

*Preparation Guide for DB2 10.5 DBA
for LUW Upgrade from DB2 10.1 Exam
311*



*Preparation Guide for DB2 10.5 DBA
for LUW Upgrade from DB2 10.1 Exam
311*



Note

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 239.

Edition Notice

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at <http://www.ibm.com/shop/publications/order>
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at <http://www.ibm.com/planetwide/>

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corporation 2014.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book	v
Who should use this book	v

Part 1. DB2 Server management 1

Chapter 1. Creating and setting up your database configuration for analytic workloads 3

Enabling parallel processing for column-organized tables	4
Enabling intrapartition parallelism for queries	5

Chapter 2. Setting the default table organization 9

Chapter 3. Space reclamation for column-organized tables 11

Enabling automatic table and index reorganization	11
---	----

Chapter 4. Default query concurrency management 13

Default workload management objects for concurrency control	16
---	----

Chapter 5. IBM Data Studio 23

Using IBM Data Studio for key tasks	23
IBM Data Studio client	25
IBM Data Studio web console	28
Database administration with IBM Data Studio	29
Administering databases with task assistants	30
Database administration commands that you can run from task assistants	35
Managing jobs in IBM Data Studio	40
Creating and managing jobs	42
Scenario: Creating and scheduling a job	42
Importing tasks from DB2 Task Center	44
Diagramming access plans with Visual Explain	45
Diagrams of access plans	48
Query blocks	48
Setting preferences for Visual Explain	49

Part 2. Physical design. 51

Chapter 6. What is new in DB2 Version 10.5 53

Chapter 7. Column-organized tables . . . 57

Synopsis tables	58
Supported system and database configurations for column-organized tables	58
Creating column-organized tables	59

INSERT, UPDATE, and DELETE (including MERGE) statement restrictions for column-organized tables	61
Loading data into column-organized tables	62
Scenario: Achieving high speed analytics over growing volumes of column-organized data	63

Chapter 8. Expression-based indexes 65

Statistics for expression-based indexes	65
Expression-based indexes and automatic statistics collection	65
Expression-based indexes and manual statistics updates	66
RUNSTATS on expression-based indexes	66
Expression-based indexes and statistics profiles	67

Chapter 9. Extended row size 71

Implementing extended row size	71
--	----

Chapter 10. Informational constraints 75

Designing informational constraints	75
Creating and modifying constraints	77

Chapter 11. Data compression 81

Table compression	81
Value compression	82
Row compression	83
Classic row compression	83
Adaptive compression	84
Estimating storage savings offered by adaptive or classic row compression	86
Creating a table that uses compression	87
Enabling compression in an existing table	89
Changing or disabling compression for a compressed table	90
Compression dictionaries	91
Table-level compression dictionary creation	92
Column compression dictionaries after a load or insert operation	95
Index compression	95
Backup compression	98

Chapter 12. DB2 compatibility features 99

DB2_COMPATIBILITY_VECTOR registry variable	100
Compatibility database configuration parameters	104
Terminology mapping: Oracle to DB2 products	104
Oracle data dictionary-compatible views	109
Oracle database link syntax	110
Setting up the DB2 environment for Oracle application enablement	111

Part 3. Monitoring DB2 Activity 115

Chapter 13. Monitoring metrics for column-organized tables 117

Chapter 14. Explain information for column-organized tables 125
TQ operator 128

Chapter 15. Monitoring routines using table functions 135
MON_GET_ROUTINE table function - get aggregated routine execution metrics 135

Chapter 16. Monitoring information for HADR. 137
MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information 137
db2pd - Monitor and troubleshoot DB2 database 140

Part 4. High availability 149

Chapter 17. Inplace (online) table reorganization 151
Reorganizing tables online 152
Locking and concurrency considerations for online table reorganization 152

Chapter 18. High availability disaster recovery (HADR) in DB2 pureScale environments 155
Restrictions for HADR in DB2 pureScale environments 155
HADR setup in a DB2 pureScale environment 156
High availability disaster recovery (HADR) monitoring in a DB2 pureScale environment 157
HADR standby replay in a DB2 pureScale environment. 160
 Changing the preferred replay member. 161
DB2 pureScale topology changes and high availability disaster recovery (HADR) 161
 Adding members to a high availability disaster recovery (HADR) setup 162
 Removing members from a high availability disaster recovery (HADR) setup 163
HADR takeover operations in a DB2 pureScale environment. 164
Scenario: Deploying HADR in a DB2 pureScale environment. 165

Chapter 19. Online fix pack updates in DB2 pureScale environments 173
Database and instance operations affected by an online fix pack update in progress 176
Applying fix packs in DB2 pureScale environments 176
 Preparing to install a fix pack 177
 Installing online fix pack updates to a higher code level on a DB2 pureScale instance. 182

Installing online fix pack updates to a higher code level in a HADR environment 188
Installing online fix pack updates to a higher code level in a GDPC environment 191
Installing offline fix pack updates to a DB2 pureScale instance (simplified method) 194
Post-installation tasks for fix packs (Linux and UNIX). 195

Chapter 20. Self-tuning memory for DB2 pureScale environments 201

Chapter 21. Explicit hierarchical locking for DB2 pureScale environments 203
Use cases for Explicit Hierarchical Locking (EHL) 204
Explicit hierarchical locking state changes and performance implications 205

Chapter 22. DB2 Advanced Copy Services (ACS) scripted interface. 207
DB2 Advanced Copy Services (ACS) protocol file 207
DB2 Advanced Copy Services (ACS) user scripts 218
Performing a snapshot backup with a script 219
Restoring from a snapshot backup image with a script 221

Part 5. Utilities 223

Chapter 23. IBM InfoSphere Optim Query Workload Tuner for DB2 for Linux, UNIX, and Windows. 225
Workload Table Organization Advisor 225
Workflow assistant 225
Access Plan Explorer 226

Chapter 24. Converting row-organized tables to column-organized tables 227

Chapter 25. Loading data into column-organized tables 229

Part 6. Appendixes 231

Appendix A. DB2 technical information 233
DB2 technical library in PDF format. 234
Displaying SQL state help from the command line processor. 235
Accessing DB2 documentation online for different DB2 versions 235
Terms and conditions. 236

Appendix B. Notices 239

Index 243

About this book

This book provides information from the DB2® for Linux, UNIX, and Windows documentation to cover all the objectives that are described in the IBM® DB2 10.5 DBA for LUW Upgrade from DB2 10.1 Exam 311.

- Part 1, “DB2 Server management,” on page 1 provides information about how to configure DB2 workloads for analytics systems, use autonomic features for BLU Acceleration, and use IBM Data Studio 4.1.
- Part 2, “Physical design,” on page 51 provides information about how to implement compression features, use expression-based indexes, information constraints, sql compatibility enhancements, and capabilities for Oracle migration and compatibility.
- Part 3, “Monitoring DB2 Activity,” on page 115 provides information about new monitoring elements and explain information that supports column-organized tables and HADR.
- Part 4, “High availability,” on page 149 provides information about enhancements for DB2 pureScale® environments and DB2 Advanced Copy Services customized scripts.
- Part 5, “Utilities,” on page 223 provides information about how to use new and modified utilities on column-organized tables and how to use capabilities in IBM Optim™ Query Workload Tuner that are related to column-organized tables.

Passing the IBM DB2 10.5 DBA for LUW Upgrade from DB2 10.1 Exam 311 is one of the requirements to obtain the *IBM Certified Database Administrator - DB2 10.5 DBA for LUW Upgrade from DB2 10.1* certification. For complete details about this certification and its requirements, see <http://www.ibm.com/certify/certs/08002108.shtml>.

Who should use this book

This book is for database administrators and other DB2 database users with intermediate to advanced administration skills who want to prepare for the certification Exam 311. For complete details about the exam, see <http://www.ibm.com/certify/tests/ovr311.shtml>.

Part 1. DB2 Server management

DB2 Server management provides information about how to configure DB2 workloads for analytics systems.

A data server refers to a computer where the DB2 database engine is installed. The DB2 engine is a full-function, robust database management system that includes optimized SQL support based on actual database usage and tools to help manage the data.

IBM offers a number data server products, including data server clients that can access all the various data servers. For a complete list of DB2 data server products, features available, and detailed descriptions and specifications, visit the product page at the following URL: <http://www.ibm.com/software/data/db2/linux-unix-windows/>.

Chapter 1. Creating and setting up your database configuration for analytic workloads

If **DB2_WORKLOAD** is not set to `=ANALYTICS`, you should create and optimally configure your database for analytic workloads.

Before you begin

- You must have an instance that was created on DB2 Version 10.5.
- Ensure that your system configuration is supported. For more information, see “Supported system and database configurations for column-organized tables” on page 58.

About this task

The way that you create a database and its configuration influence the performance of analytic workloads and column-organized

Restrictions

- Check the restrictions that are listed in “Supported system and database configurations for column-organized tables” on page 58.

Procedure

To manually create and configure your database for analytic workloads:

1. Create the database with a 32K page size, a UNICODE code set (default), and an **IDENTITY** or **IDENTITY_16BIT** collation. The following example shows how to create a database that is called **DMART** with such characteristic:

```
CREATE DATABASE DMART COLLATE USING IDENTITY PAGESIZE 32 K
```
2. Ensure that the **sheapthres** database manager configuration parameter is set to 0 (default value). Note that this setting applies to all databases in the instance.
3. Update the database configuration as follows:
 - a. Set the **dft_table_org** (default table organization for user tables) database configuration parameter to **COLUMN** so that new tables are created as column-organized tables by default; otherwise, the **ORGANIZE BY COLUMN** clause must be specified on each **CREATE TABLE** statement.
 - b. Set the **dft_degree** (default degree) database configuration parameter to **ANY**.
 - c. Set the **dft_extent_sz** (default extent size) database configuration parameter to 4.
 - d. Increase the value of the **catalogcache_sz** (catalog cache) database configuration parameter by 20% (it is set automatically during database creation).
 - e. Ensure that the **sortheap** (sort heap) and **sheapthres_shr** (sort heap threshold for shared sorts) database configuration parameters are not set to **AUTOMATIC**. Consider increasing these values significantly for analytics workloads. A reasonable starting point is setting **sheapthres_shr** to the size of the buffer pool (across all buffer pools). Set **sortheap** to some fraction (for example, 1/20) of **sheapthres_shr** to enable concurrent sort operations.
 - f. Set the **util_heap_sz** (utility heap size) database configuration parameter to 1,000,000 pages and **AUTOMATIC** to address the resource needs of the **LOAD** command. If the database server has at least 128 GB of memory, set

util_heap_sz to 4,000,000 pages. If concurrent load operations are running, increase the value of **util_heap_sz** to accommodate higher memory requirements.

- g. Set the **auto_reorg** (automatic reorganization) database configuration parameter to ON.

Note: This database configuration increases the overall database memory requirements. If the **database_memory** configuration parameter is set to AUTOMATIC, consider increasing its value.

4. Ensure that intraquery parallelism is enabled to be able to access column-organized tables. You can enable intraquery parallelism at the instance level, database level, workload level, or application level. For details, see “Enabling parallel processing for column-organized tables.”
5. Enable concurrency control on the SYSDEFAULTMANAGEDSUBCLASS service subclass by issuing the following statement:

```
ALTER THRESHOLD SYSDEFAULTCONCURRENT ENABLE
```

Enabling parallel processing for column-organized tables

Certain operations on column-organized tables such as access and query processing, require enabling of intraquery parallelism and intrapartition parallelism.

Before you begin

- You must have the proper authorization to configure instances and databases.

About this task

The processing of queries against column-organized tables require that intraquery parallelism be enabled for the application that is compiling and executing the query. The following statement types also require intraquery parallelism:

- All DML operations that reference column-organized tables
- The ALTER TABLE ADD UNIQUE / PRIMARY KEY CONSTRAINT statement against a column-organized table if rows were inserted into the table
- The **RUNSTATS** command against a column-organized table
- The **LOAD** command against a column-organized table

If an application attempts to run one of these statements or commands without intraquery parallelism enabled, an error is returned. Intraquery parallelism uses either intrapartition parallelism, interpartition parallelism, or both. Interpartition parallelism is not available because column-organized tables are supported in only single-partition environments.

Access to column-organized tables also require intrapartition parallelism. Setting **DB2_WORKLOAD** to ANALYTICS implicitly enables intrapartition parallelism for workload objects that are created with MAXIMUM DEGREE set to DEFAULT, and is recommended when you use column-organized tables. If this registry variable is not set, or is set to another value, you must explicitly enable intrapartition parallelism before you access column-organized.

Procedure

To enable parallel processing for column-organized tables:

1. Enable intrapartition parallelism, which in turn enables intraquery parallelism. For more information, see “Enabling intrapartition parallelism for queries”.
2. Ensure that the shared sort heap is allocated as parallel processing of queries requires use of the shared sort heap. Any of the following actions ensure that the shared sort heap is allocated:
 - Set the **intra_parallel** database manager configuration parameter to YES.
 - Set the **sheapthres** database manager configuration parameter to 0.
 - Activate the connection concentrator. For more information, see “Connection concentrator”.

Enabling intrapartition parallelism for queries

To enable intrapartition query parallelism, modify one or more database or database manager configuration parameters, precompile or bind options, or a special register. Alternatively, use the MAXIMUM DEGREE option on the CREATE or ALTER WORKLOAD statement, or the ADMIN_SET_INTRA_PARALLEL procedure to enable or disable intrapartition parallelism at the transaction level.

Before you begin

Use the following controls to specify what degree of intrapartition parallelism the optimizer is to use:

- CURRENT DEGREE special register (for dynamic SQL)
- DEGREE bind option (for static SQL)
- **dft_degree** database configuration parameter (provides the default value for the previous two parameters)

Use the following controls to limit the degree of intrapartition parallelism at run time. The runtime settings override the optimizer settings.

- **max_querydegree** database manager configuration parameter
- SET RUNTIME DEGREE command
- MAXIMUM DEGREE workload option

Use the following controls to enable or disable intrapartition parallelism:

- **intra_parallel** database manager configuration parameter
- ADMIN_SET_INTRA_PARALLEL stored procedure
- MAXIMUM DEGREE workload option (set to 1)

About this task

Use the **GET DATABASE CONFIGURATION** or the **GET DATABASE MANAGER CONFIGURATION** command to find the values of individual entries in a specific database or instance configuration file. To modify one or more of these entries, use the **UPDATE DATABASE CONFIGURATION** or the **UPDATE DATABASE MANAGER CONFIGURATION** command.

intra_parallel

Database manager configuration parameter that specifies whether or not the database manager can use intrapartition parallelism. The default is NO, which means that applications in this instance are run without intrapartition parallelism. For example:

```
update dbm cfg using intra_parallel yes;
get dbm cfg;
```

max_querydegree

Database manager configuration parameter that specifies the maximum degree of intrapartition parallelism that is used for any SQL statement running on this instance. An SQL statement does not use more than this value when running parallel operations within a database partition. The default is -1, which means that the system uses the degree of intrapartition parallelism that is determined by the optimizer, not the user-specified value. For example:

```
update dbm cfg using max_querydegree any;
get dbm cfg;
```

The **intra_parallel** database manager configuration parameter must also be set to YES for the value of **max_querydegree** to be used.

dft_degree

Database configuration parameter that specifies the default value for the **DEGREE** precompile or bind option and the CURRENT DEGREE special register. The default is 1. A value of -1 (or ANY) means that the system uses the degree of intrapartition parallelism that is determined by the optimizer. For example:

```
connect to sample;
update db cfg using dft_degree -1;
get db cfg;
connect reset;
```

DEGREE Precompile or bind option that specifies the degree of intrapartition parallelism for the execution of static SQL statements on a symmetric multiprocessing (SMP) system. For example:

```
connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...
```

CURRENT DEGREE

Special register that specifies the degree of intrapartition parallelism for the execution of dynamic SQL statements. Use the SET CURRENT DEGREE statement to assign a value to the CURRENT DEGREE special register. For example:

```
connect to sample;
set current degree = '1';
connect reset;
```

The **intra_parallel** database manager configuration parameter must also be set to YES to use intrapartition parallelism. If it is set to NO, the value of this special register is ignored, and the statement will not use intrapartition parallelism. The value of the **intra_parallel** database manager configuration parameter and the CURRENT DEGREE special register can be overridden in a workload by setting the MAXIMUM DEGREE workload attribute.

MAXIMUM DEGREE

CREATE WORKLOAD statement (or ALTER WORKLOAD statement) option that specifies the maximum runtime degree of parallelism for a workload.

For example, suppose that bank_trans is a packaged application that mainly executes short OLTP transactions, and bank_report is another packaged application that runs complex queries to generate a business intelligence (BI) report. Neither application can be modified, and both are

bound with degree 4 to the database. While bank_trans is running, it is assigned to workload trans, which disables intrapartition parallelism. This OLTP application will run without any performance degradation associated with intrapartition parallelism overhead. While bank_report is running, it is assigned to workload bi, which enables intrapartition parallelism and specifies a maximum runtime degree of 8. Because the compilation degree for the package is 4, the static SQL statements in this application run with only a degree of 4. If this BI application contains dynamic SQL statements, and the CURRENT DEGREE special register is set to 16, these statements run with a degree of 8.

```
connect to sample;

create workload trans
  applname('bank_trans')
  maximum degree 1
  enable;

create workload bi
  applname('bank_report')
  maximum degree 8
  enable;

connect reset;
```

ADMIN_SET_INTRA_PARALLEL

Procedure that enables or disables intrapartition parallelism for a database application. Although the procedure is called in the current transaction, it takes effect starting with the next transaction. For example, assume that the following code is part of the demoapp application, which uses the ADMIN_SET_INTRA_PARALLEL procedure with both static and dynamic SQL statements:

```
EXEC SQL CONNECT TO prod;

// Disable intrapartition parallelism:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('NO');
// Commit so that the effect of this call
// starts in the next statement:
EXEC SQL COMMIT;

// All statements in the next two transactions run
// without intrapartition parallelism:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR rstmt;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :deptname;
EXEC SQL CLOSE c1;
...
// New section for this static statement:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;

// Enable intrapartition parallelism:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('YES');
// Commit so that the effect of this call
// starts in the next statement:
EXEC SQL COMMIT;

strcpy(stmt, "SET CURRENT DEGREE='4'");
// Set the degree of parallelism to 4:
EXEC SQL EXECUTE IMMEDIATE :stmt;
```

```

// All dynamic statements in the next two transactions
// run with intrapartition parallelism and degree 4:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
EXEC SQL DECLARE c2 CURSOR FOR rstmt;
EXEC SQL OPEN c2;
EXEC SQL FETCH c2 INTO :deptname;
EXEC SQL CLOSE c2;
...
// All static statements in the next two transactions
// run with intrapartition parallelism and degree 2:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;

```

The degree of intrapartition parallelism for dynamic SQL statements is specified through the CURRENT DEGREE special register, and for static SQL statements, it is specified through the DEGREE bind option. The following commands are used to prepare and bind the demoapp application:

```

connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...

```

Chapter 2. Setting the default table organization

You can set the default table organization for user-defined tables to row or column. The default table organization is used when you create tables without specifying the `ORGANIZE BY COLUMN` or the `ORGANIZE BY ROW` clause for the `CREATE TABLE` statement. The default table organization setting is row.

Before you begin

Before you set the default table organization to `COLUMN`, ensure that you meet the prerequisites for creating column-organized tables. For details, see “Creating column-organized tables” on page 59.

Procedure

To set the default table organization, set the **dft_table_org** database configuration parameter in one of the following ways:

- Issue the **UPDATE DATABASE CONFIGURATION** command:
 - To create column-organized tables by default, specify the `COLUMN` value.
 - To create row-organized tables by default, specify the `ROW` value.
- Set the **DB2_WORKLOAD** registry variable to `ANALYTICS`. This setting establishes a configuration that is optimal for analytic workloads, which includes setting the **dft_table_org** database configuration parameter to `COLUMN` to create column-organized tables by default.

Example

The following sample command shows you how to set `COLUMN` as the default table organization for the `SAMPLE` database:

```
update db cfg for sample using dft_table_org column
```

Chapter 3. Space reclamation for column-organized tables

When data is deleted from a column-organized table, the storage extents whose pages held data that was all deleted are candidates for space reclamation. The space reclamation process finds these extents, and returns the pages that they hold to table space storage, where they can later be reused by any table in the table space.

You can start this process manually by specifying the **RECLAIM EXTENTS** option for the **REORG TABLE** command, or you can use an automated approach. If you set the **DB2_WORKLOAD** registry variable to **ANALYTICS**, a default policy is applied, and the **auto_reorg** database configuration parameter is set so that automatic reclamation is active for all column-organized tables. For more information, see “Enabling automatic table and index reorganization.”

You can monitor the progress of a table reorganization operation with the reorganization monitoring infrastructure. The **ADMIN_GET_TAB_INFO** table function returns an estimate of the amount of reclaimable space on the table, which you can use to determine when a table reorganization operation is necessary.

Note: Because **RECLAIMABLE_SPACE** output from the **ADMIN_GET_TAB_INFO** table function is an estimate that applies to the time at which the **RUNSTATS** command was run, this information might be outdated for column-organized tables.

Space reclamation after deletions

When a row in a column-organized table is deleted, the row is logically deleted, not physically deleted. As a result, the space that is used by the deleted row is not available to subsequent transactions and remains unusable until space reclamation occurs. For example, consider the case where a table is created and one million rows are inserted in batch operation A. The size of the table on disk after batch operation A is 5 MB. After some time, batch operation B inserts another 1 million rows. Now the table uses 10 MB on disk. Then, all of the rows that were inserted in batch operation A are deleted, and the table size on disk remains 10 MB. If a third batch operation C inserts another 1 million rows into the table, 5 MB of extra space is required. (With row-organized tables, the rows that are inserted in batch operation C would use the space that was vacated by the deleted rows from batch operation A.) A **REORG TABLE** command is required to reclaim the space that was used by the rows that were inserted in batch operation A.

Space reclamation after updates

When a row in a column-organized table is updated, the row is first deleted, and a new copy of the row is inserted at the end of the table. This means that an updated row uses space in proportion to the number of times that the row is updated until space reclamation occurs. All rows in the extent where the update took place must be deleted before any space reclamation can occur.

Enabling automatic table and index reorganization

Use automatic table and index reorganization to eliminate the worry of when and how to reorganize your data.

About this task

Having well-organized table and index data is critical to efficient data access and optimal workload performance. After many database operations, such as insert, update, and delete, logically sequential table data might be found on nonsequential data pages. When logically sequential table data is found on nonsequential data pages, additional read operations are required by the database manager to access data. Additional read operations are also required when accessing data in a table from which a significant number of rows are deleted. You can enable the database manager to reorganize system both catalog tables and user tables.

Procedure

To enable your database for automatic reorganization:

1. Set the `auto_maint`, `auto_tbl_maint`, and `auto_reorg` database configuration parameters to ON. You can set the parameters to ON with these commands:
 - **db2 update db cfg for <db_name> using auto_maint on**
 - **db2 update db cfg for <db_name> using auto_tbl_maint on**
 - **db2 update db cfg for <db_name> using auto_reorg on**Replace <db_name> with the name of the database on which you want to enable automatic maintenance and reorganization.
2. Connect to the database, <db_name>.
3. Specify a reorganization policy. A reorganization policy is a defined set of rules or guidelines that dictate when automated table and index maintenance takes place. You can set this policy in one of two ways:
 - a. Call the `AUTOMAINT_SET_POLICY` procedure.
 - b. Call the `AUTOMAINT_SET_POLICYFILE` procedure.

The reorganization policy is either an input argument or file both of which are in an XML format..

Chapter 4. Default query concurrency management

To ensure that heavier workloads that use column-organized data do not overload the system when many queries are submitted simultaneously, there is a limit on the number of “heavyweight” queries that can run against a database at the same time.

You can implement this limit by using the default workload management concurrency threshold. This threshold is automatically enabled on new databases if you set the value of the **DB2_WORKLOAD** registry variable to **ANALYTICS**. You can manually enable the threshold on existing databases.

The processing of queries against column-organized tables is designed to run fast by using the highly parallelized in-memory processing of data. The trade-off for this high performance is that queries that reference column-organized tables also use a relatively large amount of memory and CPU when compared to similar queries processing row-organized table data. As such, the execution of queries that process column-organized tables is optimal when relatively few queries are admitted to the system at a time. When this occurs, the queries can use more processing power and memory, and contention for the processor caches is minimized.

Also, field experience with DB2 workload management has demonstrated that in analytic environments that support mixed workloads, controlling the admission of heavyweight queries into the system yields improvements in both system stability and overall performance. A mixed workload environment is one in which queries might vary widely in their degree of complexity and resource needs. The reason for the improvements is that resource overload on the system is avoided. When the limit on the number of heavyweight queries is reached, the remaining queries are queued and must wait until other queries are complete before they can run. Queuing can help to ensure system stability when many complex ad hoc queries are running on systems that do not have a specific workload management strategy. To further optimize the execution of mixed workloads on your system, review the full range of DB2 workload management capabilities. For details, see *Implementing DB2 workload management in a data warehouse*.

Default workload management objects for concurrency control

For concurrency control, several default workload management objects are created for new or upgraded databases.

- A service subclass, **SYSDEFAULTMANAGEDSUBCLASS**, under the **SYSDEFAULTUSERCLASS** superclass, where heavyweight queries run and can be controlled and monitored as a group.
- A **CONCURRENTDBCOORDACTIVITIES** threshold, **SYSDEFAULTCONCURRENT**, which is applied to the **SYSDEFAULTMANAGEDSUBCLASS** subclass to control the number of concurrently running queries that are running in that subclass.
- A work class set, **SYSDEFAULTUSERWCS**, and a work class, **SYSMANAGEDQUERIES**, which identify the class of heavyweight queries to control. The **SYSMANAGEDQUERIES** work class includes queries that are classified as **READ DML** (a work type for work classes) and that exceed a timeron threshold that reflects heavier queries.

- A work action set, SYSDEFAULTUSERWAS, and work action, SYSMAPMANAGEDQUERIES, which map all queries that fall into the SYSMANAGEDQUERIES work class to the SYSDEFAULTMANAGEDSUBCLASS service subclass.

When you create or upgrade a database, the following behavior applies to the new default objects:

- The work action set is enabled by default so that queries that meet the criteria that you specify for the SYSMANAGEDQUERIES work class run in the SYSDEFAULTMANAGEDSUBCLASS service subclass.
- If you set the value of the **DB2_WORKLOAD** registry variable to ANALYTICS, the threshold on the SYSDEFAULTMANAGEDSUBCLASS service subclass is enabled by default for newly created databases only.
- Running the DB2 Configuration Advisor causes the work action set and the work action to be enabled if you disabled either of them and causes the threshold to be enabled or disabled, depending on whether you set the **DB2_WORKLOAD** registry variable to ANALYTICS.
- You can manually enable or disable the work action set and the threshold. For more information about manual configuration, see “Adjusting the default configuration.”

In a customized workload management environment, the expectation is that this default concurrency control might not be required. You can leave it enabled to manage the work that maps to the SYSDEFAULTUSERCLASS superclass or disable it.

The default concurrency limit

A limit on the number of queries that can run concurrently should protect the system from overload but also avoid over-throttling the workload to a point where performance might suffer. This objective can be challenging when a system is running ad hoc analytic queries that range widely in complexity and impact. The default limit is based on a heuristic calculation that factors in system hardware attributes such as the number of CPU sockets, CPU cores, and threads per core. The limit is calculated based on the current host configuration when the default workload management objects for concurrency control are created during database creation or upgrade. The limit is recalculated when you run the DB2 Configuration Advisor against the database. Rerun the DB2 Configuration Advisor if you make changes to the host, such as enabling or removing CPU capacity.

Adjusting the default configuration

Although the objective is to provide basic query management for analytic data mart deployments without the need for manual intervention, there might be scenarios where adjustments to the default configuration are a good idea.

Only the following subset of workload management DDL statements is supported by the default workload management objects for concurrency control in DB2 Version 10.5.

You can enable or disable the mapping of heavyweight READ DML queries to the SYSDEFAULTMANAGEDSUBCLASS subclass, as shown in the following examples:

```
# Enable the default work action mapping so that
# heavyweight queries are mapped to the
# SYSDEFAULTMANAGEDSUBCLASS subclass
ALTER WORK ACTION SET SYSDEFAULTUSERWAS ENABLE

# Disable the default work action mapping so that all
# queries are mapped to the SYSDEFAULTSUBCLASS subclass
ALTER WORK ACTION SET SYSDEFAULTUSERWAS DISABLE
```

You can adjust the timeron range for the mapping of heavyweight READ DML queries, as shown in the following example:

```
# Modify the default work class timeron range from A to B
ALTER WORK CLASS SET SYSDEFAULTUSERWCS
    ALTER WORK CLASS SYSMANAGEDQUERIES FOR TIMERONCOST FROM <A> TO <B>
```

You can enable or disable the concurrency threshold on the SYSDEFAULTMANAGEDSUBCLASS subclass, as shown in the following example:

```
# Enable the concurrency threshold
ALTER THRESHOLD SYSDEFAULTCONCURRENT ENABLE

# Disable the concurrency threshold
ALTER THRESHOLD SYSDEFAULTCONCURRENT DISABLE
```

You can adjust the concurrency limit, as shown in the following example:

```
# Change the concurrency limit to [N]
ALTER THRESHOLD SYSDEFAULTCONCURRENT
    WHEN CONCURRENTDBCOORDACTIVITIES > [N] STOP EXECUTION
```

The following optional clauses are supported when you alter the SYSDEFAULTCONCURRENT threshold:

```
alter-threshold-exceeded-actions
AND QUEUEDACTIVITIES
```

Tuning the settings

The recommended method for tuning the default object settings is to examine the overall system resource usage with a full workload and to perform incremental adjustments that are based on whether the system resources appear to be under-utilized or over-utilized. A system with a run queue depth of approximately 10 times the number of CPU cores and physical memory usage under 100% is considered to be in a fully used and healthy state.

The following recommendations represent a starting point for tuning the default concurrency threshold value and the default work class timeron range when the system appears to be under-utilized or over-utilized. This guidance applies to a non-customized workload management environment. For a more comprehensive set of recommendations that applies to monitoring both system use and workload characteristics, see *Implementing DB2 workload management in a data warehouse*.

- If the system appears to be under-utilized, take the following steps:
 1. Examine the WLM_QUEUE_TIME_TOTAL metric, which is reported by various system-level table functions and the statistics event monitor, to determine whether queries in the system are accumulating time by waiting on concurrency thresholds.
 - a. If no such queue time is being accumulated, the system is running under peak capacity, and no tuning is necessary.
 - b. If queue time is being accumulated, take the following steps:

- 1) Monitor the amount of work that is running in the SYSDEFAULTSUBCLASS and SYSDEFAULTMANAGEDSUBCLASS classes.
 - 2) If it appears that too large a proportion of the workload is running within the SYSMANAGEDQUERIES class, consider incrementally increasing the TIMERONCOST minimum for the SYSMANAGEDQUERIES class.
2. If the distribution of managed and unmanaged work appears reasonable, consider incrementally increasing the concurrency limit that is specified by the SYSDEFAULTCONCURRENT threshold until system resource usage reaches the target level.
- If the system appears to be over-utilized, take the following steps:
 1. Monitor the amount of work that is running in the SYSDEFAULTSUBCLASS and SYSDEFAULTMANAGEDSUBCLASS classes.
 2. If it appears that too small a proportion of the workload is running within the SYSMANAGEDQUERIES class, consider incrementally decreasing the TIMERONCOST minimum for the SYSMANAGEDQUERIES class.
 3. If the distribution of managed and unmanaged work appears reasonable, consider incrementally decreasing the concurrency limit that is specified by the SYSDEFAULTCONCURRENT threshold until system resource usage is back within the target range.

Default workload management objects for concurrency control

For concurrency control, several default workload management objects are created for new or upgraded databases.

- SYSDEFAULTMANAGEDSUBCLASS, a service subclass under the SYSDEFAULTUSERCLASS service superclass, where heavyweight queries run and can be controlled and monitored as a group
- SYSDEFAULTCONCURRENT, a CONCURRENTDBCOORDACTIVITIES threshold that is applied to the SYSDEFAULTMANAGEDSUBCLASS subclass to control the number of concurrently running queries that are running in that subclass
- SYSDEFAULTUSERWCS (a work class set) and SYSMANAGEDQUERIES (a work class), which identify the class of heavyweight queries to control
- SYSDEFAULTUSERWAS (a work action set) and SYSMANAGEDQUERIES (a work action), which map all queries in the SYSMANAGEDQUERIES work class to the SYSDEFAULTMANAGEDSUBCLASS service subclass

Default managed subclass (SYSDEFAULTMANAGEDSUBCLASS)

You can query the SYSCAT.SERVICECLASSES catalog view to obtain information about the SYSDEFAULTMANAGEDSUBCLASS service subclass. The following table shows the columns, their values, and your ability to modify these values. For information about how to enable or disable the mapping of heavyweight READ DML queries to the SYSDEFAULTMANAGEDSUBCLASS service subclass, see Chapter 4, “Default query concurrency management,” on page 13.

Table 1. SYSDEFAULTMANAGEDSUBCLASS entry in SYSCAT.SERVICECLASSES view

Column	Value	Modifiable ¹
SERVICECLASSNAME	SYSDEFAULTMANAGEDSUBCLASS	No
PARENTSERVICECLASSNAME	SYSDEFAULTUSERCLASS	No

Table 1. *SYSDEFAULTMANAGEDSUBCLASS* entry in *SYSCAT.SERVICECLASSES* view (continued)

Column	Value	Modifiable ¹
SERVICECLASSID	4	No
PARENTID	3	No
CREATE_TIME	Time stamp of database creation or upgrade	No
ALTER_TIME	Time stamp of the last update of this object	No
ENABLED	Y	No
AGENTPRIORITY	-32768 (default)	Yes
PREFETCHPRIORITY	' ' (default)	Yes
MAXDEGREE	NULL	No
BUFFERPOOLPRIORITY	' ' (default)	Yes
INBOUNDCORRELATOR	NULL	No
OUTBOUNDCORRELATOR	NULL	Yes
COLLECTAGGACTDATA	N (none)	Yes
COLLECTAGGREQDATA	N (none)	Yes
COLLECTACTDATA	N (none)	Yes
COLLECTACTPARTITION	C (coordinator member of the activity)	Yes
COLLECTREQMETRICS	N (none)	Yes
CPUSHARES	1000	Yes
CPUSHARETYPE	H (hard shares)	Yes
CPU LIMIT	-1 (no CPU limit)	Yes
SORTMEMORYPRIORITY	NULL	No
SECTIONACTUALSOPTIONS	NN (not enabled)	Yes
COLLECTAGGUOWDATA	N (none)	Yes
REMARKS	NULL	Yes
Note: 1. You can modify the value of this column by using the ALTER SERVICE CLASS statement. To issue this statement, DBADM or WLMADM authority and, for COLLECT clauses, SQLADM authority is required.		

For more information, see SYSCAT.SERVICECLASSES.

Default user work class set (SYSDEFAULTUSERWCS)

You can query the SYSCAT.WORKCLASSSETS catalog view to obtain information about the SYSDEFAULTUSERWCS work class set. The following table shows the columns, their values, and your ability to modify these values. For more information, see Chapter 4, “Default query concurrency management,” on page 13.

Table 2. *SYSDEFAULTUSERWCS* entry in *SYSCAT.WORKCLASSSETS* view

Column	Value	Modifiable ¹
WORKCLASSSETNAME	SYSDEFAULTUSERWCS	No
WORKCLASSSETID	2147483647	No

Table 2. SYSDEFAULTUSERWCS entry in SYSCAT.WORKCLASSSETS view (continued)

Column	Value	Modifiable ¹
CREATE_TIME	Time stamp of database creation or upgrade	No
ALTER_TIME	Time stamp of the last update of this object	No
REMARKS	NULL	Yes
Note: 1. You can modify the value of this column by using the ALTER SERVICE CLASS statement. To issue this statement, DBADM or WLMADM authority and, for COLLECT clauses, SQLADM authority is required.		

For more information, see SYSCAT.WORKCLASSSETS.

Managed queries work class (SYSMANAGEDQUERIES)

You can query the SYSCAT.WORKCLASSES catalog view to obtain information about the SYSMANAGEDQUERIES work class. The following table shows the columns, their values, and your ability to modify these values. For more information, see Chapter 4, “Default query concurrency management,” on page 13.

Table 3. SYSMANAGEDQUERIES entry in SYSCAT.WORKCLASSES view

Column	Value	Modifiable ¹
WORKCLASSNAME	SYSMANAGEDQUERIES	No
WORKCLASSSETNAME	SYSDEFAULTUSERWCS	No
WORKCLASSID	2147483647	No
WORKCLASSSETID	2147483647	No
CREATE_TIME	Time stamp of database creation or upgrade	No
ALTER_TIME	Time stamp of the last update of this object	No
EVALUATIONORDER	1	No
Note: 1. You can modify the value of this column by using the ALTER SERVICE CLASS statement. To issue this statement, DBADM or WLMADM authority and, for COLLECT clauses, SQLADM authority is required.		

For more information, see SYSCAT.WORKCLASSES.

You can also query the SYSCAT.WORKCLASSATTRIBUTES catalog view to obtain information about the SYSMANAGEDQUERIES work class. The following tables show the columns, their values, and your ability to modify these values.

Table 4. SYSMANAGEDQUERIES entries in SYSCAT.WORKCLASSATTRIBUTES view. Attribute 1

Column	Value	Modifiable ¹
WORKCLASSNAME	SYSMANAGEDQUERIES	No
WORKCLASSSETNAME	SYSDEFAULTUSERWCS	No
WORKCLASSID	2147483647	No

Table 4. SYSMANAGEDQUERIES entries in SYSCAT.WORKCLASSATTRIBUTES view (continued). Attribute 1

Column	Value	Modifiable ¹
WORKCLASSSETID	2147483647	No
TYPE	1 (work type attribute)	No
VALUE1	2 (read activities)	No
VALUE2	NULL	No
VALUE3	NULL	No
Note: 1. You can modify the value of this column by using the ALTER SERVICE CLASS statement. To issue this statement, DBADM or WLMADM authority and, for COLLECT clauses, SQLADM authority is required.		

Table 5. SYSMANAGEDQUERIES entries in SYSCAT.WORKCLASSATTRIBUTES view. Attribute 2

Column	Value	Modifiable ¹
WORKCLASSNAME	SYSMANAGEDQUERIES	No
WORKCLASSSETNAME	SYSDEFAULTUSERWCS	No
WORKCLASSID	2147483647	No
WORKCLASSSETID	2147483647	No
TYPE	2 (timeron cost attribute)	No
VALUE1	Minimum timeron cost, computed at database creation or when the DB2 Configuration Advisor is run	Yes
VALUE2	-1 (maximum timeron cost; unbounded)	Yes
VALUE3	NULL	No
Note: 1. You can modify the value of this column by using the ALTER SERVICE CLASS statement. To issue this statement, DBADM or WLMADM authority and, for COLLECT clauses, SQLADM authority is required.		

For more information, see SYSCAT.WORKCLASSATTRIBUTES.

Default user work action set (SYSDEFAULTUSERWAS)

You can query the SYSCAT.WORKACTIONSETS catalog view to obtain information about the SYSDEFAULTUSERWAS work action set. The following table shows the columns, their values, and your ability to modify these values. For information about how to enable or disable the mapping of heavyweight READ DML queries to the SYSDEFAULTMANAGEDSUBCLASS service subclass, see Chapter 4, “Default query concurrency management,” on page 13.

Table 6. SYSDEFAULTUSERWAS entry in SYSCAT.WORKACTIONSETS view

Column	Value	Modifiable ¹
ACTIONSETNAME	SYSDEFAULTUSERWAS	No
ACTIONSETID	2147483647	No
WORKCLASSSETNAME	SYSDEFAULTUSERWCS	No

Table 6. SYSDEFAULTUSERWAS entry in SYSCAT.WORKACTIONSETS view (continued)

Column	Value	Modifiable ¹
WORKCLASSSETID	2147483647	No
CREATE_TIME	Time stamp of database creation or upgrade	No
ALTER_TIME	Time stamp of the last update of this object	No
ENABLED	Y	Yes
OBJECTTYPE	b (work action set applies to service superclass)	No
OBJECTNAME	SYSDEFAULTUSERCLASS	No
OBJECTID	3 (service class ID of SYSDEFAULTUSERCLASS)	No
REMARKS	NULL	Yes
Note: 1. You can modify the value of this column by using the ALTER WORK ACTION SET statement. To issue this statement, DBADM or WLMADM authority is required.		

For more information, see SYSCAT.WORKACTIONSETS.

Managed queries work action (SYSMAPMANAGEDQUERIES)

You can query the SYSCAT.WORKACTIONS catalog view to obtain information about the SYSMAPMANAGEDQUERIES work action. The following table shows the columns, their values, and your ability to modify these values. For more information, see Chapter 4, “Default query concurrency management,” on page 13.

Table 7. SYSMAPMANAGEDQUERIES entry in SYSCAT.WORKACTIONS view

Column	Value	Modifiable ¹
ACTIONNAME	SYSMAPMANAGEDQUERIES	No
ACTIONID	2147483647	No
ACTIONSETNAME	SYSDEFAULTUSERWAS	No
ACTIONSETID	2147483647	No
WORKCLASSNAME	SYSMANAGEDQUERIES	No
WORKCLASSID	2147483647	No
CREATE_TIME	Time stamp of database creation or upgrade	No
ALTER_TIME	Time stamp of the last update of this object	No
ENABLED	Y	Yes
ACTIONTYPE	M (map activities to service subclass)	No
REFOBJECTID	4 (service class ID for SYSDEFAULTMANAGEDSUBCLASS)	No
REFOBJECTTYPE	NULL	No
SECTIONACTUALSOPTIONS	NN (not enabled)	No

Table 7. SYSMAPMANAGEDQUERIES entry in SYSCAT.WORKACTIONS view (continued)

Column	Value	Modifiable ¹
Note:		
1. You can modify the value of this column by using the ALTER WORK ACTION SET statement. To issue this statement, DBADM or WLMADM authority is required.		

For more information, see SYSCAT.WORKACTIONS.

Default concurrent threshold (SYSDEFAULTCONCURRENT)

You can query the SYSCAT.THRESHOLDS catalog view to obtain information about the SYSDEFAULTCONCURRENT threshold. The following table shows the columns, their values, and your ability to modify these values. For more information, see Chapter 4, “Default query concurrency management,” on page 13.

Table 8. SYSDEFAULTCONCURRENT entry in SYSCAT.THRESHOLDS view

Column	Value	Modifiable ¹
THRESHOLDNAME	SYSDEFAULTCONCURRENT	No
THRESHOLDID	2147483647	No
ORIGIN	U (not created by work action)	No
THRESHOLDCLASS	A (aggregate)	No
THRESHOLDPREDICATE	CONCDBC (concurrent database coordinator activities)	No
THRESHOLDPREDICATEID	90	No
DOMAIN	SB (service subclass)	No
DOMAINID	4 (service class ID of SYSDEFAULTMANAGEDSUBCLASS)	No
ENFORCEMENT	D (database)	No
QUEUING	Y	Yes
MAXVALUE	Computed at database creation or when the DB2 Configuration Advisor is run, based on the host hardware configuration	Yes
DATATAGLIST	NULL	No
QUEUESIZE	-1 (unbounded)	Yes
OVERFLOWPERCENT	-1	No
COLLECTACTDATA	N (none)	Yes
COLLECTACTPARTITION	C (coordinator member of the activity)	Yes
EXECUTION	S (stop execution when queue is full)	Yes
REMAPSCID	0	No
VIOLATIONRECORDLOGGED	NULL	No
CHECKINTERVAL	-1	No
ENABLED	Y	Yes
CREATE_TIME	Time stamp of database creation or upgrade	No
ALTER_TIME	Time stamp of the last update of this object	No

Table 8. SYSDEFAULTCONCURRENT entry in SYSCAT.THRESHOLDS view (continued)

Column	Value	Modifiable ¹
REMARKS	NULL	Yes
Note: 1. You can modify the value of this column by using the ALTER THRESHOLD statement. To issue this statement, DBADM or WLMADM authority and, for COLLECT clauses, SQLADM authority is required.		

For more information, see SYSCAT.THRESHOLDS.





Chapter 5. IBM Data Studio

IBM Data Studio provides application developers with a single integrated development environment that can be used to create, deploy, and debug data-centric applications. Built to extend the Eclipse framework and SQL model components, it combines Eclipse technology and shared repository extensions for database development.

IBM Data Studio consist of the following components:

- The IBM Data Studio client, which is an Eclipse-based tool that provides an integrated development environment for database and instance administration, routine and Java application development, and query tuning tasks. It can be installed with other IBM software products to share a common environment.
- The IBM Data Studio web console, which is a web-based tool with health and availability monitoring, job creation, and database administration tasks.

Related information:

-  [IBM Data Studio documentation](#)
-  [Features in IBM Data Studio](#)
-  [IBM Data Studio product Web page](#)
-  [Download IBM Data Studio](#)

Using IBM Data Studio for key tasks

IBM Data Studio includes support for key tasks across the data management lifecycle, including administration, application development, and query tuning.

Administer databases, monitor health, and run jobs

IBM Data Studio provides database management and maintenance support, including object, change, and authorization management, scripting, basic health and availability monitoring, and job scheduling to automate changes to the database.

Develop database applications

IBM Data Studio helps developers and database administrators develop, debug, and deploy database applications and database routines (SQL or Java stored procedures and user-defined functions).

Data access development support

If data access development support is enabled for Java applications, developers and database administrators can use IBM Data Studio to understand the relationship between database objects, source code, and SQL statements that are in the source code. Data access development support also provides client metrics for SQL statements.

pureQuery support

If pureQuery support is enabled, developers can use the integrated InfoSphere Optim pureQuery Runtime and the pureQuery APIs to create Java applications. With the APIs, developers can use the integrated Java editor and simple pureQuery syntax to create a simple Java data access layer with the data access object (DAO) pattern.

Tune queries

IBM Data Studio includes basic query tuning tools, such as query formatting, access path graphs, and statistics advice to help developers and SQL tuners create higher performance queries. You can also use IBM Data Studio to access the tuning features of IBM InfoSphere Optim Query Workload Tuner when you connect to a DB2 database or subsystem on which a license for InfoSphere Optim Query Workload Tuner is active.

Scenario: IBM Data Studio in a team environment

You can install multiple instances of the Data Studio components to mirror your organization and support the structure of your enterprise. For example, in an organization with users with different roles and access privileges, your team can install multiple instances of the Data Studio client.

The following illustration shows a complex use scenario that consists of a database designer and multiple database administrators and application developers who all have different access privileges to the test and production database and to the Data Studio web console.

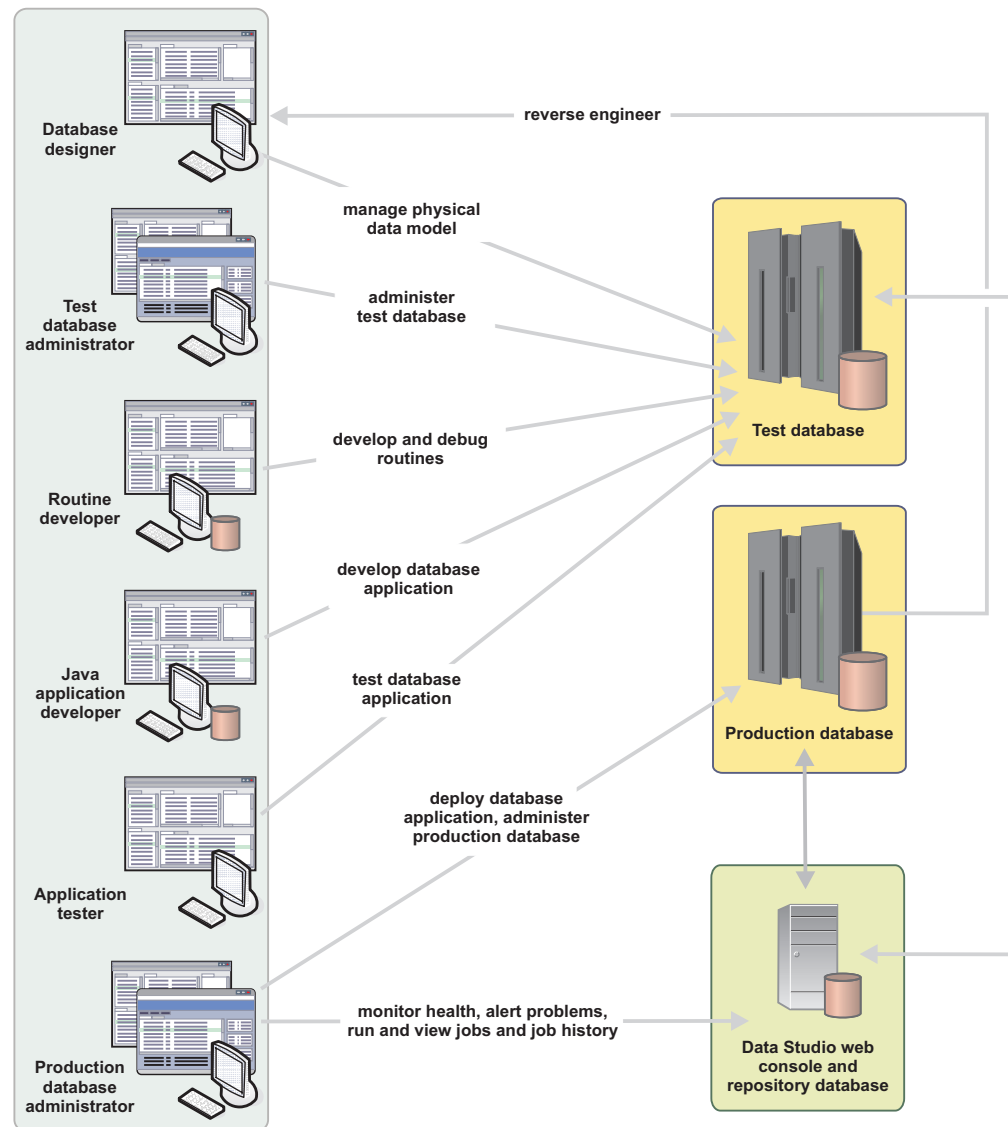


Figure 1. Topology diagram of a complex installation and use scenario.

Most users in this scenario have access to only the test database, but the test database administrator has additional access to the Data Studio web console. Similarly, the production database administrator has access to both the production database and the Data Studio web console. The administrators with access to the Data Studio web console, can monitor database health, manage jobs, and manage and share database connection information across the organization.

For information about other tools and solutions that can help you with the tasks and responsibilities throughout the data management lifecycle, see Data management and application development tools.

