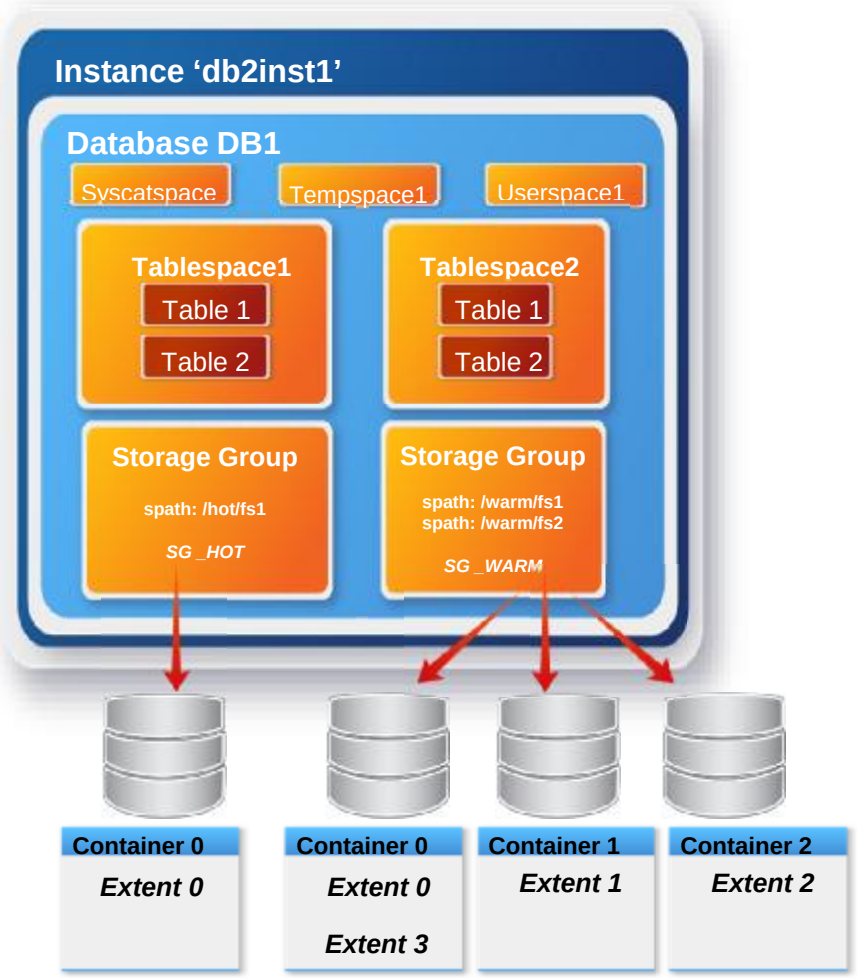
§ ***Table space***
- – Collection of containers

§ ***Container***
- – Physical storage device

§ ***Extent***:
- – Consecutive pages in a table space

§ ***Page***
- – Smallest unit of storage in DB2

§ ***Storage Group***
- – New layer of abstraction between logical (table spaces) and physical storage (containers)



PRESENTATION NOTES FOR PAGE 15

A page is the smallest unit of storage in DB2, but DB2 reads and writes extents of pages rather than single pages to optimize input and output
A series of extents encapsulate the containers

When a table space is created, containers are defined for the table space, which in turn define the physical storage of the table space. How the containers are defined, depends on the type of table space that is created. A table space can have one or more containers defined and once defined, a container can belong to ONLY one table space. Containers can include directories, files, logical devices or drive names.

An extent is a collection of consecutive pages in a table space. For performance reasons, DB2 reads and writes extents of pages rather than single pages to optimize I/O since it costs less I/O to read one extent of several pages than reading each page, writing it out and so on. An extent can ONLY contain pages for one object. For example, DB2 will not allow one page within an extent to belong to table A and another one to belong to table B.

When you create a table space with more than one container, DB2 writes data to the containers in a round-robin fashion. DB2 fills an extent in one container, then fills an extent in the next container, and so on. Data is striped across the table space containers.

DB2 stores table and index data on a page, which is the smallest unit of storage in a DB2 database. DB2 creates and manages the pages in the table space automatically, but you can control the page size for your table spaces. If you don't explicitly specify the page size when you create a table space, DB2 will use the default size of 4K. Four different page sizes are supported – 4K, 8K, 16K, and 32K.

RIDs are used to reference objects in a table space. They specify the page number and the slot on the page that a record is located in. Larger RIDs imply more pages can be referenced in table space, and potentially more rows per page, resulting in much higher table capacity.

How does DB2 help you store your data easily …

SMS (System-managed space) Table spaces
CREATE TABLESPACE space1 MANAGED BY SYSTEM USING ('c:\space1')

DMS (Database-managed space) Table spaces
CREATE TABLESPACE space2 MANAGED BY DATABASE USING (DEVICE '/dev/rdblv6' 10000)
CREATE TABLESPACE space2 MANAGED BY DATABASE USING (FILE '/myfile1' 2GB)

Automatic Storage Table spaces
CREATE TABLESPACE ts1
CREATE TABLESPACE ts2 MANAGED BY AUTOMATIC STORAGE

Listing your table spaces
LIST TABLESPACES
LIST TABLESPACES SHOW DETAIL

Starting with DB10.1 SMS permanent table spaces have been deprecated
The system managed spaces (SMS) table space type is now deprecated for permanent table spaces that are defined by the user.
Details
You can still specify the SMS type for catalog table spaces and temporary table spaces. Automatic storage continues to use SMS type for temporary table spaces. The recommended table space types for user table spaces are automatic storage or database managed spaces (DMS).

In previous releases, SMS permanent table spaces were used because they were simple to create and manage. To create a SMS table spaces, you do not have to specify an initial size but you must ensure that there is enough free disk space. The size and growth of the container files are managed at the operating system level. However, SMS table spaces do not perform as well as DMS table spaces.

With the introduction of automatic storage, management of DMS table spaces was simplified by providing a function that automatically resizes containers. IBM continues to invest and develop in automatic storage and DMS table spaces.

Agenda

- Performance Tuning Overview
- DB2 Architecture Fundamentals
- **Hardware Planning Rules of Thumb**
- Database Design & Recommendations

PRESENTATION NOTES FOR PAGE 16

CPU – the Main Independent Variable in Performance

§ Rule of Thumb for Business Intelligence (BI) environments:
- 200-300 GB of active raw data per processor core is a reasonable estimate

§ Other environments (OLTP):
- Try to gauge amount of CPU required based on one or more existing DB2 systems.

  E.g.: new system to handle 50% more users, SQL is at least as complex as on an existing system **è** reasonable to assume that 50% more CPU capacity is required

- Take a look at www.tpc.org TPC-C DB2 results, for example:
  - 8 core IBM POWER7 4.14GHz = ~1.2M TPM
  - 64 core IBM POWER6 5 GHz = ~6M TPM
  - 192 core IBM POWER7 7 3.86 GHz = ~10M TPM
  - 10 core Intel Xeon 2.4 GHz = ~3M TPM
  - 32 core Intel Xeon 2.26 GHz = ~2.3M TPM

PRESENTATION NOTES FOR PAGE 17

First start with sizing the CPU. CPU capacity is one of the main independent variables in sizing and configuring a system for performance. Because all other hardware configuration typically flows from it, it is not easy to predict how much CPU capacity is required for a given workload. Of course here is where we start to make a distinction in applications types. Business Intelligence systems characterized by fewer concurrent requests that are more resource intensive, especially in terms of data access these systems are usually sized by defining a ratio of processor power to active raw data size. An Transaction based system typically has more concurrent requests, they tend to be “standard” across like systems, accounting, MRP, ERP that is the transactions, data structures, and SQL have a particular form that allows one to find “like” systems to base the sizing on.

The TPCC transactions are actually pretty demanding OLTP type transactions, so you would likely see more throughput with your own application. That is of course a matter of scale many of the TPC configurations are designed to achieve very big and very little, that is extreme numbers via extreme allocations.

You must also consider other factors at the application level like Complex SQL, RI, triggers, LOBs, UDFs, etc. can all these things can increase CPU consumptions when being used in the database.

Storage

- Considerations:
  - I/O Throughput
    - I/Os per Second (IOPS)
    - Megabytes per Second (MBPS)
  - Storage Capacity
  - Separate dedicated (unshared) disks for logging
- Rules of Thumb:
  - 15K RPM Fibre Channel disk = ~200 IOPS @ ~10 milliseconds response time
  - Solid State Drives (SSDs) I/O service times are typically less than a millisecond, instead of up to approximately 10 ms, for typical small random reads from physical disks
- Because CPU processing speeds have increased substantially relative to spindle speeds:
  - For OLTP, ensure that there are 15 - 20 dedicated physical disks per CPU core.
  - For warehousing, ensure that there are 8 - 10 disks per CPU core
- Logging:
  - For OLTP, fast log response time is often more important than I/O service times for data, which is frequently asynchronous

PRESENTATION NOTES FOR PAGE 18

Memory

§ Decouples CPUs and Disks

§ Limited by addressable shared memory
- 32 bit ~4GBs (supported on only Windows and Linux)
- 64 bit virtually unlimited (17.2 billion gigabytes, 16.8 million terabytes, or 16 exabytes)

§ Not uncommon for some database servers to have 10's to 100's of GB of RAM

§ Rules of thumb for DB2 Systems are as follows:
- OLTP system:
  - 4GB per core for Intel/AMD (System x)
  - 8GB per core for POWER (System p)
- BLU acceleration system:
  - a minimum of 8 cores and 64 GB RAM is recommended.

