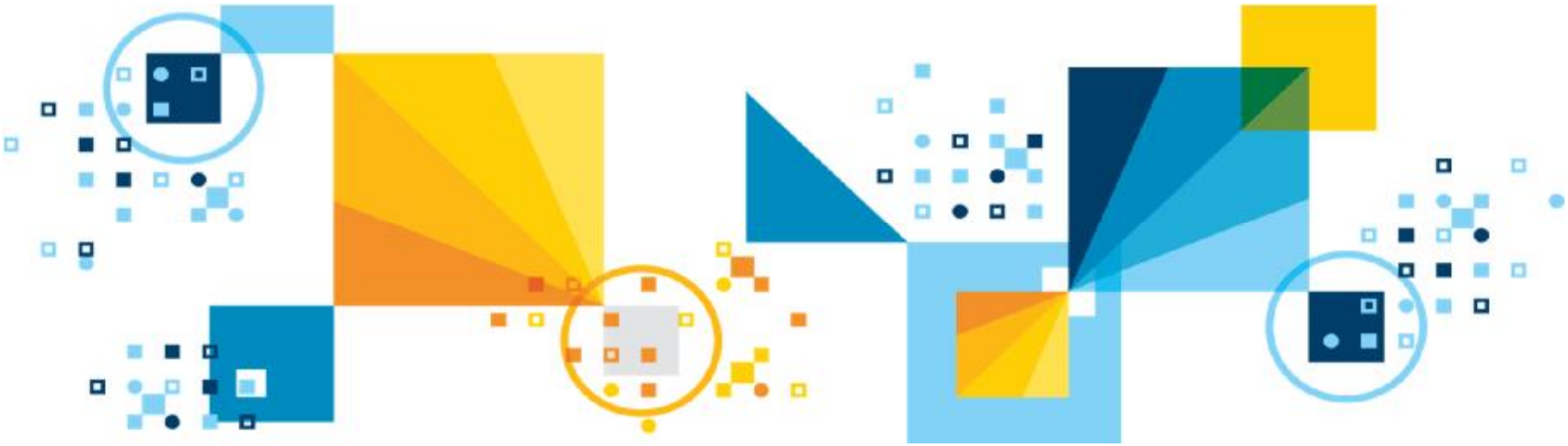
DB2 Maintenance Utilities and Performance

Module ID	10306
Length	1 hour + 2 hour hands on lab



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

February 9, 2015

PRESENTATION NOTES FOR PAGE 1

Disclaimer

PRESENTATION NOTES FOR PAGE 2

Module Information

- You should have completed or acquired the necessary knowledge for the following modules in order to complete this module:
  - Module DB2 Performance Monitoring Essentials
  - Module DB2 Performance Monitoring Essentials II
  - Module DB2 Locking and Logging for Performance

- After completing this module, you should be able to:
  - Describe the major db2 features that has a significant impact on utility performance
  - Able to perform the task to optimize the utilities' performance

PRESENTATION NOTES FOR PAGE 3

Agenda

- **Features that impact utility performance**
- **Maintenance commands, utilities and stored procedures**
- **Automating maintenance**

PRESENTATION NOTES FOR PAGE 4

Acknowledgement: The material for this presentation comes directly from the DB2 10 InfoCenter and DB2 licenses. It may or may not have been modified.

Refer to https://db2id.torolab.ibm.com/db2doc_v10/index.jsp

The intent of this presentation is to introduce you to the features in DB2 that can help you optimize the performance of utilities. If you have other information you wish to share with us for inclusion in future bootcamps please, feel free to share this information with us today in class or in follow up conversations or email.

We will use the terms utilities and commands interchangeably throughout the presentation. These terms can be overloaded at time. For example, if you want to backup a database you use the backup command.

As we discuss utilities, commands and parameters during this presentation you should be aware that we will not cover all possible options available to you. When you implement your data servers and databases you will need to choose the appropriate utilities and their options for your environment.

Some of the features we discuss in this presentation are not licensed with the base license for DB2 Enterprise Server Edition. These features, Storage Optimization and Performance Optimization must be licensed separately.

Speaker notes and slide contents have been taken directly from the DB2 10 InfoCenter.

Note to speaker:

The commands and utilities of primary interest for this presentation are highlighted in blue.

Logical and physical data organization impacts performance

- **Database partitioning**
  - Split database into multiple logical/physical partitions: DPF
- **Range (table) partitioning**
  - Split table into multiple table partitions within a database partition
- **Compression**
  - Relational and XML data / Index compression / Temporary table compression
  - Archived logs compression: This is a high-value feature
  - Adaptive Compression
- Related utilities:
  - REORGCHK
  - REORG
  - RUNSTATS

PRESENTATION NOTES FOR PAGE 5

An intelligent physical design based on best practices is the first step to creating a database that performs well. DB2 provides features such as database partitioning (clustering), range partitioning for tables and compression for relational data as well as external data such as XML or large objects that provide an initial performance gain to a database.

As the data base is used it will change, possibly growing, over time as records are inserted, updated and deleted. Both the data and metadata will change. At some point in time the initial performance boost given by a good layout will begin to decrease.

DB2 provides utilities to help maintain the performance of a database. Three such utilities are reorgchk, reorg and runstats.

Reorgchk can identify which tables or indexes need to be reorged.

Reorg is similar to a disk defrag. DB2 eliminates unused space within the tablespace containers, clustering data together more efficiently based on the indexes when possible.

Runstats gathers statics for the optimizer, and can also create a new compression dictionary if compression is used.

The Archived logs compression, As of DB2® Version 10, you can compress archived log files. This capability, in addition to data and index compression, along with backup compression, reduces the amount of disk space required for your database environment.

Disk I/O performance impacts applications and utilities

- **All utilities benefit from better disk I/O performance, including:**
  - REORGCHK, REORG, RUNSTATS, RESTORE
- **Prefetching is used to read data from disk to the buffer pools**
  - I/O servers are the processes/threads that read the data
  - Database configuration parameters
    - NUM_IOSERVERS
  - Related DB2 registry variable
    - DB2_AVOID_PREFETCH
- **Multi-temperature storage & storage groups**
  - Storage groups can be used for classifying and placing hot/cold data
  - Related operations:
    - ALTER TABLESPACE
    - ALTER STOGROUP

- **Adaptive Compression:**
  - Up to 5 times storage savings
  - New adaptive dictionaries for page level compression
  - Reduces the need for frequent reorganizations to achieve the most efficient compression level

PRESENTATION NOTES FOR PAGE 6

I/O servers are responsible for prefetching data to the buffer pool. If multiple containers exist for a table space, the database manager can initiate parallel I/O. Each I/O server processes the I/O workload for a separate container, so that several containers can be read in parallel. Parallel I/O can result in significant improvements in I/O throughput.

The general recommendation is to set num_ioservers equal to the number of disks being used by the database. If not sure you can leave it at AUTOMATIC.

The num_ioservers parameter specifies the number of I/O servers for a database. No more than this number of I/Os for prefetching and utilities can be in progress for a database at any time.

DB2 performs a CRASH RECOVERY check each time a database is started to see if a recovery operation is required. The DB2_AVOID_PREFETCH registry variable specifies whether prefetch should be used during crash recovery. If DB2_AVOID_PREFETCH is set to ON, prefetch is not used. The default value is OFF, allowing DB2 to take advantage of parallel I/O when performing a crash recovery.

If a container is added to a tablespace, or the PREFEETCH size is changed for a tablespace you should run RUNSTATS against the table.

When an indexed table has been modified many times, the data in the indexes might become fragmented. If the table is clustered with respect to an index, the table and index can get out of cluster order. Both of these factors can adversely affect the performance of scans using the index, and can impact the effectiveness of index page prefetching. REORG INDEX or REORG INDEXES can be used to reorganize one or all of the indexes on a table.

Parallel I/O can be defined for hardware configuration and runtime utility behavior

- **Set these parameters to AUTOMATIC**
  - Database manager parameter (optional): DFT_PREFETCH_SZ
  - Tablespace parameter: PREFETCHSIZE
- **State the exact number of drives for one or more tablespace's underlying disk arrays with the DB2 registry variable DB2_PARALLEL_IO**
    - Example, all table space containers reside on a RAID-4 array of 4 disks (1 drive is for parity):
      `db2set "DB2_PARALLEL_IO=*:3"`
      where * = all tablespaces and 3 is the # of drives data resides on
- **Related utilities:**
  - Data movement utilities:
    - Automatically take advantage of parallel I/O
  - Backup and Restore commands:
    - Also automatically take advantage of parallel operations, or can be explicitly set with the PARALELLISM command option

PRESENTATION NOTES FOR PAGE 7

DB2 automatically calculates the parallelism for data movement utilitites. The backup and restore utlities provide a parameter you can use to further tune or throttle their performance.

If you set the prefetch size of a table space to AUTOMATIC, the database manager uses the number of physical disks value that you specified for DB2_PARALLEL_IO to determine the prefetch size value. The default is 1 device per container because DB2 does not know about detailed RAID configurations.

If table space containers reside on an array of disks, configuring the DB2_PARALLEL_IO parameter allows DB2 to dispatch multiple prefetch requests simultaneously.

Optimal performance achieved when the DFT_PREFETCH_SZ database configuration parameter is set to AUTOMATIC (the default) and/or the tablspace's PREFETCHSIZE parameter is set to AUTOMATIC and DB2_PARALLEL_IO correctly specifies tablespace device configurations.

Data Server memory allocation impacts performance

- **DB2's Self Tuning Memory Management (STMM) handles runtime memory allocation and management**
  - No database administrator intervention required
  - Adapts to changing workload and environment
  - Automatically enabled for newly created databases
  - INSTANCE_MEMORY is a combination of DATABASE_MEMORY and APPL_MEMORY

- **Related utilities**
  - All utilities benefit from STMM

PRESENTATION NOTES FOR PAGE 8

In addition to the physical and logical layout of the database, DB2's processes, threads and allocated memory must work efficiently to provide the required performance.

DB2 provides an autonomic computing feature called Self Tuning Memory Management, or STMM, to relieve the DBA from the effort of manually monitoring and reconfiguring the runtime DB2 environment.

STMM is enabled by default for new databases. It dynamically monitors and adapts to the changing workload, recording its actions in the DB2 diagnostic log and saving its configuration changes in the STMM logs. If and when you restart the database, DB2 starts with the last known set of STMM settings.

If little memory is available, the performance benefits of self tuning will be limited. Because tuning decisions are based on database workload, workloads with rapidly changing memory requirements limit the effectiveness of the self-tuning memory manager (STMM). If the memory characteristics of your workload are constantly changing, the STMM will tune less frequently and under shifting target conditions. In this scenario, the STMM will not achieve absolute convergence, but will try instead to maintain a memory configuration that is tuned to the current workload.

Utility throttling provides a method to control the resource allocation of certain utilities and operations

- **Database manager configuration parameter:**
  - **`UTIL_IMPACT_LIM`** `[1:100], Default: 10, no throttled mode: 100`
- **Utilities:**
  - statistics collection
  - backup operations
  - rebalancing operations (ALTER TABLESPACE command)
  - asynchronous index cleanup (AIC - **`ALTER TABLE… DETACH`** command and global index definition)
- **To change the impact setting for a running utility:**
  - Command: SET **`UTIL_IMPACT_PRIORITY`** with *priority_level*:
    - **`100`** represents the highest priority and **`1`** represents the lowest priority. Will force an unthrottled utility to continue in throttled mode
    - **`0`** will force a throttled utility to continue unthrottled
- **Related commands:**
  - **`LIST UTILITIES`**
  - **`SET UTIL_IMPACT_PRIORITY FOR`** ***`utility_id`*** **`TO`** ***`priority_level`***

PRESENTATION NOTES FOR PAGE 9

Utility throttling regulates the performance impact of maintenance utilities so that they can run concurrently during production periods. Although the impact policy, a setting that allows utilities to run in throttled mode, is defined by default, you must set the impact priority, a setting that each cleaner has indicating its throttling priority, when you run a utility if you want to throttle it.

The throttling system ensures that the throttled utilities are run as frequently as possible without violating the impact policy. You can throttle statistics collection, backup operations, rebalancing operations, and asynchronous index cleanups.

You define the impact policy by setting the util_impact_lim configuration parameter.

Cleaners are integrated with the utility throttling facility. By default, each (index) cleaner has a utility impact priority of 50 (acceptable values are between 1 and 100, with 0 indicating no throttling). You can change the priority by using the SET UTIL_IMPACT_PRIORITY command or the db2UtilityControl API.

FYI, in case it needs to be discussed:

Asynchronous index cleanup (AIC) is the deferred cleanup of indexes following operations that invalidate index entries. AIC accelerates the detachment of a data partition from a partitioned table, and is initiated if the partitioned table contains one or more non-partitioned indexes. In this case, AIC removes all non-partitioned index entries that refer to the detached data partition, and any pseudo-deleted entries. After all of the indexes have been cleaned, the identifier that is associated with the detached data partition is removed from the system catalog. If the partitioned table has dependent materialized query tables (MQTs), AIC is not initiated until after a SET INTEGRITY statement is executed. Normal table access is maintained while AIC is in progress. Queries accessing the indexes ignore any invalid entries that have not yet been cleaned.