IBM Data Studio client

The IBM Data Studio client is built on Eclipse technology and provides an integrated development environment for database and instance administration, routine and Java™ application development, and query tuning.

The IBM Data Studio client is one of two IBM Data Studio components: the client and the web console. For information about the IBM Data Studio web console component, which you use for job management and health monitoring, see “IBM Data Studio web console” on page 28.

The IBM Data Studio client includes the following features:

Activities and perspectives

The tools that you use as a database administrator or application developer depend on your role. Data Studio provides two primary ways to access these tool sets: activities and perspectives.

Activities

Use the activity menu in the toolbar to switch between Data Studio activities. These preconfigured activities correspond to a subset of the Data Studio perspectives. The following activities are available: Administer Databases, Develop SQL and Routines, Develop Java Applications, Tune Queries, and Run SQL.

You can return to your preset home activity from any perspective by clicking the home activity button in the toolbar.

Perspectives

You can also access other tools that you need for your role by switching to other perspectives from the **Window > Open Perspective** menu. For example, the primary perspective for database administration is the Database Administration perspective, and the perspectives for application development are the Database Development and Java perspectives. Depending on your role, other perspectives that you can use include the Data, Java, SQL and Routine Development, and Query Tuning perspectives.

Getting started tools

The following tools can help you get started with the IBM Data Studio client:

- If you are new to the Eclipse development environment, view the Eclipse Basic tutorial. Click the link, then click the **Show in Table of Contents** icon in the information center to see the entire tutorial.
- Several tutorials are available to help you get started with IBM Data Studio. To access tutorials, expand the Tutorials category in the Contents pane of information center.
- Use the Task Launcher in the IBM Data Studio client to view and start many getting started tasks and other key tasks in the product. Task Launcher opens when you start IBM Data Studio, or you can open it by clicking **Help > Task Launcher**.

Database object management

Changing database objects requires determining which changes need to be made, specifying those changes, evaluating the effects of those changes, and then deploying them.

An editable Properties view and the Review and Deploy dialog box provides a consistent way to create, alter, and drop objects. You can also manage the privileges objects for various types of database servers. After you define your

changes in the Properties view, IBM Data Studio automatically generates the commands that can make the changes. The generated commands are displayed in the Review and Deploy dialog box, where you can review the commands and run them.

More robust change management features are provided for DB2 for Linux, UNIX, and Windows databases because a change plan is used to manage the changes. A change plan makes more complex changes possible and you can use it to change more than one object at a time. IBM Data Studio manages the dependent objects and takes resulting actions to address any side effects that are caused by your database object changes. With change plans, you can also preserve your data across database changes, undo your database changes, and track your changes with a version control system.

Data application developer features

For data application developers, IBM Data Studio provides the following key features. Working in a data development project in the Data Project Explorer, you can:

- Create, test, debug, and deploy routines, such as stored procedures and user-defined functions. See [Developing database routines](#).
- Create, edit, and run SQL queries. Use the SQL Query builder and the SQL and XQuery editor .
- View and capture performance data for SQL statements. See [Import and view SQL performance data](#) and [Run SQL statements and capture performance data](#).
- Use Visual Explain to diagram access plans. See [Diagramming access plans with Visual Explain](#).
- Use query tuning features to improve the performance of SQL statements in applications. See [Tuning single SQL statements](#).
- Debug stored procedures. See [Use the Routine debugger](#).
- Create web services that expose database operations (SQL SELECT and DML statements, XQuery expressions, or calls to stored procedures) to client applications. See [Developing and deploying Web services](#).
- Develop SQLJ applications in a Java project. See [Developing SQLJ applications](#).
- Develop XML applications. See [Use wizards and editors to develop XML applications](#).

Query tuning features

With the IBM Data Studio client, you can format SQL statements so that they are easier to read, generate visual representations of access plans, and get recommendations for collecting statistics on the objects that a statement references. You can also generate a report that summarizes information about the access plan and includes the recommendations.

If you connect the IBM Data Studio client to a DB2 database or subsystem on which a license for InfoSphere® Optim Query Workload Tuner is active, you can use the full set of tuning features.

Team features

If you are working on a large team, you can share data development projects by using supported code control systems, and you can share database connection information.

For more information, see:

- Supported source code control systems.
- Share database connection information by importing and exporting this information to XML files.

Shell-sharing with other Eclipse-based products

Shell-sharing (sharing a common environment) with other Eclipse-based products makes it easy to share the functions between products from one interface. If you install the IBM Data Studio client into the same product group as a compatible product, you install only one version of Eclipse that shares the components of each product. Shell-sharing saves disk space and avoids duplicating components that are already built into other products.

Another benefit of shell sharing is the ability to have products interact with each other, which makes each product stronger than if they were run alone. For example, the following shell-sharing scenario shows the strength of using IBM InfoSphere Data Architect and IBM Data Studio together:

1. Shell-share InfoSphere Data Architect with the IBM Data Studio client.
2. Create glossary models to standardize your naming conventions by using InfoSphere Data Architect.
3. Use the database administration features of the IBM Data Studio client to ensure that those naming conventions are followed.

To shell-share products, the base Eclipse versions must be the same. For example, you cannot shell-share an Eclipse version 3.6-based product with an Eclipse version 4.2-based product.

For more information, see:

- Coexistence considerations
- Limitations for sharing a common environment between IBM products based on Eclipse
- IBM Software products installed together that share a common environment

For more details about the installation packages that are available for IBM Data Studio, see the product web page.

IBM Data Studio web console

IBM Data Studio web console provides health and availability monitoring features and job creation and management functions for DB2 for Linux, UNIX, and Windows and DB2 for z/OS[®] databases. Use the health pages of the web console to view alerts, applications, utilities, storage, and related information and use the job manager pages to create and manage script-based jobs across your databases.

IBM Data Studio web console is available as a stand-alone web interface or integrated with the IBM Data Studio client.

Tip: You can use IBM Data Studio web console in single-user mode to test the product in a controlled environment, or in multi-user mode to share monitoring features across your production servers.

- In single-user mode, log in to the IBM Data Studio web console as a single user with full administrative rights.

- In multi-user mode, configure access to the web console and permissions to monitor and perform other actions on the specific databases for users and groups.

Database administration with IBM Data Studio

You can run database administration commands for hosts, instances, and databases that are displayed in the Administration Explorer and for databases, table spaces, tables, and indexes that are displayed in the Object List.

As a database administrator, you might be responsible for maintaining, managing, and administering DB2 instances, databases, and database objects such as table spaces, tables, and views. For example, your backup and recovery strategy might require you to take periodic backups of your databases. As another example, over time, the data in your tables might become fragmented, which can increase the size of your tables and indexes as the records are distributed over more and more data pages. To reclaim wasted space and improve data access, you likely will need to reorganize your tables and indexes.

Managing and maintaining your database systems might require you to run database administration commands, which include:

- DB2 commands
- System commands
- Utilities
- SQL statements

When the Database Administration feature of IBM Data Studio is installed, task assistants are available for DB2 for Linux, UNIX, and Windows databases. These task assistants guide you through typical database administration tasks. From the Administration Explorer, you can do the following types of administration tasks:

- Maintenance mode management for DB2 pureScale hosts
- Instance management, including starting and stopping instances
- Database management, including creating and dropping databases, configuring logging or automatic maintenance, setting up and managing the DB2 High Availability Disaster Recovery (HADR) feature, and backing up, restarting, and recovering databases

When you click a data object folder (also called a *flat folder* in the Administration Explorer) to display database objects in the Object List, you can do the following types of administration tasks:

- Table space management, including backing up, restoring, and recovering table spaces
- Table management, including unloading and loading data, and setting the integrity of tables
- Index management, including reorganizing indexes
- Package management, including rebinding packages

You can also manage databases in the Object List.

When you right-click an object in the Administration Explorer or the Object List, a context-sensitive menu displays the list of the database administration commands that are available for that object. When you select a database administration command for that object, a database administration task assistant is displayed. The task assistant guides you through the process of setting any options for the

database administration command, previewing the commands that are automatically generated, and running the commands for the object.

For databases that are not DB2 for Linux, UNIX, and Windows, you can use the SQL and XQuery editor to create and run your database administration commands.

Database administration for partitioned databases

The DB2 Database Partitioning feature (DPF) allows the partitioning of a database into two or more partitions that can reside on either the same server or on a different server.

For partitioned databases, the task assistants that guide you through the process of setting up the options for the commands include the ability to specify whether to run the commands against all of the partitions, one or more specific partitions, or partition groups. You can also choose to run the commands against the partitions in parallel, which is particularly useful for long-running commands. If you save the commands to a script, the commands will be run sequentially.

The control to run commands on individual partitions gives you more flexibility in managing your databases and resources. The granularity of an operation can determine how long it will take. For example, if a database has hundreds of partitions, backing up sets of partitions at different times or on different days might make more sense than backing up all of the partitions at the same time. As another example, depending on the system resources that are available to each partition, you might want to set certain configuration parameters across non-catalog partitions and customize the parameters for the catalog partition to optimize performance.

Database administration for the DB2 pureScale Feature for DB2 Enterprise Server Edition

In a DB2 pureScale environment, task assistants provide these additional database administration operations for members and cluster caching facilities (CFs):

Start To start members or CFs, select **Start** on the context-sensitive menu for the instance object. You can start selected members or CFs, all members and CFs, or an instance on a host.

Stop To stop members or CFs that are currently active, select **Stop** on the context-sensitive menu for the instance object. You can stop selected members or CFs, all the members and CFs, or an instance on a host.

Quiesce
To quiesce members, select **Stop** on the context-sensitive menu for the instance object and select the option to **Quiesce member with timeout**.

Configure
To change the configuration parameters for one or more members, select **Configure** on the context-sensitive menu for the instance object.

In addition, you can manage the maintenance mode for a DB2 pureScale host.

Administering databases with task assistants

IBM Data Studio provides dialogs that are called task assistants that help you create and run database administration commands for objects in DB2 for Linux, UNIX, and Windows databases. For example, you can use task assistants to start

and stop databases and instances, configure database parameters, reorganize tables and indexes, back up and restore databases or table spaces, and import and export table data.

Before you begin

To run a database administration command for an object, you must have the appropriate permission or authorization for the object, and you must have a connection to the database that contains the object.

For Linux operating systems

Before users who do not have DB2 instance level privileges can use the DB2 client CLP to run commands, including commands for importing, loading, exporting, or unloading tables, the DB2INSTANCE system environment variable must be set.

Note: You must set the DB2INSTANCE system environment variable for each time that you log in or open a command terminal.

To set the DB2INSTANCE system environment variable:

1. Ensure that the Data Studio client is closed and not running before you run the script.
2. Set the environment variable at the instance level by running one of the following scripts:
 - For a Bourne or Korn shell, run: `db2profile`
 - For a C shell, run: `db2cshrc`
3. Start IBM Data Studio client.

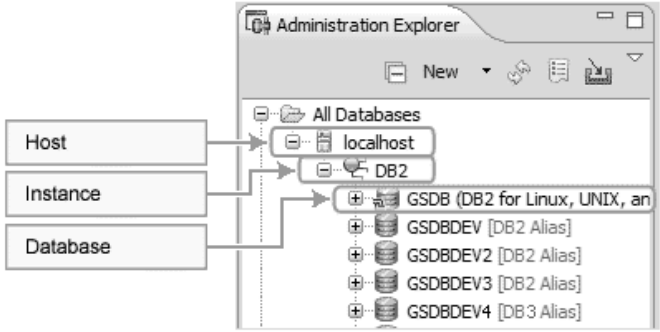
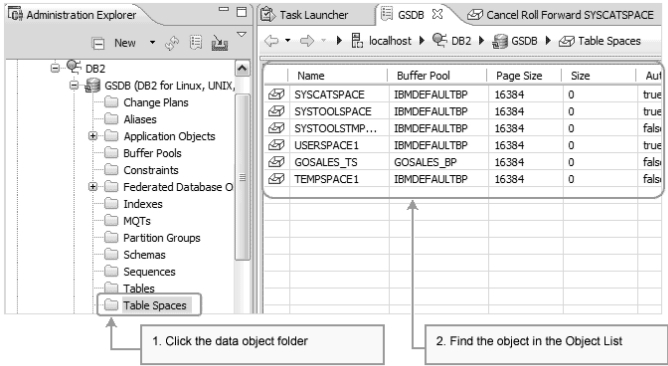
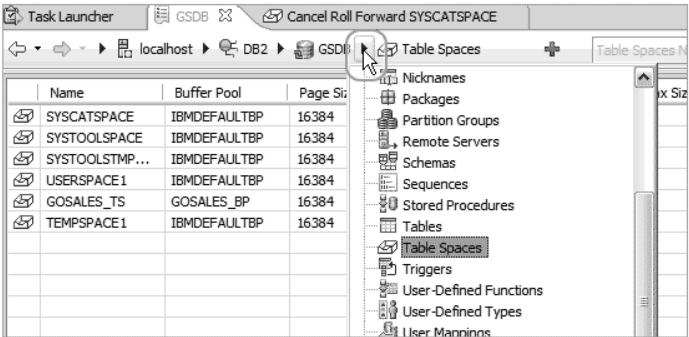
For detailed information about setting the system environment variable, see the *Setting environment variables outside the profile registries on Linux and UNIX operating systems* topic in the IBM DB2 online information for your version of the DB2 database.

Procedure

To open the task assistant for the command, specify additional settings for the command, and run the generated commands, complete the following steps:

1. Find the object that you want to work with. You can find the object either in the Administration Explorer or the Object List.

Table 9. Which view to use to find objects

View and objects	Example
Administration Explorer Hosts Instances Databases	<p>From the Administration Explorer, you can open a task assistant for hosts, instances, and databases.</p>  <p>Figure 2. Example of the Administration Explorer</p>
Object List Databases Table spaces Indexes Views Aliases Packages	<p>When you click a database or a data object folder in the Administration Explorer, the list of objects is displayed in the Object List.</p>  <p>Figure 3. Example of selecting the Table Spaces folder to display the table spaces in the Object List</p> <p>Tip: If the Object List is already open for the database, you can use the drop-down arrow that is displayed after the database name in navigation breadcrumb trail to display other objects in the Object List.</p>  <p>Figure 4. Example of using a drop-down arrow in the Object List to select other objects to display</p>

2. Right-click the object and select the command to run from the context-sensitive menu.

For example, the following figure shows how to back up the GSDB database.

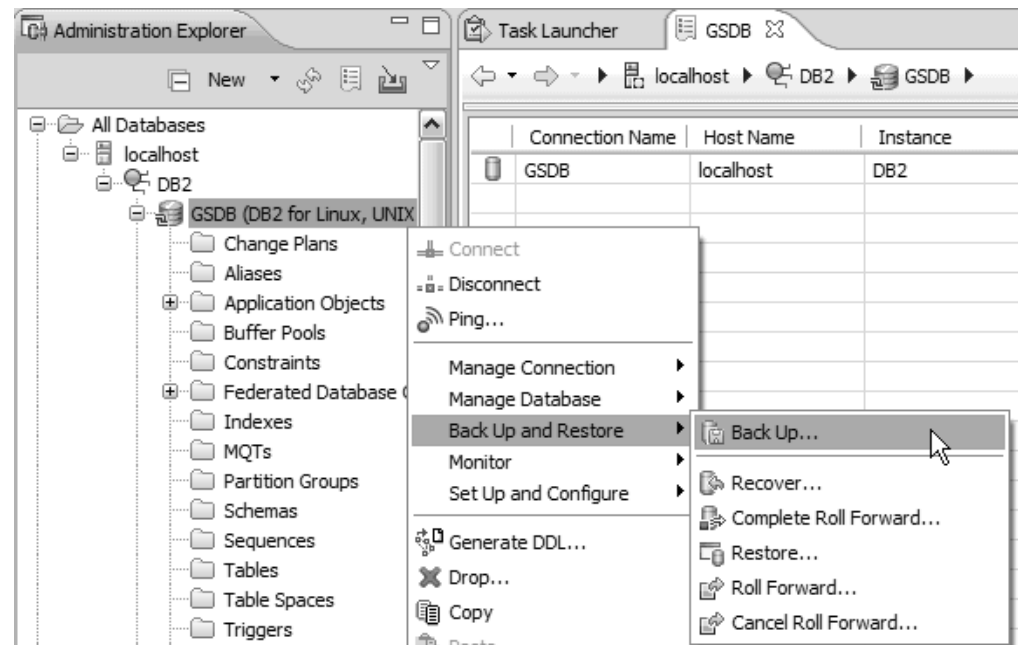


Figure 5. Example of the context-sensitive menu for databases and selecting to back up a database

The task assistant opens for the database administration task that you selected. Each task assistant has four sections: **Connection**, **Settings**, **Command**, and **Messages**. The following graphic shows how the **Connection**, **Command**, and **Messages** sections are expandable. The **Settings** section is always expanded.

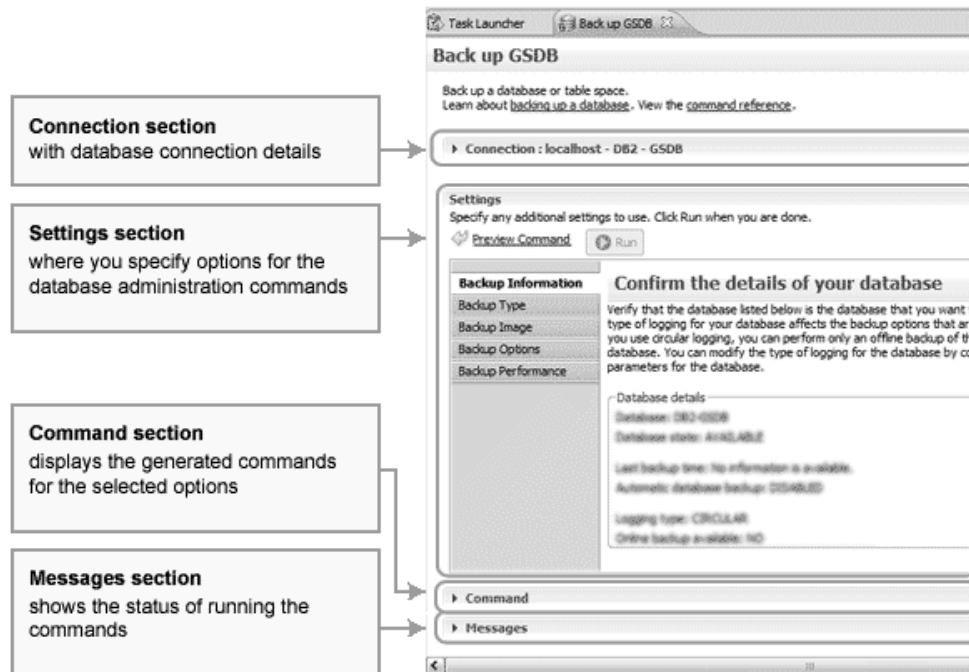


Figure 6. Example of a task assistant with its four sections

3. In the **Settings** section, specify the options for the command:
 - a. Click each of the tabs to step through the process of specifying the settings and options to use in the command.
 - b. Click **Preview Command** to shift down to and expand the **Command** section, where the generated commands that are based on the options that you specified are displayed.

For example, the following figure shows selecting the **Backup Performance** tab to specify options that improve the performance of the backup operation.

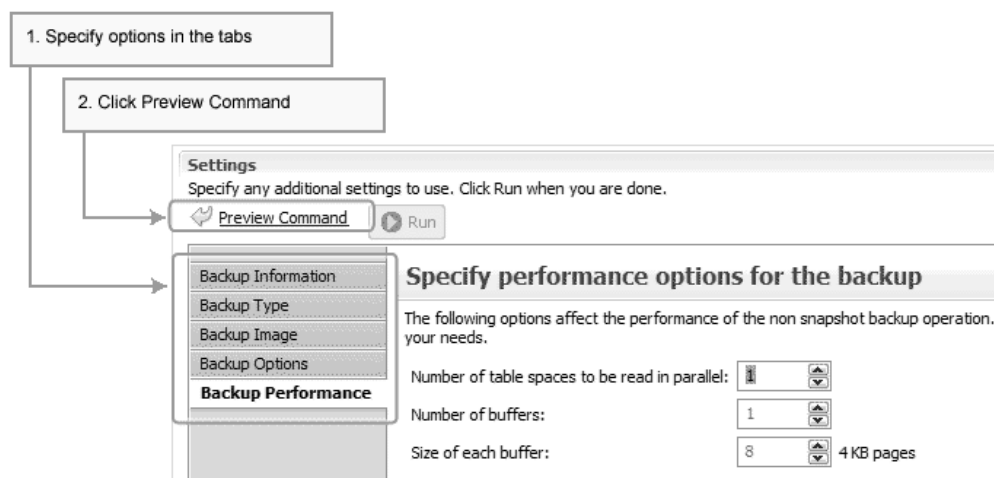


Figure 7. Example of selecting a tab and specifying options on that tab

4. In the **Command** section, review the commands that were generated and then run them.



Figure 8. Example of the commands being displayed in the Command section

If you are satisfied with the displayed commands, click **Run** to run them.

Tip: In some cases, you might want to edit the displayed commands. Click **Edit** to open the SQL and XQuery editor where you can edit and run the commands. You might also want to schedule a time to run the commands.

The focus of the task assistant shifts to the **Messages** section.

5. In the **Messages** section, monitor the progress of the commands that are being run in the progress bar and review any messages that are issued.



Figure 9. Example of the Messages section while the commands are being run

To view detailed information about any command that does not run successfully, click the message number or SQL code that is displayed.

Results

The commands to perform your database administration task were run.

Database administration commands that you can run from task assistants

Use the numerous task assistants to guide you through the process of setting options for common database administration commands, reviewing the automatically generated commands, and running the commands.

For each object, the following table shows which database administration commands are supported by a task assistant. Task assistants are available only for databases on DB2 for Linux, UNIX, and Windows.

Table 10. Task assistant support for DB2 for Linux, UNIX, and Windows database administration commands

Action	Database administration command	Description
Hosts		
Manage DB2 pureScale Host Maintenance Mode	db2cluster with one of the following options <ul style="list-style-type: none"> • -cm -enter -maintenance [-all] • -cm -exit -maintenance [-all] • -cfs -enter -maintenance [-all] • -cfs -exit -maintenance [-all] 	<p>Either puts or removes the target host or all hosts from maintenance mode. The target host must be in maintenance mode to apply software updates to DB2 cluster services.</p> <p>The host must have a least one DB2 pureScale instance, and at least one connection profile was created and connected.</p>
Verify DB2 cluster services status	db2cluster with one of the following options <ul style="list-style-type: none"> • -cm -verify -resources • -cfs -verify -configuration • -cm -verify -maintenance • -cfs -verify -maintenance • -cfs -verify -configuration -filesystem <i>fsname</i> • -cfs -list -filesystem 	<p>Verify the status and configuration of DB2 cluster services. The task assistant can do the following tasks:</p> <ul style="list-style-type: none"> • Verify that the resource model for the instance is correct, and that there are no inconsistencies in the resource model. • Verify the configuration of the current file system cluster. • Display if the cluster manager is offline on the host so that the binaries can be updated. • Display if the shared file system cluster host is offline to allow for updates to the binaries. • Verify the configuration of the current file system cluster. • List the current file systems.

Table 10. Task assistant support for DB2 for Linux, UNIX, and Windows database administration commands (continued)

Action	Database administration command	Description
Manage DB2 cluster services configuration	db2cluster with one of the following options <ul style="list-style-type: none"> • -cm -set -option -pprimary <i>host name</i> • -cm -set -option HostFailureDetectionTime [1..10] • -cm -set -option autofailback [ON/OFF] • -cm -set -tiebreaker -disk [<i>disk name</i> PVID=<i>pvid</i>] • -cfs -set -option -tiebreaker -disk <i>disk name</i> 	Manage the configuration of the DB2 cluster services. The task assistant can do the following tasks: <ul style="list-style-type: none"> • Set a new preferred primary value for DB2 cluster services. The option specifies which cluster caching facility DB2 cluster services will attempt to start in the primary role. Attention: Make sure to check that the preferred primary value was changed after you run the command by looking at the output of the command in the SQLResults view. The db2cluster command is not recognized by DB2 CLP, so the command will run as an external command. DB2 CLP always reports success for any external command irrespective of the actual result of the external command. This behavior will be addressed in future releases. • Set how long it takes to detect a host failure or network partition. The value specifies the interval for detecting host failure. • Enable or disable automatic failback for the cluster. • Set the cluster manager quorum type to a disk tiebreaker.
Instances		
Configure	UPDATE DATABASE MANAGER CONFIGURATION	Modifies individual entries in the database manager configuration file.
Quiesce	QUIESCE	Forces all users off the specified instance and puts the instance into quiesced mode.
Start	db2start	Starts the DB2 instance.
Stop	db2stop	Stops the DB2 instance.
Unquiesce	UNQUIESCE	Restores user access to instances that were quiesced for maintenance or other reasons.
Databases		
Backup	BACKUP DATABASE	Creates a backup copy of a database or table space.
Configure	UPDATE DATABASE CONFIGURATION	Modifies individual entries in a specific database configuration file.
Configure Automatic Maintenance	UPDATE DATABASE CONFIGURATION	Enables or disables the various automatic maintenance activities that can be performed and defines a maintenance interval and window in which the activities can occur. Maintenance activities can occur during the maintenance window only if DB2 determines that the maintenance is required.

Table 10. Task assistant support for DB2 for Linux, UNIX, and Windows database administration commands (continued)

Action	Database administration command	Description
Configure Database Logging	UPDATE CONFIGURATION LOGGING	Modifies the data logging options for your database, such as the type of logging to use, the size of the log files, and the location where log files will be stored.
Create	CREATE DATABASE	Creates a database with either automatic or manual storage.
Drop	DROP DATABASE	Deletes the database contents and all log files for the database, uncatalogs the database, and deletes the database subdirectory.
Revalidate	SYSCAT. INVALIDOBJECTS ADMIN_ REVALIDATE_DB _OBJECTS	<p>Lists all invalid objects for your database, and lets you revalidate these objects. The following object types are supported:</p> <p>DB2 V10.1 and higher</p> <ul style="list-style-type: none"> • COLUMN MASK • GLOBAL VARIABLE • ROUTINE • ROW PERMISSION • TRIGGER • USER-DEFINED DATA TYPE • VIEW <p>DB2 V9.7</p> <ul style="list-style-type: none"> • GLOBAL VARIABLE • ROUTINE • TRIGGER • USER-DEFINED DATA TYPE • VIEW
HADR Setup	Various	Sets up the High Availability Disaster Recovery (HADR) feature for your database. The HADR feature ensures that changes to the database can be replicated to one or more standby databases. A standby database takes over in the event of a failure on the primary system.
HADR Manage	START HADR, STOP HADR, TAKEOVER HADR	Starts and stops HADR operations on either the primary database or a standby database. You can also instruct a standby database to take over as the primary database in the event of a failure on the primary system.
List or Force Applications	FORCE APPLICATIONS	Forces local or remote users or applications off of the system to allow for maintenance on a server.
Manage Storage	ALTER DATABASE ADD STORAGE	Specifies that one or more new storage locations are to be added to the collection of storage locations that are used for automatic storage table spaces.
Quiesce	QUIESCE	Forces all users off of the specified database and puts the database into quiesced mode.

Table 10. Task assistant support for DB2 for Linux, UNIX, and Windows database administration commands (continued)

Action	Database administration command	Description
Recover	RECOVER DATABASE	Restores and rolls forward a database to a particular point in time or to the end of the logs.
Restart	RESTART DATABASE	Restarts a database that has been abnormally terminated and left in an inconsistent state.
Restore	RESTORE DATABASE	Re-creates a damaged or corrupted database that was backed up with the DB2 backup utility.
Roll Forward	ROLLFORWARD DATABASE	Recovers a database by applying transactions that were recorded in the database log files.
Cancel Roll Forward	ROLLFORWARD DATABASE with CANCEL option	Cancels the roll-forward recovery operation. The database is put in restore pending status.
Complete Roll Forward	ROLLFORWARD DATABASE with COMPLETE option	For databases that were archived and restored, but did not have the logs rolled forward, rolls forward the logs. The logs can be rolled forward to a point in time or to the end of the log.
Start	ACTIVATE DATABASE	Activates the specified database and starts all necessary database services so that the database is available for connection and use by any application.
Stop	DEACTIVATE DATABASE	Deactivates the specified database.
Unquiesce	UNQUIESCE	Restores user access databases that were quiesced for maintenance or other reasons.
Table spaces		
Backup	BACKUP	Creates a backup copy of a table space.
Restore	RESTORE	Re-creates a damaged or corrupted table space that was backed up with the DB2 backup utility. The task assistant does not support restoring multiple table spaces.
Rollforward	ROLLFORWARD DATABASE	Recovers a table space by applying transactions that were recorded in the log files.
Cancel Roll Forward	ROLLFORWARD DATABASE with CANCEL option	Cancels the roll-forward recovery operation. One or more table spaces are put in restore pending status.
Complete Roll Forward	ROLLFORWARD DATABASE with COMPLETE option	For table spaces that were archived and restored, but did not have the logs rolled forward, rolls forward the logs. The logs can be rolled forward to a point in time or to the end of the log.
Tables		
Convert Table	ADMIN_MOVE_TABLE	Converts row-organized tables to column organization.
Export Table	EXPORT	Exports data from a table to one of several external file formats.
Import Table	IMPORT	Inserts data from an external file with a supported file format into a table.
Load Table	LOAD	Loads data into a DB2 table.

Table 10. Task assistant support for DB2 for Linux, UNIX, and Windows database administration commands (continued)

Action	Database administration command	Description
Optim High Performance Unload	db2hpu	<p>Uses Optim High Performance Unload commands to unload data from a DB2 table or to copy data from source tables to target tables by using temporary files to store the data. You can also migrate data from one database system to another.</p> <p>To specify Optim High Performance Unload as the unload method, Optim High Performance Unload for Linux, UNIX, and Windows must be installed on each database system that is involved in the unload or migration.</p>
Reorg Table	REORG TABLE	Reorganizes a table.
Reorg Index	REORG INDEX	Reorganizes all of the indexes that are defined for the table.
Run Statistics	RUNSTATS	Updates the statistics about the characteristics of a table, its indexes, or both.
Set Integrity	SET INTEGRITY	Brings tables out of set integrity pending state, places tables in set integrity pending state, places tables into full access state, or prunes the contents of staging tables.
Packages		
Rebind	REBIND PACKAGE	Re-creates a package without needing the original bind file.

Tip: To create a database by using the context-sensitive menu, another database must exist. To create the first database in an instance, you can use the **New Database** icon in the toolbar at the top of the Administration Explorer.

Managing jobs in IBM Data Studio

IBM Data Studio web console provides job creation, job scheduling, and job management for your DB2 for Linux, UNIX, and Windows and DB2 for z/OS databases.

With the Data Studio web console job manager you can:

- Create and schedule jobs directly from the IBM Data Studio client workbench.
 - Use the workbench script editor to create your script and then schedule the script to run as a job in the job manager.
 - Access the Data Studio web console either embedded in the workbench or in a stand-alone web browser window.
 - Access the job history for a database directly from the Administration Explorer in the workbench.
- Create and manage jobs by using the web console graphical user interface.
 - View jobs, schedules, and notifications filtered by criteria such as database, job ID, or job type.
- Create jobs based on database scripts:

SQL-only scripts

The SQL-only scripts are run by the job manager by running the SQL commands that are outlined in the script part of the job directly against the database.

DB2 CLP scripts

The DB2 CLP script jobs are run on the database server by the job manager, which logs in to the database server by using SSH. For multiple databases, the job manager logs in as the user ID that is defined in the database connection. For a single database, based on the user's selection, the job manager logs in by using SSH credentials that the user supplies or the user ID that is defined in the database connection. When logged in, the job manager runs command line processor commands directly on the DB2 console of the server.

Important: To be able to run DB2 CLP script jobs on a database, the user ID that is used to run the job must have permission to log in to the database server by using SSH.

Executable/shell Scripts

The Executable/Shell script jobs are run on the database server by the job manager, which logs in to the database server by using SSH. For multiple databases, the job manager logs in as the user ID that is defined in the database connection. For a single database, based on the user's selection, the job manager logs in by using SSH credentials that the user supplies or the user ID that is defined in the database connection. When logged in, the job manager runs shell commands directly on the server.

Important: To be able to run Executable/Shell script jobs on a database, the user ID that is used to run the job must have permission to log in to the database server by using SSH.

- Schedule jobs to run at a specific time, or to repeat at certain intervals for one or more databases.
- Run jobs for multiple databases as the default user stored in the database connection, or specify a user ID to run the job as when running a job on one database.
- Add jobs together in chains, where the main job is followed by a secondary job dependent on the outcome of the main job, and where a finishing job, such as **RUNSTATS** and **BACKUP**, is run last.
- Configure email notifications to be sent to one or more users depending on the success or failure of the job.
- View the history of all jobs that run on your databases.
 - The job history view gives you a high-level overview of the job results and the option to drill down into each job.
 - You can configure the job manager to retain job history for all jobs that were run, or for a subset depending on the success or failure of the job.
- Manage user access to job manager tasks across your databases.
 - Enable or disable job management privileges requirements for the users of the web console.
 - For each database, grant or revoke job management privileges for each user of the web console.

Creating and managing jobs

With Data Studio web console job manager, you can create and manage your database jobs from the web console.

You create and manage your jobs by using the following tabs of the Job Manager page:

Job List

From this tab, you can create jobs for your databases or run existing jobs directly against a database without scheduling.

When you create a job or open an existing job, the job details open in the job editor. Use the tabs in the job editor to move between jobs, or use the job section view selector to drill down into the script, schedule, notification, and chain component of each job.

Tip: If you have configured your IBM Data Studio client to connect to IBM Data Studio web console you can create jobs directly from the SQL script editor.

Schedules

From this tab, you can create and manage schedules for the jobs that you created for your databases.

A schedule defines when a job will be run, whether the job is repeating, and whether the schedule is limited in number of runs or in time. The schedule also defines one or more databases on which to run the job.

Notifications

Use this tab to manage email notifications for the execution of the jobs that you created for your databases.

Job manager notifications help you monitor the execution results for your jobs across multiple databases and schedules without requiring access to the web console.

Each job can have any number of notifications configured, and each notification can be set up with different conditions, a different set of users to notify, and different collections of databases to monitor.

History

On this tab, you can view the status of jobs that ran on your databases. The job history is displayed for jobs that ran according to a schedule in addition to jobs that you ran manually over the last few days.

Tip: If you have configured your IBM Data Studio client to connect to IBM Data Studio web console you can view job history for a database directly from the Administration Explorer.

Scenario: Creating and scheduling a job

In this scenario, Alan, a database administrator with the Sample Company, uses the job manager to create and schedule a job based on a script provided by Doug, a developer, on the Sales database owned by Becky, a database administrator.

To complete the parts of the scenario, Alan uses the following web console pages of Data Studio web console:

- Databases
- Job Manager

- Job List tab
- Schedules tab
- Notifications tab
- History tab
- Console Security

Alan is a database manager for Sample Company, and is responsible for scheduling database jobs. Alan works with the database script developers for the script content of the jobs and with the database owners to get the required credentials to access the databases. Alan owns the repository database that is used by Data Studio web console to manage user access to the web console.

Alan is approached by Doug, a script developer who asks Alan to schedule a script to be run on the Sales database monthly, and to notify Doug and Doug's manager if the job fails. In addition, each time the script runs, an existing Cleanup job must be run directly afterward.

First Alan verifies with Doug that the script has been tested and verified by development, and that it runs without problems on their test databases. Doug uses other IBM Data Studio tools to verify the scripts.

Next, Alan opens the Databases page in the web console to verify that the Sales database exists as a database connection. If needed, he adds a database connection to the Sales database with information from Becky, the owner of the Sales database. Becky wants to restrict the running of jobs on the **Sales** database to a specific subset of users, so Alan configures the database connection to connect with a user ID that has the minimum required authority of **CONNECT**. To schedule the job on the Sales database Alan also needs the user credentials of a user ID that has the authorizations on the database required by the actions that the script runs. That user ID also needs the required authority to run the cleanup job afterward.

Alan then opens the Job Manager page in the web console, and clicks **Add Job** in the Job List tab to create the job. After filling out the basic information, such as a job name and a description of the job, Alan selects the correct type of job to match the script and verifies that the job is enabled for scheduling.

Working through the new job wizard, Alan pastes in the script that Doug provided into the Script component of the job, making sure that the closing character defined for the job matches what is in the script.

Alan then creates a schedule from the Schedules component of the job, setting a date and time for the first job run, and configuring it to run monthly on the **Sales** database. As the user ID used in the database connection does not have the correct authority to run some of the commands in the script, Alan selects to run the job as the specific user ID with the correct authority that was provided by the database owner.

Alan also adds the requested cleanup job to the job in the Chain component. As the only required chained job is the cleanup, Alan adds it to run at the end of the chain.

Finally, Alan adds the email addresses of Doug and Doug's manager to the Notifications component of the job, and configures notifications to be sent if the job fails.

The job is now scheduled, and Alan can view the job, schedule, and notification information for the job in the corresponding job manager tabs. Once the job has been run, any user with access to the web console can use the History page to view the job history for the job, and get a detailed view by looking at the log entry for the job. If Doug does not have access to the web console, Alan adds Doug as a repository database user and uses the Console Security page to grant Doug access the web console.

Importing tasks from DB2 Task Center

Use the Data Studio web console to import existing tasks from the Task Center in the DB2 Control Center. Imported tasks are saved as jobs in the job manager.

About this task

The imported tasks are mapped to the appropriate job manager type as shown in the following table:

Table 11. Mapping of Task Center script type to Job Manager job type

Task Center script type	Job Manager job type
DB2 command script	DB2 CLP script
OS command script	Shell/Executable script

Restrictions: The following restrictions apply to importing tasks from the DB2 Task Center:

- Task types from DB2 Task Center:

Table 12. Restrictions for task types from DB2 Task Center

Task type	Restrictions
MVS shell script	Not supported.
Grouping	Not supported.
OS command script	The script interpreters and command execution parameters are not supported. The default script interpreter is used instead.
DB2 command script	Supported.

- Schedules that are associated with tasks from DB2 Task Center:

Table 13. Restrictions for schedules from DB2 Task Center

Schedule type	Restrictions
Weekly	Only schedules set for 1 to 4 weeks are supported.
Monthly	Only schedules set for 1 month and schedules set to a specific date or last date are supported.
Yearly	Only schedules set for 1 year are supported.
Expired (that is, schedules with a starting or ending time that is earlier than the current time)	Expired schedules will be imported but marked as inactive.

- Task actions that are associated with tasks from DB2 Task Center:

Table 14. Restrictions for task actions from DB2 Task Center

Task action	Restrictions
Run task	Only the first Run task task action associated with the task will be imported.
Enable schedule of	Not supported.
Disable schedule of	Not supported.
Delete this task	Not supported.

- The success code sets that are used by the DB2 Task Center when running tasks are ignored by the job manager.
- If the tools catalog database contains a task that was previously imported to the Data Studio web console and you choose to import the task again, the task is saved as a new job with a new job ID.
- Contact lists are not imported from the DB2 Task Center.

Procedure

To import tasks from the DB2 Task Center:

1. Open the Data Studio web console in a web browser.
2. To open the Import Tasks page, from the **Open** menu, click **Product Setup > Import Tasks**.
3. Follow the instructions on the Import Tasks page to start importing tasks. You must specify a valid tools catalog database that contains the DB2 Task Center metadata, and then select the tasks to import. Only supported tasks from the tools catalog database are enabled in the Import Tasks page.

Results

If the task is imported successfully, a new job is created for the imported task in the job manager with a job name that is identical to the task name of the imported task. The job name is prefixed by "TC_toolsdb_" where *toolsdb* is the name of the DB2 tools database. The script of the imported task is not modified.

If the imported task is associated with a schedule in the Task Center, a new schedule is created for the corresponding job by the job manager and the tools catalog database is associated with the schedule by default. The schedule date format for the imported task is converted to the job manager schedule format.

What to do next

If the job that was generated from the imported task is not associated with a schedule, create a schedule and add the job to the schedule.

Diagramming access plans with Visual Explain

You can generate a diagram of the current access plan for an SQL or XPATH statement to find out how your data server processes the statement. You can use the information available from the graph to tune your SQL statements for better performance.

Before you begin

If you want to create access plan diagrams for DB2 for z/OS, you must configure the DB2 subsystem that you are using. The steps are identical to the steps for configuring a subsystem for use with the no-charge tuning features that are in IBM Data Studio.

Restriction: For IBM Informix® Dynamic Server, Visual Explain cannot explain SELECT statements that contain parameter markers or host variables.

About this task

You can use Visual Explain to:

- View the statistics that were used at the time of optimization. You can then compare these statistics to the current catalog statistics to help you determine whether rebinding the package might improve performance.
- Determine whether or not an index was used to access a table. If an index was not used, Visual Explain can help you determine which columns might benefit from being indexed.
- Obtain information about each operation in the access plan, including the total estimated cost and number of rows retrieved (cardinality).

Procedure

To generate the diagram of the current access plan for a query:

1. Optional: Set preferences for how Visual Explain operates and for how it displays diagrams.
2. Follow one of these steps:
 - In the Data Project Explorer, right-click an SQL statement, SQL stored procedure, or SQL user-defined function, and select **Open Visual Explain**.
 - In the Data Source Explorer, right-click a view or right-click an SQL stored procedure or SQL user-defined function that contains an INSERT, UPDATE, DELETE, or SELECT statement. Select **Open Visual Explain**. If the workbench finds more than one SQL statement or XQUERY statement, the workbench uses the first statement.
 - In an SQL, Routine, or Java editor, highlight and right-click the INSERT, UPDATE, DELETE, or SELECT statement, XPATH, or XQUERY statement and select **Open Visual Explain**.

Attempts to open Visual Explain from an SQL statement in a Java editor fail if the SQL statement contains variables that are declared in your application. For example, this SQL statement cannot be analyzed by Visual Explain because of the two variables in the predicate:

```
select count(*), sum(order.price)
from order
where order.date > var_date_1
and order.date < var_date_2
```

However, after you bind or deploy the application, you can use InfoSphere Optim Query Tuner or the single-query tuning features in Data Studio to capture the SQL statement from a DB2 package or from the dynamic statement cache and then tune it.

Note: Visual Explain is disabled or throws an exception if the selected SQL statement or object is not explainable. Only the SQL statements in the following list can be explained by Visual Explain:

- For DB2 for Linux, UNIX, and Windows: CALL, Compound SQL (Dynamic), DELETE, INSERT, MERGE, REFRESH, SELECT, SELECT INTO, SET INTEGRITY, UPDATE, VALUES, or VALUES INTO.
 - For DB2 for z/OS: SELECT, INSERT, or the searched form of an UPDATE or DELETE statement.
3. On the first page of the wizard, specify the terminator of the SQL, XPATH, or XQUERY statement that you want to diagram the access plan for.
 4. Optional: On the first page of the wizard, you can also specify settings for various options.
 - a. Specify whether you want to store the collected explain data in explain tables. If you choose this option, Visual Explain does not have to collect explain data the next time that you want to diagram the access plan for the same statement.
 - b. Specify the directory that you want Visual Explain to use as a working directory.
 - c. If IBM Support needs a trace, specify whether to trace the creation of the diagram of the access plan and whether to trace the collection of the explain data.
 - d. Specify whether to save your settings as the defaults for all diagrams that you create with Visual Explain. You can change these defaults with the Preferences window.
 5. On the second page of the wizard, set values for the special registers to customize the runtime environment to influence the collection of explain data. When Visual Explain runs the statement to gather explain data, it uses the values that you specify.

Attention: Please be aware of the following information regarding DB2 data servers.

 - **For DB2 for z/OS:** If you specify different values for CURRENT SCHEMA and CURRENT SQLID, Visual Explain searches for explain tables that are qualified by the value of CURRENT SQLID. If Visual Explain does not find explain tables that are qualified by the value of CURRENT SQLID, Visual Explain attempts to create the explain tables under that value.
 - **For DB2 for Linux, UNIX, and Windows:** If you change the value of CURRENT SCHEMA to a value that contains special characters, you must delimit the value with single quotation marks.
 - **For DB2 for Linux, UNIX, and Windows:** Select the **Collect column and column group statistics** check box if you want Visual Explain to collect detailed statistics about clustered columns and columns that participate in a GROUP BY clause.
 6. Optional: On the second page of the wizard, specify whether to save your settings as the defaults for all diagrams that you create with Visual Explain. You can change these defaults with the Preferences window.
 7. Click **Finish** to close the wizard and to generate the diagram.

Results

The workbench displays the diagram in the Access Plan Diagram view. In this view, you can navigate through the diagram, view descriptions of the nodes in the diagram, and search for nodes.

Diagrams of access plans

When DB2 processes a query, the DB2 optimizer generates several alternative plans for accessing the requested data. The optimizer estimates the execution cost of each plan and chooses the lowest-cost plan to execute. This plan is called the access plan.

Visual Explain graphically displays the access plan for any explainable statement. This display is called an access plan diagram, and it illustrates how DB2 accesses the data for a specified SQL statement.

The access plan diagram consists of nodes and lines that connect those nodes. The nodes represent data sources, operators, SQL statements, and query blocks. Nodes can have only one parent node, but they can have unlimited child nodes. The arrows on the edges indicate the direction of the flow. Usually, a table node is at the bottom of the graph, and the access plan proceeds upward from there.

Some operations in the access plan, such as nested loop joins or index scans, are represented in the graph by groups of nodes, which are called constructs. Many of these constructs have a defining node that indicates the operation. For example, the HBJOIN node indicates that a hybrid join operation is taking place, but the entire hybrid join is represented in the graph by a group of nodes. This group of nodes represents all of the other data sources and operations that are involved in the hybrid join.

Query blocks

An SQL statement can consist of several subqueries, which are represented in the access plan diagram by query blocks.

The subquery can be a SELECT, INSERT, UPDATE, or DELETE. A subquery can contain other subqueries in the FROM clause, the WHERE clause, or a subselect of a UNION or UNION ALL. A subquery within another subquery is called a child subquery. A subquery that contains another subquery is called a parent subquery. This parent-child relationship can be represented by a tree hierarchy.

If a subquery references at least one column of its parent subquery or of any parent subqueries that are higher up in the tree hierarchy, the subquery is a correlated subquery; otherwise it is a non-correlated subquery. A non-correlated subquery can run at the same time as the highest parent subquery that is also non-correlated. This highest parent subquery is called the "do-at-open parent subquery" in terms of its relationship to the non-correlated subquery. The execution of a correlated subquery is bound to the execution of its parent subquery. Such relationships between the relative executions of parents and children can be represented by separate trees hierarchies in the access plan graph.

Non-correlated subquery

For a non-correlated subquery, the query block node is connected to the right of the query block node for the highest parent subquery that is also non-correlated.

Correlated subquery

For a correlated subquery, the query block node is connected to the part within its parent subquery where the correlated subquery is executed.

Setting preferences for Visual Explain

Use the Preferences window to set default values for settings that determine how Visual Explain operates and how it displays diagrams.

Procedure

To set preferences for Visual Explain:

1. Select **Window > Preferences**.
2. In the tree view of the Preferences window, select **Data > Visual Explain**.
3. On the Visual Explain page, set the following options:
 - a. Specify whether to launch the Visual Explain wizard when you right-click an SQL statement, view, stored procedure, or user-defined function and select **Visual Explain**. The wizard allows you to override preferences. If you clear this option, Visual Explain uses the preferences.
 - b. If your project is associated with a DB2 data server, specify whether Visual Explain saves in the explain tables the explain data that it collects for the statement.
4. On the **Query Explain Settings** page, specify default values for special registers. Changing these values modifies how Visual Explain gathers explain data to use when generating the access plan diagram.

Attention: Please be aware of the following information regarding DB2 data servers.

- **For DB2 for z/OS:** If you specify different values for CURRENT SCHEMA and CURRENT SQLID, Visual Explain searches for explain tables that are qualified by the value of CURRENT SQLID. If Visual Explain does not find explain tables that are qualified by the value of CURRENT SQLID, Visual Explain attempts to create the explain tables under that value.
 - **For DB2 for Linux, UNIX, and Windows:** If you change the value of CURRENT SCHEMA to a value that contains special characters, you must delimit the value with single quotation marks.
 - **For DB2 for Linux, UNIX, and Windows:** Select the **Collect column and column group statistics** check box if you want Visual Explain to collect detailed statistics about clustered columns and columns that participate in a GROUP BY clause.
5. On the **Viewer** page, change various behaviors and colors of diagrams.
 6. On the **Nodes** page, change the default appearance of nodes. You can change the text, color, and shape of the different types of nodes. You can also choose whether to highlight selected nodes, shadow nodes, or show information about nodes when you move your mouse cursor over them.

Part 2. Physical design

Physical database design consists of defining database objects and implementing business rules. DB2 10.5 provides support for column compression, expression based indexes, information constraints, sql compatibility enhancements, and capabilities for Oracle migration and compatibility.

You can create the following database objects in a DB2 database:

- Tables
- Constraints
- Indexes
- Triggers
- Sequences
- Views
- Usage lists

You can use Data Definition Language (DDL) statements or tools such as IBM Data Studio to create these database objects. The DDL statements are generally prefixed by the keywords CREATE or ALTER.

Understanding the features and functionality that each of these database objects provides is important to implement a good database design that meets your current business's data storage needs while remaining flexible enough to accommodate expansion and growth over time.

Chapter 6. What is new in DB2 Version 10.5

IBM DB2 Version 10.5 for Linux, UNIX, and Windows offers accelerated analytic processing by introducing a new processing paradigm and data format within the DB2 database product. Advantages include significant reductions in time-to-value and increased consumability, which can be achieved through minimal DBA design requirements and reduced query tuning and debugging efforts. Industry-leading compression, large performance gains for analytic queries, and large reductions in performance variation round out the benefits of deploying this technology.

DB2 column-organized tables

DB2 Version 10.5 introduces compressed column-organized tables for DB2 databases. The release also includes broad support for data mart (analytic) workloads with complex queries that are commonly characterized by multi-table joins, grouping and aggregation, and table scans over a star schema.

Column-organized tables are tables whose data pages contain column data instead of row data. This new capability is in-memory optimized, CPU optimized, and I/O optimized.

The DB2 product continues to provide industry-leading performance across multiple workloads by combining unique ideas with several of the best technological developments of the past 10 years of database research. BLU Acceleration, a combination of innovations from IBM Research and the development labs that simplifies and speeds up reporting and analytics, is a new, fully integrated capability in DB2 Version 10.5. Easy to set up and self-optimizing, BLU Acceleration can typically eliminate the need for indexes, aggregates, or time-consuming database tuning to achieve top performance and storage efficiency. In most cases, no SQL or schema changes are required to take advantage of this breakthrough technology.