PRESENTATION NOTES FOR PAGE 19

CPUs and disks effectively operate on different time scales, that is nanoseconds versus microseconds, so you need to decouple them to enable reasonable processing performance. Memory provides this decoupling. With respect to databases, the main purpose of memory is to avoid I/O. Typically, the more memory that a system has. the better it can perform. Fortunately, memory costs have dropped significantly over the last several years, and systems with tens to hundreds of gigabytes (GB) of RAM are not uncommon. In general, four to eight gigabytes per processor core should be adequate for most applications.

Agenda

- Performance Tuning Overview
- DB2 Architecture Fundamentals
- Hardware Planning Rules of Thumb
- **Database Design & Recommendations**

PRESENTATION NOTES FOR PAGE 20

Default Database Factors that Affect Performance

§ "`CREATE DATABASE <dbname>`" results in:

- Configuration Advisor runs on database only with somewhat conservative options
- Automated runstats is enabled
- Adaptive Self Tuning Memory is enabled
- Unicode database
- Automatic Storage using a single location for containers (defined by DBM CFG DFTDBPATH)
  - Use ON clause to specify multiple paths
- Database internal files and log files stored in same location as table space containers (defined by DBM CFG DFTDBPATH)
  - Use DBPATH ON clause to specify location separate from storage

```
CREATE DATABASE <dbname>
    ON <path/driver>
DBPATH ON <path/driver>
USING CODESET <codeset> TERRITORY <territory>
  PAGESIZE <pagesize>
```

In DB2 10 AUTOMATIC STORAGE parameter is deprecated and might be removed in a future release

PRESENTATION NOTES FOR PAGE 21

This slide describes the defaults of several factors that affect performance when using the most basic of database creation commands.

You will likely want to rerun the Configuration Advisor after the database is created and populated. Additionally you can specify the characteristics of the application.

If you don't need Unicode, explicitly provide a code set, unicode can introduce additional overhead when character expansion is required though this is why you might need it in the first place. Additionally you should consider the overhead associated with collating sequences, culturally correct collating sequences used in databases where multiple languages reside can cause significant overhead and should be avoided when possible.

For optimal performance, spread table space containers across all available disks using the ON clause to specify multiple paths. Ideally each path will be serviced by dedicated disk resources. This is a simple way to achieve good performance by spreading data and indexes, and what's more it is a very good recommendation for application installations looking for "out of the box performance".

Database control files location is controlled by DBPATH ON option and should always be in a separate location from storage (table space containers). These files include buffer pool control files, table space control files, storage path control files, database configuration file, history and recovery files, and log control files.

Default Database Factors that Affect Performance - 2

§ "**CREATE DATABASE <dbname>**" results in:

- Default page size of 4K for default buffer pool and all table spaces
  - May not be appropriate for your environment
- Default Table spaces:
  - **SYSCATSPACE**
  - **TEMPSPACE1**
  - **USERSPACE1** (DMS Large, NO File system caching)

    File system caching more suitable for LOB data
- **IBMDEFAULTBP** buffer pool uses automatic tuning with an initial size of 1000 pages
  - May need multiple buffer pools
  - Recommend separate bufferpools for tempspace
- All databases are created with the default storage group **IBMSTOGROUP**, unless you specify otherwise
- Currently Committed Isolation Level is enabled
- DB2DETAILDEADLOCK event monitor created
  - Drop it, performance overhead – (If new Evm. for locking is to be used)

PRESENTATION NOTES FOR PAGE 22

The default page size is 4K, which is initially used for the default buffer pool IBMDEFAULTBP and the SYSCATSPACE, TEMPSPACE1, and USERSPACE1 table spaces. In a medium to large database environment you will want to create customized tablespaces that will minimize wasted space and reduce I/O. Temporary tablespaces should be serviced by a bufferpools that are dedicated to temporary tablespaces.

Currently Committed is enabled, where the committed value of a row is always returned. Under these circumstances, locks aquired and held on rows by write operations will not cause a read transactions to go into a lock wait state.

The deadlocks event monitor has been deprecated, and the function it provided is included in the locking event monitor. In addition, the DB2DETAILDEADLOCK event monitor is also deprecated. See Deprecated lock monitoring functionality for important usage information about this event monitor

The deprecated detailed deadlock event monitor, DB2DETAILDEADLOCK, is created by default for each database and starts when the database is activated. If you use the locking event monitor to detect deadlocks, consider disabling the DB2DETAILDEADLOCK event monitor. If the DB2DETAILDEADLOCK event monitor remains active while the locking event monitor also collects deadlock information, both event monitors will be collecting data, which can significantly affect performance

To remove the DB2DETAILDEADLOCK event monitor, issue the following SQL statements:

SET EVENT MONITOR DB2DETAILDEADLOCK state 0

DROP EVENT MONITOR DB2DETAILDEADLOCK

Database Log Files

- LOGPATH or NEWLOGPATH (DB CFG) – defines location of the transaction log
  - **`update db cfg for <DBNAME> using NEWLOGPATH </path/…/>`**
    - Or use the DBPATH ON option of CREATE DATABASE
  - Never keep them in default location under the database path
  - Logs should not share disk devices with any other DB2 objects (e.g., table spaces)
    - Placed on dedicated storage with sufficient throughput capacity to ensure that a bottleneck will not be created
- LOGBUFSZ (DB CFG) – defines size of transaction logger internal buffer (in 4KB pages)
  - In general, a value of 256 (1MB) to 1000 (4MB) pages is a good range

PRESENTATION NOTES FOR PAGE 23

Some parameters do not have automatic settings, and are not set by the configuration advisor. These need to be dealt with explicitly. These parameters all have performance implications.

For instance, the default path of the transaction logs will remain in effect until the NEWLOGPATH is established and the database is restarted. Ideally in a write intensive environment transaction logs should have a dedicated

Path and underlying storage.

The logbufsz is the internal buffer for transactions logs, it is by default set very low, this should be increased to a starting value between 1 an 4 mb as an initial value.

Table Space Management – MANAGED BY

§ System Managed Space (SMS)
- Directory container, each object stored as separate in the file system
- Access to data is controlled using standard I/O functions of the OS
- Space not allocated by the OS until required
- Low maintenance and monitoring
- Useful for small temporary table spaces

§ Database Managed Space (DMS)
- Data stored in pre-allocated files or on raw device
- Access to data controlled by DB2 and can bypass operating systems I/O functions, increasing performance
- Increased maintenance and monitoring (though not much anymore)
- Supports separating of table data, LOBs, and indexes into different table spaces for better parallelism
- Supports use of LARGE table spaces (Large RIDs) for increased table capacity and better I/O

PRESENTATION NOTES FOR PAGE 24

Tables spaces can be managed by the operating system or DB2 database manager. Automatic storage will use DMS tablespaces for data and indexes and SMS for temporary tablespaces this model can be emulated when creating your own tablespaces.

DMS tablespaces, with regards to performance are pre-allocated meaning they will not incur frequent overhead related to a acquiring new extents as they grow. They allow for direct IO where the operating systems filesystem cache can be bypassed, no double buffering. They also allow you to separate data, and indexes, and in the case of LOB data allows you to specify, for the LOB tablespaces the filesystem cache can used, these objects will not be brought into db2 bufferpools and require IO whenever accessed. Note

SMS permanent table spaces have been deprecated

The system managed spaces (SMS) table space type is now deprecated for permanent table spaces that are defined by the user.

Details

You can still specify the SMS type for catalog table spaces and temporary table spaces. Automatic storage continues to use SMS type for temporary table spaces. The recommended table space types for user table spaces are automatic storage or database managed spaces (DMS).

In previous releases, SMS permanent table spaces were used because they were simple to create and manage. To create a SMS table spaces, you do not have to specify an initial size but you must ensure that there is enough free disk space. The size and growth of the container files are managed at the operating system level. However, SMS table spaces do not perform as well as DMS table spaces.

Automatic Storage

- Storage is completely managed by DB2 utilizing best practices for optimally managing table spaces
- Default is to use Automatic Storage
  - Also, by default, is to use a single path for the storage
- Use the ON clause of the CREATE DATABASE command to specify multiple paths for DB2 to store table space data on
  - Paths can be added or removed later
- DB2 creates and extends containers as needed up the limits imposed by the storage paths

PRESENTATION NOTES FOR PAGE 25

As mentioned before automatic storage can be set up to give very good performance by providing multiple independent storage paths for the database to work with. It should not go without mention that they are very easy to manage since they can automatically be resized. You may be able to achieve better performance by explicitly creating your tablespaces as DMS, under certain circumstances certain large critical tables may perform better in storage defined outside of the scope of the databases storage paths. Any gains in performance must be measured against the added maintenance of these tablespaces and of course, the cost of the “extra storage”.

Note

SMS permanent table spaces have been deprecated

The system managed spaces (SMS) table space type is now deprecated for permanent table spaces that are defined by the user.

