DB2 9.7 Fix Packs are stuffed full of new enhancements to help you maximize performance, recoverability, security and operational management. Get the technical information about the new functions and features available in DB2 9.7 Fix Packs for Linux, UNIX, and Windows. Melanie will discuss the advantages and potential impact of new Fix Packs of DB2 9.7 to your environment.

Objectives

- Plan the installation and migration to DB2 9.7 Fix Packs
- Describe the new functions and options for administration of DB2 databases provided by new DB2 9.7 Fix Packs
- Assess the applicability of the DB2 9.7 Fix Packs features and functions in your application environment to maximize the performance, recoverability, security and operational management
- Discuss the advantages of the new Fix Packs of DB2 to your environment.
- Discuss the potential impact of the new Fix Packs of DB2.

1

Objective 1: Plan the installation and migration to DB2 9.7 Fix Packs.
Objective 2: Describe the new functions and options for administration of DB2 databases provided by new DB2 9.7 Fix Packs.
Objective 3: Assess the applicability of the DB2 9.7 Fix Packs' features and functions in your application environment to maximize the performance, recoverability, security and operational management.
Objective 4: Discuss the advantages of the new Fix Packs of DB2 to your environment.

Objective 5: Discuss the potential impact of the new Fix Packs of DB2.

DB2 9.7 Fix Pack 1 Checklist

Changes to Existing Functionality:

- db2rfpen command -file option is deprecated
- Detach operation for data partitions has changed
- XML schema maxOccurs attribute values greater than 5000 are parsed differently
- Collection interval time for workload management statistics has changed

New Enhancements:

- Read operations on HADR standby databases are supported
- DB2 Advanced Copy Services (ACS) supported on AIX 6.1
- Last referenced date available for tables, table partitions, indexes, and packages
- SUBSTRB scalar function
- SQL PL functionality extended for user-defined functions
- Global variable assignments in nested contexts are supported
- User-defined functions support OUT and INOUT parameters
- CREATE FUNCTION statement (PL/SQL) to modify database
- IBM Data Server Provider for .NET has been enhanced
- Fenced routines history information easier to collect
- PL/SQL compiler supports FORALL and BULK COLLECT INTO syntax
- XQuery functions to retrieve date and time values for local time zones
- Diagnostic data can be stored in separate directories
- New event monitor for dynamic and static SQL statements in package cache
- New relational monitoring interfaces for locking events
- Explain enhanced with actual values for operator cardinality
- Statements from a runtime section can be explained
- Time-spent monitor elements are more comprehensive
- Table functions for row-based formatting of monitoring information

Version 9.7 Fix Pack 1 contains important changes that might affect your product usage. You should review the technical changes and new functionality included in Version 9.7 Fix Pack 1. For a summary of DB2 Version 9.7 LUW fix pack 1 changes, please see:

http://www-01.ibm.com/support/docview.wss?uid=swg24023357&myns=swgimgmt&mynp=OCSSEPGG&mynp=OCSSEPDU&mync=E

http://www-01.ibm.com/support/docview.wss?uid=swg24023357&myns=swgimgmt&mynp=OCSSEPGG&mynp=OCSSEPDU&mync=E#Description

DB2 9.7 Fix Pack 1 Checklist, continued

New Enhancements, continued:

- MON_GET_PKG_CACHE_STMT_DETAILS retrieves info in XML form
- Monitoring table functions information can be viewed using administrative views
- New unit of work event monitor supports transaction monitoring
- Data partitions and partitioned indexes can be reorganized
- Partitioned table data remains available during roll-out operations
- Partitioned indexes on partitioned tables improve performance
- Distribution statistics collected for XML columns
- Table data can be moved online using a new stored procedure
- Relocating databases using db2relocatedb improved when paths differ
- System catalog views, system-defined administrative routines /views added and changed
- Additional system monitoring information can be generated
- Transparent LDAP authentication and group lookup supported (UNIX and Linux)
- 32-bit GSKit libraries are included in the 64-bit DB2 product installation
- GB18030 code set support has been extended
- DB2 installed on HP-UX operating systems now support long host names
- Multiple result sets can now be returned from an SQL procedure by enabling multiple instances of the same cursor
- db2support tool includes new filtering options
- Work action sets can be defined at workload level
- New time threshold limits unit of work duration
- Script facilitates migration from Query Patroller to workload manager
- New AUTOGRANT parameter for DB2 Text Search ENABLE DATABASE FOR TEXi
- LOB logging length Increased
- MONREPORT provides procedures for generating text monitoring reports

Version 9.7 Fix Pack 1 contains important changes that might affect your product usage. You should review the technical changes and new functionality included in Version 9.7 Fix Pack 1. For a summary of DB2 Version 9.7 LUW fix pack 1 changes, please see:

http://www-01.ibm.com/support/docview.wss?uid=swg24023357&myns=swgimgmt&mynp=OCSSEPGG&mynp=OCSSEPDU&mync=E

http://www-01.ibm.com/support/docview.wss?uid=swg24023357&myns=swgimgmt&mynp=OCSSEPGG&mynp=OCSSEPDU&mync=E#Description

Version 9.7 includes enhancements that improve IBM Data Server Provider for .NET support and connectivity to other data servers.

*•**ARRAY data type support** can be used with the Array data type with your stored procedure parameters. You can bind an array to a parameter in your procedure as a single argument. This simplifies the code around your SQL statements.*

*•**Compound statement support** in your SQL statements can improve performance by having the statements leverage the same access plan for a group of statements.*

*•**Host variable support** improves compatibility with applications that you use with other data servers. You can use host variables (:param) in place of positioned or named parameter markers (@param). However, you can specify only one type of parameter in a particular statement at a time.*

*•**Variable length time stamp support** makes it easier to work with other data servers. Previously, the TIMESTAMP data type had a fixed 6-digit precision. The TIMESTAMP data type now supports 0-12 digits of precision.*

db2updv97 command

- Starting with Version 9.7 Fix Pack 1, **SYSADM** runs **db2updv97** when applying new fix pack to ensure databases run as if created on that fix pack level. “This task is strongly recommended if you want to use capabilities specific to the fix pack. This task is not necessary if you installed the fix pack to create a new installation, since there are no existing databases.”
- Updates system catalog, creates new system-defined database objects, and alters existing system-defined database objects to correct definition.

  db2updv97 –d *dbname* **[-a] [-i] [-u** *userid* **–p** *password***]**
- Run on **catalog partition** in partitioned database environment.
- To build a list of commands to run per instance:
  db2 list db directory | grep -p -i
  'Directory entry type = Indirect' | grep 'Database alias' |
  awk '{print "db2updv97 -d", $4, "-a";}‘
- Updates to system catalogs include:
  - Creation of any new procedures available in fix pack level installing
  - Alteration of routines to correct the definition
  - Alteration of system administrative views to the correct definition
- http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.qb.server.doc/doc/t0024995.html

5

The **db2updv97,** update database to Version 9.7 fix pack command, updates the database system catalog to support the fix pack you have installed. This command can be used only on a database running DB2 9.7 Fix Pack 1 or later on Linux, UNIX or Windows. The command requires SYSADM authority and automatically establishes a connection to the specified database.

-d *database name* Specifies the name of the database to be updated.

-u *userid* Specifies the user ID.

-p *password* Specifies the password for the user.

-a Forces all updates to be run.

-h Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

-i Fixes the INDEX_TBSPACEID column in SYSCAT.DATAPARTITIONS (Starting in DB2 Version 9.7 Fix Pack 2).

If your databases were created or upgraded to DB2 9.7 GA and you applied DB2 9.7 Fix Pack 1 or later, then running the above command will automatically apply all updates required from Version 9.7 GA up to and including the fix pack level that you are installing.

This command must be run on the catalog partition in a partitioned database environment.

For improved performance, the **db2updv97** command will only apply updates that are required in order to make the databases appear as if they were created at the fix pack level that you are installing. Thus, if the command is issued more than once, no errors are reported and each of the system catalog updates is applied only once. However, if you wish to forcefully reapply all of the updates, you may do so by appending the **-a** parameter.

Updates to the system catalogs include the following:

•Creation of any new procedures available in the fix pack level that you are installing.

•Alteration of routines to correct the definition.

•Alteration of system administrative views to the correct definition.

FYI: Student Reference Only

The following query returns a result set that consist of tables with problems that can be resolved by the -i option:

```
SELECT TABSCHEMA, TABNAME
FROM   SYSCAT.DATAPARTITIONS
WHERE  INDEX_TBSPACEID  = 0 AND TBSPACEID IS NOT
NULL AND TABSCHEMA NOT IN ('SYSIBM', 'SYSCAT',
'SYSSTAT', 'SYSIBMADM', 'SYSCATV82') GROUP BY
(TABSCHEMA, TABNAME);
```

If this query returns any results, then you can do one of the following:

- Drop and recreate the tables shown in the query
- Run the db2updv97 with the -i option

If the problem is not fixed, then the db2look command will not print the correct DDL for tables returned from the query. Additionally, any applications that reference the INDEX_TBSPACEID column of SYSCAT.DATAPARTITIONS might contain inaccurate data.

System Catalog View Changes

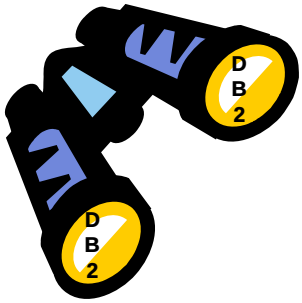
FP1

Modified

- SYSCAT.ATTRIBUTES
- SYSCAT.BUFFERPOOLS
- SYSCAT.CASTFUNCTIONS
- SYSCAT.COLUMNS
- SYSCAT.CONSTDEP
- SYSCAT.DATAPARTITIONS
- SYSCAT.DATATYPES
- SYSCAT.DBAUTH
- SYSCAT.HISTOGRAMTEMPLATEUSE
- SYSCAT.INDEXDEP
- SYSCAT.INDEXES
- SYSCAT.INDEXEXTENSIONDEP
- SYSCAT.INVALIDOBJECTS
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES
- SYSCAT.ROUTINEDEP
- SYSCAT.ROUTINEPARMS
- SYSCAT.ROUTINES
- SYSCAT.SECURITYPOLICIES
- SYSCAT.SEQUENCES
- SYSCAT.SERVICECLASSES
- SYSCAT.TABDEP
- SYSCAT.TABDETACHEDDEP
- SYSCAT.TABLES
- SYSCAT.TABLESPACES
- SYSCAT.THRESHOLDS
- SYSCAT.TRIGDEP
- SYSCAT.VARIABLEDEP
- SYSCAT.VARIABLES
- SYSCAT.WORKCLASSES
- SYSCAT.WORKLOADS
- SYSCAT.XSROBJECTDEP
- SYSSTAT.COLGROUPS

New

- SYSSTAT.COLUMNS
- SYSSTAT.INDEXES
- SYSCAT.CONDITIONS
- SYSCAT.DATATYPEDEP
- SYSCAT.INDEXPARTITIONS
- SYSCAT.INVALIDOBJECTS
- SYSCAT.MODULEAUTH
- SYSCAT.MODULEOBJECTS
- SYSCAT.MODULES
- SYSCAT.ROWFIELDS
- SYSCAT.XSROBJECTDETAILS
- SYSCAT.XMLSTRINGS



6

The system catalog views listed above in the left hand column have changed in Version 9.7. Most modifications to catalog views consist of new columns, changed descriptions, changed column data types, and increased column lengths.

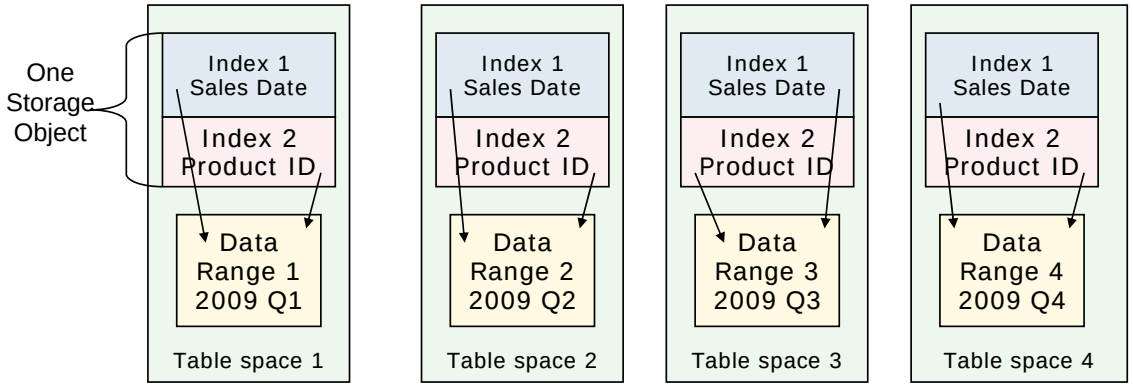
The right hand column lists system catalog views which have been added in Version 9.7.

Partitioned Indexes (DB2 9.7)

```
CREATE TABLE PARTTAB.HISTORYPART ( ACCT_ID INTEGER NOT NULL ,
       TELLER_ID SMALLINT NOT NULL ,
       BRANCH_ID SMALLINT NOT NULL ,
       BALANCE DECIMAL(15,2) NOT NULL ,
       ……..
       TEMP CHAR(6) NOT NULL   )
      PARTITION BY RANGE (BRANCH_ID)
      (STARTING FROM (1) ENDING (20) IN TSHISTP1 INDEX IN TSHISTI1 ,
       STARTING FROM (21) ENDING (40) IN TSHISTP2 INDEX IN TSHISTI2 ,
       STARTING FROM (41) ENDING (60) IN TSHISTP3 INDEX IN TSHISTI3 ,
       STARTING FROM (61) ENDING (80) IN TSHISTP4 INDEX IN TSHISTI4 ) ;

CREATE INDEX PARTTAB.HISTPIX1 ON PARTTAB.HISTORYPART (TELLER_ID) PARTITIONED;

CREATE INDEX PARTTAB.HISTPIX2 ON PARTTAB.HISTORYPART (BRANCH_ID) PARTITIONED;
```



In DB2 9.7, you can have indexes that refer to rows of data across all partitions in a partitioned table (known as *non-partitioned* indexes), or you can have the index itself partitioned such that each data partition has an associated *index partition*. You can also have both non-partitioned and partitioned indexes for partitioned tables.

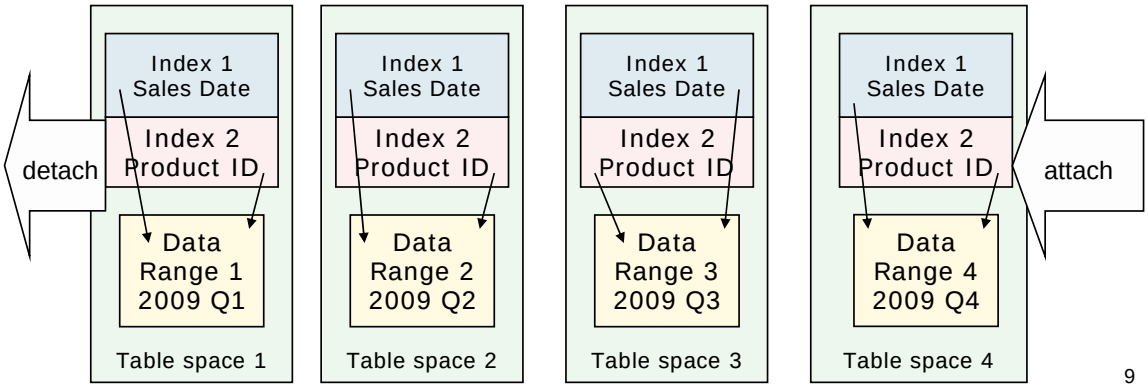
An index on an individual data partition is an index partition; the set of index partitions that make up the entire index for the table is a *partitioned index*. All of the partitioned indexes are managed as a single storage object per data range. The table space to be used for storing the partitioned indexes is based on the definition set during the CREATE TABLE processing. You can not specify the INDEX IN clause when a CREATE INDEX statement defines a new partitioned index. Since these indexes are divided into index partitions, the extra two byte partition id is not needed for each index key entry.

Reference Only: Student Notes

The example shown in the visual shows a range partitioned table with four data ranges defined in four table spaces. The two partitioned indexes are managed using four storage objects, one for each table space. This shows that the storage for partitioned indexes is closely related to each data range, unlike the non-partitioned indexes which were independent of the data ranges. The visual shows a sample CREATE TABLE statement for a range partitioned table named PARTTAB.HISTORYPART. This table has four defined data ranges, based on the BRANCH_ID column. Each data range defines a specific data and index table space to use for that range. There are two CREATE INDEX statements shown, defining two partitioned indexes. Each of these two indexes will be place into a common storage object located in the table space defined by the INDEX IN clause named in the range definition section of the CREATE TABLE statement

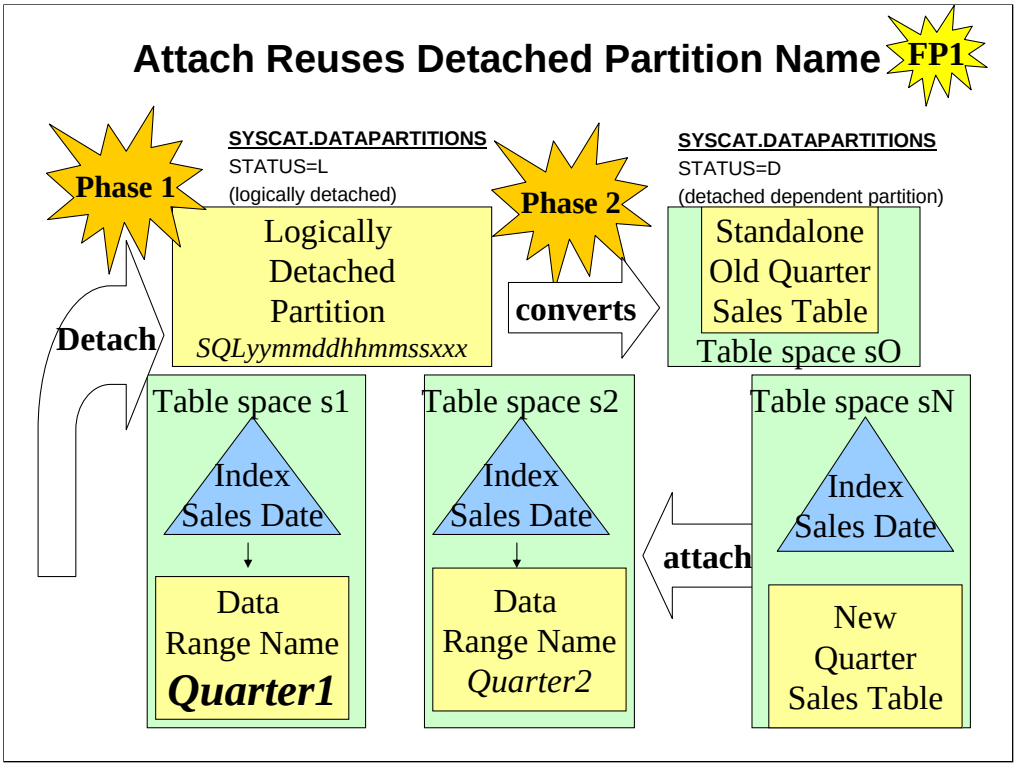
Attach/Detach using Partitioned Indexes (DB2 9.7)

- When a new data range is attached:
  - Reduced SET INTEGRITY processing, if matching Indexes exist on attached table, no indexes built for new range during SET INTEGRITY processing.
  - ERROR ON MISSING INDEXES option causes ATTACH to fail if source table does not have matching indexes. By default, any missing indexes will be created.
- When a data range is detached:
  - Index entries for the detached range are assigned to the detached table, ASYNC index processing is not needed.
  - Partition indexes are retained and assigned default names during detach.



9

You can use partitioned indexes to improve performance when you roll data into a table. Before you alter a partitioned table that uses partitioned indexes to attach a new partition or a new source table, you should create indexes on the table that you are attaching to match the partitioned indexes of the partitioned table. After attaching the source table, you still must issue a SET INTEGRITY statement to perform tasks such as range validation and constraint checking. However, if the source tables indexes match all of the partitioned indexes on the target table, SET INTEGRITY processing does not incur the performance and logging overhead associated with index maintenance. The newly rolled-in data is accessible quicker than it would otherwise be. All index key columns of the partitioned index on the target table must match with the index key columns of the index on the source table. If all other properties of the index are the same, then the index on the source table is considered a match to the partitioned index on the target table. That is, the index on the source table can be used as an index on the target table. The table here can be used to determine if the indexes are considered a match or not. The attach operation implicitly builds missing indexes on the source table corresponding to the partitioned indexes on the target table. The implicit creation of the missing indexes does take time to complete. You have an option to create an error condition if the attach operation encounters any missing indexes. The option is called ERROR ON MISSING INDEXES and is one of the attach operation options. The error returned when this happens is SQL20307N, SQLSTATE 428GE, reason code 18. Information on the non-matching indexes is placed in the administration log.



In the SYSCAT.DATAPARTITIONS catalog view, one row represents a data partition. The data partition statistics represent one database partition if the table is created on multiple database partitions. STATUS column has the following values:

•A = Data partition is newly attached

•D = Data partition is detached and detached dependents are to be incrementally maintained with respect to content of this partition

•I = Detached data partition whose entry in catalog is maintained only during asynchronous index cleanup; rows with a STATUS value of 'I' are removed when all index records referring to detached partition have been deleted

•L = Data partition is logically detached

Detach Data Partition – 2 Phase Process

ALTER TABLE … DETACH

- Data partition that is detached is converted into stand-alone table in the following two-phase process:

1. ALTER TABLE logically detaches data partition from partitioned table. Data partition name changed to system-generated name SQL*yymmddhhmmssxxx* so subsequent attach can reuse detached partition name immediately. In SYSCAT.DATAPARTITIONS, status of partition set to L (logically detached) if no detached dependent tables or D if there are detached dependent tables.
   - Modify applications that query catalog views for detached data partitions and use data partition names
2. An asynchronous partition detach task converts the logically detached partition into stand-alone table.
   - Target table unavailable until asynchronous partition detach task completes the detach.

11

In Version 9.7 Fix Pack 1 and later fix packs, the process to detach a data partition from a partitioned table is a two-phase process.

When you issue the ALTER TABLE statement with the DETACH partition clause, the data partition that you are detaching is converted into a stand-alone table in the following two-phase process:

1. The ALTER TABLE operation logically detaches the data partition from the partitioned table. The data partition name is changed to a system-generated name of the form SQL*yymmddhhmmssxxx* so that a subsequent attach can reuse the detached partition name immediately. In SYSCAT.DATAPARTITIONS, the status of the partition is set to L (logically detached) if there are no detached dependent tables or D if there are detached dependent tables.
2. An asynchronous partition detach task converts the logically detached partition into a stand-alone table.

The target table is unavailable until the asynchronous partition detach task completes the detach. For example, a DROP statement that drops the target table after a detach must wait until the asynchronous partition detach task completes the detach.

In Version 9.7 and earlier releases, the target table of an ALTER TABLE statement with the DETACH PARTITION clause became available immediately after the transaction issuing the ALTER TABLE statement committed if there were no detached dependent tables that needed to be incrementally maintained with respect to the detached data partition. If there were detached dependent tables, the target table became available after the SET INTEGRITY statement is run on all detached dependent tables.

Because the data partition name is changed to a system-generated name during the first phase of the detach process, you might need to modify applications that query the catalog views for detached data partitions and use the data partition names.