Asynchronous index cleanup for MDC tables

You can enhance the performance of a rollout deletion—an efficient method for deleting qualifying blocks of data from multidimensional clustering (MDC) tables—by using asynchronous index cleanup (AIC). AIC is the deferred cleanup of indexes following operations that invalidate index entries.

Indexes are cleaned up synchronously during a standard rollout deletion. When a table contains many record ID (RID) indexes, a significant amount of time is spent removing the index keys that reference the table rows that are being deleted. You can speed up the rollout by specifying that these indexes are to be cleaned up after the deletion operation commits.

DB2's Workload Management

- **Version 10 features extend the workload management capabilities provided in previous releases**
- **Available for**
  - DB2 Advanced Enterprise Server Edition Version 10
  - DB2 Database Enterprise Developer Edition Version 10
- **Benefits**
  - WLM can be used to control data resources used by agents running the LOAD utility and the new INGEST utility
  - WLM can now prioritize an activity based on the data that the activity accesses, either before the activity executes (predictively) or while the activity is executing (reactively)
    - Using data tags in tablespaces or storage groups
    - Using DATATAGINSC threshold

PRESENTATION NOTES FOR PAGE 10

To prioritize an activity, you use a combination of a data tag, which is a numeric identifier applied to a table space or storage group, and WLM controls. For example, if you have a table space IMPORTANT_TS containing critical data that has a data tag assigned to it, you could map any query that reads data from a table in this table space to a service class that is allocated a higher percentage of overall CPU cycles on the system.

You can assign a data tag directly to a table space or assign the data tag to the storage group for the table space and have the table space inherit the data tag from the storage group.

Predictive prioritization using work class and work action sets uses an estimated data tag list that is obtained for an activity at compile time, similar to cost and cardinality estimates. The estimated data tag list contains the data tags for all table spaces that the compiler believes will be accessed during execution of the activity. You can define work class sets to identify activities that have a particular data tag in their estimated data tag lists. You can then define a work action to map any activities matching a work class set to a specific service class before they begin to execute.

Reactive prioritization using the new DATATAGINSC threshold maps an activity to a different service class at run time when the activity accesses data that is assigned a particular data tag. For example, you can specify that an activity will be mapped to a different service class when it reads data from a table space with data tag value of 3. Reactive prioritization is useful if the compiler cannot accurately estimate the list of data tags for the activity. An example of such a case is a query against a range-partitioned table that uses parameter markers. The compiler cannot necessarily determine what table ranges are accessed in advance.

To support data tags, the following DB2 commands SQL reference statements have been added or modified:The output of the -tablespace parameter for the db2pd command now includes information about data tags.

The output of the -workclasses parameter for the db2pd command now lists the work class attributes below the basic work class information.

The ALTER TABLESPACE statement has the new DATA TAG clause.

The ALTER THRESHOLD statement has the new DATATAGINSC clause.

The ALTER WORK CLASS SET statement has the new DATA TAG LIST CONTAINS clause.

The CREATE TABLESPACE statement has the new DATA TAG clause.

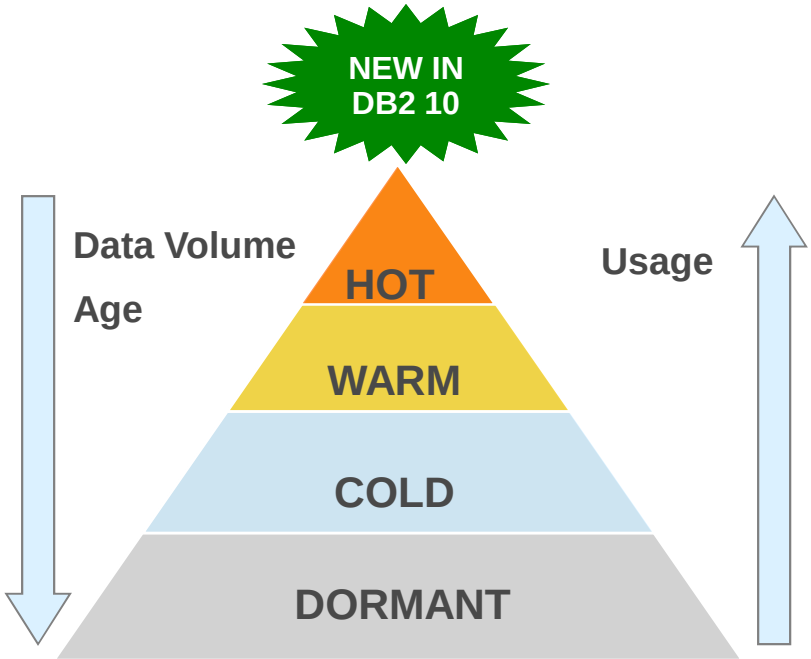
The CREATE THRESHOLD statement has the new DATATAGINSC clause.

The CREATE WORK CLASS SET statement has the new DATA TAG LIST CONTAINS clause.

Multi-temperature data storage

§ Current data is often considered to be hot data, but it typically becomes cold as it ages

§ These sets of *multi-temperature data* pose considerable challenges to DBAs who want to optimize the use of fast storage by trying not to store cold data there. As a data warehouse consumes more storage, optimizing the use of fast storage becomes increasingly important for managing storage costs



Sales data of this month = **most frequent**

Sales data of this quarter = **less frequent**

Sales data of previous quarters = **rarely accessed**

Sales data of past years = **historical data**

PRESENTATION NOTES FOR PAGE 11

With your hot data stored on your fastest storage assets, multi-temperature data storage can help to reduce the time it takes to retrieve your most frequently accessed data, while reducing the cost of storing infrequently accessed warm and cold data.

Multi-temperature data storage can provide fast access to data

- You can manage your IT budget more efficiently by configuring your database so that only frequently accessed data (*hot data*) is stored on expensive fast storage, such as solid-state drives (SSD), and infrequently accessed data (*cold data*) is stored on slower, less-expensive storage, such as low-rpm hard disk drives
- As hot data cools down and is accessed less frequently, you can dynamically move it to the slower storage, thereby extending the life of your less expensive storage assets that are used for storing warm and cold data

PRESENTATION NOTES FOR PAGE 12

It is a way to improve manageability of data and create different classes of storage where frequently accessed data (hot data) is stored on fast storage, while infrequently accessed data (warm data) is stored on slightly slower storage, and rarely accessed data (cold data) is stored on slow, less-expensive storage. As hot data cools down and is accessed less frequently, you can dynamically move it to the slower storage.

Optimizing data into classes of storage becomes increasingly important in managing storage costs as a data warehouse grows in its amount of storage.

The priority of the data is based on:

Frequency of access

Acceptable access time, defined by QoS specified in Service Level Agreement (SLA)

Volatility of the data

Application requirements

Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow

With your hot data stored on your fastest storage assets, multi-temperature data storage can help to reduce the time it takes to retrieve your most frequently accessed data, while reducing the cost of storing infrequently accessed warm and cold data.

As hot data cools down and is accessed less frequently, you can dynamically move it to the slower storage, thereby extending the life of your less expensive storage assets that are used for storing warm and cold data.

Storage Groups

- Storage Groups allow the flexibility to implement Multi-temperature Data Management in Automatic Storage table spaces
- Different Storage Groups can represent different classes of storage
  - **Hot** data assigned to storage groups with fast devices
  - **Warm** or **Cold** data assigned to slower devices
- Easy maintenance when data ages and needs to be moved to a different storage class

PRESENTATION NOTES FOR PAGE 13

So now we know there is a way to characterize data… how does DB2 leverage it? Using Storage Groups.

Multi-temperature Storage provides the ability to assign priority to data (hot, warm, cool, cold) and dynamically assign it to different classes of storage. This is carried out by the introduction of a new concept called Storage Groups. New layer of abstraction between logical (table spaces) and physical storage (containers)

A storage group contains storage paths with similar characteristics . Some critical attributes of the underlying storage to consider when creating or altering a storage group are available storage capacity, latency, data transfer rates, and the degree of RAID protection.

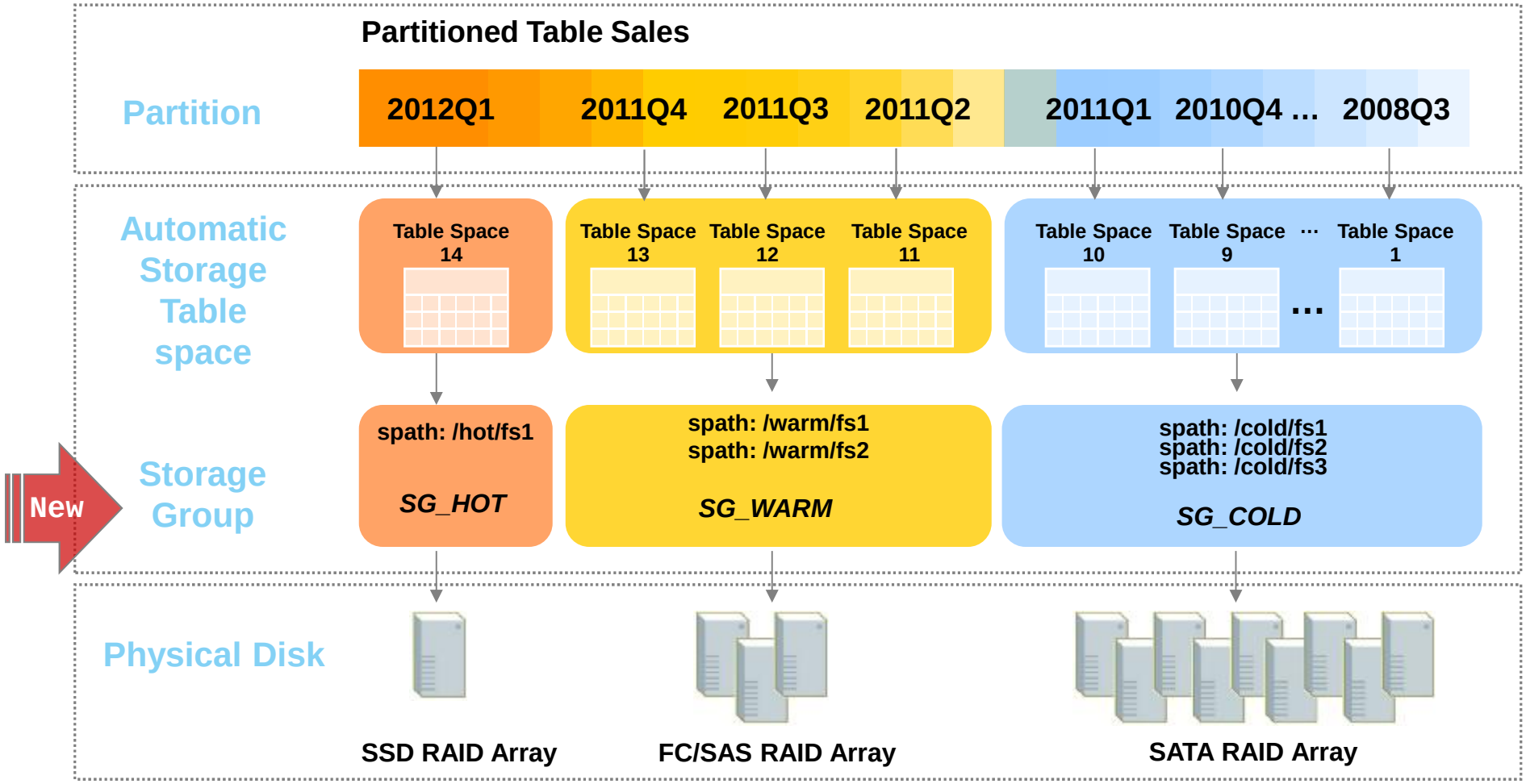
For speaker information only:

Automatic storage is the strategic direction for DB2 and it introduced the notion of storage groups, though did not externalize it.

Multi-temperature externalizes the concept of storage groups to support this strategic direction and still leverages the advantage of easier storage management by managing a small set (on the order of 3-5 layers) of storage subsystems.

Mapping table spaces directly to Solid State Drives (SSD) containers is actually possible in DB2 today without introducing the notion of storage groups but that brings back the original pain point where DBA's have to manage the storage allocation individually for hundreds of table spaces. In fact with DMS table spaces, customers could implement multi-temperature today by creating containers on the desired storage subsystem, but this would not work with automatic storage.

Leveraging Storage Groups



PRESENTATION NOTES FOR PAGE 14

With introduction of storage groups, now there is an extra layer of abstraction between tbsp and disks

Now you can group table spaces sharing similar characteristics into same group of storage paths.

After you create storage groups that map to the different classes of storage in your database management system, you can assign automatic storage table spaces to those storage groups, based on which table spaces have hot or cold data. You can dynamically reassign a table space to a different storage group as the data changes or your business direction changes.