DB2 Version 10.5 provides the following enhancements:

- The **ANALYTICS** option of the **DB2_WORKLOAD** registry variable, which provides a single setting to enable column organization, space reclamation, and automatic workload management, and to automatically configure memory, page size, and extent size
- New syntax for the **CREATE TABLE** statement to specify table storage organization
- A new database configuration parameter (**dft_table_org**) to change the default table organization
- The new **db2convert** utility to convert row-organized tables into column-organized tables
- The ability to use the **REORG TABLE** command to reclaim storage in column-organized tables
- Automated workload management, which can significantly improve the performance of workloads with several queries running at the same time
- Dynamic list prefetching, a new prefetching type that is used in query execution plans that access column-organized tables
- Support for **NOT ENFORCED** primary key and unique constraints, which you can use to reduce performance costs and space requirements when it is known

that the data already conforms to the constraint, and for ENFORCED primary key and unique constraints on column-organized tables

The following additional enhancements pertain to column-organized tables specifically:

- *Semi-join* support for queries to reduce the amount of memory that is consumed by large inner tables; in some cases, queries that use semi-joins might run faster. A semi-join is an optimization technique that uses one or more predicates on the outer table and other inner tables in the query. The combined filtering effect of all eligible predicates is pushed down to one or more large inner tables to reduce memory consumption. DB2 explain output is enhanced to indicate whether the inner table of a particular hash join operator is a large inner table for which the semi-join optimization technique is applied.
- Late decompression, the ability to operate directly on compressed data for certain operations, thereby reducing memory usage.
- Multiplied CPU power that uses single instruction, multiple data (SIMD) processing for many operations.
- A vector processing engine for processing vectors of column data instead of individual values.
- Improved system scaling across cores.
- An enhanced caching strategy for buffer pools to substantially reduce I/O.
- A smaller footprint for database storage than in DB2 Version 10.1.
- In-memory optimization for maximal performance.
- A system that is unconstrained by main memory size, in which the amount of space used by the data that is being processed can be considerably larger than the RAM.

For the most up-to-date installation requirements for BLU Acceleration, see DB2 Advanced Enterprise Server Edition 10.5 - *Detailed system requirements for the BLU deployable component* .

IBM DB2 pureScale Feature enhancements

The DB2 pureScale Feature was first introduced in Version 9.8. V10.5 builds on DB2 pureScale Feature support.

The DB2 pureScale Feature provides extreme capacity and application transparency which exceed even the strictest industry standard. Continued improvements in continuous availability and high availability, the DB2 pureScale Feature tolerates both planned maintenance and component failure with ease. The DB2 Version 10.5 release provides:

- Support for high availability disaster recovery (HADR)
- Increased availability
- Improved workload balancing
- Restore mobility between DB2 pureScale Feature and DB2 Enterprise Server Edition

In addition, V10.5 introduces DB2 pureScale Feature enhancements in high availability, performance, workload management, and installation.

Performance improvements

Continuing to build on prior release performance improvements, V10.5 performance improvements focus on explain information for column-organized tables and extension to the CREATE INDEX statement to create an index containing an expression-based key.

SQL compatibility

If you work with relational database products other than DB2 products, V10.5 builds on existing functionality, interfaces, and compatibility features to provide additional enhancements that make DB2 products more familiar to you. These enhancements reduce the time and complexity of enabling applications that are written for other relational database products to run quickly in a DB2 environment.

Chapter 7. Column-organized tables

DB2 column-organized tables add columnar capabilities to DB2 databases, which include data that is stored with column organization and vector processing of column data. Using this table format with star schema data marts provides significant improvements to storage, query performance, and ease of use through simplified design and tuning.

If most tables in your database are going to be column-organized tables, set the **DB2_WORKLOAD** registry variable to **ANALYTICS** *before* you create the database. Doing so helps to configure memory, table organization, page size, and extent size, and enables workload management. For more information about this setting, see “System environment variables”.

If the workload is entirely an analytics or OLAP workload, the recommended approach is to put as many tables as possible into column-organized format. These workloads are characterized by nonselective data access (that is, queries access more than approximately 5% of the data), and extensive scanning, grouping, and aggregation.

The process of inserting data into column-organized tables, or updating previously inserted data, is optimized for large insert transactions. As a result, workloads that are transactional in nature should not use column-organized tables. Traditional row-organized tables with index access are generally better suited for these types of workloads. With column-organized tables, performance is optimal if 100 or more rows are impacted by each insert or update transaction.

For mixed workloads, which include a combination of analytic query processing and very selective access (involving less than 2% of the data), a mix of row-organized and column-organized tables might be suitable.

The process of loading data has some differences for column-organized tables. For more information, see “Loading data into column-organized tables” on page 62.

You can establish a configuration that is optimal for analytic workloads and set the default table organization to **COLUMN** automatically by setting the **DB2_WORKLOAD** registry variable to **ANALYTICS**.

If you cannot create your database and have it auto-configured by setting the **DB2_WORKLOAD** registry variable to **ANALYTICS**, you must create your database and optimally configure it for analytic workloads. For more information, see Chapter 1, “Creating and setting up your database configuration for analytic workloads,” on page 3.

Support for column-organized tables is available in specific system and database configurations. For more information, see “Supported system and database configurations for column-organized tables” on page 58.

Parallel processing must be enabled to access and query column-organized tables. For more information, see “Enabling parallel processing for column-organized tables” on page 4.

To create a column-organized table, you can specify the **ORGANIZE BY COLUMN** clause on the **CREATE TABLE** statement. Alternatively, if you want to create tables with a specific table organization without having to specify the **ORGANIZE BY COLUMN** or the **ORGANIZE BY ROW** clause, you can change the default table organization by setting the **dft_table_org** database configuration parameter.

Synopsis tables

A synopsis table is a column-organized table that is automatically created and maintained by the system to store metadata for an associated user-defined column-organized table.

The synopsis table contains all the user table's non-character columns (that is, datetime, Boolean, or numeric columns) and those character columns that are part of a primary key or foreign key definition. As of DB2 Cancun Release 10.5.0.4, the synopsis table for a new column-organized table also includes **CHAR**, **VARCHAR**, **GRAPHIC**, and **VARGRAPHIC** columns.

The synopsis table stores the minimum and maximum values for each column across a range of rows and uses those values to skip over data that is of no interest to a query during evaluation of certain type of predicates (**=**, **>**, **>=**, **<**, **<=**, **BETWEEN**, **NOT BETWEEN**, **IS NOT NULL**, **IN**, and **NOT IN**).

The only supported operation against a synopsis table is a select operation. To determine the name of the synopsis table for a particular user table, query the **SYSCAT.TABLES** catalog view, as shown in the following example:

```
SELECT tabschema, tabname, tableorg
  FROM syscat.tables
 WHERE tableorg = 'C';
```

TABSCHEMA	TABNAME	TABLEORG
DB2INST1	SALES_COL	C
SYSIBM	SYN130330165216275152_SALES_COL	C

The synopsis table has schema **SYSIBM**.

The relationship between a user table and its synopsis table is recorded in the **SYSCAT.TABDEP** catalog view. You can query this catalog view for details, as shown in the following example:

```
SELECT tabname, bname
  FROM syscat.tabdep
 WHERE dtype = '7';
```

Supported system and database configurations for column-organized tables

Before you start using column-organized tables, ensure that you have a system and database configuration that is supported.

The following operating systems support column-organized:

- Linux (x86-x64, Intel and AMD processors)
- AIX® (POWER® processors)

The following system and database configurations do not support column-organized tables in the DB2 Version 10.5 release.

- Databases in a DB2 pureScale environment
- Partitioned databases
- Databases that are created without automatic storage enabled
- Table spaces that are not enabled for reclaimable storage
- Databases whose code set and collation are not UNICODE and IDENTITY or IDENTITY_16BIT

The following additional restrictions apply to column-organized tables in the DB2 Version 10.5 release:

- Schemas that include column-organized tables cannot be transported.
- Data federation with column-organized tables is not supported.
- For data replication with column-organized tables as either source or target:
 - A CREATE TABLE statement with the DATA CAPTURE CHANGES clause is not supported.
- Automatic tuning of sort memory is not supported in environments using column-organized tables.
- For label-based access control (LBAC), the following syntax is not supported: ALTER TABLE...SECURITY POLICY *policyname*
- For row and column access control (RCAC), the following syntax is not supported:
 - ALTER TABLE...ACTIVATE ROW | COLUMN ACCESS CONTROL
 - CREATE PERMISSION / CREATE MASK
- The following SQL statements do not support column-organized tables as target or source tables:
 - CREATE EVENT MONITOR
 - CREATE GLOBAL TEMPORARY TABLE
 - CREATE INDEX and ALTER INDEX
 - CREATE MASK
 - CREATE NICKNAME
 - CREATE PERMISSION
 - CREATE TRIGGER
 - When processing a trigger, INSERT, UPDATE or MERGE to a column-organized table
 - DECLARE GLOBAL TEMPORARY TABLE
 - SET INTEGRITY
- Queries using the RR or RS isolation level are not supported with column-organized tables.
- XA transactions are not supported.

Creating column-organized tables

Create column-organized tables to store the data from a single column together on a set of data pages instead of storing the data for complete rows together on a set of data pages.

Before you begin

1. Set the **SORTHEAP** database configuration parameter and the **SHEAPTHRES_SHR** database configuration parameter to a value other than AUTOMATIC.

2. Set the **DB2_WORKLOAD** registry variable to **ANALYTICS** before you create the database. This setting establishes an optimal default configuration when you use the database for analytic workloads. For more information, see “System environment variables” in the *Database Administration Concepts and Configuration Reference*.
3. The database must be single partition and use the **UNICODE** code set and **IDENTITY** collation.
4. To address the resource needs of the **LOAD** command, set the **util_heap_sz** (utility heap size) database configuration parameter to at least 1,000,000 pages and **AUTOMATIC**.

About this task

By creating column-organized tables in star schema data marts, you can benefit from significant improvements to storage, query performance, and ease of use through simplified design and tuning.

Restrictions

- Review any applicable restrictions in “CREATE TABLE statement” in the *SQL Reference Volume 2*.
- Review the page size-specific limits for column-organized tables in “Database manager page size-specific limits” in the *SQL Reference Volume 1*.
- Constraints
 - **ENFORCED** check and foreign key (referential integrity) constraints are not supported on column-organized tables. These constraints are supported as informational (**NOT ENFORCED**) constraints.
 - You cannot specify the **WITH CHECK OPTION** when creating a view that is based on column-organized tables.

Creating column-organized tables might result in the automatic creation of tables that store metadata.

Procedure

To create a column-organized table:

1. Issue the **CREATE TABLE** statement, specifying the **ORGANIZE BY COLUMN** clause and creating a primary key on the column-organized table.
2. Populate the table by issuing the **LOAD** command.

Important: When populating a column-organized table with data for the first time, it is strongly recommended that the majority of the data be added by using a single large load operation, because the column compression dictionaries are created on the basis of the first table load operation.

Example

The following example shows you how to create a column-organized table called **JTNISBET.STAFF**.

1. Issue the **CREATE TABLE** statement as follows:

```
CREATE TABLE JTNISBET.STAFF (  
  ID SMALLINT NOT NULL,  
  NAME VARCHAR(9),  
  DEPT SMALLINT,  
  JOB CHAR(5),
```

```

YEARS SMALLINT,
SALARY DECIMAL(7,2),
COMM DECIMAL(7,2) )
ORGANIZE BY COLUMN;

```

2. Populate the JTNISBET.STAFF table with data by using the **LOAD** command, as in the following example:

```

LOAD FROM /TEST/DATA.DEL OF DEL MODIFIED BY COLDEL,
REPLACE INTO JTNISBET.STAFF;

```

The following query returns a list of all column-organized tables in the database:

```

SELECT
  SUBSTR(TABNAME, 1, 24) AS TABNAME,
  SUBSTR(TABSCHEMA, 1, 24) AS TABSCHEMA
FROM SYSCAT.TABLES
WHERE TABLEORG = 'C';

```

INSERT, UPDATE, and DELETE (including MERGE) statement restrictions for column-organized tables

Some options on the INSERT, UPDATE, and DELETE statements are not supported for column-organized tables in DB2Version 10.5.

The following items are not supported. The restrictions on column-organized tables also apply to views on one or more column-organized tables:

- SELECT FROM INSERT / UPDATE / DELETE, when the target table of the INSERT, UPDATE, or DELETE statement is a column-organized table. As a result, there is no need to support the *include-columns* clause of the INSERT, UPDATE, and DELETE statements. That clause is supported only in cases where a select operation is run from the INSERT, UPDATE, or DELETE statement. Because the *include-columns* clause is not supported, the *assignment-clause* clause in a DELETE statement is not supported.
- Triggers, when the subject table or the target of an INSERT, UPDATE, or DELETE statement that is in the body of the trigger is a column-organized table.
- Positioned UPDATE or DELETE. That is, an UPDATE or DELETE WHERE CURRENT OF *cursor-name*, where the target table is a column-organized table. You cannot declare a cursor that can be updated or deleted when the outer fullselect includes a reference to a column-organized table.
- SELECT...FOR UPDATE when the target table is a column-organized table.
- Prior to DB2 Cancun Release 10.5.0.4, the MERGE statement when the target table is a column-organized table. As of DB2 Cancun Release 10.5.0.4, this restriction is lifted.
- INSERT, UPDATE, or DELETE statement inside an atomic block when the target table is a column-organized table.
- INSERT, UPDATE, or DELETE statement on a UNION ALL view when the UNION ALL view includes a column-organized table.
- Extended indicator variable values of DEFAULT or UNASSIGNED in statements that are modifying column-organized tables.
- Repeatable read (RR) and read stability (RS) isolation levels when the table that is scanned is a column-organized table. As a result, the WITH *isolation-level* clause of the INSERT, UPDATE, or DELETE statement accepts only the uncommitted read (UR) or cursor stability (CS) isolation level if the underlying table that is scanned is a column-organized table. Setting the **cur_commit** database configuration parameter to DISABLED for CS scans on column-organized tables is not supported. Additionally, if the **ConcurrentAccessResolution**

CLI/ODBC keyword specifies `CurrentlyCommitted` for a CS scan on a column-organized table, an error is returned. As of DB2 Cancun Release 10.5.0.4, this restriction also applies to a MERGE statement that includes the `WITH isolation-level` clause.

Loading data into column-organized tables

Although loading data into column-organized tables is very similar to loading data into row-organized tables, you should be aware of the few exceptions and configure your database to handle the additional resources that are required.

Before you begin

- You must have an established connection to the database or be able to implicitly connect to the database into which you want to load the data.

About this task

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

In addition, statistics are collected during the load according to the profile defined by the RUNSTATS command.

Restrictions

- Check the restrictions of the **LOAD** command as they also apply to row-organized tables.

Procedure

To load data in column-organized tables:

1. Set the **blocknonlogged** (block creation of tables that allow non-logged activity) database configuration parameter to NO before you load data into a column-organized table. If this parameter is not set, the error message SQL2032N is returned.
2. Ensure that the **util_heap_sz** (utility heap size) database configuration parameter is set to at least 1,000,000 pages and AUTOMATIC to address the resource needs of the LOAD command.

If the database server has at least 128 GB of memory, set **util_heap_sz** 4,000,000 pages. If concurrent load utilities are running, increase the **util_heap_sz** value to accommodate higher memory requirements. If memory is scarce, the **util_heap_sz** value can be increased dynamically only when a load operation is running.

If you set **DB2_WORKLOAD** to ANALYTICS before the creation of your database, the **util_heap_sz** value is automatically configured during database creation.

3. Ensure that the path for load temporary files has sufficient storage space (equivalent to the raw size of the input data) to build the column compression dictionary.

If a column compression dictionary must be built, the input data source (such as a pipe or a socket) is processed twice. If the input source can be reopened, it is read twice. If the input source cannot be reopened, its contents are temporarily cached in the load temporary file directory.

The default path for load temporary files is located under the instance directory, or in a location that is specified by the `TEMPFILES PATH` option on the **LOAD** command.

4. Ensure that you have enough memory.

Memory requirements temporarily increase when the column compression dictionary is being created.

For optimal load performance, additional cache memory is required to write column-organized data in extent-sized amounts, rather than one page at a time, therefore reducing I/O costs.

Scenario: Achieving high speed analytics over growing volumes of column-organized data

This scenario describes an active data warehouse that is used for reporting and analysis. The scenario is built by using an IBM Cognos® and DB2 stack on Intel hardware. The three-tier stack includes a web interface front end, Cognos Business Intelligence (BI) 10.2, a BI server with Cognos Dynamic Cubes running on the middle tier, and a relational database tier that uses DB2 Version 10.5.

This stack can be used with wide-ranging database sizes, up to terabytes of data on a single machine. For this scenario, a 1 TB database was used to demonstrate the ease with which you can achieve high speed analytics over a large data volume.

Cognos Business Intelligence 10.2 Dynamic Cubes extends Dynamic Query with in-memory caching of members, data, expressions, results, and aggregates. Use it to define in-memory aggregates that are loaded from the database whenever the dynamic cube is started or refreshed. This performance optimization results in fast response times for predefined reports and some ad hoc queries (whenever it can use an in-memory aggregate). When an in-memory aggregate is not applicable to service a particular report, the database is queried directly. DB2 Version 10.5 provides a next generation database for analytics that is hardware optimized right out of the box.

Although this scenario is focused on a single point in time, the stack is intended to be used with a dynamic warehouse. In a *dynamic warehouse*, data is updated regularly and new data is loaded once or twice a day to refresh the warehouse contents.

The prerequisite for this stack is data that conforms to a star schema. A single star schema is used throughout the examples in this scenario, but multiple star schemas could be included within the single stack. The star schema in this scenario includes a single fact table with approximately 3 billion rows and 23 columns and 10 dimension tables with approximately 20 million rows total and 3 - 29 columns per table. Typical reporting and analytic queries join the fact table to one or more dimension tables and aggregate values for one or more columns in the fact table. In a standard database environment with row-organized tables, performance is impacted when many tables are joined, even when few columns are accessed. The column-organized table functionality in DB2 Version 10.5 provides significant performance and ease-of-use benefits for this type of workload.

The first step is to build the database. Table 15 on page 64 shows that there are key differences in the database setup and the initial load operation between DB2 Version 10.5 and DB2 10.1 that simplify things considerably.

Table 15. Comparison of the database setup steps when you use DB2 10.1 and DB2 Version 10.5

DB2 10.1	DB2 Version 10.5
CREATE DATABASE command	There is no difference.
CREATE BUFFERPOOL statement	There is no difference.
CREATE TABLESPACE statement	There is no difference.
CREATE TABLE statement	Use the ORGANIZE BY COLUMN clause; otherwise, the syntax is unchanged.
LOAD command for each table	There is no difference.
CREATE INDEX statement	This statement is not required.
Define constraints	There is no difference.
Define or refresh MQTs	This step is not required, because performance is already optimized.
RUNSTATS command on all tables and MQTs	This step is not required, because table RUNSTATS command operations are performed as part of the data load process and MQTs are not created.
Create statistical views and issue the RUNSTATS command against those views	This step is not required, because performance is already optimized.

In DB2 Version 10.5, database creation and the initial load operation are simplified. The time-consuming process of determining optimal indexes, MQTs, and statistical views to benefit your workload is not required. Moreover, the data is available much sooner. In this scenario, the time that was required to load the database was reduced by over 40%, and storage savings increased significantly. With adaptive compression in DB2 10.1, the tables and associated indexes for the star schema required approximately 370 GB of space. With DB2 Version 10.5, the tables required only 180 GB of space. Any MQTs that were defined in DB2 10.1 would require even more space.

More importantly, the out-of-the-box performance improvements are also significant. With no changes to the database and database manager configuration that was used with DB2 10.1, single-user queries in this scenario experienced up to an 18-fold performance improvement with DB2 Version 10.5. Cube load times in DB2 Version 10.5 were 70% lower than in DB2 10.1, not including the MQT creation and **RUNSTATS** command steps.

After the stack is up and running, there are more operational activities that benefit from improvements in DB2 Version 10.5. The only required manual operations for refreshing data in the warehouse are loading data and restarting the dynamic cube. No additional reorganization operations or **RUNSTATS** command operations are required, because they run automatically. You do not have to refresh any MQTs, and there are no indexes to maintain when you load data into the tables, because no MQTs or indexes are required. As soon as a load operation is complete, data access is optimized for performance, and you can restart the dynamic cube to refresh the in-memory caches.

Using the ORGANIZE BY COLUMN clause of the CREATE TABLE statement when you define fact and dimension tables simplifies the building and maintenance of the database and optimizes performance for both predefined Cognos reports and ad hoc database queries.

Chapter 8. Expression-based indexes

With expression-based indexes, you can create an index that includes expressions. The performance of queries that involves expressions is improved if the database manager chooses an index that is created on the same expressions.

Expression-based indexes are best suited when you want an efficient evaluation of queries that involve a column expression. Simple index keys consist of a concatenation of one or more specified table columns. Compared to simple indexes, the index key values of expression-based indexes are not the same as values in the table columns. The values are transformed by the expressions that you specify.

You can create the index with the CREATE INDEX statement. If an index is created with the UNIQUE option, the uniqueness is enforced against the values that are stored in the index. The uniqueness is not enforced against the original column values.

Statistics for expression-based indexes

In an expression-based index, the key includes one or more expressions that consist of more than just a column name. An expression-based index can provide faster and more efficient access to data. However, before you implement this feature, consider how statistics are collected for expression-based indexes.

To benefit from expression-based indexes, you do not have to carry out any special activities. Statistics for expression-based key columns are collected along with other statistics on the table and its indexes. Typically, the same options that you apply to a table's columns are also suitable for the expression-based key columns. However, if you want to have more control, review the following topics:

Expression-based indexes and automatic statistics collection

With automatic statistics collection, you can allow the database manager to determine whether to update the statistics for a table, including a table with expression-based indexes. The **auto_runstats** database configuration parameter and real-time statistics (rts) feature are enabled by default when you create a database.

The **auto_runstats** database configuration parameter and the rts feature apply the statistics profiles in the same manner as a manually issued **RUNSTATS** command. They apply the statistics profile of the statistical view if the following conditions are met:

- A statistical view with a statistics profile.
- A base table with a statistic profile.
- A base table that includes an expression-based index.

Neither the **auto_runstats** database configuration parameter nor the rts feature act directly on the automatically generated statistical view. The statistical view statistics are gathered when the **auto_runstats** database configuration parameter and the rts feature act on the table itself. The statistical view statistics are gathered regardless of whether statistics profile exist on the table or on the associated statistical views.

The rts feature does not fabricate statistical view statistics for the expression-based index.

Expression-based indexes and manual statistics updates

You can manually update statistics for expression-based key columns by issuing the UPDATE statement against the SYSSTAT.TABLES, SYSSTAT.COLUMNS, and SYSSTAT.COLDIST catalog views.

When you issue the UPDATE statement against the SYSSTAT views, specify the statistical view's name and the name of the column as it appears in the statistical view.

Automatic statistics collection does not occur if you manually update a table's statistics by issuing an UPDATE statement against SYSSTAT catalog views. This condition includes manual updates of expression-based key column statistics for any expression-based indexes that the table has, even though those statistics are in a separate statistical view. Similarly, if you manually update any statistics in the statistical view that is associated with a table with an expression-based index, the underlying table is excluded from automatic statistics collection. To get automatic statistics collection to consider the table, issue the **RUNSTATS** command against the table rather than the statistical view.

RUNSTATS on expression-based indexes

Issuing the **RUNSTATS** command on a table with expression-based indexes updates and stores the index statistics as usual. However, statistics for each expression-based key column within the index are also collected. The optimizer uses these statistics to choose query execution plans for queries that involve the expressions.

The statistics for the expression-based keys are collected and stored in a statistical view. This statistical view is automatically created when you create an expression-based index. Issuing the **RUNSTATS** command on a table and its expression-based index resembles issuing the **RUNSTATS** command on the table and then issuing it again on the statistical view. However, although you can issue the **RUNSTATS** command directly on the statistical view, you should issue the command on the table and its indexes instead. This results in the gathering and updating of the statistics that are stored in the statistical view and the statistics for the index and table. If you issue the **RUNSTATS** command on the statistical view, the statistical view statistics are updated, but the index and table statistics are untouched.

To issue the **RUNSTATS** command on an index with expression-based keys, explicitly include the index by name, or implicitly include the index by using the **AND INDEXES ALL** clause or the **FOR INDEXES ALL** clause. If you do not explicitly or implicitly include the index, the index and the statistical view statistics are not updated.

You cannot name expressions as columns for the **RUNSTATS** command. However, expressions are considered key columns. Therefore, to gather distribution statistics for expression-based columns, include the index in the **RUNSTATS** command, or use a statistics profile. In addition, specify that distribution statistics are to be gathered on all columns or on key columns. Statistical view statistics are updated for non-expression columns only if they are included in the **RUNSTATS** column specification. Statistical view statistics for the non-expression columns can be disregarded, as they are not referenced by the optimizer.

Expression-based key columns exist only in the index. They do not exist in the base table. Therefore, the **INDEXSAMPLE** clause, rather than the **TABLESAMPLE** clause, determines how the expression column data is sampled.

Example

The following conditions apply to all the examples:

- Table TBL1 is created with expression-based index IND1.
- Associated statistical view IND1_V is automatically created.

Example 1

The **RUNSTATS** command is issued on table TBL1 in two ways:

- No index is specified:
`runstats on table TBL1`
- An index that is not IND1 is specified:
`runstats on table TBL1 with distribution on key columns and index IND2`

The results in both cases are the same: the index statistics for IND1 are not updated, and the statistical view statistics for IND1_V are not updated, even though the **ON KEY COLUMNS** parameter was specified in the second case. (Specifying the **ALL COLUMNS** parameter would not change the results, either.) To gather statistics on expression-based key columns in an expression-based index, you must explicitly or implicitly include that index in the **RUNSTATS** command.

Example 2

The **RUNSTATS** command is issued on table TBL1 in three ways:

- Index IND1 is specified:
`runstats on table TBL1 and index IND1`
`runstats on table TBL1 on columns (c1,c3) and indexes IND1,IND2`
- The **ALL** parameter is specified:
`runstats on table TBL1 on key columns and indexes all`

In all of these cases, the index statistics for IND1 are updated. As well, the statistical view statistics for IND1_V are updated with basic column statistics for all expression columns. These results apply even though the **ON COLUMNS AND** clause was specified.

Expression-based indexes and statistics profiles

The **RUNSTATS** command's statistics profile facility can automatically gather customized statistics for a particular table, including a table with an expression-based index. This facility simplifies statistics collection by storing the **RUNSTATS** command's options for convenient future use.

The **RUNSTATS** command provides the option to register and use a statistics profile, which specifies the type of statistics that are to be collected for a particular table. To register a profile and collect statistics at the same time, issue the **RUNSTATS** command with the **SET PROFILE** parameter. To register a profile only, issue the **RUNSTATS** command with the **SET PROFILE ONLY** parameter. To collect statistics with a profile that was already registered, issue the **RUNSTATS** command with the **USE PROFILE** parameter.

A statistics profile is useful if you want to gather nonstandard statistics for particular columns. However, an expression cannot be named as a column for a **RUNSTATS** command. Therefore, to gather nonstandard statistics, you would issue the **RUNSTATS** command on the base table and the statistical view. This strategy raises the problem of keeping the table and statistical view statistics in sync. Instead, issuing the **RUNSTATS** command uses a table's statistics profile and, if that profile includes an index with expression-based keys in its specifications, then any statistics profile on the statistical view that is associated with that index is also applied. So, all statistics are gathered with a just one **RUNSTATS** operation on the table because all relevant profiles are applied.

If the table does not have a statistics profile that is associated with it (or you do not specify the **USE PROFILE** parameter), then no profile on the associated statistical views is applied while the **RUNSTATS** facility is running. This behavior applies only to the statistical views associated with indexes that includes expression-based keys and not statistical views in general.

Creating a statistics profile on a statistical view that is associated with an index, which includes expression-based keys, automatically creates a statistics profile on the base table (but only if one does not exist).

If a statistics profile is associated with the base table and a statistics profile is associated with the statistical view, the profiles are used as follows while the **RUNSTATS** command is running on a table with an expression-based index:

- The table's profile controls the statistics for the non-expression columns and the **RUNSTATS** command overall.
- The statistical view's statistics profile controls the statistics for the expression columns.

The **auto_runstats** database configuration parameter and real-time statistics (rts) feature apply the statistics profiles in the same manner as a manually issued **RUNSTATS** command. If the statistics profile for a statistical view exists and the underlying table has a statistical profile that includes an expression-based index, the **auto_runstats** database configuration parameter and the rts feature apply the statistics profile for the statistical view. Neither the **auto_runstats** database configuration parameter nor the rts feature act directly on the expression-based index's statistical view. The statistical view statistics are gathered when the **auto_runstats** database configuration parameter and the rts feature act on the table itself. In addition, the rts feature does not fabricate statistical view statistics for the expression-based index.

Example

The following initial conditions apply to the examples:

- Table TBL1 has no statistics profile.
- Table TBL1 has an expression-based index IND1.
- Index IND1 has an automatically generated statistical view IND1_V.
- The **NUM_FREQVALUES** database configuration parameter is set to the default value of 10.

Example 1

The following command sets a statistics profile on the statistical view IND1_V, which gathers extra distribution statistics:

```
RUNSTATS ON VIEW IND1_V WITH DISTRIBUTION DEFAULT NUM_FREQVALUES 40 SET PROFILE ONLY
```

Because there is no statistics profile on the table, a statistics profile is generated when the command is issued in the following form:

```
RUNSTATS ON TABLE TBL1 WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL
```

Now, the following command is issued:

```
RUNSTATS ON TABLE TBL1 USE PROFILE
```

The statistics profile on the table and the statistics profile on the statistical view are applied. Statistics are gathered for the table and the expression columns in the index. The columns in the table have the usual amount of frequent value statistics and the columns in the statistical view have more frequent value statistics.

Example 2

The following command sets a statistics profile that samples a large table with the aim to shorten the execution time for the **RUNSTATS** command and to reduce the impact on the overall system (the profile is set, but the command to sample the table is not issued):

```
RUNSTATS ON TABLE TBL1 WITH DISTRIBUTION AND INDEXES ALL TABLESAMPLE SYSTEM (2.5) INDEXSAMPLE SYSTEM (10) SET PROFILE ONLY UTIL_IMPACT_PRIORITY 30
```

It is decided that distribution statistics are not needed on the expression-based keys in index IND1. However, LIKE statistics are required on the second key column in the index. According to the definition for statistical view IND1_V in the catalogs, the second column in the view is named K01.

The following command is issued to modify the statistics profile to gather LIKE statistics on column K01 from statistical view IND1_V (the profile is set, but the command to gather statistics is not issued):

```
RUNSTATS ON VIEW IND1_V ON ALL COLUMNS AND COLUMNS(K01 LIKE STATISTICS) SET PROFILE ONLY
```

Now the statistics profile is set and the following command is issued to gather statistics that are based on the statistics profile:

```
RUNSTATS ON TABLE TBL1 USE PROFILE
```

The statistics profile on the table and the statistics profile on the statistical view are applied. Statistics are gathered for the table and the expression-based columns. The results are as follows:

- Distribution statistics are gathered for the base table columns but not for the expression-based key columns.
- The LIKE statistics are gathered for the specified expression-based key column.
- While the **RUNSTATS** command is running, the expression-based key column values are sampled at the rate dictated by the **INDEXSAMPLE SYSTEM(10)** parameter in the table's profile.
- The table's **UTIL_IMPACT_PRIORITY** parameter setting governs the priority of the entire **RUNSTATS** command operation.

Chapter 9. Extended row size

Starting with DB2 Version 10.5, you can create a table whose row length can exceed the maximum record length for the page size of the table space.

Rows of table data are organized into blocks called pages, which can be four sizes: 4, 8, 16, or 32 KB. All tables that you create within a table space of a particular size have a matching page size. In previous releases, the maximum number of bytes in a table row was depended on the page size of the table space. Any attempt to create a table whose row length exceeded the maximum record length for the page size resulted in an error (SQLSTATE 54010). For example, in previous releases, you could not create the table in the following statement in a table space with a 4 KB page size because of the table's large row size:

```
CREATE TABLE T1 (C1 INTEGER, C2 VARCHAR(5000))
```

However, starting with DB2 Version 10.5, you can create table T1 in a table space with a 4 KB page size.

You can use extended row size support for the following purposes:

- To migrate tables that have row sizes exceeding 32 KB to DB2 Version 10.5.
- To help improve the performance of applications where the majority of data rows can fit on a smaller page but the table definition requires a bigger page size.
- To create tables with more VARCHAR or VARGRAPHIC columns. The maximum number of columns is not changing, but the ability to exceed the maximum record length for the page size allows for more columns.

You can alter tables to take advantage of extended row size support.

Implementing extended row size

Implement extended row size to create tables whose row length can exceed the maximum record length for the page size of the table space.

Rows of table data are organized into blocks that are called pages which can be four sizes: 4, 8, 16, and 32 kilobytes. All tables that are created within a table space of a particular size have a matching page size.

Table 16. Limits for number of columns and row size in each table space page size

Page size	Row size limit	Column count limit
4K	4 005	500
8K	8 101	1 012
16K	16 293	1 012
32K	32 677	1 012

Without extended row size support, the maximum number of bytes allowed in a table row is dependant on the page size of the table space. Any attempt to create a table whose row length exceeds the maximum record length for the page size results in an error (SQLSTATE 54010). For example, the following table could not be created in a 4K page size table space because of its row size.

```
CREATE TABLE T1 (C1 INTEGER, C2 VARCHAR(5000))
```

The row size for this table is 5010 bytes; calculated as 5 bytes (C1 plus nullable column overhead) + 5005 bytes (C2 plus varying length column overhead)

With extended row size support, tables that contain large rows that exceed the maximum record length for the page size of the table space can be created. With extended row size support, the table T1 can be created in a 4K page size table space.

Extended row size support can be used to:

- Migrate tables that have row sizes that exceed 32K to DB2 Version 10.5.
- Improve the performance of applications where most of data rows can fit on a smaller page, but the table definition requires a bigger page size.
- Create tables with more VARCHAR or VARGRAPHIC columns. The maximum number of columns does not change, but the ability to exceed the maximum record length for the page size allows for more columns.

Existing tables can be altered to take advantage of extended row size support. For example, more columns can be added or the length of character and graphic string columns can be increased. Without extended row size support, these changes would exceed the row size for the page size of the table space.

Any row-organized tables created by a user support extended row size except for range clustered tables (RCT).

Working with tables with extended row size

Enabling extended row size support for a table requires that:

1. The **extended_row_sz** database configuration parameter must be set to ENABLE.
2. The table definition must contain at least one varying length string column (VARCHAR or VARGRAPHIC).
3. The row size of the table cannot exceed 1048319 bytes (SQLSTATE 54010).
4. Queries requiring an explicit or implicit system temporary table with extended rows needs a system temporary table space that can fully contain the minimum width of the row. The minimum width of the row is calculated in the same way as the maximum width except that all VARCHAR and VARGRAPHIC columns are assumed to have a length of 1.

Tables with extended row size support can be identified by checking the EXTENDED_ROW_SIZE column of the SYSCAT.TABLES catalog view.

Inserting data into a table with extended row size support

When a data row is inserted or updated in a table with extended row size support and the physical data row length exceeds the maximum record length for the table space, a subset of the varying length string columns (VARCHAR or VARGRAPHIC) is stored as large object (LOB) data outside of the data row. The table column in the base row is replaced by a descriptor that is 24 bytes in size. For columns of VARCHAR(n) where n is less than or equal to 24, or VARGRAPHIC(n) where n is less than or equal to 12, data remains in the base data row.

If some VARCHAR or VARGRAPHIC data is stored out of row in a LOB data object, there is no change to the data type. The VARCHAR or VARGRAPHIC column does not become a LOB column. Operations like

IMPORT, EXPORT, and LOAD do not require any LOB modifiers to work with the VARCHAR or VARGRAPHIC data.

Examples

In the following examples the **extended_row_sz** database configuration parameter is set to ENABLE.

Creating a table without specifying a table space

The following CREATE TABLE statement is issued:

```
CREATE TABLE T1 (C1 INT, C2 VARCHAR(4000));
```

The row size for T1 is 4010 bytes. If a table space with a page size of at least 8K cannot be found, then T1 is created in a 4K page size table space and if required some varying length data might be stored out of row. If there is no 4K page size table space that can be used, an error is returned (SQLSTATE 42727).

Adding a column

Table T1 was created with the following CREATE TABLE statement.

```
CREATE TABLE T1 (C1 INT, C2 VARCHAR(3995)) in TS1;
```

Table space TS1 has a 4K page size and so table T1 is created without the need to store any data out of row because its byte count is 4005. A new column is added to table T1 with the following ALTER TABLE statement:

```
ALTER TABLE T1 ADD C3 CLOB(1M);
```

The byte count for the table now exceeds the maximum record length. The ALTER TABLE is successful because the **extended_row_sz** database configuration parameter is set to ENABLE.

Deactivating VALUE COMPRESSION

Table T1 was created with the following CREATE TABLE statement.

```
CREATE TABLE T1 (C1 INT, C2 VARCHAR(1993), C3 VARCHAR(2000))  
IN TS1 VALUE COMPRESSION;
```

Table space TS1 has a 4K page size and table T1 has a byte count of 4005 and so no data is stored out of row. Compression on table T1 is deactivated with the following ALTER TABLE statement:

```
ALTER TABLE T1 DEACTIVATE VALUE COMPRESSION;
```

This changes the byte count of table T1 to 4008. The ALTER TABLE is successful because the **extended_row_sz** database configuration parameter is set to ENABLE. If required some varying length data might be stored out of row.

If VALUE COMPRESSION is reactivated, any subsequent inserts of VARCHAR data are stored in the base row. All rows that were inserted when VALUE COMPRESSION was deactivated remain out of row until they are updated or table T1 is reorganized.

Creating an index

Table T1 was created with the following CREATE TABLE statement.

```
CREATE TABLE T1
(C1 INT, C2 VARCHAR(1000),
 C3 VARCHAR(1000),
 C4 VARCHAR(1000),
 C5 VARCHAR(1000))
IN TS1;
```

Table space TS1 has a 4K page size and table T1 is successfully created because the **extended_row_sz** database configuration parameter is set to ENABLE. No special handling is required to create indexes on table T1.

```
CREATE INDEX I1 on T1 (C2);
CREATE INDEX I2 on T1 (C3);
CREATE INDEX I3 on T1 (C4);
CREATE INDEX I4 on T1 (C5);
```

Chapter 10. Informational constraints

An *informational constraint* is a constraint attribute that can be used by the SQL compiler to improve the access to data. Informational constraints are not enforced by the database manager, and are not used for additional verification of data; rather, they are used to improve query performance.

You define informational constraints by using the CREATE TABLE or ALTER TABLE statement. You first add constraints and then associate them with constraint attributes, specifying whether the database manager is to enforce the constraints. For primary key constraints, unique constraints, and check constraints, you can further specify that the constraint can be trusted. For referential integrity constraints, if the constraint is not enforced, you can further specify whether the constraint can be trusted. A not-enforced and not-trusted constraint is also known as a statistical referential integrity constraint. You can specify whether a referential integrity constraint or check constraint is to be used for query optimization.

Informational RI (referential integrity) constraints are used to optimize query performance, the incremental processing of REFRESH IMMEDIATE MQT, and staging tables. Query results, MQT data, and staging tables might be incorrect if informational constraints are violated.

For example, the order in which parent-child tables are maintained is important. When you want to add rows to a parent-child table, you must insert rows into the parent table first. To remove rows from a parent-child table, you must delete rows from the child table first. This ensures that there are no orphan rows in the child table at any time. Otherwise the informational constraint violation might affect the correctness of queries being executed during table maintenance, as well as the correctness of the incremental maintenance of dependent MQT data and staging tables.

You can create a not-enforced primary key or unique constraint for either a column-organized or row-organized table. Unlike an enforced primary key or unique constraint, a not-enforced primary key or unique constraint does not create an index on the data. Specify an informational constraint only if the table data is independently known to conform to the constraint. Because the DB2 database manager does not enforce uniqueness for these constraints, if the table data violates the not-enforced constraint, incorrect results can occur. You cannot reference not-enforced primary key constraints in any enforced referential integrity definitions.

Designing informational constraints

Constraints that are enforced by the database manager when records are inserted or updated can lead to high amounts of system activity, especially when loading large quantities of records that have referential integrity constraints. If an application has already verified information before inserting a record into the table, it might be more efficient to use informational constraints, rather than normal constraints.

Informational constraints tell the database manager what rules the data conforms to, but the rules are not enforced by the database manager. However, this information can be used by the DB2 optimizer and could result in better performance of SQL queries.

The following example illustrates the use of information constraints and how they work. This simple table contains information about applicants' age and gender:

```
CREATE TABLE APPLICANTS
(
  AP_NO INT NOT NULL,
  GENDER CHAR(1) NOT NULL,
  CONSTRAINT GENDEROK
  CHECK (GENDER IN ('M', 'F'))
  NOT ENFORCED
  ENABLE QUERY OPTIMIZATION,
  AGE INT NOT NULL,
  CONSTRAINT AGEOK
  CHECK (AGE BETWEEN 1 AND 80)
  NOT ENFORCED
  ENABLE QUERY OPTIMIZATION,
);
```

This example contains two options that change the behavior of the column constraints. The first option is **NOT ENFORCED**, which instructs the database manager not to enforce the checking of this column when data is inserted or updated. This option can be further specified to be either **TRUSTED** or **NOT TRUSTED**. If the informational constraint is specified to be **TRUSTED** then the database manager can trust that the data will conform to the constraint. This is the default option. If **NOT TRUSTED** is specified then the database manager knows that most of the data, but not all, will not conform to the constraint. In this example, the option is **NOT ENFORCED TRUSTED** by default since the option of trusted or not trusted was not specified.

The second option is **ENABLE QUERY OPTIMIZATION** which is used by the database manager when **SELECT** statements are run against this table. When this value is specified, the database manager will use the information in the constraint when optimizing the SQL.

If the table contains the **NOT ENFORCED** option, the behavior of insert statements might appear odd. The following SQL will not result in any errors when run against the **APPLICANTS** table:

```
INSERT INTO APPLICANTS VALUES
(1, 'M', 54),
(2, 'F', 38),
(3, 'M', 21),
(4, 'F', 89),
(5, 'C', 10),
(6, 'S', 100),
```

Applicant number five has a gender (C), for child, and applicant number six has both an unusual gender and exceeds the age limits of the **AGE** column. In both cases the database manager will allow the insert to occur since the constraints are **NOT ENFORCED** and **TRUSTED**. The result of a select statement against the table is shown in the following example:

```
SELECT * FROM APPLICANTS
WHERE GENDER = 'C';
```

```
APPLICANT  GENDER  AGE
-----  -
```

0 record(s) selected.

The database manager returned the incorrect answer to the query, even though the value 'C' is found within the table, but the constraint on this column tells the database manager that the only valid values are either 'M' or 'F'. The `ENABLE QUERY OPTIMIZATION` keyword also allowed the database manager to use this constraint information when optimizing the statement. If this is not the behavior that you want, then the constraint needs to be changed through the use of the `ALTER TABLE` statement, as shown in the following example:

```
ALTER TABLE APPLICANTS
ALTER CHECK AGEOK DISABLE QUERY OPTIMIZATION
```

If the query is reissued, the database manager will return the following correct results:

```
SELECT * FROM APPLICANTS
WHERE SEC = 'C';
```

```
APPLICANT  GENDER  AGE
-----  -
          5  C      10
```

1 record(s) selected.

Note: If the constraint attributes `NOT ENFORCED` `NOT TRUSTED` and `ENABLE QUERY OPTIMIZATION` were specified from the beginning for the table `APPLICANTS`, then the correct results shown previously would have been returned after the first `SELECT` statement was issued.

The best scenario for using `NOT ENFORCED TRUSTED` informational constraints occurs when you can guarantee that the application program is the only application inserting and updating the data. If the application already checks all of the information beforehand (such as gender and age in the previous example) then using informational constraints can result in faster performance and no duplication of effort. Another possible use of informational constraints is in the design of data warehouses. Also, if you cannot guarantee that the data in the table will always conform to the constraint you can set the constraints to be `NOT ENFORCED` and `NOT TRUSTED`. This type of constraint can be used when strict matching between the values in the foreign keys and the primary keys are not needed. This constraint can also still be used as part of a statistical view enabling the optimization of certain SQL queries.

Creating and modifying constraints

Constraints can be added to existing tables with the `ALTER TABLE` statement.

About this task

The constraint name cannot be the same as any other constraint specified within an `ALTER TABLE` statement, and must be unique within the table (this includes the names of any referential integrity constraints that are defined). Existing data is checked against the new condition before the statement succeeds.

Creating and modifying unique constraints

Unique constraints can be added to an existing table. The constraint name cannot be the same as any other constraint specified within the `ALTER`

TABLE statement, and must be unique within the table (this includes the names of any referential integrity constraints that are defined). Existing data is checked against the new condition before the statement succeeds.