Attach Reusing Partition Name


```
db2 +c “alter table t1 detach
        partition QUARTER1
        into OLD_QUARTER_SALES_TABLE”

db2 +c “alter table t1 attach
        partition QUARTER1
        starting ’01/01/2010’ ending ’03/31/2010’
        from table
        NEW_QUARTER_SALES_TABLE”

db2 +c “rename tablespace s1 to sO”
db2 +c “rename tablespace sN to s1”
db2 commit work
```

12

Prior to DB2 9.7 Fix Pack 1 the above syntax would have failed because the partition name was not allowed to be used in the same unit of work. Starting in DB2 9.7 Fix Pack 1, the attach partition syntax can reuse the same partition name.

REORGCHK Shows Partitioned Index Stats (DB2 9.7)

```
SCHEMA.NAME                INDCARD  LEAF ELEAF LVLS  NDEL    KEYS   F4  F5  F6  F7  F8 REORG
-------------------------------------------------------------------------------------------
    -
Table: PARTTAB.HISTORYPART
Index: PARTTAB.HISTPIX1
Data Partition: PART0       353042   815     0    3     0    1000   10  92  30   0   0 *----

Index: PARTTAB.HISTPIX1
Data Partition: PART1        40966    96     0    2     0     999   12  93   -   0   0 *----

Index: PARTTAB.HISTPIX1
Data Partition: PART2        37418    88     0    2     0     996   12  93   -   0   0 *----

Index: PARTTAB.HISTPIX1
Data Partition: PART3        41530    97     0    2     0     998   11  93   -   0   0 *----

Index: PARTTAB.HISTPIX2
Data Partition: PART0       353042   814     0    3     0      20   81  92  30   0   0 -----

Index: PARTTAB.HISTPIX2
Data Partition: PART1        40966    95     0    2     0      20   69  93   -   0   0 *----

Index: PARTTAB.HISTPIX2
Data Partition: PART2        37418    87     0    2     0      20   69  93   -   0   0 *----

Index: PARTTAB.HISTPIX2
Data Partition: PART3        41530    96     0    2     0      20   69  93   -   0   0 *----
-------------------------------------------------------------------------------------------
    -
```

13

The visual shows the index portion from a sample REORGCHK report for a range partitioned table with four data ranges defined. There are two partitioned indexes on this table. The REORGCHK report uses the statistics collected by RUNSTATS which collects information about each index partition. The report shows the size and efficiency characteristics for each index partition. In the sample since the data range for PART0 is larger than the other data ranges, the index partitions for PART0 have three levels instead of the two levels needed for the smaller data ranges.

Reorganize Data Partitions/Partitioned Indexes

- **REORG TABLE ….**
  ON DATA PARTITION <table partition name>
  REORG INDEXES ALL …
  ON DATA PARTITION <table partition name>

- Classic table reorg on specified data partition while allowing other table data partitions to have read and write access when **there are NO nonpartitioned indexes** on table.
- Supported access on partition being reorganized are ALLOW NO ACCESS and ALLOW READ ACCESS.
- If nonpartitioned indexes on table, ALLOW NO ACCESS is only supported access mode for entire table.
- REORG TABLE/INDEXES ALL allows full read and write access to remaining table data partitions.

14

In Version 9.7 Fix Pack 1 and later fix packs, you can use the REORG command on a partitioned table to perform a reorganization of the data of a specific partition or the partitioned indexes of a specific partition. Only access to the specified data partition is restricted, the remaining data partitions of the table retain full read and write access. On a partitioned table, using the REORG TABLE or REORG INDEXES ALL command with the ON DATA PARTITION clause specifying a partition of the table supports the following features:

•REORG TABLE performs a classic table reorganization on the specified data partition while allowing the other data partitions of the table to be fully accessible for read and write operations when there are no nonpartitioned indexes (other than system-generated XML path indexes) on the table. The supported access modes on the partition being reorganized are ALLOW NO ACCESS and ALLOW READ ACCESS. When there are nonpartitioned indexes on the table (other than system-generated XML path indexes), the ALLOW NO ACCESS mode is the default and the only supported access mode for the entire table.

•REORG INDEXES ALL performs an index reorganization on a specified data partition while allowing full read and write access to the remaining data partitions of the table. All access modes are supported.

You can issue REORG TABLE commands and REORG INDEXES ALL commands on a data partitioned table to concurrently reorganize different data partitions or partitioned indexes on a partition. When concurrently reorganizing data partitions or the partitioned indexes on a partition, users can access the unaffected partitions but cannot access the affected partitions. All the following criteria must be met to issue REORG commands that operate concurrently on the same table:

•Each REORG command must specify a different partition with the ON DATA PARTITION clause.

•Each REORG command must use the ALLOW NO ACCESS mode to restrict access to the data partitions.

•The partitioned table must have only partitioned indexes if issuing REORG TABLE commands. No nonpartitioned indexes (except system-generated XML path indexes) can be defined on the table.

The db2Reorg API also supports reorganization of a data partition or its partitioned indexes.

db2pd –db <dbname> -reorgs

- **PartID**
  - The data partition identifier. One row is returned for each data partition, showing the reorganization information.
- **MasterTbs**
  - For partitioned tables, this is the logical table space identifier to which the partitioned table belongs. For non-partitioned tables, this value corresponds to the TbspaceID.
- **MasterTab**
  - For partitioned tables, this is the logical table identifier of the partitioned table. For non-partitioned tables, this value corresponds to the TableID.
- **TableName**
  - The name of the table.

- See student notes for more details about command output.

15

For the -reorgs parameter, the following information is returned:

Retrieval time - Retrieval time of this set of index reorg statistics information.

TabSpaceID - The table space identifier.

TableID - The table identifier.

Schema - Table schema.

Access level - Allow none, Allow read, Allow write

PartID - The data partition identifier. One row is returned for each data partition, showing the reorganization information.

MasterTbs - For partitioned tables, this is the logical table space identifier to which the partitioned table belongs. For non-partitioned tables, this value corresponds to the TbspaceID.

MasterTab - For partitioned tables, this is the logical table identifier of the partitioned table. For non-partitioned tables, this value corresponds to the TableID.

TableName - The name of the table.

Type - The type of reorganization. The possible values are Online and Offline.

IndexID - The identifier of the index that is being used to reorganize the table.

TempSpaceID - The table space in which the table is being reorganized.

Table Reorg Stats:

Address - A hexadecimal value.

TableName - The name of the table.

Start - The time that the table reorganization started.

End - The time that the table reorganization ended.

PhaseStart - The start time for a phase of table reorganization.

MaxPhase - The maximum number of reorganization phases that will occur during the reorganization. This value only applies to offline table reorganization.

Phase - The phase of the table reorganization. This value only applies to offline table reorganization. The possible values are Sort, Build, Replace, and InxRecreat.

CurCount - A unit of progress that indicates the amount of table reorganization that has been completed. The amount of progress represented by this value is relative to the value of MaxCount, which indicates the total amount of work required to reorganize the table.

MaxCount - A value that indicates the total amount of work required to reorganize the table. This value can be used in conjunction with CurCount to determine the progress of the table reorganization.

Status - The status of an online table reorganization. This value does not apply to offline table reorganizations. The possible values are Started, Paused, Stopped, Done, and Truncat.

Completion - The success indicator for the table reorganization. The possible values are 0 (successful) and -1 (failed).

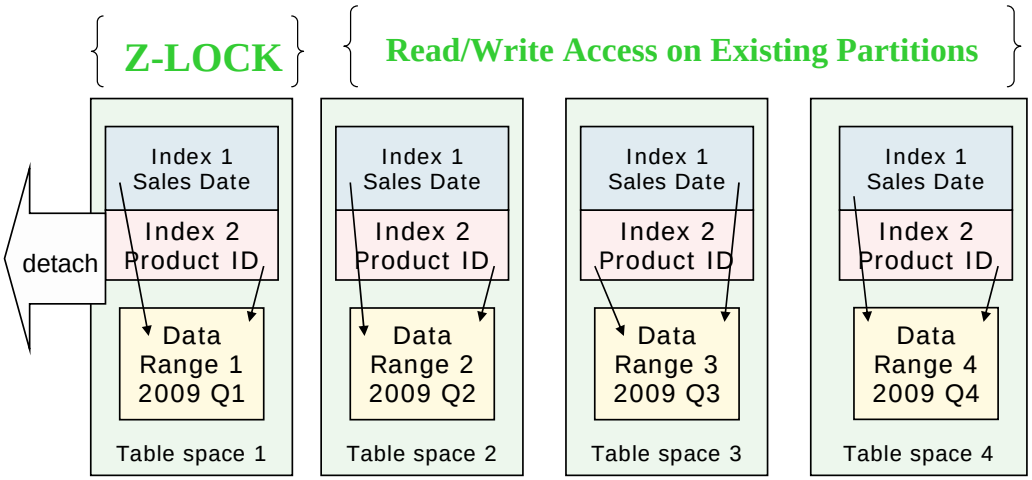
Locking during Detach with Partitioned indexes

FP1

- DB2 9.7, Alter Table … DETACH
  - Locks entire table in Z lock

Index Meta Data (IID)
SYSINDEXPARTITIONS
SYSINDEXES

- DB2 9.7 Fix Pack 1, Alter Table … DETACH
  - Locks only detached partition in Z lock
  - If dynamic non-UR queries have locked the partition to be detached, DETACH operation waits for lock to be released.



16

The attach operation drops indexes on the source table that do not match the partitioned indexes on the target table. The identification and dropping of these non-matching indexes takes time to complete. You should drop these indexes before attempting the attach operation.

When the ALTER TABLE DETACH is used for a range partitioned table with partitioned indexes, each of the index partitions defined on the source table for the data partition being detached becomes an index on the target table. The index object is not physically moved during the detach operation. However, the metadata for the index partitions of the table partition being detached are removed from the catalog table SYSINDEXPARTITIONS and new index entries are added in SYSINDEXES for the new table as a result of the detach operation. The original index identifier (IID) is kept and stays unique just as it was on the source table.

The index name for the surviving indexes on the target table are system-generated (using the form SQLyymmddhhmmssxxx). The schema for these indexes is the same as the schema of the target table except for any path indexes, regions indexes, and MDC Block indexes, which are in the SYSIBM schema. Other system-generated indexes like those to enforce unique and primary key constraints will have a schema of the target table because the indexes are carried over to the detached table but the constraints are not. You can use the RENAME command to rename the indexes that are not in the SYSIBM schema.

Partitioned Table Data Remains Available During Roll-out

- When detach table data partition, queries can access unaffected table data partitions during a roll-out operation

  ALTER TABLE...DETACH PARTITION

- DETACH operation does not wait for dynamic uncommitted read (UR) isolation level queries before it proceeds, nor does it interrupt any currently running dynamic UR queries. Occurs even when UR query is accessing partition being detached.
- If dynamic non-UR queries (read or write queries) have not locked the partition to be detached, DETACH operation can complete while dynamic non-UR queries are running against the table.
- If dynamic non-UR queries have locked the partition to be detached, DETACH operation waits for lock to be released.
- Hard invalidation must occur on all static packages that are dependent on the table before DETACH operation can proceed.

- The data partition being detached is converted into a stand-alone table in two-phase process:
  1. The ALTER TABLE...DETACH PARTITION operation logically detaches the data partition from the partitioned table.
  2. An asynchronous partition detach task converts the logically detached partition into a stand-alone table.

17

In DB2 Version 9.7 Fix Pack 1 and later fix packs, when detaching a data partition of a partitioned table, queries can continue to access the unaffected data partitions of the table during a roll-out operation initiated by the ALTER TABLE...DETACH PARTITION statement. When detaching a data partition from a partitioned table using the ALTER TABLE statement with the DETACH PARTITION clause, the source partitioned table remains online, and queries running against the table continue to run. The data partition being detached is converted into a stand-alone table in the following two-phase process:

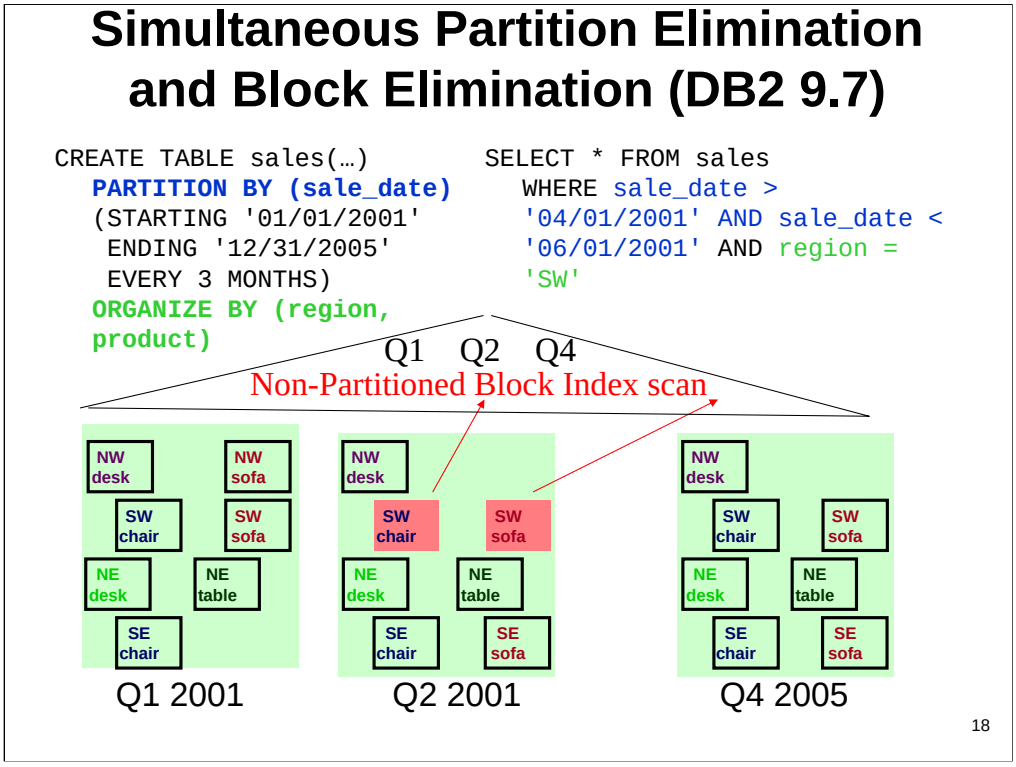
1. The ALTER TABLE...DETACH PARTITION operation logically detaches the data partition from the partitioned table.
2. An asynchronous partition detach task converts the logically detached partition into a stand-alone table.

If there are any dependent tables that need to be incrementally maintained with respect to the detached data partition (these dependent tables are referred to as detached dependent tables), the asynchronous partition detach task starts only after the SET INTEGRITY statement is run on all detached dependent tables.

In absence of detached dependents, the asynchronous partition detach task starts after the transaction issuing the ALTER TABLE...DETACH PARTITION statement commits.

The ALTER TABLE...DETACH PARTITION operation performs in the following manner:

- The DETACH operation does not wait for dynamic uncommitted read (UR) isolation level queries before it proceeds, nor does it interrupt any currently running dynamic UR queries. This behavior occurs even when the UR query is accessing the partition being detached.
- If dynamic non-UR queries (read or write queries) have not locked the partition to be detached, the DETACH operation can complete while dynamic non-UR queries are running against the table.
- If dynamic non-UR queries have locked the partition to be detached, the DETACH operation waits for the lock to be released.
- Hard invalidation must occur on all static packages that are dependent on the table before the DETACH operation can proceed.
- The following restrictions that apply to data definition language (DDL) statements also apply to a DETACH operation because DETACH requires catalogs to be updated:
  - New queries cannot be compiled against the table.
  - A bind or rebind cannot be performed on queries that run against the table.
- To minimize the impact of these restrictions, issue a COMMIT immediately after a DETACH operation.



In this example, a table was created that combined table partitions with multi-dimensional clustering as follows:

CREATE TABLE sales(...) PARTITION BY (sale_date) (STARTING '01/01/2001' ENDING '12/31/2005' EVERY 3 MONTHS) ORGANIZE BY (region, product)

A query runs with the following predicates:

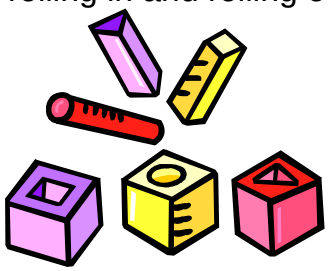
SELECT * FROM sales WHERE sale_date > '04/01/2001' AND sale_date < '06/01/2001' AND region = 'SW'

The query includes a predicate with a range of dates for the sale_date column. The table was defined with table partitions, each holding three months of data based on the column, so the DB2 optimizer would be able to determine that all of the rows that could possibly match this predicate would come from a single data partition. All of the other data partitions would be eliminated.

The query also contains a predicate on the region column, which is one of the two columns defined as MDC dimensions. The block index for the region column could be efficiently used to locate the slice of data within the one data partition that would match this second predicate. The block index entries would each contain the partition Ids as part of the index pointer.

MDC BID Indexes now Partitioned Indexes FP1

- Starting in DB2 9.7 Fix Pack 1, when creating table that uses both multidimensional clustering (MDC) and data partitioning, the system-created MDC block indexes are created as partitioned indexes.
- Data partitioned MDC tables can take advantage of the features available with partitioned tables such as the rolling in and rolling out of table data.



- For MDC tables that use table partitioning created with DB2 V9.7 and earlier, block indexes are nonpartitioned. Convert with **ADMIN_MOVE_TABLE**

19

In Version 9.7, you can have indexes that refer to rows of data across all partitions in a data partitioned table (known as *nonpartitioned* indexes), or you can have the index itself partitioned such that each data partition has an associated *index partition*. You can also have both nonpartitioned and partitioned indexes for partitioned tables. An index on an individual data partition is an index partition; the set of index partitions that make up the entire index for the table is a *partitioned index*.

Before Version 9.7, if you used an ALTER TABLE statement to attach a source table to a partitioned table as a new partition, the data in the new partition was not visible until after you issued a SET INTEGRITY statement to perform tasks such as updating indexes, enforcing constraints, and checking ranges. If the source table that you attached had a large amount of data, SET INTEGRITY processing might be slow and use a considerable amount of log space. Access to the data might be delayed.

Starting in Version 9.7, you can use partitioned indexes to improve performance when you roll data into a table. Before you alter a partitioned table that uses partitioned indexes to attach a new partition or a new source table, you should create indexes on the table that you are attaching to match the partitioned indexes of the partitioned table. After attaching the source table, you still must issue a SET INTEGRITY statement to perform tasks such as range validation and constraint checking. However, if the source tables indexes match all of the partitioned indexes on the target table, SET INTEGRITY processing does not incur the performance and logging overhead associated with index maintenance. The newly rolled-in data is accessible quicker than it would otherwise be.

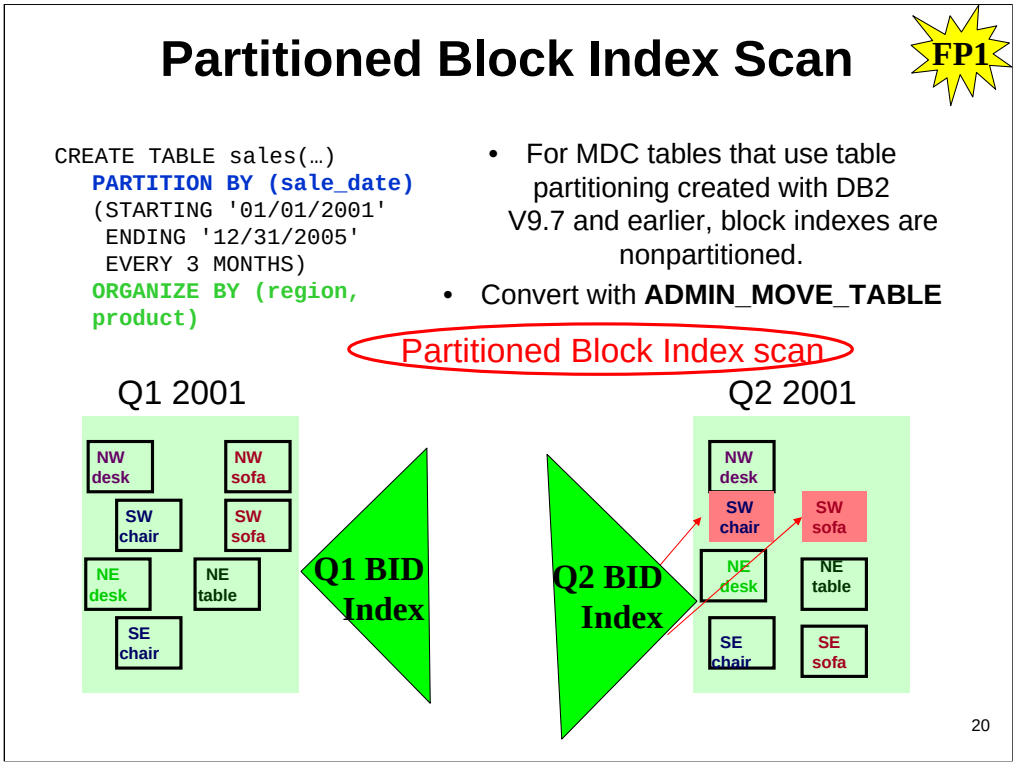
Partitioned indexes can also improve performance when you roll data out of a table. When you alter the table to detach one of its data partitions, that data partition takes its partitioned indexes with it, becoming a stand-alone table with its own indexes. You do not have to recreate the indexes for the table after detaching the data partition. Unlike nonpartitioned indexes, when you detach a data partition from a table that uses partitioned indexes, the associated index partitions go with it. As a result, there is no need for asynchronous index cleanup (AIC).

In addition, partition elimination for queries against a partitioned table that uses partitioned indexes can be more efficient. For nonpartitioned indexes, partition elimination can only eliminate data partitions. For partitioned indexes, partition elimination can eliminate both data and index partitions. This can result in having to scan fewer keys and index pages than a similar query over a nonpartitioned index.

By default, when you create indexes on partitioned tables, they are partitioned indexes. You can also include the PARTITIONED keyword of the CREATE INDEX statement to have a partitioned index created. You must use the NOT PARTITIONED keywords if you want a nonpartitioned index. All partitioned indexes for a data partition are stored in the same index object, regardless of whether the index partitions are stored in the same table space used for the data partition or in a different table space.

As in previous releases, you can use the ALTER TABLE statement with the ADD PARTITION clause to create a data partition for a partitioned table. To specify that partitioned indexes on the new data partition are to be stored in a different table space than the table space used for the data partition, use the INDEX IN option of the ADD PARTITION clause. If partitioned indexes exist on the partitioned table, the ADD PARTITION operation extends these indexes to the new partition, and the partitioned indexes are stored in the table space that you specify. If you do not use the INDEX IN option, the partitioned indexes are stored in the same table space in which the new data partition is stored.

Starting with DB2® V9.7 Fix Pack 1, when creating a table that uses both multidimensional clustering (MDC) and data partitioning, the system-created MDC block indexes are created as partitioned indexes. Data partitioned MDC tables can take advantage of the features available with partitioned tables such as the rolling in and rolling out of table data. For MDC tables that use table partitioning created with DB2 V9.7 and earlier, the block indexes are nonpartitioned.



Starting in DB2 9.7 Fix Pack 1, when creating table that uses both multidimensional clustering (MDC) and data partitioning, the system-created MDC block indexes are created as partitioned indexes.