Details

You can still specify the SMS type for catalog table spaces and temporary table spaces. Automatic storage continues to use SMS type for temporary table spaces. The recommended table space types for user table spaces are automatic storage or database managed spaces (DMS).

In previous releases, SMS permanent table spaces were used because they were simple to create and manage. To create a SMS table spaces, you do not have to specify an initial size but you must ensure that there is enough free disk space. The size and growth of the container files are managed at the operating system level. However, SMS table spaces do not perform as well as DMS table spaces.

Tablespace Design Considerations

§ User Table Spaces :
- Use automatic storage tablespace
- Separate table data, indexes, and LOBs
  - Separating data helps decrease I/O contention and increase prefetching performance

§ Temporary Table Spaces:
- Use automatic storage
- Other options: use SMS for small ones and DMS for large
  - DB2_SMS_TRUNC_TMPTABLE_THRESH can be set to a number of extents, avoids OS overhead for file creation

PRESENTATION NOTES FOR PAGE 26

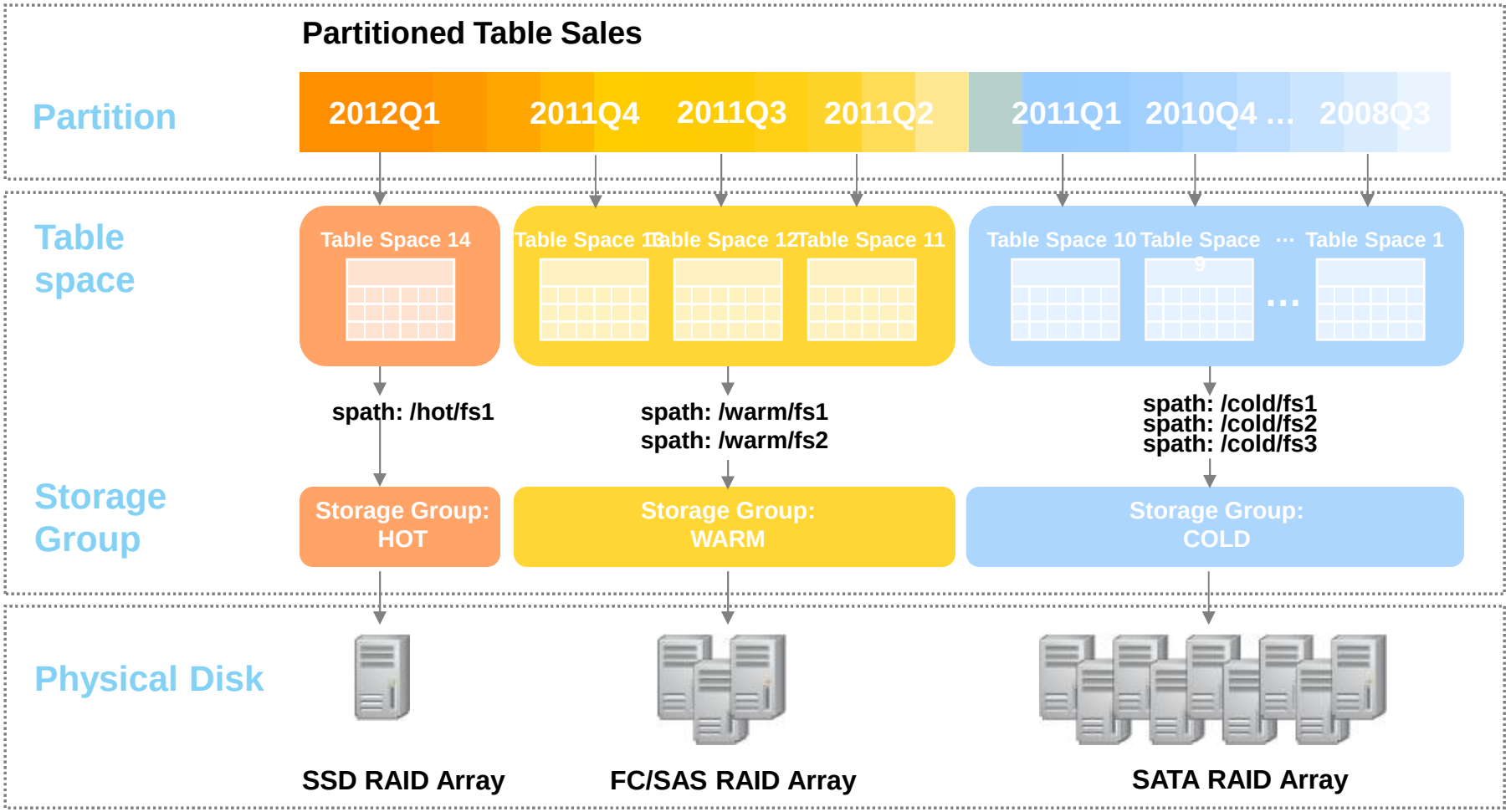
If you go no automatic storage route you will want to use DMS tablespaces, this will afford you the capability of splitting indexes and LOB data from regular data, which not only allow for decreasing I/O but you can also customize Bufferpools assigning different bufferpools to different tablespaces.

Temporary Tablespaces can be created as either SMS or DMS, what will work best may depend on the size and variability of the temporary tables that your system creates. The DB2_SMS_TRUNC_TMPTABLE_THRESH can be set to a low number, this will prevent the complete deletion of the underlying files that are created for temporary tables and the overhead associated with creating new files by the OS when more temporary tables are required.

Direct I/O eliminates the “double buffering” of data and is the default for tablespaces, however if you are creating separate tablespaces for your LOB data, you should specify that the table space USE file system caching since this data is not read into DB2 bufferpools.

Multi-Temperature Data Management: Storage Groups

NEW IN DB2 10



PRESENTATION NOTES FOR PAGE 27

When defining storage groups, ensure that you group the storage paths according to their quality of service characteristics. The common quality of service characteristics for data follow an aging pattern where the most recent data is frequently accessed and requires the fastest access time (hot data) while older data is less frequently accessed and can tolerate higher access time (warm data or cold data). The priority of the data is based on:

Frequency of access

Acceptable access time

Volatility of the data

Application requirements

Typically, the priority of data is inversely proportional to the volume, where there is significantly more cold and warm data and only a small portion of data is hot. You can use the DB2® Work Load Manager (WLM) to define rules about how activities are treated based on a tag that can be assigned to accessed data through the definition of a table space or a storage group.

Table Spaces and page size

- Large DMS table space is the default
- Supports Large RIDS, which means it can fit about 2000 rows on a page (DMS only)
- Choose page size for table space based on the average row size for tables to be placed in the table space
  - Group tables with smaller rows in smaller page size table spaces
  - Group tables with larger rows in larger page size table spaces
- Example: Table with row length of 10 bytes only is placed in a 32k tablespace. It will only use up to about 20k per page wasting 12k of space per page
  - Furthermore data compression may not be effective
- There should be a system temporary table space for each page size used, to allow for more efficient REORGs

PRESENTATION NOTES FOR PAGE 28

Page size is both critical to storage efficiency and of course I/O efficiency. Since DB2 reads data in at the page level, the more rows per page, the more effective your reads can be, especially when dealing with clustered data. DB2 introduced Large RIDs, A RID is a row ID that is used to address where the data resides in a tablespace, by increasing the RID size from 3:1 (page:slot) to (4:2) the number of pages and the number of rows per page can be increased dramatically. If you are licensed for the Storage Optimization Feature Pack you have the option of compressing your rows, decreasing your average row size allowing you to take even more advantage of the large RID sizes that have been introduced.

Extended row size new in DB2 Version 10.5

- Migrate tables that have row sizes that exceed 32K to DB2 Version 10.5.
- Improve the performance of applications where most of data rows can fit on a smaller page, but the table definition requires a bigger page size.
- Create tables with more VARCHAR or VARGRAPHIC columns.
- Existing tables can be altered
- **Working with tables with extended row size**
- Any row-organized tables support extended row size except for range clustered tables (RCT).
- The extended_row_sz database configuration parameter must be set to ENABLE.
- The table definition must contain at least one varying length string column (VARCHAR or VARGRAPHIC).
- The row size of the table cannot exceed 1048319 bytes (SQLSTATE 54010).
- Tables with extended row size support can be identified by checking the EXTENDED_ROW_SIZE column of the SYSCAT.TABLES catalog view.