Multi-temperature Storage Integrates with DB2 WLM

- Existing WLM perspectives are user-centric (who) and request-centric (what)
- Introducing a new perspective “data-centric” (where)
  - New **data tag** attribute
    - For storage group or table space
    - Priority can be given to requests based on what data is accessed [Values 1 (high) – 9 (low)]
- WLM work class and threshold DLL have been extended to support the new data tag attribute
  - DB2 optimizer can provide an estimated list of data tags for data touched by a query at compile
    - The data tag can influence the initial placement of the activity into a service class
- *New* Data tag threshold DATATAGINSC uses information that is available at runtime to remap an activity to a different service subclass

NEW IN DB2 10

PRESENTATION NOTES FOR PAGE 15

DB2 Workload Manager gives users the ability to prioritize incoming work based on “who” submitted the work and “what” type of work it is

Beginning with DB2 Galileo, WLM gives users a data-centric (“where”) approach so that priority can be given to requests based on what data is accessed

Users can assign a data tag attribute (a value from 0 to 9) to a storage group or table space. The data tag can be used by WLM to determine how to treat the work

Data-centric Example

One customer use case is where a large number of users makes it difficult to discriminate based on who is running the activity (i.e. workloads through connection attributes) and what type of activity is being run (i.e. through work action sets). Therefore business priority is best assigned based on the business priority given to sets of data rather than to users or activity types.

WLM can use data tags predictively

The Optimizer gathers a list of data tags for all table spaces used by an SQL statement at compile time

The data tag can influence the initial placement of the statement into a service class

WLM can also use data tags reactively

At run time, based on the tag of a table space being accessed, a statement can be remapped into a lower priority subclass

Submitter-centric (Who) : The business priority assigned to the submitting process/role determines the processing priority of the database requests. WLM provides this support though DB2 Workloads, based on 10 different connection attributes.

Request-centric (What) :The business priority assigned to the type of work being submitted determines the processing priority of the database requests. WLM provides this support through DB2 Work Action Sets based on SQL statement types, compiler estimates and routine schemas

DB2's Deep Compression, now gets Smarter: Adaptive Compression

§ Available for
- – DB2 Advanced Enterprise Server Edition Version 10
  - • IBM® DB2 Storage Optimization Feature and PVU charge metric
- – DB2 Database Enterprise Developer Edition Version 10
- – IBM InfoSphere Warehouse Enterprise Base Edition Version 10
- – IBM Base Warehouse Feature for DB2 Version 10

§ Components
- – Table compression
- – Index compression
- – Row and page level compression

§ Benefits
- – Data is compressed on disk and in memory
  - • Compressed data means fewer I/Os to read/write
  - • More data in memory means better buffer pool hit ratios
  - • **Maintenance is minimized and compression ratio is improved**

    Page compression adapts the dictionary over time based on the new incoming data, therefore REORG operations are needed much less often for dictionary rebuild

§ Related utilities: RUNSTATS, BACKUP, RESTORE, REORG

PRESENTATION NOTES FOR PAGE 16

The Storage Optimization feature is not licensed with the base license for DB2 Enterprise Server Edition. It must be licensed separately.

Storage Optimization feature include for AESE and Database Enterprise Developer Edition

runstats is used to manage the new compression dictionary.

Backup and restore utilities include the compression dictionary when they are used.

Adaptive compression improves upon the compression rates that can be achieved using classic row compression by itself. Adaptive compression incorporates classic row compression; however, it also works on a page-by-page basis to further compress data. Of the various data compression techniques in the DB2 product, adaptive compression offers the most dramatic possibilities for storage savings.

Adaptive compression actually uses two compression approaches. The first employs the same table-level compression dictionary used in classic row compression to compress data based on repetition within a sampling of data from the table as a whole. The second approach uses a page-level dictionary-based compression algorithm to compress data based on data repetition within each page of data. The dictionaries map repeated byte patterns to much smaller symbols; these symbols then replace the longer byte patterns in the table. The table-level compression dictionary is stored within the table object for which it is created, and is used to compress data throughout the table. The page-level compression dictionary is stored with the data in the data page, and is used to compression only the data within that page. For more information about the role each of these dictionaries in compressing data, see Compression dictionaries.

Note to speaker: This information comes from the Storage Optimization feature license.

Agenda

- Features that impact utility performance
- **Maintenance commands, utilities, and stored procedures**
- Automating maintenance

PRESENTATION NOTES FOR PAGE 17

We'll discuss some of the commands, utilities and stored procedures that can be used for data maintenance. The first set of these deals with utilities data that generally is at some point in time offline, or at rest. These utilities generally move data into the database or out of it.

DB2 provides commands and utilities to move data into and out of a database

§ Commands and utilities

- DB2MOVE : EXPORT, IMPORT, LOAD and COPY option
  New: parallel processing for COPY is supported!
- EXPORT : moves data out of database into flat files
- IMPORT : inserts data into database from flat files
- LOAD : fast method to load large amounts of data
- LOAD QUERY : check state/phase of LOAD operation or state of a table
- INGEST : high-speed client-side DB2 utility that streams data from files and pipes into DB2 target tables, without affecting normal workload

§ Stored Procedures

- ADMIN_MOVE_TABLE: move a table, online or offline
- ADMIN_COPY_SCHEMA: copy a specific schema and all objects contained in it and copy tables with or without the data of the original tables

PRESENTATION NOTES FOR PAGE 18

We'll cover the EXPORT, IMPORT, LOAD and DB2MOVE commands in the next few slides. Here's a quick overview:

The EXPORT command exports data from a database, into external (flat) files, to one of several external file formats.

The IMPORT command inserts data from an external file with a supported file format into a table, hierarchy, view or nickname.

The LOAD command loads data into a DB2 table from either a local or remote location.

The LOAD QUERY command checks the status of a load operation during processing and returns the table state. If a load is not processing, then the table state alone is returned. A connection to the same database, and a separate CLP session are also required to successfully invoke this command. It can be used either by local or remote users.

Following a load operation, the loaded table might be in set integrity pending state in either READ or NO ACCESS mode. The SET INTEGRITY statement is used to:

* Bring one or more tables out of set integrity pending state
* Place one or more tables in set integrity pending state.
* Place one or more tables into full access state.
* Prune the contents of one or more staging tables.

SET CONSTRAINTS can be specified in place of SET INTEGRITY for compatibility with previous versions of DB2.

The DB2MOVE command simplifies moving large numbers of tables

- **db2move command is not available with DB2 clients. Issue it directly on the server**
- **This utility attempts to copy all allowable schema objects except for:**
  - **Hierarchy**
  - **Staging tables**
  - **jars (Java routine archives)**
  - **Nicknames**
  - **Packages**
  - **View hierarchies**
  - **Object privileges (All new objects are created with default authorizations)**
  - **Statistics (New objects do not contain statistics information)**
  - **Index extensions (user-defined structured type related)**
  - **User-defined structured types and their transform functions**

PRESENTATION NOTES FOR PAGE 19

The DB2MOVE data movement tool, when used in the EXPORT/IMPORT/LOAD mode, facilitates the movement of large numbers of tables between DB2 databases located. The tool queries the system catalog tables for a particular database and compiles a list of all user tables. It then exports these tables in PC/IXF format. The PC/IXF files can be imported or loaded to another local DB2 database on the same system, or can be transferred to another workstation platform and imported or loaded to a DB2 database on that platform. Tables with structured type columns are not moved when this tool is used. When used in the COPY mode, this tool facilitates the duplication of a schema.

Performance for the db2move command with the IMPORT or LOAD actions can be improved by altering the default buffer pool, IBMDEFAULTBP, and by updating the configuration parameters sortheap, util_heap_sz, logfilsiz, and logprimary.

In case the subject comes up:

A structured type is a user-defined data type containing one or more named attributes, each of which has a data type. Attributes are properties that describe an instance of a type. A geometric shape, for example, might have attributes such as its list of Cartesian coordinates. A person might have attributes of name, address, and so on. A department might have attributes of a name or some other kind of ID.

The DB2MOVE command simplifies moving large numbers of tables

§ COPY

- **Copies schema templates (DDL with or without data) from a source database to a target database**
- **move an entire schema from a source database to a target database (LOAD only)**
- **The target database must be a local database**
- **Schema and Tablespace mapping definition supported**
- **PARALLEL [0:16] option to have the load operations for the tables in the schema(s) spread across a number of threads**

```
db2move dbsrc COPY
    -sn schema1
    -co TARGET_DB dbtgt USER myuser1 USING mypass1
        SCHEMA_MAP ((schema1,newschema1))
        TABLESPACE_MAP ((ts1,ts2), SYS_ANY))
```

PRESENTATION NOTES FOR PAGE 20

The DB2MOVE data movement tool, when used in the EXPORT/IMPORT/LOAD mode, facilitates the movement of large numbers of tables between DB2 databases located. The tool queries the system catalog tables for a particular database and compiles a list of all user tables. It then exports these tables in PC/IXF format. The PC/IXF files can be imported or loaded to another local DB2 database on the same system, or can be transferred to another workstation platform and imported or loaded to a DB2 database on that platform. Tables with structured type columns are not moved when this tool is used. When used in the COPY mode, this tool facilitates the duplication of a schema.

Performance for the db2move command with the IMPORT or LOAD actions can be improved by altering the default buffer pool, IBMDEFAULTBP, and by updating the configuration parameters sortheap, util_heap_sz, logfilsiz, and logprimary.

In case the subject comes up:

A structured type is a user-defined data type containing one or more named attributes, each of which has a data type. Attributes are properties that describe an instance of a type. A geometric shape, for example, might have attributes such as its list of Cartesian coordinates. A person might have attributes of name, address, and so on. A department might have attributes of a name or some other kind of ID.

The DB2MOVE command simplifies moving large numbers of tables

§ **EXPORT, IMPORT, LOAD**

- **Facilitates the movement of a large numbers of tables between DB2 databases**
- **Uses `db2move.lst` staging file**
- **Exports tables in PC/IXF format**
- **Stores LOB/XML data type separately**
  - **Round-robin fashion if more than one path is defined**
- **Useful for cloning databases when there is no support for cross-platform backup and restore operations**

```
db2move sample export
                    -tc userid1,us*rid2
                    -tn tbname1,*tbname2

db2move sample import -l D:\LOBPATH1, C:\LOBPATH2

db2move sample load -l /home/userid/lobpath,/tmp
```

PRESENTATION NOTES FOR PAGE 21

The DB2MOVE data movement tool, when used in the EXPORT/IMPORT/LOAD mode, facilitates the movement of large numbers of tables between DB2 databases located. The tool queries the system catalog tables for a particular database and compiles a list of all user tables. It then exports these tables in PC/IXF format. The PC/IXF files can be imported or loaded to another local DB2 database on the same system, or can be transferred to another workstation platform and imported or loaded to a DB2 database on that platform. Tables with structured type columns are not moved when this tool is used. When used in the COPY mode, this tool facilitates the duplication of a schema.

Performance for the db2move command with the IMPORT or LOAD actions can be improved by altering the default buffer pool, IBMDEFAULTBP, and by updating the configuration parameters sortheap, util_heap_sz, logfilsiz, and logprimary.

In case the subject comes up:

A structured type is a user-defined data type containing one or more named attributes, each of which has a data type. Attributes are properties that describe an instance of a type. A geometric shape, for example, might have attributes such as its list of Cartesian coordinates. A person might have attributes of name, address, and so on. A department might have attributes of a name or some other kind of ID.

The EXPORT utility

- Extracts data using an SQL select or an XQuery statement and places that information into a file
- Simple and flexible data movement utility
- You can activate it by:
  - Issuing the EXPORT command in the CLP
  - Calling the ADMIN_CMD stored procedure
  - Calling the db2Export API through a user application


```
db2 export to
          myfile.ixf of ixf
          messages msgs.txt
          select * from staff
```

PRESENTATION NOTES FOR PAGE 22

The EXPORT utility is best suited in situations where you want to store data in an external file, either to process it further or move the data to another table.

There are a few ways to improve the EXPORT utility's performance. The EXPORT utility is an embedded SQL application and does SQL fetches internally. Optimization techniques that apply to SQL operations apply to the EXPORT utility as well. Consider taking advantage of large buffer pools, indexing, and sort heaps. In addition, try to minimize device contention on the output files by placing them away from the containers and log devices.

Large objects and XML files can be exported in round robin fashion by specifying multiple directories.

It is necessary to manually move all other database objects associated with the tables (such as aliases, views, or triggers) as well as objects that these tables may depend on (such as user-defined types or user-defined functions). These can be created with the DB2LOOK command.

IBM provides the High Performance Unload (HPU) product as an alternative to the EXPORT utility. HPU must be purchased separately and is not covered in this presentation.

You can use the export utility with DB2 Connect™ if you need to move data in IXF format.

You must have DATAACCESS authority or the CONTROL or SELECT privilege for each table or view participating in the export operation.

The EXPORT command is used to extract data from a database

- Cross-platform compatible
- An embedded SQL application executing SQL fetches
- Does not move aliases, views, triggers, user defined types or user defined functions
- Can store data in ASCII or PC/IXF format
- Optimization techniques
  - Create indexes for data (Design Advisor can help with this)
  - Use large buffer pools (STMM can help with this)
  - Allocate enough memory for sort heaps (STMM can help with this)
  - Minimize device contention on/for output files by placing them away from the containers and log devices
- Optimization parameters
  - None. DB2 automatically calculates buffer pools, buffer pool sizes, etc.
  - DB2 10 uses new query performance improvements!