Data partitioned MDC tables can take advantage of the features available with partitioned tables such as the
rolling in and rolling out of table data.

Partitioned Indexes over XML Data

- Starting in DB2 9.7 Fix Pack 1, can create index over XML data on partitioned table as partitioned or nonpartitioned.
  - default is partitioned index.

```
create index idx1 on T(doc) generate keys
  using xmlpattern '/product/id' as sql
  double;
```

- To create nonpartitioned index, specify NOT PARTITIONED option for CREATE INDEX statement.
- To convert nonpartitioned XML index to partitioned:
  1. Drop nonpartitioned index
  2. CREATE INDEX statement without NOT PARTITIONED option

21

On partitioned tables, indexes over XML data that you create with DB2 V9.7 or earlier are nonpartitioned. Starting in DB2 Version 9.7 Fix Pack 1, you can create an index over XML data on a partitioned table as either partitioned or nonpartitioned. The default is a partitioned index.

To create a nonpartitioned index, specify the NOT PARTITIONED option for the CREATE INDEX statement. To convert a nonpartitioned index over XML data to a partitioned index:

1. Drop the nonpartitioned index.
2. Create index by using the CREATE INDEX statement without the NOT PARTITIONED option.

Distribution Stats Collected for XML Columns

FP1

- Distribution statistics collected for XML columns to support faster queries over data in XML columns.
- Distribution statistics collected for indexes over XML data of type VARCHAR, DOUBLE, TIMESTAMP, and DATE.
- Collected when automatic table RUNSTATS performed

- To collect distribution statistics on XML column using RUNSTATS utility, both distribution statistics and table statistics must be collected.
  - because XML distribution statistics are stored with table statistics.
  - As default, RUNSTATS collects maximum of 250 quantiles for distribution statistics for each index over XML data.
  - Maximum number of quantiles for a column can be specified when executing the RUNSTATS.

22

Starting with DB2 Version 9.7 Fix Pack 1, distribution statistics can be collected for XML columns to support faster queries over the data in XML columns. Distribution statistics are collected for indexes over XML data of type VARCHAR, DOUBLE, TIMESTAMP, and DATE.

XML distribution statistics are not collected for indexes over XML data of type VARCHAR HASHED. Distribution statistics are collected for each index over XML data specified on an XML column.

XML distribution statistics are collected when automatic table RUNSTATS operations are performed. To collect distribution statistics on an XML column using the RUNSTATS utility, both distribution statistics and table statistics must be collected. Table statistics must be gathered in order for distribution statistics to be collected because XML distribution statistics are stored with table statistics. As the default, the RUNSTATS utility collects a maximum of 250 quantiles for distribution statistics for each index over XML data. The maximum number of quantiles for a column can be specified when executing the RUNSTATS utility.

The following list describes the situations in which XML distribution statistics are not created or collected:

- XML distribution statistics are not created when loading data with the STATISTICS option.
- XML distribution statistics are not collected for partitioned indexes over XML data defined on a data partitioned table.
- XML distribution statistics are not collected when collecting only index statistics, or collecting index statistics during index creation.

Distribution Stats Collected for XML Columns FP1

- Freqvalue statistics on XML columns governed by two DB2 database system registry values:
  - DB2_XML_RUNSTATS_PATHID_K
  - DB2_XML_RUNSTATS_PATHVALUE_K
    - default of 200 will be used for both parameters.
- **EXCLUDING XML COLUMNS**
  - facilitates collection of statistics on non-XML columns because inclusion of XML data can require greater system resources
- Statistics not collected on XML columns when ON KEY COLUMNS specified
- After LOAD executed with STATISTICS option, use RUNSTATS to collect statistics on XML columns.
  - When RUNSTATS used to collect statistics for XML columns only, existing statistics for non-XML columns are retained.

23

If XML type columns are specified in a column group, the XML type columns will be ignored for the purpose of collecting distinct values for the group. However, basic XML column statistics will be collected for the XML type columns in the column group.

EXCLUDING XML COLUMNS

This clause allows you to omit all XML type columns from statistics collection. This clause facilitates the collection of statistics on non-XML columns because the inclusion of XML data can require greater system resources. The EXCLUDING XML COLUMNS clause takes precedence over other clauses that specify XML columns for statistics collection. For example, if you use the EXCLUDING XML COLUMNS clause, and you also specify XML type columns with the ON COLUMNS clause or you use the ON ALL COLUMNS clause, all XML type columns will be ignored during statistics collection. For DB2 V9.7 Fix Pack 1 and later releases, distribution statistics over XML type columns are not collected when this clause is specified.

XML type columns are by definition not a key column and will not be included for statistics collection by the ON KEY COLUMNS clause.

For DB2 V9.7 Fix Pack 1 and later releases, distribution statistics for each index over XML data uses a maximum of 250 quantiles as the default. The default can be changed by specifying the NUM_QUANTILES parameter in the ON COLUMNS or the DEFAULT clause. The num_quantiles database configuration parameter is ignored while collecting XML distribution statistics.

After LOAD has been executed with the STATISTICS option, use the RUNSTATS utility to collect statistics on XML columns. Statistics for XML columns are never collected during LOAD, even when LOAD is executed with the STATISTICS option. When RUNSTATS is used to collect statistics for XML columns only, existing statistics for non-XML columns that have been collected by LOAD or a previous execution of the RUNSTATS utility are retained. In the case where statistics on some XML columns have been collected previously, the previously collected statistics for an XML column will either be dropped if no statistics on that XML column are collected by the current command, or be replaced if statistics on that XML column are collected by the current command.

If statistics for XML columns in the table are not required, the EXCLUDING XML COLUMNS option can be used to exclude all XML columns. This option takes precedence over all other clauses that specify XML columns for statistics collection.

RUNSTATS ON TABLE … FOR INDEXES ALL will cause the previously collected distribution statistics to be retained. If the RUNSTATS command is run on XML columns only, then previously collected basic column statistics and distribution statistics are retained. In the case where statistics on some XML columns have been collected previously, the previously collected statistics for an XML column will either be dropped if no statistics on that XML column are collected by the current command, or be replaced if statistics on that XML column are collected by the current command.

Reference Only: Student Notes

For DB2 V9.7 Fix Pack 1 and later releases, distribution statistics are collected on indexes over XML data defined on an XML column. When the RUNSTATS command is run on a table with the WITH DISTRIBUTION clause, the following apply to the collection of distribution statistics on a column of type XML:

•Distribution statistics are collected for each index over XML data specified on an XML column.

Notes: For Reference Only

The RUNSTATS command must collect both distribution statistics and table statistics to collect distribution statistics for indexes over XML data defined on an XML column. Table statistics must be gathered in order for distribution statistics to be collected since XML distribution statistics are stored with table statistics. An index clause is not required to collect XML distribution statistics. Specifying only an index clause does not collect XML distribution statistics.

By default, XML distribution statistics use a maximum of 250 quantiles for each index over XML data. When collecting distribution statistics on an XML column, you can change the maximum number of quantiles by specifying a value with NUM_QUANTILES parameter in the ON COLUMNS or the DEFAULT clause.

Distribution statistics are collected for indexes over XML data of type VARCHAR, DOUBLE, TIMESTAMP, and DATE. Distribution statistics are not collected over indexes of type VARCHAR HASHED.

Distribution statistics are not collected for partitioned indexes over XML data defined on a partitioned table.

Statistics collection on XML type columns is governed by two DB2 database system registry values: DB2_XML_RUNSTATS_PATHID_K and DB2_XML_RUNSTATS_PATHVALUE_K. These two parameters are similar to the NUM_FREQVALUES parameter in that they specify the number of frequency values to collect. If not set, a default of 200 will be used for both parameters.

ADMIN_MOVE_TABLE Online During all Phases FP1

- Starting in Version 9.7 Fix Pack 1, can access target table during the **copy** and **swap** phases by issuing **NO_TARGET_LOCKSIZE_TABLE** which disables default behavior of locksize table. The default is to have locksize table on the target table to prevent locking overhead, when no unique index is specified on the source table.

- Can also specify **CLUSTER** option that enables to read the data from source table with or without an ORDER BY.
  NON_CLUSTER reads data without ORDERBY regardless if cluster index specified.
  - improves data movement speed.

24

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. This feature reduces your total cost of ownership (TCO) and complexity by automating the process of moving table data to a new table object while allowing the data to remain online for select, insert, update, and delete access.

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.

Starting in Version 9.7 Fix Pack 1 and later fix packs, you can access the target table during the copy and swap phases by issuing NO_TARGET_LOCKSIZE_TABLE option which disables the default behavior of the locksize table. This option does not keep locksize table on the target table during the COPY and SWAP phases. The default is to have locksize table on the target table to prevent locking overhead, when no unique index is specified on the source table.

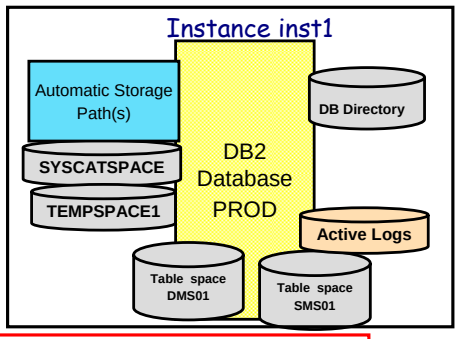
You can also specify the option that enables to read the data from the source table with or without an ORDER BY clause. This option improves the data movement speed. CLUSTER reads the data from the source table with an ORDER BY clause when a cluster index exists on the source table or a copy index has been specified. NON_CLUSTER reads the data from the source table without an ORDER BY clause regardless if a cluster index or copy index has been specified. Note: When neither CLUSTER or NON_CLUSTER options are specified, it will read the data from the source table with an ORDER BY clause only when a cluster index exists on the source table.

db2relocatedb – New Options to Change Paths

FP1

After copying the files, use db2relocatedb to:

1. Change database name
2. Change database path
3. Change DB2 instance
4. Change Automatic Storage Path(s)
5. Change Log path
6. Change Mirrorlogpath
7. Change Failarchivepath
8. Change Logarchmeth1
9. Change Logarchmeth2
10. Change Overflowlogpath
11. Change Table space container(s)



db2relocatedb -f filename

```
DB_NAME=oldName,newName
DB_PATH=oldPath,newPath
INSTANCE=oldInst,newInst
NODENUM=nodeNumber
LOG_DIR=oldDirPath,newDirPath
MIRRORLOGPATH=oldMirrorLogPath,newMirrorLogPath
FAILARCHIVEPATH=oldFailArchivePath,newFailArchivePath
LOGARCHMETH1=oldLogArchiveMeth1,newLogArchiveMeth1
LOGARCHMETH2=oldLogArchiveMeth2,newLogArchiveMeth2
OVERFLOWPATH=oldOverflowLogPath,newOverflowLogPath
CONT_PATH=oldContPath1,newContPath1
CONT_PATH=oldContPath2,newContPath2
STORAGE_PATH=oldStorPath1,newStorPath1
```

25

Starting in Fix Pack 1, you can specify additional keywords in the db2relocatedb command configuration file that make it easier to relocate a database when the paths used are different.

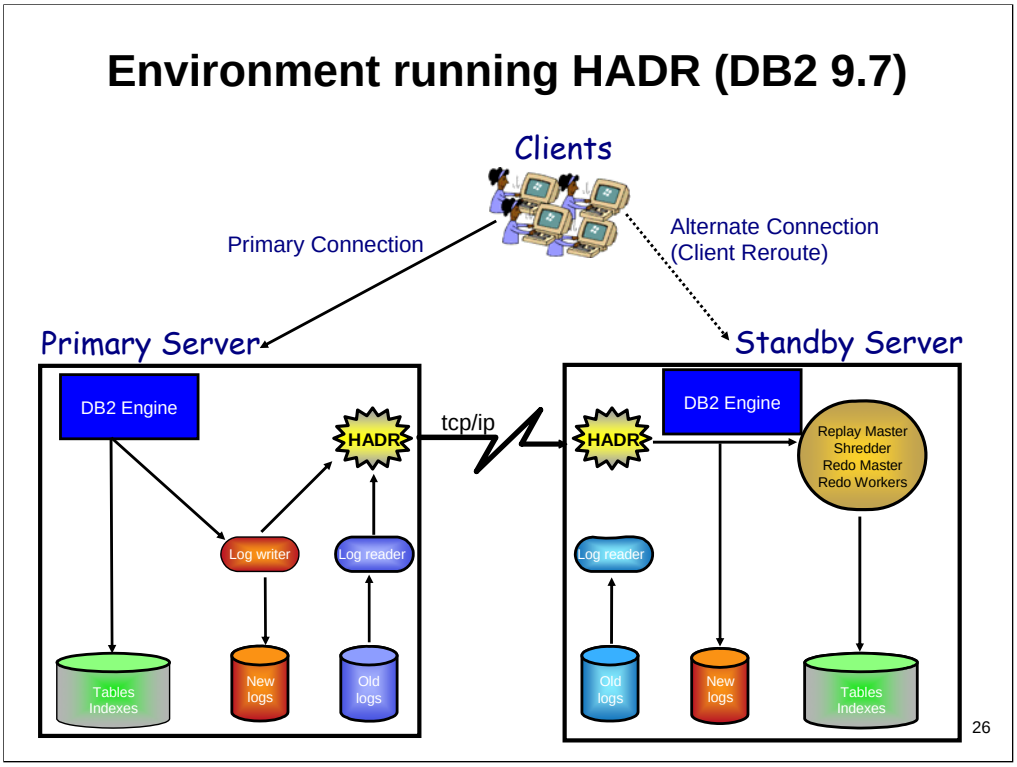
The db2relocatedb configuration file can contain new values for the mirrorlogpath, failarchivepath, logarchmeth1, logarchmeth2, and overflowlogpath database configuration parameters.

When you run the db2relocatedb command, the database configuration parameters of the relocated database are updated with the values specified in the configuration file. If you do not specify any of the new keywords, the relocated database maintains the original parameters values.

LOB Logging Length Increased

- Maximum LOB Length is 2 GB.
- Prior to DB2 9.7 Fixpack 1, maximum LOB length that could be logged was 1 GB.
- Starting with DB2 9.7 Fixpack 1, maximum LOB length that can be logged is 2 GB.
  - DB2 9.5 increased maximum LOGFILSZ from 1 GB to 2 GB.

Lobs can now be logged up to a length of 2 GB.



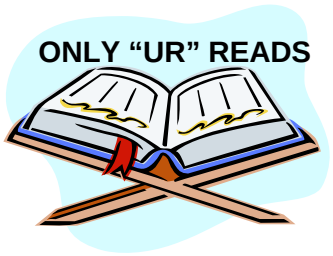
The HADR facility utilizes two DB2 databases, a Primary database and a Standby database. In most cases, the two databases would be running on two different machines, but for HADR testing, a single machine could be used for both. The Primary database processes the transactions for the connected client applications and sends the logged database changes to the Standby database using a TCP/IP network connection.

The Standby database, which is cloned from the Primary, receives the changes from the Primary and stores them in its copy of the active log files and immediately applies the changes to the Standby database. If the Primary DB2 database experiences a failure, the Standby will be able to quickly assume the role of Primary and begin to handle the normal transaction workload. Depending on the problem that caused the Primary database to fail, it may be possible to restart the old Primary and bring the database back in synch with the new/acting Primary database. Once the two databases are back into Peer state, a command can be used to switch database roles back to the normal status, with the original Primary acting as the Primary database again.

HADR Reads on Standby

FP1

- Standby enabled for read access
  - DB2_HADR_ROS = ON
- Supports all synchronization modes
  - ASYNC
  - NEAR SYNC
  - SYNC
- Uncommitted Read Isolation level
  - Can force applications to use UR
  - db2set DB2_STANDBY_ISO = UR
    - If not set, application requesting isolation level other than UR fails with SQL1773N
- Writers blocked



27

Starting with Version 9.7 Fix Pack 1, you can perform read operations on your High Availability and Disaster Recovery (HADR) standby database. All types of read queries, including scrollable and non-scrollable cursors, are supported on the standby. Read capability is supported in all three HADR synchronization modes (SYNC, NEARSYNC, and ASYNC) and in all HADR states except local catchup.

Previously, use of the HADR standby database was limited to replaying the logs shipped from the primary database, and user applications could not connect to the standby database. The new functionality does not affect the primacy of log replay, so the standby continues to remain constantly ready to take over the regular database workload from the HADR primary in the case of an outage.

The main benefit of the reads on standby capability is that it improves the utilization of the HADR standby. You can run queries on the standby if they do not entail the writing of a log record. By shifting various workloads to the HADR standby, you can freeing up resources to perform more work on the primary. You can also use the standby for reporting functions.

ROS Limitations

FP1

- Write operations are not permitted on the standby.
- Standby inaccessible during replay of DDL log records or maintenance operations (*replay-only window*).
  - When standby in replay-only window, existing connections to standby are terminated and new connections to standby are blocked (SQL1776N Reason Code 4).
- Inaccessible when in local catchup state.
- Only UR isolation level.
- instance level audit configuration not replicated to standby.
- DGTTs are not supported on standby.
- CGTTs only be created on primary and definitions replicated to standby. Access to CGTTs is not supported on standby.
- Not Logged Initially tables cannot be accessed on standby.
- Queries on standby only use SMS system temporary table spaces. DMS system temporary table spaces may result in an error.
- XML, LOB, LF, distinct type based on these types, and structured type columns can not be accessed.
- db2exfmt, db2expln, visual explain and db2batch tools are not supported.
- Explicit binding and rebinding and implicit rebinding of packages is not supported.
- STMM is not supported on the standby.
- WLM DDL statements on primary are replayed on the standby, but not effective. Any definitions that exist in db backup used to set up standby will be active on ROS.
- Creation and alteration of sequences not supported on standby. Cannot use NEXT VALUE expression to generate next value in a sequence.
- Runtime revalidation of invalid objects not supported on standby.
- Standby cannot be configured as a Federation Server.
- Backup and archive operations are not supported on standby.
- Quiesce operations are not supported on standby.

The High Availability and Disaster Recovery (HADR) reads on standby feature allows you to run read-only workloads on an HADR active standby database. In addition to the read-only restriction, this feature has the above limitations that you should be aware of.

ROS - Replay-Only Window

- **Maintenance operations that trigger replay-only window:**
  - Classic, or offline, reorg
  - Inplace, or online, reorg
  - Index reorg (indexes all, individual index)
  - MDC reclaim reorg
  - Load
  - Bind or rebind
  - DDL
  - db2rbind
  - Runstats
  - Table move
  - Auto statistics
  - Auto reorg
  - Real Time Statistics
  - Automatic Dictionary Creation for tables with COMPRESS YES attribute
  - Asynchronous Index Cleanup on detached table partition
  - Implicit rebind
  - Implicit index rebuild
  - Manual update of statistics.
  - Deferred MDC rollout
  - Asynchronous Index cleanup after MDC rollout
  - Reuse of a deleted MDC block on insert into MDC table
  - Asynchronous background processes updating catalog tables SYSJOBS and SYSTASKS

When an HADR active standby database is replaying DDL log records or maintenance operations, the standby enters the *replay-only window*. When the standby is in the replay-only window, existing connections to the standby are terminated and new connections to the standby are blocked (SQL1776N Reason Code 4). New connections are allowed on the standby after the replay of all active DDL or maintenance operations has completed.

The only user connections that can remain active on a standby in the replay-only window are connections that are executing DEACTIVATE DATABASE or TAKEOVER commands. When applications are forced off at the outset of the replay-only window, an error is returned (SQL1224N). Depending on the number of readers connected to the active standby, there may be a slight delay before the DDL log records or maintenance operations are replayed on the standby.

Monitoring Replay-Only Window Status

db2pd -db hadrdb –hadr

- **ReplayWindowStartTime** - indicates time current replay-only window became active
- **MaintenanceTxCount** or **DDLTxCount -** indicates total existing uncommitted DDL or maintenance transactions executed so far in current replay-only window

Recommendations for minimizing impact of replay-only window:

- Run DDL and maintenance operations during scheduled maintenance window, preferably at off-peak hours.
- Run DDL operations collectively rather than in multiple groups.
- Run REORG or RUNSTATS only on required tables instead of all tables.
- Terminate applications on active standby using FORCE APPLICATION ALL before running DDL or maintenance operations on primary. Monitor replay-only window to determine when it is inactive, and redeploy applications on standby.

To monitor a replay-only window on an active standby, use the db2pd command with the -hadr option.

Although an HADR active standby database can be used to run read-only workloads, its main role is to replay log records in order to stay synchronized with the HADR primary in case it has to take over the primary's role. In cases where the read-only workload is causing the standby to fall behind in its log replay, you might want to temporarily terminate all of the connections to the standby to allow it to catch up.

Use the following procedure to temporarily make an active standby inaccessible to readers.

- Issue the FORCE APPLICATION command. This terminates existing connections to the standby.
- Change the virtual IP configuration. This prevents new connections to the standby.

After the standby has caught up with the primary through log replay, you need to revert the virtual IP configuration to its original setting to allow the connections to the active standby to resume.

Last Referenced Data Available for Tables, MQTs, Table Partitions, Indexes and Packages

FP1

- Last referenced date updated in LASTUSED column of corresponding catalog table when access object with DML or LOAD.
- Last referenced date used to identify objects that have not been accessed for an extended period of time and might be considered candidates for removal.
- Removing indexes that are never used in queries saves both disk space and maintenance overhead (overhead when insertions and updates performed on table on which index was defined).

28

The last referenced date indicates the last date that an object was used and is available for tables, table data partitions, indexes, packages, and materialized query tables (MQTs). The last referenced date is accessed through the LASTUSED column of the corresponding catalog table for the object.

The last referenced date is used to identify objects that have not been accessed for an extended period of time and might be considered candidates for removal. For example, removing indexes that are never used in queries saves both disk space and maintenance overhead (that is, overhead when insertions and updates are performed on the table on which the index was defined).

Determining a Date Object LASTUSED FP1

- The last referenced date is of interest when an object has not been used for an extended period of time (for example, several months). LASTUSED is useful in the following cases:
  - Tables and table data partitions: can identify opportunities to reclaim unused space
  - Indexes: can identify opportunities to reclaim unused space, avoid unnecessary inserts and maintenance, and improve compile time by reducing the number of choices for an index to consider
  - Packages: can detect unused package versions which can be freed
  - MQTs: can help to detect unused MQTs, to reclaim unused space, or help to investigate and understand why an MQT is not being used

- **SYSCAT.DATAPARTITIONS.LASTUSED**
- **SYSCAT.INDEXES.LASTUSED**
- **SYSCAT.PACKAGES.LASTUSED**
- **SYSCAT.TABLES.LASTUSED**

29

The following examples describe some specific scenarios in which the last referenced date can be useful:

To identify opportunities to save space and maintenance overhead, you can examine last used information for indexes every year by checking the LASTUSED column in the SYSCAT.INDEXES catalog view. If an index has not been used in the last year, the index can be considered as a candidate for being dropped. The final decision to drop an index remains under your control because there might be circumstances in which dropping an index is not desired. For example, you might have a table which is known to be accessed only under emergency or infrequent cases where fast access is critical, or the index for a table might be unique and used to enforce the uniqueness constraint even though it is never explicitly used. The last used date information can be used as an aid in making decisions to remove indexes.