To define unique constraints using the command line, use the ADD CONSTRAINT option of the ALTER TABLE statement. For example, the following statement adds a unique constraint to the EMPLOYEE table that represents a new way to uniquely identify employees in the table:

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT NEWID UNIQUE(EMPNO, HIREDATE)
```

To modify this constraint, you would have to drop it, and then re-create it.

Creating and modifying primary key constraints

A primary key constraint can be added to an existing table. The constraint name must be unique within the table (this includes the names of any referential integrity constraints that are defined). Existing data is checked against the new condition before the statement succeeds.

To add primary keys using the command line, enter:

```
ALTER TABLE <name>
  ADD CONSTRAINT <column_name>
  PRIMARY KEY <column_name>
```

An existing constraint cannot be modified. To define another column, or set of columns, as the primary key, the existing primary key definition must first be dropped, and then re-created.

Creating and modifying check constraints

When a table check constraint is added, packages and cached dynamic SQL that insert or update the table might be marked as invalid.

To add a table check constraint using the command line, enter:

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

To modify this constraint, you would have to drop it, and then re-create it.

Creating and modifying foreign key (referential) constraints

A foreign key is a reference to the data values in another table. There are different types of foreign key constraints.

When a foreign key is added to a table, packages and cached dynamic SQL containing the following statements might be marked as invalid:

- Statements that insert or update the table containing the foreign key
- Statements that update or delete the parent table.

To add foreign keys using the command line, enter:

```
ALTER TABLE <name>
  ADD CONSTRAINT <column_name>
  FOREIGN KEY <column_name>
  ON DELETE <action_type>
  ON UPDATE <action_type>
```

The following examples show the ALTER TABLE statement to add primary keys and foreign keys to a table:

```
ALTER TABLE PROJECT
  ADD CONSTRAINT PROJECT_KEY
  PRIMARY KEY (PROJNO)
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
```

```

PRIMARY KEY (EMPNO, PROJNO, ACTNO)
ADD CONSTRAINT ACT_EMP_REF
FOREIGN KEY (EMPNO)
REFERENCES EMPLOYEE
ON DELETE RESTRICT
ADD CONSTRAINT ACT_PROJ_REF
FOREIGN KEY (PROJNO)
REFERENCES PROJECT
ON DELETE CASCADE

```

To modify this constraint, you would have to drop it and then re-create it.

Creating and modifying informational constraints

To improve the performance of queries, you can add informational constraints to your tables. You add informational constraints using the CREATE TABLE or ALTER TABLE statement when you specify the NOT ENFORCED option on the DDL. Along with the NOT ENFORCED option you can further specify the constraint to be either TRUSTED or NOT TRUSTED.

Restriction: After you define informational constraints on a table, you can only alter the column names for that table after you remove the informational constraints.

To specify informational constraints on a table using the command line, enter one of the following commands for a new table:

```

ALTER TABLE <name> <constraint attributes> NOT ENFORCED
ALTER TABLE <name> <constraint attributes> NOT ENFORCED TRUSTED
ALTER TABLE <name> <constraint attributes> NOT ENFORCED NOT TRUSTED

```

ENFORCED or NOT ENFORCED: Specifies whether the constraint is enforced by the database manager during normal operations such as insert, update, or delete.

- ENFORCED cannot be specified for a functional dependency (SQLSTATE 42621).
- NOT ENFORCED should only be specified if the table data is independently known to conform to the constraint. Query results might be unpredictable if the data does not actually conform to the constraint. You can also specify if the NOT ENFORCED constraint is to be TRUSTED or NOT TRUSTED.
 - TRUSTED: Informs the database manager that the data can be trusted to conform to the constraint. This is the default option. This option must only be used if the data is independently known to conform to the constraint
 - NOT TRUSTED: Informs the database manager that the data cannot be trusted to conform to the constraint. This option is intended for cases where the data conforms to the constraint for most rows, but it is not independently known to conform to the constraint. NOT TRUSTED can be specified only for referential integrity constraints (SQLSTATE 42613).

To modify this constraint, you would have to drop it and then re-create it.

Chapter 11. Data compression

You can reduce storage needed for your data by using the compression capabilities built into DB2 for Linux, UNIX, and Windows to reduce the size of tables, indexes and even your backup images.

Tables and indexes often contain repeated information. This repetition can range from individual or combined column values, to common prefixes for column values, or to repeating patterns in XML data. There are a number of compression capabilities that you can use to reduce the amount of space required to store your tables and indexes, along with features you can employ to determine the savings compression can offer.

You can also use backup compression to reduce the size of your backups.¹

Compression capabilities included with most editions of DB2 for Linux, UNIX, and Windows include:

- Value compression
- Backup compression.

The following additional compression capabilities are available with the a license for the DB2 Storage Optimization Feature:

- Row compression, including compression for XML storage objects.
- Temporary table compression
- Index compression.

For more details about index compression, see “Index compression” on page 95.

For more details about backup compression, see “Backup compression” on page 98.

Table compression

You can use less disk space for your tables by taking advantage of the DB2 table compression capabilities. Compression saves disk storage space by using fewer database pages to store data.

Also, because you can store more rows per page, fewer pages must be read to access the same amount of data. Therefore, queries on a compressed table need fewer I/O operations to access the same amount of data. Since there are more rows of data on a buffer pool page, the likelihood that needed rows are in the buffer pool increases. For this reason, compression can improve performance through improved buffer pool hit ratios. In a similar way, compression can also speed up backup and restore operations, as fewer pages of need to be transferred to the backup or restore the same amount of data.

You can use compression with both new and existing tables. Temporary tables are also compressed automatically, if the database manager deems it to be advantageous to do so.

1. See “Backup compression” in *Data Recovery and High Availability Guide and Reference* for more information.

There are two main types of data compression available for tables:

- Row compression (available with a license for the DB2 Storage Optimization Feature).
- Value compression

For a particular table, you can use row compression and value compression together or individually. However, you can use only one type of row compression for a particular table.

Value compression

Value compression optimizes space usage for the representation of data, and the storage structures used internally by the database management system to store data. Value compression involves removing duplicate entries for a value, and only storing one copy. The stored copy keeps track of the location of any references to the stored value.

When creating a table, you can use the optional `VALUE COMPRESSION` clause of the `CREATE TABLE` statement to specify that the table is to use value compression. You can enable value compression in an existing table with the `ACTIVATE VALUE COMPRESSION` clause of the `ALTER TABLE` statement. To disable value compression in a table, you use the `DEACTIVATE VALUE COMPRESSION` clause of the `ALTER TABLE` statement.

When `VALUE COMPRESSION` is used, `NULLs` and zero-length data that has been assigned to defined variable-length data types (`VARCHAR`, `VARGRAPHICS`, `LONG VARCHAR`, `LONG VARGRAPHIC`, `BLOB`, `CLOB`, and `DBCLOB`) will not be stored on disk.

If `VALUE COMPRESSION` is used then the optional `COMPRESS SYSTEM DEFAULT` option can also be used to further reduce disk space usage. Minimal disk space is used if the inserted or updated value is equal to the system default value for the data type of the column, as the default value will not be stored on disk. Data types that support `COMPRESS SYSTEM DEFAULT` include all numeric type columns, fixed-length character, and fixed-length graphic string data types. This means that zeros and blanks can be compressed.

When using value compression, the byte count of a compressed column in a row might be larger than that of the uncompressed version of the same column. If your row size approaches the maximum allowed for your page size, you must ensure that sum of the byte counts for compressed and uncompressed columns does not exceed allowable row length of the table in the table space. For example, in a table space with 4 KB page size, the allowable row length is 4005 bytes. If the allowable row length is exceeded, the error message `SQL0670N` is returned. The formula used to determine the byte counts for compressed and uncompressed columns is documented as part of the `CREATE TABLE` statement.

If you deactivate value compression:

- `COMPRESS SYSTEM DEFAULTS` will also be deactivated implicitly, if it had previously been enabled
- The uncompressed columns might cause the row size to exceed the maximum allowed by the current page size of the current table space. If this occurs, the error message `SQL0670N` will be returned.
- Existing compressed data will remain compressed until the row is updated or you perform a table reorganization with the `REORG` command.

Row compression

Row compression uses a dictionary-based compression algorithm to replace recurring strings with shorter symbols within data rows.

There are two types of row compression that you can choose from:

- “Classic” row compression.
- Adaptive compression

Row compression is available with a license for the DB2 Storage Optimization Feature. Depending on the DB2 product edition that you have, this feature might be included, or it might be an option that you order separately.

Classic row compression

Classic row compression, sometimes referred to as *static compression*, compresses data rows by replacing patterns of values that repeat across rows with shorter symbol strings.

The benefits of using classic row compression are similar to those of adaptive compression, in that you can store data in less space, which can significantly save storage costs. Unlike adaptive compression, however, classic row compression uses only a table-level dictionary to store globally recurring patterns; it doesn't use the page-level dictionaries that are used to compress data dynamically.

How classic row compression works

Classic row compression uses a table-level compression dictionary to compress data by row. The dictionary is used to map repeated byte patterns from table rows to much smaller symbols; these symbols then replace the longer byte patterns in the table rows. The compression dictionary is stored with the table data rows in the data object portions of the table.

What data gets compressed?

Data that is stored in base table rows and log records is eligible for classic row compression. In addition, the data in XML storage objects is eligible for compression. You can compress LOB data that you place inline in a table row; however, storage objects for long data objects are not compressed.

Restriction: You cannot compress data in XML columns that you created with DB2 Version 9.5 or DB2 Version 9.1. However, you can compress inline XML columns that you add to a table using DB2 Version 9.7 or later, provided the table was created without XML columns in an earlier release of the product. If a table that you created in an earlier release already has one or more XML columns and you want to add a compressed XML column by using DB2 Version 9.7 or later, you must use the `ADMIN_MOVE_TABLE` stored procedure to migrate the table before you can use compression.

Turning classic row compression on or off

To use classic row compression, you must have a license for the DB2 Storage Optimization Feature. You compress table data by setting the `COMPRESS` attribute of the table to `YES STATIC`. You can set this attribute when you create the table by specifying the `COMPRESS YES STATIC` option for the `CREATE TABLE` statement. You can also alter an existing table to use compression by using the same option for the `ALTER TABLE` statement. After you enable compression, operations that

add data to the table, such as an **INSERT**, **LOAD INSERT**, or **IMPORT INSERT** command operation, can use classic row compression. In addition, index compression is enabled for new indexes on the table. Indexes are created as compressed indexes unless you specify otherwise and if they are the types of indexes that can be compressed.

Important: When you enable classic row compression for a table, you enable it for the entire table, even if a table comprises more than one table partition.

To disable compression for a table, use the **ALTER TABLE** statement with the **COMPRESS NO** option; rows that you subsequently add are not compressed. To extract the entire table, you must perform a table reorganization with the **REORG TABLE** command.

If you have a license for the DB2 Storage Optimization Feature, compression for temporary tables is enabled automatically. You cannot enable or disable compression for temporary tables.

Effects of update activity on logs and compressed tables

Depending upon update activity and which columns are updated within a data row, log usage might increase.

If a row increases in size, the new version of the row might not fit on the current data page. Rather, the new image of the row is stored on an overflow page. To minimize the creation of pointer-overflow records, increase the percentage of each page that is to be left as free space after a reorganization by using the **ALTER TABLE** statement with the **PCTFREE** option. For example, if you set the **PCTFREE** option to 5% before you enabled compression, you might change it to 10% when you enable compression. Increasing the percentage of each page to be left as free space is especially important for data that is heavily updated.

Classic row compression for temporary tables

Compression for temporary tables is enabled automatically with the DB2 Storage Optimization Feature. When executing queries, the DB2 optimizer considers the storage savings and the impact on query performance that compression of temporary tables offers to determine whether it is worthwhile to use compression. If it is worthwhile, compression is used automatically. The minimum size that a table must be before compression is used is larger for temporary tables than for regular tables.

You can use the explain facility or the **db2pd** tool to see whether the optimizer used compression for temporary tables.

Adaptive compression

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

How adaptive compression works

Adaptive compression actually uses two compression approaches. The first employs the same table-level compression dictionary used in classic row compression to compress data based on repetition within a sampling of data from the table as a whole. The second approach uses a page-level dictionary-based compression algorithm to compress data based on data repetition within each page of data. The dictionaries map repeated byte patterns to much smaller symbols; these symbols then replace the longer byte patterns in the table. The table-level compression dictionary is stored within the table object for which it is created, and is used to compress data throughout the table. The page-level compression dictionary is stored with the data in the data page, and is used to compress only the data within that page. For more information about the role each of these dictionaries in compressing data, see “Compression dictionaries” on page 91.

Note: You can specify that a table be compressed with classic row compression only by using a table-level compression dictionary. However, you cannot specify that tables be compressed by using only page-level compression dictionaries. Adaptive compression uses both table-level and page-level compression dictionaries.

Data that is eligible for compression

Data that is stored *within* data rows, including inlined LOB or XML values, can be compressed with both adaptive and classic row compression. XML storage objects can be compressed using static compression. However storage objects for long data objects that are stored outside table rows is not compressed. In addition, though log records themselves are not compressed, the amount of log data written as a result of insert, update or delete operations is reduced by virtue of the rows being compressed.

Restriction: You cannot compress data in XML columns that you created with DB2 Version 9.5 or DB2 Version 9.1. However, you can compress inline XML columns that you add to a table using DB2 Version 9.7 or later, provided the table was created without XML columns in an earlier release of the product. If a table that you created in an earlier release already has one or more XML columns and you want to add a compressed XML column by using DB2 Version 9.7 or later, you must use the `ADMIN_MOVE_TABLE` stored procedure to migrate the table before you can use compression.

Turning adaptive compression on or off

To use adaptive compression, you must have a license for the DB2 Storage Optimization Feature. You compress table data by setting the `COMPRESS` attribute of the table to `YES`. You can set this attribute when you create the table by specifying the `COMPRESS YES` option for the `CREATE TABLE` statement. You can also alter an existing table to use compression by using the same option for the `ALTER TABLE` statement. After you enable compression, operations that add data to the table, such as an **INSERT**, **LOAD INSERT**, or **IMPORT INSERT** command operation, can use adaptive compression. In addition, index compression is enabled for new indexes on the table. Indexes are created as compressed indexes unless you specify otherwise and if they are the types of indexes that can be compressed.

Important: When you enable adaptive compression for a table, you enable it for the entire table, even if the table comprises more than one table partition.

To disable compression for a table, use the ALTER TABLE statement with the COMPRESS NO option; rows that you later add are not compressed. Existing rows remain compressed. To extract the entire table after you turn off compression, you must perform a table reorganization with the **REORG TABLE** command.

If you apply the licence for the DB2 Storage Optimization Feature, compression for temporary tables is enabled automatically if the database manager deems it valuable. You cannot enable or disable compression for temporary tables.

Effects of update activity on logs and compressed tables

Depending upon update activity and the position of updates in a data row, log usage might increase.

If a row increases in size after adding new data to it, the new version of the row might not fit on the current data page. Rather, the new image of the row is stored on an overflow page. To minimize the creation of pointer-overflow records, increase the percentage of each page that is to be left as free space after a reorganization by using the ALTER TABLE statement with the PCTFREE option. For example, if you set the PCTFREE option to 5% before you enabled compression, you might change it to 10% when you enable compression. Increasing the percentage of each page to be left as free space is especially important for data that is heavily updated.

Compression for temporary tables

Compression for temporary tables is enabled automatically with the DB2 Storage Optimization Feature. Only classic row compression is used for temporary tables.

System temporary tables

When executing queries, the DB2 optimizer considers the storage savings and the impact on query performance that compression of system-created temporary tables offers to determine whether it is worthwhile to use compression. If it is worthwhile, classic row compression is used automatically. The minimum size that a table must be before compression is used is larger for temporary tables than for regular tables.

User-created temporary tables

Created global temporary tables (CGTTs) and declared global temporary tables (DGTs) are always compressed using classic row compression.

You can use the explain facility or the **db2pd** tool to see whether the optimizer used compression for system temporary tables.

Estimating storage savings offered by adaptive or classic row compression

You can view an estimate of the storage savings adaptive or classic row compression can provide for a table by using the ADMIN_GET_TAB_COMPRESS_INFO table function.

Before you begin

The estimated savings that adaptive or classic row compression offers depend on the statistics generated by running the **RUNSTATS** command. To get the most accurate estimate of the savings that can be achieved, run the **RUNSTATS** command before you perform the following steps.

Procedure

To estimate the storage savings adaptive or classic row compression can offer using the ADMIN_GET_TAB_COMPRESS_INFO table function:

1. Formulate a SELECT statement that uses the ADMIN_GET_TAB_COMPRESS_INFO table function. For example, for a table named SAMPLE1.T1, enter:

```
SELECT * FROM TABLE(SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SAMPLE1', 'T1'))
```

2. Execute the SELECT statement. Executing the statement shown in Step 1 might yield a report like the following:

TABSCHEMA	TABNAME	DBPARTITIONNUM	DATAPARTITIONID	OBJECT_TYPE	ROWCOMPMODE	...
SAMPLE1	T1	0	0	DATA	A	...

1 record(s) selected.

PCTPAGESSAVED_CURRENT	AVGROWSIZE_CURRENT	PCTPAGESSAVED_STATIC	...
96	24	81	...

AVGROWSIZE_STATIC	PCTPAGESSAVED_ADAPTIVE	AVGROWSIZE_ADAPTIVE	...
148	93	44	...

Creating a table that uses compression

When you create a new table by issuing the CREATE TABLE statement, you have the option to compress the data contained in table rows.

Before you begin

You must decide which type of compression you want to use: adaptive compression, classic row compression, value compression, or a combination of value compression with either of the two types of row compression. Adaptive compression and classic row compression almost always save storage because they attempt to replace data patterns that span multiple columns with shorter symbol strings. Value compression can offer savings if you have many rows with columns that contain the same value, such as a city or country name, or if you have columns that contain the default value for the data type of the column.

Procedure

To create a table that uses compression, issue a CREATE TABLE statement.

- If you want to use adaptive compression, include the COMPRESS YES ADAPTIVE clause.
- If you want to use classic row compression, include the COMPRESS YES STATIC clause.
- If you want to use value compression, include the VALUE COMPRESSION clause. If you want to compress data that represents system default column values, also include the COMPRESS SYSTEM DEFAULT clause.

Results

After you create the table, all data that you add to the table from that point in time on is compressed. Any indexes that are associated with the table are also compressed, unless you specify otherwise by using the COMPRESS NO clause of the CREATE INDEX or ALTER INDEX statements.

Examples

Example 1: The following statement creates a table for customer information with adaptive compression enabled. In this example, the table is compressed by using both table-level and page-level compression dictionaries.

```
CREATE TABLE CUSTOMER
(CUSTOMERNUM    INTEGER,
 CUSTOMERNAME   VARCHAR(80),
 ADDRESS        VARCHAR(200),
 CITY           VARCHAR(50),
 COUNTRY        VARCHAR(50),
 CODE           VARCHAR(15),
 CUSTOMERNUMDIM INTEGER)
COMPRESS YES ADAPTIVE;
```

Example 2: The following statement creates a table for customer information with classic row compression enabled. In this example, the table is compressed by using only a table-level compression dictionary.

```
CREATE TABLE CUSTOMER
(CUSTOMERNUM    INTEGER,
 CUSTOMERNAME   VARCHAR(80),
 ADDRESS        VARCHAR(200),
 CITY           VARCHAR(50),
 COUNTRY        VARCHAR(50),
 CODE           VARCHAR(15),
 CUSTOMERNUMDIM INTEGER)
COMPRESS YES STATIC;
```

Example 3: The following statement creates a table for employee salaries. The SALARY column has a default value of 0, and row compression and system default compression are specified for the column.

```
CREATE TABLE EMPLOYEE_SALARY
(DEPTNO  CHAR(3)    NOT NULL,
 DEPTNAME VARCHAR(36) NOT NULL,
 EMPNO   CHAR(6)    NOT NULL,
 SALARY  DECIMAL(9,2) NOT NULL WITH DEFAULT COMPRESS SYSTEM DEFAULT)
COMPRESS YES ADAPTIVE;
```

Note that the VALUE COMPRESSION clause was omitted from this statement. This statement creates a table that is called EMPLOYEE_SALARY; however, a warning message is returned:

```
SQL20140W COMPRESS column attribute ignored because VALUE COMPRESSION is
deactivated for the table.  SQLSTATE=01648
```

In this case, the COMPRESS SYSTEM DEFAULT clause is not applied to the SALARY column.

Example 4: The following statement creates a table for employee salaries. The SALARY column has a default value of 0, and row compression and system default compression are enabled for the column.

```
CREATE TABLE EMPLOYEE_SALARY
(DEPTNO  CHAR(3)    NOT NULL,
 DEPTNAME VARCHAR(36) NOT NULL,
 EMPNO   CHAR(6)    NOT NULL,
 SALARY  DECIMAL(9,2) NOT NULL WITH DEFAULT COMPRESS SYSTEM DEFAULT)
VALUE COMPRESSION COMPRESS YES ADAPTIVE;
```

In this example, the VALUE COMPRESSION clause is included in the statement, which compresses the default value for the SALARY column.

Enabling compression in an existing table

By using the ALTER TABLE statement, you can modify an existing table to take advantage of the storage-saving benefits of compression.

Before you begin

You must decide which type of compression you want to use: adaptive compression, classic row compression, value compression, or a combination of value compression with either of the two types of row compression. Adaptive compression and classic row compression almost always save storage because they attempt to replace data patterns that span multiple columns with shorter symbol strings. Value compression can offer savings if you have many rows with columns that contain the same value, such as a city or country name, or if you have columns that contain the default value for the data type of the column.

Procedure

To enable compression in an existing table:

1. Issue the ALTER TABLE statement.
 - If you want to use adaptive compression, include the COMPRESS YES ADAPTIVE clause.
 - If you want to use classic row compression, include the COMPRESS YES STATIC clause.
 - If you want to use value compression, include the ACTIVATE VALUE COMPRESSION clause for each column that contains a value you want compressed. If you want to compress data in columns that contain system default values, also include the COMPRESS SYSTEM DEFAULT clause.

All rows that you subsequently append, insert, load, or update use the new compressed format.

2. Optional: To immediately apply compression to all the existing rows of a table, perform a table reorganization by using the **REORG TABLE** command. If you do not apply compression to all rows at this point, uncompressed rows will not be stored in the new compressed format until the next time that you update them, or the next time the REORG TABLE command runs.

Examples

Example 1: The following statement applies adaptive compression to an existing table that is named CUSTOMER:

```
ALTER TABLE CUSTOMER COMPRESS YES ADAPTIVE
```

Example 2: The following statement applies classic row compression to an existing table that is named CUSTOMER:

```
ALTER TABLE CUSTOMER COMPRESS YES STATIC
```

Example 3: The following statements apply row, value, and system default compression to the SALARY column of an existing table that is named EMPLOYEE_SALARY. The table is then reorganized.

```
ALTER TABLE EMPLOYEE_SALARY  
ALTER SALARY COMPRESS SYSTEM DEFAULT  
COMPRESS YES ACTIVATE VALUE COMPRESSION;
```

```
REORG TABLE EMPLOYEE_SALARY
```

Changing or disabling compression for a compressed table

You can change how a table is compressed or disable compression entirely for a table that has adaptive, classic row, or value compression enabled by using one or more of the various compression-related clauses of the ALTER TABLE statement.

About this task

If you deactivate adaptive or classic row compression, index compression is not affected. If you want to uncompress an index, you must use the ALTER INDEX statement.

Procedure

To deactivate compression for a table, or to change from one type of row compression to another:

1. Issue an ALTER TABLE statement.
 - If you want to deactivate adaptive or classic row compression, include the COMPRESS NO clause.
 - If you want to change to a different type of row compression, specify the type of compression you want using the COMPRESS YES ADAPTIVE or COMPRESS YES STATIC clauses. For example, if you have a table that currently uses classic row compression, and you want to change to adaptive compression, execute the ALTER TABLE statement with the COMPRESS YES ADAPTIVE clause
 - If you want to deactivate value compression, include the DEACTIVATE VALUE COMPRESSION clause.
 - If you want to deactivate the compression of system default values, include the COMPRESS OFF option for the ALTER *column name* clause.
2. Perform an offline table reorganization using the **REORG TABLE** command.

Results

- If you turned off row compression using the COMPRESS NO clause, all row data is uncompressed.
- If you changed from one type of row compression to another, the entire table is compressed using the type of row compression you specified in the ALTER TABLE statement. (See Example 2.)
- Deactivating value compression has the following effects:
 - If a table had columns with COMPRESS SYSTEM DEFAULT enabled, compression is no longer enabled for these columns.
 - Uncompressed columns might cause the row size to exceed the maximum that the current page size of the current table space allows. If this occurs, error message SQL0670N is returned.

Examples

Example 1: Turning off row compression: The following statements turn off adaptive or classic row compression in a table named CUSTOMER and then reorganizes the table to uncompress that data that was previously compressed:

```
ALTER TABLE CUSTOMER COMPRESS NO
REORG TABLE CUSTOMER
```

Example 2: Changing from static to adaptive compression: Assumes that the SALES table currently uses classic row compression. The following statements change the type of compression used to adaptive compression:

```
ALTER TABLE SALES COMPRESS ADAPTIVE YES  
REORG TABLE SALES
```

Compression dictionaries

The database manager creates a table-level compression dictionary for each table that you enable for either adaptive or classic row compression. For tables that you enable for adaptive compression, the database manager also creates page-level compression dictionaries. Column compression dictionaries are created for column-organized tables.

Compression dictionaries are used to map repeated byte patterns to much smaller symbols, which then replace the longer byte patterns in the table.

Table-level compression dictionaries

To build table-level dictionaries, entire rows in the table are scanned for repeated patterns. The database manager then builds a compression dictionary, assigning short, numeric keys to those repeated patterns. In general, text strings provide greater opportunities for compression than numeric data; compressing numeric data involves replacing one number with another. Depending on the size of the numbers that are being replaced, the storage savings might not be as significant as those achieved by compressing text.

When a table-level dictionary is first created, it is built using a sample of data in the table. The dictionary is static, which means that it is not updated again unless you explicitly cause the dictionary to be rebuilt by using a classic, offline table reorganization. Even if you rebuild the dictionary, the dictionary reflects only a sample of the data from the entire table.

The table-level compression dictionary is stored in hidden rows in the same object that they apply to and is cached in memory for quick access. This dictionary does not occupy much space. Even for extremely large tables, the compression dictionary typically occupies only approximately 100 KB.

Page-level compression dictionaries

Adaptive compression uses page-level dictionaries in addition to table-level dictionaries. However, unlike table-level dictionaries, page-level dictionaries are automatically created or re-created as the database manager fills pages. Like table-level compression dictionaries, page-level dictionaries are stored in hidden rows within the table.

Column compression dictionaries

A column compression dictionary is used to compress data in a column of a column-organized table. When you load data into a column-organized table, the first phase is the analyze phase, which is unique to column-organized tables. The analyze phase occurs only if column compression dictionaries must be built, which happens during a **LOAD REPLACE** operation, a **LOAD REPLACE RESETDICTIONARY** operation, a **LOAD REPLACE RESETDICTIONARYONLY** operation, or a **LOAD INSERT**

operation (if the column-organized table is empty). The load utility analyzes the input data to determine the best encoding schemes for building column compression dictionaries.

If column compression dictionaries exist when you add a column to a table, the column compression dictionary for the new column contains only the default value for the new column. That means that any nondefault values that you later add to that column remain uncompressed. If column compression dictionaries do not exist when you add a column to the table, dictionaries for the new column and pre-existing columns are created when automatic dictionary creation (ADC) is triggered for the table. Although added columns generally do not compress as well as pre-existing columns of the same table, new columns can still benefit from page-level compression.

Table-level compression dictionary creation

Table-level compression dictionaries for tables that you enable for adaptive or classic row compression can be built automatically or manually. Tables that you enable for adaptive compression include page-level data dictionaries, which are always automatically created.

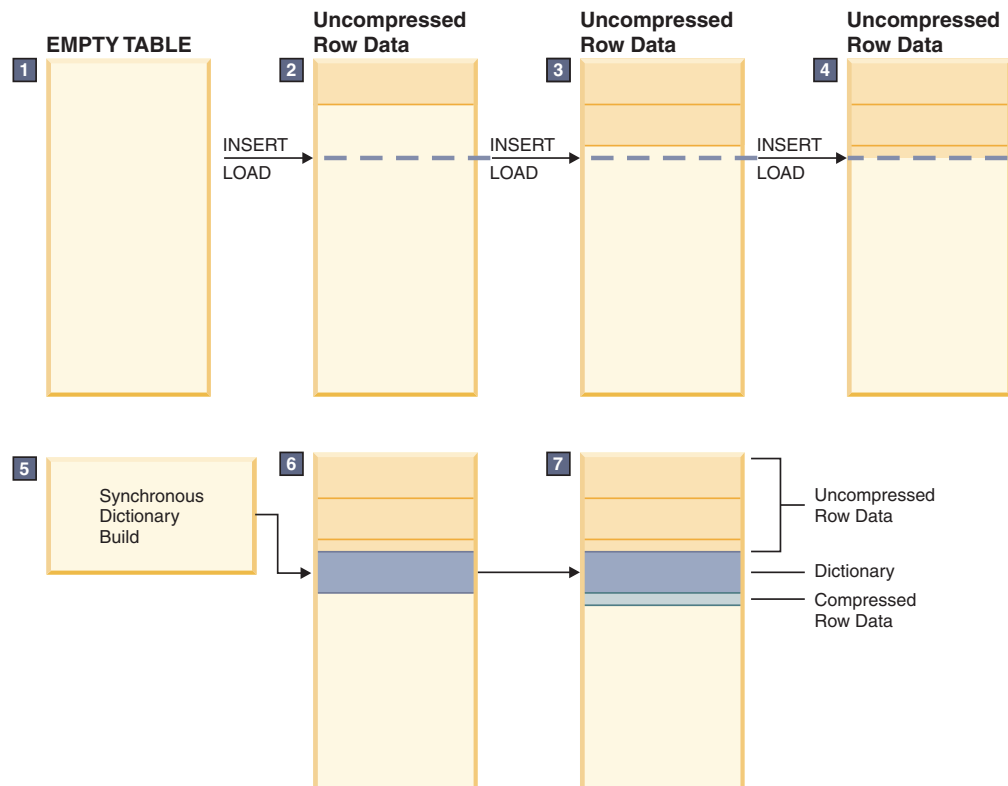
Automatic dictionary creation

Starting with DB2 Version 9.5, a table-level compression dictionary is created automatically if each of the following conditions is met:

- You set the COMPRESS attribute for the table by using the CREATE TABLE or ALTER TABLE statement with the COMPRESS YES ADAPTIVE or COMPRESS YES STATIC clause.
- A table-level compression dictionary does not already exist for the table.
- The table contains sufficient data for constructing a dictionary of repeated data.

Data that you move into the table after the dictionary is created is compressed using the dictionary if compression remains enabled.

The following diagram shows the process by which the compression dictionary is automatically created:



The sequence of events illustrated in the diagram is as follows:

1. A compression dictionary is not yet created, because the table is empty.
2. Data is inserted into the table by using insert or load operations and remains uncompressed.
3. As more data is inserted or loaded into the table, the data remains uncompressed.
4. After a threshold is reached, dictionary creation is triggered automatically if the COMPRESS attribute is set to YES ADAPTIVE or YES STATIC.
5. The dictionary is created.
6. The dictionary is appended to the table.
7. From this point forward, table-level compression is enabled, and the rows that are later inserted or added are compressed by the table-level compression dictionary.

Important: The rows that existed in a table before the dictionary was created remain uncompressed unless you change them or manually rebuild the dictionary.

If you create a table with DB2 Version 9.7 or later and the table contains at least one column of type XML, a second compression dictionary is created. This dictionary is used to compress the XML data that is stored in the default XML storage object that is associated with the table. Compression dictionary creation for XML data occurs automatically if each of the following conditions is met:

- You set the COMPRESS attribute on the table to YES ADAPTIVE or YES STATIC.
- A compression dictionary does not exist within that XML storage object.
- There is sufficient data in the XML storage object.

Restriction: You cannot compress data in XML columns that you created with DB2 Version 9.5 or DB2 Version 9.1. However, you can compress inline XML columns that you add to a table using DB2 Version 9.7 or later, provided the table was created without XML columns in an earlier release of the product. If a table that you created in an earlier release already has one or more XML columns and you want to add a compressed XML column by using DB2 Version 9.7 or later, you must use the `ADMIN_MOVE_TABLE` stored procedure to migrate the table before you can use compression.

The mechanism for creating table-level compression dictionaries for temporary tables is similar to the mechanism that is used for permanent tables. However, the database manager automatically makes the determination whether to use classic row compression for temporary tables, based on factors such as query complexity and the size of the result set.

Manual dictionary creation

Although dictionaries are created automatically when compression-enabled tables grow to a sufficient size, you can also force a table-level compression dictionary to be created if none exists by using the **REORG TABLE** command with the **RESETDICTIONARY** parameter. This command forces the creation of a compression dictionary if there is at least one row of data in the table. Table reorganization is an offline operation; one benefit of using automatic dictionary creation is that the table remains online as the dictionary is built.

Instead of using the **REORG TABLE** command to force the creation of a new dictionary, you can also use the **INSPECT** command with the **ROWCOMPESTIMATE** parameter. This command creates a compression dictionary if the table does not already have one. The advantage of this approach over performing a table reorganization is that the table remains online. Rows that you add later are subject to compression; however, rows that existed before you ran the **INSPECT** command remain uncompressed until you perform a table reorganization. However, if compression is enabled, automatic dictionary creation will usually take place shortly after you activate compression, likely before you even have a chance to use the **INSPECT** command.

Resetting compression dictionaries

Whether a table-level compression dictionary is created automatically or manually, the dictionary is static; after it is built, it does not change. As you add or update rows, they are compressed based on the data that exists in the compression dictionary. For many situations, this behavior is appropriate. Consider, for example, a table in a database that is used for maintaining customer accounts for a city water utility. Such a table might have columns such as `STREET_ADDRESS`, `CITY`, `PROVINCE`, `TELEPHONE_NUM`, `POSTAL_CODE`, and `ACCOUNT_TYPE`. If a compression dictionary is built with data from a such table, even if it is only a modestly sized table, there is likely sufficient repetitive information for classic row compression to yield significant space savings. Much of the data might be common from customer to customer, for example, the values of the `CITY`, `POSTAL_CODE`, or `PROVINCE` column or portions of the value in the `STREET_ADDRESS` or `TELEPHONE_NUM` column.

However, other tables might change significantly over time. Consider a table that is used for retail sales data as follows:

- A master table is used to accumulate data on a month-by-month basis.

- Each month, a new set of records is loaded into the table.

In this case, a compression dictionary created in, for example, April might not reflect repeating data from sales in later parts of the year. In situations where data in a table changes significantly over time, you might want to reset your compression dictionaries by using the **REORG TABLE** command with the **RESETDICTIONARY** parameter. The advantage of resetting the compression dictionary is that data from the entire table is considered when the dictionary is built.

Column compression dictionaries after a load or insert operation

The load utility analyzes the input data to determine the best encoding schemes for building column compression dictionaries.

Load operations can apply both column-level and page-level compression to all the input data because those dictionaries are built during the initial load operation.

If an insert operation is used instead of a load operation, automatic creation of the column compression dictionaries is not initiated until a threshold number of values is inserted. This design ensures that a large enough sample of values exists to build dictionaries that yield an adequate compression ratio. Values that are inserted before the column compression dictionaries are populated are not compressed by those dictionaries.

As of DB2 Cancun Release 10.5.0.4, page-level compression for insert operations is enabled by default. Such operations, including import and ingest operations, can take advantage of higher compression ratios and data clustering at the page level for improved insert performance. Values can be compressed by page-level compression dictionaries if the space savings from encoded values outweigh the cost of storing the page-level compression dictionaries in the data pages.

In general, you obtain a better compression ratio if you load a set of data rather than insert the same data. To help build the best dictionaries and get the maximum compression, use a good set of representative data during the initial load operation. This approach is preferable to loading a small initial subset of data that contains only a few distinct values and then appending the rest of the data by using multiple load operations.

If the compression ratio decreases significantly because new values are not being compressed, you can unload the table and then reload it to rebuild the dictionaries and help improve compression.

Index compression

Indexes, including indexes on declared or created temporary tables, can be compressed in order to reduce storage costs. This is especially useful for large OLTP and data warehouse environments.

By default, index compression is enabled for compressed tables, and disabled for uncompressed tables. You can override this default behavior by using the **COMPRESS YES** option of the **CREATE INDEX** statement. When working with existing indexes, use the **ALTER INDEX** statement to enable or disable index compression; you must then perform an index reorganization to rebuild the index.

Restriction: Index compression is not supported for the following types of indexes:

- block indexes
- XML path indexes.

In addition:

- Index specifications cannot be compressed
- Compression attributes for indexes on temporary tables cannot be altered with the ALTER INDEX command.

When index compression is enabled, the on-disk and memory format of index pages are modified based on the compression algorithms chosen by the database manager so as to minimize storage space. The degree of compression achieved will vary based on the type of index you are creating, as well as the data the index contains. For example, the database manager can compress an index with a large number of duplicate keys by storing an abbreviated format of the record identifier (RID) for the duplicate keys. In an index where there is a high degree of commonality in the prefixes of the index keys, the database manager can apply compression based on the similarities in prefixes of index keys.

There can be limitations and trade-offs associated with compression. If the indexes do not share common index column values or partial common prefixes, the benefits of index compression in terms of reduced storage might be negligible. And although a unique index on a timestamp column might have very high compression capabilities due to common values for year, month, day, hour, minute, or even seconds on the same leaf page, examining if common prefixes exist could cause performance to degrade.

If you believe that compression is not offering a benefit in your particular situation, you can either re-create the indexes without compression or alter the indexes and then perform an index reorganization to disable index compression.

There are a few things you should keep in mind when you are considering using index compression:

- If you enable row compression using the **COMPRESS YES** option on the CREATE TABLE or ALTER TABLE command, then by default, compression is enabled for all indexes for which compression is supported that are created after that point for that table, unless explicitly disabled by the CREATE INDEX or ALTER INDEX commands. Similarly, if you disable row compression with the CREATE TABLE or ALTER TABLE command, index compression is disabled for all indexes created after that point for that table unless explicitly enabled by the CREATE INDEX or ALTER INDEX commands.
- If you enable index compression using the ALTER INDEX command, compression will not take place until an index reorganization is performed. Similarly, if you disable compression, the index will remain compressed until you perform an index reorganization.
- During database migration, compression is not enabled for any indexes that might have been migrated. If you want compression to be used, you must use the ALTER INDEX command and then perform an index reorganization.
- CPU usage might increase slightly as a result of the processing required for index compression or decompression. If this is not acceptable, you can disable index compression for new or existing indexes.

Examples

Example 1: Checking whether an index is compressed.

The two statements that follow create a new table T1 that is enabled for row compression, and create an index I1 on T1.

```
CREATE TABLE T1 (C1 INT, C2 INT, C3 INT) COMPRESS YES
CREATE INDEX I1 ON T1(C1)
```

By default, indexes for T1 are compressed. The *compression attribute* for index T1, which shows whether compression is enabled, can be checked by using the catalog table or the admin table function:

```
SELECT COMPRESSION FROM SYSCAT.INDEXES WHERE TABNAME='T1'

COMPRESSION
-----
Y

1 record(s) selected.
```

Example 2: Determining whether compressed indexes require reorganization.

To see if compressed indexes require reorganization, use the **REORGCHK** command. Figure 10 shows the command being run on a table called T1:

```
REORGCHK ON TABLE SCHEMA1.T1
```

```
Doing RUNSTATS ....
```

```
Table statistics:
```

```
F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80
```

SCHEMA.NAME	CARD	OV	NP	FP	ACTBLK	TSIZE	F1	F2	F3	REORG

Table: SCHEMA1.T1	879	0	14	14	-	51861	0	100	100	---

```
Index statistics:
```

```
F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (Space used on leaf pages / Space available on non-empty leaf pages) >
    MIN(50, (100 - PCTFREE))
F6: (100 - PCTFREE) * (Amount of space available in an index with one less level /
    Amount of space required for all keys) < 100
F7: 100 * (Number of pseudo-deleted RIDs / Total number of RIDs) < 20
F8: 100 * (Number of pseudo-empty leaf pages / Total number of leaf pages) < 20
```

SCHEMA.NAME	INDCARD	LEAF	ELEAF	LVLS	NDEL	KEYS	LEAF_RECSIZE	NLEAF_RECSIZE...

Table: SCHEMA1.T1								
Index: SCHEMA1.I1	879	15	0	2	0	682	20	20...

...LEAF_PAGE_OVERHEAD	NLEAF_PAGE_OVERHEAD	PCT_PAGES_SAVED	F4	F5	F6	F7	F8	REORG

...	596		596	28	56	31	-	0 0 -----

Figure 10. Output of REORGCHK command

The output of the **REORGCHK** command has been formatted to fit the page.

Example 3: Determining the potential space savings of index compression.

For an example of how you can calculate potential index compression savings, refer to the documentation for the `ADMIN_GET_INDEX_COMPRESS_INFO` table function.

Backup compression

In addition to the storage savings you can achieve through row compression in your active database, you can also use backup compression to reduce the size of your database backups.

Whereas row compression works on a table-by-table basis, when you use compression for your backups, *all* of the data in the backup image is compressed, including catalog tables, index objects, LOB objects, auxiliary database files and database meta-data.

You can use backup compression with tables that use row compression. Keep in mind, however, that backup compression requires additional CPU resources and extra time. It may be sufficient to use table compression alone to achieve a reduction in your backup storage requirements. If you are using row compression, consider using backup compression only if storage optimization is of higher priority than the extra time it takes to perform the backup.

Tip: Consider using backup compression only on table spaces that do not contain compressed data if the following conditions apply:

- Data and index objects are separate from LOB and long field data, and
- You use row and index compression on the majority of your data tables and indexes, respectively

To use compression for your backups, use the `COMPRESS` option on the **BACKUP DATABASE** command.

Chapter 12. DB2 compatibility features

The DB2 product provides a number of features that reduce the time and complexity of enabling some applications that were written for relational database products other than the DB2 product to run on a DB2 system.

Some of these features, including the following ones, are enabled by default.

- Implicit casting (weak typing), which reduces the number of SQL statements that you must modify to enable applications to run on the DB2 product.
- New built-in scalar functions. For details, see Built-in functions (see *SQL Reference Volume 1*).
- Improvements to the `TIMESTAMP_FORMAT` and `VARCHAR_FORMAT` scalar functions. The `TIMESTAMP_FORMAT` function returns a timestamp for an input string, using a specified format. The `VARCHAR_FORMAT` function returns a string representation of an input expression that has been formatted according to a specified character template. `TO_DATE` and `TO_TIMESTAMP` are synonyms for `TIMESTAMP_FORMAT`, and `TO_CHAR` is a synonym for `VARCHAR_FORMAT`.
- The lifting of several SQL restrictions, resulting in more compatible syntax between products. For example, the use of correlation names in subqueries and table functions is now optional.
- Synonyms for syntax that is used by other database products. Examples are as follows:
 - `UNIQUE` is a synonym for `DISTINCT` in the column functions and the select list of a query.
 - `MINUS` is a synonym for the `EXCEPT` set operator
 - You can use `seqname.NEXTVAL` in place of the SQL standard syntax `NEXT VALUE FOR seqname`. You can also use `seqname.CURRVAL` in place of the SQL standard syntax `PREVIOUS VALUE FOR seqname`.
- Global variables, which you can use to easily map package variables, emulate `@@nested`, `@@level` or `@errorlevel` global variables, or pass information from DB2 applications to triggers, functions, or procedures.
- An `ARRAY` collection data type that you use to easily map to `VARRAY` constructs in SQL procedures.
- Increased identifier length limits.
- The pseudocolumn `ROWID`, which you can use to refer to the `RID`. An unqualified `ROWID` reference is equivalent to `RID_BIT()`, and a qualified `ROWID` reference, such as `EMPLOYEE.ROWID`, is equivalent to `RID_BIT(EMPLOYEE)`.

You can optionally enable the following other features by setting the **DB2_COMPATIBILITY_VECTOR** registry variable. These features are disabled by default.

- An implementation of hierarchical queries using `CONNECT BY PRIOR` syntax.
- Support for outer joins using the outer join operator, which is the plus sign (+).
- Use of the `DATE` data type as `TIMESTAMP(0)`, a combined date and time value.
- Syntax and semantics to support the `NUMBER` data type.
- Syntax and semantics to support the `VARCHAR2` data type.

- The ROWNUM pseudocolumn, which is a synonym for ROW_NUMBER() OVER(). However, the ROWNUM pseudocolumn is allowed in the SELECT list and in the WHERE clause of the SELECT statement.
- A dummy table named DUAL, which provides a capability that is similar to that of the SYSIBM.SYSDUMMY1 table.
- Alternative semantics for the TRUNCATE statement, such that IMMEDIATE is an optional keyword that is assumed to be the default if not specified. An implicit commit operation is performed before the TRUNCATE statement executes if the TRUNCATE statement is not the first statement in the logical unit of work.
- Support for assigning the CHAR or GRAPHIC data type instead of the VARCHAR or VARGRAPHIC data type to character and graphic string constants whose byte lengths are less than or equal to 254.
- Use of collection methods to perform operations on arrays, such as FIRST, LAST, NEXT, and previous.
- Support for creating Oracle data dictionary-compatible views.
- Support for compiling and executing PL/SQL statements and other language elements.
- Support for making cursors insensitive to subsequent statements by materializing the cursors at OPEN time.
- Support for INOUT parameters in procedures that you define with defaults and can invoke without specifying the arguments for those parameters.

Additional resources

For information about the IBM Database Conversion Workbench (DCW), see Database Conversion Workbench.

For information about the DB2 Oracle database compatibility features, see Oracle to DB2 Conversion Guide: Compatibility Made Easy.

DB2_COMPATIBILITY_VECTOR registry variable

The **DB2_COMPATIBILITY_VECTOR** registry variable enables one or more DB2 compatibility features. These features ease the task of migrating applications that were written for relational database products other than the DB2 product to the DB2 product.

Registry variable settings

Values are as follows:

- NULL (default). This means that no compatibility features are supported.
- A hexadecimal value of 0000 - FFFF. Each bit in the variable value enables an individual compatibility feature. For the meanings of each bit, see table-1.
- ORA. This value, which is equivalent to the hexadecimal value 70FFF, enables all the DB2 compatibility features for Oracle applications.
- SYB. This value, which is equivalent to the hexadecimal value 3004, enables all the DB2 compatibility features for Sybase applications.
- MYS. This value, which is equivalent to the hexadecimal value 4000, enables all the DB2 compatibility features for MySQL applications. Currently, the only supported MySQL compatibility feature is enablement of the LIMIT and OFFSET clauses.

A setting of ORA, SYB, or MYS can enable all the compatibility features.

Important: Enable compatibility features only if you require them for a specific compatibility purpose. If you enable DB2 compatibility features, some SQL behavior changes from what is documented in the SQL reference information. To determine the potential effects of a compatibility feature on your SQL applications, see the documentation that is associated with the compatibility feature.

The following table specifies the settings that you require to enable individual compatibility features.

Table 17. DB2_COMPATIBILITY_VECTOR registry variable values

Bit position (hexadecimal value)	Compatibility feature	Description
1 (0x01)	ROWNUM pseudocolumn	Enables the use of the ROWNUM pseudocolumn as a synonym for the ROW_NUMBER() OVER() function and permits the ROWNUM pseudocolumn to appear in the WHERE clause of SQL statements.
2 (0x02)	DUAL table	Resolves unqualified references to the DUAL table as SYSIBM.DUAL.
3 (0x04)	Outer join operator	Enables support for the outer join operator, which is the plus sign (+).
4 (0x08)	Hierarchical queries	Enables support for hierarchical queries, which use the CONNECT BY clause.
5 (0x10)	NUMBER data type ¹	Enables support for the NUMBER data type and associated numeric processing. When you create a database with this support enabled, the number_compat database configuration parameter is set to ON.
6 (0x20)	VARCHAR2 data type ¹	Enables support for the VARCHAR2 and NVARCHAR2 data types and associated character string processing. When you create a database with this support enabled, the varchar2_compat database configuration parameter is set to ON.
7 (0x40)	DATE data type ¹	Enables the interpretation of the DATE data type as the TIMESTAMP(0) data type, a combined date and time value. For example, "VALUES CURRENT DATE" in date compatibility mode returns a value such as 2011-02-17-10.43.55. When you create a database with this support enabled, the date_compat database configuration parameter is set to ON.

Table 17. DB2_COMPATIBILITY_VECTOR registry variable values (continued)

Bit position (hexadecimal value)	Compatibility feature	Description
8 (0x80)	TRUNCATE TABLE	Enables alternative semantics for the TRUNCATE statement, under which IMMEDIATE is an optional keyword that is assumed to be the default if you do not specify it. An implicit commit operation is performed before the TRUNCATE statement executes if the TRUNCATE statement is not the first statement in the logical unit of work.
9 (0x100)	Character literals	Enables the assignment of the CHAR or GRAPHIC data type, instead of the VARCHAR or VARGRAPHIC data type, to character and graphic string constants whose byte lengths are less than or equal to 254.
10 (0x200)	Collection methods	Enables the use of methods to perform operations on arrays, such as first, last, next, and previous. This value also enables the use of parentheses in place of square brackets in references to specific elements in an array. For example, array1(<i>i</i>) refers to element <i>i</i> of array1.
11 (0x400)	Oracle data dictionary-compatible views ¹	Enables the creation of Oracle data dictionary-compatible views.
12 (0x800)	PL/SQL compilation ²	Enables the compilation and execution of PL/SQL statements and language elements.
13 (0x1000)	Insensitive cursors	Enables cursors that are defined with WITH RETURN to be insensitive if the select-statement does not explicitly specify FOR UPDATE.
14 (0x2000)	INOUT parameters	Enables the specification of DEFAULT for INOUT parameter declarations.
15 (0x4000)	LIMIT and OFFSET clauses	Enables the use of the MySQL-compatible and PostgreSQL-compatible LIMIT and OFFSET clauses in fullselect, UPDATE, and DELETE statements.
17 (0x10000)	SQL data-access-level enforcement	Enables routines to enforce SQL data-access levels at run time.
18 (0x20000)	Oracle database link syntax	Enables Oracle database link syntax for accessing objects in other databases.

Table 17. *DB2_COMPATIBILITY_VECTOR* registry variable values (continued)

Bit position (hexadecimal value)	Compatibility feature	Description
19 (0x40000)	Synonym usage	Enables the use of synonyms. When you set the DB2_COMPATIBILITY_VECTOR registry variable to support the use of synonyms, you cannot issue the alter, drop, rename, or truncate statements with a table synonym as the target. You cannot issue the alter or drop statements with a view synonym as the target. You cannot issue the alter or drop statement with a sequence synonym as the target.
<ol style="list-style-type: none"> 1. This feature is applicable only during database creation. Enabling or disabling this feature after creating a database affects only subsequently created databases. 2. See Restrictions on PL/SQL support. 		

Usage

You set and update the **DB2_COMPATIBILITY_VECTOR** registry variable by using the **db2set** command. You can set the **DB2_COMPATIBILITY_VECTOR** registry variable with combination of the compatibility features by adding the digits of the hexadecimal values that are associated with the compatibility features. A new setting for the registry variable does not take effect until after you stop and restart the instance. Also, you must rebind DB2 packages for the change to take effect. Packages that you do not rebind explicitly will pick up the change at the next implicit rebind.

If you set the **DB2_COMPATIBILITY_VECTOR** registry variable, create databases as Unicode databases.

Example 1

This example shows how to set the registry variable to enable all the supported Oracle compatibility features:

```
db2set DB2_COMPATIBILITY_VECTOR=ORA
db2stop
db2start
```

Example 2

This example shows how to set the registry variable to provide both the ROWNUM pseudocolumn (0x01) and DUAL table (0x02) support that is specified in the previous table:

```
db2set DB2_COMPATIBILITY_VECTOR=03
db2stop
db2start
```

Example 3

This example shows how to disable all compatibility features by resetting the **DB2_COMPATIBILITY_VECTOR** registry variable:

```
db2set DB2_COMPATIBILITY_VECTOR=  
db2stop  
db2start
```

If you create a database when any of the following features are enabled and then disable all the compatibility features, the database will still be enabled for these features:

- The NUMBER data type
- The VARCHAR2 data type
- The DATE data type as TIMESTAMP(0)
- Creation of Oracle data dictionary-compatible views

Compatibility database configuration parameters

You can use database configuration parameters to indicate whether the compatibility semantics associated with the certain data types are applied to the connected database.

The compatibility parameters that can be checked are:

date_compat

Indicates whether the DATE data type compatibility semantics that are associated with the TIMESTAMP(0) data type are applied to the connected database.

number_compat

Indicates whether the compatibility semantics that are associated with the NUMBER data type are applied to the connected database.

varchar2_compat

Indicates whether the compatibility semantics that are associated with the VARCHAR2 data type are applied to the connected database.

The value of each of these parameters is determined at database creation time, and is based on the setting of the **DB2_COMPATIBILITY_VECTOR** registry variable. You cannot change the value.

Terminology mapping: Oracle to DB2 products

Because Oracle applications can be enabled to work with DB2 data servers when the DB2 environment is set up appropriately, it is important to understand how certain Oracle concepts map to DB2 concepts.

This section provides an overview of the data management concepts used by Oracle, and the similarities or differences between these concepts and those used by the DB2 product. Table 18 provides a concise summary of commonly used Oracle terms and their DB2 equivalents.

Table 18. Mapping of common Oracle concepts to DB2 concepts

Oracle concept	DB2 concept	Notes
active log	active log	The concepts are the same.
actual parameter	argument	The concepts are the same.

Table 18. Mapping of common Oracle concepts to DB2 concepts (continued)

Oracle concept	DB2 concept	Notes
alert log	db2diag log files and administration notification log	The db2diag log files are primarily intended for use by IBM Software Support for troubleshooting. The administration notification log is primarily intended for use by database and system administrators for troubleshooting. Administration notification log messages are also logged in the db2diag log files, using a standardized message format.
archive log	offline archive log	The concepts are the same.
archive log mode	log archiving	The concepts are the same.
background_dump_dest	diagpath	The concepts are the same.
created global temporary table	created global temporary table	The concepts are the same.
cursor sharing	statement concentrator	The concepts are the same.
data block	data page	The concepts are the same.
data buffer cache	buffer pool	The concepts are the same. However, in the DB2 product, you can have as many buffer pools as you like, of any page size.
data dictionary	system catalog	The DB2 system catalog contains metadata in the form of tables and views. The database manager creates and maintains two sets of system catalog views that are defined on the base system catalog tables: SYSCAT views Read-only views. SYSSTAT views Updatable views that contain statistical information that is used by the optimizer.
data dictionary cache	catalog cache	The concepts are the same.
data file	container	DB2 data is physically stored in containers, which contain objects.
database link	nickname	A nickname is an identifier that refers to an object at a remote data source, that is, a federated database object.

Table 18. Mapping of common Oracle concepts to DB2 concepts (continued)

Oracle concept	DB2 concept	Notes
dual table	dual table	The concepts are the same.
dynamic performance views	SQL administrative views	SQL administrative views, which use schema SYSIBMADM, return system data, database configuration, and monitor data about a specific area of the database system.
extent	extent	A DB2 extent is made up of a set of contiguous data pages.
formal parameter	parameter	The concepts are the same.
global index	nonpartitioned index	The concepts are the same.
inactive log	online archive log	The concepts are the same.
init.ora file and Server Parameter File (SPFILE)	database manager configuration file and database configuration file	A DB2 instance can contain multiple databases. Therefore, configuration parameters and their values are stored at both the instance level, in the database manager configuration file, and at the database level, in the database configuration file. You manage the database manager configuration file through the GET DBM CFG or UPDATE DBM CFG command. You manage the database configuration file through the GET DB CFG or UPDATE DB CFG command.
instance	instance or database manager	An instance is a combination of background processes and shared memory. A DB2 instance is also known as a database manager.
large pool	utility heap	The utility heap is used by the backup, restore, and load utilities.
library cache	package cache	The package cache, which is allocated from database shared memory, is used to cache sections for static and dynamic SQL and XQuery statements executed on a database.
local index	partitioned index	This is the same concept.

Table 18. Mapping of common Oracle concepts to DB2 concepts (continued)

Oracle concept	DB2 concept	Notes
materialized view	materialized query table (MQT)	An MQT is a table whose definition is based on the results of a query and can help improve performance. The DB2 SQL compiler determines whether a query would run more efficiently against an MQT than it would against the base table on which the MQT is based.
noarchive log mode	circular logging	The concepts are the same.
Oracle Call Interface (OCI) Oracle Call Interface (OCI)	DB2CI	DB2CI is a C and C++ application programming interface that uses function calls to connect to DB2 databases, manage cursors, and perform SQL statements.
Oracle Call Interface (OCI) Oracle Call Interface (OCI)	Call Level Interface (CLI)	CLI is a C and C++ application programming interface that uses function calls to pass dynamic SQL statements as function arguments. In most cases, you can replace an OCI function with a CLI function and relevant changes to the supporting program code.
ORACLE_SID environment variable	DB2INSTANCE environment variable	The concepts are the same.
partitioned tables	partitioned tables	The concepts are the same.
Procedural Language/Structured Query Language (PL/SQL)	SQL Procedural Language (SQL PL)	<p>SQL PL is an extension of SQL that consists of statements and other language elements. SQL PL provides statements for declaring variables and condition handlers, assigning values to variables, and implementing procedural logic. SQL PL is a subset of the SQL/Persistent Stored Modules (SQL/PSM) language standard.</p> <p>You can use DB2 interfaces to compile and execute Oracle PL/SQL statements.</p>

Table 18. Mapping of common Oracle concepts to DB2 concepts (continued)

Oracle concept	DB2 concept	Notes
program global area (PGA)	application shared memory and agent private memory	Application shared memory stores information that is shared between a database and a particular application: primarily, rows of data that are passed to or from the database. Agent private memory stores information that is used to service a particular application, such as sort heaps, cursor information, and session contexts.
redo log	transaction log	The transaction log records database transactions. You can use it for recovery.
role	role	The concepts are the same.
segment	storage object	The concepts are the same.
session	session; database connection	The concepts are the same.
startup nomount command	db2start command	The command used to start the instance.
synonym	alias	An alias is an alternative name for a table, a view, a nickname, or another alias. The term <i>synonym</i> can be specified instead of <i>alias</i> . Aliases are not used to control what version of a DB2 procedure or user-defined function is used by an application. To control the version, use the SET PATH statement to add the required schema to the value of the CURRENT PATH special register.
system global area (SGA)	instance shared memory and database shared memory	The instance shared memory stores all of the information for a particular instance, such as lists of all active connections and security information. The database shared memory stores information for a particular database, such as package caches, log buffers, and buffer pools.
SYSTEM table space	SYSCATSPACE table space	The SYSCATSPACE table space contains the system catalog. This table space is created by default when you create a database.
table space	table space	The concepts are the same.

Table 18. Mapping of common Oracle concepts to DB2 concepts (continued)

Oracle concept	DB2 concept	Notes
user global area (UGA)	application global memory	Application global memory comprises application shared memory and application-specific memory.

Oracle data dictionary-compatible views

When you set the **DB2_COMPATIBILITY_VECTOR** registry variable to support Oracle data dictionary-compatible views, the views are automatically created when you create a database.

You enable Oracle data dictionary-compatible view support by setting the **DB2_COMPATIBILITY_VECTOR** registry variable to hexadecimal value 0x400 (bit position 11), and then stop and restart the instance to have the new setting take effect.