PRESENTATION NOTES FOR PAGE 23

The IMPORT utility

- The import utility populates a table, typed table, nickname or view with data, from an external file, using an SQL INSERT statement
- If the table or view receiving the imported data already contains data, the input data can either replace or be appended to the existing data
- **Import modes:**

Mode	Best practice usage
INSERT	Inserts input data into target table without changing existing data
INSERT_UPDATE	Updates rows with matching primary key values with values of input rows Where there's no matching row, inserts imported row into the table
REPLACE	Deletes all existing data and inserts imported data, while keeping table and index definitions
REPLACE_CREATE	Deletes all existing data and inserts imported data, while keeping table and index definitions Creates target table and index if they don't exist
CREATE	Creates target table and index Can specify the name of the table space where the new table is created

PRESENTATION NOTES FOR PAGE 24

Like export, import is a relatively simple data movement utility. It can be activated by issuing CLP commands, by calling the ADMIN_CMD stored procedure, or by calling its API, db2Import, through a user application.

There are a number of data formats that import supports, as well as features that can be used with import:

Import supports IXF, ASC, and DEL data formats.

Import can be used with file type modifiers to customize the import operation.

Import can be used to move hierarchical data and typed tables.

Import logs all activity, updates indexes, verifies constraints, and fires triggers.

Import allows you to specify the names of the columns within the table or view into which the data is to be inserted.

Import can be used with DB2 Connect.

The IMPORT command is used to INSERT data into a database

- Cross-platform compatible
- Can read data in ASCII or PC/IXF format
- An alternative to the LOAD command when the target table
  - Is a view
  - Has constraints and you don't want the table put in the "**set integrity pending**" state
  - Has triggers and you want them fired
- Optimization techniques
  - Minimize device contention on input files by placing them away from the containers and log devices
- Optimization parameters
  - **ALLOW NO | WRITE ACCESS**
    - Online only for **INSERT** or **INSERT_UPDATE** import modes
  - **COMMITCOUNT AUTOMATIC | *n***
    - If Online a valid *n* value is required

```
db2 import from datafile1.del of del
          method P(1, 3, 4)
          replace into table1 (c1, c3, c4)
```

PRESENTATION NOTES FOR PAGE 25

The IMPORT command inserts data from an external file with a supported file format into a table, hierarchy, view or nickname.

The default behavior is to not allow other applications write access while the IMPORT operation is in progress. This is an offline IMPORT and is faster than an online IMPORT. This is controlled with the ALLOW NO|WRITE ACCESS parameter.

You can control disk I/O with the COMMITCOUNT parameter. If you specify a number, IMPORT will perform a COMMIT after every n records are imported. If you specify AUTOMATIC, import internally determines when a commit needs to be performed. In this case, the utility will commit for either one of two reasons:

* to avoid running out of active log space
* to avoid lock escalation from row level to table level

If the ALLOW WRITE ACCESS option is specified, and the COMMITCOUNT option is not specified, the import utility will perform commits as if COMMITCOUNT AUTOMATIC had been specified.

Use the LOAD command if you need a faster option than the IMPORT command

- Cross-platform compatible
- Can read data in ASCII or PC/IXF format or from a CURSOR, from either a local or remote system
- It is faster then the import utility because it writes formatted pages directly into the database rather than using SQL INSERTS
- Allows the option to not log the data or use the COPY option to save a copy of the loaded data
- Optimization techniques
  - Minimize device contention on input files by placing them away from the containers and log devices
  - Use hardware and network performance tuning techniques when not loading from a local system or disk
  - Use workload management to:
    - limit or expand resources used by LOAD
    - prioritize the activity with data tags and WLM controls (DATATAGINSC)
  - Customize optimization parameters (see next slide)

PRESENTATION NOTES FOR PAGE 26

The LOAD command loads data into a DB2 table. Data residing on the server can be in the form of a file, tape, or named pipe. Data residing on a remotely connected client can be in the form of a fully qualified file or named pipe. Data can be loaded from a user-defined cursor or by using a user-written script or application.

Options for improving load performance

There are various command parameters that you can use to optimize load performance. There are also a number of file type modifiers unique to load which can, in some cases, significantly improve that utility's performance. We'll cover those on the next slide.
Command parameters

Following is information about the performance implications of various options available through the load utility:

ALLOW READ ACCESS
This option allows you to query a table while a load operation is in progress. You can only view data that existed in the table prior to the load operation. If the INDEXING MODE INCREMENTAL option is also specified, and the load operation fails, the subsequent load terminate operation might have to correct inconsistencies in the index. This requires an index scan which involves considerable I/O. If the ALLOW READ ACCESS option is also specified for the load terminate operation, the buffer pool is used for I/O.
COPY YES or NO
Use this parameter to specify whether a copy of the input data is to be made during a load operation. COPY YES, which is only applicable when forward recovery is enabled, reduces load performance because all of the loading data is copied during the load operation. The increased I/O activity might increase the load time on an I/O-bound system. Specifying multiple devices or directories (on different disks) can offset some of the performance penalty resulting from this operation. COPY NO, which is only applicable when forward recovery is enabled, does not affect load performance. However, all table spaces related to the loaded table will be placed in a Backup Pending state, and those table spaces must be backed up before the table can be accessed.
CPU_PARALLELISM
Use this parameter to exploit the number of processes running per database partition (if this is part of your machine's capability), and significantly improve load performance. The parameter specifies the number of processes or threads used by the load utility to parse, convert, and format data records. The maximum number allowed is 30. If there is insufficient memory to support the specified value, the utility adjusts the value. If this parameter is not specified, the load utility selects a default value that is based on the number of CPUs on the system.

* the anyorder file type modifier is not specified
* the PARTITIONING_DBPARTNUMS option (and more than one partition is to be used for partitioning) is not specified

If tables include either LOB or LONG VARCHAR data, CPU_PARALLELISM is set to 1. Parallelism is not supported in this case.

Although use of this parameter is not restricted to symmetric multiprocessor (SMP) hardware, you might not obtain any discernible performance benefit from using it in non-SMP environments.
Figure 1. Record Order in the Source Data is Preserved When the Number of Processes Running Per Database Partition is Exploited During a Load Operation
Record order in source data is preserved when the number of processes running per database partition is exploited.
DATA BUFFER
The DATA BUFFER parameter specifies the total amount of memory, in 4 KB units, allocated to the load utility as a buffer. It is recommended that this buffer be several extents in size. The data buffer is allocated from the utility heap; however, the data buffer can exceed the setting for the util_heap_sz database configuration parameter as long as there is available memory in the system.
DISK_PARALLELISM
The DISK_PARALLELISM parameter specifies the number of processes or threads used by the load utility to write data records to disk. Use this parameter to exploit available containers when loading data, and significantly improve load performance. The maximum number allowed is the greater of four times the CPU_PARALLELISM value (actually used by the load utility), or 50. By default, DISK_PARALLELISM is equal to the sum of the table space containers on all table spaces containing objects for the table being loaded, except where this value exceeds the maximum number allowed.
NONRECOVERABLE
If forward recovery is enabled, use this parameter if you do not need to be able to recover load transactions against a table upon rollforward. A NONRECOVERABLE load and a COPY NO load have identical performance. However, there is a significant difference in terms of potential data loss. A NONRECOVERABLE load marks a table as not rollforward recoverable while leaving the table fully accessible. This can create a problematic situation in which if you need to rollforward through the load operation, then the loaded data as well as all subsequent updates to the table will be lost. A COPY NO load places all dependent table spaces in the Backup Pending state which renders the table inaccessible until a backup is performed. Because you are forced to take a backup after that type of load, you will not risk losing the loaded data or subsequent updates to the table. That is to say, a COPY NO load is totally recoverable.
Note: When these load transactions are encountered during subsequent restore and rollforward recovery operations, the table is not updated, and is marked invalid. Further actions against this table are ignored. After the rollforward operation is complete, the table can only be dropped.
SAVECOUNT
Use this parameter to set an interval for the establishment of consistency points during the load phase of a load operation. The synchronization of activities performed to establish a consistency point takes time. If done too frequently, there is a noticeable reduction in load performance. If a very large number of rows is to be loaded, it is recommended that a large SAVECOUNT value be specified (for example, a value of 10 million in the case of a load operation involving 100 million records).

A load restart operation automatically continues from the last consistency point, provided that the load restart operation resumes from the load phase.
STATISTICS USE PROFILE
Collect statistics specified in table statistics profile. Use this parameter to collect data distribution and index statistics more efficiently than through invocation of the RUNSTATS utility following completion of the load operation, even though performance of the load operation itself decreases (particularly when DETAILED INDEXES ALL is specified).

For optimal performance, applications require the best data distribution and index statistics possible. Once the statistics are updated, applications can use new access paths to the table data based on the latest statistics. New access paths to a table can be created by rebinding the application packages using the BIND command. The table statistics profile is created by running the RUNSTATS command with the SET PROFILE options.

When loading data into large tables, it is recommended that a larger value for the stat_heap_sz (statistics heap size) database configuration parameter be specified.
USE <tablespace-name>
When an ALLOW READ ACCESS load is taking place and the indexing mode is REBUILD, this parameter allows an index to be rebuilt in a system temporary table space and copied back to the index table space during the index copy phase of a load operation.

By default, the fully rebuilt index (also known as the shadow index) is built in the same table space as the original index. This might cause resource problems as both the original and the shadow index reside in the same table space simultaneously. If the shadow index is built in the same table space as the original index, the original index is instantaneously replaced by the shadow. However, if the shadow index is built in a system temporary table space, the load operation requires an index copy phase which copies the index from a system temporary table space to the index table space. There is considerable I/O involved in the copy. If either of the table spaces is a DMS table space, the I/O on the system temporary table space might not be sequential. The values specified by the DISK_PARALLELISM option are respected during the index copy phase.
WARNINGCOUNT
Use this parameter to specify the number of warnings that can be returned by the utility before a load operation is forced to terminate. Set the WARNINGCOUNT parameter to a relatively low number if you are expecting only a few or no warnings. The load operation stops after the WARNINGCOUNT number is reached. This gives you the opportunity to correct problems before attempting to complete the load operation.

File type modifiers

ANYORDER
By default, the load utility preserves record order of source data. When load is operating under an SMP environment, synchronization between parallel processing is required to ensure that order is preserved.
In an SMP environment, specifying the anyorder file type modifier instructs the load utility to not preserve the order, which improves efficiency by avoiding the synchronization necessary to preserve that order. However, if the data to be loaded is presorted, anyorder might corrupt the presorted order, and the benefits of presorting are lost for subsequent queries.
Note: The anyorder file type modifier has no effect if CPU_PARALLELISM is 1, and it is not compatible with the SAVECOUNT option.

BINARYNUMERICS, ZONEDDECIMAL and PACKEDDECIMAL
For fixed length non-delimited ASCII (ASC) source data, representing numeric data in binary can result in improved performance when loading. If the packeddecimal file type modifier is specified, decimal data is interpreted by the load utility to be in packed decimal format (two digits per byte). If the zoneddecimal file type modifier is specified, decimal data is interpreted by the load utility to be in zoned decimal format (one digit per byte). For all other numeric types, if the binarynumerics file type modifier is specified, data is interpreted by the load utility to be in binary format.
Note:

* When the binarynumerics, packeddecimal, or zoneddecimal file type modifiers are specified, numeric data is interpreted in big-endian (high byte first) format, regardless of platform.
* The packeddecimal and zoneddecimal file type modifiers are mutually exclusive.
* The packeddecimal and zoneddecimal file type modifiers only apply to the decimal target columns, and the binary data must match the target column definitions.
* The reclen file type modifier must be specified when the binarynumerics, packeddecimal, or zoneddecimal file type modifiers are specified.

FASTPARSE
Use with caution. In situations where the data being loaded is known to be valid, it can be unnecessary to have load perform the same amount of syntax checking as with more suspect data. In fact, decreasing the scope of this step can improve load's performance by about 10 or 20 percent. This can be done by using the fastparse file type modifier, which reduces the data checking that is performed on user-supplied column values from ASC and DEL files.

NOROWWARNINGS
During a load operation, warning messages about rejected rows are written to a specified file. However, if the load utility has to process a large volume of rejected, invalid or truncated records, it can adversely affect load's performance. In cases where many warnings are anticipated, it is useful to use the norowwarnings file type modifier to suppress the recording of these warnings.

PAGEFREESPACE, INDEXFREESPACE, and TOTALFREESPACE
As data is inserted and updated in tables over time, the need for table and index reorganization grows. One solution is to increase the amount of free space for tables and indexes using pagefreespace, indexfreespace, and totalfreespace. The first two modifiers, which take precedence over the PCTFREE value, specify the percentage of data and index pages that is to be left as free space, while totalfreespace specifies the percentage of the total number of pages that is to be appended to the table as free space.

DB2 automatically calculates values for the LOAD command parameters if you do not specify a value

§ LOAD command optimization parameters

– statistics *no* | *use profile*	relates to RUNSTATS
– data buffer *buffer-size*	number of 4KB pages used for data transfer within the utility
– sort buffer *buffer-size*	override sortheap db cfg parameter
– cpu_parallelism *n*	number of processes or threads used to build table objects
– disk_parallelism *n*	number of processes or threads used to write to tablespace containers
– fetch_parallelism *no* | *yes*	parallelize fetching from remote cursor
– sourceuserexit ... parallelize	invoke multiple user exit processes simultaneously in multipartition databases
– open *num-sess* sessions	number of I/O sessions to be used with TSM or other vendor product (if COPY YES)

PRESENTATION NOTES FOR PAGE 27

NOTE TO SPEAKER: Default values are shown in blue.