Your company has internal applications that were deployed on the database and were either replaced or are no longer in use after a period of months or years. The retired applications have been identified as opportunities to save space. The last used date information can be used to identify database objects that are no longer in use and were not cleaned up after an application was retired. For example, these database objects might be tables storing values used to populate a GUI. The last used date for these tables can be found in the LASTUSED column of the SYSCAT.TABLES catalog view and this date can be used as a starting point in the investigation of table objects that can be removed to reclaim space.

db2pd – Additional System Monitoring Info

- V9.7: **summary option** for **-pages**
  - Generates more compact report that contains only buffer pool summary information section.
  - Additional columns about table space IDs, dirty pages, permanent pages, and temporary pages displayed.
- V9.7: new parameter **-wlocks**
  - Monitor dynamically applications with locks in lock wait mode.
  - **file=*filename*** option saves information into separate files.
- V9.7: new parameter **-apinfo**
  - Captures detailed runtime information about specific application or for all applications. ***AppHandl***, ***MaxStmt,*** **file=*filename*** options
  - db2pd -apinfo 12 -db mydb10
- V9.7 Fix Pack 1: **index option** for **-reorgs**
  - Displays progress information about RUNSTATS
  - Displays progress info about table and index reorganizations.

30

Version 9.7 contain db2pd improvements that make it easier to monitor system activities, including table and index reorganization progress information. Starting in Version 9.7, a summary option is available for the -pages command parameter, and new command parameters (-wlocks and -apinfo) are also available.

You can use the summary option for the -pages parameter to generate a more compact report that contains only the buffer pool summary information section. Additional columns that include information about table space IDs, dirty pages, permanent pages, and temporary pages are displayed in the summary section.

You can use the -wlocks parameter to monitor dynamically the applications with locks that are in lock wait mode. It displays the owner and waiter information for each lock being waited on. The lock status (Sts) value of G designates the owner of the lock, while a Sts value of W designates the waiter of that lock. **file=*filename* s**ends the -wlocks output to a specified file.

You can use the -apinfo parameter to capture detailed runtime information about a specific application or for all applications. Both parameters have options to save the information into separate files. You can specify the following –apinfo options:

- ***AppHandl* -** If an application handle is specified, information is returned about that particular application. The default is to display information for all applications running at that partition.
- ***MaxStmt* -** If a number of maximum statements is specified, the information for the most recent of SQL statements, equalling the maximum number specified, is returned. The default is to display information for all the executed SQL statements.
- **file=*filename* -** Sends the -apinfo output to a specified file.

In Version 9.7 Fix Pack 1 and later fix packs, you can use the index option of the -reorgs parameter to display progress information about the RUNSTATS command as well as table and index reorganizations.

db2pd -apinfo 12 -db mydb10 FP1

```
Database Partition 0 -- Database MYDB10 -- Active -- Up 0 days 00:03:28
Application :
  Address :                0x0780000000D76EE0
  AppHandl [nod-index] :   12       [000-00012]
  Application PID :        1384708
  Application Node Name :  boson
  IP Address:              n/a
  Connection Start Time :  (1195265036)Fri Nov 16 21:03:56 2007
  Client User ID :         venus
  System Auth ID :         VENUS
  Coordinator EDU ID :     1801
  Coordinator Partition :  0
  Number of Agents :       1
  Locks timeout value :    4294967294 seconds
  Locks Escalation :       No
  Workload ID :            1
  Workload Occurrence ID : 1
  Trusted Context :        n/a
  Connection Trust Type :  non trusted
  Role Inherited :         n/a
  Application Status :     Lock-wait
  Application Name :       db2bp
  Application ID :         *LOCAL.venus.071117020356
  ClientUserID :           n/a
  ClientWrkstnName :       n/a
  ClientApplName :         n/a
  ClientAccntng :          n/a
  List of active statements :
  *UOW-ID :          8
   Activity ID :     2
   Package Schema :  NULLID
   Package Name :    SQLC2G13
   Package Version :
   Section Number :  201
   SQL Type :        Dynamic
   Isolation :       CS
   Statement Type :  DML, Select (blockable)
   Statement :       select * from t2
  List of inactive statements of current UOW :
   UOW-ID :          8
   Activity ID :     1
   Package Schema :  NULLID
   Package Name :    SQLC2G13
   Package Version :
   Section Number :  203
   SQL Type :        Dynamic
   Isolation :       CS
   Statement Type :  DML, Insert/Update/Delete
   Statement :       insert into t1 values 1
```

31

The above is a sample of the output of the db2pd -apinfo command.

db2support Tool Enhancements FP1

-history *history period* and **-time *time interval*** limit data gathered. Can specify start and stop time.

-Archive *archive path* creates copy of DIAGPATH in archive path specified. Archived directory's name appended with hostname and current time stamp.

-basic limits data gathered to optimizer-related diagnostics.

-ol enhanced to gather data for multiple optimization levels.

-extenddb2batch allows db2batch info to be collected for all optimization levels when used with -ol and –cl.

-nodb2look and **-nocatalog** prevent collection of db2look information and catalog information.

32

Starting in Fix Pack 1, the db2support tool includes new filtering options that you can use to gather specific diagnostic data more easily and an archiving option for storing diagnostic files in a different location.

The **H *history_period* | -history *history_period*** option limits the data collected by db2support to a particular interval of time. The history_period variable must be specified. The history_period variable may be specified with a number and time type with an optional beginning time value separated by a colon. The available types are:d = days h = hours m = minutes s = seconds The beginning of time value is specified in time stamp format. Time stamp format is YYYY-MM-DD-hh.mm.ss.nnnnnn where YYYY specifies a year, MM for a month of a year (01 through 12), DD for a day of a month (01 through 31), hh for an hour of a day (00 through 23), mm for minute of an hour (00 through 59), ss for the seconds of a minute (00 through 59), nnnnnn for microseconds on UNIX systems or milliseconds on Windows systems. Some or all of the fields that follow the year field may be omitted. If fields are omitted, the default values will be used. The default values are 1 for the month and day, and 0 for all other fields. The number and time type may be positive or negative specified with + or - signs. If only a number and time type are specified the default is negative. If a number and time type are specified and a beginning time value is specified the default is positive. For example, -history 6d will collect data for the past 6 days. -history 6d:2009 will collect data for the first 6 days of 2009. This option cannot be used with the -time or -t option.

The -Archive *archive path* option archives all the data from the directory specified in the DIAGPATH configuration parameter into the specified archive path. A new directory will be created in the specified archive path with the name “DB2DUMP” with the system hostname and timestamp appended, for example "DB2DUMP_systemhostname_2009-01-12-12.01.01".

Notes: For Reference Only

The **B | -basic option** restricts the collection to only optimizer information. No other information will be collected except information for the db2supp_opt.zip file. The -basic parameter must be used with the -st, -sf, or -se parameters or a syntax error will be returned.

The -ol levels | -optlevel levels specifies the value of the optimization level special register to use. The default is the value of the dft_queryopt database configuration parameter. The optimization level value may be specified as a single value or multiple values separated by a comma. If multiple values are specified, all optimization information is collected for the first value. For each additional optimization level value specified, the explain plans will be collected and stored in a separate file along with the initial and end times of the collection for each level.

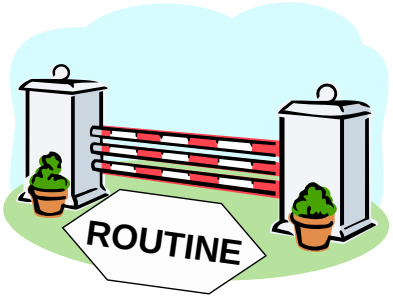
The -extenddb2batch specifies that db2batch information for all the optimization levels specified with the -ol or -optlevel parameter are to be captured. At least one value for the -ol parameter and a -cl parameter value of 3 must be specified with the -extenddb2batch parameter or db2support will return a syntax error.

The -nodb2look and -nocatalog options prevent the collection of db2look information and catalog information respectively. The default is to gather both.

The -t time_interval | -time time_interval option limits the data collected by db2support to a particular time interval specified by the time interval variable. The time interval can be specified as a start time, end time, or both in time stamp format separated by a colon. Time stamp format is YYYY-MM-DD-hh.mm.ss.nnnnnn where YYYY specifies a year, MM for a month of a year (01 through 12), DD for a day of a month (01 through 31), hh for an hour of a day (00 through 23), mm for minute of an hour (00 through 59), ss for the seconds of a minute (00 through 59), nnnnnn for microseconds on UNIX systems or milliseconds on Windows systems. Some or all of the fields that follow the year field may be omitted. If fields are omitted, the default values will be used. The default values are 1 for the month and day, and 0 for all other fields. If only a start time is specified (-t 2009), db2support will collect files that are modified after the start time. If only an end time is specified (-t :2009), db2support will collect files that are modified before end time. If both of inputs are specified (-t 2008:2009), db2support will collect files that are modified within the interval of the start time and end time. There is no default value for this option. At least one of time stamps must be specified. This option cannot be used with the -history or -H option.

Track Execution History of Fenced Routines FP1

- **db2pd –fmpexechistory -genquery**
  - Display execute history of fenced routines (including routines attempted to be loaded/run) to diagnose FMP process related issues
  - Use –**genquery** option to generate query that you can save and reuse to return the routine schema, module, name, and specific name according to a routine unique ID
    - Can run query after dbm stopped and started as long as no routine dropped
    - Query result reflects routine execution history collected at point when db2pd command was run



33

Starting in DB2 Version 9.7 Fix Pack 1, you can keep track of the execution history of fenced routines (including those that attempted to be loaded) more easily using the output of the db2pd command with the new -fmpexechistory parameter.

You can use the -fmpexechistory parameter to display the execution history of fenced routines (including the routines that attempted to run) in order to diagnose some FMP process related issues.

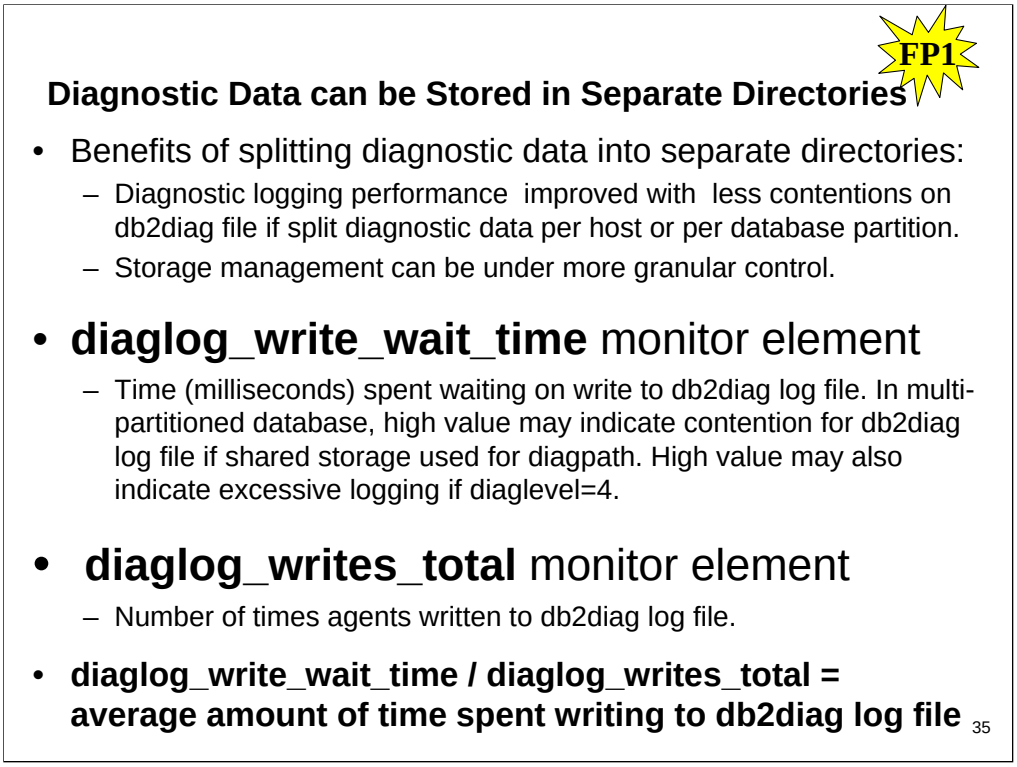
To help interpret the fenced routines history information provided by the db2pd command, you can use the genquery option to generate a query that you can save and reuse to return the routine schema, module, name, and specific name according to a routine unique ID. You can run this query after the database manager is stopped and started, and as long as no routine is dropped, the query result will reflect the routine execution history collected at the point when the db2pd command was run

db2pd -fmpexechistory Returns Following Information:

- **FMP Process:**
  - FmpPid - Process ID of the FMP process.
  - Bit - Bit mode. Values are 32 bit or 64 bit.
  - Flags - State flags for the FMP process. Possible values are:
    - 0x00000000 - JVM initialized
    - 0x00000002 - Is threaded
    - 0x00000004 - Used to run federated wrappers
    - 0x00000008 - Used for Health Monitor
    - 0x00000010 - Marked for shutdown and will not accept new tasks
    - 0x00000020 - Marked for cleanup by db2sysc
    - 0x00000040 - Marked for agent cleanup
    - 0x00000100 - All ipcs for the process have been removed
    - 0x00000200 - .NET runtime initialized
    - 0x00000400 - JVM initialized for debugging
    - 0x00000800 - Termination flag
  - ActiveThrd - Number of active threads running in the FMP process.
  - PooledThrd - Number of pooled threads held by the FMP process.
  - ForcedThrd - Number of forced threads generated by the FMP process.
  - Active - Active state of the FMP process. Values are Yes or No.
- **Active Threads:**
  - EduPid - EDU process ID that this thread is working.
  - ThreadId - Active thread ID.
  - RoutineID - The routine identifier.
  - Timestamp - A unique identifier for the usage of an application handle.
- **Pooled Threads:**
  - ThreadId - Pooled thread ID.
  - RoutineID - The routine identifier.
  - Timestamp - A unique identifier for the usage of an application handle.
- **Forced Threads:**
  - ThreadId - Forced thread ID.
  - RoutineID - The routine identifier.
  - Timestamp - A unique identifier for the usage of an application handle.

34

For the -fmpexechistory | -fmpe parameter the above information is collected.



Starting with Version 9.7 Fix Pack 1, you can specify to store DB2 diagnostic data in separate directories named according to the physical host, database partition, or both by setting the enhanced diagpath database manager configuration parameter. Separate db2diag log files can later be merged together using the db2diag -merge command.

The benefits of splitting the diagnostic data into separate directories are as follows:

•Diagnostic logging performance can be improved because of less contentions on the db2diag log file if you split the diagnostic data per host or per database partition.

•Storage management can be under more granular control.

Table Function Collection Level for diaglog_write_wait_time and diaglog_writes_total monitor elements :

•MON_GET_CONNECTION_DETAILS

•MON_GET_SERVICE_SUBCLASS_DETAILS

•MON_GET_UNIT_OF_WORK_DETAILS

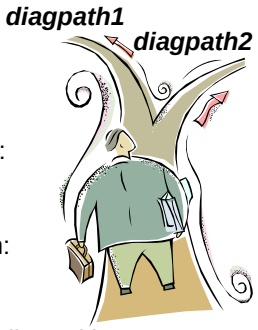
•MON_GET_WORKLOAD_DETAILS

•MON_GET_ACTIVITY_DETAILS

Diagnostic Data can be Stored in Separate Directories

FP1

- Split default diagnostic directory path by physical host:
  db2 update dbm cfg using diagpath '"$h"'
- Split specified diagnostic directory path by physical host:
  db2 update dbm cfg using diagpath '"*pathname* $h"'
- Split default diagnostic directory path by database partition:
  db2 update dbm cfg using diagpath '"$n"'
- Split specified diagnostic data directory path by db partition:
  db2 update dbm cfg using diagpath '"*pathname* $n"'



- Split default diagnostic directory path by physical host and db partition:
  db2 update dbm cfg using diagpath '"hn"'
- Split specified diagnostic directory path by physical host and db partition:
  db2 update dbm cfg using diagpath '"*pathname* hn"'
- Merge separate db2diag log files with **db2diag –merge** command

36

To split the diagnostic data into separate directories, set the diagpath database manager configuration parameter to one of the following values:

•Split default diagnostic data directory path according to physical host:

db2 update dbm cfg using diagpath '"$h"'

•Split your own specified diagnostic data directory path according to physical host:

db2 update dbm cfg using diagpath '"*pathname* $h"'

•Split default diagnostic data directory path according to database partition:

db2 update dbm cfg using diagpath '"$n"'

•Split your own specified diagnostic data directory path according to database partition:

db2 update dbm cfg using diagpath '"*pathname* $n"'

•Split default diagnostic data directory path according to physical host and database partition:

db2 update dbm cfg using diagpath '"hn"'

•Split your own specified diagnostic data directory path according to physical host and database partition:

db2 update dbm cfg using diagpath '"*pathname* hn"'

Merging separate db2diag log files can, at times, make analysis and troubleshooting easier. In that case, you can use the db2diag -merge command.

Merging Files and Sorting Records According to Timestamps

FP1

The two db2diag log files to merge are the following:

- db2diag.0.log; contains records of Level:Error with the following timestamps:
  - 2009-02-26-05.28.49.822637
  - 2009-02-26-05.28.49.835733
  - 2009-02-26-05.28.50.258887
  - 2009-02-26-05.28.50.259685
- db2diag.1.log; contains records of Level:Error with the following timestamps:
  - 2009-02-26-05.28.11.480542
  - 2009-02-26-05.28.49.764762
  - 2009-02-26-05.29.11.872184
  - 2009-02-26-05.29.11.872968
- To merge the two diagnostic log files and sort the records according to timestamps, execute the following command:
  db2diag -merge db2diag.0.log db2diag.1.log -fmt %{ts} -level error
- The result of the merge and sort of the records is the following:
  - 2009-02-26-05.28.11.480542
  - 2009-02-26-05.28.49.764762
  - 2009-02-26-05.28.49.822637
  - 2009-02-26-05.28.49.835733
  - 2009-02-26-05.28.50.258887
  - 2009-02-26-05.28.50.259685
  - 2009-02-26-05.29.11.872184
  - 2009-02-26-05.29.11.872968
- where the timestamps are merged and sorted chronologically.

36

Please see the following for more details:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.mon.doc/doc/c0056434.html

Merging Split Diagnostic Directory Path Files from Single Host and Sorting Records by Timestamps

FP1

- Three database partitions on the current host.
- The diagpath was set in the following way:
  db2 update dbm cfg using diagpath '"$n"'
- The following is a list of the three db2diag log files to merge:
  - ~/sqllib/db2dump/NODE0000/db2diag.log
  - ~/sqllib/db2dump/NODE0001/db2diag.log
  - ~/sqllib/db2dump/NODE0002/db2diag.log
- To merge the three diagnostic log files and sort the records according to timestamps, execute the following command:
  db2diag -merge

37

Please see the following for more details:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.mon.doc/doc/c0056434.html

Merging Split Diagnostic Directory Path Files from Multiple Hosts and Database Partitions

FP1

- The default diagnostic data directory path was split according to physical host and database partition by setting diagpath to:
  db2 update dbm cfg using diagpath '"hn"'
- To obtain an output of all records from all the diagnostic logs and merge the diagnostic log files from three database partitions on each of two hosts, bower and horton. The following is a list of the six db2diag log files:
  - ~/sqllib/db2dump/HOST_bower/NODE0000/db2diag.log
  - ~/sqllib/db2dump/HOST_bower/NODE0001/db2diag.log
  - ~/sqllib/db2dump/HOST_bower/NODE0002/db2diag.log
  - ~/sqllib/db2dump/HOST_horton/NODE0003/db2diag.log
  - ~/sqllib/db2dump/HOST_horton/NODE0004/db2diag.log
  - ~/sqllib/db2dump/HOST_horton/NODE0005/db2diag.log
- To output the records from all six db2diag log files, run the following command:
  db2diag -global

38

Please see the following for more details:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.mon.doc/doc/c0056434.html

Merge all db2diag Log Files & Format Based upon Timestamp FP1

- To merge all six db2diag log files in the diagnostic data directory path from all three database partitions on each of the hosts bower and horton and format the output based on the timestamp, execute the following command:

db2diag -global -merge -sdir /temp/keon -fmt %{ts}

- where /temp/keon is a shared directory, shared by the hosts bower and horton, to store temporary merged files from each host during processing.

39

Please see the following for more details:

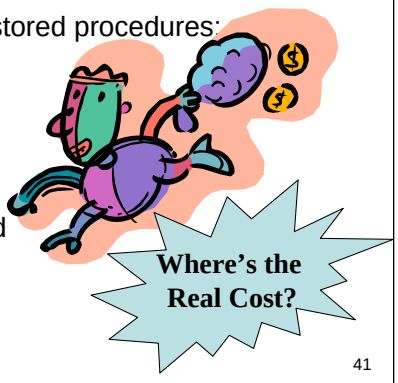
http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.mon.doc/doc/c0056434.html

Explain Statements from Runtime Section

FP1

- Perform explain directly from contents of runtime section
  - Advantage of SECTION EXPLAIN over traditional explain is will recompile statement being explained. If compilation environment or table statistics have changed when EXPLAIN statement is issued, compiler may generate different access plan.
  - Section explain always provides exact access plan that was executed, since access plan is reconstructed directly from executable section.

- Section explain accessible through set of stored procedures:
  - EXPLAIN_FROM_ACTIVITY
  - EXPLAIN_FROM_CATALOG
  - EXPLAIN_FROM_DATA
  - EXPLAIN_FROM_SECTION
    - either in memory, catalogs, captured by event monitor, directly inputted



41

Starting with Version 9.7 Fix Pack 1, DB2 database manager has the ability to perform an explain directly from the contents of a runtime section. This functionality is known as a section explain. The advantage of a section explain over a traditional explain using an EXPLAIN statement is that the EXPLAIN statement will recompile the statement that is being explained. If the compilation environment or table statistics have changed when the EXPLAIN statement is issued, the compiler may generate a different access plan.

A section explain will always provide the exact access plan that was executed, since the access plan is reconstructed directly from the executable section. A section explain is similar to the functionality provided by the db2expln command, but provides a level of detail approaching that which is provided by the EXPLAIN statement.

The section explain functionality is accessible through a set of stored procedures. The stored procedures take input parameters that are used to locate a section (either in memory, catalogs, captured by an event monitor, or provided directly as input) and perform the explain, populating the explain tables similar to the EXPLAIN statement. The stored procedures output the key fields for the explain instance that was populated in the explain tables. These key fields can be used as input to existing explain formatting tools, for example db2exfmt, which extract the information from the explain tables and present it in a formatted output.

The section explain functionality captures (either directly or via tooling) explain information about a statement using only the contents of the runtime section. The section explain is similar to the functionality provided by the db2expln command, but the section explain gives a level of detail approaching that which is provided by the explain facility.

By explaining a statement using the contents of the runtime section, you can obtain information and diagnostics about what will actually be run (or was run, if the section was captured after execution), as opposed to issuing an EXPLAIN statement which might produce a different access plan (for example, in the case of dynamic SQL, the statistics might have been updated since the last execution of the statement resulting in a different access plan being chosen when the EXPLAIN statement compiles the statement being explained).

The section explain interfaces will populate the explain tables with information that is similar to what is produced by an EXPLAIN statement. However, there are some differences. After the data has been written to the explain tables, it may be processed by any of the existing explain tools you want to use (for example, the db2exfmt command).

Notes: For Reference Only

There are four interface procedures, in the following list, that can perform a section explain. The procedures differ by only the input that is provided (that is, the means by which the section is located):

EXPLAIN_FROM_ACTIVITY

Takes application ID, activity ID, uow ID, and activity event monitor name as input. The procedure searches for the section corresponding to this activity in the activity event monitor (an SQL activity is a specific execution of a section). A section explain using this interface contains section actuals because a specific execution of the section is being performed.