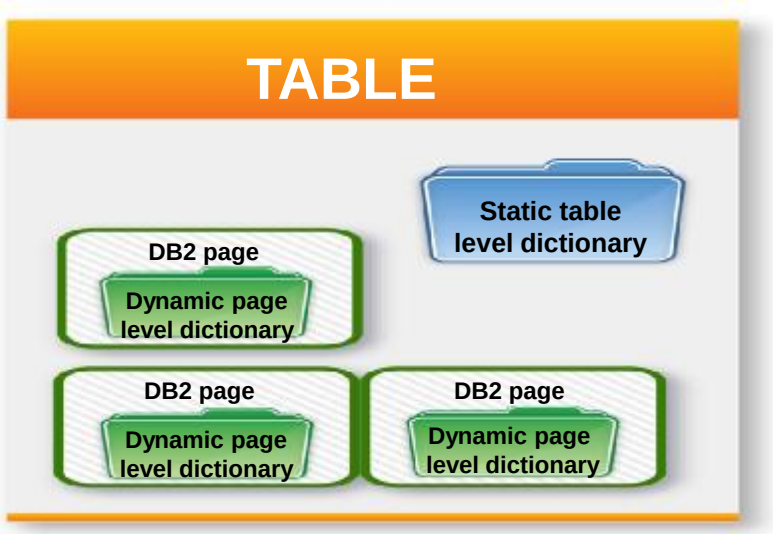
PRESENTATION NOTES FOR PAGE 29

Adaptive Compression

NEW IN DB2 10

§ Adaptive Compression is an enhancement to the Classic Row Compression
  – Compress rows by using a combination of two types of dictionaries
    • Global static table level dictionary
    • Local page level dictionaries

§ Benefits
  – Page level dictionaries adapt to data skew over a period of time
  – No REORGs required to maintain high compression ratio
  – Less disk space for data and logs
  – Reduced I/O. Fewer pages to process



- **Better compression ratios than Classic Row Compression**
- **Over time reduces need of a REORG as page-level dictionaries adapt to data skew over time (i.e. maintain compression rates)**

PRESENTATION NOTES FOR PAGE 30

Adaptive Compression is included only in the Storage Optimization feature, and as such is only available for:

- Enterprise Server Edition and
- Advanced Enterprise Server Edition

Significant enhancements to DB2's industry leading compression technologies come in the new Adaptive Compression which can further reduce your storage needs. The enhancements deliver efficient compression of high amounts of new and changing data. The improved compression ratio further reduces storage needs and allows for more data in memory therefore increasing performance. The new approach used in Adaptive Compression also reduces the need for table reorganization. As a consequence, overall maintenance of compressed data is reduced, providing additional cost savings.

- New archive log compression reduces log archive storage
- Simply turn it on and DB2 does the compression for you
- Large SAP customer generates 60GB of log per day and they keep 8 weeks of archives. Storing 3.3TB of archived log files. Compression of 4x results in storage of only 825GB for 8 weeks.

Adaptive Compression: Performance Considerations

NEW IN DB2 10

§ Adaptive Compression always gives better compression ratios

– You can also compress index objects, especially useful for large OLTP and data warehouse environments

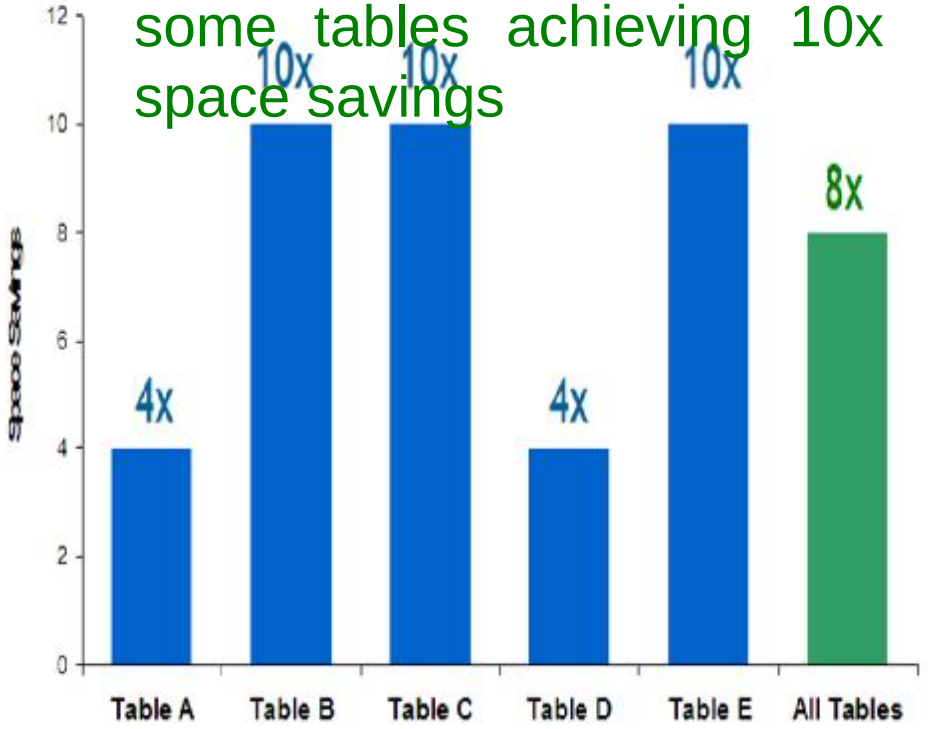
– Index compression can significantly improve performance

– Adaptively apply both table-level compression and page-level compression

– Table re-orgs not required to maintain high compression

– Compress archive logs

Has provided 7x or greater overall space savings for more than one client, with some tables achieving 10x space savings



PRESENTATION NOTES FOR PAGE 31

Real customer data from beta client shows that DB2 10 Adaptive Compression saves big. This graph represents the largest 5 tables in their database. With DB2 9.7 they were getting between 3X and 6X compression. Using the new adaptive compression in DB2 10 they are now getting between 4X and 10X compression. Note that it was actually their largest table that had the highest savings. For their largest table with 1.5 billion rows they are seeing 10X compression with DB2 10.

When we look at their overall table savings, they were seeing 5X compression overall with DB2 9.7 (the last bar on the right). Now with DB2 10 that has increased to 7X compression overall.

Choosing Extent sizes

§ Specified at table space creation time and cannot be changed afterwards

§ Choosing Extent size:

- Default of 32 Pages is typically the best choice in most cases
- BI/OLAP type workloads typically benefit from larger extent sizes
  - Improves prefetching performance
- OLTP workloads typically benefit from smaller extent sizes
  - Less wasted space in buffer pools and faster access

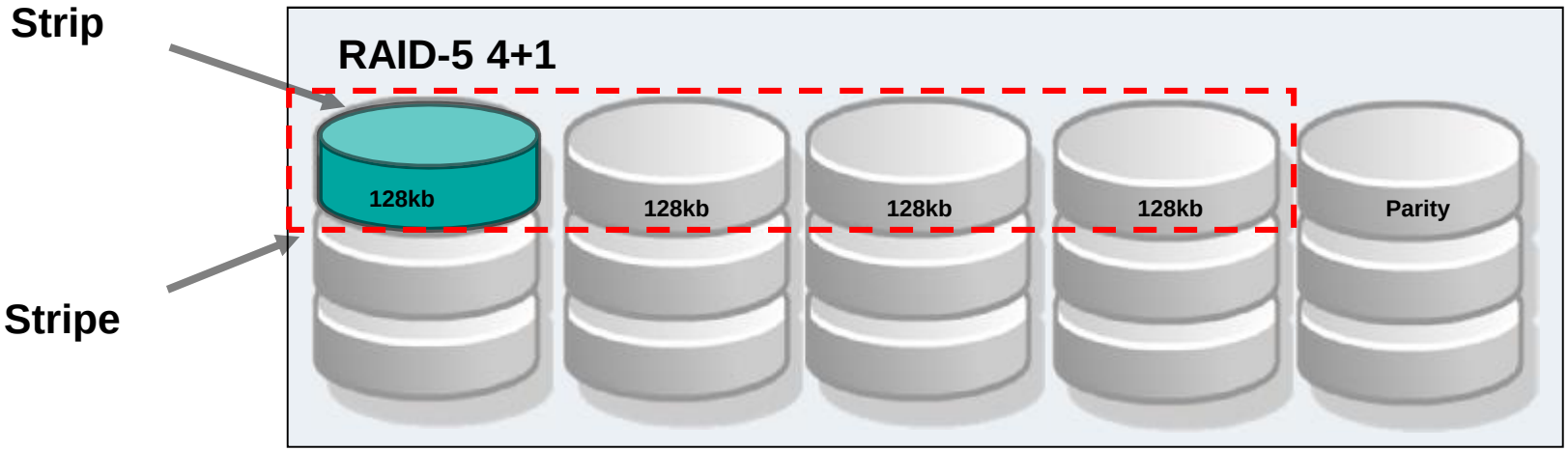
PRESENTATION NOTES FOR PAGE 32

Large extents better sequential prefetching, all pages in extent in one place.

Small extents works better for OLTP, genearlly random access, update one row in a table, have to read at least one extent, large extent would mean reading a lot of pages that are not needed.

Set EXTENTSIZE to RAID stripe size – Best Practices

§ The EXTENTSIZE for a table space should be set to the number of pages that are included in an entire RAID stripe



Stripe = 4 (# of data disk) x 128 (Strip Size KB) = 512 KB

$$\text{EXTENTSIZE} = \frac{\text{Stripe (KB)}}{\text{Page Size (KB)}}$$

http://www.ibm.com/developerworks/db2/bestpractices/

PRESENTATION NOTES FOR PAGE 33

The amount of contiguous data on each spindle in a RAID array is known as a strip, and the amount of data across the array that comprises all the strips in a single array is called a stripe.

Set EXTENTSIZE for table spaces should be set to a number of pages that includes the entire RAID stripe.