The load utility attempts to deliver the best performance possible by determining optimal values for DISK_PARALLELISM, CPU_PARALLELISM, and DATA BUFFER if these parameters have not be specified by the user. Optimization is based on the size and the free space available in the utility heap. Consider using the autonomic DISK_PARALLELISM and CPU_PARALLELISM settings before attempting to tune these parameters for your particular needs.

SAVECOUNT n - Specifies that the load utility is to establish consistency points after every n rows. This option should be selected if the load operation will be monitored using LOAD QUERY. If the value of n is not sufficiently high, the synchronization of activities performed at each consistency point will impact performance.

STATISTICS USE PROFILE - Collect statistics during the load according using a profile created earlier by the RUNSTAT command. If statistics are invalid for target table then performance will be impacted.

STATISTICS NO - no statistics will be collected; statistics in the catalogs will not be altered. NO is the default. Use RUNSTATS after the LOAD to generate statistics.

CPU_PARALLELISM n - number of processes or threads that the load utility will create.

DATA BUFFER buffer-size - Specifies the number of 4 KB pages (regardless of the degree of parallelism) to use as buffered space for transferring data within the utility. If the value specified is less than the algorithmic minimum, the minimum required resource is used.

SORT BUFFER buffer-size - Override the sortheap database configuration parameter when loading tables with indexes and only when the INDEXING MODE parameter is not specified as DEFERRED. Useful for throttling the sort memory that is used when loading tables with many indexes without changing the value of sortheap, which would also affect general query processing.

DISK_PARALLELISM n

Specifies the number of processes or threads that the load utility will create for writing data to the table space containers.

FETCH_PARALLELISM YES | NO - When performing a load from a cursor, the load utility attempts to parallelize fetching from the remote data source if possible

SOURCEUSEREXIT executable ... PARALLELIZE

Specifies an executable filename which will be called to feed data into the utility.

PARALLELIZE increases the throughput of data coming into the load utility by invoking multiple user exit processes simultaneously. This option is only applicable in multi-partition database environments and is ignored in single-partition database environments.

OPEN num-sess SESSIONS

This sub-parameter of the COPY YES parameter specifies the number of I/O sessions to be used with TSM or the vendor product. The default value is 1.

LOAD Utility - Restrictions

- Loading data into nicknames is not supported
- Loading data into typed tables, or tables with structured type columns, is not supported
- Loading data into declared temporary tables and created temporary tables is not supported
- XML data can only be read from the server side; if you want to have the XML files read from the client, use the import utility
- You cannot create or drop tables in a table space that is in Backup Pending state
- If an error occurs during a LOAD REPLACE operation, the original data in the table is lost. Retain a copy of the input data to allow the load operation to be restarted
- Triggers are not activated on newly loaded rows. Business rules associated with triggers are not enforced by the load utility
- Loading encrypted data is not supported
- When loading into a partitioned table:
  - Loading data into a subset of data partitions while keeping the remaining data partitions fully online is not supported
  - The exception table used by a load operation or a set integrity pending operation cannot be partitioned
  - A unique index cannot be rebuilt when the load utility is running in insert mode or restart mode, and the load target table has any detached dependents

PRESENTATION NOTES FOR PAGE 28

NOTE TO SPEAKER: Default values are shown in blue.

The load utility attempts to deliver the best performance possible by determining optimal values for DISK_PARALLELISM, CPU_PARALLELISM, and DATA BUFFER if these parameters have not be specified by the user. Optimization is based on the size and the free space available in the utility heap. Consider using the autonomic DISK_PARALLELISM and CPU_PARALLELISM settings before attempting to tune these parameters for your particular needs.

SAVECOUNT n - Specifies that the load utility is to establish consistency points after every n rows. This option should be selected if the load operation will be monitored using LOAD QUERY. If the value of n is not sufficiently high, the synchronization of activities performed at each consistency point will impact performance.

STATISTICS USE PROFILE - Collect statistics during the load according using a profile created earlier by the RUNSTAT command. If statistics are invalid for target table then performance will be impacted.

STATISTICS NO - no statistics will be collected; statistics in the catalogs will not be altered. NO is the default. Use RUNSTATS after the LOAD to generate statistics.

CPU_PARALLELISM n - number of processes or threads that the load utility will create.

DATA BUFFER buffer-size - Specifies the number of 4 KB pages (regardless of the degree of parallelism) to use as buffered space for transferring data within the utility. If the value specified is less than the algorithmic minimum, the minimum required resource is used.

SORT BUFFER buffer-size - Override the sortheap database configuration parameter when loading tables with indexes and only when the INDEXING MODE parameter is not specified as DEFERRED. Useful for throttling the sort memory that is used when loading tables with many indexes without changing the value of sortheap, which would also affect general query processing.

DISK_PARALLELISM n

Specifies the number of processes or threads that the load utility will create for writing data to the table space containers.

FETCH_PARALLELISM YES | NO - When performing a load from a cursor, the load utility attempts to parallelize fetching from the remote data source if possible

SOURCEUSEREXIT executable ... PARALLELIZE

Specifies an executable filename which will be called to feed data into the utility.

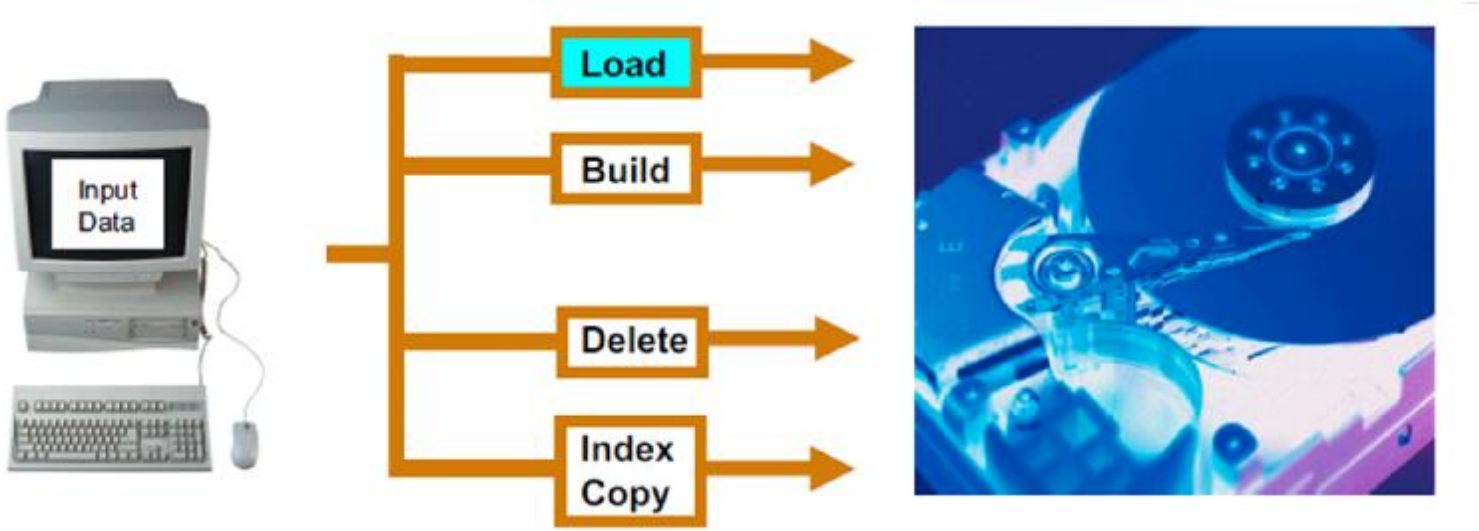
PARALLELIZE increases the throughput of data coming into the load utility by invoking multiple user exit processes simultaneously. This option is only applicable in multi-partition database environments and is ignored in single-partition database environments.

OPEN num-sess SESSIONS

This sub-parameter of the COPY YES parameter specifies the number of I/O sessions to be used with TSM or the vendor product. The default value is 1.

Use the LOAD QUERY Command

§ Checks the status of a load operation during processing and returns the table state. If a load is not processing, then the table state alone is returned



– **Normal**
  • A table is in Normal state if it is not in any of the other (abnormal) table states
– **Set Integrity Pending**
  • The table has constraints which have not yet been verified
– **Load in Progress**
  • This is a transient state that is only in effect during a load operation

PRESENTATION NOTES FOR PAGE 29

When data is being loaded into a column-organized table, the first phase is the ANALYZE phase, which is unique to column-organized tables. The column compression dictionary is built during the ANALYZE phase. This phase is followed by the LOAD, BUILD, and DELETE phases. The INDEX COPY phase applies to row-organized tables only.

Use the LOAD QUERY Command (continued)

§ Load Pending
- A load operation has been active on this table but has been aborted before the data could be committed. Issue a LOAD TERMINATE, LOAD RESTART, or LOAD REPLACE command to bring the table out of this state

§ Read Access Only
- A table is in this state during a load operation if the ALLOW READ ACCESS option was specified

§ Reorg Pending
- A REORG command recommended ALTER TABLE statement has been executed on the table

§ Unavailable
- The table is unavailable. The table can only be dropped or restored from a backup. Rolling forward through a non-recoverable load operation will place a table in the unavailable state

§ Not Load Restartable
- The table is in a partially loaded state that will not allow a load restart operation. The table will also be in load pending state. Issue a LOAD TERMINATE or a LOAD REPLACE command to bring the table out of the not load restartable state

§ Unknown
- The LOAD QUERY command is unable to determine the table state

PRESENTATION NOTES FOR PAGE 30

The INGEST utility

NEW IN DB2 10

§ Features of the ingest utility include:

- **Supports a variety of SQL operations, including array insert, update, merge, replace, and delete**
- **Can use SQL expressions to build individual column values from more than one data field**
- **Commit by time or number of rows.** You can use the **COMMIT_COUNT** ingest configuration parameter to have commit frequency determined by the number of written rows or use the default **COMMIT_PERIOD** ingest configuration parameter to have commit frequency determined by a specified time (**INGEST SET** command)
- **Support for copying rejected records to a file or table, or discarding them.** You can specify what the INGEST command does with rows rejected by the ingest utility (using the **DUMPFILE** parameter) or by DB2 (using the **EXCEPTION TABLE** parameter)
- **Support for restart and recovery.** By default, all INGEST commands are restartable from the last commit point. In addition, the ingest utility attempts to recover from certain errors if you have set the **RETRY_COUNT** ingest configuration parameter

PRESENTATION NOTES FOR PAGE 31

INGEST utility

NEW IN DB2 10

- **High-speed client-side DB2 utility that ingests data from files and pipes into DB2 LUW tables, using SQL-like commands**

- **Advantages:**
  - Move and process large amounts of **real-time data** **without affecting availability**
    - Allows decisions to be based on the latest set of data
    - Increases data analysis capabilities

No need to choose between data concurrency and availability!

PRESENTATION NOTES FOR PAGE 32

The ingest utility (aka CDI) is a high-speed client-side DB2 utility capable of stream data from files and pipes into DB2 target tables.

It can run continually and thus it can process a continuous data stream through pipes

The data is ingested at speeds that are high enough to populate even large tables in partitioned database environments without affecting the data availability. Thus, users do not need to choose between the data currency and availability, and therefore business decisions can be taken based on the latest set of data, leading to an effective business intelligence. This improves your data analysis capabilities to make quicker assessments, which for a business can result in an increase in revenues and maybe lower costs.