EXPLAIN_FROM_CATALOG

Takes package name, package schema, unique ID, and section number as input. The procedure searches the catalog tables for the specific section.

EXPLAIN_FROM_DATA

Takes executable ID, section, and statement text as input.

EXPLAIN_FROM_SECTION

Takes executable ID and location as input, where location is specified by using one of the following:

In-memory package cache

Package cache event monitor name

The procedure searches for the section in the given location.

An executable ID uniquely and consistently identifies a section. The executable ID is an opaque, binary token generated at the data server for each section that has been executed. The executable ID is used as input to query monitoring data for the section, and to perform a section explain.

In each case, the procedure performs an explain, using the information contained in the identified runtime section, and writes the explain information to the explain tables identified by an explain_schema input parameter. It is the responsibility of the caller to perform a commit after invoking the procedure.

Explain Enhanced with Actual Operator Cardinality Values

- Explain facility output displays both section actuals and estimated access plan values for your comparison. Stored in EXPLAIN_ACTUALS table.
  - can point to out-of-date statistics used by optimizer to select incorrect access plan. Can update statistics using RUNSTATS and retry application with up-to-date access plan in place.
- Section actuals currently only available when section explain is performed and section was captured using an activity event monitor (EXPLAIN_FROM_ACTIVITY procedure)
- Section actuals must be explicitly enabled (set to BASE) using **section_actuals** db cfg parameter.
  - cannot be enabled if auto_stats_prof enabled (SQLCODE -5153).
- The ability to collect section actuals information can help to resolve SQL query performance slow downs.

40

Starting with Version 9.7 Fix Pack 1, runtime statistics can be gathered for access plan operators during the execution of a section. These statistics are referred to as section actuals. In Fix Pack 1, the only statistic available is cardinality for access plan operators.

The explain facility output conveniently displays both the section actuals and estimated access plan values for your comparison. The result of this comparison can point to out-of-date statistics used by the optimizer to select an incorrect access plan. Action can then be taken to update the statistics using the RUNSTATS command and then retrying the application with an up-to-date access plan in place. Section actuals are only available when a section explain is performed and the section was captured using an activity event monitor.

Note: Section actuals must be explicitly enabled (set to BASE) using the section_actuals database configuration parameter. Section actuals cannot be enabled if automatic statistics profile generation (auto_stats_prof) is enabled in the database configuration (SQLCODE -5153).

The ability to collect section actuals information can help to resolve SQL query performance slow downs.

Capturing Section Actuals Information (1) FP1

1) Enable Section Actuals
update db cfg for SAMPLE using SECTION_ACTUALS BASE;

2) Create Workload and/or Event Monitor
create workload MYWORKLOAD current client_acctng('MYWORKLOAD') service class sysdefaultuserclass collect activity data on all database partitions with details, section;
grant usage on workload MYWORKLOAD to public;
call wlm_set_client_info(null, null, null, 'MYWORKLOAD', null);

create event monitor MYMON for activities write to table;
set event monitor MYMON state 1;

3) Collect Section Actuals for Statement of Interest
delete from ACTIVITYSTMT_MYMON;
select * from staff where dept=20;
set event monitor MYMON state 0;
call wlm_set_client_info(null, null, null, null, null);

43

Section actuals are runtime statistics collected during the execution of the section for an access plan. To capture a section with actuals, you use the activity event monitor. To access the section actuals, you perform a section explain using the EXPLAIN_FROM_ACTIVITY stored procedure. To be able to view section actuals, you must perform a section explain on a section for which section actuals were captured (that is, both the section and the section actuals are the inputs to the explain facility). Information about enabling, capturing, and accessing section actuals is provided here.

Section actuals will only be updated at runtime if they have been enabled. Enable section actuals using the section_actuals database configuration parameter. To enable section actuals, set the parameter to BASE (the default value is NONE). For example:db2 update database configuration using section_actuals base. The setting of the section_actuals database configuration parameter that was in effect at the start of the unit of work is applied to all statements in that unit of work. When the section_actuals database configuration parameter is changed dynamically, the new value will not be seen by an application until the next unit of work. Section actuals cannot be enabled if automatic statistics profile generation is enabled (SQLCODE -5153).

Reference Only: Student Notes

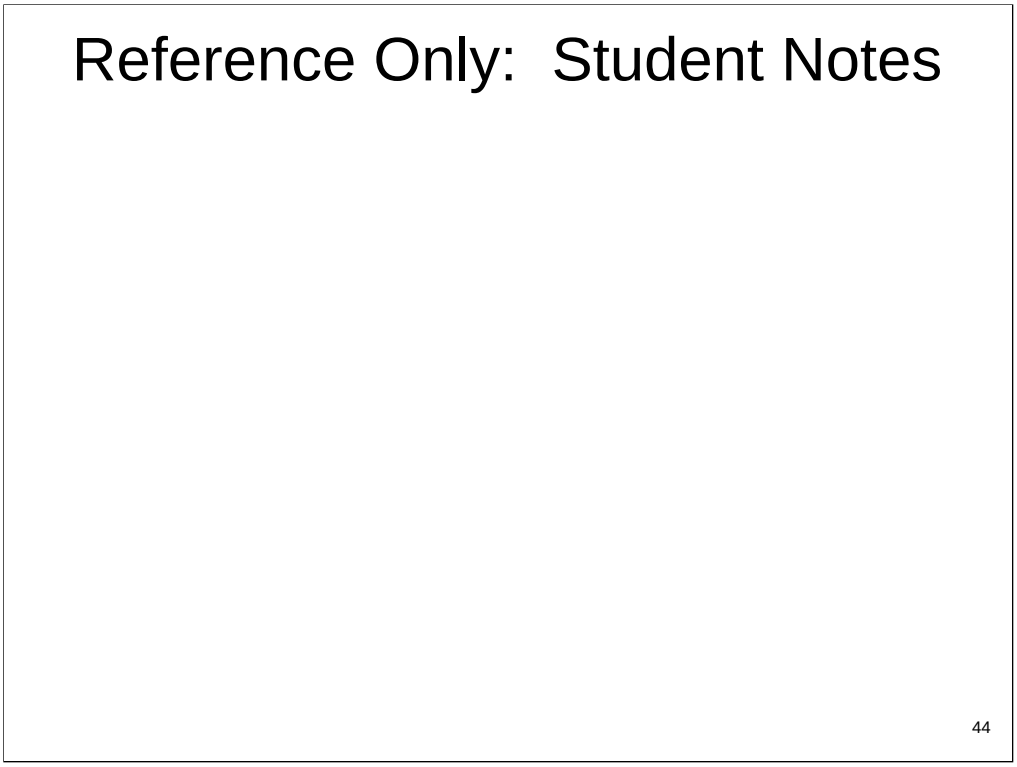
44

The mechanism for capturing a section, with section actuals, is the activity event monitor. An activity event monitor writes out details of an activity when the activity completes execution, if collection of activity information is enabled. Activity information collection is enabled using the COLLECT ACTIVITY DATA clause on a workload, service class, threshold, or work action. To specify collection of a section and actuals (if the latter is enabled), the SECTION option of the COLLECT ACTIVITY DATA clause is used. For example, the following statement indicates that any SQL statement, issued by a connection associated with the WL1 workload, will have information (including section and actuals) collected by any active activity event monitor when the statement completes: ALTER WORKLOAD WL1 COLLECT ACTIVITY DATA WITH DETAILS,SECTION In a partitioned database environment, section actuals are captured by an activity event monitor on all partitions where the activity was executed, if the statement being executed has a COLLECT ACTIVITY DATA clause applied to it and the COLLECT ACTIVITY DATA clause specifies both the SECTION keyword and the ON ALL DATABASE PARTITIONS clause. If the ON ALL DATABASE PARTITIONS clause is not specified, then actuals are captured on only the coordinator partition. Section actuals can be accessed using the EXPLAIN_FROM_ACTIVITY procedure. When you perform a section explain on an activity for which section actuals were captured, the EXPLAIN_ACTUALS explain table will be populated with the actuals information.Note: Section actuals are only available when a section explain is performed using the EXPLAIN_FROM_ACTIVITY procedure. The EXPLAIN_ACTUALS table is the child table of the existing EXPLAIN_OPERATOR explain table. When EXPLAIN_FROM_ACTIVITY is invoked, if the section actuals are available, the EXPLAIN_ACTUALS table will be populated with the actuals data. If the section actuals are collected on multiple database partitions, there is one row per database partition for each operator in the EXPLAIN_ACTUALS table. To resolve a SQL query performance slow down, you can begin by obtaining a section explain that includes section actuals information. The section actuals values can then be compared with the estimated access plan values generated by the optimizer to assess the validity of the access plan. This task takes you through the process of obtaining section actuals to investigate poor query performance.

The CURRENT CLIENT_ACCTNG (or CLIENT ACCTNG) special register contains the value of the accounting string from the client information specified for this connection. The data type of the register is VARCHAR(255). The default value of this register is an empty string. The value of the accounting string can be changed by using the Set Client Information (sqleseti) API. Note that the value provided via the sqleseti API is in the application code page, and the special register value is stored in the database code page. Depending on the data values used when setting the client information, truncation of the data value stored in the special register may occur during code page conversion.

Example: Get the current value of the accounting string for this connection. **VALUES** (CURRENT CLIENT_ACCTNG) **INTO** :ACCT_STRING

The WLM_SET_CLIENT_INFO procedure sets client information associated with the current connection at the DB2 server. By using this procedure, you can set the client's user ID, application name, workstation name, accounting information, or workload information at the DB2 server.



Calling this procedure changes the stored values of the relevant transaction processor (TP) monitor client information fields and special register settings for this connection.

The client information fields are used at the DB2 server for determining the identity of the application or user currently using the connection. The client information fields for a connection are considered during DB2 workload evaluation and also displayed in any DB2 audit records or application snapshots generated for this connection.

Unlike the sqleseti API, this procedure does not set client information at the client but instead sets the corresponding client attributes on the DB2 server. Therefore, you cannot use the sqleqry API to query the client information that is set at the DB2 server using this procedure.

The data values provided with the procedure are converted to the appropriate database code page before being stored in the related TP monitor fields or special registers. Any data value which exceeds the maximum supported size after conversion to the database code page is truncated before being stored at the server. The truncated values are returned by both the TP monitor fields and the special registers when those stored values are queried.

The WLM_SET_CLIENT_INFO procedure is not under transaction control, and client information changes made by the procedure are independent of committing or rolling back units of work. However, because workload reevaluation occurs at the beginning of the next unit of work for each application, you must issue either a COMMIT or a ROLLBACK statement to make client information changes effective.

WLM_SET_CLIENT_INFO (*client_userid*, *client_wrkstnname*, *client_applname, client_acctstr, client_workload*)

Capturing Section Actuals Information (2) FP1

4) Locate the Application, UOW and Activity ID for the data

```
select appl_id, uow_id, activity_id, substr(stmt_text,1,80)
        as stmt from ACTIVITYSTMT_MYMON;
```

APPL_ID	UOW_ID	ACTIVITY_ID	STMT
*LOCAL.skapoor.100505151716	11	1	select * from staff where dept=20

5) Explain the data into Explain tables

```
call explain_from_activity
('*LOCAL.skapoor.100505151716',11,1,'MYMON','SKAPOOR',?,?,?,?,?)
```

6) Run db2exfmt to generate access plan

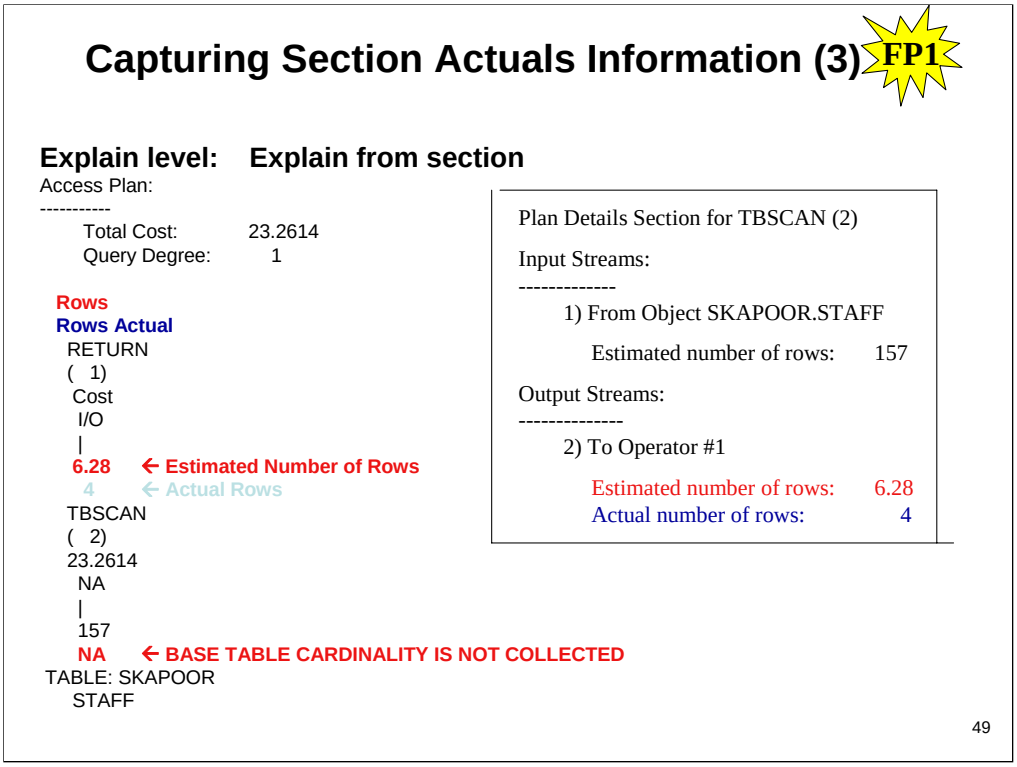
```
db2exfmt –d sample -1 –o exfmt_activity.out
```

7) Examine the Output

46

Output from Explaining the data into Explain tables:

```
Value of output parameters
--------------------------
Parameter Name  : EXPLAIN_SCHEMA
Parameter Value : SKAPOOR
Parameter Name  : EXPLAIN_REQUESTER
Parameter Value : SKAPOOR
Parameter Name  : EXPLAIN_TIME
Parameter Value : 2010-05-05-11.20.14.668725
Parameter Name  : SOURCE_NAME
Parameter Value : SQLC2H20
```



Limitations

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

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

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

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

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

WLM Enhancements

FP1

- Work action sets can be defined at workload level
  - Can monitor and control activities submitted by specific application without having to map those activities to distinct service class
- New time threshold limits unit of work duration
  - **UOWTOTALTIME** threshold specifies maximum that can elapse from time UOW first becomes active.
  - Previously, had to use Governor to restrict UOW time to a duration.
  - Define at workload, service superclass, and database domains

- **Automatic WLM statistics collection synchronized to a fixed base time**
  - 9.5 collection interval controlled using wlm_collect_int db cfg parm
  - New implementation has intervals starting from fixed base of Sunday 00:00
  - E.g. Interval is 1 hour, catalog partition starts at 3:35 PM
  - Old implementation would collect next interval at 4:35 PM
  - New implementation will collect next interval at 4:00 PM
- Script facilitates migration from Query Patroller to WLM
  - Sample script (**qpwlmmig.pl**) generates file which contains DDL to create WLM objects that most closely reflect current QP setup
  - Can continue to use same QP system control approach until determine how to best use WLM capabilities

49

Starting with Version 9.7 Fix Pack 1, you can define work action sets at the workload level to control activities submitted by occurrences of those workloads, based on the type and size of work. The option to control work at the workload level complements the control options already available at the service superclass and database levels. A work action set, defined on a workload definition, applies to all work submitted by connections that are currently mapped to that workload definition.

With the ability to define work action sets at the workload level, you can monitor and control activities submitted by a specific application without having to map those activities to a distinct service class. Control of incoming work includes the application of activity thresholds to activities submitted by occurrences of the workload, as well as the ability to introduce a concurrency threshold on some or all of that same work.

The following list contains the types of work actions that are available when a work action set is applied at the workload level:

•COUNT ACTIVITY
•PREVENT EXECUTION
•COLLECT ACTIVITY DATA
•COLLECT AGGREGATE ACTIVITY DATA

Thresholds that apply to each individual activity in the matching work class:

- •ESTIMATEDSQLCOST
- •SQLROWSRETURNED
- •ACTIVITYTOTALTIME
- •SQLTEMPSPACE
- •SQLROWSREAD
- •CPUTIME

CONCURRENTDBCOORDACTIVITIES threshold that applies to all activities as a group in the matching work class. This threshold controls the number of concurrent activities in the matching work class from all workload occurrences in the workload.

Starting with Version 9.7 Fix Pack 1, you can use the UOWTOTALTIME threshold to specify the maximum amount of time that can elapse from the time that a unit of work first becomes active. Previously, you had to use the DB2 governor to restrict a unit of work to a specific duration. Occasionally, an application might start transactions that run longer than a desired amount of time, resulting in locks being held which prevent other more important applications from proceeding. The UOWTOTALTIME threshold triggers the termination of the long-running application or rolls back the transaction in favor of continuing progress with other work. You can define this new threshold at the workload, service superclass, and database domains of workload management.

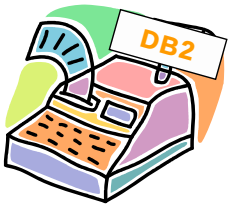
Starting with Version 9.7 Fix Pack 1, a sample script (qpwlmmig.pl) has been provided to facilitate the migration from the deprecated DB2 Query Patroller (QP) environment to the DB2 workload manager (WLM) environment. This script generates a file which contains the DDL statements to create the WLM objects that most closely reflect your current QP setup.

For the most part, you can continue to use the same system control approach that QP currently uses until you determine how best to use the WLM capabilities.

New Registry Variables

FP1

- DB2_MIN_IDLE_RESOURCES
  - Linux can specify that activated databases use minimal processing resources when dbm is idle. Useful for virtual Linux like zVM.



- DB2_USE_FAST_PREALLOCATION
  - Veritas fast allocation file system feature can be used to reserve table space, and speed up process of creating or altering large table spaces and database restore operations.
  - Might improve runtime performance, at cost of slower table space creation and database restore times, especially AIX, when there is a large volume of inserts and selects on same table space.
- DB2TCP_CLIENT_KEEPALIVE_TIMEOUT
  - Specify a keep alive setting that is lower than the system default, allowing the database manager to detect connection failures sooner
- DB2_WORKLOAD
  - INFOR_ERP_LN, configures registry variables for Infor ERP Baan

50

For Linux operating systems, users can specify that activated databases are to use minimal processing resources when the database manager is idle, using a new registry variable DB2_MIN_IDLE_RESOURCES. Default: OFF, Values: OFF or ON. This might be useful in some virtual Linux environments (for example, zVM) where the small resource savings can help the host virtual machine monitor schedule its CPU and memory resources across all its virtual machines more efficiently.

With the new DB2_USE_FAST_PREALLOCATION registry variable, the Veritas fast allocation file system feature can be used to reserve table space, and speed up the process of creating or altering large table spaces and database restore operations. This speed improvement is implemented at a small delta cost of performing actual space allocation during runtime when rows are inserted. To disable fast preallocation, set DB2_USE_FAST_PREALLOCATION to OFF. This might improve runtime performance, at the cost of slower table space creation and database restore times, on some operating systems, especially AIX, when there is a large volume of inserts and selects on same table space. Operating system: AIX, Linux and Solaris on VeritasVxFS or JFS2 file systems. Default: ON, Values: ON or OFF. Note that once fast preallocation is disabled, the database has to be restored.

With the new DB2TCP_CLIENT_KEEPALIVE_TIMEOUT registry variable, users can specify a keep alive setting that is lower than the system default, allowing the database manager to detect connection failures sooner. Operating system: AIX, Linux, Windows (client only). Default=0 (not set) Values: 0 - 32 767 seconds. Specifies the maximum time in seconds before an unresponsive connection is detected as no longer alive. When this variable is not set, the system default TCP/IP keep alive setting is used (typically two hours). Setting DB2TCP_CLIENT_KEEPALIVE_TIMEOUT to a lower value than the system default allows the database manager to detect connection failures sooner, and avoids the need to reconfigure the system default which would impact all TCP/IP traffic and not just connections established by DB2.

The DB2_WORKLOAD aggregate registry variable now has a new value, INFOR_ERP_LN, which configures a set of registry variables for Infor ERP Baan. For more information, see the "DB2_WORKLOAD" entry in System environment variables.

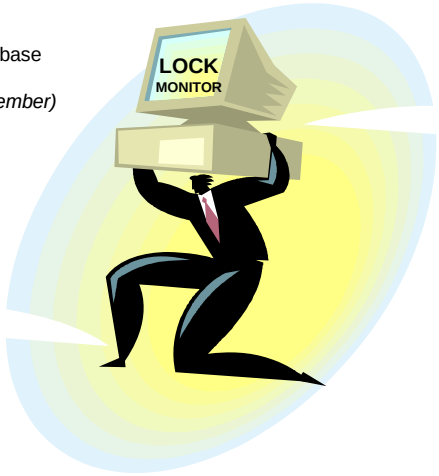
A New Monitoring Infrastructure

- **Lock Table Functions**
  - Direct access to internal locking information
  - Allow ad-hoc analysis of locking problems
- **Row-based Access for “In-memory” Metrics**
  - New format functions to produce generic row-based output for “time spent” metrics from XML documents
  - Can be used on table function output and event monitor XML columns

New Lightweight Lock Monitors

FP1

- MON_GET_APPL_LOCKWAITS (*application_handle*, *member)*
  - Table Function returns LOCK_WAIT_START_TIME, LOCK_NAME, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_CURRENT_MODE, LOCK_MODE_REQUESTED, LOCK_STATUS, LOCK_ESCALATION, LOCK_ATTRIBUTES, LOCK_RRIID, LOCK_COUNT, TBSP_ID, TAB_FILE_ID, SUBSECTION_NUMBER
- MON_GET_LOCKS (*application_handle*, *member)*
  - Table Function returns a list of all locks held on database
- MON_FORMAT_LOCK_NAME (*application_handle*, *member)*
  - Table Function formats internal lock name and returns details about the lock in a row-based format. Each row consists of a key-value pair pertaining to a particular lock.
- MON_LOCKWAITS
  - View returns information about agents working on behalf of applications that are waiting to obtain locks in currently connected database.
- Deprecated:
  - SNAPLOCK, SNAPLOCKWAIT, LOCKS_HELD, LOCKWAITS views



53

Starting with Version 9.7 Fix Pack 1, the MON_GET_APPL_LOCKWAITS, MON_GET_LOCKS, and MON_FORMAT_LOCK_NAME relational monitoring interfaces can be used to collect locking event data to help you quickly identify locking issues that might be causing problems.

The monitoring interfaces are more efficient and have a lower impact on the system than existing snapshot interfaces. These new interfaces report monitoring elements related to locking events. Use the following routines to collect information about locks:

•MON_GET_APPL_LOCKWAITS - Returns information about the locks that all the applications are waiting to acquire on the currently connected database.

•MON_GET_LOCKS - Returns a list of all locks on the currently connected database.

•MON_FORMAT_LOCK_NAME - Formats the internal lock name and returns details about the lock in a row-based format. Each row consists of a key-value pair pertaining to a particular lock.