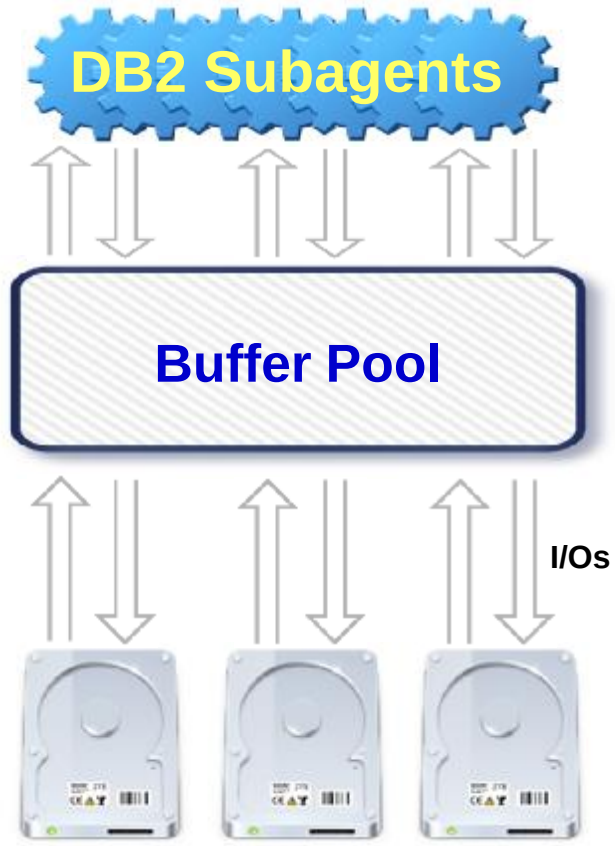
Typical RAID strip sizes are 32kb, 64kb, 128kb, and so forth.

A RAID-5 4+1P array would have a stripe size equal to 4 times the strip size, as there are 4 non-parity spindles in that array.

Base on the same example as in the previous slide, a system with a 128kb stip size that uses RAID-5 4+1, the stripe size will be 512kb (4 data disk x 128kb strip size). If the page size used is 8kb, then an EXTENSIZE of 64 (512kb/8kb = 64) is appropriate.

Buffer Pools

- Area of main memory used to cache data and indexes
- Purpose is to decrease time required to access data and reduce I/O contention
- Each database must have at least one buffer pool per page size
- Automatic buffer pool tuning with Self-Tuning Memory Manager (STMM)



PRESENTATION NOTES FOR PAGE 34

32-bit is limited by virtual addressable shared memory, while there is really no such limit on 64 bit.
Each database must have at least one buffer pool
By default IBMDEFAULTBP is used
SYSCAT.BUFFERPOOLS catalog view accesses the information for the buffer pools defined in the database

Every table space is associated to a specific buffer pool of the same page size
Match buffer pool size with purpose of table to increase hit ratio

Self-Tuning Memory Manager (STMM) optimizes BP utilization

Buffer Pool Creation

§ **`CREATE BUFFERPOOL <bpname>`**

– If STMM is active, creates a buffer pool with an initial size of 1000 pages with automatic tuning

– If pagesize not is specified, the default value is provided by the ***pagesize*** database configuration parameter, which is set when the database is created

- NPAGES value of -2 in **SYSCAT.BUFFERPOOLS** view indicates that the buffer pool is enabled for automatic tuning

– Configuration Advisor can be run to start with better initial values for all buffer pools in the database based on available memory (you can simply use the recommendations instead of automatically applying)

§ **`CREATE BUFFERPOOL <bpname> SIZE <size>`**

– Explicitly providing a size and omitting automatic clause creates a buffer pool that must be manually tuned

– Non-Automatic Bufferpools can be ALTERED to become AUTOMATIC

PRESENTATION NOTES FOR PAGE 35

The CREATE BUFFERPOOL statement creates a new buffer pool to be used by the database manager.
Once a buffer pool has been created, it can be assigned to a table space using the ALTER TABLESPACE command.
When creating a table space, a buffer pool can be assigned to the table space.

The authorization ID of the statement must have SYSCTRL or SYSADM authority.

Buffer Pool Design

- For each page size used in the database, a buffer pool using the same page size must exist
- OLTP – Buffer pools typically consume 75% of available memory
  - Multiple buffer pools can increase performance by caching different amounts of data depending on the purpose of the associated table space(s)
- BI/OLAP – Buffer pools typically consume 50% of available memory (50% to SORTHEAP)
  - In most cases only one buffer pool is recommended (assuming one page size) for regular & user temporary tablespaces and one for system temporary tablespaces

If DB2 is unable to obtain memory for the regular buffer pools, it will attempt to start up with small system buffer pools of 16 pages, one for each page size (4K, 8K, 16K and 32K)

Name	PageSz	PA-NumPgs
IBMSYSTEMBP4K	4096	16
IBMSYSTEMBP8K	8192	16
IBMSYSTEMBP16K	16384	16
IBMSYSTEMBP32K	32768	16

PRESENTATION NOTES FOR PAGE 36

Never allocate more memory than you have available or you will incur costly OS memory paging. Generally speaking, it may be pretty hard to know how much memory to initially allocate to each buffer pool without monitoring, which is why STMM is so beneficial.

For OLTP type workloads, 75% of available memory being allocated to the buffer pool(s) is a very good starting point. Then let STMM do its job (or at least for a couple of days and then disable it once they are performing well).

For OLAP/DSS, the rule of thumb is to give 50% of your available memory to a single buffer pool (given that only one page size is being used) and the other 50% to the SORTHEAP.

Table Design

§ CREATE/ALTER TABLE options

- COMPRESSION can save up to 70% disk space and provide about 20% performance boost in I/O bound workloads
  - Fewer I/O operations needed to retrieve same amount of data
  - Accessing data from disk is the slowest database operation
  - Deep Compression w/ Storage Optimization Feature Pack
- APPEND ON for tables which have heavy inserts to avoid searching for free space and instead simply append the row to the end of the table data
  - Information about free space on pages will not be kept
  - Alternatively, DB2MAXFSCRSEARCH can be used to improve insert performance
- PCTFREE to maintain free space to help with future INSERTs, LOADs and REORGs
  - default is 10; try 20-35 for tables with clustered indexes and a heavy insert volume
  - If using with APPEND ON, set PCTFREE to 0
- PARTITION BY RANGE to support fast roll-in and roll-out of data

PRESENTATION NOTES FOR PAGE 37

Columns of VARCHAR(30) or less typically use more space than the equivalent CHAR column. Wasting space can even affect query times if the volume is significant.

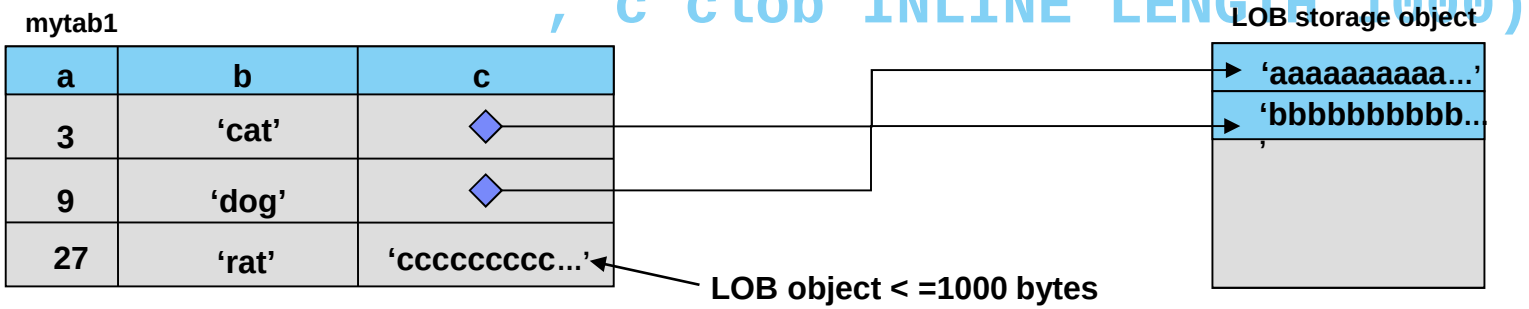
System default values are the default values used for the data types when no specific values are specified. This can also help improve query time when volume is significant. Compressed columns will incur a small overhead if a value is inserted or updated.

If you rely on the table being in a special order and cannot afford to perform REORGs, avoid using APPEND ON.

VOLATILE will only use an index if an appropriate one exists, where the index contains all the columns referenced or when the index is able to apply a predicate in the index scan.

DB2 – XML and LOB In-lining

- XML and LOBs are stored outside the base table in a separate storage object or if sufficiently sized, can be stored in the formatted rows of the base table
  - Depending on page size, the maximum length a LOB can be, to qualify for in-lining, is 32 669 bytes
- Descriptors stored in the base table rows are used to keep track of the associated XML/LOB data in the storage object
- Example:
  -
```
CREATE TABLE mytab1 ( a int
                    , b char(5)
                    , c clob INLINE LENGTH 1000)
```



PRESENTATION NOTES FOR PAGE 38

For the purposes of this and the next slide, LOB and XML can be used interchangeably. For a table with LOB columns, the LOBs themselves actually reside in a separate LOB storage object. Here is a fictitious example – a table with three columns, a, b, and c, one of which is a LOB column. In the case of rows 1 and 2, the integer and character data are stored in the base table, while the LOBs are stored elsewhere. In place of the actual LOB data in the base table are LOB descriptors, which refer to their respective LOBs in the LOB storage object.