Ingest Utility (continued)

§ The performance is just one of the benefits. Lets see some examples:

```
INGEST FROM FILE my_file.txt FORMAT DELIMITED BY '|'
( $prod_ID CHAR(8),
  $description CHAR(32),
  $price DECIMAL(5,2) EXTERNAL,
  $sales_tax DECIMAL(4,2) EXTERNAL,
  $shipping DECIMAL(3,2) EXTERNAL )
INSERT INTO my_table(prod_ID, description, total_price) VALUES
 ($prod_id, $description,
  $price + $sales_tax + $shipping);
```

External fields are used for calculated values!

```
INGEST FROM FILE my_file.txt, my_file2.txt, my_file3.txt
FORMAT DELIMITED
( $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32) )
INSERT INTO my_table VALUES($field1, $field2, $field3);
```

PRESENTATION NOTES FOR PAGE 33

The INGEST command

- **The INGEST command supports the following input data formats:**
  - Delimited text
  - Positional text and binary
  - Columns in various orders and formats
- **In addition to regular tables and nicknames, the INGEST command supports the following table types:**
  - multidimensional clustering (MDC) and insert time clustering (ITC) tables
  - range-partitioned tables
  - range-clustered tables (RCT)
  - materialized query tables (MQTs) that are defined as MAINTAINED BY USER, including summary tables
  - temporal tables
  - updatable views (except typed views)

PRESENTATION NOTES FOR PAGE 34

The INGEST command (continued)

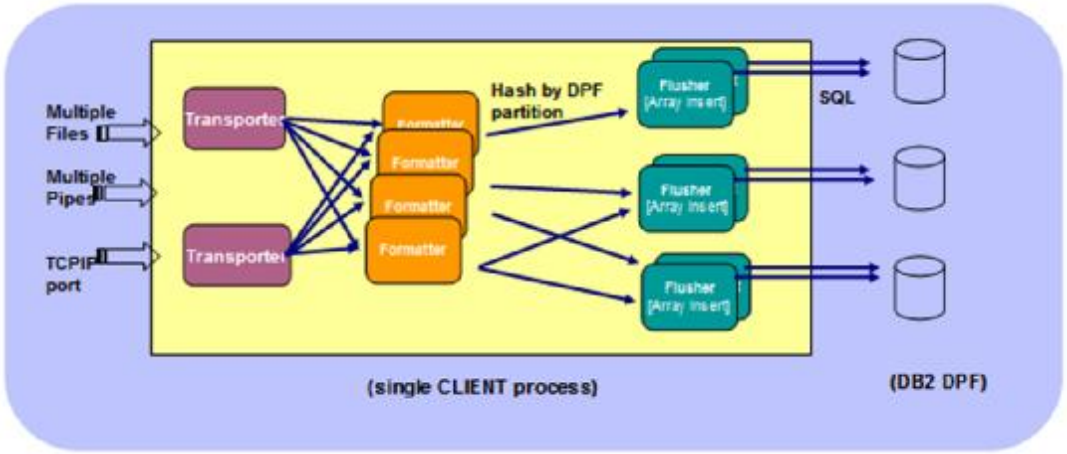
NEW IN DB2 10

§ A single INGEST command goes through three major phases:

§ **Transport**
- The transporters read from the data source and put records on the formatter queues

§ **Format**
- The formatters parse each record, convert the data into the format that DB2 database systems require, and put each formatted record on one of the flusher queues for that record's partition

§ **Flush**
- The flushers issue the SQL statements to perform the operations on the DB2 tables



PRESENTATION NOTES FOR PAGE 35

Transport

The transporters read from the data source and put records on the formatter queues. For INSERT and MERGE operations, there is one transporter thread for each input source (for example, one thread for each input file). For UPDATE and DELETE operations, there is only one transporter thread

Format

The formatters parse each record, convert the data into the format that DB2 database systems require, and put each formatted record on one of the flusher queues for that record's partition. The number of formatter threads is specified by the NUM_FORMATTERS configuration parameter. The default is (number of logical CPUs)/2

Flush

The flushers issue the SQL statements to perform the operations on the DB2 tables. The number of flushers for each partition is specified by the NUM_FLUSHERS_PER_PARTITION configuration parameter. The default is max(1, ((number of logical CPUs)/2)/(number of partitions))

Comparison between the ingest, import, and load utilities

§ The following tables summarize some of the key similarities and differences between the ingest, import, and load utilities

Supported Table Types

Table type	Ingest	Load	Import
Detached table	NS	NS	NS
Global temporary table	NS	NS	NS
Multidimensional clustering (MDC) or insert time clustering (ITC) table	S	S	S
Materialized query table (MQT) that is maintained by user	S	S	S
Nickname	S	NS	S
Range-clustered table (RCT)	S	NS	S
Range-partitioned table	S	S	S
Summary table	S	S	S
Temporal table	S	S	S
Typed table	NS	NS	S
Untyped (regular) table	S	S	S
Updatable view (except typed view)	S	NS	S

PRESENTATION NOTES FOR PAGE 36

There are a number of other important differences that distinguish the ingest utility from the load and import utility:The ingest utility allows the input records to contain extra fields between the fields that correspond to columns.

The ingest utility supports update, delete, and merge.

The ingest utility supports constructing column values from expressions containing field values.

The ingest utility allows other applications to update the target table while ingest is running.

Comparison between the ingest, import, and load utilities

Supported Data Types

Table type	Ingest	Load	Import
Numeric: SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE, DECFLOAT	S	S	S
Character: CHAR, VARCHAR, NCHAR, NVARCHAR, plus corresponding FOR BIT DATA types	S	S	S
Graphic: GRAPHIC, VARGRAPHIC	S	S	S
Long types: LONG VARCHAR, LONG VARGRAPHIC	S	S	S
Date/time: DATE, TIME, TIMESTAMP, including TIMESTAMP(p)	S	S	S
DB2SECURITYLABEL	S	S	S
LOBs from files: BLOB, CLOB, DBCLOB, NCLOB	NS	S	S
Inline LOBs	NS	S	S
XML from files	NS	S	S
Inline XML	NS	S	S
Distinct type	S (if based on a S built-in data type)	S	S
Structured type	NS	NS	S
Reference type	S	S	S

PRESENTATION NOTES FOR PAGE 37

Comparison between the ingest, import, and load utilities

- The ingest utility supports update, delete, and merge
- The ingest utility supports constructing column values from expressions containing field values
- The ingest utility allows other applications to update the target table while ingest is running

PRESENTATION NOTES FOR PAGE 38

Table data can be moved online or offline using the ADMIN_MOVE_TABLE procedure

- The ADMIN_MOVE_TABLE stored procedure moves the data in an active table into a new table object with the same name, while the data remains online and available for select, insert, update, and delete access
- You can also generate a compression dictionary when a table is moved
- **Method 1:** modify only certain parts of the table definition for the target table
  - For instance, if you had a table definition that is quite large, and all you want to do is modify the table spaces for the table, you can do so without having to determine the entire CREATE TABLE statement needed to re-create the source table. All you need to do is to fill out the data_tbsp, index_tbsp, and lob_tbsp parameters, leaving the other optional parameters blank
- **Method 2:** more control and flexibility by allowing you to create the target table beforehand, rather than having the stored procedure create the target table. This enables you to create a target table that would not be possible using the first method

PRESENTATION NOTES FOR PAGE 39

You can now call the ADMIN_MOVE_TABLE stored procedure to move the data in a table to a new table object of the same name (but with possibly different storage characteristics) while the data remains online and available for access. You can also generate a new optimal compression dictionary when a table is moved.

Here's a brief overview of some of the parameters that impact the stored procedure's performance.

COPY_USE_LOAD – increases performance by using db2Load API. When using the COPY_USE_LOAD option, it is necessary to perform a backup of the target tablespace(s) before the SWAP phase in order to ensure recoverability.

COPY_WITH_INDEXES – reduces copy performance because indexes must be built

REORG: This option sets up an extra offline REORG on the target table before performing the swap. If you use this option to improve your compression dictionary, be advised that using the default sampling approach is a better method to create an optimal compression dictionary. However, if you require an optimal XML compression dictionary, then REORG is the only method.

The default behavior when performing a table move on a table where statistics are collected is to perform RUNSTATS on the table during the SWAP phase. If a statistics profile is found, RUNSTATS will be called using the statistics profile.

NO_STATS: Does not call RUNSTATS. DB2 will automatically create new statistics afterwards if you use the AUTO_RUNSTATS or AUTO_STMT_STATS database configuration parameters.

COPY_STATS: This option copies the statistics from the source table to the target table before performing the swap. This may cause inaccurate physical statistics, especially if the page size is changed. However, setting this option saves computing time as RUNSTATS is not called to compute new statistics.

The ADMIN_MOVE_TABLE_UTIL procedure modifies the performance characteristics of the ADMIN_MOVE_TABLE procedure

- **ADMIN_MOVE_TABLE_UTIL** procedure “key” parameter values:
  - COMMIT_AFTER_N_ROWS
  - DEEPCOMPRESSION_SAMPLE
  - COPY_ARRAY_SIZE
  - REORG_USE_TEMPSPACE

PRESENTATION NOTES FOR PAGE 40

The ADMIN_MOVE_TABLE procedure creates a shadow copy of the table. During the copy phase, insert, update, and delete operations against the original table are captured using triggers and placed into a staging table. After the copy phase has completed, the data change operations that were captured in the staging table are replayed to the shadow copy. The copy of the table includes all table options, indexes, and views. The procedure then briefly takes the table offline to swap the object names.

You can modify the performance characteristics of the ADMIN_MOVE_TABLE stored procedure with the ADMIN_MOVE_TABLE_UTIL stored procedure.

ADMIN_MOVE_TABLE_UTIL “key” parameters that impact performance:

COMMIT_AFTER_N_ROWS: During the COPY phase, a commit is executed after this many rows are copied. A value of 0 means no commits are executed during COPY.

DEEPCOMPRESSION_SAMPLE: If the source table has compression enabled, this field specifies how much data (in KB) is sampled when creating a dictionary for compression. A value of 0 means no sampling is done.

COPY_ARRAY_SIZE: Specifies the ARRAY size for COPY_ARRAY_INSERT, a value less than or equal to 0 means do not use COPY_ARRAY_INSERT.

REORG_USE_TEMPSPACE: If you call the REORG option in the table move, you can also specify a temporary table space for the USE clause of the REORG command. If a value is not specified here, the REORG command uses the same table space as the table being reorganized.

The ADMIN_COPY_SCHEMA procedure

- The **ADMIN_COPY_SCHEMA** procedure is used to copy a specific schema and all objects contained in it
- The new target schema objects will be created using the same object names as the objects in the source schema, but with the target schema qualifier. The ADMIN_COPY_SCHEMA procedure can be used to copy tables with or without the data of the original tables

Syntax

```
>>-ADMIN_COPY_SCHEMA--(--sourceschema--,--targetschema--,--------->
>--copymode--,--objectowner--,--sourcetbsp--,--targettbsp--,------>
>--errortabschema--,--errortab--)------------------------------><
```

PRESENTATION NOTES FOR PAGE 41

Restrictions

Only DDL copymode is supported for HADR databases.

XML with COPY or COPY NO is not supported.

Using the ADMIN_COPY_SCHEMA procedure with the COPYNO option places the table spaces in which the target database object resides in backup pending state. After the load operation completes, target schema tables are in set integrity pending state, and the ADMIN_COPY_SCHEMA procedure issues a SET INTEGRITY statement to get the tables out of this state. Because the table spaces are already in backup pending state, the SET INTEGRITY statement fails. For information about how to resolve this problem, see "Copying a schema".

Usage notes

References to fully qualified objects within the objects being copied will not be modified. The ADMIN_COPY_SCHEMA procedure only changes the qualifying schema of the object being created, not any schema names that appear within SQL expressions for those objects. This includes objects such as generated columns and trigger bodies.

This procedure does not support copying the following objects:

index extensions

nicknames

packages

typed tables

array types

user-defined structured types (and their transform functions)

typed views

jars (Java routine archives)

staging tables

aliases with base objects that do not belong to the same source schema

If one of these objects exists in the schema being copied, the object is not copied but an entry is added to the error table indicating that the object has not been copied.

When a replicated table is copied, the new copy of the table does not have subscriptions enabled. The table is re-created as a basic table only.

The operation of this procedure requires the existence of the SYSTOOLSPACE table space. This table space is used to hold metadata used by the ADMIN_COPY_SCHEMA procedure as well as error tables returned by this procedure. If the table space does not exist, an error is returned.

Statistics for the objects in the target schema are set to default.

If a table has a generated identity column, and copymode is either 'COPY' or 'COPYNO', the data values from the source table are preserved during the load.

A new catalog entry is created for each external routine, referencing the binary of the original source routine.

If a table is in set integrity pending state at the beginning of the copy operation, the data is not loaded into the target table and an entry is logged in errortab indicating that the data was not loaded for that table.

If a Load or DDL operation fails, an entry is logged in errortab for any object that was not created. All objects that are successfully created remain. To recover, a manual load can be initiated, or the new schema can be dropped using the ADMIN_DROP_SCHEMA procedure and the ADMIN_COPY_SCHEMA procedure can be called again.

During DDL replay, the default schema is overridden to the target schema if it matches the source schema.

The function path used to compile a trigger, view or SQL function is the path used to create the source object, with the following exception: if the object's function path contains the source schema name, this entry in the path is modified to the target schema name during DDL replay.

Running multiple ADMIN_COPY_SCHEMA procedures will result in deadlocks. Only one ADMIN_COPY_SCHEMA procedure call should be issued at a time. Changes to tables in the source schema during copy processing might mean that the data in the target schema is not identical following a copy operation.

Careful consideration should be taken when copying a schema with tables from a table space in a single-partition database partition group to a table space in a multiple-partition database partition group. Unless automatic distribution key selection is preferred, the distribution key should be defined on the tables before the copy schema operation is undertaken. Altering the distribution key can only be done to a table whose table space is associated with a single-partition database partition group.

The ADMIN_COPY_SCHEMA procedure (continued)

- ADMIN_MOVE_TABLE_UTIL procedure “key” parameter values:
  - SOURCESCHEMA
  - TARGETSCHEMA
  - COPYMODE
  - OBJECTOWNER
  - SOURCETBSP
  - TARGETTBSP
  - ERRORTABSCHEMA
  - ERRORTAB

PRESENTATION NOTES FOR PAGE 42

sourceschema

An input argument of type VARCHAR(128) that specifies the name of the schema whose objects are being copied. The name is case-sensitive.

targetschema

An input argument of type VARCHAR(128) that specifies a unique schema name to create the copied objects into. The name is case-sensitive. If the schema name already exists, the procedure call will fail and return a message indicating that the schema must be removed before invoking the procedure.

copymode

An input argument of type VARCHAR(128) that specifies the mode of copy operation. Valid options are:

'DDL': create empty copies of all supported objects from the source schema.