Use the following administrative view to collect lock wait information:

•MON_LOCKWAITS - Returns information about agents working on behalf of applications that are waiting to obtain locks in the currently connected database. It is a useful query for identifying locking problems.

Detailed Monitoring Table Functions

- MON_FORMAT_XML_WAIT_TIMES_BY_ROW
  - Returns a list of wait time monitor elements, such as total_wait_time and lock_wait_time for each XML document.
- MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW
  - Returns a list of component time monitor elements, including processing time monitor elements, such as total_compile_time, total_compile_proc_time, and stmt_exec_time, for each XML document.
- MON_FORMAT_XML_METRICS_BY_ROW
  - Returns all the metrics contained in the XML document.
- **Functions format metrics from any of the XML output documents as produced from:**
  - MON_GET_*_DETAILS table functions
  - EVMON_FORMAT_UE_TO_XML table function
    - Contents of METRICS column (XML document) in relation tables produced by EVMON_FORMAT_UE_TO_TABLES
  - Activity or statistics event monitor XML columns

57

In Version 9.7 Fix Pack 1 and later fix packs, monitor elements reported in XML documents can be displayed and analyzed in a generic fashion using new row-based formatting table functions. Detailed monitoring table functions, such a MON_GET_WORKLOAD_DETAILS, return an XML document called DETAILS, containing a number of detailed monitor elements. In addition, the statistics event monitor returns a DETAILS XML document, the activity event monitor returns a DETAILS_XML XML document, the EVMON_FORMAT_UE_TO_XML table function returns an XMLREPORT XML document, and the EVMON_FORMAT_UE_TO_TABLES procedure returns a METRICS XML document. You can review and analyze the monitor elements returned in these XML document by using the new row-based formatting table functions. Which monitor elements are returned depends on the table function or event monitor which produced the XML document.

Example: MON_FORMAT_XML_WAIT_TIMES_BY_ROW

```
SELECT SUBSTR(TFXML.WORKLOAD_NAME, 1, 25) AS
    WORKLOAD_NAME, SUBSTR(WAITS.METRIC_NAME, 1, 25) AS
    METRIC_NAME, WAITS.TOTAL_TIME_VALUE_TIME, WAITS.COUNT
  FROM TABLE( MON_GET_WORKLOAD_DETAILS( NULL, -2 ) ) AS
    TFXML, TABLE( MON_FORMAT_XML_WAIT_TIMES_BY_ROW(
  TFXML.DETAILS )) AS WAITS ORDER BY WAITS.TOTAL_TIME_VALUE
    DESC
```

WORKLOAD_NAME	METRIC_NAME	TOTAL_TIME_VALUE	COUNT
PAYROLL	CLIENT_IDLE_WAIT_TIME	2193672	174
FINANCE	CLIENT_IDLE_WAIT_TIME	738290	16
PAYROLL	DIRECT_READ_TIME	67892	81
FINANCE	DIRECT_READ_TIME	32343	8
FINANCE	LOCK_WAIT_TIME	8463	3
PAYROLL	LOCK_WAIT_TIME	55	1

…

Time Spent Metrics: Component Times

- **Another category of "time spent" metric being introduced:**
  - Component times measure the time spent processing within DB2 within a particular component area

- **Components represent major processing areas or stages within DB2 such as query compilation, section execution, and commit / rollback time**
  - When combined with wait times, reflect comprehensive coverage of where time is being spent within DB2

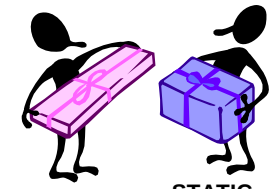
- **Like Wait times, component times:**
  - Are available in both system and activity perspectives
  - Designed with no overlap to allow comprehensive breakdown of component time in DB2

- **Total Request Time in DB2: Combined Wait + Component Times**
  - **Direct I/O**
  - **Bufferpool I/O**
  - **Lock Wait Time**
  - **Compile Proc Time**
  - **Section Proc Time**
  - **Commit / Rollback**
  - **Proc Time**
  - **Other Proc Time**

Event Monitor for Package Cache

FP1

- CREATE EVENT MONITOR FOR PACKAGE CACHE
  - **Records events from both dynamic and static SQL when they are flushed from package cache**
  - **Entries begin to be captured as soon as event monitor activated**
  - Granularity is defined on WHERE clause in event monitor definition
    - Number of executions, overall aggregate execution time, updated since last boundary time set via MON_GET_PKG_CACHE_STMT
  - Level of information captured controlled by event monitor definition:
    - BASE: Statement text, compilation environment, execution metrics
    - EXTENDED: BASE plus section
  - **Uses new UE Table target type**
- Can view the information from event monitor as
  - An XML document created by the new EVMON_FORMAT_UE_TO_XML table function
  - Relational tables populated by the new EVMON_FORMAT_UE_TO_TABLES procedure



51

Starting with Version 9.7 Fix Pack 1, the package cache event monitor (CREATE EVENT MONITOR FOR PACKAGE CACHE) records events from both dynamic and static SQL statements when they are flushed from the database package cache.

The new package cache event monitor captures information about cached statement entries after they have been flushed from the database package cache. The event monitor captures an accurate history about statements, that were in the package cache, which can help to resolve SQL query performance and problem determination issues.

The core data collected for a package cache event are the monitor elements reported through the MON_GET_PKG_CACHE_STMT table function. In addition, the event monitor collects information about the executable section of the activity. The collected information is the same for both dynamic and static SQL statements.

After the event monitor data has been captured, the following is a list of the methods you can use to access that data:

•An XML document created by the new EVMON_FORMAT_UE_TO_XML table function

•Relational tables populated by the new EVMON_FORMAT_UE_TO_TABLES procedure

•An XML or text document using the Java™-based db2evmonfmt tool

MON_GET_PKG_CACHE_STMT_DETAILS FP1

```
SELECT SUBSTR(DETMETRICS.STMT_TEXT, 1, 40) STMT_TEXT,
    DETMETRICS.ROWS_RETURNED,  DETMETRICS.STMT_EXEC_TIME
  FROM TABLE (MON_GET_PKG_CACHE_STMT_DETAILS
  (CAST(NULL AS CHAR(1)),
   CAST(NULL AS VARCHAR(32) FOR BIT DATA),
   CAST(NULL AS CLOB(1K)), -1)) AS STMT_METRICS,
     XMLTABLE (XMLNAMESPACES
     (DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
     '$DETMETRICS/db2_pkg_cache_stmt_details'
   PASSING  XMLPARSE (DOCUMENT STMT_METRICS.DETAILS)
     as "DETMETRICS"
  COLUMNS "STMT_TEXT" CLOB PATH 'stmt_text',
  "ROWS_RETURNED"  BIGINT PATH 'activity_metrics/rows_returned',
    "STMT_EXEC_TIME" BIGINT PATH 'activity_metrics/stmt_exec_time' )
     AS DETMETRICS
   ORDER BY rows_returned DESC
   FETCH FIRST 10 ROWS ONLY
```

- Accumulation of all metrics for statements in the package cache.

58

The MON_GET_PKG_CACHE_STMT_DETAILS table function returns detailed metrics for one or more package cache entries.

Note: If your database was created in Version 9.7 prior to Fix Pack 1, to run this routine you must have already run the db2updv97 command. If your database was created before Version 9.7, it is not necessary to run the db2updv97 command (because the catalog update is automatically taken care of by the database migration). If you downgrade to Version 9.7, this routine will no longer work.

The metrics returned by the MON_GET_PKG_CACHE_STMT_DETAILS table function represent the accumulation of all metrics for statements in the package cache. Statement metrics are rolled up to the package cache upon activity completion.

Syntax

>>-MON_GET_PKG_CACHE_STMT_DETAILS--(--*section_type*--,---------->>--*executable_id*--,--*search_args*--,--*member*--)--------->< The schema is SYSPROC.

Table function parameters

section_type

An optional input argument (either "D" or "S") of type CHAR(1) that specifies information type for the returned statement. If the argument is NULL or an empty string, information is returned for all SQL statements. Not case sensitive: "D" stands for dynamic; "S" for static.

executable_id

An optional input argument of type VARCHAR (32) for bit data that specifies a unique section of the database package cache. If a null value is specified, information is returned for all SQL statements. When the *executable_id* is specified, the *section_type* argument is ignored. For example, if an *executable_id* is specified for a dynamic statement, the dynamic statement details will be returned by this table function even if *section_type* is specified as static ("S").

search_args

An optional input parameter of type CLOB(1K), that allows you to specify one or more optional search argument strings. For example:

'<modified_within>5</modified_within><update_boundary_time>myPkgEvmon </update_boundary_time>' The available search argument tags are as follows:

'<modified_within>X</modified_within>'

Returns only those statement entries that have either been inserted into the cache or executed within the last *X* minutes (where *X* is a positive integer value). If the argument is not specified, all entries in the cache are returned.

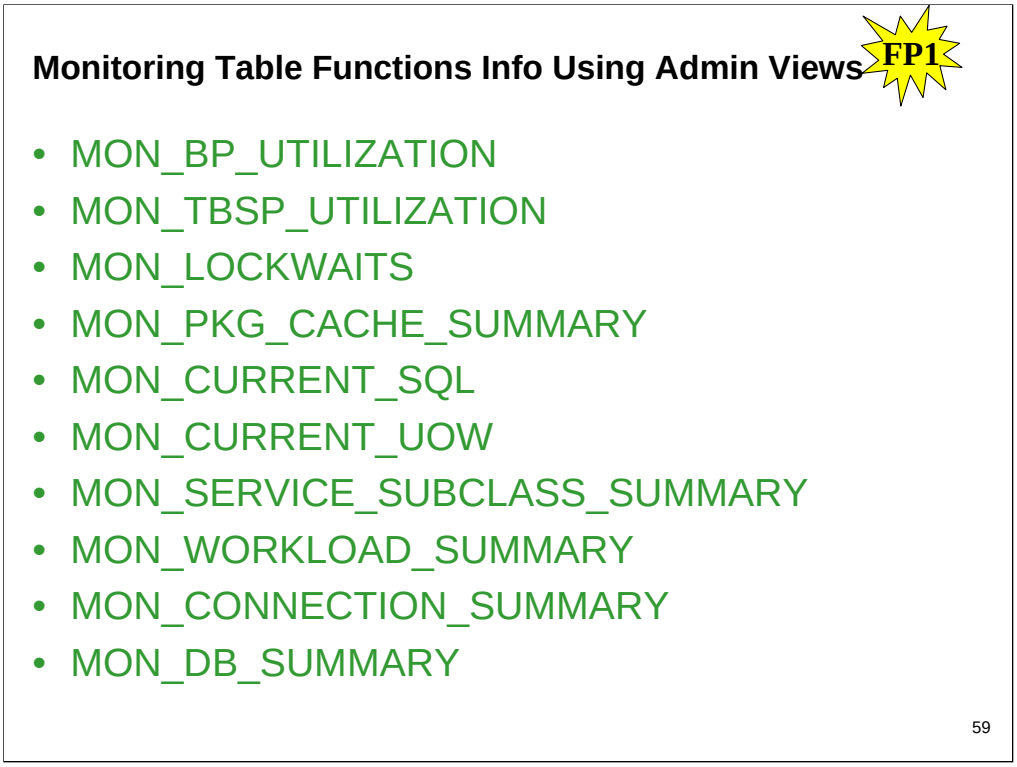
'<update_boundary_time>evmon_name</update_boundary_time>'

Updates the event monitor boundary timestamp to the current time for the package cache event monitor specified by *evmon_name*. If this event monitor specifies where updated_since_boundary_time as an output criteria in its WHERE clause, only package cache entries that subsequently have their metrics updated are captured when evicted from the package cache. This operation only has an effect if the specified package cache event monitor is active when the command is issued.

'<stmt_details>true</stmt_details>' or '<stmt_details>false</stmt_details>'

Includes or excludes the *stmt_text* and *comp_env_desc* data in the resulting XML document. This allows you to exclude these relatively large portions of the document when you do not need them (for example, if you are using the XML document to provide input for the MON_FORMAT_XML_* table functions that return formatted row-based output). If this argument tag is not specified, the *stmt_text* and *comp_env_desc* data are included by default.

Each input argument can be specified only once. The search argument tags must be specified in lowercase.



New administrative views encapsulate key queries using the new monitoring table functions introduced in DB2 9.7 and 9.7 Fix Pack 1.

The new monitoring table functions introduced in DB2 9.7 and 9.7 Fix Pack 1 provide many detailed metrics describing the database objects and environment. To see the most important metrics in an easily readable format, you can use the new monitoring administrative views. You can simply issue a SELECT * command to see the main metrics from each table function, as well as some common calculated values.

MON_DB_SUMMARY - TOTAL_APP_COMMITS, TOTAL_APP_ROLLBACKS, ACT_COMPLETED_TOTAL, APP_RQSTS_COMPLETED_TOTAL, AVG_RQST_CPU_TIME, ROUTINE_TIME_RQST_PERCENT, RQST_WAIT_TIME_PERCENT, ACT_WAIT_TIME_PERCENT, IO_WAIT_TIME_PERCENT, LOCK_WAIT_TIME_PERCENT, AGENT_WAIT_TIME_PERCENT, NETWORK_WAIT_TIME_PERCENT

CREATE EVENT MONITOR FOR UNIT OF WORK

- Enhanced replacement to deprecated CREATE EVENT MONITOR FOR TRANSACTIONS
- Determine how much to charge application users based on amount of resources used by application
- Also collect listing of packages used within each UOW, including nesting level and elapsed time for each package. FP1
- Collected for each invocation of a routine which facilitates stored procedure troubleshooting. FP1

60

The new unit of work event monitor (CREATE EVENT MONITOR FOR UNIT OF WORK) is an enhanced replacement to the deprecated transaction event monitor (CREATE EVENT MONITOR FOR TRANSACTIONS). The new unit of work event monitor contains many additional monitor elements and is more efficient than the transaction event monitor.

A common use for the new unit of work event monitor would be, as a data server provider, to determine how much to charge application users based on the amount of resources used by the application. In such billing circumstances, total CPU usage is the most commonly used resource upon which to base chargeback billing. Total CPU usage is one of the monitor elements for which data is collected in the new unit of work event monitor.

The core data collected for a unit of work event are the monitor elements reported through the MON_GET_UNIT_OF_WORK and MON_GET_UNIT_OF_WORK_DETAILS table functions. This data is enriched with a variety of information, including attributes at the database level, connection level, and unit of work level.

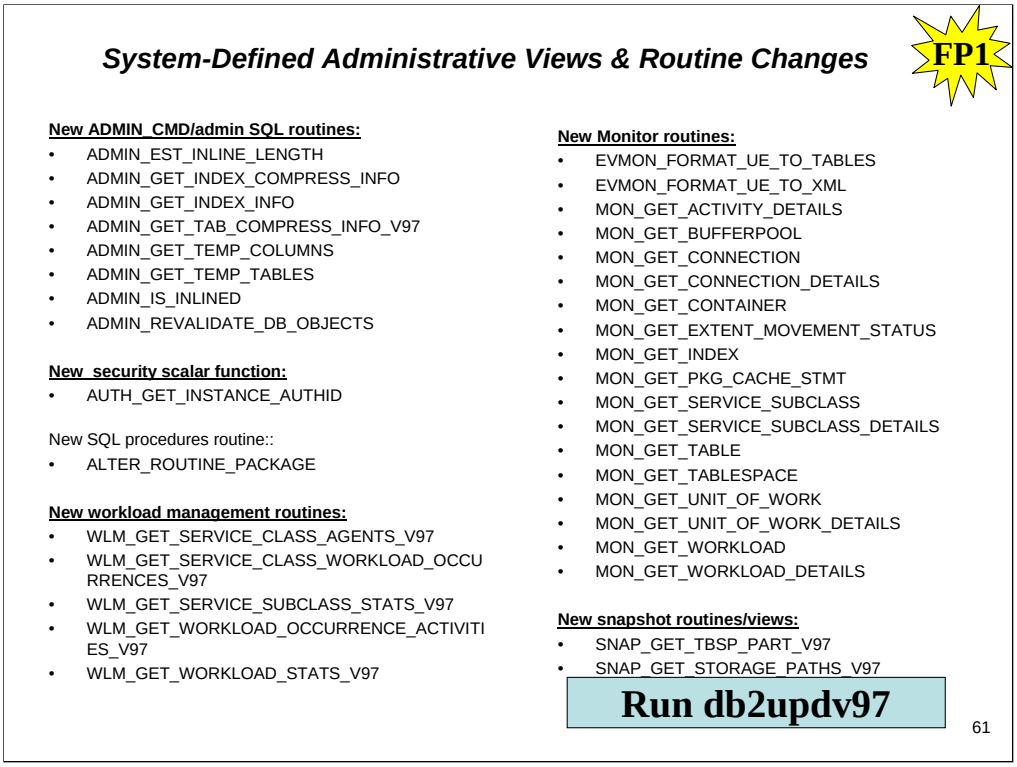
In Version 9.7 Fix Pack 1 and later fix packs, the unit of work event monitor can also collect a listing of packages used within each unit of work, including the nesting level and the elapsed time for each package. Unique information is collected for each invocation of a routine. The package listing information helps facilitate stored procedure troubleshooting.

After the unit of work event monitor data has been captured, you can access it using one of the following methods:

•An XML document created by the new EVMON_FORMAT_UE_TO_XML table function

•Relational tables populated by the new EVMON_FORMAT_UE_TO_TABLES procedure

•An XML or text document using the Java™-based db2evmonfmt tool



To have access to new administrative routines in Version 9.7 Fix Pack 1 in databases created in Version 9.7 prior to Fix Pack 1, you must have already run the db2updv97 command. If your database was created before Version 9.7, it is not necessary to run the db2updv97 command (because the system catalog is automatically updated by the database upgrade).

The following table functions have been deprecated in Version 9.7:

•HEALTH_CONT_HI
•HEALTH_CONT_HI_HIS
•HEALTH_CONT_INFO
•HEALTH_DB_HI
•HEALTH_DB_HI_HIS
•HEALTH_DB_HIC
•HEALTH_DB_HIC_HIS
•HEALTH_DB_INFO
•HEALTH_DBM_HI
•HEALTH_DBM_HI_HIS
•HEALTH_DBM_INFO
•HEALTH_GET_ALERT_ACTION_CFG
•HEALTH_GET_ALERT_CFG
•HEALTH_GET_IND_DEFINITION
•HEALTH_HI_REC
•HEALTH_TBS_HI
•HEALTH_TBS_HI_HIS
•HEALTH_TBS_INFO
•SNAP_GET_LOCK (deprecated starting in Version 9.7 Fix Pack 1)
•SNAP_GET_LOCKWAIT (deprecated starting in Version 9.7 Fix Pack 1)
•SNAP_GET_STORAGE_PATHS
•SNAP_GET_TBSP_PART_V91
•WLM_GET_ACTIVITY_DETAILS
•WLM_GET_SERVICE_CLASS_AGENTS
•WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
•WLM_GET_SERVICE_SUBCLASS_STATS
•WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES
•WLM_GET_WORKLOAD_STATS

The following administrative views have been deprecated in Version 9.7 Fix Pack 1:

•SNAPLOCK
•SNAPLOCKWAIT
•LOCKS_HELD
•LOCKWAITS

MONREPORT Module

MONREPORT provides a set of procedures for retrieving a variety of monitoring data and generating text reports.

- default monitoring interval of 10 seconds

call monreport.connection (*monitoring_interval*, *application_handle*)

System-defined routines available in the MONREPORT module:

- **CONNECTION** - presents monitor data for each connection.
- **CURRENTAPPS** - presents current instantaneous state of processing of units of work, agents, and activities for each connection. Starts with state information summed across connections, followed by a section for details for each connection.
- **CURRENTSQL** - lists top activities currently running, as measured by various metrics.
- **DBSUMMARY** - contains in-depth monitor data for entire database, as well as key performance indicators for each connection, workload, service class, and database member.
- **LOCKWAIT** - contains information about each lock wait currently in progress. Details include lock holder and requestor details, plus characteristics of the lock held and the lock requested.
- **PKGCACHE** - lists top statements accumulated in package cache as measured by various metrics.

Reports are implemented using SQL procedures within modules, and as such can be impacted by the package cache configuration. If you observe slow performance when running the reports, inspect your package cache configuration to ensure it is sufficient for your workload.

Example db2monreport.connection

db2 "call db2monreport.connection()”

…

…

```
------------------------------------------------------------------------------
Monitoring report by connection
------------------------------------------------------------------------------
Database: SAMPLE
Generated: 06/26/2009 00:28:23
Interval monitored: 10
-- Command options --
APPLICATION_HANDLE: All
```

…

```
------------------------------------------------------------------------------
     APPLICATION_    TOTAL_      TOTAL_      ACT_COMPLETED      TOTAL_WAIT_    CLIENT_IDLE
#    HANDLE          CPU_TIME    ACT_TIME    _TOTAL             TIME           WAIT_TIME
--   --------------- ----------- ----------- ------------------ -------------- ---------------
1    1207            0             0         0                  0                0
...
```

To generate a report that includes a section for each connection, with the default monitoring interval of 10 seconds, make the following call to the MONREPORT.CONNECTION procedure:
call monreport.connection()

To generate a report that includes a section only for the connection with application handle 32, with the default monitoring interval of 10 seconds, you can make either of the following calls to the MONREPORT.CONNECTION procedure:

call monreport.connection(DEFAULT, 32)

call monreport.connection(10, 32)

To generate a report that includes a section for each connection, with a monitoring interval of 60 seconds, you can make either of the following calls to the MONREPORT.CONNECTION procedure:

call monreport.connection(60)

call monreport.connection(60, null)

By default, the reports in this module are generated in English. To change the language in which the reports are generated, change the CURRENT LOCALE LC_MESSAGES special register. For example, to generate the CONNECTION report in French, issue the following commands:

SET CURRENT LOCALE LC_MESSAGES = 'CLDR 1.5:fr_FR'

CALL MONREPORT.CONNECTION

DB2 9.7 Fix Pack 2 Checklist

New Enhancements:

- DB2_RESTORE_GRANT_ADMIN_AUTHORITIES
- SELECT … WAIT FOR OUTCOME
- db2has Data Collector command
- RESTRICTED ACCESS restricts Database Connections within Quiesced Instance
- Transportable Schema
- WLM_SET_CONN_ENV() Procedure
- MON_GET_FCM & MON_GET_FCM_CONNECTION_LIST
- ADMIN_MOVE_TABLE Enhancements
- Auditing Capability allows SECADM Ability to Replay Past Database Activities

64

Version 9.7 Fix Pack 2 contains important changes that might affect your product usage. You should review the technical changes and new functionality included in Version 9.7 Fix Pack 2. For a summary of DB2 Version 9.7 LUW fix pack 2 changes, please see:

http://www-01.ibm.com/support/docview.wss?uid=swg24027906

DB2_RESTORE_GRANT_ADMIN_AUTHORITIES FP2

- If ON and restoring to new database, then SECADM, DBADM, DATAACESS and ACCESSCTRL authorities are granted to user that issues the restore operation**.**
- The following methods of restore are supported when DB2_RESTORE_GRANT_ADMIN_AUTHORITIES is set to ON:
  - Online and offline table space restore with RESTORE DATABASE command
  - Split mirror backups
  - ACS Snapshot backups
- If DB2_WORKLOAD = SAP, DB2_RESTORE_GRANT_ADMIN_AUTHORITIES=ON

5

If the DB2 registry variable DB2_RESTORE_GRANT_ADMIN_AUTHORITIES is set to ON, and you are restoring to a new database, then SECADM, DBADM, DATAACCESS, and ACCESSCTRL authorities are granted to the user that issues the restore operation.