Now, instead of storing LOBs in the LOB object exclusively, they can be stored in the base table – that is. they can be inlined. In order to qualify for LOB inlining, the LOBs must be smaller than a certain size. In this case, for the third row the LOB happens to meet the inline length criteria specified, 1000 bytes, and is stored inline.

The absolute max size a LOB can be to qualify for inlining is roughly 32KBs, though it really depends on the page size specified; 32KBs corresponds to a 32K page size. In general, the max LOB size for inlining with respect to page size is the max row size for that page size, minus a few bytes for overhead. When creating a table with LOB columns, by default, an inline length is set. The default inline length is dependant on the maximum length of the LOB column. The smallest LOB inline length allowable is 68 bytes, which is what one would get with a LOB column of size 1024 bytes (1KB).

Why Use XML and LOB In-lining?

§ If a table possesses XML or LOB data that can be in-lined, there are considerable benefits with respect to performance and storage use

Performance

- In-lined XML or LOB can be buffered in the bufferpool
- LOB data can be compressed in storage area or when in-lined, saving on IO costs
- XML data can be compressed in XDA or when in-lined

Storage

- Overall Storage allocated to the storage object is reduced by in-lining XML/LOB data (though base table storage increases)
- In-lining small XML/LOBs can result in a noticeable decrease in net total storage since the decrease in storage size is greater than the increase in base table storage size

PRESENTATION NOTES FOR PAGE 39

There are considerable benefits with respect to performance and storage. Unlike base table data, LOB data is not buffered, which translates to going to disk each time the LOB object is accessed. An Insert of LOB data, requires a write to disk, while a Query for LOB data, requires a read from disk.

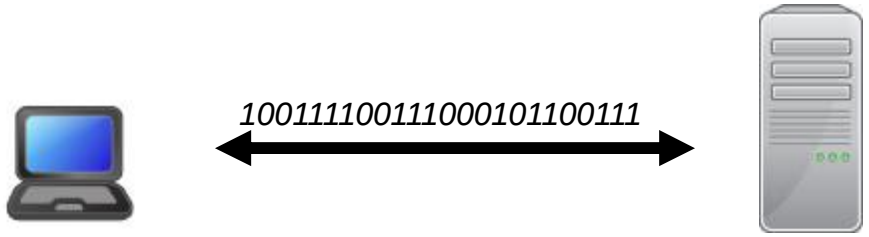
For inlined LOBs, the LOBs essentially become a part of the base table data and therefore become subject to buffering, reducing the I/O costs. Storage-wise, the amount of space allocated for the LOB object decreases, though the storage for the base table increases by a similar amount. However, depending on the size of your LOBs, the amount of storage decrease in the LOB object may be greater than the increase in base table storage size. This would result in a net total storage savings and it likely to happen for a table with small LOBs (the likelihood of this behaviour diminishes with larger and larger LOBs).

For inlined LOBs, the data essentially becomes base table data and can therefore be compressed for storage savings (potentially – depends on the nature of the LOB data).

pureXML enhancements

NEW IN DB2 10

§ Continued **improvement of pureXML® performance** with the addition of support for binary XML data transmission format and expansion of XML indexing support



§ You can now create indexes of type DECIMAL and INTEGER over XML data

§ DB2 server has been optimized to improve performance for certain commonly used queries, such as those that use the XMLTABLE function

§ Use functional XML indexes using the fn:upper-case and fn:exists functions

PRESENTATION NOTES FOR PAGE 40

pureXML

- The new binary XML format provides a faster way to transmit and receive XML data between certain Java pureXML® applications and a DB2® 10 server. For these Java applications, unnecessary XML parsing costs are eliminated, therefore improving performance.

- Functional XML indexes

- DOUBLE used to be the only numeric type for XML indexes. You can now create indexes of type DECIMAL and INTEGER over XML data, which can potentially provide faster query response times

you can create functional XML indexes using the fn:upper-case and fn:exists functions. Indexes created using fn:upper-case can speed up case-insensitive searches of XML

data.

Indexes created using fn:exists can speed up queries that search for specific elements or for the lack of specific elements

New binary XML format improves performance for certain Java clients

The new binary XML format provides a faster way to transmit and receive XML data between certain Java pureXML® applications and a DB2® 10 server. For these Java applications, unnecessary XML parsing costs are eliminated, therefore improving performance.

Binary XML data refers to data that is in the Extensible Dynamic Binary XML DB2 Binary XML Format, also known as XDBX format.

In DB2® 10, the DB2 server has been optimized to improve performance for certain commonly used queries, such as those that use the XMLTABLE function.

Examples of queries that might show faster response are as follows:

Queries that use the XMLTABLE function. For example:SELECT T.* FROM TEST, XMLTABLE('$doc/a/b' passing TEST.XMLCOL as "doc" columns c varchar(10) path 'c1/c2/c' d varchar(10) path 'd1/d2/d' e varchar(10) path 'e1/e2/e') AS T;

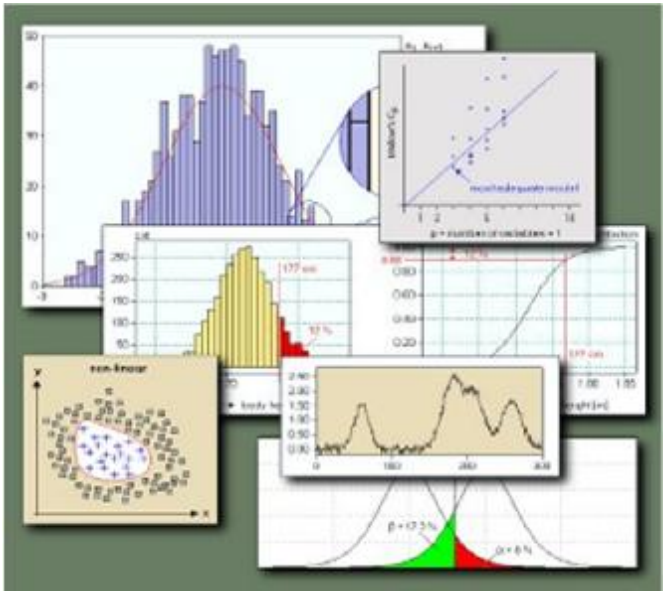
Non-linear XQuery queries (with multiple paths, or branches). For example:xquery for $a in db2-fn:xmlcolumn('XTAB.DOC')/a for $b in $a/b for $c in $a/c return <res>{$b,$c}</res>

Queries with early-out join predicates. An early-out join is a join in which at most one row from the inner table must match a row in the outer table. For example, the following XMLTABLE query has an early-out join on a row generator:SELECT stat, gen FROM custacc, XMLTABLE('$CADOC/Customer [DateOfBirth >= xs:date("1910-01-01") and BankingInfo/PremiumCustomer = "No"] ' COLUMNS GEN VARCHAR(20) PATH 'Gender', Nationality VARCHAR(20) PATH 'Customer/Nationality, STAT VARCHAR(20) PATH 'BankingInfo/CustomerStatus');

Index Management Re-defined

NEW IN DB2 10

- Jump Scan
  - Need fewer indexes
- Smart Index Pre-fetching
  - Faster index access & fewer index reorgs
- Smart Data Pre-fetching
  - Faster index scans & fewer index reorgs
- Predicate Evaluation Avoidance
  - Faster index scans
- Higher performance
  - Faster index performance
- Lower costs
  - Fewer indexes to maintain
  - Dramatic reduction in index reorgs



PRESENTATION NOTES FOR PAGE 41

Jump Scan – skip sequential Improve performance of indexes with gaps in column definitions
Reduced number of indexes
Smart Index Pre-fetching Improve performance of index scans for poorly organized indexes
Reduced need for index reorgs

Smart Data Pre-fetching Improve performance of index scan for + data fetch for poorly clustered tables
Reduced need for table reorgs
Predicate evaluation avoidance for duplicate keys Improve performance of index scans with duplicate keys

Index Design – Best Practices

- Avoid over indexing
- It is sometimes possible to have one larger (composite) index that will cover several queries
- Columns with high cardinality are good candidates for indexing
- Avoid adding an index which is similar to a pre-existing index
- Use INCLUDE clause to add columns referenced in query to index
  - Used with unique indexes only and additional columns are not used in enforcing uniqueness
- Remove unused or no longer used indexes.
  - Check the **SYSCAT.INDEXES.LASTUSED** column for possible candidate

PRESENTATION NOTES FOR PAGE 42

Thanks to the Design Advisor, the burden of designing indexes has been reduced significantly. Nonetheless, the following are some index related issues that you should be aware of.