```
db2set DB2_COMPATIBILITY_VECTOR=400
db2stop
db2start
```

To take full advantage of the DB2 compatibility features for Oracle applications, the recommended setting for the **DB2_COMPATIBILITY_VECTOR** is ORA, which sets all of the compatibility bits.

The data dictionary is a repository for database metadata. The data dictionary views are self-describing. The **DICTIONARY** view returns a listing of all data dictionary views with comments that describe the content of each view. The **DICTIONARY_COLUMNS** view returns a list of all columns in all data dictionary views. With these two views, you can determine what information is available and how to access it.

There are three different versions of each data dictionary view, and each version is identified by the prefix of the view name.

- **ALL_*** views return information about objects to which the current user has access.
- **DBA_*** views return information about all objects in the database, regardless of who owns them.
- **USER_*** views return information about objects that are owned by the current database user.

Not all versions apply to each view.

The data dictionary definition includes **CREATE VIEW**, **CREATE PUBLIC SYNONYM**, and **COMMENT** statements for each view that is compatible with the Oracle data dictionary. The views, which are created in the **SYSIBMADM** schema, are listed in Table 19.

Table 19. Oracle data dictionary-compatible views

Category	Defined views
General	DICTIONARY, DICTIONARY_COLUMNS USER_CATALOG, DBA_CATALOG, ALL_CATALOG USER_DEPENDENCIES, DBA_DEPENDENCIES, ALL_DEPENDENCIES USER_OBJECTS, DBA_OBJECTS, ALL_OBJECTS USER_SEQUENCES, DBA_SEQUENCES, ALL_SEQUENCES USER_TABLESPACES, DBA_TABLESPACES

Table 19. Oracle data dictionary-compatible views (continued)

Category	Defined views
Tables or views	USER_CONSTRAINTS, DBA_CONSTRAINTS, ALL_CONSTRAINTS USER_CONS_COLUMNS, DBA_CONS_COLUMNS, ALL_CONS_COLUMNS USER_INDEXES, DBA_INDEXES, ALL_INDEXES USER_IND_COLUMNS, DBA_IND_COLUMNS, ALL_IND_COLUMNS USER_TAB_PARTITIONS, DBA_TAB_PARTITIONS, ALL_TAB_PARTITIONS USER_PART_TABLES, DBA_PART_TABLES, ALL_PART_TABLES USER_PART_KEY_COLUMNS, DBA_PART_KEY_COLUMNS, ALL_PART_KEY_COLUMNS USER_SYNONYMS, DBA_SYNONYMS, ALL_SYNONYMS USER_TABLES, DBA_TABLES, ALL_TABLES USER_TAB_COMMENTS, DBA_TAB_COMMENTS, ALL_TAB_COMMENTS USER_TAB_COLUMNS, DBA_TAB_COLUMNS, ALL_TAB_COLUMNS USER_COL_COMMENTS, DBA_COL_COMMENTS, ALL_COL_COMMENTS USER_TAB_COL_STATISTICS, DBA_TAB_COL_STATISTICS, ALL_TAB_COL_STATISTICS USER_VIEWS, DBA_VIEWS, ALL_VIEWS USER_VIEW_COLUMNS, DBA_VIEW_COLUMNS, ALL_VIEW_COLUMNS
Programming objects	USER_PROCEDURES, DBA_PROCEDURES, ALL_PROCEDURES USER_SOURCE, DBA_SOURCE, ALL_SOURCE USER_TRIGGERS, DBA_TRIGGERS, ALL_TRIGGERS USER_ERRORS, DBA_ERRORS, ALL_ERRORS USER_ARGUMENTS, DBA_ARGUMENTS, ALL_ARGUMENTS
Security	USER_ROLE_PRIVS, DBA_ROLE_PRIVS, ROLE_ROLE_PRIVS SESSION_ROLES USER_SYS_PRIVS, DBA_SYS_PRIVS, ROLE_SYS_PRIVS SESSION_PRIVS USER_TAB_PRIVS, DBA_TAB_PRIVS, ALL_TAB_PRIVS, ROLE_TAB_PRIVS USER_TAB_PRIVS_MADE, ALL_TAB_PRIVS_MADE USER_TAB_PRIVS_RECD, ALL_TAB_PRIVS_RECD DBA_ROLES

Examples

The following examples show how to enable, get information about, and use data dictionary-compatible views for a database that is named MYDB:

- Enable the creation of data dictionary-compatible views:


```
db2set DB2_COMPATIBILITY_VECTOR=ORA
db2stop
db2start
db2 create db mydb
```
- Determine what data dictionary-compatible views are available:


```
connect to mydb
select * from dictionary
```
- Use the USER_SYS_PRIVS view to show all the system privileges that the current user has been granted:


```
connect to mydb
select * from user_sys_privs
```
- Determine the column definitions for the DBA_TABLES view:


```
connect to mydb
describe select * from dba_tables
```

Oracle database link syntax

When you set the DB2_COMPATIBILITY_VECTOR registry variable to support Oracle database link syntax, you can connect with a remote database, table, or view.

Enablement

You enable Oracle database link syntax support by setting the DB2_COMPATIBILITY_VECTOR registry variable to hexadecimal value 0x20000 (bit position 18), and then stop and restart the instance to have the new setting take effect.

```
db2set DB2_COMPATIBILITY_VECTOR=20000
db2stop
db2start
```

To take full advantage of the DB2 compatibility features for Oracle applications, the recommended setting for the DB2_COMPATIBILITY_VECTOR is ORA, which sets all of the compatibility bits.

The database link syntax uses the @ (at sign) to indicate an in or membership condition. For example, to access a remote object pencils under schema user using a database link to stock, you can use:

```
SELECT * FROM user.pencils@stock;
```

Note: The DB2 system supports the use of the @ character as a valid character in an ordinary identifier. For example, you can create a table with pencils@stock as its name. When database link support is enabled, the @ character is treated as a special delimiter in table, view, and column references. If you want to use @ in database object names when link support is enabled, you must enclose the name with double quotes.

Example

Remote object references are formatted as:

```
<schema_name>,<object_name>@<server_name>
```

Column references can also be included:

```
<schema_name>,<object_name>,<column_name>@<server_name>
```

The following SELECT statements query a remote table named EMPLOYEE:

```
SELECT birthdate FROM rschema.employee@sudb WHERE firstname='SAM'
SELECT rschema.employee.birthdate@sudb FROM rschema.employee@sudb
WHERE rschema.employee.firstname@sudb ='SAM'
```

You can also issue UPDATE, INSERT, and DELETE statements against a remote table:

```
UPDATE rschema.employee@sudb SET firstname='MARY'
INSERT INTO rschema.employee@sudb VALUES ('Bob')
DELETE FROM rschema.employee@sudb
```

Setting up the DB2 environment for Oracle application enablement

You can reduce the time and complexity of enabling Oracle applications to work with DB2 data servers if you set up the DB2 environment appropriately.

Before you begin

- A DB2 data server product must be installed.
- You require SYSADM and the appropriate operating system authority to issue the **db2set** command.
- You require SYSADM or SYSCTRL authority to issue the **CREATE DATABASE** command.

About this task

The DB2 product can support many commonly referenced features from other database products. This task is a prerequisite for executing PL/SQL statements or

SQL statements that reference Oracle data types from DB2 interfaces or for using any other SQL compatibility features. You enable DB2 compatibility features at the database level; you cannot disable them.

Procedure

To enable Oracle applications to work with DB2 data servers:

1. In a DB2 command window, start the DB2 database manager by issuing the following command:

```
db2start
```

2. Set the **DB2_COMPATIBILITY_VECTOR** registry variable to one of the following values:

- The hexadecimal value that enables the specific compatibility feature that you want to use.
- To take advantage of all the DB2 compatibility features, ORA, as shown in the following command. This is the recommended setting.

```
db2set DB2_COMPATIBILITY_VECTOR=ORA
```

3. Enable deferred prepare support by setting the **DB2_DEFERRED_PREPARE_SEMANTICS** registry variable to YES, as shown:

```
db2set DB2_DEFERRED_PREPARE_SEMANTICS=YES
```

If you set the **DB2_COMPATIBILITY_VECTOR** registry variable to ORA and do not set the **DB2_DEFERRED_PREPARE_SEMANTICS** registry variable, a default value of YES is used. However, it is recommended that you explicitly set the **DB2_DEFERRED_PREPARE_SEMANTICS** registry variable to YES.

4. Stop the database manager by issuing the **db2stop** command:

```
db2stop
```

5. Start the database manager by issuing the **db2start** command:

```
db2start
```

6. Create your DB2 database by issuing the **CREATE DATABASE** command. By default, databases are created as Unicode databases (this is also the preferred setting). For example, to create a database that is named DB, issue the following command:

```
db2 CREATE DATABASE DB
```

7. Optional: Run a Command Line Processor Plus (CLPPlus) or command line processor (CLP) script (for example, script.sql) to verify that the database supports PL/SQL statements and data types. The following CLPPlus script creates and then calls a simple procedure:

```
CONNECT user@hostname:port/dbname;
```

```
CREATE TABLE t1 (c1 NUMBER);
```

```
CREATE OR REPLACE PROCEDURE testdb(num IN NUMBER, message OUT VARCHAR2)
```

```
AS
```

```
BEGIN
```

```
INSERT INTO t1 VALUES (num);
```

```
message := 'The number you passed is: ' || TO_CHAR(num);
```

```
END;
```

```
/
```

```
CALL testdb(100, ?);
```

```
DISCONNECT;
```

```
EXIT;
```


To run the CLPPlus script, issue the following command:

```
clpplus @script.sql
```

The following example shows the CLP version of the same script. This script uses the **SET SQLCOMPAT PLSQL** command to enable recognition of the forward slash character (/) on a new line as a PL/SQL statement termination character.

```
CONNECT TO DB;
```

```
SET SQLCOMPAT PLSQL;
```

```
-- Semicolon is used to terminate  
-- the CREATE TABLE statement:
```

```
CREATE TABLE t1 (c1 NUMBER);
```

```
-- Forward slash on a new line is used to terminate  
-- the CREATE PROCEDURE statement:
```

```
CREATE OR REPLACE PROCEDURE testdb(num IN NUMBER, message OUT VARCHAR2)
```

```
AS
```

```
BEGIN
```

```
    INSERT INTO t1 VALUES (num);
```

```
    message := 'The number you passed is: ' || TO_CHAR(num);  
END;
```

```
/
```

```
CALL testdb(100, ?);
```

```
SET SQLCOMPAT DB2;
```

```
CONNECT RESET;
```

To run the CLP script, issue the following command:

```
db2 -tvf script.sql
```

Results

The DB2 database that you created is enabled for Oracle applications. You can now use the compatibility features that you enabled. Only databases created after the **DB2_COMPATIBILITY_VECTOR** registry variable is set are enabled for Oracle applications.

What to do next

- Start using the CLPPlus interface.
- Execute PL/SQL scripts and statements.
- Transfer database object definitions.
- Enable database applications.

Part 3. Monitoring DB2 Activity

Monitoring DB2 Activity consist of performing tasks associated with examining the operational status of your database. DB2 provides multiples interfaces for database and workload monitoring. It also provides tools for obtaining information about access plans and troubleshooting problems.

Database monitoring is a vital activity for the maintenance of the performance and health of your database management system. To facilitate monitoring, DB2 collects information from the database manager, its databases, and any connected applications. With this information you can perform the following types of tasks, and more:

- Forecast hardware requirements based on database usage patterns.
- Analyze the performance of individual applications or SQL queries.
- Track the usage of indexes and tables.
- Pinpoint the cause of poor system performance.
- Assess the impact of optimization activities (for example, altering database manager configuration parameters, adding indexes, or modifying SQL queries).

Chapter 13. Monitoring metrics for column-organized tables

New monitor elements for column-organized tables can help you to understand and tune database server workloads that involve queries against these tables. Using the DB2 monitor interfaces, you can obtain most of the information that you require to monitor these workloads.

Note: All elements that include *TEMP* or *GBP* in their names are reserved for future use.

New monitor elements to assess buffer pool efficiency

You can use a new set of monitor elements to monitor data page I/O for column-organized tables separately from row-organized tables. You can use these monitor elements to understand what portion of the I/O is being driven by access to column-organized tables when a workload impacts both row-organized and column-organized tables. These elements can also help you to tune the system, for example, by helping you to decide whether to place column-organized tables in separate table spaces, or whether to use a separate buffer pool.

The new monitor elements are as follows:

- Counters for total logical and physical column-organized data page reads and pages found:
 - POOL_COL_L_READS
 - POOL_TEMP_COL_L_READS
 - POOL_COL_P_READS
 - POOL_TEMP_COL_P_READS
 - POOL_COL_LBP_PAGES_FOUND
- Counter for column-organized data page writes:
 - POOL_COL_WRITES
- Counters for asynchronous column-organized data page reads and writes and pages found:
 - POOL_ASYNC_COL_READS
 - POOL_ASYNC_COL_READ_REQS
 - POOL_ASYNC_COL_WRITES
 - POOL_ASYNC_COL_LBP_PAGES_FOUND
- Counters for column-organized data page reads per table and per statement per table, reported through monitor usage lists:
 - OBJECT_COL_L_READS
 - OBJECT_COL_P_READS
 - OBJECT_COL_GBP_L_READS
 - OBJECT_COL_GBP_P_READS
 - OBJECT_COL_GBP_INVALID_PAGES
 - OBJECT_COL_LBP_PAGES_FOUND
 - OBJECT_COL_GBP_INDEP_PAGES_FOUND_IN_LBP

New monitor elements to monitor prefetch requests for data in column-organized tables

The prefetch logic for queries that access column-organized tables is used to asynchronously fetch only those pages that each thread will read for each column that is accessed during query execution. If the pages for a particular column are consistently available in the buffer pool, prefetching for that column is disabled until the pages are being read synchronously, at which time prefetching for that column is enabled again.

Although the number of pages that a thread can prefetch simultaneously is limited by the prefetch size of the table space that is being accessed, several threads can prefetch pages simultaneously.

New monitor elements can help you to track the volume of requests for data in column-organized tables that are submitted to prefetchers and the number of pages that prefetchers skipped reading because the pages were in memory. Efficient prefetching of data in column-organized tables is important for mitigating the I/O costs of data scans.

The following monitor elements enable the monitoring of prefetch requests for data in column-organized tables:

- POOL_QUEUED_ASYNC_COL_REQS
- POOL_QUEUED_ASYNC_TEMP_COL_REQS
- POOL_QUEUED_ASYNC_COL_PAGES
- POOL_QUEUED_ASYNC_TEMP_COL_PAGES
- POOL_FAILED_ASYNC_COL_REQS
- POOL_FAILED_ASYNC_TEMP_COL_REQS
- SKIPPED_PREFETCH_COL_P_READS
- SKIPPED_PREFETCH_TEMP_COL_P_READS
- SKIPPED_PREFETCH_UOW_COL_P_READS
- SKIPPED_PREFETCH_UOW_TEMP_COL_P_READS

New monitor elements to measure column data size

A column-organized table is associated with a new table object where the column data is stored.

The following monitor elements help you to estimate the size of the column data:

- COL_OBJECT_L_SIZE
- COL_OBJECT_P_SIZE
- COL_OBJECT_L_PAGES

The first two elements accurately reflect the logical and physical size of the column-organized data object but are more expensive to determine because they must calculate the space that is being used. These elements are reported by the ADMIN_GET_TAB_INFO table function. The COL_OBJECT_L_PAGES element is similar to the existing DATA_OBJECT_L_PAGES element and provides a cheaper alternative to estimating size (the number of pages multiplied by the page size), although this estimate might be inaccurate.

New monitor element to report information about data organization

A new monitor element, TAB_ORGANIZATION, reports information about the organization of data in a table and is returned by the MON_GET_TABLE table function.

New monitor elements to measure time spent

Time-spent monitor elements provide information about how the DB2 database manager spends time processing column-organized tables. The time-spent elements are broadly categorized into wait time elements and processing time elements.

The following monitor elements are added to the time-spent hierarchy:

- TOTAL_COL_TIME
- TOTAL_COL_PROC_TIME
- TOTAL_COL_EXECUTIONS

Those three monitor elements count the total time that is spent in column-organized data processing across all column-organized processing subagents. The TOTAL_COL_TIME element represents the total elapsed time over all column-organized processing subagents. The TOTAL_COL_PROC_TIME element represents the subset of this total elapsed time in which the column-organized processing subagents were not idle on a measured wait time (for example, lock wait or IO). The TOTAL_COL_EXECUTIONS element represents the total number of times that data in column-organized tables was accessed during statement execution.

The parent element of TOTAL_COL_TIME is TOTAL_SECTION_TIME. The parent element of TOTAL_COL_PROC_TIME is TOTAL_SECTION_PROC_TIME. The parent elements are the same in both the request and activity dimensions.

New monitor elements for the hashed GROUP BY operator

The GROUP BY operator on column-organized tables uses hashing as the grouping method. Hashed GROUP BY operators are consumers of sort memory. The following new monitor elements support the monitoring of sort memory consumption during hashed GROUP BY operations. These elements are similar to existing monitor elements for other sort memory consumers.

- TOTAL_HASH_GRPBYS
- ACTIVE_HASH_GRPBYS
- HASH_GRPBY_OVERFLOW
- POST_THRESHOLD_HASH_GRPBYS
- ACTIVE_HASH_GRPBYS_TOP

New monitor elements for columnar vector memory

Columnar vector memory is the sort heap memory that is used in the vector processing of data that is stored in column-organized tables. Starting in DB2 Cancun Release 10.5.0.4, the following new monitor elements allow you to monitor the amount of columnar vector memory used by sort consumers:

- ACTIVE_COL_VECTOR_CONSUMERS
- ACTIVE_COL_VECTOR_CONSUMERS_TOP

- POST_THRESHOLD_COL_VECTOR_CONSUMERS
- TOTAL_COL_VECTOR_CONSUMERS

Monitor interfaces to get request metrics that are aggregated along different dimensions

The following monitor interfaces report request metrics:

- MON_GET_DATABASE
- MON_GET_DATABASE_DETAILS
- MON_GET_WORKLOAD
- MON_GET_WORKLOAD_DETAILS
- MON_GET_UNIT_OF_WORK
- MON_GET_UNIT_OF_WORK_DETAILS
- MON_GET_SERVICE_SUBCLASS
- MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_CONNECTION
- MON_GET_CONNECTION_DETAILS
- EVMON_FORMAT_UE_TO_XML
- MON_FORMAT_XML_METRICS_BY_ROW
- Unit of work event monitor
- Statistics event monitor

All request metrics interfaces return the following new monitor elements:

- POOL_COL_L_READS
- POOL_TEMP_COL_L_READS
- POOL_COL_P_READS
- POOL_TEMP_COL_P_READS
- POOL_COL_LBP_PAGES_FOUND
- POOL_COL_WRITES
- POOL_COL_GBP_L_READS
- POOL_COL_GBP_P_READS
- POOL_COL_GBP_INVALID_PAGES
- POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP
- POOL_QUEUED_ASYNC_COL_REQS
- POOL_QUEUED_ASYNC_TEMP_COL_REQS
- POOL_QUEUED_ASYNC_COL_PAGES
- POOL_QUEUED_ASYNC_TEMP_COL_PAGES
- POOL_FAILED_ASYNC_COL_REQS
- POOL_FAILED_ASYNC_TEMP_COL_REQS
- TOTAL_COL_TIME
- TOTAL_COL_PROC_TIME
- TOTAL_COL_EXECUTIONS

The data type for each of these elements is BIGINT. These elements are reported when the REQUEST METRICS control is set to BASE. For the MON_GET_DATABASE and MON_GET_DATABASE_DETAILS interfaces, REQUEST METRICS controls only the collection of the TOTAL_COL_TIME, TOTAL_COL_PROC_TIME, and TOTAL_COL_EXECUTIONS elements. The other

elements are reported when the OBJECT METRICS control is set to BASE.

Monitor interfaces to get activity metrics

Activity metrics are a subset of request metrics that are measured during execution of an SQL statement.

The following monitor interfaces report activity metrics:

- MON_GET_ACTIVITY
- MON_GET_ACTIVITY_DETAILS
- MON_GET_PKG_CACHE_STMT
- MON_GET_PKG_CACHE_STMT_DETAILS
- EVMON_FORMAT_UE_TO_XML
- MON_FORMAT_XML_METRICS_BY_ROW
- Activity event monitor
- Package cache event monitor

All activity metrics interfaces return the following new monitor elements:

- POOL_COL_L_READS
- POOL_TEMP_COL_L_READS
- POOL_COL_P_READS
- POOL_TEMP_COL_P_READS
- POOL_COL_LBP_PAGES_FOUND
- POOL_COL_WRITES
- POOL_COL_GBP_L_READS
- POOL_COL_GBP_P_READS
- POOL_COL_GBP_INVALID_PAGES
- POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP
- POOL_QUEUED_ASYNC_COL_REQS
- POOL_QUEUED_ASYNC_TEMP_COL_REQS
- POOL_QUEUED_ASYNC_COL_PAGES
- POOL_QUEUED_ASYNC_TEMP_COL_PAGES
- POOL_FAILED_ASYNC_COL_REQS
- POOL_FAILED_ASYNC_TEMP_COL_REQS
- TOTAL_COL_TIME
- TOTAL_COL_PROC_TIME
- TOTAL_COL_EXECUTIONS

The data type for each of these elements is BIGINT. These elements are reported when the ACTIVITY METRICS control is set to BASE.

Monitor interfaces to get database object metrics

Object metrics interfaces report monitor elements for a specific database object or for an entire database.

- The MON_GET_DATABASE, MON_GET_DATABASE_DETAILS, MON_GET_BUFFERPOOL, and MON_GET_TABLESPACE table functions return the following new monitor elements:
 - POOL_COL_L_READS

- POOL_TEMP_COL_L_READS
- POOL_COL_P_READS
- POOL_TEMP_COL_P_READS
- POOL_COL_LBP_PAGES_FOUND
- POOL_COL_WRITES
- POOL_ASYNC_COL_READS
- POOL_ASYNC_COL_READ_REQS
- POOL_ASYNC_COL_WRITES
- POOL_ASYNC_COL_LBP_PAGES_FOUND
- POOL_COL_GBP_L_READS
- POOL_COL_GBP_P_READS
- POOL_COL_GBP_INVALID_PAGES
- POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP
- POOL_ASYNC_COL_GBP_L_READS
- POOL_ASYNC_COL_GBP_P_READS
- POOL_ASYNC_COL_GBP_INVALID_PAGES
- POOL_ASYNC_COL_GBP_INDEP_PAGES_FOUND_IN_LBP
- POOL_QUEUED_ASYNC_COL_REQS
- POOL_QUEUED_ASYNC_TEMP_COL_REQS
- POOL_QUEUED_ASYNC_COL_PAGES
- POOL_QUEUED_ASYNC_TEMP_COL_PAGES
- POOL_FAILED_ASYNC_COL_REQS
- POOL_FAILED_ASYNC_TEMP_COL_REQS
- SKIPPED_PREFETCH_COL_P_READS
- SKIPPED_PREFETCH_TEMP_COL_P_READS
- SKIPPED_PREFETCH_UOW_COL_P_READS
- SKIPPED_PREFETCH_UOW_TEMP_COL_P_READS

The data type for each of these elements is BIGINT. These elements are reported when the OBJECT METRICS control is set to BASE.

- The MON_GET_TABLE table function returns the following new monitor elements:
 - COL_OBJECT_L_PAGES
 - NUM_COLUMNS_REFERENCED
 - OBJECT_COL_L_READS
 - OBJECT_COL_P_READS
 - OBJECT_COL_GBP_L_READS
 - OBJECT_COL_GBP_P_READS
 - OBJECT_COL_GBP_INVALID_PAGES
 - OBJECT_COL_LBP_PAGES_FOUND
 - OBJECT_COL_GBP_INDEP_PAGES_FOUND_IN_LBP
 - SECTION_EXEC_WITH_COL_REFERENCES
 - TAB_ORGANIZATION

The data type for each of these elements is BIGINT. The TAB_ORGANIZATION and COL_OBJECT_L_PAGES elements are always reported. The other elements are reported when the OBJECT METRICS control is set to EXTENDED.

- The MON_GET_TABLE_USAGE_LIST table function returns the following new monitor elements:
 - OBJECT_COL_L_READS
 - OBJECT_COL_P_READS
 - OBJECT_COL_GBP_L_READS
 - OBJECT_COL_GBP_P_READS
 - OBJECT_COL_GBP_INVALID_PAGES
 - OBJECT_COL_LBP_PAGES_FOUND
 - OBJECT_COL_GBP_INDEP_PAGES_FOUND_IN_LBP

The data type for each of these elements is BIGINT. These elements are reported when the OBJECT METRICS control is set to EXTENDED.

- The ADMIN_GET_TAB_INFO table function returns the following new monitor elements:
 - COL_OBJECT_L_SIZE
 - COL_OBJECT_P_SIZE

Administrative view interfaces to get hit ratio metrics

Administrative views are predefined views that are built on top of monitor interfaces that perform common calculations on monitor data.

The MON_BP_UTILIZATION administrative view returns the following new monitor elements:

- COL_PHYSICAL_READS
- COL_HIT_RATIO_PERCENT
- GBP_COL_HIT_RATIO_PERCENT

The MON_TBSP_UTILIZATION administrative view returns the following new monitor elements:

- COL_PHYSICAL_READS
- COL_HIT_RATIO_PERCENT
- GBP_COL_HIT_RATIO_PERCENT

The MON_WORKLOAD_SUMMARY, MON_SERVICE_SUBCLASS_SUMMARY, MON_CONNECTION_SUMMARY, and MON_DB_SUMMARY administrative views return the following monitor elements, which are modified to take into account new COL pool metrics:

- TOTAL_BP_HIT_RATIO_PERCENT
- TOTAL_GBP_HIT_RATIO_PERCENT

Changed administrative procedures generate reports that contain monitor data

The following administrative procedures are updated to include COL pool monitor elements.

- MONREPORT.DBSUMMARY
- MONREPORT.CONNECTION

Changed monitor interface gets information about the agents that run within a particular service class

The MON_GET_AGENT function displays information about the agents that are running within a particular service class. The following changes were made to account for agents that are performing parallel query processing on column-organized tables:

- AGENT_TYPE: The SMPSUBAGENT type is returned for agents that are processing column-organized tables.
- AGENT_SUBTYPE: The new COLUMNAR subtype is returned to identify SMP agents that are processing column-organized tables.
- REQUEST_TYPE: Agents with the COLUMNAR subtype have the same behavior as agents with the DSS or SMP subtype. Specifically, if the subsection number is nonzero, the returned value is SUBSECTION:*subsection_number*; otherwise, the null value is returned.

Chapter 14. Explain information for column-organized tables

Explain information is captured to support column-organized tables. Use this information to determine how your application performs when it uses these tables.

The CTQ plan operator represents the transition between column-organized data processing and row-organized data processing.

The steps that you use to capture the explain information for column-organized tables are the same steps that you use for running queries against row-organized tables.

- Set the EXPLAIN mode on by using the CURRENT EXPLAIN MODE special register as follows:

```
db2 SET CURRENT EXPLAIN MODE YES
```

- Issue your query against column-organized tables.
- Issue the **db2exfmt** command to format the contents of the explain tables and obtain the access plan. The following example shows you how to use this command against the SAMPLE database:

```
db2exfmt -d sample -1 -o output.exfmt
```

In Version 10.5 Fix Pack 2 and later fix packs, the performance of a select, update, or delete operation that affects only one row in a column-organized table can be improved if the table has uniqueness constraints or a primary key constraint, because the query optimizer can use an index scan instead of a full table scan. Index access to column-organized data that requires the retrieval of additional columns or the application of additional predicates is shown in the following example plan:

```
      Rows
      RETURN
      ( 1)
      Cost
      I/O
      |
      1
      CTQ
      ( 2)
      41.3466
      6
      |
      1
      NLJOIN
      ( 3)
      41.3449
      6
      /-----\
      1          1
      CTQ      TBSCAN
      ( 4)      ( 6)
      6.91242   34.4325
      1          5
      |          |
      1          98168
      IXSCAN   CO-TABLE: VICCHANG
      ( 5)      /BIC/SZCCUST
      6.91242   Q1
      1
      |
```

98168
INDEX: VICCHANG
/BIC/SZCCUST~0
Q1

This plan is equivalent to a FETCH-IXSCAN combination that is used to access row-organized data. For index access to column-organized data, row-organized data processing retrieves the rowid from the index by using IXSCAN(5) and passes it to column-organized data processing using CTQ(4). CTQ(4) represents a column-organized table queue that passes data from row-organized data processing to column-organized data processing. TBSCAN(6) locates the columns that are identified by the rowid. TBSCAN(6) might apply additional predicates if necessary, or reapply the IXSCAN predicates in some situations. Specifically, if the table is being accessed under the UR isolation level, or the access is in support of an update or delete operation, the TBSCAN needs to apply only those predicates that were not already applied by the IXSCAN. Otherwise, the TBSCAN needs to reapply all of the IXSCAN predicates. NLJOIN(3) represents the process of retrieving the rowid from row-organized data processing and passing it to the column-organized TBSCAN.

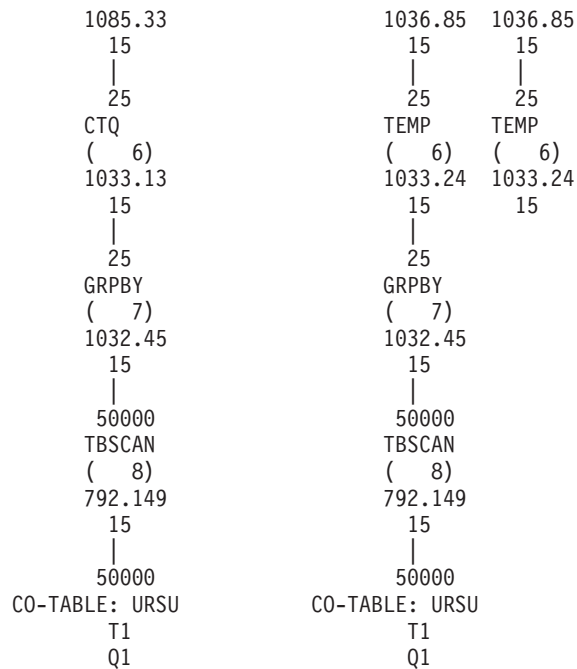
A *common table expression* defines the result of a table that you can specify in the FROM clause of an SQL statement. In DB2 Cancun Release 10.5.0.4 and later fix packs, statements with common table expressions against column-organized tables can have more efficient execution plans.

Consider the following query against a column-organized table T1 that has two columns, C1 and C2:

```
WITH cse(c1,c2) AS (SELECT t1.c1, MAX(c2) FROM t1 GROUP BY t1.c1)
SELECT a.c2 FROM cse a, cse b
WHERE a.c1 = b.c2;
```

The following sample execution plans correspond to that query. The plan on the left is for a query that is run in Version 10.5 Fix Pack 3 or earlier, and the plan on the right is for a query that is run in DB2 Cancun Release 10.5.0.4.

<pre> Rows RETURN (1) Cost I/O 25 HSJOIN^ (2) 2181.6 30 /---\ 25 25 TBSCAN TBSCAN (3) (9) 1090.59 1090.59 15 15 25 25 TEMP TEMP (4) (4) 1086.99 1086.99 15 15 25 25 LTQ LTQ (5) (5)</pre>	<pre> Rows RETURN (1) Cost I/O 25 LTQ (2) 2078.07 30 25 CTQ (3) 2074 30 25 ^HSJOIN (4) 2073.79 30 /---\ 25 25 TBSCAN TBSCAN (5) (9)</pre>
---	--



- The execution plan prior to DB2 Cancun Release 10.5.0.4 includes the CTQ(6) operator, which sends the results of the common table expression from the subsection that is processing column-organized data to the subsection that is processing row-organized data. In that section, the results are materialized by the TEMP(4) operator. Operators TBSCAN(3) and TBSCAN(9) scan the output of the TEMP(4) operator and send the data to the HSJN(2) operator.
- The DB2 Cancun Release execution plan includes the TEMP(6) operator, which materializes the results of the common table expression during column-organized data processing. Operators TBSCAN(5) and TBSCAN(9) scan the output of the TEMP(6) operator and send the data to the HSJN(4) operator. Afterward, the CTQ(3) operator sends the results of the join operation from column-organized data processing to row-organized data processing.

In DB2 Cancun Release 10.5.0.4 and later fix packs, index access for SELECT statements that are run with isolation level CS (cursor stability) is supported. An extra runtime optimization is available to improve the performance of column-organized UR and CS index scans when the data does not require column-organized processing. In some situations, it is more efficient for the access to be done by using row-organized processing because this approach avoids the overhead of switching between column-organized and row-organized formats. This additional optimization, in which index access is performed by using row-organized data processing, is not possible if all the following conditions apply:

- The index access occurs directly on the inner of a nested-loop join (NLJOIN).
- There are join predicates to be applied by the index scan.
- The join occurs between tables in the same subselect.

Index-only access to a column-organized table with isolation level CS is not supported.

The query that is shown in the following example qualifies for these additional optimizations.

```
SELECT * FROM T1 WHERE T1.PK = ?
```

The **db2exfmt** output shows an example of a column-organized table that is accessed by using an index with isolation level CS. This query runs by using row-organized processing because all the predicates can be applied at the index scan and the table is not being joined to any other table.

```

      Rows
      RETURN
      ( 1)
      Cost
      I/O
      |
      1
      CTQ
      ( 2)
      43.9788
      6
      |
      1
      NLJOIN
      ( 3)
      43.7775
      6
      /-----\
      1           1
      CTQ         TBSCAN
      ( 4)         ( 6)
      9.10425      34.6733
      1           5
      |           |
      1           1000
      IXSCAN      CO-TABLE: BLUUSER
      ( 5)         T1
      9.10425      Q1
      1
      |
      1000
      INDEX: BLUUSER
      PK
      Q1

```

TQ operator

This operator represents a table queue.

Operator name: TQ

Represents: A table queue that is used to pass table data from one database agent to another when multiple database agents are processing a query. Multiple database agents are used to process a query when parallelism is involved.

The CTQ operator represents a boundary within the DB2 query engine. Operators that appear below the boundary process data as compressed column-organized vectors and tuples, whereas operators that are above the boundary operate on tuples that are not encoded.

Example for CTQ operator

This example is based on the following query for the TPCDS.CUSTOMER and TPCDS.STORE_SALES column-organized tables:

```
set current explain mode yes;
```

```
SELECT
  C_BIRTH_COUNTRY,
```



```

SUM(CAST(STORE_SALES.SS_QUANTITY AS DECIMAL(31, 0))) AS C2
FROM
  TPCDS.CUSTOMER CUSTOMER
  INNER JOIN TPCDS.STORE_SALES STORE_SALES
  ON CUSTOMER.C_CUSTOMER_SK = STORE_SALES.SS_CUSTOMER_SK
GROUP BY
  C_BIRTH_COUNTRY;

db2exfmt -d mydb -l -o output.exfmt;

```

Partial output of the **db2exfmt** command is as follows:

Access Plan:

```

-----
Total Cost:  570246
Query Degree: 32

      Rows
      RETURN
      (  1)
      Cost
      I/O
      |
      191
      LTQ
      (  2)
      570246
      505856
      |
      191
      CTQ
      (  3)
      570245
      505856
      |
      191
      GRPBY
      (  4)
      570245
      505856
      |
      2.87997e+08
      ^HSJOIN
      (  5)
      377881
      505856
      /-----\
      2.87997e+08      2e+06
      TBSCAN          TBSCAN
      (  6)            (  7)
      329484          4080.56
      499008          6848
      |              |
      2.87997e+08      2e+06
      CO-TABLE: TPCDS  CO-TABLE: TPCDS
      STORE_SALES      CUSTOMER
      Q1               Q2

```

Operator Symbols :

```

-----
Symbol      Description
-----
>JOIN       : Left outer join
JOIN<       : Right outer join
>JOIN<      : Full outer join

```

```

xJOIN      : Left antijoin
JOINx      : Right antijoin
^JOIN      : Left early out
JOIN^      : Right early out
ATQ        : Asynchrony
BTQ        : Broadcast
CTQ        : Column-Organized Data
DTQ        : Directed
LTQ        : Intra-partition parallelism
MTQ        : Merging (sorted)
STQ        : Scatter
XTQ        : XML aggregation
TQ*        : Listener

```

```

3) TQ      : (Table Queue)
Cumulative Total Cost: 570245
Cumulative CPU Cost: 7.63257e+11
Cumulative I/O Cost: 505856
Cumulative Re-Total Cost: 377881
Cumulative Re-CPU Cost: 2.82348e+11
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 570245
Estimated Bufferpool Buffers: 0

```

Arguments:

```

-----
LISTENER: (Listener Table Queue type)
FALSE
    SCANTYPE: (Intra-Partition Parallelism Scan Type)
    LOCAL PARALLEL
    TQDEGREE: (Degree of Intra-Partition parallelism)
    8
TQMERGE : (Merging Table Queue flag)
FALSE
TQORIGIN: (Table Queue Origin type)
COLUMN-ORGANIZED DATA
TQREAD : (Table Queue Read type)
READ AHEAD
UNIQUE : (Uniqueness required flag)
FALSE

```

Input Streams:

```

-----
8) From Operator #4

Estimated number of rows: 191
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:
-----
+Q5.C2+Q5.C_BIRTH_COUNTRY

```

Output Streams:

```

-----
9) To Operator #2

Estimated number of rows: 191
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:
-----
+Q5.C2+Q5.C_BIRTH_COUNTRY

```

```

4) GRPBY : (Group By)
Cumulative Total Cost: 570245
Cumulative CPU Cost: 7.63257e+11
Cumulative I/O Cost: 505856
Cumulative Re-Total Cost: 377881
Cumulative Re-CPU Cost: 2.82348e+11
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 570245
Estimated Bufferpool Buffers: 0

Arguments:
-----
AGGMODE : (Aggregation Mode)
HASHED COMPLETE
GROUPBYC: (Group By columns)
TRUE
GROUPBYN: (Number of Group By columns)
1
GROUPBYR: (Group By requirement)
1: Q3.C_BIRTH_COUNTRY
ONEFETCH: (One Fetch flag)
FALSE

Input Streams:
-----
7) From Operator #5

Estimated number of rows: 191
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:
-----
+Q3.C_BIRTH_COUNTRY(A)+Q3.SS_QUANTITY

Output Streams:
-----
8) To Operator #3

Estimated number of rows: 191
Number of columns: 2
Subquery predicate ID: Not Applicable

Column Names:
-----
+Q5.C2+Q5.C_B

5) HSJOIN: (Hash Join)
Cumulative Total Cost: 377881
Cumulative CPU Cost: 2.82348e+11
Cumulative I/O Cost: 505856
Cumulative Re-Total Cost: 377881
Cumulative Re-CPU Cost: 2.82348e+11
Cumulative Re-I/O Cost: 505856
Cumulative First Row Cost: 377881
Estimated Bufferpool Buffers: 499008

Arguments:
-----
BITFLTR : (Hash Join Bit Filter used)
FALSE
EARLYOUT: (Early Out flag)
LEFT
HASHCODE: (Hash Code Size)
24 BIT

```

```

HASHTBSZ: (Number of hash table entries)
2000001
SEMIJOIN: (Semi-join flag)
FALSE
TEMPSIZE: (Temporary Table Page Size)
32768
TUPBLKSZ: (Tuple Block Size (bytes))
16000

```

Predicates:

```

-----
3) Predicate used in Join,
Comparison Operator: Equal (=)
Subquery Input Required: No
Filter Factor: 5e-07

```

Predicate Text:

```

-----
(Q2.C_CUSTOMER_SK = Q1.SS_CUSTOMER_SK)

```

Input Streams:

```

-----
2) From Operator #6

Estimated number of rows: 2.87997e+08
Number of columns: 2
Subquery predicate ID: Not Applicable

```

Column Names:

```

-----
+Q1.SS_QUANTITY+Q1.SS_CUSTOMER_SK

```

4) From Operator #7

```

Estimated number of rows: 2e+06
Number of columns: 2
Subquery predicate ID: Not Applicable

```

Column Names:

```

-----
+Q2.C_BIRTH_COUNTRY+Q2.C_CUSTOMER_SK

```

Output Streams:

```

-----
5) To Operator #4

Estimated number of rows: 2.87997e+08
Number of columns: 2
Subquery predicate ID: Not Applicable

```

Column Names:

```

-----
+Q3.SS_QUANTITY+Q3.C_BIRTH_COUNTRY

```

The following text is an extract of the explain information about the two column-organized tables that are used in the query example:

```

Schema: TPCDS
Name: CUSTOMER
Type: Column-Organized Table
Time of creation: 2012-09-13-15.44.51.792706
Last statistics update: 2012-09-13-17.50.36.506577
Number of columns: 18
Number of rows: 2000001

```

Width of rows: 64
Number of buffer pool pages: 6848
Number of data partitions: 1
Distinct row values: No
Tablespace name: TS_PD_DATA_001
Tablespace overhead: 3.630000
Tablespace transfer rate: 0.070000
Source for statistics: Single Node
Prefetch page count: 256
Container extent page count: 32
Table overflow record count: -9
Table Active Blocks: -1
Average Row Compression Ratio: -1
Percentage Rows Compressed: -1
Average Compressed Row Size: -1

Schema: TPCDS

Name: STORE_SALES

Type: Column-Organized Table

Time of creation: 2012-09-13-15.44.53.243967
Last statistics update: 2012-09-13-18.27.24.035212
Number of columns: 23
Number of rows: 287997024
Width of rows: 71
Number of buffer pool pages: 499008
Number of data partitions: 1
Distinct row values: No
Tablespace name: TS_PD_DATA_001
Tablespace overhead: 3.630000
Tablespace transfer rate: 0.070000
Source for statistics: Single Node
Prefetch page count: 256
Container extent page count: 32
Table overflow record count: -9
Table Active Blocks: -1
Average Row Compression Ratio: -1
Percentage Rows Compressed: -1
Average Compressed Row Size: -1

Chapter 15. Monitoring routines using table functions

You can use table functions to retrieve information about routines.

Table functions can be used to monitor routines and provide the following information:

- Aggregated metrics that report the total cost of the routine. Metrics are aggregated across all invocations of the routine and include metrics for all child statements and requests that are executed by the routine.
- Lists of statements that are executed by routines that assist you in drilling down and problem determination
- Lists of routines that might be invoked by a statement that aid you in performing additional drill downs on routine-related details.

Use the following monitor functions to access information about routines:

- MON_GET_ROUTINE
- MON_GET_ROUTINE_DETAILS
- MON_GET_ROUTINE_EXEC_LIST
- MON_GET_SECTION_ROUTINE

MON_GET_ROUTINE table function - get aggregated routine execution metrics

The MON_GET_ROUTINE table function returns aggregated execution metrics for procedures, external procedures, compiled functions, external functions, compiled triggers, and anonymous blocks invoked since the database was activated.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Usage

Use the MON_GET_ROUTINE table function to identify the most expensive routines on the database server.

This table function returns one row of metrics for each routine or trigger and each member that matches the input arguments. Input argument values are complementary. Metrics returned are aggregates of all executions of the routine on that member. No aggregation across members is performed. However, an aggregation across members is possible through SQL queries (as shown in the Examples section). If a routine executes subroutines, the work done in the subroutines is included in the metrics of the parent routine.

Routine monitoring data collection must be explicitly enabled using the `mon_rtn_data` database configuration parameter. If this configuration parameter is set to `NONE`, no information is returned.

The counters and time-spent monitor elements returned by this table function are controlled with the `COLLECT REQUEST METRICS` clause on service classes and the `mon_req_metrics` database configuration parameter at the database level. If neither control is enabled, the counters and time-spent monitor elements reported are 0.

When the package for a dynamically prepared compound SQL statements is removed from the package cache, information for this routine is no longer reported by `MON_GET_ROUTINE` function. Similarly, when a routine or trigger is dropped, information about the routine or trigger is no longer reported.

Any routines that were not executed during the previous 24 hour period are pruned from memory and not returned.

For a complete list of returned information, see `MON_GET_ROUTINE` - get aggregated routine execution metrics.

Examples

1. List the highest CPU consuming routines of any type in module MOD1.

```
SELECT ROUTINE_TYPE, ROUTINE_SCHEMA, ROUTINE_NAME, SPECIFIC_NAME, TOTAL_CPU_TIME
FROM TABLE(MON_GET_ROUTINE(NULL, NULL, 'MOD1', NULL, NULL))
AS T ORDER BY TOTAL_CPU_TIME DESC
```

returns

ROUTINE_TYPE	ROUTINE_SCHEMA	ROUTINE_NAME	SPECIFIC_NAME	TOTAL_CPU_TIME
F	DRICARD	FUNCTION3	FUNCTION3	19425
F	DRICARD	FUNCTION4	FUNCTION4	5780
P	DRICARD	P1	SQL120801142627900	4685
C	SYSIBMINTERNAL	COMPILED_ANON_BLOCK_INVOKE	SQL120801153841490	3471
P	SYSPROC	SYSINSTALLOBJECTS	SYSINSTALLOBJECTS	1158
F	DRICARD	FUNCTION1	FUNCTION1	2632
F	DRICARD	FUNCTION2	FUNCTION2	2029

7 record(s) selected.

2. List aggregate monitor data for all stored procedures.


```
SELECT * FROM TABLE(MON_GET_ROUTINE('P', NULL, NULL, NULL, -2)) AS T
```
3. List aggregate monitor data for routines of all types.


```
SELECT * FROM TABLE(MON_GET_ROUTINE(NULL, NULL, NULL, NULL, -2)) AS T
```
4. List aggregate monitor data for all procedures named PROC1 in schema TEST.


```
SELECT * FROM TABLE(MON_GET_ROUTINE('P', 'TEST', NULL, 'PROC1', -2)) AS T
```
5. List aggregate monitor data for overloaded procedure PROC1 with specific name PROC1_OVERLOAD in schema TEST.


```
SELECT * FROM TABLE(MON_GET_ROUTINE('SP', 'TEST', NULL, 'PROC1_OVERLOAD', -2))
AS T
```
6. List aggregate monitor data for the anonymous block with executable id x'010000000000000052010000000000001000000010020120822205618607103'.