DB2_RESTORE_GRANT_ADMIN_AUTHORITIES - Operating system: All; Default: OFF, Values: ON or OFF

If DB2_RESTORE_GRANT_ADMIN_AUTHORITIES is set to ON, and you are restoring to a new database, then SECADM, DBADM, DATAACCESS, and ACCESSCTRL authorities are granted to the user that issues the restore operation.

The following methods of restore are supported when DB2_RESTORE_GRANT_ADMIN_AUTHORITIES is set to ON:

•Online and offline table space restore with the RESTORE DATABASE command

•Split mirror backups

•ACS Snapshot backups

DB2_RESTORE_GRANT_ADMIN_AUTHORITIES is supported starting in DB2 Version 9.7 Fix Pack 2.

If DB2_WORKLOAD is set to SAP, DB2_RESTORE_GRANT_ADMIN_AUTHORITIES will be set to ON.

SELECT … WAIT FOR OUTCOME

- Indicates concurrent access resolution
- Specifies to wait for commit or rollback when encountering data in process of being updated, deleted, or inserted
- Settings for registry variables DB2_EVALUNCOMMITTED, DB2_SKIPDELETED, and DB2_SKIPINSERTED are ignored
- Applies when isolation level is CS or RS. Ignored when isolation level is UR or RR.
- Causes default behavior for currently committed to be overridden as well as any higher level setting such as bind options, CLI settings, JDBC settings, or lock modifications.

5

You can use the new WAIT FOR OUTCOME keyword in a SELECT statement to indicate the concurrent access resolution. WAIT FOR OUTCOME specifies to wait for the commit or rollback when encountering data in the process of being updated, deleted, or inserted.

WAIT FOR OUTCOME specifies to wait for the commit or rollback when encountering data in the process of being updated or deleted. Rows encountered that are in the process of being inserted are not skipped. The settings for the registry variables DB2_EVALUNCOMMITTED, DB2_SKIPDELETED, and DB2_SKIPINSERTED are ignored. This clause applies when the isolation level is CS or RS and is ignored when an isolation level of UR or RR is in effect. This clause causes the default behavior for currently committed that is defined by the cur_commit configuration parameter to be overridden as well as any higher level setting such as bind options, CLI settings, JDBC settings, or lock modifications.

db2has Data Collector command

FP2

- Collects DB2 health information about DB2 instance, databases, and operating environment and sends to DB2 Health Advisor Service at IBM for analysis and assessment.

  db2has -icn FC123456 **-name** "Fake 1 Company" **-address** "123 Main St., Anywhere, CA 99999**" -phone** "555-555-5555" **-email** "john.smith@fake1company.com" **-desc** "Insurance services" **-systype** test **-workload** OLTP **-send**

- Data collected for all databases activated on “test”. Priority can be set to lowest setting to minimize performance impact, which is normally negligible.
- Resulting compressed file, db2has_*hostname_timestamp*.zip, placed in ~/sqllib/db2hasdir and sent, by Enhanced Customer Data Repository (ECuRep), to DB2 Health Advisor Service
- Report with findings and recommendations will be sent to DBA John Smith using provided e-mail address

5

Intra-table space parallelism can reduce elapsed time for backups. Starting in Version 9.7 Fix Pack 2, intra-table space parallelism is introduced and can be used during backups other than delta and incremental backups. Intra-table space parallelism can reduce the time needed for a backup operation by allowing multiple threads to read the same table space in parallel during the whole backup operation.

When you use intra-table space parallelism, the table spaces are split into mutually exclusive page ranges. You enable and tune intra-table space parallelism using the DB2_ITP_LEVEL registry variable.

The environments that benefit the most from this enhancement are the ones in which the sizes of table spaces vary widely.

RESTRICTED ACCESS restricts Database Connections within Quiesced Instance

db2 QUIESCE INSTANCE inst1 USER mel GROUP group1 RESTRICTED ACCESS IMMEDIATE

- Prevents authorization checking for all connect attempts to databases of quiesced instance
- Used when need exclusive connections to database within quiesced instance
- When specify RESTRICTED ACCESS using QUIESCE INSTANCE or START DATABASE MANAGER, any user ID trying to connect to database within quiesced instance, which has DBADM authority or QUIESCE_CONNECT privilege, will not be allowed to connect
- Only SYSADM, SYSCTRL, or SYSMAINT and user or group specified with commands will be allowed to connect
- Use RESTRICTED ACCESS when need exclusive connections to database within quiesced instance
  - offline backup or performing other maintenance activities

The RESTRICTED ACCESS option can be specified to prevent authorization checking for all connect attempts to the databases of a quiesced DB2 instance. The new option can also be used when there is a need to have exclusive connections to a database within the quiesced instance. For more information, see FP2: New RESTRICTED ACCESS option restricts database connections within quiesced instance.
Starting with DB2® Version 9.7 Fix Pack 2, the new RESTRICTED ACCESS option can be specified to prevent authorization checking for all connect attempts to the databases of a quiesced DB2 instance. The new option can also be used when there is a need to have exclusive connections to a database within the quiesced instance.

When the RESTRICTED ACCESS option is specified using the QUIESCE INSTANCE or START DATABASE MANAGER commands, or the db2InstanceQuiesce or db2InstanceStart APIs, authorization checking is prevented to determine if the user ID has DBADM authority. Instance-level authorization checking can still occur; checking a user ID for SYSADM, SYSCTRL, or SYSMAINT authority does not require a database to be activated.

With the RESTRICTED ACCESS option specified, any user ID trying to connect to a database within a quiesced instance, which has DBADM authority or QUIESCE_CONNECT privilege on the database, will not be allowed to connect. Only user IDs which have SYSADM, SYSCTRL, or SYSMAINT authority and the user or group specified with the commands will be allowed to connect to the database.

You can use the RESTRICTED ACCESS option when there is a need to have exclusive connections to a database within the quiesced instance. Such cases can include making an offline backup or performing other maintenance activities.

Transportable Schema

FP2

- Ability to efficiently transport set of schemas (“transport set”) from one database to another
  - Regulatory compliance – convenient way of providing access to selected historical data via normal backups
  - Offloading operations (eg. create indexes on another computer before transporting schema to target)
- Use redirected restore
  - specify containers to use in target DB
  - specify transport set and target DB

- Restores catalog table space and table spaces required for specified transport set to a temporary database (using specified containers)
  - Rolls-forward table spaces to a point of consistency
  - Transfers ownership of table spaces and containers (except catalog) to target database
  - Recreates schema objects in target database

Transport Sets

• Schemas to be transported must be contained in table spaces that do not contain objects from other schemas.

You can use the RESTORE command with the TRANSPORT option to copy table spaces and SQL schemas as a set from a database backup image to another active database. Starting with DB2 V9.7 Fix Pack 2, table spaces and SQL schemas can be restored as a set from one database to another using transportable sets. You can also use the db2move command to move tables between DB2 databases.

By using the RESTORE command with the TRANSPORT option, you can restore data in a set of table spaces from a backup image into another existing database. You can re-create database objects in the SQL schemas that reference the data in the restored table spaces. The restored table spaces and SQL schemas can function as part of the new database.

You can also use this feature to simplify the process of restoring schemas from other database solutions to DB2 V9.7.

Note: When you transport table spaces, a log record with a special format is created on the target database. This format cannot be read by previous DB2 versions. If you transport table spaces and then downgrade to a version earlier than DB2 V9.7 Fix Pack 2, then you cannot recover the target database containing the table spaces that were transported. To ensure that the target database is compatible with earlier DB2 versions, you can roll forward the target database to a point in time before the transport operation.

What is Transported

- Tables / CGTTs / MQTs / 'normal + stats'
- Views
- Table Statistics + profiles
- Generated columns
  - Expression
  - Identity
  - Row change timestamp
  - Row change token
- UDFs / UDTs + generated functions
- Constraints
  - Check
  - Foreign key
  - Unique / primary
  - Functional dependency
- Indexes
- Triggers
- Sequences
- Procedure – not external routine executable
- Object authorizations
- Privileges
- Security / Access control / Audit definitions
- Packages

What is not Transported:

•Created global variables

•Aliases

•Jobs

•Index extensions

•Hierarchical tables, typed tables, typed views

•Nicknames, servers, wrappers

•Structured types

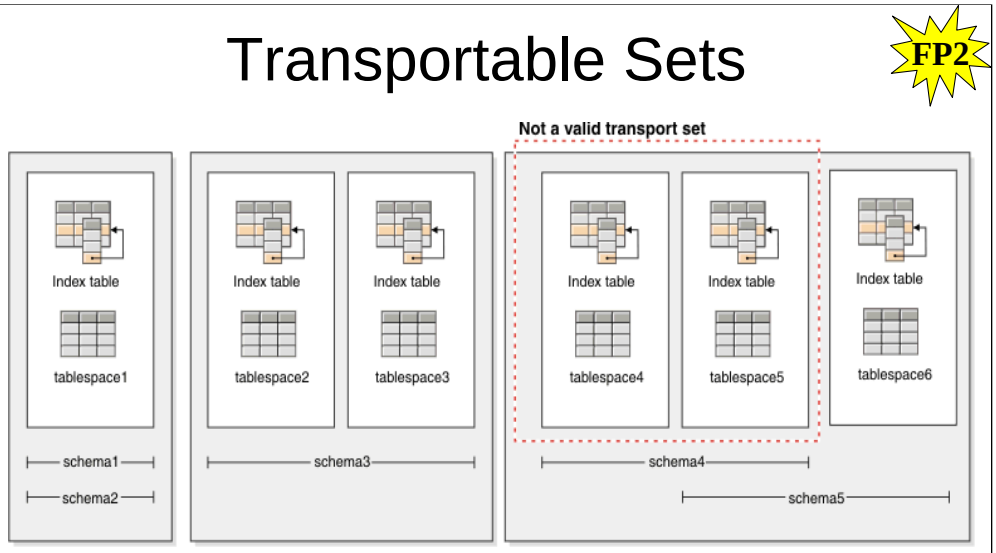
•Functional mappings & templates

•External functions and SP

•XML data

•DPF environments will not be supported

Transportable Sets



Following combinations of table spaces and schemas are valid ***transportable sets***:
- Tablespace1 with schema1 and schema2
- Tablespace2 and tablespace3 with schema3
- Tablespace4, tablespace5, and tablespace6, with schema4 and schema5
- Combination of valid transportable sets also constitutes a valid transportable set:
  - tablespace1, tablespace2, and tablespace3, with schema1, schema2, schema3

71

A database schema must be transported in its entirety. If a table space contains both the schema you want to transport, as well as another schema, you must transport all data objects from both schemas. These sets of schemas that have no references to other database schemas are called ***transportable sets***. The data in the table spaces and logical objects in the schemas in a transportable set reference only table spaces and schemas in the transportable set. For example, tables have table dependencies only on other tables in the transportable set.

The above diagram illustrates a database with several table spaces and schemas. In the diagram, the table spaces referenced by the schemas are above the schemas. Some schemas reference multiple table spaces and some table spaces are referenced by multiple schemas.

The following combinations of table spaces and schemas are valid transportable sets:

- tablespace1 with schema1 and schema2
- tablespace2 and tablespace3 with schema3
- tablespace4, tablespace5, and tablespace6, with schema4 and schema5
- A combination of valid transportable sets also constitutes a valid transportable set:
  - tablespace1, tablespace2, and tablespace3, with schema1, schema2, and schema3

The set tablespace4 and tablespace5 with schema4 is not a valid transportable set because there are references between tablespace5 and schema5 and between schema5 and tablespace6. The set requires tablespace6 with schema5 to be a valid transportable set.

RESTORE … TRANSPORT INTO … REDIRECT


```
RESTORE DB originaldb
TABLESPACE (“tablespace1”)
SCHEMA (“schema1”, “schema2”)
TRANSPORT INTO targetdb REDIRECT;

SET TABLESPACE CONTAINERS FOR 1
USING (PATH ‘targetdb/mydata1’);

RESTORE DB originaldb CONTINUE
```

Use redirected restore and specify containers to use in target database, transport set and target database.

DB2 restores catalog table space and table spaces required for specified transport set to a temporary database using specified containers, rolls-forward table spaces to a point of consistency, transfers ownership of table spaces and containers (except catalog) to target database, and recreates schema objects in target database.

WLM_SET_CONN_ENV() Procedure

- **A procedure that lets you set the collection attributes for a session without defining a workload**
  - I.e. the same capabilities as the COLLECT ACTIVITY clause in the DB2 Workload definition
  - Can set them for your own session or others
  - Can also turn on “section actuals” collection for a session
- **Complementary procedure**
  - WLM_GET_CONN_ENV() to read the current collection settings for a session

WLM_SET_CONN_ENV is a procedure that lets you set the collection attributes for a session without defining a workload.

MON_GET_FCM & MON_GET_FCM_CONNECTION_LIST

- Table functions collect FCM data to help identify communication issues
- These table functions are more efficient and have lower impact on system than existing snapshot interfaces
- MON_GET_FCM
  - Returns metrics for FCM
- MON_GET_FCM_CONNECTION_LIST
  - Returns monitor metrics for all the FCM connections on the specified member
- In addition, FCM-related metrics added to outputs of both db2pd command and GET SNAPSHOT command

Two new table functions, MON_GET_FCM and MON_GET_FCM_CONNECTION_LIST, improve the monitoring of fast communications manager (FCM).

Starting with V9.7 Fix Pack 2, the MON_GET_FCM and MON_GET_FCM_CONNECTION_LIST table functions can be used to collect fast communication manager (FCM) data to help you identify communication issues more easily. These table functions are more efficient and have a lower impact on the system than existing snapshot interfaces.

Use the following table functions to collect information about FCM:

•**MON_GET_FCM -** Returns metrics for FCM.

•**MON_GET_FCM_CONNECTION_LIST -** Returns monitor metrics for all the FCM connections on the specified member.

In addition, FCM-related metrics have been added to the outputs of both the db2pd command and the GET SNAPSHOT command

ADMIN_MOVE_TABLE Enhancements

- When using ADMIN_MOVE_TABLE procedure, LOAD_MSGPATH defines load message file path.
- FORCE option no longer needs to be specified with COPY_USE_LOAD option.
- LOAD_MSGPATH <path> defines load message file path when COPY_USE_LOAD option specified. If LOAD_MSGPATH is not specified, then diagpath will be used as default path.
- Restrictions on *DB2_SKIPDELETED* registry variable during online ADMIN_MOVE_TABLE are removed

When using the ADMIN_MOVE_TABLE procedure, you can use the new LOAD_MSGPATH option to define the load message file path. The FORCE option no longer needs to be specified with the COPY_USE_LOAD option.

LOAD_MSGPATH <path> can be used to define the load message file path when the COPY_USE_LOAD option specified. If the LOAD_MSGPATH option is not specified, then diagpath will be used as the default path. This option is available starting in DB2 V9.7 Fix Pack 2.

Auditing Capability allows SECADM Ability to Replay Past Database Activities

- To enable replay and analysis of any past activities, as SECADM, create audit policy that audits EXECUTE category and applies policy to database. EXECUTE category, when logged WITH DATA contains necessary information to replay past SQL statements, assuming data in database is restored to state it was when statement was issued.

  CREATE AUDIT POLICY STATEMENTS CATEGORIES EXECUTE WITH DATA STATUS BOTH ERROR TYPE AUDIT;

  AUDIT DATABASE USING POLICY STATEMENTS;

- Regularly archive audit log to create archive copy.

  CALL SYSPROC.AUDIT_ARCHIVE('/auditarchive', -2);

  -2 indicates archive should be run on all partitions

- Check that audit log files were created. Keep archived files for number of years specified by company's business policy.

  SELECT FILE FROM SESSION.AUDIT_ARCHIVE_RESULTS;

Auditing improvements have been added to allow for the replay of past database activities. DB2 V9.7 Fix Pack 2 adds auditing capability that gives security administrators the ability to replay past database activities. As part of a comprehensive security policy, a company might require that they retain the ability to retroactively go back a set number of years and analyze the effects of any particular request against certain tables in their database. To do this, they can institute a policy of archiving their weekly backups and associated log files such that they can reconstitute the database for any chosen moment in time. The database audit now captures sufficient information about every request made against the database to allow for the replay and analysis of any request against the relevant, restored database. This requirement covers both static and dynamic SQL statements.

As part of a comprehensive security policy, a company can require the ability to retroactively go back a set number of years and analyze the effects of any particular request against certain tables in their database.

Student Notes – For Reference Only

Before you begin

A company must institute a policy of archiving their weekly backups and associated log files such that they can reconstitute the database for any chosen moment in time.

About this task

To allow, at any future time, the replay and analysis of any request against the relevant, restored database, sufficient database audit information must be captured about every request made against the database. This requirement can cover both static and dynamic SQL statements. The EXECUTE category, when logged WITH DATA contains the necessary information to replay past SQL statements, assuming that the data in the database is restored to the state it was when the statement was issued.

Restrictions

The following authority and privileges are required:

•SECADM authority is required to create the audit policies,

•EXECUTE privilege is required for the audit routines and procedures.

DB2 9.7 Fix Pack 3/3a Checklist

New Enhancements:

- LOB strings with actual length < 32672 bytes supported as operands in other predicates and simple CASE expression.
- HP-UX 32-bit client support has been deprecated.
- Trap resilience functionality extended to Load.
- Support for AIX 7.1 operating system.
- Improved support for target storage devices that support data deduplication.
- Specify that system controller thread does not adjust resources below specific values by using FCM_CFG_BASE_AS_FLOOR option of DB2_FCM_SETTINGS registry variable.
- DB2 Text Search and Net Search Extender text indexes can now coexist on same table column.
- Range-clustered tables (RCT) are supported in DPF.
- db2caem (db2 Capture Activity Event Monitor data tool) created to simplify process of capturing detailed diagnostic and runtime information about one or more statements.
- Ability to trace only members (or partitions) specified and to trace based on a specific application ID (or application handle) with db2trc.
- New OLAP specification, RATIO_TO_REPORT, can provide ratio of a value compared to sum of a group of values.
- DB2 Connect enhancements in Fix Pack 3a

64

Version 9.7 Fix Pack 3 contains important changes that might affect your product usage. You should review the technical changes and new functionality included in Version 9.7 Fix Pack 3. For a summary of DB2 Version 9.7 LUW fix pack 3 changes, please see:

http://www-01.ibm.com/support/docview.wss?uid=swg24027906

FCM_CFG_BASE_AS_FLOOR option of DB2_FCM_SETTINGS FP3

- Can specify that system controller thread does not adjust resources below specific values by using FCM_CFG_BASE_AS_FLOOR option of DB2_FCM_SETTINGS registry variable.
- Can set DB2_FCM_SETTINGS with FCM_MAXIMIZE_SET_SIZE token to preallocate a default 2 GB FCM buffer. The token must have value of either YES or TRUE to enable this feature.
- Can now use FCM_CFG_BASE_AS_FLOOR option to set base value as the floor for *fcm_num_buffers* and *fcm_num_channels*
- When FCM_CFG_BASE_AS_FLOOR option = YES or TRUE, and *fcm_num_buffers* and *fcm_num_channels*=AUTOMATIC and have an initial or starting value, database manager will not tune them below this value.

You can specify that the system controller thread does not adjust resources below specific values by using the new FCM_CFG_BASE_AS_FLOOR option of the DB2_FCM_SETTINGS registry variable. For more information, see DB2_FCM_SETTINGS.

DB2_FCM_SETTINGS

Operating system: Linux

Default=YES, Values:

- •FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE]. The default value for FCM_MAXIMIZE_SET_SIZE is YES.
- •FCM_CFG_BASE_AS_FLOOR:[YES|TRUE|NO|FALSE]. The default value for FCM_CFG_BASE_AS_FLOOR is NO.

You can set the DB2_FCM_SETTINGS registry variable with the FCM_MAXIMIZE_SET_SIZE token to preallocate a default 2 GB of space for the fast communication manager (FCM) buffer. The token must have a value of either YES or TRUE to enable this feature.

In Version 9.7 Fix Pack 3 and later fix packs, you can set the DB2_FCM_SETTINGS registry variable with the FCM_CFG_BASE_AS_FLOOR option to set the base value as the floor for the *fcm_num_buffers* and *fcm_num_channels* database manager configuration parameters. When the FCM_CFG_BASE_AS_FLOOR option is set to YES or TRUE, and these parameters are set to AUTOMATIC and have an initial or starting value, the database manager will not tune them below this value.

db2caem - Capture Activity Event Monitor Data Tool

FP3

- Run db2caem to create activity event monitor to capture data for an SQL statement. Info collected includes:
  - Detailed activity information captured by an activity event monitor including monitor metrics including total_cpu_time for statement execution.
  - Formatted EXPLAIN output, including section actuals (statistics for different operators in the access plan).
- Data can be collected with db2support.
  - DB2CAEM_<timestamp> directory created.

db2caem –d sample –st "select * from staff"

- Creates activity event monitor and captures information of details, section and values, as well as actuals for the SQL statement.

db2caem –d sample -actevm mymon –appid *LOCAL.mikita.100203234904 -uowid 44 -actid 1

- Captures activity event monitor information of details, section and values, as well as actuals for the SQL statement identified by the event monitor options from the existing activity event monitor. The db2caem tool will not create activity event monitor in this example.

Run db2exfmt to generate access plan and examine output

db2exfmt –d sample -1 –o exfmt_activity.out

New db2support event monitor options simplify capturing activity event monitor data with the db2caem tool (-aem , -actevm, -appid, -uowid, and -actid options). These options can be specified by themselves, or can be combined with one of the available SQL statement options to capture data for an SQL statement (–st, -sf or –se option).

The db2caem tool automates the procedure of creating an activity event monitor.

Run the db2caem command to create the activity event monitor to capture data for an SQL statement. This data can be collected with the db2support command. The information collected and generated by the db2caem tool includes:

•Detailed activity information captured by an activity event monitor including monitor metrics, for example total_cpu_time for statement execution

•Formatted EXPLAIN output, including section actuals (statistics for different operators in the access plan).

The db2caem tool uses an activity event monitor to capture information about the statements and then extracts and formats the information. The db2caem tool automates the process for creating an activity event monitor, enabling capture for the statements of interest, invoking the statements (each statement is rolled back after being executed to prevent side effects in the database), and formatting the output information (including exporting activity information for the statements of interest and generation of formatted explain output from the captured section and section actuals).

To create activity event monitor, the privileges must include one of the following: DBADM authority, SQLADM authority, WLMADM authority, and also EXECUTE privilege on the WLM_SET_CONN_ENV procedure.

If there is not a need to create activity event monitor, the following privileges and authority are required: EXECUTE privilege on the EXPLAIN_FROM_ACTIVITY procedure, INSERT privilege on the Explain tables in the specified schema, SELECT privilege on the event monitor tables for the source activity event monitor, and also one of the following: DATAACCESS authority on the activity event monitor tables, CONTROL or SELECT privilege on the activity event monitor tables.

The db2caem tool is used to create the activity event monitor for capturing data which can be collected with the db2support command. DB2CAEM_<timestamp> directory will be generated to contain all the information captured by the db2caem tool.