Avoid over indexing! Excessive number of indexes can slow down update operations and consume extra space

It is sometimes possible to have one larger (composite) index that will cover several queries

For composite indexes, place the column which is referenced most in queries first in the definition

Avoid adding an index which is similar to a pre-existing index, as it creates more work for the optimizer when generating an access plan and will slow down update operations; instead alter the pre-existing index to contain additional columns

Clustering Indexes

- The Design Advisor can be used to recommend clustered indexes
- Clustering indexes can be created to order the rows in the table in the same physical order of the desired result set
- They are created using the CLUSTER option of the CREATE INDEX statement
- For best performance, create the index over small data types
- Set the PCTFREE CREATE/ALTER TABLE option
  - 15-35% suggested

PRESENTATION NOTES FOR PAGE 43

Clustered indexes allow for a more linear access pattern to data pages and more effective prefetching, and help avoid sorts. This means longer inserts, but quicker selects. Consider increasing the free space on data and index pages when using clustered indexes to about 15-35 (instead of the default 10 for PCTFREE) for high volumes of inserts. For tables which are heavily inserted into, consider using a single dimension MDC table (perhaps using a generated column like idcolumn/1000 or INT(date)/100). This will cause the data to be block indexed (on the dimension) rather than on the row – the resulting index is smaller and log contention is significantly reduced during insert.

Clustering indexes can be created to order the rows in the table in the same physical order of the desired result set

One clustering index per table at any one given time

For best performance, create the index over small data types (like integer and char(10)), unique columns, and columns most often used in range searches.

Set the PCTFREE CREATE/ALTER TABLE option somewhere between 15-35 for tables with clustered indexes to ensure that the clustered indexes do not become too fragmented

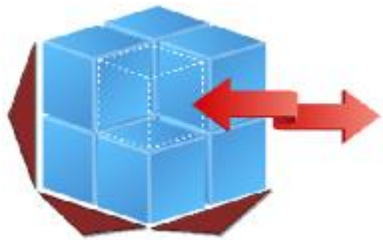
DB2 V10.5 Index improvements

- **Expression based indexes improve query performance**
- You can create an index that includes expressions.
- Best suited when you want an efficient evaluation of queries that involve a column expression.
- Values are transformed by the expressions that you specify.
- For indexes created with the UNIQUE option, the uniqueness is enforced against the values that are stored in the index. The uniqueness is not enforced against the original column values.
- **Clause EXCLUDE NULL KEYS from Index**
- Resolves issues around sparsity of key values
- Reduces size of index object

PRESENTATION NOTES FOR PAGE 44

Multidimensional Clustering (MDC)

- The Design Advisor can be used to recommend conversion to MDC tables
- Provides for flexible, continuous, and automatic clustering of data along multiple dimensions
- Physically clusters the table's data along multiple dimensions simultaneously, which is similar to having multiple, independent clustered indexes on the table
- Typically used to help speed up the performance of complex queries on large tables in BI/OLAP environments
- Best candidates for an MDC:
  - Non-unique columns
  - For best performance, you will want at least enough rows to fill a full block of each cell



PRESENTATION NOTES FOR PAGE 45

MDC provides for flexible, continuous, and automatic clustering of data along multiple dimensions. It improves query performance and reduces the need for REORG and index maintenance during insert, update, and delete.

Multi-Dimensional Clustering will physically cluster the table's data along multiple dimensions simultaneously, which is similar to having multiple, independent clustered indexes on the table. MDCs are typically used to help speed up the performance of complex queries on large tables. There is no need to use REORG to recluster the index since MDCs maintain clustering in each dimension automatically and dynamically.

The best candidates for an MDC are those queries which have range, equality, and join predicates that access many rows. Never use a unique column as a dimension since that can cause a table to be unnecessarily large. Avoid using too many dimensions if there are not a significant number of rows with each combination of dimension values (i.e., cell). For best performance, you will want at least enough rows to fill a full block of each cell, which is equal to the extent size of the table space that the table resides in.

Best candidates for an MDC:

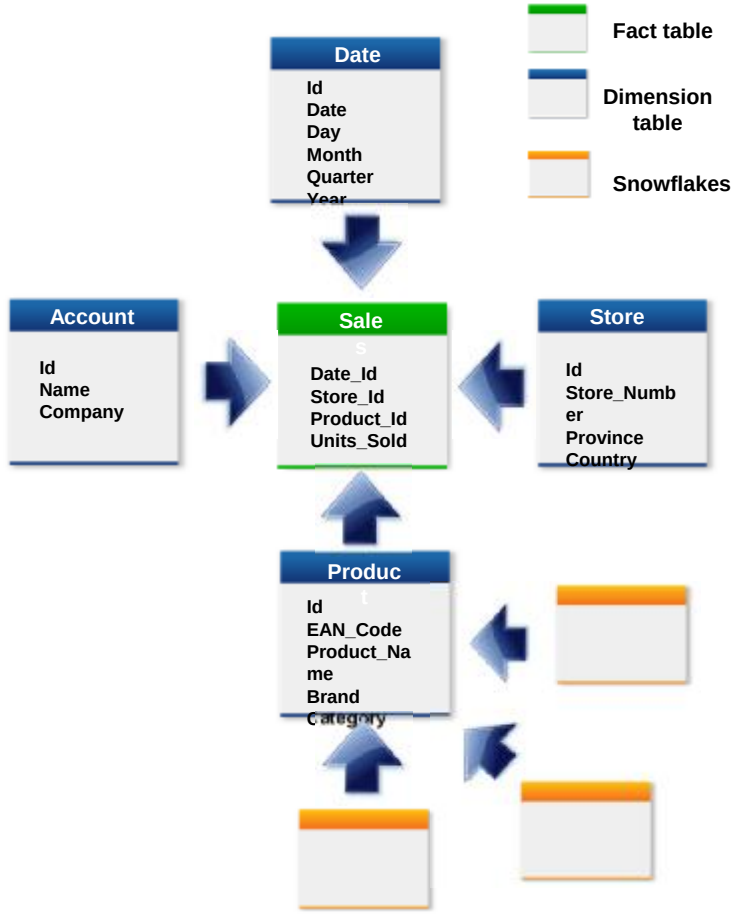
Non-unique columns, as the dimensions would cause the table to be unnecessarily large

For best performance, you will want at least enough rows to fill a full block of each cell, which is equal to the extent size of the table space that the table resides in.

Star Schema Enhancements in DB2 10

NEW IN DB2 10

- New star schema detection method
- New zigzag join method
  - Works seamlessly for range partitioned tables and serial, SMP and DPF, and pureScale
  - Consistent query performance in warehousing environments
- Supports multiple fact table query
- Exploits indexes even when there is a *gap in probing key*
- Optim Query Tuner helps to find out the multi-columns indexes to enable the zigzag join



PRESENTATION NOTES FOR PAGE 46

In DB2 10, an improved star schema detection algorithm allows the query optimizer to detect queries based on unique attributes such as primary keys, unique indexes, or unique constraints.

In addition, to improve performance of queries using the star schema in data warehouse and data mart environments, the new zigzag join method can be used to join one or more fact tables with two or more dimension tables. This new method works seamlessly in partitioned database environments, symmetric multiprocessor (SMP) environments, with range partitioned tables and MDC, and also in DB2 pureScale environments.

The new star schema also supports multiple fact table query, where the query optimizer will split the query into a query with multiple starts in it.

Indexes can be exploited even if there is a gap in the composite indexes, due to a missing join predicate or a missing dimension in the query. In this case, the query optimizer recognizes the star shape of a query and then select an appropriate access plan with a zigzag join.

OTHERS

The schema works seamlessly for range partitioned tables and in serial, SMP and DPF environments.

Can use MDC block indexes on the fact table for enabling zigzag join.

In order to help you to find out the multi-column indexes to enable the zigzag join, you can look at the explain format diagnostics and index advisor in Optim Query Tuner.

Insert Time Clustered Tables

- Insert time clustering (ITC) tables provide an effective way of maintaining data clustering and easier management of space utilization
- ITC tables have similar characteristics to MDC tables
  - For example, these table types use block based allocation and block indexes.
  - ITC and MDC tables differ in how data is clustered
  - ITC tables cluster data by using a virtual column which clusters rows together that are inserted at a similar time. Clustering dimensions on MDC tables are specified by the creator
- ITC tables are created with the CREATE TABLE command by specifying the ORGANIZE BY INSERT TIME clause
- A convenient, online way to convert existing tables to ITC tables is the ADMIN_MOVE_TABLE procedure
  - Another method to convert existing tables to ITC tables is export/import or a load from table. Existing tables cannot be altered to become ITC tables

PRESENTATION NOTES FOR PAGE 47

Materialized Query Tables (MQTs)

- The Design Advisor can be used to recommend MQTs
- MQTs are summary tables used to improve queries which use the GROUP BY, GROUPING, RANK, or ROLLUP OLAP functions
- Transparent to the user and DB2 chooses when to use them for optimization purposes
- Used by DB2 to internally maintain summarized results of the required grouping, which allows a user to access a maintained grouping instead of reading through what could be multiple GBs of data to find the answer
- Use “refresh deferred”; otherwise, insert, update, delete operations could take longer