```
SELECT * FROM TABLE(MON_GET_ROUTINE
('A', 'MYSCHEMA', NULL, 'SQL181500027522310', -1)) AS T
WHERE
DYN_COMPOUND_EXEC_ID = x'010000000000000052010000000000001000000010020120822205618607103'
```

Chapter 16. Monitoring information for HADR

Use the new information to improve HADR monitoring. Several new columns were added to the MON_GET_HADR table function and the information returned by the -hadr parameter of the **db2pd** command.

You can use the following new columns to monitor the health of your HADR databases:

- STANDBY_SPOOL_PERCENT lets you know how much of your HADR log spooling space is being used.
- HADR_FLAGS contains a subset of fields that indicate when assisted remote catchup is occurring or if there are issues with log receipt or retrieval on the standby.

Starting in DB2 Cancun Release 10.5.0.4, to enhance your ability to determine whether there are errors or communication problems affecting your HADR deployment, the following columns were also added to the MON_GET_HADR table function and the information returned by the **db2pd** command:

- HEARTBEAT_MISSED and HEARTBEAT_EXPECTED allow you to determine whether the accumulated number of missed heartbeats is a cause for concern.
- STANDBY_ERROR_TIME indicates the last time the standby had a major error.

MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information

This function returns high availability disaster recovery (HADR) monitoring information.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Usage

HADR pair view

Certain fields are applicable to primary or standby only. For example, PEER_WAIT_LIMIT is applicable only to primary, STANDBY_RECV_BUF_SIZE, STANDBY_SPOOL_LIMIT, READS_ON_STANDBY_ENABLED are applicable only to standby. When this kind of information is reported, the database currently in the role is used (which may be the remote database), rather than the local database. For example, PEER_WAIT_LIMIT seen on a standby database is the value configured on the primary database, not the local config of standby database (which will be used only when the standby turns into primary).

Information about remote database

Primary and standby exchange monitoring information via heartbeat messages. Therefore information about the remote database can be slightly out of date. See heartbeat interval (reported in table function) to estimate timeliness of information (network latency can add additional delay). If a database has never connected to its partner database since activation, information about remote database may be returned as SQL NULL to indicate "unknown".

Log shipping channel end points

The end points for a log shipping channel is uniquely identified by host, instance and member:

- Primary side: PRIMARY_MEMBER_HOST, PRIMARY_INSTANCE, PRIMARY_MEMBER
- Standby side: STANDBY_MEMBER_HOST, STANDBY_INSTANCE, STANDBY_MEMBER

Until a connection is made, end point information of remote end may not be available. When information not available, empty strings will be returned for host and instance names and zero returned for member ID. In addition when in a DB2 Enterprise Server Edition environment, 0 is always returned for the member ID.

Note on unit of time duration

Per monitor table function convention, all MON_GET_HADR time duration fields use milliseconds as unit. For those fields reflecting a configuration parameter (such as HADR_TIMEOUT, HADR_PEER_WINDOW) whose unit in configuration is seconds, the number returned by MON_GET_HADR table function will be different from the number used in **db2 get/update db cfg** command, and the number returned by SYSIBMADM.DBCFG admin view or SYSPROC.DB_GET_CFG() table function. For example, for a 60 second HADR_TIMEOUT value, MON_GET_HADR will return 60000, while the configuration oriented interfaces will return 60. To convert the millisecond number to second, use column_name/1000 in your query.

Usage during takeover

During takeover, there may be a period when clients cannot connect to either primary or standby database. The recommended monitoring method during takeover is **db2pd -hadr**.

Column Order and Groups:

1. Cluster level summary: HADR_ROLE, REPLAY_TYPE, HADR_SYNCMODE.
2. Log stream level summary: STANDBY_ID, LOG_STREAM_ID, HADR_STATE
3. Log shipping channel end points:
 - a. Primary side: PRIMARY_MEMBER_HOST, PRIMARY_INSTANCE, PRIMARY_MEMBER
 - b. Standby side: STANDBY_MEMBER_HOST, STANDBY_INSTANCE, STANDBY_MEMBER

The end points uniquely identify an HADR log shipping channel in all scenarios. Host, instance or MEMBER_ID uniquely identifies a member.
4. Connection details:
 - a. Status: HADR_CONNECT_STATUS, HADR_CONNECT_STATUS_TIME
 - b. Network timing: HEARTBEAT_INTERVAL, HADR_TIMEOUT, TIME_SINCE_LAST_RECV

- c. Logger wait timing: PEER_WAIT_LIMIT, LOG_HADR_WAIT_CUR, LOG_HADR_WAIT_TIME, LOG_HADR_WAITS_TOTAL
 - d. TCP buffer size: SOCK_SEND_BUF_REQUESTED, SOCK_SEND_BUF_ACTUAL, SOCK_RECV_BUF_REQUESTED, SOCK_RECV_BUF_ACTUAL
5. Log position details:
 - a. Primary log position: PRIMARY_LOG_FILE, PRIMARY_LOG_PAGE, PRIMARY_LOG_POS, PRIMARY_LOG_TIME
 - b. Standby log receive position: STANDBY_LOG_FILE, STANDBY_LOG_PAGE, STANDBY_LOG_POS, STANDBY_LOG_TIME
 - c. Primary-standby log gap: HADR_LOG_GAP
 - d. Standby log replay position: STANDBY_REPLAY_LOG_FILE, STANDBY_REPLAY_LOG_PAGE, STANDBY_REPLAY_LOG_POS, STANDBY_REPLAY_LOG_TIME
 - e. Standby receive-replay gap: STANDBY_RECV_REPLAY_GAP
 - f. Replay delay: STANDBY_REPLAY_DELAY
 6. Log buffer and spooling: STANDBY_RECV_BUF_SIZE, STANDBY_RECV_BUF_PERCENT, STANDBY_SPOOL_LIMIT
 7. Peer window: PEER_WINDOW, PEER_WINDOW_END
 8. Takeover: TAKEOVER_APP_REMAINING_PRIMARY, TAKEOVER_APP_REMAINING_STANDBY
 9. Reads on Standby: READS_ON_STANDBY_ENABLED, STANDBY_REPLAY_ONLY_WINDOW_ACTIVE, STANDBY_REPLAY_ONLY_WINDOW_START, STANDBY_REPLAY_ONLY_WINDOW_TRAN_COUNT

For a complete list of returned information, see MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information.

Examples

1.

```
SELECT HADR_ROLE, STANDBY_ID, HADR_STATE, varchar(PRIMARY_MEMBER_HOST ,20)
      as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST ,20)
      as STANDBY_MEMBER_HOST from table(MON_GET_HADR(NULL))
```

The following is an example of output from this query.

HADR_ROLE	STANDBY_ID	HADR_STATE	PRIMARY_MEMBER_HOST
PRIMARY	1	PEER	hostP.ibm.com
PRIMARY	2	REMOTE_CATCHUP	hostP.ibm.com
PRIMARY	3	REMOTE_CATCHUP	hostP.ibm.com

STANDBY_MEMBER_HOST
hostS1.ibm.com
hostS2.ibm.com
hostS3.ibm.com

3 record(s) selected.

Query is issued to a primary database with 3 standbys in which 3 rows are returned. Each row represents a primary-standby log shipping channel. The HADR_ROLE column represents the role of the database to which the query is issued. Therefore it is PRIMARY on all rows.

2.

```
SELECT HADR_ROLE, STANDBY_ID, HADR_STATE, varchar(PRIMARY_MEMBER_HOST ,20)
      as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST ,20)
      as STANDBY_MEMBER_HOST from table(MON_GET_HADR(NULL))
```

The following is an example of output from this query.

HADR_ROLE	STANDBY_ID	HADR_STATE	PRIMARY_MEMBER_HOST
STANDBY	0	PEER	hostP.ibm.com

STANDBY_MEMBER_HOST
hostS1.ibm.com

1 record(s) selected.

Query is issued to a standby database with reads on standby enabled. Standby only knows about its own primary. Only one row is returned even if the standby is part of a multiple standby system. STANDBY_ID is always zero when query is issued to a standby.

- The following example is for HADR in a DB2 pureScale environment only. The environment for this example has three member clusters. On the primary, member 0 is assisting member 1. Standby member 0 is the replay member.

- With member ID 0, the HADR information for log stream 0, and the assisted log stream (log stream 1) is output:

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(0))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
0	0	0	PEER	
1	0	0	REMOTE_CATCHUP	ASSISTED_REMOTE_CATCHUP

- With member ID 1, the HADR information for log steam 1 is output:

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(1))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
1	1	0	DISCONNECTED	

- For log stream ID 2, the HADR information for log stream 2 is output:

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(2))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
2	2	0	PEER	

- When specifying -2 as member ID, HADR information on all log streams is reported. For assisted remote catchup, only the record from the assisting member is returned, and HADR_FLAGS indicates that it is in an assisted remote catchup state.

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(-2))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
0	0	0	PEER	
1	0	0	REMOTE_CATCHUP	ASSISTED_REMOTE_CATCHUP
2	2	0	PEER	

db2pd - Monitor and troubleshoot DB2 database

Retrieves information from the DB2 database system memory sets.

Authorization

One of the following authority levels is required:

- The SYSADM authority level.

- The SYSCTRL authority level.
- The SYSMANT authority level.
- The SYSMON authority level.

When the SYSMON authority level is granted, the following options are not available:

- **dump**
- **memblocks**
- **stack**

Note: On Windows, you must have administrator authority to use the **db2pd** command.

Required connection

There is no minimum connection requirement. However, if a database scope option is specified, that database must be active before the command can return the requested information.

Usage notes

The output produced for the **-hadr** parameter is as follows:

-hadr parameter

For the **-hadr** parameter, information related to high availability disaster recovery is returned. The information returned by this command depends on where it is issued:

- If it's issued from a standby, the command returns information about that standby and the primary only.
- If it's issued from a primary, the command returns information about the primary and all of the standbys.

In a DB2 pureScale environment, the command returns HADR information about log streams being processed by the local member. On a standby, if the command is issued on the replay member, it returns HADR information about all log streams, otherwise, it returns a message indicating that the database is not active on that member and no HADR information. If you use the **-allmembers** option, it returns the concatenated output from all members. This is one way to tell which member is the replay member. The other way is to look at **STANDBY_MEMBER** field from primary's monitoring output. If it's issued from a member on the primary, the command returns information about the stream owned by the member and all streams being assisted by the member. To see all of the streams in the cluster, issue the command with the **-allmembers** option.

Only information relevant to the current settings is shown, so for example if reads on standby is not enabled, information about the replay-only window is not shown.

HADR_ROLE

The current HADR role of the local database. Possible values are:

- PRIMARY
- STANDBY

REPLAY_TYPE

The type of HADR replication of the database. The possible value is:

- PHYSICAL

HADR_SYNCMODE

The current HADR synchronization mode of the local database. Possible values are:

- ASYNC
- NEARSYNC
- SUPERASYNC
- SYNC

Note: The HADR_SYNCMODE value of the standby is shown as an empty string (a zero-length string) until the primary connects to the standby database.

STANDBY_ID

The identifier for all the standbys in the current setup. This value has meaning only when the command is issued on the primary. If you issue it on a standby, it always returns 0 because standbys are not visible to each other. The 1 identifier is always assigned to the standby if there is only one standby. If you have multiple standbys in your setup, 1 indicates the principal standby.

LOG_STREAM_ID

The identifier for the log stream that is being shipped from the primary database.

HADR_STATE

The current HADR state of the database. Possible values are:

- DISCONNECTED
- DISCONNECTED_PEER
- LOCAL_CATCHUP
- PEER
- REMOTE_CATCHUP
- REMOTE_CATCHUP_PENDING

HADR_FLAGS

A string of one or more of the following flags indicating HADR condition:

ASSISTED_REMOTE_CATCHUP

The stream is in assisted remote catchup.

ASSISTED_MEMBER_ACTIVE

During assisted remote catchup, the member on the primary that is being assisted is active. This is an abnormal condition because an active member is expected to connect to standby directly.

STANDBY_LOG_RETRIEVAL

The standby database is interacting with the log archive device to retrieve log files.

STANDBY_RECV_BLOCKED

The standby temporarily cannot receive more logs. Possible causes are:

- When log spooling is disabled, the log receive buffer is full (STANDBY_RECV_BUF_PERCENT is 100%).

- When log spooling is enabled, spooling has reached the spool limit (STANDBY_SPOOL_PERCENT is 100%).
- The standby log device is full (indicated by the STANDBY_LOG_DEVICE_FULL flag). This condition can happen when spooling is enabled or disabled.

In all of these cases, after replay makes progress, more space is released and log receive can resume. .

STANDBY_LOG_DEVICE_FULL

The standby log device is full. This condition blocks log receive until some more space is released as replay proceeds.

STANDBY_REPLAY_NOT_ON_PREFERRED

The current replay member on the standby is not the preferred replay member.

PRIMARY_MEMBER_HOST

The local host, indicated by the **hadr_local_host** configuration parameter, of the member on the primary that is processing the log stream.

PRIMARY_INSTANCE

The instance name of the primary database processing the log stream.

PRIMARY_MEMBER

The member on the primary that is processing the log stream.

STANDBY_MEMBER_HOST

The local host, indicated by the **hadr_local_host** configuration parameter, of the standby member processing the log stream.

STANDBY_INSTANCE

The instance name of the standby database processing the log stream.

STANDBY_MEMBER

The standby member processing the log stream.

HADR_CONNECT_STATUS

The current HADR connection status of the database. Possible values are:

- CONGESTED
- CONNECTED
- DISCONNECTED

HADR_CONNECT_STATUS_TIME

The time when the current HADR connection status began. Depending on the HADR_CONNECT_STATUS value, the HADR_CONNECT_STATUS_TIME value indicates:

- Congestion start time
- Connection start time
- Disconnection time

HEARTBEAT_INTERVAL

The heartbeat interval in seconds, which is computed from various factors such as the values of the **hadr_timeout** and **hadr_peer_window** configuration parameters. The HEARTBEAT_INTERVAL element indicates how often the primary and standby exchange monitor information.

HEARTBEAT_MISSED

Number of heartbeat messages not received on time on this log stream. Messages start accumulating when a database is started on the local member. This number should be viewed relative to the HEARTBEAT_EXPECTED

value. For example, 100 missed heartbeats when HEARTBEAT_EXPECTED is 1000 is a 10% miss rate. This miss rate indicates a network problem. However, 100 missed heartbeats when HEARTBEAT_EXPECTED is 10000 is a 1% miss rate and is unlikely to be a network issue. Take the HEARTBEAT_INTERVAL value into account when assessing the HEARTBEAT_EXPECTED value. A short HEARTBEAT_INTERVAL value can cause the HEARTBEAT_MISSED value to appear high even though it is safe.

HEARTBEAT_EXPECTED

Number of heartbeat messages expected on this log stream. These messages accumulate when a database is started on the local member. With the HEARTBEAT_MISSED value, you can determine the health of a network for a given time duration.

HADR_TIMEOUT

The time, in seconds, by which an HADR database must receive a message from its partner database. After this period of time, an HADR database server considers that the connection between the databases has failed and disconnects.

TIME_SINCE_LAST_RECV

The time, in seconds, that has elapsed since the last message was received, so the larger the number, the longer the delay in message delivery. When the TIME_SINCE_LAST_RECV value equals the HADR_TIMEOUT value, the connection between the databases is closed.

PEER_WAIT_LIMIT

The length of time, in seconds, that the primary database waits before breaking out of peer state if logging is blocked waiting for HADR log shipping to the standby. A value of 0 indicates no timeout.

LOG_HADR_WAIT_CUR

The length of time, in seconds, that the logger has been waiting on an HADR log shipping request. A value of 0 is returned if the logger is not waiting. When the wait time reaches the value that is returned in the PEER_WAIT_LIMIT field, HADR breaks out of peer state to unblock the primary database.

LOG_HADR_WAIT_RECENT_AVG

The average time, in seconds, for each log flush.

LOG_HADR_WAIT_ACCUMULATED

The accumulated time, in seconds, that the logger has spent waiting for HADR to ship logs.

LOG_HADR_WAIT_COUNT

The total count of HADR wait events in the logger. The count is incremented every time the logger initiates a wait on HADR log shipping, even if the wait returns immediately. As a result, this count is effectively the number of log flushes while the databases are in peer state.

SOCK_SEND_BUF_REQUESTED,ACTUAL

- The requested socket send buffer size (SOCK_SEND_BUF_REQUESTED), in bytes. A value of 0 indicates no request (the system default is used).
- The actual socket send buffer size (SOCK_SEND_BUF_ACTUAL), in bytes.

SOCK_RECV_BUF_REQUESTED,ACTUAL

- The requested socket receive buffer size (SOCK_RECV_BUF_REQUESTED), in bytes. A value of 0 indicates no request (the system default is used).
- The actual socket receive buffer size (SOCK_RECV_BUF_ACTUAL), in bytes.

PRIMARY_LOG_FILE, PAGE, POS

- The name of the current log file of the log stream on the primary database (PRIMARY_LOG_FILE).
- The page number in the current log file indicating the current log position on the primary HADR database. The page number is relative to its position in the log file. For example, page 0 is the beginning of the file (PRIMARY_LOG_PAGE).
- The current receive log position (byte offset) of the log stream on the primary database (PRIMARY_LOG_POS).

STANDBY_LOG_FILE, PAGE, POS

- The name of the log file corresponding to the standby receive log position on the log stream (STANDBY_LOG_FILE).
- The page number (relative to its position in the log file) corresponding to the standby receive log position (STANDBY_LOG_PAGE).
- The current log position of the standby HADR database (STANDBY_LOG_POS).

HADR_LOG_GAP

The running average, in bytes, of the gap between the PRIMARY_LOG_POS value and STANDBY_LOG_POS value.

STANDBY_REPLAY_LOG_FILE, PAGE, POS

- The name of the log file corresponding to the standby replay log position on the log stream (STANDBY_REPLAY_LOG_FILE).
- The page number in the standby replay log file corresponding to the standby replay log position (STANDBY_REPLAY_LOG_PAGE). The page number is relative to its position in the log file. For example, page 0 is the beginning of the file.
- The byte offset of the standby replay log position on the log stream (STANDBY_REPLAY_LOG_POS).

STANDBY_RECV_REPLAY_GAP

The average, in bytes, of the gap between the standby log receive position and the standby log replay position. If the value of this gap reaches the combined value of the standby's receive buffer size and the standby's spool limit, the standby stops receiving logs and blocks the primary if it is in peer state.

PRIMARY_LOG_TIME

The latest transaction timestamp of the log stream on the primary database.

STANDBY_LOG_TIME

The latest transaction timestamp of received logs on the log stream on the standby database.

STANDBY_REPLAY_LOG_TIME

The transaction timestamp of logs being replayed on the standby database.

STANDBY_RECV_BUF_SIZE

The standby receive buffer size, in pages.

STANDBY_RECV_BUF_PERCENT

The percentage of standby log receive buffer that is currently being used. Even if this value is 100, indicating that the receive buffer is full, the standby can continue to receive logs if you enabled log spooling.

STANDBY_SPOOL_LIMIT

The maximum number of pages to spool. A value of 0 indicates that log spooling is disabled; a value of -1 indicates that there is no limit. When the **hadr_spool_limit** configuration parameter is AUTOMATIC (the default in V10.5), this field returns the computed spool size in units of pages; that is, the actual maximum size of the spool.

STANDBY_SPOOL_PERCENT

The percentage of spool space used, relative to the configured spool limit. If the spool limit is 0 (spooling disabled) or -1 (unlimited spooling), NULL is returned. When STANDBY_SPOOL_PERCENT percent reaches 100%, the standby stops receiving logs, until some more space is released as replay proceeds. Note that spooling can stop before the limit is hit (before STANDBY_SPOOL_PERCENT reaches 100%), if the spool device (standby log path) is full.

STANDBY_ERROR_TIME

The most recent time when the standby database encountered a major error. Check the administration notification log and db2diag.log for error messages that have occurred since the last time you checked for errors. Check the logs fully, not just until the value reported by the STANDBY_ERROR_TIME value. There might be multiple errors. Log entries might include, but are not limited to, the following errors:

- Replay errors taking a table space to an abnormal state
- Load replay errors taking a table to an invalid state

The STANDBY_ERROR_TIME value is reset to NULL when a database changes its role from primary or standard to standby. It is not reset when a standby database is deactivated and reactivated.

PEER_WINDOW

The value of the **hadr_peer_window** database configuration parameter.

READS_ON_STANDBY_ENABLED

An indicator of whether the HADR reads on standby feature is enabled. Possible values are:

- Y
- N

STANDBY_REPLAY_ONLY_WINDOW_ACTIVE

An indicator of whether the replay-only window (caused by DDL or maintenance-operation replay) is in progress on the standby, meaning that readers are not allowed on the standby. Possible values are:

- Y
- N

PEER_WINDOW_END

The point in time until which the primary database stays in peer or disconnected peer state, as long as the primary database is active. The field is displayed only if you enabled a peer window.

STANDBY_REPLAY_DELAY

Indicates the value of the **hadr_replay_delay** database configuration parameter.

TAKEOVER_APP_REMAINING_PRIMARY

The current number of applications still to be forced off the primary during a non-forced takeover. This field is displayed only if there is a non-forced takeover in progress.

TAKEOVER_APP_REMAINING_STANDBY

The current number of applications still to be forced off the read-enabled standby during a takeover. This field is displayed only if there is a takeover in progress.

STANDBY_REPLAY_ONLY_WINDOW_START

The time at which the current replay-only window became active. This field is displayed only if there is an active replay-only window on the read-enabled standby.

STANDBY_REPLAY_ONLY_WINDOW_TRAN_COUNT

The total number of existing uncommitted DDL or maintenance transactions that have been executed so far in the current replay-only window. This field is displayed only if there is an active replay-only window on the read-enabled standby.

Part 4. High availability

The availability of a database solution is a measure of how successful user applications are at performing their required database tasks.

If user applications cannot connect to the database, or if their transactions fail because of errors or time out because of load on the system, the database solution is not very available. If user applications are successfully connecting to the database and performing their work, the database solution is highly available.

Designing a highly available database solution, or increasing the availability of an existing solution requires an understanding of the needs of the applications accessing the database. To get the greatest benefit from the expense of additional storage space, faster processors, or more software licenses, focus on making your database solution as available as required to the most important applications for your business at the time when those applications need it most.

Unplanned outages

Unexpected system failures that could affect the availability of your database solution to users include: power interruption; network outage; hardware failure; operating system or other software errors; and complete system failure in the event of a disaster. If such a failure occurs at a time when users expect to be able to do work with the database, a highly available database solution must do the following:

- Shield user applications from the failure, so the user applications are not aware of the failure. For example, DB2 Data Server can reroute database client connections to alternate database servers if a database server fails.
- Respond to the failure to contain its effect. For example, if a failure occurs on one machine in a cluster, the cluster manager can remove that machine from the cluster so that no further transactions are routed to be processed on the failed machine.
- Recover from the failure to return the system to normal operations. For example, if standby database takes over database operations for a failed primary database, the failed database might restart, recover, and take over once again as the primary database.

These three tasks must be accomplished with a minimum effect on the availability of the solution to user applications.

Planned outage

In a highly available database solution, the impact of maintenance activities on the availability of the database to user applications must be minimized as well.

For example, if the database solution serves a traditional store front that is open for business between the hours of 9am to 5pm, then maintenance activities can occur offline, outside of those business hours without affecting the availability of the database for user applications. If the database solution serves an online banking business that is expected to be available for customers to access through the Internet 24 hours per day, then maintenance activities must be run online, or scheduled for off-peak activity periods to have minimal impact on the availability of the database to the customers.

When you are making business decisions and design choices about the availability of your database solution, you must weigh the following two factors:

- The cost to your business of the database being unavailable to customers
- The cost of implementing a certain degree of availability

For example, consider an Internet-based business that makes a certain amount of revenue, X , every hour the database solution is serving customers. A high availability strategy that saves 10 hours of downtime per year will earn the business $10X$ extra revenue per year. If the cost of implementing this high availability strategy is less than the expected extra revenue, it would be worth implementing.

Chapter 17. Inplace (online) table reorganization

Inplace table reorganization reorganizes a table and allows full access to data in the table. The cost of this uninterrupted access to the data is a slower table REORG operation.

Starting in DB2 Cancun Release 10.5.0.4, inplace table reorganization is supported in DB2 pureScale environments.

During an inplace or online table REORG operation, portions of a table are reorganized sequentially. Data is not copied to a temporary table space; instead, rows are moved within the existing table object to reestablish clustering, reclaim free space, and eliminate overflow rows.

There are four main phases in an online table REORG operation:

1. SELECT n pages

During this phase, the database manager selects a range of n pages, where n is the size of an extent with a minimum of 32 sequential pages for REORG processing.

2. Vacate the range

The REORG utility moves all rows within this range to free pages in the table. Each row that is moved leaves behind a REORG table pointer (RP) record that contains the record ID (RID) of the row's new location. The row is placed on a free page in the table as a REORG table overflow (RO) record that contains the data. After the utility finishes moving a set of rows, it waits until all applications that are accessing data in the table are finished. These "old scanners" use old RIDs when table data is accessed. Any table access that starts during this waiting period (a "new scanner") uses new RIDs to access the data. After all of the old scanners are complete, the REORG utility cleans up the moved rows, deleting RP records and converting RO records into regular records.

3. Fill the range

After all rows in a specific range are vacated, they are written back in a reorganized format, they are sorted according to any indexes that were used, and obeying any PCTFREE restrictions that were defined. When all of the pages in the range are rewritten, the next n sequential pages in the table are selected, and the process is repeated.

4. Truncate the table

By default, when all pages in the table are reorganized, the table is truncated to reclaim space. If the NOTRUNCATE option is specified, the reorganized table is not truncated.

Files created during an online table REORG operation

During an online table REORG operation, an .OLR state file is created for each database partition. This binary file has a name whose format is xxxxyyyy.OLR, where xxxx is the table space ID and yyyy is the object ID in hexadecimal format. This file contains the following information that is required to resume an online REORG operation from the paused state:

- The type of REORG operation
- The life log sequence number (LSN) of the table that is reorganized

- The next range to be vacated
- Whether the REORG operation is clustering the data or just reclaiming space
- The ID of the index that is being used to cluster the data

A checksum is run on the .OLR file. If the file becomes corrupted, causing checksum errors, or if the table LSN does not match the life LSN, a new REORG operation is initiated, and a new state file is created.

If the .OLR state file is deleted, the REORG process cannot resume, SQL2219N is returned, and a new REORG operation must be initiated.

The files that are associated with the reorganization process must not be manually removed from your system.

Reorganizing tables online

An online or inplace table reorganization allows users to access a table while it is being reorganized.

Before you begin

You must have SYSADM, SYSCTRL, SYSMAINT, DBADM, or SQLADM authority, or CONTROL privilege on the table that is to be reorganized. You must also have a database connection to reorganize a table.

About this task

After you have identified the tables that require reorganization, you can run the reorg utility against those tables and, optionally, against any indexes that are defined on those tables.

Procedure

- To reorganize a table online using the **REORG TABLE** command, specify the name of the table and the **INPLACE** parameter. For example:

```
reorg table employee inplace
```
- To reorganize a table online using an SQL CALL statement, specify the **REORG TABLE** command with the ADMIN_CMD procedure. For example:

```
call sysproc.admin_cmd ('reorg table employee inplace')
```
- To reorganize a table online using the administrative application programming interface, call the db2Reorg API.

What to do next

After reorganizing a table, collect statistics on that table so that the optimizer has the most accurate data for evaluating query access plans.

Locking and concurrency considerations for online table reorganization

One of the most important aspects of online table reorganization-because it is so crucial to application concurrency-is how locking is controlled.

An online table reorg operation can hold the following locks:

- To ensure write access to table spaces, an IX lock is acquired on the table spaces that are affected by the reorg operation.
- A table lock is acquired and held during the entire reorg operation. The level of locking is dependent on the access mode that is in effect during reorganization:
 - If ALLOW WRITE ACCESS was specified, an IS table lock is acquired.
 - If ALLOW READ ACCESS was specified, an S table lock is acquired.
- An S lock on the table is requested during the truncation phase. Until the S lock is acquired, rows can be inserted by concurrent transactions. These inserted rows might not be seen by the reorg utility, and could prevent the table from being truncated. After the S table lock is acquired, rows that prevent the table from being truncated are moved to compact the table. After the table is compacted, it is truncated, but only after all transactions that are accessing the table at the time the truncation point is determined have completed.
- A row lock might be acquired, depending on the type of table lock:
 - If an S lock is held on the table, there is no need for individual row-level S locks, and further locking is unnecessary.
 - If an IS lock is held on the table, an NS row lock is acquired before the row is moved, and then released after the move is complete.
- Certain internal locks might also be acquired during an online table reorg operation.

Locking has an impact on the performance of both online table reorg operations and concurrent user applications. You can use lock snapshot data to help you to understand the locking activity that occurs during online table reorganizations.

Chapter 18. High availability disaster recovery (HADR) in DB2 pureScale environments

HADR support in DB2 pureScale environments combines the continuous availability of the DB2 pureScale Feature with the robust disaster recovery capabilities of HADR.

The integration of HADR with the DB2 pureScale Feature provides a number of advantages:

- Better synchronization. Both SUPERASYNC and ASYNC synchronization modes are supported in DB2 pureScale environments.
- DDL support. HADR replicates DDL operations.
- Ease of use. HADR is easy to configure and maintain.
- Native takeover support. By using the **TAKEOVER HADR** command, you can perform either a graceful takeover (*role switch*) or a forced takeover (*failover*).

Configuring and managing HADR in a DB2 pureScale environment is very similar to configuring and managing HADR in other environments. You create a standby database by restoring using a backup image or split mirror from the primary database, set various HADR configuration parameters, and start HADR on the standby and then the primary. The standby can quickly take over as the primary in the event of a role switch or a failover. All the administration commands are the same as what you are used to with HADR in other environments, but you can use only the **db2pd** command and the MON_GET_HADR table function to monitor HADR. Other monitor interfaces such as snapshot **do not** report HADR information in DB2 pureScale environments.

There are, however, some important differences for HADR in DB2 pureScale environments. An HADR pair is made up of a primary cluster and a standby cluster. Each cluster is made up of multiple members and at least one cluster caching facility; the member topology must be the same in the two clusters. The member from which you issue the **START HADR** command, on both the primary and the standby, is designated as the *preferred replay member*. When the database operates as a standby, only one member (the replay member) is activated. The database selects the preferred replay member as the replay member if the DB2 instance is online on the member, otherwise, another member is selected. That replay member replays all of the logs, and the other members are not activated. An HADR TCP connection is established between each member on the primary and the current replay member on the standby. Each member on the primary ships its logs to the standby replay member through the TCP connection. The HADR standby merges and replays the log streams. If the standby cannot connect to a particular member, A, on the primary (because of network problems or because the member is inactive) another member, B, on the primary that can connect to the standby sends the logs for member A to the standby. This process is known as *assisted remote catchup*.

Restrictions for HADR in DB2 pureScale environments

There are a number of restrictions that you should be aware of if you are planning to deploy HADR in a DB2 pureScale environment.

The restrictions are as follows:

- The HADR synchronization modes SYNC and NEARSYNC are not supported. You must specify either the ASYNC or SUPERASYNC option for the **hadr_syncmode** configuration parameter.
- A peer window is not supported because it requires either SYNC or NEARSYNC to be the synchronization mode.
- You cannot have more than one HADR standby database.
- The topology of the primary and the standby must be synchronized. If you add a member on the primary, that operation is replayed on the standby. If you drop a member on the primary, you must reinitialize the standby by using a backup or a split mirror from the primary's new topology.
- The reads on standby feature is not supported.
- You cannot use the integrated cluster manager, IBM Tivoli® System Automation for Multiplatforms (SA MP), to manage automated failover; it manages high availability within the local cluster only.
- Network address translation (NAT) between the primary and standby sites is not supported.

HADR setup in a DB2 pureScale environment

There are a few considerations for setting up an HADR database pair in a DB2 pureScale environment.

Asymmetric standby members

Because only one member on the standby replays logs, consider configuring a standby member with more CPU power and memory to serve as the preferred replay member. Similarly, consider which member on the primary cluster has the most CPU and memory, so that you can select it to be the preferred replay member if the current primary become the standby after a role switch. In both cases, you designate the preferred replay member by issuing the **START HADR** command from that member, with either AS PRIMARY or AS STANDBY option. The preferred replay member designation is persistent. It remains in place until changed by the next **START HADR** command.

Because the member topologies on the primary and standby must be the same, if you add members on the primary, you must also add members on the standby. If you have resource constraints, such as hardware constraints, you can configure the new standby members as logical members that share hosts. If the standby takes over the primary role, this new primary will not be as powerful as the old primary.

Standby cluster caching facilities

The cluster caching facility (CF) does not have to be the same on the primary and standby clusters. The standby makes minimal use of the CF because only one member performs the replay, so it is possible to have only one CF on the standby cluster. If, however, the standby takes over as the new primary, you should add a CF to help ensure that your DB2 pureScale environment is highly available. Adding that secondary CF requires an outage because you cannot add it without stopping the DB2 pureScale instance.

Member subsetting

You can use member subsetting to specify a subset of members to use for a database. The subset is stored internally as a list of member numbers. The database

then maps the members to host names for the client to connect to. If this database uses HADR, you can only specify the subset list on the primary database. The subset member list is replicated to the standby.

High availability disaster recovery (HADR) monitoring in a DB2 pureScale environment

You must use the **db2pd** command or the `MON_GET_HADR` table function to monitor your HADR databases in a DB2 pureScale environment.

Other interfaces, such as the **GET SNAPSHOT FOR DATABASE** command, the `SNAPHADR` administrative view, and the `SNAP_GET_HADR` table function, do not return any HADR information, so it will appear as if HADR is not configured.

The **db2pd** command and the `MON_GET_HADR` table function return essentially the same information, but because reads on standby is not supported in a DB2 pureScale environment, you can only use the **db2pd** command to monitor from a standby. As well, the **db2pd** command is preferred during takeover because there could be a time window during which neither the primary nor the standby allows client connections.

db2pd command

In a DB2 pureScale environment, the **db2pd** command returns a section in its output for each log stream being processed by the member on which the command is issued. On the primary, any assisted remote catchup streams are reported by their owner (that is, the assisted member) and the assisting member. On the standby, all of the streams are returned if the command is issued on the replay member; if you issue **db2pd** on a non-replay member, no data is returned.

If you want to see all of the log streams from the primary, use the `-allmembers` and `-hadr` options with the **db2pd** command. If you use the `-allmembers` option on the standby, for each non replay member, the output indicates that database is not active on the member; for the replay member, all streams are returned. As a result, this command option is only useful on the standby if you want to find out which member is the current replay member (alternatively, you can check the `STANDBY_MEMBER` field in the **db2pd** output from the primary).

The following example is for an HADR setup using three-member clusters: members 0, 1, and 2. Member 1 is active but it is in assisted remote catchup state and is being assisted by member 0; the standby replay member is member 0. The DBA issues the **db2pd** command for member 0 on the primary. Notice that two sets of data are returned: one for each log stream member 0 is processing:

```
Database Member 0 -- Database HADRDB -- Active -- Up 0 days 00:23:17 --  
Date 06/08/2011 13:57:23
```

```
      HADR_ROLE = PRIMARY  
      REPLAY_TYPE = PHYSICAL  
      HADR_SYNCMODE = ASYNC  
      STANDBY_ID = 1  
      LOG_STREAM_ID = 0  
      HADR_STATE = PEER  
      HADR_FLAGS =  
PRIMARY_MEMBER_HOST = hostP.ibm.com  
PRIMARY_INSTANCE = db2inst  
PRIMARY_MEMBER = 0  
STANDBY_MEMBER_HOST = hostS.ibm.com  
STANDBY_INSTANCE = db2inst
```

```

        STANDBY_MEMBER = 0
        HADR_CONNECT_STATUS = CONNECTED
        HADR_CONNECT_STATUS_TIME = 06/08/2011 13:38:10.199479 (1307565490)
        HEARTBEAT_INTERVAL(seconds) = 25
        HADR_TIMEOUT(seconds) = 100
        TIME_SINCE_LAST_RECV(seconds) = 3
        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
        LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
        LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
        LOG_HADR_WAIT_COUNT = 82
        SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
        SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
        PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
        STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
        HADR_LOG_GAP(bytes) = 0
        STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
        STANDBY_RECV_REPLAY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
        STANDBY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
        STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:19.000000 (1307566159)
        STANDBY_RECV_BUF_SIZE(pages) = 16
        STANDBY_RECV_BUF_PERCENT = 0
        STANDBY_SPOOL_LIMIT(pages) = 13000
        STANDBY_SPOOL_PERCENT = 0
        PEER_WINDOW(seconds) = 0
        READS_ON_STANDBY_ENABLED = N

        HADR_ROLE = PRIMARY
        REPLAY_TYPE = PHYSICAL
        HADR_SYNCMODE = ASYNC
        STANDBY_ID = 1
        LOG_STREAM_ID = 1
        HADR_STATE = REMOTE_CATCHUP
        HADR_FLAGS = ASSISTED_REMOTE_CATCHUP ASSISTED_MEMBER_ACTIVE
        PRIMARY_MEMBER_HOST = hostP.ibm.com
        PRIMARY_INSTANCE = db2inst
        PRIMARY_MEMBER = 0
        STANDBY_MEMBER_HOST = hostS.ibm.com
        STANDBY_INSTANCE = db2inst
        STANDBY_MEMBER = 0
        HADR_CONNECT_STATUS = CONNECTED
        HADR_CONNECT_STATUS_TIME = 06/08/2011 13:35:51.724447 (1307565351)
        HEARTBEAT_INTERVAL(seconds) = 25
        HADR_TIMEOUT(seconds) = 100
        TIME_SINCE_LAST_RECV(seconds) = 16
        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
        LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.005631
        LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.837
        LOG_HADR_WAIT_COUNT = 124
        SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
        SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
        PRIMARY_LOG_FILE,PAGE,POS = S0000012.LOG, 1, 56541576
        STANDBY_LOG_FILE,PAGE,POS = S0000012.LOG, 1, 56541576
        HADR_LOG_GAP(bytes) = 0
        STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000012.LOG, 1, 56541576
        STANDBY_RECV_REPLAY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 06/08/2011 13:49:25.000000 (1307566165)
        STANDBY_LOG_TIME = 06/08/2011 13:49:25.000000 (1307566165)
        STANDBY_REPLAY_LOG_TIME = 06/08/2011 13:49:25.000000 (1307566165)
        STANDBY_RECV_BUF_SIZE(pages) = 16
        STANDBY_RECV_BUF_PERCENT = 0
        STANDBY_SPOOL_LIMIT(pages) = 13000
        STANDBY_SPOOL_PERCENT = 0
        PEER_WINDOW(seconds) = 0
        READS_ON_STANDBY_ENABLED = N

```

MON_GET_HADR table function

In a DB2 pureScale environment, the MON_GET_HADR table function returns a row for each log stream. The table function cannot be issued on the standby because reads on standby is not supported in a DB2 pureScale environment. Use the LOG_STREAM_ID field in the table function output to identify the log stream and the PRIMARY_MEMBER and STANDBY_MEMBER fields to identify the members processing the stream on the primary and standby sides.

The table function takes a member argument and returns the stream that the specified member owns and all remote catchup streams that it is assisting. If the argument is an assisting member, the assisted remote catchup streams have their HADR_STATE field reported as being in REMOTE_CATCHUP with the ASSISTED_REMOTE_CATCHUP flag set in the HADR_FLAGS field. If the argument is an assisted member, the assisted remote catchup stream has its HADR_STATE field reported as DISCONNECTED.

If you specify -1 or NULL as the argument, the results for the current database member (that is, the member processing the query) are returned. If you specify -2 as the argument, the results for all members on the primary are returned. Any assisted remote catchup streams are reported on the assisting member only. If a member is inactive and assisted remote catchup has not yet been established for that member's log stream, that log stream does not appear in the output. The table function request is passed to all active members and the results are merged, so inactive members are not represented.

In the following examples, the DBA calls the **MON_GET_HADR** table function for monitoring an HADR setup using three-member clusters: members 0, 1, and 2. Member 1 is active but it is in assisted remote catchup state and is being assisted by member 0; the standby replay member is member 0. The DBA calls the table function with argument 0, 1, 2 and -2 (for all members). Notice that two rows are returned when the argument is 0: one for each log stream that member 0 is processing:

Example for member 0

```
select LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
from table (mon_get_hadr(0))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
0	0	0	PEER	
1	0	0	REMOTE_CATCHUP	ASSISTED_REMOTE_CATCHUP

Example for member 1

```
select LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
from table (mon_get_hadr(1))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
1	1	0	DISCONNECTED	

Example for member 2

```
select LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
from table (mon_get_hadr(2))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
2	2	0	PEER	

Example for all members

```
select LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
from table (mon_get_hadr(-2))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE	HADR_FLAGS
0	0	0	PEER	
1	0	0	REMOTE_CATCHUP	ASSISTED_REMOTE_CATCHUP
2	2	0	PEER	

HADR standby replay in a DB2 pureScale environment

In a DB2 pureScale environment, only one member of the HADR standby cluster replays logs, and other members remain inactive. Members in the HADR primary cluster ship their logs to the replay member directly by using a TCP connection or indirectly through assisted remote catchup.

When a standby database is started, standby replay service is activated on the member that is designated as the *preferred replay member* if the DB2 instance is online on the member. Otherwise, replay is activated on another member. There is no way to control the selection among non preferred members; however, any member that is running in restart light mode (that is, a member that is not active on its home host) is given the lowest priority. Even though the preferred replay member designation is persistent, if replay is active on another member because the preferred replay member was not available, replay does not automatically revert to the preferred member when that member becomes available. The only way to force replay onto the preferred replay member is to deactivate and reactivate HADR on the standby.

Because the replay member on the standby replays logs that are generated by all members on the primary, there is a possibility that it can become a bottleneck. To avoid a potential impact, you should select the member with more resources, such as CPU and memory, as the preferred replay member. You implicitly designate the preferred replay member by issuing the **START HADR** command on it. The member from which you issue the **START HADR AS STANDBY** command is the preferred replay member on the standby cluster; the member from which you issue the **START HADR AS PRIMARY** command is the preferred replay member on the primary cluster. The status of preferred replay member on the primary takes effect only when the primary becomes a standby

If the current replay member goes down abnormally (for example, as a result of a software or hardware failure) or normally (for example, as a result of a user command to deactivate the particular member), replay is automatically migrated to another member. If the current replay member goes down abnormally, member crash recovery occurs, and a member is selected to resume replay, with preference to the preferred replay member during the selection (the old replay member might or might not be reselected). As long as there is one online member in the standby cluster, replay continues. To stop replay, deactivate the whole standby database.

You can find out which member is the current replay member from the primary or the standby. On the primary, use the **db2pd** command with the **-hadr** parameter or the **MON_GET_HADR** table function. The replay member is indicated in the **STANDBY_MEMBER** field. If you want to determine the current replay member from the standby, you can use only the **db2pd** command because the table function cannot be called from a standby in a DB2 pureScale environment. Because you do not know which replay member is active, you must issue the following command:

```
db2pd -hadr -db DB_name -allmembers
```

In the output, only the current replay member has HADR information; all non-replay members show Database *DB_name* not activated on database member *X*.

Changing the preferred replay member

You designate a preferred replay member by issuing the **START HADR** command on that member. If you want to change that designation, you have to reissue the command.

Note that if a database is already active and in the desired role, the **START HADR** command will be a nop (no operation performed) that returns an error, and the preferred replay member is not updated. Use the following procedure to designate or redesignate the preferred replay member.

About this task

The preferred replay member is the member that is preferred for replaying logs on an HADR standby database in a DB2 pureScale environment. On a standby, it still might not be the member doing the actual replay. On the primary, it is the first member that the standby replay service attempts to start on if that primary becomes the standby. The preferred replay member designation is persistent and can only be changed by starting and stopping HADR.

Procedure

To designate a preferred replay member

- On the standby:
 1. Issue the **DEACTIVATE DATABASE** command from any member in the standby cluster.
 2. Issue the **START HADR** command with the AS STANDBY option on the member that you want to designate as the preferred replay member.
- On the primary:
 1. Issue the **STOP HADR** command from any member in the primary cluster.

Note: The primary remains active during this procedure.

2. Issue the **START HADR** command with the AS PRIMARY option on the member that you want to designate as the preferred replay member.

Note: This designation only takes effect if the primary database switches to the standby role.

Results

If the **START HADR** command completes successfully, the preferred replay member is updated. If the **START HADR** command fails, the preferred member might or might not have been updated, depending on how far the command execution went. To ensure that the designation applies to the correct member, rerun the procedure described in this task.

DB2 pureScale topology changes and high availability disaster recovery (HADR)

In a DB2 pureScale environment, making changes to the HADR primary cluster and HADR standby cluster can require an outage.

In general, the primary and standby clusters must have the same member topology; that is, each instance must have the same number of members with the

same member IDs. The only exception is when you add members to the standby. You can add members when the database is either offline or online. If you drop a member from the primary cluster (dropping a member is not allowed on the standby), you must stop HADR, deactivate the primary, and reinitialize the standby.

If you add a cluster caching facility (CF), you also require an outage on the DB2 pureScale instance.

Adding members to a high availability disaster recovery (HADR) setup

You can scale out your DB2 pureScale instance by adding members without impacting your HADR setup. You can add members online or offline.

Procedure

To add a member to an HADR setup in a DB2 pureScale instance:

1. Add the new member to the standby cluster. From a standby host that is part of the DB2 pureScale instance, issue the following command:

```
db2iupdt -add -m member_ID -mnet net_name instance_name
```

This command adds the member to the member topology but not to the database topology.

2. Update the member-specific configuration parameters for the new member on the standby:

```
UPDATE DATABASE CONFIGURATION FOR db_name MEMBER member_ID
  USING hadr_local_host standby_member_host

UPDATE DATABASE CONFIGURATION FOR db_name MEMBER member_ID
  USING hadr_local_svc standby_member_port
```

3. Add the new member to the primary cluster. From a primary host that is part of the DB2 pureScale instance, issue the following command:

```
db2iupdt -add -m member_ID -mnet net_name instance_name
```

You must use the same member ID that you specified when adding the member to the standby cluster. This command adds the member to the member topology but not to the database topology.

4. Update the member-specific configuration parameters for the new member on the primary:

```
UPDATE DATABASE CONFIGURATION FOR db_name MEMBER member_ID
  USING hadr_local_host primary_member_host

UPDATE DATABASE CONFIGURATION FOR db_name MEMBER member_ID
  USING hadr_local_svc primary_member_port
```

On the primary this member is not listed because it is not currently activated. On the standby this member is not listed because it does not yet exist in the database topology.

5. Activate the new member on the primary by doing one of the following steps:
 - Connect to the database on the new member.
 - Issue the **ACTIVATE DATABASE** command.

If you have not added the member to the standby cluster by the time that it receives the add member log record that results from the member activation on the primary, the standby database will be shut down.

What to do next

Add the new member to the target list on the primary and the standby.

Removing members from a high availability disaster recovery (HADR) setup

Removing a member from your DB2 pureScale instance requires you to reinitialize the standby based on the primary's updated topology.

About this task

To drop a member, you need to stop HADR and the DB2 pureScale instance. You cannot drop the last member in the instance using this procedure.

Procedure

To remove a member from an HADR setup in a DB2 pureScale instance:

1. Remove the member from the primary cluster. You must do this from a host that will still belong to the instance after the member is dropped.
 - a. Stop HADR on the primary database using the **STOP HADR** command.
 - b. Stop the DB2 pureScale instance using the **db2stop** command.
 - c. Drop the member by running the following command:

```
db2iupdt -drop -m member_ID instance_name
```

Note: You cannot directly drop a member from an HADR standby database.

2. Remove the member from the standby cluster. You must do this from a host that will still belong to the instance after the member is dropped.
 - a. Deactivate the database on the standby database using the **DEACTIVATE DATABASE** command.