'COPY': create empty copies of all objects from the source schema, then load each target schema table with data. Load is done in 'NONRECOVERABLE' mode. A backup must be taken after calling the ADMIN_COPY_SCHEMA, otherwise the copied tables will be inaccessible following recovery.

'COPYNO': create empty copies of all objects from the source schema, then load each target schema table with data. Load is done in 'COPYNO' mode.

Note: If copymode is 'COPY' or 'COPYNO', a fully qualified filename, for example 'COPYNO /home/mckeough/loadoutput', can be specified along with the copymode parameter value. When a path is passed in, load messages will be logged to the file indicated. The file name must be writable by the user ID used for fenced routine invocations on the instance. If no path is specified, then load message files will be discarded (default behavior).

objectowner

An input argument of type VARCHAR(128) that specifies the authorization ID to be used as the owner of the copied objects. If NULL, then the owner will be the authorization ID of the user performing the copy operation.

sourcetbsp

An input argument of type CLOB(2 M) that specifies a list of source table spaces for the copy, separated by commas. Delimited table space names are supported. For each table being created, any table space found in this list, and the tables definition, will be converted to the nth entry in the targettbsp list. If NULL is specified for this parameter, new objects will be created using the same table spaces as the source objects use.

targettbsp

An input argument of type CLOB(2 M) that specifies a list of target table spaces for the copy, separated by commas. Delimited table space names are supported. One table space must be specified for each entry in the sourcetbsp list of table spaces. The nth table space in the sourcetbsp list will be mapped to the nth table space in the targettbsp list during DDL replay. It is possible to specify 'SYS_ANY' as the final table space (an additional table space name, that does not correspond to any name in the source list). When 'SYS_ANY' is encountered, the default table space selection algorithm will be used when creating objects (refer to the IN tablespace-name1 option of the CREATE TABLE statement documentation for further information about the selection algorithm). If NULL is specified for this parameter, new objects will be created using the same table spaces as the source objects use.

errortabschema

An input and output argument of type VARCHAR(128) that specifies the schema name of a table containing error information for objects that could not be copied. This table is created for the user by the ADMIN_COPY_SCHEMA procedure in the SYSTOOLSPACE table space. If no errors occurred, then this parameter is NULL on output.

errortab

An input and output argument of type VARCHAR(128) that specifies the name of a table containing error information for objects that could not be copied. This table is created for the user by the ADMIN_COPY_SCHEMA procedure in the SYSTOOLSPACE table space. This table is owned by the user ID that invoked the procedure. If no errors occurred, then this parameter is NULL on output. If the table cannot be created or already exists, the procedure operation fails and an error message is returned. The table must be cleaned up by the user following any call to the ADMIN_COPY_SCHEMA procedure; that is, the table must be dropped in order to reclaim the space it is consuming in SYSTOOLSPACE.

DB2 data recovery utilities can be integrated with 3rd party software and specific hardware technologies

§ BACKUP
- To disk
- To 3rd party software (Tivoli Storage Manager, etc.)
- Full database backup or table space backups
- Incremental backups

§ Snapshot backup (DB2 Advanced Copy Services)
- Used to take fast snapshots of database that can be used as:
  - Hot standby
  - Mirror copy to be used as backup image
  - Clone of primary database to be used for read activity

§ RECOVER:
- RECOVER DATABASE – performs following actions under the cover:
  - RESTORE DATABASE (New: can now take advantage of capability to redirect to specific storage paths!)
  - ROLLFORWARD DATABASE
- Crash Recovery

PRESENTATION NOTES FOR PAGE 43

DB2 provides the ability to backup, restore and recover databases while online (hot) or offline (cold). You can perform full database backups (offline) as well as online incremental and delta backups. Backup and restore can be to/from disk, tape or 3rd party software.

The RECOVER command can perform both a restore and rollforward.

We'll cover backup, restore and recover in the next set of slides.

DB2 always starts in crash recovery mode. By this, we mean that DB2 checks to see if the database objects are accessible and current. If they are not current, DB2 uses its log files to bring the database to a current state. If objects are not accessible, which might occur if a disk is offline for some reason, DB2 will notify you that it cannot start the database. No performance optimization techniques are available for this feature.

The DB2 InfoCenter discusses the best practices for DB2 Advanced Copy Services (ACS). We won't cover ACS in this presentation, but did want you to be aware that it is available for specific hardware and software configurations.

FOR SPEAKER, in case you are asked.

You need specific hardware, either IBM or NetAPP, to use ACS.

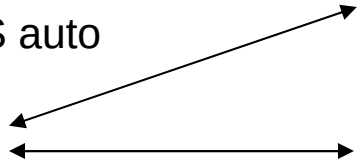
Volume sharing is not supported. If a database partition resides on the same storage volume as any other database partition, snapshot operations are not permitted.

Log paths be contained within their own snapshot volume independent from the database directory and database containers.

In a database partitioning feature (DPF) environment, each database partition must reside on a set of snapshot volumes independent of the other database partitions.

DB2 automatically configures BACKUP performance parameters based on system resources and database design

§ Optimization parameters
- PARALLELISM *n auto.*
- WITH *num-buffers* BUFFERS auto
- BUFFER *buffer-size auto*
- UTIL_IMPACT_PRIORITY



Set the UTIL_IMPACT_LIM database manager configuration parameter.

Modify priority with the SET UTIL_IMPACT_PRIORITY command.

§ Optimization techniques
- Increase the value of the PARALLELISM so that it reflects the number of table spaces being backed up auto
- Increase the backup buffer size auto
- Increase the number of buffers auto
- Increase the utility heap size (UTIL_HEAP_SZ (causes DB2 to increase the 3 parameters above)
- Specify the table space backup option on the BACKUP DATABASE command.
- Use multiple target devices
- Do not overload the I/O device controller bandwidth

PRESENTATION NOTES FOR PAGE 44

On this page and the next, you need to find a way to clarify what is automatically configured by DB2, and what must be optionally configured (or left alone) by the DBA. Your title says “automatically”, but in both bullets below, you are mixing automatic and manual options. I have indicated in red by each bullet what is automatic. Please find a way to clearly explain to the students what is automatic and what is not.

When you perform a backup operation, DB2 automatically chooses an optimal value for the number of buffers, the buffer size and the parallelism settings. The values are based on the amount of utility heap memory available, the number of processors available, and the database configuration. Therefore, depending on the amount of storage available on your system, you should consider allocating more memory by increasing the UTIL_HEAP_SZ configuration parameter. The objective is to minimize the time it takes to complete a backup operation.

If you do not specify the number of buffers and the buffer size, DB2's values should have minimal effect on large databases. However, for small databases, it can cause a large percentage increase in backup image size. Even if the last data buffer written to disk contains little data, the full buffer is written to the image anyway. In a small database, this means that a considerable percentage of the image size might be empty.

The following apply for both backup and restore operations:

- Multiple devices should be used.
- Do not overload the I/O device controller bandwidth.

Note that the BACKUP command is one of the two utilities that can be throttled.

The impact policy for throttled utilities is defined by default with the the UTIL_IMPACT_LIM database manager configuration parameter.

If you want to run a throttled utility you must set the impact priority using the UTIL_IMPACT_PRIORITY parameter.

DB2 automatically configures RESTORE performance parameters based on system resources and database design

§ Optimization parameters
- PARALLELISM *n auto*
- WITH *num-buffers* BUFFERS auto
- BUFFER *buffer-size auto*

§ Optimization techniques
- Increase the value of the PARALLELISM parameter auto
- Increase the restore buffer size auto
- Increase the number of buffers auto
- Increase the utility heap size (UTIL_HEAP_SZ) (causes DB2 to increase the 3 parameters above)
- For tables with large amount of long fields and LOBs, store the LOBs in a separate table space
  - Use NOT LOGGED option for the LOB data where possible
- Use multiple target devices
- Do not overload the I/O device controller bandwidth

PRESENTATION NOTES FOR PAGE 45

When you perform a restore operation, DB2 will automatically choose an optimal value for the number of buffers, the buffer size and the parallelism settings. The values will be based on the amount of utility heap memory available, the number of processors available and the database configuration. Therefore, depending on the amount of storage available on your system, you should consider allocating more memory by increasing the UTIL_HEAP_SZ configuration parameter. The objective is to minimize the time it takes to complete a restore operation.

For restore operations, a multiple of the buffer size used by the backup operation will always be used. You can specify a buffer size when you issue the RESTORE DATABASE command but you need to make sure that it is a multiple of the backup buffer size.

You can also choose to do any of the following to reduce the amount of time required to complete a restore operation:

* Increase the restore buffer size.
* Increase the number of buffers.
* Increase the value of the PARALLELISM parameter.
* Increase the utility heap size

The following apply for both backup and restore operations:

- Multiple devices should be used.
- Do not overload the I/O device controller bandwidth.

The RECOVER DATABASE command does not provide any parameters to modify its performance

- Optimization parameters
  - None. DB2 will try to optimize the RESTORE operation (see previous slide)
- Optimization techniques
  - Increase the utility heap size (UTIL_HEAP_SZ).
  - Place the logs on a separate device
  - Use multiple source devices
  - Use multiple target devices
  - Do not overload the I/O device controller bandwidth
  - Restore selected table spaces if they contain large amounts of long field and LOB data

PRESENTATION NOTES FOR PAGE 46

The following should be considered when thinking about recovery performance:

DB2 automatically attempts to minimize the time it takes to complete a backup or restore operation by choosing an optimal value for the number of buffers, the buffer size and the parallelism settings. The values are based on the amount of utility heap memory available, the number of processors available and the database configuration. Depending on the amount of storage available on your system, you should consider allocating more memory by increasing the UTIL_HEAP_SZ configuration parameter.

You can improve performance for databases that are frequently updated by placing the logs on a separate device. In the case of an online transaction processing (OLTP) environment, often more I/O is needed to write data to the logs than to store a row of data. Placing the logs on a separate device will minimize the disk arm movement that is required to move between a log and the database files.

The following apply for both backup and restore operations: Multiple devices should be used; Do not overload the I/O device controller bandwidth.

* If a table contains large amounts of long field and LOB data, restoring it could be very time consuming. If the database is enabled for rollforward recovery, the RESTORE command provides the capability to restore selected table spaces. If the long field and LOB data is critical to your business, restoring these table spaces should be considered against the time required to complete the backup task for these table spaces. By storing long field and LOB data in separate table spaces, the time required to complete the restore operation can be reduced by choosing not to restore the table spaces containing the long field and LOB data. If the LOB data can be reproduced from a separate source, choose the NOT LOGGED option when creating or altering a table to include LOB columns. If you choose not to restore the table spaces that contain long field and LOB data, but you need to restore the table spaces that contain the table, you must roll forward to the end of the logs so that all table spaces that contain table data are consistent.

DB2 provides several utilities that can be used to optimize online, active data

- Optimizing data
  - REORGCHK
  - REORG
  - RUNSTATS

PRESENTATION NOTES FOR PAGE 47

DB2 provides the REORGCHK, REORG AND RUNSTATS commands to help you maintain the organization of data on disk and statistics about the data which can be used by the DB2 Optimizer. Using these utilities helps optimize the performance of your applications.

The REORGCHK command is used to identify tables and indexes that could benefit from a REORG

- This command can be issued from any database partition in the **db2nodes.cfg** file. It can be used to update table and index statistics in the catalogs
- Evaluation of tables and indexes can be based on the current statistics or the statistics can be updated prior to the check
- The tables and/or indexes which have been identified as needing a REORG will show one or more asterisks in the reorg columns of the REORGCHK output

PRESENTATION NOTES FOR PAGE 48

REORGCHK is used to identify those tables and indexes that are in need of a REORG and can also be used to collect statistics for all the tables in the database.

REORGCHK to determine if reorganization is required

```
Table statistics:

F1: ... < 5
F2: ... > 70
F3: ... > 80

SCHEMA.NAME                    ...   F1  F2  F3 REORG
-----------------------------------------------------
Table: DB2INST1.XMLFILES       ...    0  89  98 ---
-----------------------------------------------------

Index statistics:

F4: ... > 80
F5: ... > MIN(50, (100 - PCTFREE))
F6: ... < 100
F7: ... < 20
F8: ... < 20

SCHEMA.NAME                    ... PCT_PAGES_SAVED  F4  F5  F6  F7  F8 REORG
------------------------------ ... ---------------------------------------
Table: DB2INST1.XMLFILES
Index: DB2INST1.IDX1           ...              50  98  82   -   0   0 -----
Index: DB2INST1.IDX2           ...               0  71   -   -   0   0 *----
Index: DB2INST1.IDX3           ...              33  98 220   -   0   0 -----
-----------------------------------------------------------------------------
```

Does not meet evaluation criteria and would benefit from a REORG

PRESENTATION NOTES FOR PAGE 49

We don't include the entire output of the REORGCHK command here. It is quite detailed on voluminous.

There are two basic sections of the output. The first is for the table. It contains Formulas F1, F2 and F3. If the analysis results are less than the formula requires then the table data would benefit from a REORG.