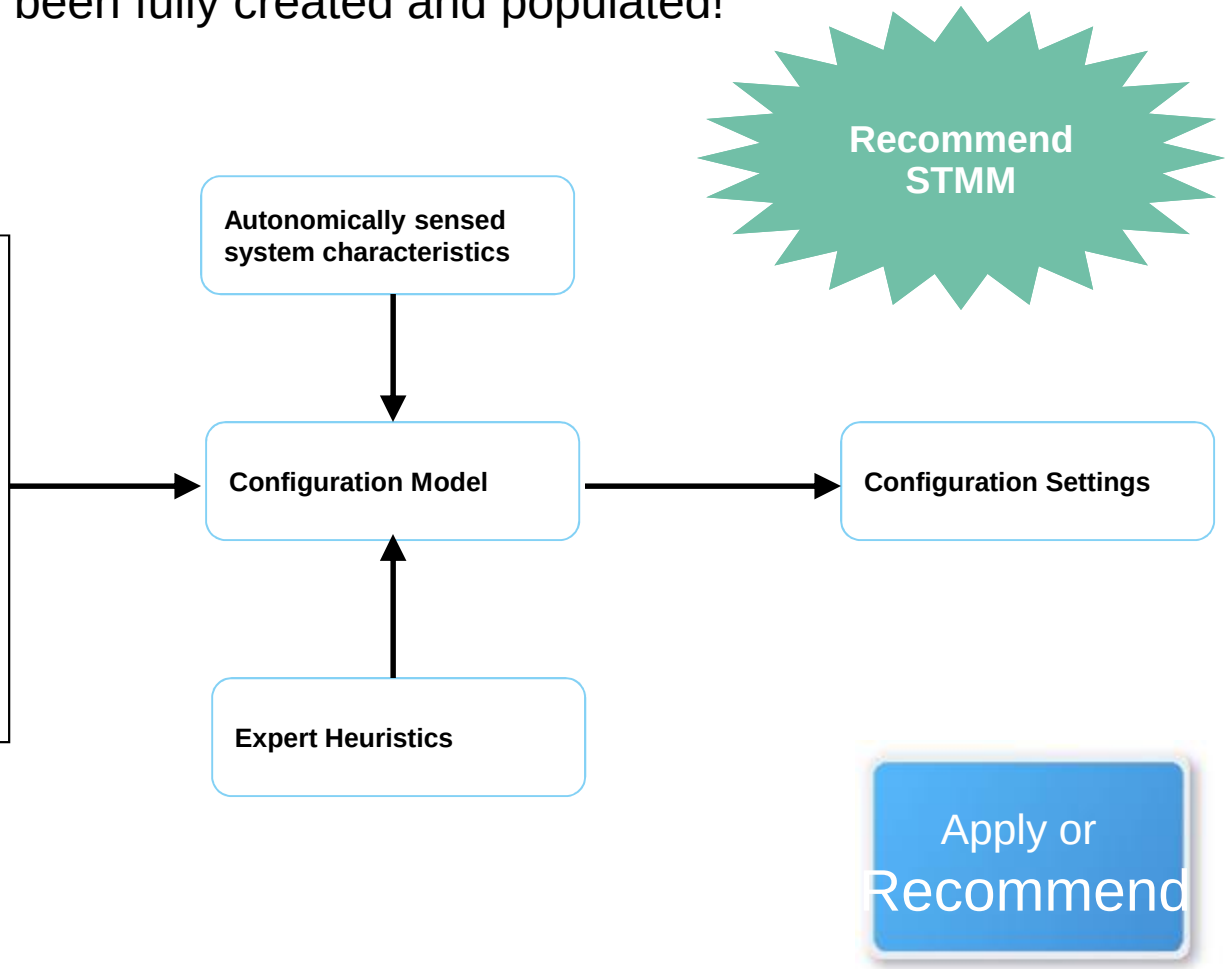
PRESENTATION NOTES FOR PAGE 48

Can also be replicated across partitions to help performance of collocated joins by avoiding the broadcasting of this information between partitions.

AUTOCONFIGURE Example

Rerun after database has been fully created and populated!

```
db2 autoconfigure using
  mem_percent 60
  workload_type mixed
  num_stmts 20
  tpm 500
  admin_priority both
  is_populated yes
  num_local_apps 0
  num_remote_apps 20
  isolation rr
  bp_resizeable yes
 apply none
```



PRESENTATION NOTES FOR PAGE 49

This command makes configuration recommendations for the currently connected database and assumes that the database is the only active database on the instance. If you have not enabled the self tuning memory manager and you have more than one active database on the instance, specify a mem_percent value that reflects the database memory distribution. For example, if you have two active databases on the instance that should use 80% of the instance memory and should share the resources equally, specify 40% (80% divided by 2 databases) as the mem_percent value.

If you have multiple instances on the same computer and the self tuning memory manager is not enabled, you should set a fixed value for instance_memory on each instance or specify a mem_percent value that reflects the database memory distribution. For example, if all active databases should use 80% of the computer memory and there are 4 instances each with one database, specify 20% (80% divided by 4 databases) as the mem_percent value.

Running the AUTOCONFIGURE command on a database will recommend enablement of the Self Tuning Memory Manager provided adequate memory. (not sure what the threshold is but in the lab autoconfgure will take autoconfigured bps and hard code values since very low memory available) However, if you run the AUTOCONFIGURE command on a database in an instance where sheapthres is not zero, sort memory tuning (sortheap) will not be enabled automatically. To enable sort memory tuning (sortheap), you must set sheapthres equal to zero using the UPDATE DATABASE MANAGER CONFIGURATION command. Note that changing the value of sheapthres may affect the sort memory usage in your previously existing databases.

$$

When explicitly invoking the Configuration Advisor with the AUTOCONFIGURE command, the setting of the DB2_ENABLE_AUTOCONFIG_DEFAULT registry variable will be ignored.

Summary

- Performance is the way that a computer system behaves in response to a particular workload measured in terms of system response time, throughput, and resource utilization
- Remember the law of diminishing returns
- Do not tune just for the sake of tuning
- Consider the whole system
- Change one parameter at a time
- Measure and configure by levels
- Check for hardware as well as software problems
- Understand the problem before you upgrade your hardware

PRESENTATION NOTES FOR PAGE 50

In summary, Data Studio replaces Control Center starting in DB2 10, Control Center will not be available anymore.

In addition to the features that Control Center has, Data Studio has a ton of more powerful tools that facilitate database management.

Multi OS support

Multi data server types support

Summary – DB2 10 Enhancements

- Runstats now supports index sampling
- SQL compiler registry variables can now be set for a specific SQL statement or application
- Statistical views improve the accuracy of cost estimation
- Intra-partition parallelism improvements better utilize multi-core processors
- Enhanced memory sharing on large POWER7® systems running AIX®
- Improved query performance through more efficient data prefetching (this also reduces the need for reorganization)
- Improved query performance on tables with composite indices, expression based indices, sparse indices
- Improved performance of star schema based queries

PRESENTATION NOTES FOR PAGE 51

The next steps…

PRESENTATION NOTES FOR PAGE 52

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 10300_WB1_DB2_Introduction_to_Performance.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 53

The Next Steps…

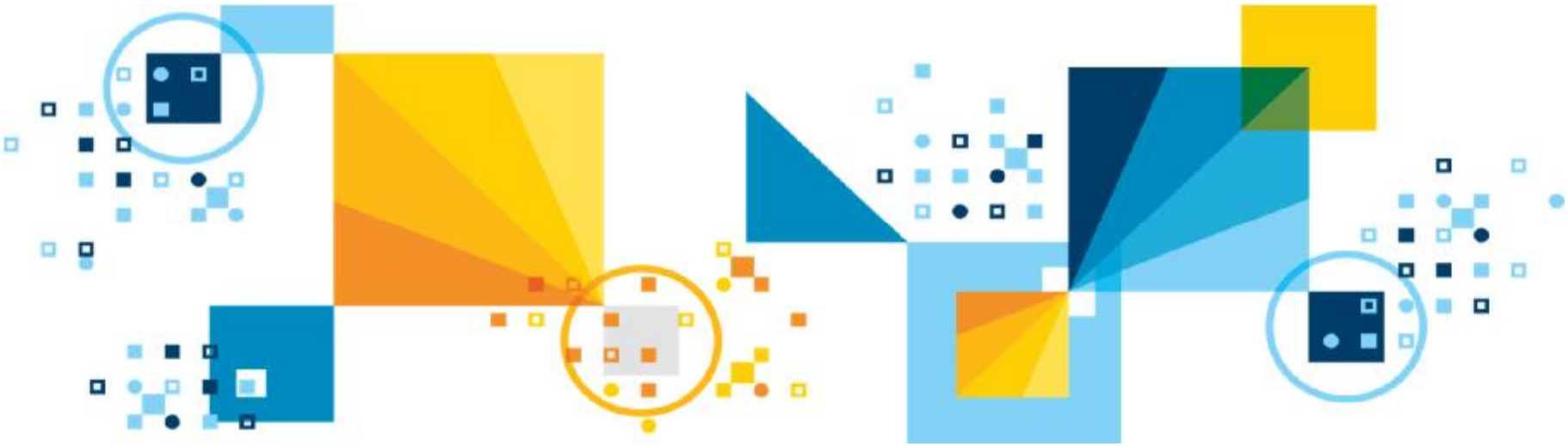
- The next set of modules to consider :
  - Module DB2 Performance Monitoring Essentials

PRESENTATION NOTES FOR PAGE 54

Questions?

askdata@ca.ibm.com

Subject: DB2 10.5 Performance and Monitoring



PRESENTATION NOTES FOR PAGE 55