```
DEACTIVATE DATABASE db_name
```
 - b. Drop the database using the following command:

```
DROP DATABASE db_name
```
 - c. Drop the member by running the following command:

```
db2iupdt -drop -m member_ID instance_name
```

You must use the same member ID that you specified when removing the member from the primary cluster.

3. Create the standby database by restoring a backup image or by initializing a split mirror, based on the primary's updated topology after step 1.
 - a. On the primary, issue the following command:

```
BACKUP DB dbname
```
 - b. Restore the standby by issuing the following command:

```
RESTORE DB dbname
```
4. Update the HADR-specific database configuration parameters on the standby cluster.
5. Start HADR on the primary:

```
START HADR AS PRIMARY
```
6. Start HADR on the standby:

```
START HADR AS STANDBY
```

HADR takeover operations in a DB2 pureScale environment

When an HADR standby database takes over as the primary database in a DB2 pureScale environment, there are a number of important differences from HADR in other environments.

With HADR, there are two types of takeover: *role switch* and *failover*. Role switch, sometimes called graceful takeover or non-forced takeover, can be performed only when the primary is available and it switches the role of primary and standby. Failover, or forced takeover, can be performed when the primary is not available. It is commonly used in primary failure cases to make the standby the new primary. The old primary remains in the primary role in a forced takeover, but the standby sends it a message to disable it. Both types of takeover are supported in a DB2 pureScale environment, and both can be issued from any of the standby database members and not just the current replay member. However, after the standby completes the transition to the primary role, the database is only started on the member that served as the replay member before the takeover. The database can be started on the other members by issuing an **ACTIVATE DATABASE** command or implicitly through a client connection.

Role switch

After a role switch, which is initiated by issuing the **TAKEOVER HADR** command from any standby member, the standby cluster becomes the primary cluster and vice versa. Role switch helps ensure that no data is lost between the old primary and new primary. You can initiate a role switch in the following circumstances only:

- Crash recovery is not occurring on the primary cluster, including member crash recovery that is pending or in progress.
- All the log streams are in peer or assisted remote catchup state.
- All the log streams are in remote catchup state or in assisted remote catchup state, and the synchronization mode is SUPERASYNC.

Before you initiate a role switch in remote catchup or assisted remote catchup state, check the log gap between the primary and standby log streams. A large gap can result in a long takeover time because all of the logs in that gap must be shipped and replayed first.

During a role switch, the following steps occur on the primary:

1. New connections are rejected on all members, any open transactions are rolled back, and all remaining logs are shipped to the standby.
2. The primary cluster's database role changes to standby.
3. A member that has a direct connection to the standby is chosen as the replay member, with preference given to the preferred replay member (that is, the member that HADR was started from).
4. Log receiving and replay starts on the replay member.
5. The database is shut down on the other non-replay members of the cluster.

And the following steps occur on the standby:

1. Log receiving is stopped on the replay member after the end of logs is reached on each log stream, helping ensure no data loss.
2. The replay member finishes replaying all received logs.
3. After it is confirmed that the primary cluster is now in the standby role, the replay member changes the standby cluster's role to primary.

4. The database is opened for client connections, but it is only activated on the member that was previously the standby replay member.

Failover

After a failover, which is initiated by issuing the **TAKEOVER HADR** command with the **BY FORCE** option from any standby member, the standby cluster becomes the primary cluster. The old primary cluster is sent a disabling message, but its role is not changed. Any member on the primary that receives this message disables the whole primary cluster. By initiating a failover, you are accepting the trade-off between potential data loss and having a working database. You cannot initiate a failover if the databases are in local catchup state.

Note: Unlike in previous releases, you can now initiate a failover even if log archive retrieval is in progress.

During a failover, the following steps occur on the primary (assuming it is online and connected to the standby):

1. After it receives the disabling message, the database is shut down and log writing is stopped.

And the following steps occur on the standby, all of which are carried out from the replay member:

1. A disabling message is sent to the primary, if it is connected.
2. Log shipping and log retrieval is stopped, which entails a risk of data loss.
3. The replay member finishes replaying all received logs (that is, the logs that are stored in the log path).
4. Any open transactions are rolled back.
5. The replay member changes the standby cluster's role to primary.
6. The database is opened for client connections, but it is only activated on the member that was previously the standby replay member.

You can reintegrate the old primary as a new standby only if its log streams did not diverge from the new primary's log streams. Before you can start HADR, the database must be offline on all of the old primary's members; the cluster caching facilities, however, can stay online. If any members are online, kill them instead of issuing the **DEACTIVATE DATABASE** command on them.

Scenario: Deploying HADR in a DB2 pureScale environment

This scenario describes the planning, configuring, and deploying of a high availability disaster recovery (HADR) setup for an online travel service called ExampleFlightsExpress (EFE), which is currently using the DB2 pureScale Feature. All these steps can be done without any downtime.

Background

EFE chose to use the DB2 pureScale Feature for two reasons:

- Continuous availability. Downtime is fatal in the online retailing business, where customers are literally accessing services 24x7.
- Scalability. Because customer demand ebbs and flows depending on the time of year, EFE must be able to add capacity easily and without taking an outage.

EFE does not have a complete disaster recovery plan. EFE's setup is resilient unless there is a widespread outage that brings down the whole DB2 pureScale cluster. To

address this shortcoming, EFE is going to use HADR, which is supported with the DB2 pureScale Feature. In a DB2 pureScale environment, HADR has a few limitations, such as no support for reads on standby and no support for SYNC or NEARSYNC synchronization modes, but those are acceptable because EFE wants HADR only for disaster recovery.

Planning

EFE is going to use the company's head office (site A) as the location for the HADR primary cluster and a regional office (site B), which is 500 miles (800 km) away as the location for the standby cluster. The two sites are connected by a WAN. Other details about the environment are as follows:

- Database name: `hadr_db`
- Instance owner on all hosts: `db2inst`
- TCP port that is used for HADR primary-standby communication: 4000
- TCP port that is used for SQL client/server communication: 8000
- Hosts for cluster caching facilities (with IDs 128 and 129) and members (with IDs 0, 1, 2, and 3) on the primary: `cfp0`, `cfp1`, `p0`, `p1`, `p2`, and `p3`
- Hosts for cluster caching facilities and members on the standby: `cfs0`, `cfs1`, `s0`, `s1`, `s2`, and `s3`

Preferred replay members

Only one member on the standby performs all the replay for the logs that are generated on all the members on the primary. Therefore, EFE's DBA determines which member hosts in the primary and standby clusters have the most memory and CPU resources and designates those members as the preferred replay members. It is necessary to do this planning even for the primary because that designated member performs all the replay if the primary database fails and is reintegrated as the standby. On the primary, this is `p0` and on the standby, it is `s0`; in both cases, member 0 is the resident member on those hosts.

Synchronization mode

EFE's DBA must choose between ASYNC (the default) and SUPERASYNC for the synchronization mode. To do this, the DBA analyzes the WAN and determines that network throughput is 300 Mbit/second and that the round-trip time is 80 ms. Next, the DBA measures the logging rate, which is 20 MB/second at the cluster level. The network throughput is sufficient to support the logging rate and allow peak logging to reach 37 MB/second, so ASYNC is a suitable mode. If the throughput were closer to the logging rate, SUPERASYNC would be a better choice because it would allow the primary to get far ahead of the standby during peak transaction time.

Scaling considerations

Because EFE tends to add temporary members during peak times of the year, EFE must decide how to scale out the standby, because the member topology must be the same across the HADR pair. To avoid additional costs, EFE decides that when it deploys additional members on the primary, on the standby cluster, it uses multiple members on the host machines. This would likely result in a less powerful database if the standby must take over as the new primary, but the savings are worth this potential drawback.

Configuring HADR

The DBA performs the following steps:

1. The DBA takes a backup of the intended primary database, `hadr_db`:

```
db2 BACKUP DB hadr_db TO backup_dir
```
2. The DBA restores the backup onto the intended standby cluster by issuing the following command:

```
db2 RESTORE DB hadr_db FROM backup_dir
```
3. On the primary, the DBA sets the cluster-level HADR parameters that specify the standby cluster and the synchronization mode. Particularly important is the **`hadr_target_list`** parameter, which lists the remote members. Only one remote member is required to be listed in the **`hadr_target_list`**. DB2 retrieves the other members' addresses after the initial contact with the listed member. However, providing multiple addresses prevents a single point of failure, that is, the clusters cannot connect to each other if the one and only listed member is down. The DBA issues the following command:

```
db2 "UPDATE DB CFG FOR hadr_db USING
    HADR_TARGET_LIST {s0:4000|s1:4000|s2:4000|s3:4000}
    HADR_REMOTE_HOST {s0:4000|s1:4000|s2:4000|s3:4000}
    HADR_REMOTE_INST db2inst
    HADR_SYNCMODE     async"
```

Because there is only one standby, the **`hadr_remote_host`** parameter specifies the same group of addresses as the **`hadr_target_list`** parameter.

4. The DBA sets the member-level HADR parameters on the primary, which identify the address and port for each member:

- For member 0:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 0 USING
    HADR_LOCAL_HOST p0
    HADR_LOCAL_SVC  4000"
```

- For member 1:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 1 USING
    HADR_LOCAL_HOST p1
    HADR_LOCAL_SVC  4000"
```

- For member 2:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 2 USING
    HADR_LOCAL_HOST p2
    HADR_LOCAL_SVC  4000"
```

- For member 3:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 3 USING
    HADR_LOCAL_HOST p3
    HADR_LOCAL_SVC  4000"
```

5. On the standby, the DBA sets the cluster-level HADR parameters that specify the primary cluster and the synchronization mode:

```
db2 "UPDATE DB CFG FOR hadr_db USING
    HADR_TARGET_LIST {p0:4000|p1:4000|p2:4000|p3:4000}
    HADR_REMOTE_HOST {p0:4000|p1:4000|p2:4000|p3:4000}
    HADR_REMOTE_INST db2inst
    HADR_SYNCMODE     async"
```

6. The DBA sets the member-level HADR parameters on the standby, which identify the address and port for each member:

- For member 0:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 0 USING
    HADR_LOCAL_HOST s0
    HADR_LOCAL_SVC  4000"
```


- For member 1:
db2 "UPDATE DB CFG FOR hadr_db MEMBER 1 USING
HADR_LOCAL_HOST s1
HADR_LOCAL_SVC 4000"
- For member 2:
db2 "UPDATE DB CFG FOR hadr_db MEMBER 2 USING
HADR_LOCAL_HOST s2
HADR_LOCAL_SVC 4000"
- For member 3:
db2 "UPDATE DB CFG FOR hadr_db MEMBER 3 USING
HADR_LOCAL_HOST s3
HADR_LOCAL_SVC 4000"

Starting HADR

As with other HADR environments, the standby database must be started first. Because the member that the **START HADR** command is issued from is designated the preferred replay member, the DBA issues the following commands:

- From member 0 on the standby:
db2 START HADR ON DB hadr_db AS STANDBY
- From member 0 on the primary:
db2 START HADR ON DB hadr_db AS PRIMARY

To determine that HADR is up and running, the DBA calls the MON_GET_HADR table function from the primary:

```
select LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE
from table (mon_get_hadr(-2))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE
0	0	0	PEER
1	1	0	PEER
2	2	0	PEER
3	3	0	PEER

The DBA confirms that standby member 0, the preferred replay member, is indeed the current replay member by looking at the STANDBY_MEMBER field. Every log stream reports the same member on the standby because all the members on the primary are connected to that standby member.

Role switch

The DBA has to perform a role switch; that is, the current standby will take over the primary role, and the current primary will take over the standby role. This will allow some maintenance which requires a shutdown of the cluster to be performed at site A. This procedure takes place during a time of low usage in order to minimize impact on applications currently connected to the database.

1. The DBA ensures that none of the members on the primary are in an abnormal state:

```
SELECT ID,
       varchar(STATE,21) AS STATE,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

ID	STATE	HOME_HOST	CUR_HOST	ALERT
----	-------	-----------	----------	-------

0	STARTED	p0	p0	NO
1	STARTED	p1	p1	NO
2	STARTED	p2	p2	NO
3	STARTED	p3	p3	NO

4 record(s) selected.

- The DBA ensures that all of the log streams are in PEER state:

```
select LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE
from table (mon_get_hadr(-2))
```

LOG_STREAM_ID	PRIMARY_MEMBER	STANDBY_MEMBER	HADR_STATE
0	0	0	PEER
1	1	0	PEER
2	2	0	PEER
3	3	0	PEER

- At site B, the DBA issues the **TAKEOVER HADR** command on member 0:

```
TAKEOVER HADR ON DB hadr_db
```

After the command completes, member 0 on the new standby (the preferred replay member) is chosen as the replay member and the database is shut down on the other members on the standby cluster. On the new primary, the database is only activated on member 0; other members are activated with a client connection or if the DBA explicitly issues the **ACTIVATE DATABASE** command on each of them. Automatic client reroute sends any new clients to site B.

- At site A, the DBA deactivates the database on the standby (this keeps the database in its role as an HADR standby):

```
DEACTIVATE DATABASE hadr_db
```

- At site A, the DBA stops DB2 on the standby:

```
db2stop
```

- At site A, the DBA performs the required maintenance.

- At site A, the DBA starts DB2 on the standby:

```
db2start
```

- At site A, the DBA activates the database on the standby:

```
ACTIVATE DATABASE hadr_db
```

The database is activated as an HADR primary with one replay member.

- To revert to the original setup, the DBA issues the **TAKEOVER HADR** command on member 0 at site A:

```
TAKEOVER HADR ON DB hadr_db
```

Failover

The DBA has to perform a failover; that is, an unexpected outage at site A requires that the standby at site B take over the primary role. An important difference for HADR in a DB2 pureScale environment is that there is no support for using IBM Tivoli System Automation for Multiplatforms (SA MP) to automate the failover (it's already being used to ensure high availability in the DB2 pureScale cluster). At any rate, in this scenario the DBA wants to have manual control over this kind of response to an outage.

- The DBA performs a forced takeover from the standby database at site B.

```
TAKEOVER HADR ON DB hadr_db BY FORCE
```

The standby sends a disabling message to shut down the primary. After stopping log shipping and retrieval, the standby completes the replay of any logs in its log path. Finally, the standby becomes the new primary.

2. The DBA issues the **db2pd** command on the new primary to ensure that it has taken over the primary role.

```
db2pd -hadr -db hadr_db
```

3. After addressing the cause of the outage and getting site A up and running again, the DBA attempts to reintegrate the old primary as a standby.

```
START HADR ON DB hadr_db AS STANDBY
```

4. The DBA verifies that the site A is now the standby by checking the HADR_CONNECT_STATUS and HADR_STATE fields to ensure that the show the database is connected and in either peer or remote catchup state.

```
db2pd -hadr -db hadr_db
```

Unfortunately, the log streams of the databases at the two sites have diverged, so the database is showing as disconnected. The DBA looks at the diag.log file of one of the members on the old primary and sees a message indicating that the database on site A cannot be made consistent with the new primary database.

5. The DBA has to drop the database and reinitialize it as an HADR standby at site A.

- a. Drop the database:

```
DROP DATABASE DB hadr_db
```

- b. Take a backup of the database at site B.

```
BACKUP DATABASE DB hadr_db ONLINE TO backup_dir
```

- c. Restore the backup image to site A.

```
db2 RESTORE DB hadr_db FROM backup_dir
```

- d. Set the cluster-level and member-level configuration parameters on the database at site A.

```
db2 "UPDATE DB CFG FOR hadr_db USING
      HADR_TARGET_LIST {s0:4000|s1:4000|s2:4000|s3:4000}
      HADR_REMOTE_HOST {s0:4000|s1:4000|s2:4000|s3:4000}
      HADR_REMOTE_INST db2inst
      HADR_SYNCMODE      async"
```

- For member 0:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 0 USING
      HADR_LOCAL_HOST p0
      HADR_LOCAL_SVC 4000"
```

- For member 1:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 1 USING
      HADR_LOCAL_HOST p1
      HADR_LOCAL_SVC 4000"
```

- For member 2:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 2 USING
      HADR_LOCAL_HOST p2
      HADR_LOCAL_SVC 4000"
```

- For member 3:

```
db2 "UPDATE DB CFG FOR hadr_db MEMBER 3 USING
      HADR_LOCAL_HOST p3
      HADR_LOCAL_SVC 4000"
```

6. The DBA wants to designate member 0 as the preferred replay member and issues the **START HADR** command from member 0 on the site A:

```
db2 START HADR ON DB hadr_db AS STANDBY
```

7. The DBA verifies that the site A is now the standby by checking the HADR_CONNECT_STATUS and HADR_STATE fields to ensure that the show the database is connected and is catching up to the primary.
`db2pd -hadr -db hadr_db`
8. To revert to the original setup, the DBA can perform a role switch as described in the previous section.

Chapter 19. Online fix pack updates in DB2 pureScale environments

Use an online fix pack update on members or cluster caching facilities (CFs) to update a DB2 pureScale instance to a new fix pack level or special build while the instance remains available. Members and CFs that are not being updated remain available for processing.

Starting with Version 10.5, a DB2 release has an architecture level and a code level. When you apply an online update, your DB2 pureScale instance must be at the required minimum committed code level. A new fix pack level has a different code level than an earlier fix pack release. However, the architecture level can be the same or different from an earlier fix pack release. Figure 11 on page 174 shows the architecture level and code level for a fix pack release.

In a DB2 pureScale instance, the Current Effective Code Level (CECL), and Current Effective Architecture Level (CEAL) enforce the level at which all members and CFs must operate, even if some of those members or CFs are updated to a newer level of code.

The **curr_eff_arch_level** database manager configuration parameter displays the CEAL for a DB2 pureScale instance and the **curr_eff_code_level** database manager configuration parameter display the CECL for the instance.

Each fix pack release or special build has a required minimum committed code level. Online updates to a specific fix pack release are supported from a required minimum committed code level. You can run the **installFixPack -show_level_info** command to know the minimum committed code level information for online updates. The sample output of the command is provided below:

```
installFixPack -show_level_info :
```

```
/devinst/db2_kepler/aix64/s130528/server> ./installFixPack -show_level_info
```

```
Code level = Version:10 Release:5 Modification:0 Fixpack:0  
Architecture level = Version:10 Release:5 Modification:0 Fixpack:0  
Section level = Version:10 Release:5 Modification:0 Fixpack:0
```

```
Supports online update = Yes
```

```
Minimum committed code level required for online install =  
Version:10 Release:5 Modification:0 Fixpack:0
```

```
The execution completed successfully.
```

```
For more information see the DB2 installation log at "/tmp/installFixPack.log.$PID".  
DBI1070I Program installFixPack completed successfully.
```

If you want to update an instance from a fix pack level that is earlier than the required committed code level, you can apply an offline update. Alternatively, you can apply an online update to the required minimum committed level followed by an online update to the specific fix pack level.

Online fix pack updates are always supported provided the new level of code (fix pack or special build) supports online updates. You can run the **installFixPack -show_level_info** command to know the new level of code.

Note: For special builds with a different architecture level, performing an online update requires a service password to be set.

During an offline or online fix pack update of a DB2 pureScale instance, you can have members and CFs at a code level that is different from the CECL. Here, the instance is in a heterogeneous state. You cannot update members or CFs to multiple code levels that are different from the CECL. After all the members and CFs are updated and the instance is committed, the instance returns to a homogeneous state as described in the following example:

1. Start with all members at FP1. Here CECL = FP1.
2. Update first member to FP2. Here the instance is in heterogeneous state.
3. Update all other members and CFs to FP2. Here the instance is still in heterogeneous state because CECL = FP1.
4. Commit the instance at FP2. Here the instance returns to homogeneous state.

Example of updating a DB2 pureScale instance to a fix pack with a higher code level

This example uses an online fix pack update to apply DB2 Version 10.5 FPy on an instance that has Version 10.5 FPx. The Version 10.5 FPy has the following characteristics:

- The architecture level is Version 10.5 FPy.
- The code level is Version 10.5 FPy.
- The minimum committed level is Version 10.5 FPw. Because Version 10.5 FPx is a later release than Version 10.5 FPw, online update from Version 10.5 FPx to Version 10.5 FPy is supported.

The following figure shows the architecture level and code level for *Instance1* and members before *Instance1* is updated to Version 10.5 FPy:

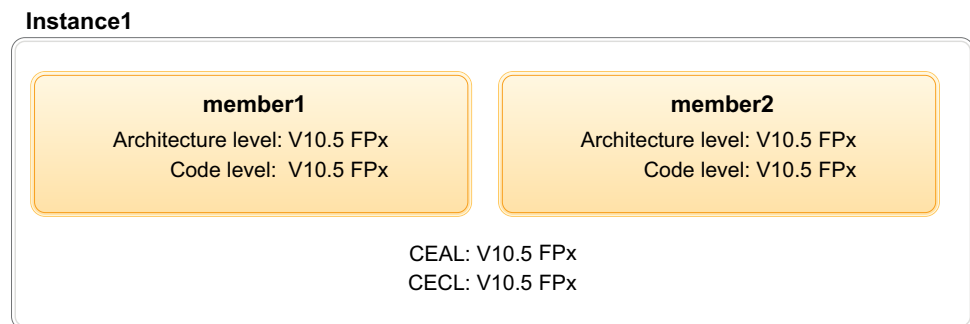


Figure 11. Architecture and code level values before *Instance1* is updated to Version 10.5 FPy.

After an online fix pack update is applied on *member2* to Version 10.5 FPy, note that the code level and architecture level for *member2* changed. However, the CECL and CEAL remain unchanged. The instance is in a heterogeneous state. The following figure shows the architecture level and code level for *Instance1* and members:

Instance1

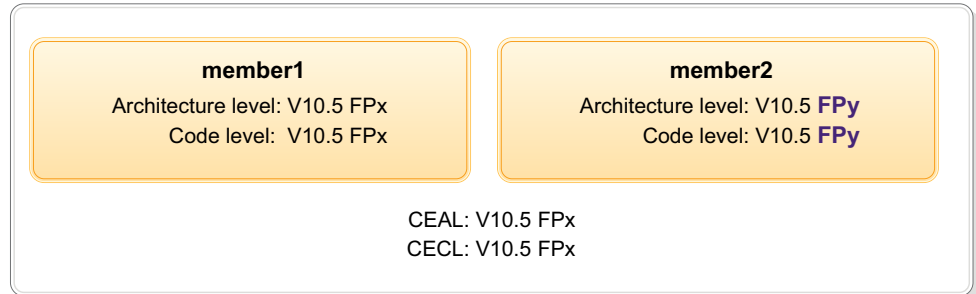


Figure 12. Architecture and code level values after member2 is updated to Version 10.5 FPy

After an online fix pack update is applied on *member1* to Version 10.5 FPy, note that the code level and architecture level for *member1* changed. However, the CECL and CEAL remain unchanged. The instance remains in a heterogeneous state. The following figure shows the architecture level and code level for *Instance1* and members:

Instance1

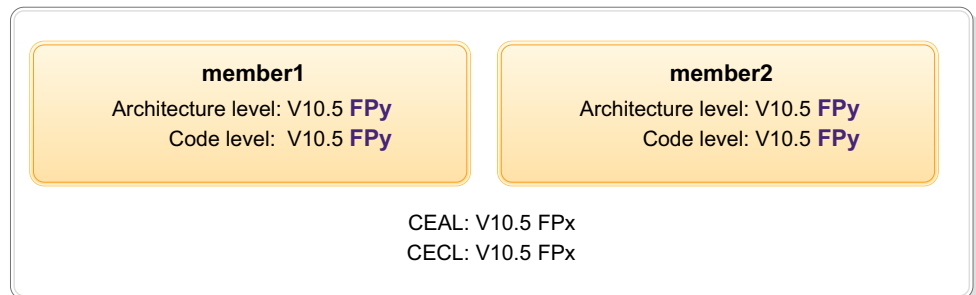


Figure 13. Architecture and code level values after member1 is updated to Version 10.5 FPy

After the online update is committed to Version 10.5 fix pack FPy, the instance CEAL and the CECL changed. The instance is in a homogeneous state again. The following figure shows the architecture level and code level for *Instance1* and members:

Instance1

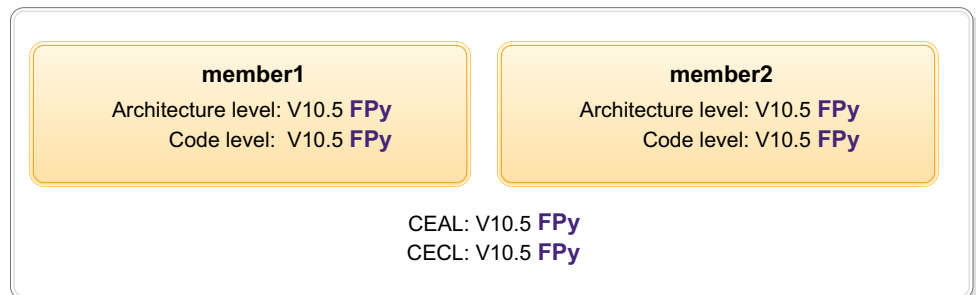


Figure 14. Architecture and code level values after online update is committed to Version 10.5 FPy

Suppose that you have a special build with a code level of Version 10.5 FPy and an architecture level of Version 10.5 FPx, then after committing the online update the CEAL remains unchanged. The following figure shows the architecture level and code level for *Instance1* and members:

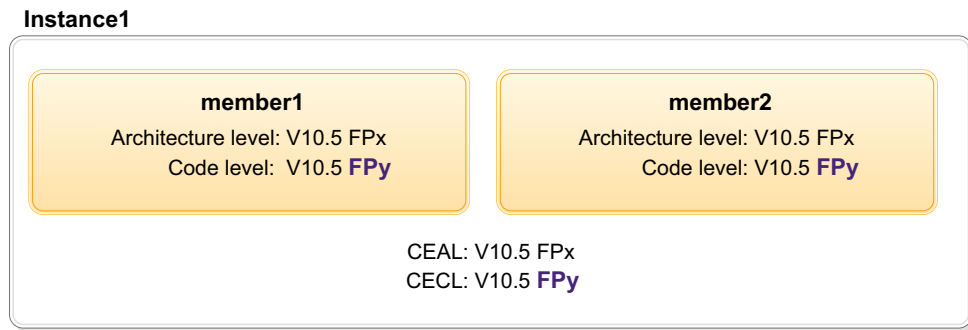


Figure 15. Architecture and code level values after online update is committed to FPy.

Database and instance operations affected by an online fix pack update in progress

Certain database or instance operations are restricted while an online fix pack update is in progress.

After you install the DB2 Version 10.5 release in a DB2 pureScale environment, all members and CFs have the same code level. The instance is in a homogeneous state.

While an online fix pack update is in progress, members and CFs can have a different code level than the current effective code level. The instance is then in a heterogeneous state.

The following restrictions apply to instances in heterogeneous state:

- You cannot add a member or CF.
- You cannot create a database from a member that has a code level higher than the current effective code level.
- You cannot drop the whole cluster.
- You cannot update the monitor event tables with the EVMON_UPGRADE_TABLES procedure.
- You cannot update the database system catalog with the **db2updv105** command.

Use the ENV_GET_INSTANCE_CODE_LEVELS table function to obtain information about the instance, members, or CFs during online fix pack updates.

You can also run the **db2pd -ruStatus** command to know about the instance state.

Applying fix packs in DB2 pureScale environments

Maintain your DB2 pureScale environment at the latest fix pack level to ensure all the updates and fixes for problems are available for your DB2 software. To install a fix pack successfully, perform all of the necessary preinstallation and postinstallation tasks.

About this task

A DB2 fix pack contains updates, fixes for problems (authorized program analysis reports, or "APARs") found during testing at IBM, and fixes for problems that are

reported by customers. For a complete list of the fixes that are contained in each fix pack, see <http://www.ibm.com/support/docview.wss?uid=swg21633303>.

Fix packs are cumulative; the latest fix pack for any given version of DB2 software contains all of the updates from previous fix packs for the same version of DB2 software.

On IBM DB2 pureScale environments, a fix pack image can be applied offline or online.

Procedure

To install a fix pack:

1. Check fix pack prerequisites.
2. Perform the tasks in “Preparing to install a fix pack.”
3. Perform any of the following tasks to install the fix pack:
 - “Installing online fix pack updates to a higher code level on a DB2 pureScale instance” on page 182
 - “Installing online fix pack updates to a higher code level in a HADR environment” on page 188
 - “Installing online fix pack updates to a higher code level in a GDPC environment” on page 191
 - “Installing offline fix pack updates to a DB2 pureScale instance (simplified method)” on page 194
4. Perform the “Post-installation tasks for fix packs (Linux and UNIX)” on page 195.
5. Apply the appropriate DB2 pureScale Feature license.

Preparing to install a fix pack

In order to install a fix pack, you must first download and uncompress the fix pack. You must also stop the DB2 instances that you plan to update to the new fix pack level.

Before you begin

If a IBM PowerHA® SystemMirror for AIX cluster is running, you cannot perform a IBM Tivoli System Automation for Multiplatforms (SA MP) installation, upgrade, or update because SA MP bundles Reliable Scalable Cluster Technology (RSCT) file sets that are dependent on PowerHA SystemMirror. You can skip the SA MP installation when you install a fix pack. For information about installing or upgrading SA MP using a PowerHA SystemMirror cluster, see the white paper entitled “Upgrade guide for DB2 Servers in HACMP™ Environments”, which is available from the IBM Support and downloads website (<http://www.ibm.com/support/docview.wss?uid=swg21045033>).

Procedure

To prepare to install a fix pack:

1. Check the fix pack prerequisites. See “Checking fix pack prerequisites” on page 178.
- 2.

Important:

If you are currently using column-organized tables with CHAR or GRAPHIC columns, you must follow the instructions in “Steps to determine whether APAR IV53366 is affecting your DB2 databases” (see <http://www-01.ibm.com/support/docview.wss?uid=swg21663252>) before applying Fix Pack 3, or DB2 Cancun Release 10.5.0.4.

3. Get the fix pack.
4. Uncompress the fix pack. For more information, see “Uncompressing fix packs (Linux and UNIX)” on page 180
5. Optional: Reduce the size of the fix pack.

You can use the **db2iprune** command to reduce the size of your DB2 fix pack installation image. Before installing a pruned fix pack, you must ensure that the pruned fix pack image contains at least the same components that are in the DB2 copy. If you prune too much from the fix pack image, the fix pack installation fails.

6. If you already have DB2 database products installed in the selected installation path:
 - a. Optional: Back up your current configuration and diagnostic information.
Gathering diagnostic information might be helpful when you are about to install a fix pack. This information will help to diagnose any problems that might arise after the installation. See “Backing up DB2 server configuration and diagnostic information” in *Upgrading to DB2 Version 10.5*.
 - b. Stop all DB2 processes For more information, see “Stopping all DB2 processes (Linux and UNIX)” on page 180.

Checking fix pack prerequisites

Ensure that you meet all of the software, hardware and operating system prerequisites before you download a fix pack.

Procedure

Before downloading a fix pack, perform the following steps:

1. Review the Flashes and open Authorized Problem Analysis Reports (APARs) on the DB2 for Linux, UNIX and Windows product support website:
http://www.ibm.com/software/data/db2/support/db2_9/.
Open APARs describe defects that are not yet addressed in a fix pack, and they might contain workarounds that will impact your use of the database system. For a list of open DB2 for Linux, UNIX and Windows APARs, refer to:
<http://www.ibm.com/support>
2. If you already have a DB2 database product installed and have obtained special fixes, contact IBM support to determine whether you need an updated version of the fixes before you install a fix pack.
This helps to ensure that your system is in a consistent state and that no special fixes are lost.
3. Ensure that your system meets all of the installation requirements. Run the **db2prereqcheck** command to determine if your system satisfies the DB2 installation prerequisites. .
This action prevents technical problems that might occur after the installation and configuration of the fix pack.
4. For AIX, if DB2 database products already exist on the system and the Trusted Computing Base (TCB) is enabled, ensure it is in a consistent state.

Ensure that the TCB is in a consistent state for any file or directory related to a DB2 instance, the DB2 Administration Server, and installed DB2 files. The TCB can be in an inconsistent state if, for example, DB2 instances were removed without using the **db2idrop** command.

To check the TCB state, enter the following command as root:

```
tcbck -n ALL
```

Refer to AIX documentation for details regarding the Trusted Computing Base.

5. Ensure that you have sufficient file system free space to download and extract the fix pack.

On Linux and UNIX, you need to have a file system with two gigabyte of free space to contain the .tar.gz file and the uncompressed installation image. If you also plan to install a national language fix pack, you need up to two gigabytes of free space. For DB2 pureScale feature, verify that sufficient file system free space is available on all members and cluster caching facilities (CFs) in the cluster.

6. Ensure that you have the free space required to install the fix pack.

The free space must be available in the location of the DB2 copy that you are updating or in the location where you plan to create a new DB2 installation.

- On Linux and UNIX:
 - If you do not already have a DB2 database product installed and you are using the fix pack to create a new installation, ensure that an appropriate amount of disk space is available and allocate memory accordingly..
 - If you already have DB2 database product installed, the space required to install the fix pack is equal to the space consumed by the existing DB2 database products. This space is only required temporarily during the fix pack installation process.

To determine the space used by the existing DB2 database products, perform the command:

```
du -k -s DB2DIR
```

where *DB2DIR* represents the location where the DB2 database product is installed.

7. Optional: Review the “Product overviews” in the *DB2 Information Center*.

Results

Once you have ensured that you meet all of these prerequisites, complete the remainder of the necessary tasks before installing a fix pack.

Getting fix packs

After checking the fix pack prerequisites, download the fix pack.

Before you begin

Check the fix pack prerequisites. See “Checking fix pack prerequisites” on page 178.

Procedure

To get a fix pack:

1. Determine which fix pack you need.

In general, choose the most recent fix pack to avoid encountering problems caused by software defects that are already known and corrected.

2. Locate the fix pack on the DB2 for Linux, UNIX, and Windows product support website: www.ibm.com/support/docview.wss?rs=71&uid=swg27007053.

Ensure that you choose the appropriate fix pack for your operating system. Choose between DB2 database product-specific fix packs and universal fix packs.

3. Download the fix pack.

In most cases, you can choose to access the FTP folder directly or use a Java applet called Download Director to download the files.

On Linux and UNIX operating systems, there must not be any spaces in the directory path where you plan to download and uncompress the fix pack. If there are spaces in the directory path, the installation fails. For example, make sure that your directory path resembles the following example:

/home/DB2FixPack/FP1/. It must not resemble the following: /home/DB2
FixPack/FP1/.

What to do next

After you successfully download the fix pack, perform the remaining preparatory steps before installing a fix pack. See “Preparing to install a fix pack” on page 177.

Uncompressing fix packs (Linux and UNIX)

All fix pack installation images on the FTP site are compressed using **gzip**. Before you can install a fix pack, you must copy the image to a temporary directory and use **gunzip** and **tar** to extract the fix pack installation image.

Procedure

To uncompress a fix pack installation image, perform the following steps:

1. Copy the gzipped image to a temporary location.
2. Change to the directory where you copied the image.
3. Enter the following command to uncompress the file:

```
gunzip -c filename.tar.gz | tar -xvf -
```

where *filename* is the fix pack you are installing.

Note: **gunzip** is part of the AIX 5L™ default installation setup. If you do not have **gunzip**, install the rpm.rte fileset from the AIX 5L installation media. The rpm.rte fileset contains **gunzip**. You can also download **gzip** for AIX 5L from the website: www.ibm.com/servers/aix/products/aixos/linux/rpmgroups.html

Stopping all DB2 processes (Linux and UNIX)

Before installing a fix pack, if there are DB2 database products installed in the selected installation path, you must stop all of the DB2 processes. If you have multiple DB2 copies, stop only the DB2 processes that are associated with the copy that you are updating.

Procedure

To stop all DB2 processes, perform the following steps:

1. Log on as root.

2. Determine which instances are associated with the DB2 copy. Issue the command:

```
DB2DIR/instance/db2ilist
```

where *DB2DIR* represents the location where the DB2 copy is installed.

3. Run the following commands for each instance in the DB2 copy:

```
su - iname
. $HOME/sql1lib/db2profile
db2 force applications all
db2 terminate
db2stop
db2licd -end      # run at each physical partition
exit
```

where *iname* represents the instance owner name. If you are an PowerHA SystemMirror user, you must use the **ha_db2stop** command to stop DB2 instead of the **db2stop** command. If you use the **db2stop** command instead of the **ha_db2stop** command, you will trigger a failure event.

4. If the DB2 Administration Server (DAS) belongs to the DB2 copy that you are updating, stop the DAS:

```
su - aname
. $HOME/das/dasprofile
db2admin stop
exit
```

where *aname* represents the DAS owner name.

Note: Since there can only be one DAS on the system, this step affects all other DB2 copies on the system.

5. Optional: On AIX, run **slibclean** to unload unused shared libraries from memory before installation:

```
/usr/sbin/slibclean
```

6. Disable the fault monitor processes. To stop the Fault Monitor Daemon, issue the command:

```
DB2DIR/bin/db2fm -i iname -D
```

where *DB2DIR* is the location where the DB2 copy is installed and *iname* represents the instance owner name. The command must be performed once for each instance in the DB2 copy.

7. If the Fault Monitor Coordinator (FMC) is started, prevent your instances from auto-starting:

- a. To determine whether the FMC is started, issue the command:

```
DB2DIR/bin/db2fmcu
```

where *DB2DIR* is the location where the DB2 copy is installed. If the FMC is started, you will see output similar to the following: FMC: up: PID = 3415 . If the FMC is disabled, the output from the **db2fmcu** command will be: FMC: down.

- b. If the FMC is started, determine whether any instances are configured to auto-start after each system restart. Issue the command:

```
DB2DIR/instance/db2iset -i iname -all
```

where *DB2DIR* is the location where the DB2 copy is installed and *iname* represents the instance owner name. The command must be performed once

for each instance in the DB2 copy. If the output from the **db2iset** command includes the following, it means that the instance is configured to auto-start:DB2AUTOSTART=YES

- c. Prevent the instances from auto-starting. Issue the command:

```
DB2DIR/instance/db2iauto -off iname
```

where *DB2DIR* is the location where the DB2 copy is installed and *iname* represents the instance owner name. After you have completed the fix pack installation, you can re-start instance auto-start:

```
DB2DIR/instance/db2iauto -on iname
```

8. Ensure all DB2 interprocess communications are cleaned for the instance to be updated. As the instance owner, run the following command at each physical partition:

```
$HOME/sql1lib/bin/ipclean
```

Installing online fix pack updates to a higher code level on a DB2 pureScale instance

Install online fix pack updates on members and cluster caching facilities (CFs) one at a time to update a DB2 pureScale instance to a fix pack or special build with a higher code level while the instance remains available.

Before you begin

- Ensure that you meet all of the requirements before you install a fix pack. For more information, see “Preparing to install a fix pack” in *Installing DB2 Servers*.
- Ensure that you have root user authority and instance owner authority.
- Ensure that online fix pack updates are supported between the DB2 version that is installed on your DB2 pureScale instance and the DB2 version of the fix pack or special build by issuing the **installFixPack -show_level_info** command from the new fix pack or special build image location `<new_fix_pack_image_location>/<product>/installFixPack`. The following text displays the sample command output:

```
Code level = Version:10 Release:5 Modification:0 Fixpack:1
Architecture level = Version:10 Release:5 Modification:0 Fixpack:1
Section level = Version:10 Release:5 Modification:0 Fixpack:1
```

```
Supports online update = Yes
```

```
Minimum committed code level required for online install =
Version:10 Release:5 Modification:0 Fixpack:0
```

```
The execution completed successfully.
```

```
For more information see the DB2 installation log at "/tmp/installFixPack.log.24174".
DBI1070I Program installFixPack completed successfully.
```

About this task

You can update one or more members or cluster caching facilities (CFs) while the remaining members and CFs continue to process transactions. You must update all members and CFs before you can commit the changes and update the DB2 pureScale instance. If the members and CFs are located on the same host, you must apply the fix pack update only once per host.

Procedure

To install an online fix pack update on a DB2 pureScale instance:

1. Uncompress the fix pack or special build image to a directory that is accessible to all members and CF hosts.
2. Apply the fix pack online on each of the members one at a time:
 - a. Log on to the member server with root user authority.
 - b. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. In addition, the *FP-install-path* must be a different path than the currently installed DB2 version. The **-online** parameter is the default and can be omitted.

If you receive an error message regarding the current version of IBM Tivoli System Automation for Multiplatforms (SA MP), IBM General Parallel File System (GPFS™), or IBM Reliable Scalable Cluster Technology (RSCT), this error occurs because the current version of the specified software is different from the version that was previously installed by the DB2 installer. In this case, you can force the installFixPack command to continue the fix pack installation process and force the version update of the specified software. For example, if the error message indicates that the current version of RSCT is different from the RSCT version that was previously installed by the DB2 installer, reissue the installFixPack command with the **-f RSCT** parameter:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t trace-file-name -f RSCT
```

However, when you use the **-f** parameter to force the fix pack installation to continue, later if you decide to move back to the previous version of the fix pack by canceling the online fix pack update, the DB2 installer is not able to revert the specified software (RSCT, SA MP or GPFS) to the same earlier version.

Ensure you apply the fix packs to each member before you continue with the next step.

3. Apply the fix pack online on the secondary CF:
 - a. Log on to the secondary CF server with root user authority.
 - b. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. In addition, the *FP-install-path* must be a different path than the currently installed DB2 version. The **-online** parameter is the default and can be omitted.

4. Apply the fix pack online on the primary CF:
 - a. Ensure that the secondary CF of your DB2 pureScale instance is in PEER state by issuing the following command as the instance owner:

```
db2instance -list
```

The secondary CF must be in PEER state before updating the primary CF.

If running **db2instance -list** shows that the secondary CF is in CATCHUP state, you can check the catch up progress percent by querying the DB2_CF administrative view. For example:

```
db2 "SELECT ID as CF_ID, varchar(CURRENT_HOST,21) AS HOST, varchar(STATE,14) AS CF_STATE FROM SYSIBMADM.DB2_CF"
```

CF_ID	HOST	CF_STATE
-----	-----	-----


```

128 cfserver56      CATCHUP(79%)
129 cfserver54      PRIMARY
2 record(s) selected.

```

The CATCHUP percentage value represents the amount to which the secondary CF is caught up to the current state of the primary CF.

b. Log on to the primary CF server with root user authority.

c. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. The **-online** parameter is the default and can be omitted.

5. Determine whether the online fix pack update was successful on all members and CFs by issuing the following command:

```
media-dir/installFixPack -check_commit -I instance-name
```

If the command is successful, you can continue with the next step.

6. Commit the online fix pack update so that your DB2 pureScale instance is updated to the new fix pack level by issuing the following command:

```
media-dir/installFixPack -commit_level -I instance-name -l log-file-name -t trace-file-name
```

7. Verify that your instance and databases show the new committed fix pack level by issuing the following command as an instance user:

```
db2pd -ruStatus
```

8. If you want to use capabilities specific to the fix pack, update the system catalog objects in your databases:

a. Log on as the instance owner.

b. For each database in the instance, issue the **db2updv105** command as follows:

```
db2updv105 -d db-name
```

Example

The following example illustrates how to use online fix pack updates to update a DB2 pureScale instance that is called *Instance1* from Version 10.5 Fix Pack 3 to DB2 Cancun Release 10.5.0.4. *Instance1* has two CFs called *cfserver56* and *cfserver54* and two members that are called *mbserver53* and *mbserver55*. The command output text is formatted for better reading by removing blanks or adding lines.

1. The administrator updates *mbserver53*:

a. Log on to the DB2 database server with root user authority.

b. Uncompress the fix pack or special build image in the */local/tmp/FP4image* directory.

c. Issue the **installFixPack** command as follows:

```
/local/tmp/FP4image/installFixPack -p /opt/ibm/db2/V10.5FP4 -I Instance1
                                   -online -l /tmp/FP4log_mbserver53
```

d. Issue the **db2pd -ruStatus** command to verify that code level changed. The following text displays the command output:

```

ROLLING UPDATE STATUS:  Disk Value                                Memory Value
Record Type             = INSTANCE
ID                       = 0
Code Level               = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
Architecture Level     = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
State                   = [NONE]
Last updated            = 2013/04/18:02:58:58

Record Type             = MEMBER

```



```

ID = 0
Code Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000) V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CECL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000) V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CEAL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000) V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated = 2013/04/18:07:59:48

mbserver53.domain.com: db2pd -ruStatus -localhost ... completed ok

Record Type = MEMBER
ID = 1
Code Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CECL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CEAL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000) V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated = 2013/04/18:05:12:20

mbserver55.domain.com: db2pd -ruStatus -localhost ... completed ok

Record Type = CF
ID = 128
Code Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000400000000) Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000400000000) Not Applicable
Last updated = 2013/04/18:07:31:14

Record Type = CF
ID = 129
Code Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000400000000) Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000400000000) Not Applicable
Last updated = 2013/04/18:07:25:55

```

2. The administrator updates the secondary CF *cfserver56*:

- Log on to the *cfserver56* with root user authority.
- Uncompress the fix pack or special build image in the `/local/tmp/FP4` image directory.
- Issue the **installFixPack** command as follows:

```

/local/tmp/FP4image/installFixPack -online -p /opt/ibm/db2/V10.5FP4 -I Instance1 -t trace-file-name
-l /tmp/FP4log_cfserver56

```

- Issue the **db2pd -ruStatus** command to verify that the code level changed to DB2 Cancun Release 10.5.0.4. The following text displays the command output:

```

ROLLING UPDATE STATUS: Disk Value Memory Value

Record Type = INSTANCE
ID = 0
Code Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) Not Applicable
State = [NONE]
Last updated = 2013/04/18:02:58:58

Record Type = MEMBER
ID = 0
Code Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CECL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CEAL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000) V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated = 2013/04/18:01:57:35

mbserver53.domain.com: db2pd -ruStatus -localhost ... completed ok

Record Type = MEMBER
ID = 1
Code Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CECL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CEAL = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000) V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000) V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated = 2013/04/18:05:12:20

mbserver55.domain.com: db2pd -ruStatus -localhost ... completed ok

Record Type = CF
ID = 128
Code Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000400000000) Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000400000000) Not Applicable
Last updated = 2013/04/18:02:58:52

Record Type = CF
ID = 129
Code Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000) Not Applicable
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000) Not Applicable
Last updated = 2013/04/18:07:25:55

```

3. The administrator updates the primary CF server (*cfserver54*):

- The administrator verifies that the secondary CF server (*cfserver56*) is in PEER state by logging in as instance owner and issuing the following command:

```
db2instance -list
```

The command output shows that the *cfserver56* is in PEER state:

ID	TYPE	STATE	HOME_HOST	CURRENT_HOST	ALERT	PARTITION_NUMBER	LOGICAL_PORT	NETNAME
0	MEMBER	STARTED	mbserver53	mbserver53	NO	0	0	0 mbserver53-ib0
1	MEMBER	STARTED	mbserver55	mbserver55	NO	0	0	0 mbserver55-ib0
128	CF	PRIMARY	cfserver54	cfserver54	NO	-	-	0 cfserver54-ib0
129	CF	PEER	cfserver56	cfserver56	NO	-	-	0 cfserver56-ib0

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
cfserver56	ACTIVE	NO	NO
cfserver54	ACTIVE	NO	NO
mbserver55	ACTIVE	NO	NO
mbserver53	ACTIVE	NO	NO

- Log on to the DB2 database server with root user authority.
- Uncompress the fix pack or special build image in the /local/tmp/FP4image directory.
- Issue the **installFixPack** command as follows:

```
/local/tmp/FP4image/installFixPack -p /opt/ibm/db2/V10.5FP4 -I Instance1
-online -l /tmp/FP4log_cfserver54
```
- Issue the **db2pd -ruStatus** command to verify that the code level changed.
The following text displays the command output:

```
ROLLING UPDATE STATUS: Disk Value                                Memory Value

Record Type      = INSTANCE
ID               = 0
Code Level       = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
State            = [NONE]
Last updated     = 2013/04/18:02:58:58


Record Type      = MEMBER
ID               = 0
Code Level       = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CECL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CEAL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level    = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)  V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated     = 2013/04/18:01:57:35


mbserver53.domain.com: db2pd -ruStatus -localhost ... completed ok


Record Type      = MEMBER
ID               = 1
Code Level       = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CECL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CEAL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level    = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)  V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated     = 2013/04/18:05:12:20


mbserver55.domain.com: db2pd -ruStatus -localhost ... completed ok


Record Type      = CF
ID               = 128
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Last updated     = 2013/04/18:07:31:14


Record Type      = CF
ID               = 129
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Last updated     = 2013/04/18:07:25:55
```

- After all members are updated, the administrator verifies that all members and CFs have the same architecture and code level by issuing the **db2pd -ruStatus** command. The following text displays the command output:

```
ROLLING UPDATE STATUS: Disk Value                                Memory Value

Record Type      = INSTANCE
ID               = 0
Code Level       = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
State            = [NONE]
Last updated     = 2013/04/18:02:58:58


Record Type      = MEMBER
ID               = 0
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CECL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CEAL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level    = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)  V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated     = 2013/04/18:07:59:48


mbserver53.domain.com: db2pd -ruStatus -localhost ... completed ok


Record Type      = MEMBER
ID               = 1
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CECL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CEAL             = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level    = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)  V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated     = 2013/04/18:09:24:18
```

```
mbsrver55.domain.com: db2pd -ruStatus -localhost ... completed ok
```

```
Record Type      = CF
ID               = 128
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Last updated     = 2013/04/18:07:31:14

Record Type      = CF
ID               = 129
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Last updated     = 2013/04/18:07:25:55
```

- The administrator determines that online fix pack update was successful in all members and CFs by issuing the following command:

```
/local/tmp/FP4image/installFixPack -check_commit -I Instance1
                                     -l /tmp/checkcommit.log
```

The following command output shows that online update was successful and that all members and CF are ready for issuing the commit of the fix pack update:

```
DBI1446I The installFixPack command is running.
```

The pre-commit verification process for an online fix pack update has started....
The checks for the pre-commit verification process have been completed successfully.

If you perform a commit, the new level will be =
Version:10 Release:5 Modification:0 Fixpack:4

The execution completed successfully.

For more information see the DB2 installation log at "/tmp/FP4_checkcommit.log".
DBI1070I Program installFixPack completed successfully.

- The administrator then commits the online fix pack updates so that *Instance1* is updated to the new fix pack level by issuing the following command:

```
/local/tmp/FP4image/installFixPack -commit_level -I Instance1
                                     -l /tmp/FP4_commit.log
```

The following command output shows the commit was successful:

```
DBI1446I The installFixPack command is running.
```

The execution completed successfully.

For more information see the DB2 installation log at "/tmp/FP4_commit.log".
DBI1070I Program installFixPack completed successfully.

- The administrator verifies that *Instance1*, members, and CFs show the same new committed code level and architecture level by issuing the **db2pd -ruStatus** command. The following text displays the command output:

```
ROLLING UPDATE STATUS: Disk Value                                Memory Value

Record Type      = INSTANCE
ID               = 0
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
State            = [NONE]
Last updated     = 2013/04/18:08:58:21

Record Type      = MEMBER
ID               = 0
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CECL             = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CEAL             = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
Section Level    = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)  V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated     = 2013/04/18:07:59:48

mbsrver53.domain.com: db2pd -ruStatus -localhost ... completed ok

Record Type      = MEMBER
ID               = 1
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CECL             = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
CEAL             = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)
Section Level    = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)  V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000000000000)
Last updated     = 2013/04/18:09:24:18

mbsrver55.domain.com: db2pd -ruStatus -localhost ... completed ok

Record Type      = CF
ID               = 128
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000)  Not Applicable
```

```

Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000) Not Applicable
Last updated      = 2013/04/18:07:31:14

Record Type      = CF
ID               = 129
Code Level       = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000) Not Applicable
Architecture Level = V:10 R:5 M:0 F:4 I:0 SB:0 (0x0A05000400000000) Not Applicable
Last updated      = 2013/04/18:07:25:55

```

Installing online fix pack updates to a higher code level in a HADR environment

In an HADR environment, install online fix pack updates on members and cluster caching facilities (CFs) one at a time to update a DB2 pureScale instance to a fix pack or special build with a higher code level while the instance remains available.

Before you begin

- Ensure that you meet all of the requirements before you install a fix pack. For more details, see “Preparing to install a fix pack” in *Installing DB2 Servers*.
- Ensure that you have root user authority and instance owner authority.
- Ensure that online fix pack updates are supported between the DB2 version that is installed on your DB2 pureScale instance and the DB2 version of the fix pack or special build by issuing the **installFixPack -show_level_info** command. The following text displays the command output:

```

Code level          = Version:10 Release:5 Modification:0 Fixpack:4
Architecture level  = Version:10 Release:5 Modification:0 Fixpack:4
Section level       = Version:10 Release:5 Modification:0 Fixpack:4

```

Supports online update = Yes

```

Minimum committed code level required for online install =
Version:10 Release:5 Modification:0 Fixpack:1

```

The execution completed successfully.

For more information see the DB2 installation log at "/tmp/installFixPack.log.8541".
DBI1070I Program installFixPack completed successfully.

About this task

In an HADR environment, you can update one or more members or cluster caching facilities (CFs) while the remaining members and CFs continue to process transactions. You must update all members and CFs in both the primary and standby clusters before you can commit the changes and update the DB2 pureScale instance.

Procedure

To install an online fix pack update on a DB2 pureScale instance in a HADR environment:

1. Install the online fix pack update on each of the members in the standby cluster:

- a. Log on to the member server in the standby cluster as root user.
- b. Uncompress the fix pack or special build image in a directory that is accessible to the instance owner and root user.

To reduce interruption to standby log replay, install the online fix pack update on all members except the current replay member. Then install the online fix pack update on the current replay member.

- c. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -online -I instance-name -l log-file-name -t /tmp/trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts.

2. Install the online fix pack update on the secondary CF server in the standby cluster:
 - a. Log on to the secondary CF server in the standby cluster as root user.
 - b. Uncompress the fix pack or special build image in a local directory that is accessible to the instance owner and root user.
 - c. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t /tmp/trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. The **-online** parameter is the default and can be omitted.

3. Install the online fix pack update on the primary CF in the standby cluster:
 - a. Ensure that the secondary CF in the standby cluster is in PEER state by issuing the following command as the instance owner:
db2instance -list

The secondary CF must be in PEER state before you update the primary CF.

- b. Log on to the primary CF server in the standby cluster as root user.
 - c. Uncompress the fix pack or special build image in a local directory that is accessible to the instance owner and root user.
 - d. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -online -I instance-name -l log-file-name -t /tmp/trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts.