Formulas are not included in the output, but the metric each is compared against is shown. Much of the column data has been removed since we're concerned with whether or not we have a table or index that might need a REORG. This information is always at the end of the line.

In this example the index DB2INST1.IDX2 for table DB2INST1.XMLFILES does not satisfy the metrics for formula 4.

The description of the REORGCHK command in the DB2 documentation tells you which parameters to specify when using the REORG command, based on the results of combinations of these formulas. Executing only the required maintenance reduces the impact of the REORG command on running applications.

The REORG command should be run after there have been significant changes

- Reclaims space from deleted data and places data into physically contiguous pages
- Can order the data in a table in physical sequence according to a specific index
- Can rebuild skewed indexes
- Improve the availability using ALLOW READ|WRITE|NO ACCESS
- New clause in REORG INDEX to reclaim storage (RECLAIM EXTENTS clause)
- Use ADMIN_GET_TABLESPACE and ADMIN_GET_INDEX_INFO to validate reiclamable storage
- Smart index and data prefetching
  - **This new and exciting feature** not only **improves query peformance, it reduces the need to reorganize tables and indexes**
  - **Sequential detection prefetching is used when the data pages are stored sequentially, and readahead prefetching is used when the data pages are poorly clustered**

PRESENTATION NOTES FOR PAGE 50

The REORG command is used to reorganize (defragment) an index or a table. Running a reorg for a table reconstructs, or rebuilds the rows, compacting the table by eliminating fragmented data. It can also order a table according to a specific index sequence (e.g., in support of a clustered index) and can be performed online and paused. In addition, the REORG command can rebuild the data compression dictionary and recompress the data accordingly.

When an indexed table has been modified many times, the data in the indexes might become fragmented. If the table is clustered with respect to an index, the table and index can get out of cluster order. Both of these factors can adversely affect the performance of scans using the index, and can impact the effectiveness of index page prefetching. REORG INDEX or REORG INDEXES can be used to reorganize one or all of the indexes on a table. Index reorganization will remove any fragmentation and restore physical clustering to the leaf pages. Use the REORGCHK command to help determine if an index needs reorganizing. Be sure to complete all database operations and release all locks before invoking index reorganization. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.

A classic table reorganization (offline reorganization) rebuilds the indexes during the last phase of the reorganization. However, the inplace table reorganization (online reorganization) does not rebuild the indexes. It is recommended that you issue a REORG INDEXES command after the completion of an inplace table reorganization. An inplace table reorganization is asynchronous, therefore care must be taken to ensure that the inplace table reorganization is complete before issuing the REORG INDEXES command.

You can reduce the impact to running applications and reduce the amount of time needed fro RUNSTATS if you refer to the output of the REORGCHK to determine which parameters should be used.

The RUNSTATS command

- Collects statistics on tables, indexes and statistical views for the optimizer to use when creating an access plan
- Can register and use a statistics profile
- The RUNSTATS command can be issued:
  - From any database partition in the db2nodes.cfg file. It can be used to update the catalogs on the catalog database partition
- For tables:
  - This command collects statistics for a table on the database partition from which it is invoked. If the table does not exist on that database partition, the first database partition in the database partition group is selected
- For views:
  - This command collects statistics using data from tables on all participating database partitions
- Can register and use a statistics profile

PRESENTATION NOTES FOR PAGE 51

The RUNSTATS command updates statistics about the characteristics of a table and/or associated indexes, or statistical views. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

For a table, this utility should be called when the table has had many updates, or after reorganizing the table.

For a statistical view, this utility should be called when changes to underlying tables have substantially affected the rows returned by the view. The view must have been previously enabled for use in query optimization using the ALTER VIEW statement.

The runstats utility also provides the option to register and use a statistics profile, which specifies the type of statistics that are to be collected for a particular table; for example, table statistics, index statistics, or distribution statistics. This feature simplifies statistics collection by enabling you to store runstats options for convenient future use.

The System Catalog tables also benefit from having RUNSTATS performed on them.

In order for the RUNSTATS utility to be throttled,

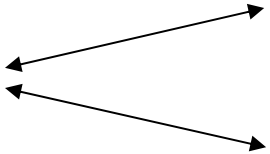
The impact policy for throttled utilities is defined by default with the UTIL_IMPACT_LIM database manager configuration parameter.

If you want to run a throttled utility you must set the impact priority using the UTIL_IMPACT_PRIORITY parameter of the RUNSTATS or BACKUP command.

RUNSTATS is a utility that can be throttled

§ Optimization parameters

– UTIL_IMPACT_PRIORITY



Set the **UTIL_IMPACT_LIM** database manager configuration parameter.

Modify priority with the SET **UTIL_IMPACT_PRIORITY** command.

§ Optimization techniques

– Limit statistics collection to the set of columns used in predicates
– Limit statistics collection to “KEY” columns
– Use the TABLESAMPLE option to collect statistics on a subset of the table data
– Use the SAMPLED option if there are many indexes on the table and DETAILED (extended) information on the indexes might improve access plans
– Do not specify high NUM_FREQVALUES or NUM_QUANTILES values for columns that are not used in predicates

§ New in DB2 10

– The default for DETAILED index, is sampled
– Sampled index statistics are collected
– There is no need to specify the schema for the table
– The VIEW parameter is included for statistical views, instead of using TABLE
– New auto_sampling parameter for automatic sampled statistics when auto_runstats is ON

PRESENTATION NOTES FOR PAGE 52

The RUNSTATS command updates statistics about the characteristics of a table and/or associated indexes, or statistical views. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

For a table, this utility should be called when the table has had many updates, or after reorganizing the table.

For a statistical view, this utility should be called when changes to underlying tables have substantially affected the rows returned by the view. The view must have been previously enabled for use in query optimization using the ALTER VIEW statement.

The runstats utility also provides the option to register and use a statistics profile, which specifies the type of statistics that are to be collected for a particular table; for example, table statistics, index statistics, or distribution statistics. This feature simplifies statistics collection by enabling you to store runstats options for convenient future use.

The System Catalog tables also benefit from having RUNSTATS performed on them.

In order for the RUNSTATS utility to be throttled,

The impact policy for throttled utilities is defined by default with the UTIL_IMPACT_LIM database manager configuration parameter.

If you want to run a throttled utility you must set the impact priority using the UTIL_IMPACT_PRIORITY parameter of the RUNSTATS or BACKUP command.

Agenda

- Features that impact utility performance
- Maintenance commands, utilities, and stored procedures
- **Automating maintenance**

PRESENTATION NOTES FOR PAGE 53

Automatic maintenance is enabled for a database when it is created

```
Database Configuration for Database
...
 Automatic maintenance                       (AUTO_MAINT) = ON
   Automatic database backup                 (AUTO_DB_BACKUP) = OFF
   Automatic table maintenance               (AUTO_TBL_MAINT) = ON
     Automatic runstats                       (AUTO_RUNSTATS) = ON
       Real-time statistics                  (AUTO_STMT_STATS) = ON
       Statistical views                     (AUTO_STATS_VIEWS) = OFF
       Automatic Sampling                      (AUTO_SAMPLING) = OFF
     Automatic statistics profiling          (AUTO_STATS_PROF) = OFF
       Statistics profile updates              (AUTO_PROF_UPD) = OFF
     Automatic reorganization                    (AUTO_REORG) = OFF
```

PRESENTATION NOTES FOR PAGE 54

The database manager provides automatic maintenance capabilities for performing database backups, keeping statistics current, and reorganizing tables and indexes as necessary. Some components are enabled by default, as shown in these excerpt from the GET DATABASE CONFIGURATION command.

The hierarchy shown here allows you to disable or enable related parameters. If Automatic maintenance is OFF, no automatic maintenance will be run, even if the child's value is set to ON. Accordingly, If Automatic table maintenance is OFF, none of its child parameters will run.

You can define automatic maintenance windows and customize the maintenance utilities

- Sample policies are in ~/sqllib/samples/automaintcfg
  - DB2MaintenanceWindowPolicySample.xml
  - DB2AutoRunstatsPolicySample.xml
  - DB2AutoReorgPolicySample.xml
  - DB2AutoBackupPolicySample.xml
- Define windows for both online and offline maintenance
- Automatic maintenance runs only if the requirements in the policy are met
- Maintenance utilities will run to completion if they are started
- Optimization parameters
  - None

PRESENTATION NOTES FOR PAGE 55

Performing maintenance activities on your databases is essential in ensuring that they are optimized for performance and recoverability.

A maintenance window is a time period that you define for the running of automatic maintenance activities, which are backup, statistics collection, statistics profiling, and reorganization. An offline window might be the time period when access to a database is unavailable. An online window might be the time period when users are permitted to connect to a database.

A maintenance window is different from a task schedule. During a maintenance window, each automatic maintenance activity is not necessarily run. Instead, the database manager evaluates the system to determine the need for each maintenance activity to be run. If the maintenance requirements are not met, the maintenance activity is run. If the database is already well maintained, the maintenance activity is not run.

The backup policy for a database is created automatically when the DB2 Health Monitor first runs.

Automatic statistics can be collected during maintenance windows or real-time

- Optimization parameters:
  - Use auto_sampling for a less intrusive auto maintenance system load
- Procedure:
  - Set the auto_maint and the auto_tbl_maint database configuration parameters to ON. This is the default
  - To enable real-time statistics collection, set both auto_stmt_stats and auto_runstats database configuration parameters to ON. This is the default
  - To enable background statistics collection, set the auto_runstats database configuration parameter to ON. This is the default
  - To enable automatic statistics profile generation, set auto_prof_upd database configuration parameters to ON. If the auto_runstats database configuration parameter is also set to ON, statistics are collected automatically using the generated profiles

PRESENTATION NOTES FOR PAGE 56

Having accurate and complete database statistics is critical to efficient data access and optimal workload performance. Use the automatic statistics collection feature of the automated table maintenance functionality to update and maintain relevant database statistics.

You can enhance this functionality in environments where a single database partition operates on a single processor by collecting query data and generating statistics profiles that help the DB2 server to automatically collect the exact set of statistics that is required by your workload. This option is not available in partitioned database environments, certain federated database environments, or environments in which intra-partition parallelism is enabled.

automatic statistics , this should not be used in production environments unless performance problems point to bad statistics.

Summary

§ In this Module you learned about:
- DB2 features that impact DB2 utility performance
- Performance tuning for DB2 maintenance utilities
- DB2 automating maintenance

PRESENTATION NOTES FOR PAGE 57

The next steps…

PRESENTATION NOTES FOR PAGE 58

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 10306_WB1_DB2 Mtaintenace_Utilities_and_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

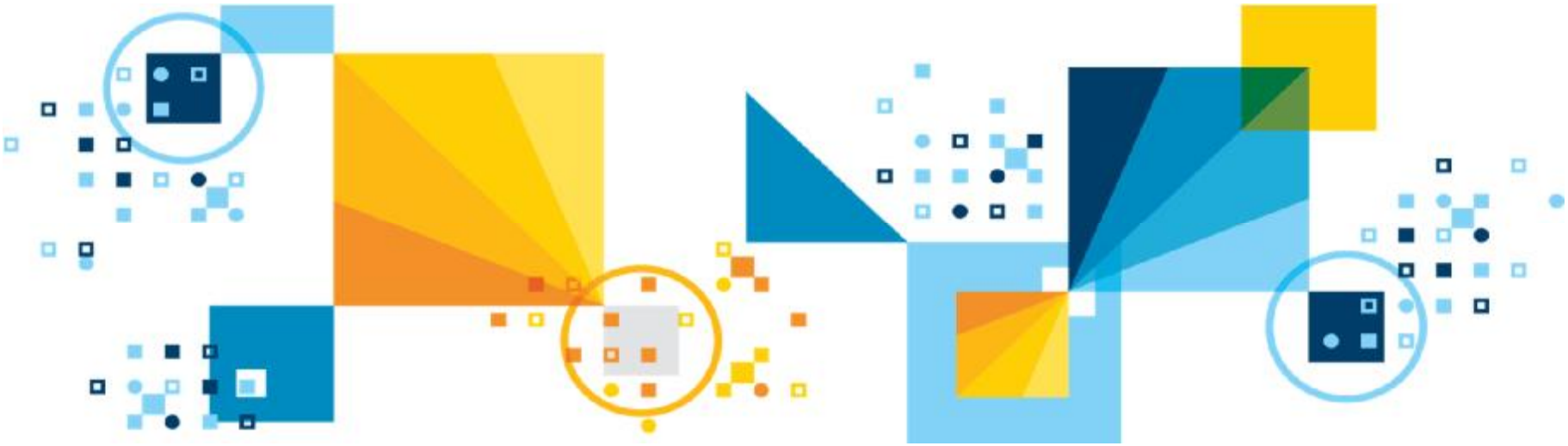
PRESENTATION NOTES FOR PAGE 59

The Next Steps…

- The next set of modules to consider :
  - This is the last module of Curriculum, Performance Monitoring and Tuning for DB2 10.5 with BLU Acceleration for LUW.

PRESENTATION NOTES FOR PAGE 60

Questions?
askdata@ca.ibm.com



PRESENTATION NOTES FOR PAGE 61