DB2 LUW 9.7 Fix Pack 3a

DB2 LUW 9.7 Fix Pack 3a needed for DB2 z/OS 10 new functions:

- ✓Dynamic statement cache enhancements.
- ✓Timestamp with timezone.
- ✓Greater timestamp precision.
- ✓Client/Server Binary XML Format
- ✓Extended indicator variables
- ✓Explain mode special register
- ✓Unicode collection and package names

http://www.ibm.com/support/docview.wss?uid=swg24028306

DB2 9 &10 Require DB2 Client & Connect 9 FP1 - Recommend Latest Level

• DB2 Connect V8 out of service April 30, 2009

The recommended level for client & DB2 Connect is as current as possible to improve availability, resolve problems and deliver the changes needed in Java, stored procedure and web services environments. DB2 10 new function comes in 9.7 fixpack 3a. Client and DB2 Connect V9.7 fixpack 3a are needed for a number of DB2 10 new functions: dynamic statement cache enhancements, timestamp with timezone, greater timestamp precision, client/server binary XML format, extended indicator variables, explain mode special register, and Unicode collection and package names. Get it from http://www.ibm.com/support/docview.wss?uid=swg24028306

If you are moving to DB2 9 or DB2 10 for z/OS, then the minimum supported level is DB2 Connect 9 FP1. If you are not at the minimum levels, then it's time to migrate to the most current level you can, DB2 9.7 fixpak 3a. If you get current now, then migration to DB2 9 for z/OS will not encounter the same problem. So if a client is down level, then get it migrated to current.

If you have a specific set of problems, then you can check the fixpak levels to see where those problems are resolved. But then you still need to test to make sure that the problem is resolved without introducing any new ones. Later fixpaks generally resolve a lot more problems. If you are migrating, find a good one and move to that level. Running with code that is out of support or missing a few thousand fixes is problematical. DB2 Connect V7 and V8 are not supported, and generally do not work. We don't generally separate CM from NFM on our discussions of fixpaks. Some new function comes in CM. See the web for fixpak information.

http://www.ibm.com/support/docview.wss?rs=56&uid=swg27007053
http://www.ibm.com/support/docview.wss?rs=71&uid=swg21255572
http://www.ibm.com/support/docview.wss?rs=71&uid=swg21256235

DB2 LUW 9.7 Fix Pack 3a

- DB2 Workgroup Server Edition
  - maximum allowed memory increased from 16 to 64 GB

- DB2 Advanced Enterprise Server Edition for Linux, UNIX, and Windows.
  - based on the DB2 ESE edition.
  - ideal foundation for building on-demand, enterprise-wide solutions.
  - can deploy on servers with any number of CPUs.
  - DB2 9.7 Enterprise Server Edition
  - DB2 9.7 Storage Optimization feature
  - DB2 Advanced Access Control feature
  - DB2 Workload Management feature
  - Optim Performance Manager
  - Data Studio 2.2.1
  - Optim Development Studio 2.2.1
  - Optim Database Administrator 2.2.3
  - InfoSphere Federation Server between DB2 and Oracle data sources

The recommended level for client & DB2 Connect is as current as possible to improve availability, resolve problems and deliver the changes needed in Java, stored procedure and web services environments. DB2 10 new function comes in 9.7 fixpack 3a. Client and DB2 Connect V9.7 fixpack 3a are needed for a number of DB2 10 new functions: dynamic statement cache enhancements, timestamp with timezone, greater timestamp precision, client/server binary XML format, extended indicator variables, explain mode special register, and Unicode collection and package names. Get it from http://www.ibm.com/support/docview.wss?uid=swg24028306
If you are moving to DB2 9 or DB2 10 for z/OS, then the minimum supported level is DB2 Connect 9 FP1. If you are not at the minimum levels, then it's time to migrate to the most current level you can, DB2 9.7 fixpak 3a. If you get current now, then migration to DB2 9 for z/OS will not encounter the same problem. So if a client is down level, then get it migrated to current.
If you have a specific set of problems, then you can check the fixpak levels to see where those problems are resolved. But then you still need to test to make sure that the problem is resolved without introducing any new ones. Later fixpaks generally resolve a lot more problems. If you are migrating, find a good one and move to that level. Running with code that is out of support or missing a few thousand fixes is problematical. DB2 Connect V7 and V8 are not supported, and generally do not work. We don't generally separate CM from NFM on our discussions of fixpaks. Some new function comes in CM. See the web for fixpak information.
http://www.ibm.com/support/docview.wss?rs=56&uid=swg27007053
http://www.ibm.com/support/docview.wss?rs=71&uid=swg21255572
http://www.ibm.com/support/docview.wss?rs=71&uid=swg21256235

DB2 9.7 Fix Pack 4 Checklist

New Enhancements:

- ListAgg aggregate function
- db2look enhancements
- WLM_COLLECT_STATISTICS procedure
- db2pd –recovery determines if catalog partition failed
- db2cklog tool checks validity of archive log files
- MemberConnectTimeout specifies number of seconds before attempt to open socket fails
- Create Trigger enhancements
- More resilient Diagnostic log
- FODC Member level Settings and Redirection
- New Support Scripts and db2diag enhancements
- ActivityMetrics Logical Data Group

64

Version 9.7 Fix Pack 4 contains important changes that might affect your product usage. You should review the technical changes and new functionality included in Version 9.7 Fix Pack 4. For a summary of DB2 Version 9.7 LUW fix pack 4 changes, please see:

http://www-01.ibm.com/support/docview.wss?uid=swg24027906

LISTAGG and db2look

- **LISTAGG** function aggregates set of string elements into one string by concatenating the strings.
  - To produce an alphabetical list of comma-separated names, grouped by department.

  ***SELECT** workdept, **LISTAGG**(lastname, ', ') **WITHIN GROUP** **(ORDER BY** lastname**) AS** employees **FROM** emp **GROUP BY** workdept*
- **db2look**
  - specify two-part names for tables (schema.table) and views (schema.view). extended to selecting tables for DDL statement generation using pattern matching, which you can do by using the -tw parameter.
  - Parameters **-xdep** and **-xddep** generate DDL statements for parent and dependent objects in different schemas and generate authorization DDL statements for dependent objects.
    - on tables, specified by either -t or -tw parameter, and their dependent objects.

A new aggregate function, LISTAGG, has been added that aggregates a set of string elements into one string by concatenating the strings. Optionally, a separator string can be provided which is inserted between contiguous input strings.

To produce an alphabetical list of comma-separated names, grouped by department:

SELECT workdept, **LISTAGG**(lastname, ', ') **WITHIN GROUP (ORDER BY** lastname**) AS** employees **FROM** emp **GROUP BY** workdept

You can use pattern expressions in a LIKE predicate which are based on other columns. The LIKE predicate can therefore be used as a fuzzy join predicate where the joined table provides the pattern. You can, for example, look up patterns such as area codes to filter telephone numbers from another table.

Starting with DB2 9.7 Fix Pack 4, you can use a two-part name for a table, in the format of *schema*.*table*, or a view, in the format of *schema*.*view*, when the table or view has dependent objects that are in a different schema and you require DDL statements to be also generated for these dependent objects. The ability to specify a two-part name is also extended to selecting tables for DDL statement generation using pattern matching, which you can do by using the -tw parameter.

The **-xdep** and **-xddep** parameters generate authorization DDL statements (for example, GRANT statements) for dependent and parent objects.

WLM_COLLECT_STATS procedure

FP4

- Takes input parameter called **wait**, which specifies the procedure does not return until all statistics have been written and flushed to statistics event monitor tables.
- If not specified, procedure returns immediately after initiating a statistics collection and reset.

CALL WLM_COLLECT_STATS
('Y', :collect_timestamp) *WAIT*

The WLM_COLLECT_STATS procedure now takes an input parameter called wait, which specifies that the procedure does not return until all the statistics have been written and flushed to the statistics event monitor tables. If you do not specify this parameter, the procedure returns immediately after initiating a statistics collection and reset. Monitoring tools can use the new functionality to collect WLM statistics in synchronous mode so that the tools are aware that all the data has been written to the statistics event monitor tables by the time the procedure returns. The WLM_COLLECT_STATS procedure gathers statistics for service classes, workloads, work classes, and threshold queues and writes them to the statistics event monitor. The procedure also resets the statistics for service classes, workloads, work classes, and threshold queues. If there is no active statistics event monitor, the procedure only resets the statistics. Call WLM_COLLECT_STATS to collect and reset statistics, but not return until data has been written to statistics event monitor tables.

CALL WLM_COLLECT_STATS('Y', ::collect_timestamp) ***wait***

An optional input argument of type CHAR that specifies whether this procedure returns immediately after initiating a statistics collection and reset. If 'Y' is specified, then the procedure will not return until all statistics have been written and flushed to the statistics event monitor tables. Otherwise, the procedure will return immediately after initiating a statistics collection and reset.

statistics_timestamp

An optional output argument of type TIMESTAMP that returns the timestamp value for the beginning of the statistics collection.

db2pd -recovery

- Determine whether catalog database partition has failed.
- **Database State** shows state of catalog partition. If catalog partition fails, **CATALOGNODEFAIL** state is returned. Otherwise, no information is returned.
- State can be displayed from any database partition.

You can use the -recovery option of the db2pd command to determine whether the catalog database partition has failed.

For the -recovery parameter, the following information is returned:
Database State

In Version 9.7 Fix Pack 4 and later fix packs, the state of the catalog partition in partitioned database environments if the database catalog partition fails. If the database catalog partition fails, the CATALOGNODEFAIL state is returned. Otherwise, no information is returned.

This state can be displayed from any database partition.

db2cklog tool

- Check validity of archive log files before using files during rollforward recovery operation.
- Ensures recovery operation does not fail because of problem with log file.
- Reads either single log file or range of log files and performs internal validity checks on these files.
- If a log file fails validation with error message or if warning is returned, indicates that must not use that log file during rollforward recovery.
- IBM Software Support may ask to run if suspect an invalid log file is behind a problem with your data server.

In Version 9.7 Fix Pack 4 and later fix packs, you can check the validity of archive log files with the db2cklog tool before using these files during a rollforward recovery operation. Checking your archive log files immediately before a rollforward recovery ensures that the recovery operation does not fail because of a problem with a log file. You can also use the tool preventively, after each log file is closed and copied to the log archive directory. The db2cklog tool works by reading either a single log file or by reading a range of log files and performing internal validity checks on these files. Log files that pass validation by the db2cklog tool without any error messages or warnings can be used during a rollforward recovery operation. If a log file fails validation with an error message or if a warning is returned, then this indicates that you must not use that log file during rollforward recovery. You can follow the recommended responses for dealing with archive log files that fail validation or return a warning.

You might be asked to run the db2cklog tool, if IBM Software Support suspects that an invalid log file is behind a problem with your data server.

MemberConnectTimeout (configuration keyword)

- Finer grained and more precise timeout value can be used to be set for application reroute scenarios
- Only applicable to IBM Data Server Driver – not CLI or .NET
- Specifies number of seconds before an attempt to open a socket fails when dealing with socket opens to members in Data Sharing member list in Sysplex setup or member list of DB2 pureScale instance.
- If set and Sysplex or DB2 pureScale is active, value is used on every open socket to a member in member list. After all attempts to open a socket fail to each member, then when retrying on a group IP address, tcpipConnectTimeout is used before failing the connection request.
- **db2dsdriver.cfg configuration syntax**
  - <parameter name="MemberConnectTimeout" value=" 1"/>
  - If = 0 or a negative value, tcpipConnectTimeout keyword value is used.
  - For DB2 z/OS servers, default value = 1 second.

Starting with DB2 Version 9.7 Fix Pack 4, the MemberConnectTimeout configuration keyword enables a finer grained, more precise timeout value to be set for reroute scenarios. By using the MemberConnectTimeout configuration keyword, the socket open will normally be faster than opening the socket with use of ConnectionTimeout keyword, or with no keyword at all. The MemberConnectTimeout is configuration keyword is only applicable to the IBM Data Server Driver. Specifies the number of seconds before an attempt to open a socket fails. This smaller timeout value is used when dealing with socket opens to members in the data sharing member list in sysplex setup or member list of DB2 pureScale instance. In these scenarios, the socket open would be faster than in the general case.

Equivalent CLI keyword N/A; **Equivalent IBM Data Server Provider for .NET connection string keyword** N/A

db2dsdriver.cfg configuration syntax <parameter name="MemberConnectTimeout" value=" 1"/>

Default setting: None. If the MemberConnectTimeout keyword value is set to 0 or a negative value, the tcpipConnectTimeout keyword value is used. For DB2 for z/OS servers, the default value is 1 second.

If MemberConnectTimeout is set and Sysplex or DB2 pureScale exploitation is active, this value is used on every open socket to a member in the member list. After all attempts to open a socket fail to each member, then when retrying on a group IP address, tcpipConnectTimeout is used before failing the connection request.

The CREATE TRIGGER statement has changed. A trigger event clause can now contain UPDATE, DELETE, and INSERT operations together in a single clause. Additionally, a BEFORE trigger can contain UPDATE, DELETE, INSERT, and modifying data routines in a compound SQL (compiled) statement.

When creating a trigger with the CREATE TRIGGER statement, you can Include more than one operation in the trigger event clause. You now have the ability to use UPDATE, DELETE, and INSERT operations together in a single clause. This capability means that the trigger is activated by the occurrence of any of the specified events. One, two, or all three trigger events can be arbitrarily specified in a CREATE TRIGGER statement. However, an operation cannot be specified more than once.

Identify the event that activated a trigger. The trigger event predicates of UPDATING, INSERTING, and DELETING can be used as Boolean conditions for identifying trigger actions. Trigger event predicates can only be used in the trigger action of a CREATE TRIGGER statement that uses a compound SQL (compiled) statement.

Use statement level triggers in PL/SQL. You can create triggers that fire only one time per statement irrespective of the number of rows affected.

Diagnostic Data Log More Resilient

- Set alternate path for diagnostic data with **ALT_DIAGPATH** dbm cfg parameter.
- Used when primary diagnostic path (DIAGPATH) is unavailable.
- No default value. If DB2 fails to write to diagpath, important diagnostic information is lost. If set alt_diagpath to same path as diagpath receive error message.
- Following values are available:
  - '"*pathname* $h"'
  - '"*pathname* $h/*trailing-dir*"'
  - '"*pathname* $n"'
  - '"*pathname* $n/*trailing-dir*"'
  - '"*pathname* hn"'
  - '"*pathname* hn/*trailing-dir*"'

The alternate diagnostic data directory can contain the same diagnostic data as the primary diagnostic data directory set with the diagpath parameter. When alt_diagpath is set and the primary diagnostic data directory becomes unavailable, diagnostic logging continues in the alternate diagnostic data directory path specified, then resumes in its original location when the primary diagnostic path becomes available again. If this parameter is null and the primary diagnostic data directory specified by the diagpath parameter is unavailable, no further diagnostic information is written until the primary diagnostic path becomes available again. For improved resilience, set the alternate diagnostic data directory to point to a different file system than the primary diagnostic data directory.

The serviceability of large database systems has been improved. A number of functional enhancements have been made that address common pain points on large database systems, resulting in: Reduced amounts of accumulated diagnostic data; reduced overhead due to data collection on large systems; improvements to the accessibility of diagnostic data to service personnel; and, improvements to the ease of use of troubleshooting tools in complex systems.

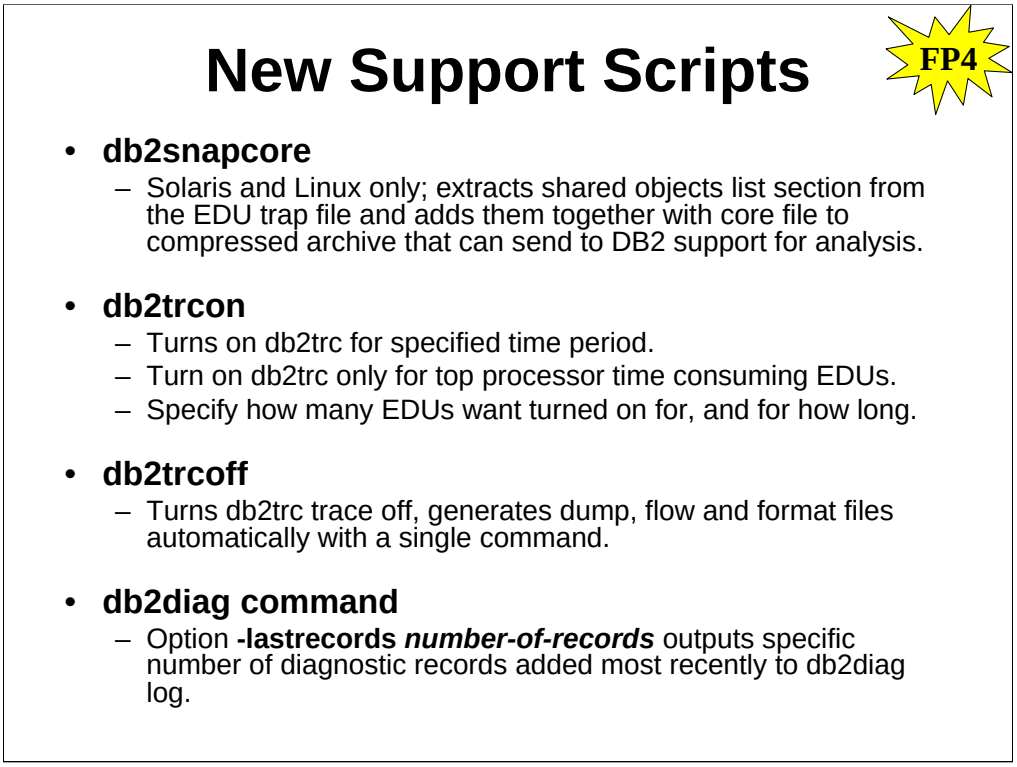
FODC Member-level Settings and Redirection

FP4

- Implementation of FODC allows each member in db system to have own FODC settings.
- Gives greater control than instance-level settings
- Easier to locate diagnostic information for specific member or run multiple automatic or manual FODC processes in parallel.
- Can now collect diagnostic data only from a specific member that encounters a problem and not have diagnostic data from other members on the same host included.
- To avoid scenario where FODC fills all available space in file system and impacts your data server, can chose where FODC data is stored with FODCPATH registry variable.

The implementation of FODC has changed, so that each member in the database system can now have its own FODC settings. Member-level FODC settings give you greater control than the instance-level or host-level settings supported in previous releases and fix packs. As a result, it is now easier to locate the diagnostic information for a specific member in the database environment, or to run multiple, automatic or manual FODC processes in parallel. For example, you can now collect diagnostic data only from a specific member that encounters a problem and not have diagnostic data from other members on the same host included.

When errors occur, the automatic capture of important diagnostic data can generate a significant volume of diagnostic data that requires space on the file system to store. To avoid a scenario where FODC fills all the available space in the file system and impacts your data server, you can now chose where the FODC data is stored with the FODCPATH registry variable.



New support scripts: db2snapcore, db2trcon, db2trcoff

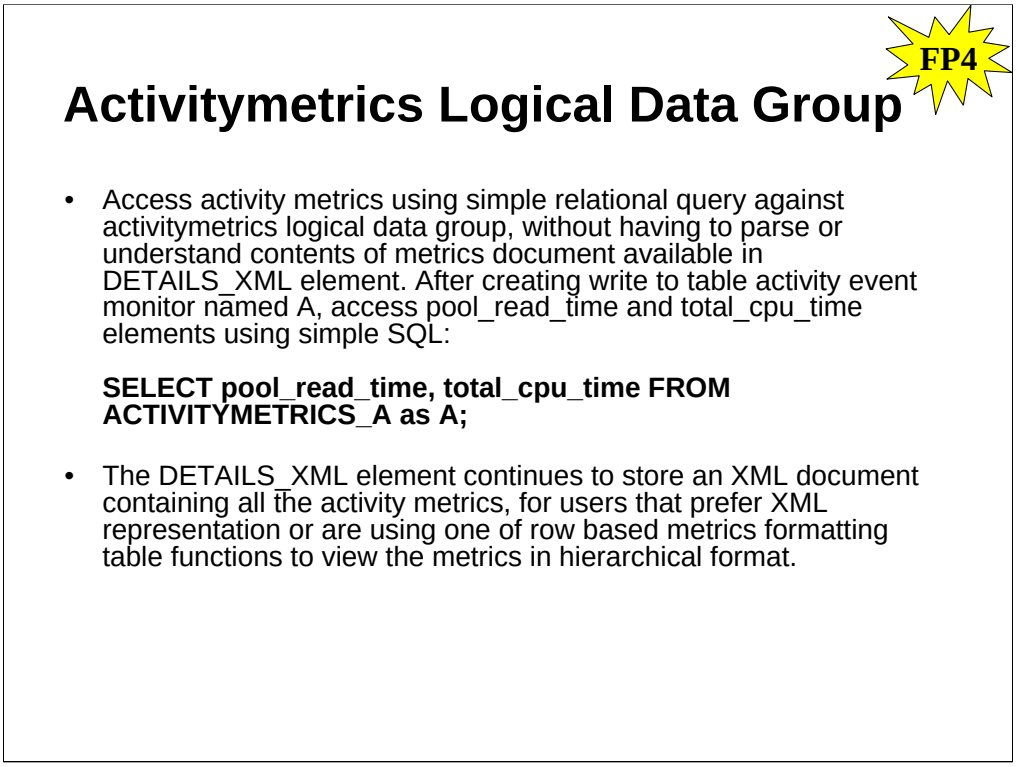
The following new scripts are tools IBM service analysts can use during problem diagnosis and are now shipped with the product:

db2snapcore - on Solaris and Linux operating systems only, this command extracts the shared objects list section from the EDU trap file and adds them together with the core file to a compressed archive that you can send to DB2 support for analysis. The functionality provided by db2snapcore is similar to the snapcore command on the AIX operating system.

db2trcon - turns on the db2trc trace for a time period you specify. You can use this script to turn on db2trc only for top processor time consuming engine dispatchable units (EDUs). You can specify how many EDUs you want db2trc to be turned on for, and for how long.

db2trcoff - turns the db2trc trace off, and generates dump, flow and format files automatically with a single command.

The db2diag command supports a new **-lastrecords *number-of-records*** parameter option. You use this option to output a specific number of diagnostic records added most recently to the db2diag log file.



You can access activity metrics using a simple relational query against the activitymetrics logical data group, without having to parse or understand the contents of the metrics document available in the DETAILS_XML element. For example, after creating a write to table activity event monitor named A, you can access the pool_read_time and total_cpu_time elements using a simple SQL statement such as the following:

SELECT pool_read_time, total_cpu_time FROM ACTIVITYMETRICS_A as A;

The DETAILS_XML element continues to store an XML document containing all the activity metrics, for those users that prefer the XML representation or are using one of the row based metrics formatting table functions to view the metrics in a hierarchical format.

Bio

Melanie Stopfer is a Consulting Learning Specialist and Developer for IBM Software Group. As a Certified DB2 9.7 Database Administrator, DB2 Certified Advanced Technical Expert and Certified Learning Facilitation Specialist, she has provided hands-on in-depth technical support to customers specializing in both data warehouse and transaction systems. She has worked with development labs and worldwide customers to provide DB2 solutions since 1988. In 2009, Melanie was the first DB2 LUW speaker to be inducted into the IDUG Speaker Hall of Fame and was also selected as Best Overall Speaker at IDUG NA in 2009 and 2008, Top Ten Speaker at IDUG NA 2010, 2009, 2008, 2007, and IDUG Europe 2009, 2008 and 2007. She has received numerous awards for development of DB2 recovery, administration, performance, and database upgrade and migration best practice solutions.