4. Determine whether the online fix pack update was successful in all members and CFs in the standby cluster by issuing the following command:

```
media-dir/installFixPack -check_commit -I instance-name
```

If the command output shows any problems, fix them before you continue with the next step.

5. Install the online fix pack update on each of the members in the primary cluster:
 - a. Log on to the member server in the primary cluster as root user.
 - b. Uncompress the fix pack or special build image in a directory that is accessible to the instance owner and root user.
 - c. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -online -I instance-name -l log-file-name -t /tmp/trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts.

6. Install the online fix pack update on the secondary CF server in the primary cluster:
 - a. Log on to the secondary CF server in the primary cluster as root user.

- b. Uncompress the fix pack or special build image in a local directory that is accessible to the instance owner and root user.
- c. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t /tmp/trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. The **-online** parameter is the default and can be omitted.

7. Install the online fix pack update on the primary CF server in the primary cluster:

- a. Ensure that the secondary CF in the primary cluster is in PEER state by issuing the following command as the instance owner:

```
db2instance -list
```

The secondary CF must be in PEER state before you update the primary CF.

- b. Log on to the primary CF server in the primary cluster as root user.
- c. Uncompress the fix pack or special build image in a local directory that is accessible to the instance owner and root user.
- d. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -online -I instance-name -l log-file-name -t /tmp/trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts.

8. Determine whether the online fix pack update was successful in all members and CFs in the primary cluster by issuing the following command:

```
media-dir/installFixPack -check_commit -I instance-name
```

If the command output shows any problems, fix them before you continue with the next step.

9. Commit the online fix pack update in the standby cluster so that your DB2 pureScale instance is updated to the new fix pack level by issuing the following command:

```
media-dir/installFixPack -commit_level -I instance-name -l log-file-name -t /tmp/trace-file-name
```

10. Verify that your instance and databases show the new committed fix pack level in the standby cluster by issuing the following command:

```
db2pd -ruStatus
```

For example, if you are updating the fix pack level from DB2 Version 10.5 Fix Pack 1 to DB2 Version 10.5 Fix Pack 3, the following text displays the command output.

ROLLING UPDATE STATUS: Disk Value		Memory Value
Record Type	= INSTANCE	
ID	= 0	
Code Level	= V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)	Not Applicable
Architecture Level	= V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)	Not Applicable
State	= [NONE]	
Last updated	= 2013/11/14:04:38:47	
Record Type	= MEMBER	
ID	= 0	
Code Level	= V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)	V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CECL	= V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)	V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level	= V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)	V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CEAL	= V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)	V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level	= V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)	V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
State	= [NONE]	
Last updated	= 2013/11/13:03:39:33	

```

coralpi19c.torolab.ibm.com: db2pd -rustatus -localhost ... completed ok

Record Type      = MEMBER
ID               = 1
Code Level       = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CECL            = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
CEAL            = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
Section Level    = V:10 R:5 M:0 F:0 I:0 SB:0 (0x0A05000300000000)  V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)
State            = [NONE]
Last updated     = 2013/11/13:00:20:02

coralpi19d.torolab.ibm.com: db2pd -rustatus -localhost ... completed ok

Record Type      = CF
ID               = 128
Code Level       = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
State            = [NONE]
Last updated     = 2013/11/13:03:39:34

Record Type      = CF
ID               = 129
Code Level       = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
Architecture Level = V:10 R:5 M:0 F:3 I:0 SB:0 (0x0A05000300000000)  Not Applicable
State            = [NONE]
Last updated     = 2013/11/13:00:20:03

```

In the command output, V:10 R:5 M:0 F:3 specifies the updated fix pack level namely DB2 Version 10.5 Fix Pack 3.

11. Commit the online fix pack update in the primary cluster so that your DB2 pureScale instance is updated to the new fix pack level by issuing the following command:

```
media-dir/installFixPack -commit_level -I instance-name -l log-file-name -t /tmp/trace-file-name
```

12. Verify that your instance and databases show the new committed fix pack level in the primary cluster by issuing the following command:

```
db2pd -ruStatus
```

The command output displays text that is similar to the example provided in step 10.

13. If you want to use capabilities specific to the fix pack, update the system catalog objects in your databases in the primary cluster:
 - a. Log on as the instance owner.
 - b. For each database in the instance, issue the **db2updv105** command as follows:

```
db2updv105 -d db-name
```

Installing online fix pack updates to a higher code level in a GDPC environment

In a geographically dispersed DB2 pureScale cluster (GDPC) environment, install online fix pack updates on members and cluster caching facilities (CFs) one at a time to update a DB2 pureScale instance to a fix pack or special build with a higher code level while the instance remains available.

Before you begin

- Ensure that you meet all of the requirements before you install a fix pack. For more details, see “Preparing to install a fix pack” in *Installing DB2 Servers*.
- Ensure that you have root user authority and instance owner authority.
- Ensure that online fix pack updates are supported between the DB2 version that is installed on your DB2 pureScale instance and the DB2 version of the fix pack or special build by issuing the **installFixPack -show_level_info** command. The following text displays the command output:

```

Code level      = Version:10 Release:5 Modification:0 Fixpack:4
Architecture level = Version:10 Release:5 Modification:0 Fixpack:4
Section level   = Version:10 Release:5 Modification:0 Fixpack:4

```

```
Supports online update = Yes
```



```
Minimum committed code level required for online install =  
Version:10 Release:5 Modification:0 Fixpack:1
```

The execution completed successfully.

For more information see the DB2 installation log at "/tmp/installFixPack.log.8541".
DB11070I Program installFixPack completed successfully.

Note: IBM support for a geographically dispersed DB2 pureScale cluster (GDPC) implementation requires engagement of IBM Lab Services for separately charged initial installation services. Contact your IBM sales representative for details.

About this task

In a GDPC environment, you can update one or more members or cluster caching facilities (CFs) while the remaining members and CFs continue to process transactions. You must update all members and CFs, and the tiebreaker host, before you can commit the changes and update the DB2 pureScale instance. If the members and CFs are located on the same host, you must apply the fix pack update only once per host.

Restrictions

Procedure

To install an online fix pack update in a GDPC environment:

1. Uncompress the fix pack or special build image to a directory that is accessible to all members and CF hosts.
2. Apply the fix pack online on each of the members one at a time:
 - a. Log on to the member server with root user authority.
 - b. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. The **-online** parameter is the default and can be omitted.

3. Apply the fix pack online on the secondary CF:
 - a. Log on to the secondary CF server with root user authority.
 - b. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. The **-online** parameter is the default and can be omitted.

4. Apply the fix pack online on the primary CF:
 - a. Ensure that the secondary CF of your DB2 pureScale instance is in PEER state by issuing the following command as the instance owner:

```
db2instance -list
```

The secondary CF must be in PEER state before updating the primary CF.

If running **db2instance -list** shows that the secondary CF is in CATCHUP state, you can check the catch up progress percent by querying the DB2_CF administrative view. For example:


```
db2 "SELECT ID as CF_ID, varchar(CURRENT_HOST,21) AS HOST, varchar(STATE,14) AS CF_STATE FROM SYSIBMADM.DB2_CF"
```

CF_ID	HOST	CF_STATE
128	cfserver56	CATCHUP (79%)
129	cfserver54	PRIMARY

2 record(s) selected.

The CATCHUP percentage value represents the amount to which the secondary CF is caught up to the current state of the primary CF.

- b. Log on to the primary CF server with root user authority.
- c. Issue the **installFixPack** command as follows:

```
media-dir/installFixPack -p FP-install-path -I instance-name -online -l log-file-name -t trace-file-name
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts. The **-online** parameter is the default and can be omitted.

5. Update the tiebreaker host:

- a. Enter maintenance mode from the old code level. For example:

```
<OLD-FP-install-path>/bin/db2cluster -cm -enter -maintenance
<OLD-FP-install-path>/bin/db2cluster -cfs -enter -maintenance
```

- b. To update the tiebreaker host, run **installFixPack** command from the target fix pack image.

```
media-dir/installFixPack -b <OLD-FP-install-path> -p <FP-install-path> -L
```

Where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts.

For example:

```
media-dir/installFixPack -b /opt/ibm/db2/V10.5/ -p /opt/ibm/db2/V10.5fp4 -L
```

- c. Exit maintenance mode from the new code level: For example:

```
<FP-install-path>/bin/db2cluster -cm -exit -maintenance
<FP-install-path>/bin/db2cluster -cfs -exit -maintenance
```

6. Determine whether the online fix pack update was successful on all members and CFs by issuing the following command:

```
media-dir/installFixPack -check_commit -I instance-name
```

If the command is successful, you can continue with the next step.

7. Commit the online fix pack update so that your DB2 pureScale instance is updated to the new fix pack level by issuing the following command:

```
media-dir/installFixPack -commit_level -I instance-name -l log-file-name -t trace-file-name
```

8. Verify that your instance and databases show the new committed fix pack level by issuing the following command as an instance user:

```
db2pd -ruStatus
```

9. If you want to use capabilities specific to the fix pack, update the system catalog objects in your databases:

- a. Log on as the instance owner.

- b. For each database in the instance, issue the **db2updv105** command as follows:

```
db2updv105 -d db-name
```

Installing offline fix pack updates to a DB2 pureScale instance (simplified method)

Use the new parameters in the **installFixPack** command to update an existing DB2 pureScale to a new Version 10.5 fix pack level.

Before you begin

- Ensure that you meet all of the requirements before you install a fix pack.
- Ensure that you are logged in as root.
- The instance must be offline.

About this task

This task uses the new capabilities of the **installFixPack** command to update an existing DB2 pureScale instance to the latest fix pack in offline mode. These new capabilities simplify the process of updating the instances and automate the update of additional software components. To update the cluster for a DB2 pureScale instance, specify the instance name in the enhanced **installFixPack** command.

Procedure

To update an existing DB2 pureScale instance to a new fix pack level:

1. Install the new fix pack level on all hosts in the DB2 pureScale instance by issuing the **installFixPack** command from the fix pack installation media directory. The media directory must be accessible to the root and the instance user from all the members, secondary CF and primary CF. For example, to update to Fix Pack 1, issue the **installFixPack** command on each host as follows:

```
media-dir/installFixPack -p FP-install-path -I InstName -offline -l install-log-file -t trace-file-name
```

where *FP-install-path* is the directory where you want to install the fix pack and *media-dir* is the directory where you uncompressed the fix pack image. *FP-install-path* must be the same on all hosts.

2. Run the **db2instance -list** command to ensure that the cluster is in a consistent state. If there are any alerts of inconsistent state in the cluster, refresh the resource model as the instance owner:

```
db2cluster -cm -repair -resources
```

3. Determine whether the offline fix pack update was successful in all members and CFs. For example, to verify the update to Fix Pack 1, issue the **installFixPack** command as follows:

```
media-dir/installFixPack -check_commit -I instance-name -t trace-file-name  
-l check-commit-log-dir
```

If the command output shows any problems, fix them before you continue with the next step.

4. Commit the DB2 instance to the new level. For example, to commit the update to Fix Pack 1, issue the **installFixPack** command as follows:

```
media-dir/installFixPack -commit_level -I InstName -l commit-log-dir
```

where *media-dir* is the directory where you uncompressed the fix pack image.

5. Start the database manager in all members and CFs by issuing the **db2start instance** command in each host as follows:

```
su - InstName  
db2start instance on host-name
```

where *InstName* represents the instance owner name.

6. Start the database manager for the instance by issuing the **db2start** command as follows:

```
su - InstName
db2start
exit
```

where *InstName* represents the instance owner name.

7. To uninstall the previously installed DB2 copy, run the **db2_deinstall** command:

```
DB2DIR/install/db2_deinstall -a
```

where *DB2DIR* is the installation path of the previously installed DB2 copy.

Post-installation tasks for fix packs (Linux and UNIX)

As part of a fix pack installation, updating DB2 instances and binding of the database utilities (**IMPORT**, **EXPORT**, **REORG**, the Command Line Processor) and the CLI bind files are done automatically.

However, if an error occurs, you can manually update the DB2 instances and bind the database utilities and the CLI bind files. Depending on your database products and the fix pack installation method used, you might need to update the DB2 instances, restart the DB2 instances, restart the DB2 Administration Server and, if you have InfoSphere Federation Server installed, start the **djxlink** command.

Procedure

Perform the following actions:

1. If you have InfoSphere Federation Server installed, run the **djxlink** command.

Perform the following tasks after installing the fix pack and before running **db2iupdt**:

- a. Log on as root.
- b. Remove or rename the file `djxlink.out`, which is located in the *DB2DIR/lib* directory, where *DB2DIR* is the DB2 installation directory.
- c. Ensure that all of the appropriate variables are set, either in your current environment or in the `db2dj.ini` file. For example, if you are using a federated server to connect to an Oracle data source, set the environment variable **ORACLE_HOME** to the Oracle home directory.
- d. Run the command:

```
djxlink
```

2. Update instances to use the new DB2 database level.

All existing instances in the DB2 copy must be updated after a fix pack is installed. By default, the **installFixPack** command updates the DB2 instances automatically. However, if an error occurs, you can update instances manually.

Perform the following steps:

- a. Log on as root.
- b. Determine which instances are associated with the DB2 copy by issuing the command:

```
DB2DIR/instance/db2ilist
```

where *DB2DIR* represents the location where the DB2 copy is installed.

- c. If you made any changes to the `db2profile` or `db2cshrc` scripts, either back up the scripts or copy the changes into the `userprofile` and `usercshrc` scripts, respectively.

This action is required because the **db2iupdt** command overwrites the `db2profile` and `db2cshrc` scripts. It does not overwrite the `userprofile` and `usercshrc` scripts.

- d. For each instance, issue the command as follows. In a DB2 pureScale environment, skip this step.

```
DB2DIR/instance/db2iupdt iname
```

where *iname* represents the instance name and *DB2DIR* represents the location where the DB2 copy is installed.

- e. If the DB2 Administration Server (DAS) belongs to the DB2 copy where you installed the fix pack, issue the command:

```
DB2DIR/instance/dasupdt
```

where *DB2DIR* is the location where the DB2 copy is installed. If this DB2 copy is now running at a more recent fix pack level than all of the other DB2 copies, consider updating the DAS to belong to this DB2 copy.

3. Optional: Update the system catalog objects in your databases to support the fix pack. 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. For each instance in the DB2 copy where you applied the fix pack, perform the following actions:

- a. Log in as the instance owner.
- b. For each database, issue the command:

```
db2updv105 -d dbname
```

where *dbname* represents the name of the database.

Note: Backup your database before running `db2updv105`. Some system objects might become unusable after moving back to an earlier fix pack, and your database will need to be restored.

4. Restart the DB2 instances and the DB2 Administration Server (DAS).

This step is required if you installed a fix pack to update an existing installation. If you installed the fix pack to create a new installation, this step is not required.

To restart an instance:

- a. Log in as the instance owner.
- b. Issue the command **db2start**.

Repeat for each instance.

To restart the DB2 administration server, log in as the DAS owner and run the **db2admin start** command.

5. Optional: If you issued the **db2iauto** command to prevent instances from auto-starting before installing the fix pack, enable auto-start for the instances again. Issue the following command while logged on as root:

```
DB2DIR/instance/db2iauto -on iname
```

where *DB2DIR* is the location where the DB2 copy is installed and *iname* represents the instance owner name. The command must be performed once for each instance that you altered with the **db2iauto** command before you installed the fix pack.

6. Optional: Bind the bind files. Binding of the database utilities and the CLI bind files occurs automatically during a new DB2 installation or when applying a fix pack. However, if an error occurs, you can manually bind the database utilities and the CLI bind files. Refer to “Binding bind files after installing fix packs.”
7. Optional: Recompile applications.
To take advantage of any changes to the files linked to in the application, recompiling applications is recommended.

Results

After you have completed these tasks, the fix pack installation and configuration is complete.

Updating an instance to a higher level within a release using the **db2iupdt** command

The **db2iupdt** command can be used to update an instance to a higher level within a release.

Before you begin

Before running the **db2iupdt** command, you must first stop the instance and all processes that are running for the instance. When using this command, ensure you have reviewed the prerequisites and the pre-installation checklist so that your instance and hosts are compliant.

About this task

The **db2iupdt** command can be issued against instances of the same version that are associated with the same or a different DB2 copy. In all cases, it will update the instance so that it runs against the code located in the same DB2 copy as where you issued the **db2iupdt** command. Use this command to:

- Install a new DB2 database product or feature to the DB2 copy associated to the DB2 instance.
- Update a DB2 instance from one DB2 copy to another DB2 copy of the same version of DB2 database product.

You must run this command on each host in the DB2 pureScale instance.

Procedure

1. Log in as root.
2. Stop the instance and all process that are running for the instance.
3. Run the **db2iupdt** command as follows to update the db2inst1 instance:

```
/opt/IBM/db2/<install_path>/instance/db2iupdt -d db2inst1
```

Note: If this command is run from a DB2 pureScale Feature copy, the existing db2inst1 must have an instance type of dsf. You must run the **db2iupdt** command on all members and CFs in the cluster. Commit the DB2 DBMS instance to the new level by issuing the following command:

```
db2iupdt -commit_level instance-name
```

Binding bind files after installing fix packs

As part of a fix pack installation on the server, binding of the database utilities (**IMPORT**, **EXPORT**, **REORG**, the Command Line Processor) and the CLI bind files occurs automatically.

However, if you install a fix pack on the client or an error occurs, you can manually bind the database utilities and the CLI bind files. Different subsets of bind files must be bound for DB2 Database for Linux, UNIX, and Windows and host or System i® database servers.

Before you begin

Ensure that you have the necessary authority to perform the **BIND** command.

About this task

Note: To ensure that not all users have access to databases created with RESTRICTIVE mode, do not grant privileges to PUBLIC for a database created with the RESTRICTIVE mode.

Procedure

To bind the bind files:

1. If you installed the fix pack on DB2 database products that have existing databases, perform the following commands once for each database:

```
db2 terminate
db2 CONNECT TO dbname user USERID using PASSWORD
db2 BIND path\db2schema.bnd BLOCKING ALL GRANT PUBLIC SQLERROR CONTINUE
db2 BIND path\@db2ubind.lst BLOCKING ALL GRANT PUBLIC ACTION ADD
db2 BIND path\@db2cli.lst BLOCKING ALL GRANT PUBLIC ACTION ADD
db2 terminate
```

where *dbname* represents the name of a database to which the files should be bound, and where *path* is the full path name of the directory where the bind files are located, such as *INSTHOME*\sqllib\bnd where *INSTHOME* represents the home directory of the DB2 instance. db2ubind.lst and db2cli.lst contain lists of required bind files used by DB2 database products. Packages that are already bound will return an SQL0719N error. This is expected.

2. Optional: If you installed the fix pack on DB2 database products that have existing databases, rebind the packages by running the **REBIND** or **db2rbind** command.

After you install a fix pack, some packages are marked as invalid. Packages marked as invalid are implicitly rebound the first time an application uses them. To eliminate this overhead and to ensure that the rebind is successful, manually rebind all packages. For example, issue the **db2rbind** command:

```
db2rbind dbname -l logfile all
```

where *dbname* represents the name of a database whose packages are to be revalidated, and where *logfile* is the name of the file to be used for recording errors during the package revalidation procedure.

3. If you installed the fix pack on DB2 database products that have existing spatial-enabled databases, perform the following commands once for each database:

```
db2 terminate
db2 CONNECT TO dbname
db2 BIND path\BND\@db2gse.lst
db2 terminate
```

where *dbname* represents the name of a database to which the files should be bound, and where *path* is the full path name of the directory where the bind files are located, such as *INSTHOME*\sqllib\bnd where *INSTHOME* represents the

home directory of the DB2 instance. `db2gse.lst` contains the names of the bind files for the stored procedures that DB2 Spatial Extender provides.

4. If you connect to DB2 databases on host or System i servers, perform the following actions:

- For DB2 databases on z/OS or OS/390®:

```
db2 terminate
db2 CONNECT TO dbname user USERID using PASSWORD
db2 BIND path\@ddcsmv.s.lst BLOCKING ALL SQLERROR CONTINUE GRANT PUBLIC ACTION ADD
db2 terminate
```

- For DB2 databases on VM:

```
db2 terminate
db2 CONNECT TO dbname user USERID using PASSWORD
db2 BIND path\@ddcsvm.s.lst BLOCKING ALL SQLERROR CONTINUE GRANT PUBLIC ACTION ADD
db2 terminate
```

- For DB2 databases on VSE:

```
db2 terminate
db2 CONNECT TO dbname user USERID using PASSWORD
db2 BIND path\@ddcsvse.s.lst BLOCKING ALL SQLERROR CONTINUE GRANT PUBLIC ACTION ADD
db2 terminate
```

- For DB2 databases on System i:

```
db2 terminate
db2 CONNECT TO dbname user USERID using PASSWORD
db2 BIND path\@ddcs400.s.lst BLOCKING ALL SQLERROR CONTINUE GRANT PUBLIC ACTION ADD
db2 terminate
```

where *dbname* represents the name of a host or System i database to which the files should be bound, and where *path* is the full path name of the directory where the bind files are located, such as `INSTHOME\sqliib\bnd` where `INSTHOME` represents the home directory of the DB2 instance.

5. If you connect to databases that are running on different operating systems (Linux, UNIX or Windows) or at different DB2 versions or service levels, bind the database utilities and CLI bind files against those databases.

Note:

- The actions required are the same irrespective of whether you connect to a database on another DB2 database system or in another DB2 copy on the same machine.
- If you have installed the fix pack in multiple locations, perform the actions once from each unique combination of operating system and DB2 version or service level.

Perform the following actions:

```
db2 terminate
db2 CONNECT TO dbname user USERID using PASSWORD
db2 BIND path\@db2ubind.lst BLOCKING ALL GRANT PUBLIC ACTION ADD
db2 BIND path\@db2cli.lst BLOCKING ALL GRANT PUBLIC ACTION ADD
db2 terminate
```

where *dbname* represents the name of a database to which the files should be bound, and where *path* is the full path name of the directory where the bind files are located, such as `INSTHOME\sqliib\bnd` where `INSTHOME` represents the home directory of the instance where you are issuing the commands.

`db2ubind.lst` and `db2cli.lst` contain lists of required bind files used by DB2 database products. Packages that are already bound will return an SQL0719N error. This is expected.

Binding federated databases

If you have existing federated databases, you must bind the bind files `db2dsproc.bnd` and `db2stats.bnd` after you install a DB2 fix pack. To bind the bind files, you must have one of the following authorities:

- DBADM authority
- ALTERIN privilege on the schema
- BIND privilege on the package

To bind the bind files `db2dsproc.bnd` and `db2stats.bnd`, connect to the database and run the **BIND** command. For example:

```
db2 CONNECT TO dbname user USERID using PASSWORD
db2 bind path/db2dsproc.bnd blocking all grant public
db2 bind path/db2stats.bnd blocking all grant public
db2 terminate
```

where *dbname* represents the name of the federated database, and *path* represents the full path name of the directory where the bind files are located, such as `$HOME/sql1lib/bnd` where `$HOME` represents the DB2 instance home directory.

Chapter 20. Self-tuning memory for DB2 pureScale environments

In a DB2 pureScale environment, each member has its own self-tuning memory manager (STMM) tuner, which actively tunes the memory configurations of the particular member that is based on dynamic workload characteristics and local resources.

For a new database that is created in the V10.5 release, the default value of STMM tuning member in the SYSCAT table is -2. This setting ensures that each member has its own tuner, tuning independently to balance the following factors:

- Workload
- DB2 memory requirements
- System memory requirements

The memory requirements might be different on each member.

Ensuring that STMM tuning capabilities are present on each member is important in the following scenarios:

- A consolidated environment where multiple databases can have workload peak at different time of the day.
- The “member subsetting” capability is enabled so that a workload can be spread across selected members.

STMM decides on which member the STMM tuner is active based on the value in the SYSCAT table. The SYSCAT table is updated by using the **UPDATE STMM TUNING MEMBER** stored procedure, as shown:

```
CALL SYSPROC.ADMIN_CMD('update stmm tuning member member')
```

Here is a summary of the members on which the STMM tuner is active based on the value in the SYSCAT table.

Table 20. Determining the members on which STMM tuner is active

Value in SYSCAT table	Member where STMM Tuner is running
-2	All members.
-1	One member, which is chosen by STMM.
Any number that matches a member number	Member with number that matches the value in SYSCAT
Any number that does not match a member number	Defaults to -1, where the tuner runs on one member, which is chosen by STMM

Note that when the tuning member changes, some data collected from the member which was running the tuner, is discarded. This data must be recollected on the new tuning member. During this short period of time when the data is being recollected, the memory tuner will still tune the system; however, the tuning can occur slightly differently than it did on the original member.

Starting the memory tuner in a DB2 pureScale environment

In a DB2 pureScale environment, the memory tuner will run whenever the database is active on one or more members that have **self_tuning_mem** set to ON.

Disabling self-tuning memory for a specific member

- To disable self-tuning memory for a subset of database members, set the **self_tuning_mem** database configuration parameter to OFF for those members.
- To disable self-tuning memory for a subset of the memory consumers that are controlled by configuration parameters on a specific member, set the value of the relevant configuration parameter to a fixed value on that member. It is recommended that self-tuning memory configuration parameter values be consistent across all running members.
- To disable self-tuning memory for a particular buffer pool on a specific member, issue the ALTER BUFFERPOOL statement, specifying a size value and the member on which self-tuning memory is to be disabled.

An ALTER BUFFERPOOL statement that specifies the size of a buffer pool on a particular member will create an exception entry (or update an existing entry) for that buffer pool in the SYSCAT.BUFFERPOOLEXCEPTIONS catalog view. If an exception entry for a buffer pool exists, that buffer pool will not participate in self-tuning operations when the default buffer pool size is set to AUTOMATIC. To remove an exception entry so that a buffer pool can be used for self tuning:

1. Disable self tuning for this buffer pool by issuing an ALTER BUFFERPOOL statement, setting the buffer pool size to a specific value.
2. Issue another ALTER BUFFERPOOL statement to set the size of the buffer pool on this member to the default.
3. Enable self tuning for this buffer pool by issuing another ALTER BUFFERPOOL statement, setting the buffer pool size to AUTOMATIC.

Chapter 21. Explicit hierarchical locking for DB2 pureScale environments

Explicit hierarchical locking (EHL) for IBM DB2 pureScale Feature takes advantage of the implicit internal locking hierarchy that exists between table locks, row locks, and page locks. EHL functionality helps avoid most communication and data sharing memory usage for tables.

Table locks supersede row locks or page locks in the locking hierarchy. When a table lock is held in super exclusive mode, EHL enhances performance for DB2 pureScale instances by not propagating row locks, page locks, or page writes to the caching facility (CF).

EHL is not enabled by default in DB2 pureScale environments. However, it can be enabled or disabled by using the **opt_direct_wrkld** database configuration parameter. When turned on, tables which are detected to be accessed primarily by a single member are optimized to avoid CF communication. The table lock is held in super exclusive mode on such a member while this state is in effect. Attempts to access the table by a remote member automatically terminates this mode.

There are two EHL states:

Table 21. EHL table states

Table state	Description
NOT_SHARED / DIRECTED_ACCESS	Refers to a table that is in explicit hierarchical locking state. Row locks, page locks, and page writes are managed only on the local member.
SHARED/FULLY SHARED	Refers to a table that is not in explicit hierarchical locking state. Row locks, page locks, and page writes are coordinated by using the CF.

Regular tables, range partitioned tables, or partitioned indexes might exist in one of the prior states, or in a transitional state between the SHARED and NOT_SHARED states:

EHL is useful for the following types of workloads, as they are able to take advantage of this optimization:

- Grid deployments, where each application has affinities to a single member and where most of its data access is only for these particular applications. In a database grid environment a DB2 pureScale cluster has multiple databases, but any single database is accessed only by a single member. In this case, all the tables in each database move into the NOT_SHARED state.
- Partitioned or partitionable workloads where work is directed such that certain tables are accessed only by a single member. These workloads include directed access workloads where applications from different members do not access the same table.
- One member configurations or batch window workloads that use only a single member. A system is set up to have nightly batch processing with almost no OLTP activity. Because the batch workload is often run by a single application, it is the only one accessing tables and they can move into the NOT_SHARED state.

An application can be partitioned so that only certain tables are accessed by the connections to a single member. Using this partitioning approach, these tables move into the NOT_SHARED state when the **opt_direct_wrkld** configuration parameter is enabled.

EHL for directed workloads must avoid workload balancing (WLB). Instead, use client affinity and the member subsetting capability for directed workloads that do not use WLB.

Use cases for Explicit Hierarchical Locking (EHL)

Explicit hierarchical locking (EHL) for the IBM DB2 pureScale Feature is designed to improve performance by avoiding CF communications for tables that are accessed from only one member. When EHL is enabled, if only one member accesses a data table, partitioned table, or partitioned index then the table transition to the NOT_SHARED state.

Transitioning to this state is an ideal scenario for data tables, partitioned tables, or partitioned indexes. Grid deployments, partitioned or partitionable workloads with directed access, or batch window workloads that use only a single member are access patterns that can benefit from using EHL. The DB2 database server automatically detects these access patterns and transition applicable tables into the NOT_SHARED state.

Example 1: When the following sample SQL statements are issued, the DB2 database server detects that tables tab1, tab2, tab3, and tab4 are accessed by only one member and would benefit from EHL. These tables transition into the NOT_SHARED state.

```
Member 1:
db2 -v "select * from tab1"
db2 -v "delete from tab3 where col1 > 100"
```

```
Member 2:
db2 -v "insert into tab2 values (20,20)"
db2 -v "select * from tab4"
```

To ensure that tables remain in the NOT_SHARED state, tune your applications or use EHL for workloads where only a single member accesses a data table, partitioned table, or partitioned index.

Example 2: In the following example, the DB2 database server detects that the EHL optimization does not apply. Multiple members are all attempting to access the same table tab1. The table does not transition to the NOT_SHARED state.

```
Member 1:
db2 -v "select * from tab1"
```

```
Member 2:
db2 -v "insert into tab1 values (20,20)"
```

```
Member 1:
db2 -v "delete from tab1 where col1 > 100"
```

```
Member 2:
db2 -v "select * from tab1"
```

Use MON_GET_TABLE to monitor whether or not tables transition the NOT_SHARED state.

Explicit hierarchical locking state changes and performance implications

In a DB2 pureScale environment, regular tables, range partitioned tables, or partitioned indexes exist in the SHARED state, the NOT_SHARED state, or in transition between these two states.

Objects entering or already in NOT_SHARED state

A table might enter the NOT_SHARED state as a result of the following activity:

- If an INSERT, UPDATE or DELETE operation, or a scan is performed on a table.
- If a table is range partitioned, a data partition that is accessed by using the preceding actions might enter NOT_SHARED state, while the logical table is unaffected.
- The **CREATE INDEX** operation in non-partitioned indexes triggers a NOT_SHARED state transition of the logical table and the index anchors. If these indexes might enter the NOT_SHARED state independent of the data partitions.

As tables in a NOT_SHARED state are running in a mode similar to DB2 Enterprise Server Edition, they also have properties similar to DB2 Enterprise Server Edition tables. When a regular table is in the NOT_SHARED state, all of its table objects, such as data, index, LONG, LOB, XDA, are also in the NOT_SHARED state. However, only the table and row locks are behaving as in the NOT_SHARED state.

A range partitioned table can have its partitions enter or leave the NOT_SHARED state independently of each other, which is beneficial because not all transactions would access all the partitions on a table. So, partition independence allows members to access different partitions without conflict on the DATA_SHARING lock. Furthermore, all table objects (such as partitioned indexes) within a partition also inherit properties as in the regular table case.

Note: Nonpartitioned indexes on a partitioned table all must enter or leave the NOT_SHARED state simultaneously. This simultaneous action is because all index anchor objects are under the protection of a single logical table lock. In other words, when a nonpartitioned index is being accessed its logical table lock enters the NOT_SHARED state, which causes all nonpartitioned indexes to enter the NOT_SHARED state as well. This behavior has no effect on the partitions since they have their own partition locks for protection.

Objects exiting the NOT_SHARED state

A table exits NOT_SHARED state as a result of the following activity:

- Any access to the table from another member
- Drop table or database deactivation
- ATTACH or DETACH partition for a partitioned table

Any access to a table from another member results in an exit from NOT_SHARED state on the first member to allow the DB2 database server to grant a lock on the table to the other member. When a table is moving out of NOT_SHARED to the SHARED state, its table lock is held in Z mode (super exclusive mode) until all page locks and row locks are registered on the global lock manager (GLM) and all dirty buffer pool pages are written to the group buffer pool (GBP). This can be a lengthy process.

The DB2 database server can detect when a transition to NOT_SHARED state is not optimal, and it avoids the state change.

For member crash recovery, the entire table is unavailable until recovery is completed. When EHL is enabled, member crash recovery takes a length of time similar to crash recovery for an DB2 Enterprise Server Edition database. Time is required because changed data pages are not cached in the caching facility (CF), only in the local buffer pool. So modified pages must be rebuilt by replaying log records. Use the **page_age_trgt_mcr** database configuration parameter to control the length of time the pages remain buffered in the local buffer pool.

Exiting EHL is not immediate and involves CF communication for locks and pages for the table. Memory and time that is required for this operation is proportional to the number of locks and pages that are held for the object in the buffer pool. A small table typically takes only a few seconds to exit EHL. However, a very large table might take several minutes to exit EHL.

In extremely rare circumstances, it is possible for the GLM to become full during EHL exit and you are unable to send all required locks to the CF. This will prevent EHL exit from completing. If this condition occurs, other members will not be able to access this table until EHL exit is able to complete. This may result in lock time out events on other members accessing the table. When this condition is detected, and if the **CF_GBP_SZ** and **CF_LOCK_SZ** database configuration parameters are both configured to AUTOMATIC, the DB2 database server will attempt to trade memory from the group buffer pool (GBP) to the GLM, to allow lock registration to complete. The size of the GBP will be slightly reduced and the size of the GLM will be increased by this amount. There is a limit on the amount of GBP memory that can be traded in this way. However, if the **CF_GBP_SZ** and **CF_LOCK_SZ** database configuration parameters are not configured to AUTOMATIC, or if these actions do not free up enough GLM memory to allow EHL exit to complete within 40 seconds of detecting this condition, then all applications holding the lock that cannot be sent to the GLM will be forced. Forcing the applications will allow the lock to be released so that it does not need to be sent to the CF and this allows EHL exit to continue. ADM1504 will be logged when this memory trading occurs and ADM1503 will be logged for each application that is forced.

Monitoring EHL

EHL can be monitored using the following monitors and APIs:

- The LOCK WAIT event monitor
- MON_GET_DATABASE() administrative API
- MON_GET_TABLE() administrative API
- MON_GET_LOCKS() administrative API
- MON_GET_APPL_LOCKWAIT() administrative API

Chapter 22. DB2 Advanced Copy Services (ACS) scripted interface

If you want to perform snapshot operations with a storage device that does not provide a vendor library to implement the DB2 ACS API, you have to create your own script.

A script allows the DB2 ACS API to directly communicate with the storage system and create a snapshot of volumes which contain data and log files for a database. Afterward, you can use a different script to perform the complementary action of restoring the snapshot image, or even deleting the image.

By creating your own script for performing snapshots, you can use unsupported storage boxes, or boxes that are available before a vendor library is available for interfacing with DB2 ACS. A vendor library provides the necessary extensions for implementing snapshot-based backup and restore. A script serves a similar role. With the improved interfacing with scripts introduced in V10.5, DB2 removes the need for the script to account for some of the more error prone actions, like suspending and resuming operations when taking a snapshot backup. Like snapshot operations with supported storage hardware, snapshot operations that use scripts generate a history file entry, meaning that you can monitor the success or failure of snapshot backups.

The DB2 ACS API is wrapped in the library for DB2 ACS, which is included with the DB2 product. The library for ACS writes the protocol files to the protocol file repository and invokes the script that you specify for your snapshot operation.

DB2 Advanced Copy Services (ACS) protocol file

The DB2 Advanced Copy Services (ACS) protocol files are created by the library for DB2 ACS and contain information that is needed by scripts for snapshot operations.

The protocol files are located in the protocol file repository. You should create a directory for this repository before performing your snapshot operation. You specify the repository by using the **options** parameter with the relevant command. If you do not create a directory, the protocol file repository will be the same directory that the script is located.

The protocol files serve two purposes:

- They show the progress of the running operation. In the case of failed operations, they also contain some information that you can use for debugging.
- They contain information and options provided from the library for DB2 ACS to the script. Some of the information, such as the metadata string, is also needed by the library for DB2 ACS or DB2 to restore the snapshot.

A protocol file is divided into different sections, each of which shows the progress and options of each function call. The output in each section contains the following information:

- Function name. For example db2ACSInitialize
- Beginning and ending timestamp for the function call

- Commands that were used to invoke the script, in the following format

```
cmd: path_to_script -a action
      -c protocol_file_repository/protocol_file_name.cfg
```
- Options that were given in the function calls. See Table 22 for a list and description of the options.

Table 22. Options written by the library for DB2 ACS

Key name	Description
ACTION	Action that is being performed: <ul style="list-style-type: none"> • DB2ACS_ACTION_READ_BY_GROUP during restore of parts of the database, in particular restore excluding log files • DB2ACS_ACTION_READ_BY_OBJECT during restore of the whole database, DB2ACS_ACTION_WRITE during snapshot
APP_OPTIONS	Hex value that combines the DB2BACKUP_*
DATAPATH_AUTOSTORAGE	Key for each storage path that is configured in the database
DATAPATH_DB	Database paths configured in the database
DATAPATH_GENERIC	<ul style="list-style-type: none"> • Additional paths configured in the database • Placeholder for future types
DATAPATH_LOCAL_DB	Local database paths configured in the database
DATAPATH_TBSP_CONTAINER	Key for each DMS container that is configured in the database
DATAPATH_TBSP_DEVICE	Key for each raw device that is configured in the database
DATAPATH_TBSP_DIR	Key for each SMS storage path that is configured in the database
DB_NAME	Name of the database for that the operation is done
DBPARTNUM	Database partition number to be operated on
DB2BACKUP_MODE	Whether the backup is offline or online
DB2BACKUP_LOGS	Whether log files are included in or excluded from the backup. If logs are excluded, LOG_DIR and MIRRORLOG_DIR are not contained in the protocol file.
DELETE_OBJ_ID	Object ID of the object to be deleted
EXTERNAL_OPTIONS	Lists any options that you specify in the backup and restore command and are automatically copied into to the custom script.
EXTERNAL_SCRIPT	Name of the script for the snapshot operation
FIRST_ACTIVE_LOG_CHAIN	The log chain of the first active log

Table 22. Options written by the library for DB2 ACS (continued)

Key name	Description
FIRST_ACTIVE_LOG_ID	Number of first active log of the database during the time the snapshot was taken
INSTANCE	Name of the instance for the database
LOGPATH_MIRROR	Mirror log directory
LOGPATH_PRIMARY	Log directory of the database
METADATA	String that represents the Base64 encoded meta data memory block
METADATA_CHECKSUM	Checksum of the Base64 metadata string
METADATA_SIZE	Size of the encoded metadata string
METADATA_DECODED_SIZE	Size of the decoded metadata block
OBJ_DB2ID_LEVEL	Fix pack level that was used during the backup
OBJ_DB2ID_RELEASE	Release level of the DB2 product that was used during the backup
OBJ_DB2ID_VERSION	Version of the DB2 product that was used during backup
OBJ_HOST	Host server where the database partition resides
OBJ_ID	Unique identifier for each stored object
OBJ_OWNER	Owner of the object
OBJ_TYPE	Snapshot
OPERATION	Operation identifier: <ul style="list-style-type: none"> • Delete • Restore • Snapshot
QUERY_DB	Name of the database that is queried for
QUERY_HOST	Name in the host in the object
QUERY_INSTANCE	Name of the instance of the database that is contained in the backup image
QUERY_OWNER	Owner of the object
QUERY_DBPARTNUM	<ul style="list-style-type: none"> • Number of the database partition backed up to the object • -1 for the generic case
QUERY_TIMESTAMP	Timestamp queried for
QUERY_TYPE	Type of the object to be queried, snapshot or hex code representing the type
RC_DELETE	Return code of the deletion operation. A non-zero value indicates that an error happened in the section.
RC_OPERATION	Return code of the complete backup operation. A non-zero value indicates that an error happened in the section.
RC_PREPARE	Return code of the prepare action. A non-zero value indicates that an error happened in the section.

Table 22. Options written by the library for DB2 ACS (continued)

Key name	Description
RC_RESTORE	Return code of the complete restore operation. A non-zero value indicates that an error happened in the section.
RC_SNAPSHOT	Return code of the snapshot action. A non-zero value indicates that an error happened in the section.
RC_STORE_METADATA	Return code of store_metadata operation. A non-zero value indicates that an error happened in the section.
RC_VERIFY	Return code of the verify action. A non-zero value indicates that an error happened in the section.
RESULT_n_FILE	The name of the <i>n</i> th file during query, delete and restore
SIGNATURE	Software level of the DB2 version being used

Example protocol file for a snapshot backup

This section contains an example protocol file written for a snapshot backup operation invoking the sample script. For illustrative purposes, it has been broken up into sections for each DB2 ACS API function that is a part of the operation.

db2ACSInitialize

After loading the library for DB2 ACS and querying the version of the DB2 ACS API, the database manager establishes a DB2 ACS session by calling db2ACSInitialize(). This step is required for all operations.

The flags that are of most interest for monitoring purposes are:

- EXTERNAL_SCRIPT: the name and path of the script
- DB_NAME: the database name
- INSTANCE: the DB2 instance name
- DBPARTNUM: the database partition number

```
# =====
# db2ACSInitialize(): BEGIN [2012-11-30 08:15:45]
EXTERNAL_SCRIPT=/home/hotellnx99/jklauke/libacssc.sh
HANDLE=1354281345
START_TIME=1354281345
DB_NAME=SAMPLE
INSTANCE=jklauke
DBPARTNUM=0
SIGNATURE=SQL10020
EXTERNAL_OPTIONS=/home/hotellnx99/jklauke/repository 2ndoption
# db2ACSInitialize(): END
# =====
```

db2ACSBeginOperation

The database manager calls db2ACSBeginOperation() to begin the specified operation (indicated in the OPERATION flag).

```
# =====
# db2ACSBeginOperation(): BEGIN [2012-11-30 08:15:45]
OPERATION=snapshot
# db2ACSBeginOperation(): END
# =====
```

db2ACSPartition

The database manager calls db2ACSPartition(), which associates a group identifier with each of the paths listed by the database manager as belonging to a database partition. The library for DB2 ACS groups database path information for a single database partition together, so the partition ID is unique for every path. This makes it possible to take a snapshot at the file-set, file-system, and volume-group level. The path-related flags that are of interest in this section are:

- LOG_DIR, MIRRORLOG_DIR: the log paths
- DB_PATH, LOCAL_DB_PATH: the database paths
- STORAGE_PATH, CONT_PATH, TBSP_DIR

The SYSIBMADM.DBPATHS administrative view provides these same path types.

A number of flags provide information about the settings for the current operation:

- DB2BACKUP_MODE: Offline or online backup
- DB2BACKUP_LOGS: Exclude or include logs. In this example, the logs are included, so the sample customer script compresses the log files but in a different file than the other database files.

```
# =====
# db2ACSPartition(): BEGIN [2012-11-30 08:15:06]
OBJ_HOST=hotel1nx99
OBJ_OWNER=
OBJ_TYPE=snapshot
OBJ_DB2ID_LEVEL=0
OBJ_DB2ID_RELEASE=2
OBJ_DB2ID_VERSION=10
APP_OPTIONS=0x1000
DB2BACKUP_MODE=OFFLINE
DB2BACKUP_LOGS=INCLUDE
LOGPATH_PRIMARY=/home/hotel1nx99/jklauke/jklauke/NODE0000/SQL00001/LOGSTREAM0000/
DATAPATH_DB=/home/hotel1nx99/jklauke/jklauke/NODE0000/SQL00001/MEMBER0000/
DATAPATH_LOCAL_DB=/home/hotel1nx99/jklauke/jklauke/NODE0000/sqlbdir/
DATAPATH_DB=/home/hotel1nx99/jklauke/jklauke/NODE0000/SQL00001/
DATAPATH_AUTOSTORAGE=/home/hotel1nx99/jklauke/jklauke/NODE0000/SAMPLE/
# db2ACSPartition(): END
# =====
```

db2ACSPrepare

The database manager calls db2ACSPrepare() to prepare to perform the snapshot. In the protocol file, the prepare section shows the command with which the script was invoked and the return code of the preparation.

```
# =====
# db2ACSPrepare(): BEGIN [2012-11-30 08:15:45]
# cmd: /home/hotel1nx99/jklauke/libacssc.sh -a prepare
#      -c /home/hotel1nx99/jklauke/repository/db2acs.SAMPLE.0.jklauke.1353420068.cfg
#      /home/hotel1nx99/jklauke/repository 2ndoption
RC_PREPARE=0
# db2ACSPrepare(): END
# =====
```

If this step completes successfully, the database manager puts the database in SET WRITE SUSPEND state (assuming the snapshot backup is online).

db2ACSSnapshot

The database manager calls db2ACSSnapshot() to perform the snapshot. The protocol file shows the command used during the snapshot and the return code of the snapshot operation. This part of the protocol file shows

the point at which the real snapshot is taken and the vendor tools are triggered that run the operations on the storage boxes.

Note that the content between ACTION=DB2ACS_ACTION_WRITE and RC_SNAPSHOT is specific to the sample script, which compresses all of the paths shown in the db2ACSPartition section of the protocol file into one tar file and all log files (primary and mirror log files) to a second tar file.

```
# =====
# db2ACSSnapshot(): BEGIN [2013-01-15 10:18:23]
OBJ_ID=0
ACTION=DB2ACS_ACTION_WRITE
# cmd: /home/hotellnx99/jklauke/sql/lib/samples/BARVendor/libacssc.sh -a snapshot
# -c /home/hotellnx99/jklauke/repository/db2acs.SAMPLE.0.jklauke.
1358263103.cfg
BACKUP_FILE=/home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.
0.20130115101824.001.tar
# cmd: awk -F= '/^DATAPATH/
# { print $2; }' /home/hotellnx99/jklauke/repository/db2acs.SAMPLE.
0.jklauke.1358263103.cfg
# | tar -Pcf /home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.
0.20130115101824.001.tar
# -T - 2>/dev/null && echo 0 || echo 1
# backup tar created, rc=0
# Logs to be included
BACKUP_LOGS=/home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.
0.20130115101824.log.tar
# cmd: awk -F= '/^LOGPATH/ { print $2; }'
# /home/hotellnx99/jklauke/repository/db2acs.SAMPLE.0.jklauke.1358263103.cfg
# | tar -Pcf /home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.
0.20130115101824.log.tar
# -T - 2>/dev/null && echo 0 || echo 1
# tar for logs created, rc=0
RC_SNAPSHOT=0
# db2ACSSnapshot(): END [2013-01-15 10:18:24]
# =====
```

After this step completes, the database manager puts the database in WRITE RESUME state.

db2ACSVerify

The database manager calls db2ACSVerify() to verify that the snapshot backup succeeded. If your script contains a verify action, the library for DB2 ACS invokes your script. In the example script, the verify step only checks for the existence of the two tar files (if EXCLUDE LOGS were specified it would not checks for the existence of the tar file for the logs).

```
# =====
# db2ACSVerify(): BEGIN [2012-11-30 08:15:08]
FIRST_ACTIVE_LOG_ID=2
FIRST_ACTIVE_LOG_CHAIN=3
# cmd: /home/hotellnx99/jklauke/libacssc.sh -a verify
# -c /home/hotellnx99/jklauke/repository/db2acs_SAMPLE_1354281306_0.cfg
# /home/hotellnx99/jklauke/repository 2ndoption
# Backup '/home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.
0.1354281306.001.tar' checked: looks okay
# Logs '/home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.
0.1354281306.log.tar' checked: looks okay
RC_VERIFY=0
# db2ACSVerify(): END
# =====
```

If the script returns a non-zero return code, the following db2ACSStoreMetaData() call is skipped and db2ACSEndOperation is called instead. In the case of the example script, the library for DB2 ACS invokes the script with the rollback action. For an example of this, see this section.

db2ACSSStoreMetaData

The database manager calls db2ACSSStoreMetaData() to store metadata about the recovery object created by the operation. If your script contains a store_metadata action, the library for DB2 ACS invokes your script to perform actions such as:

- backing up the protocol file (it has to exist for a snapshot to be restored, queried, or deleted)
- renaming the backup

```
# =====
# db2ACSSStoreMetaData(): BEGIN [2013-01-15 10:18:24]
START_TIME=1358263104
METADATA_SIZE=12024
METADATA=U1FMV...
METADATA_CHECKSUM=16941
# cmd: /home/hotellnx99/jklauke/sql1lib/samples/BARVendor/libacssc.sh
-a store_metadata
-c /home/hotellnx99/jklauke/repository/db2acs.SAMPLE.0.jklauke.1358263103.cfg
RC_STORE_METADATA=0
# db2ACSSStoreMetaData(): END [2013-01-15 10:18:24]
# =====
```

db2ACSEndOperation

The database manager calls db2ACSEndOperation() to end the operation.

Successful operations

The return code of 0 indicates that the snapshot operation was successful.

```
# =====
# db2ACSEndOperation(): BEGIN [2012-11-30 08:15:08]
RC_OPERATION=0
# db2ACSEndOperation(): END
```

Failed operations

If the snapshot operation failed--that is, a call to the customer script had a non-zero return code or there was an internal error in the library for DB2 ACS--the db2ACSEndOperation section of the protocol file has a non-zero return code. If you specify a rollback action in your script, the script is called at this point. In the case of the sample script, the protocol file contains the following output:

```
# =====
# db2ACSEndOperation(): BEGIN [2013-01-18 05:26:06]
RC_OPERATION=1
# cmd:/home/hotellnx99/jklauke/sql1lib/samples/BARVendor/libacssc.sh -a rollback
-c /home/hotellnx99/jklauke/repository/db2acs.SAMPLE.0.jklauke.1358504766.cfg
# Delete old backup file :
/home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.0.20130118052606.001.tar
# Delete old backup file :
/home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.0.20130118052606.log.tar
RC_ROLLBACK=0
# db2ACSEndOperation(): END [2013-01-18 05:26:06]
# =====
```

db2ACSTerminate

The database manager calls db2ACSTerminate() to terminate the session.

```
# =====
# db2ACSTerminate(): BEGIN [2012-11-30 08:15:08]
# db2ACSTerminate(): END
# =====
```

Example protocol file for a snapshot restore

This section contains an example protocol file written for a snapshot restore operation invoking the sample script. A snapshot restore reads the protocol file for the snapshot backup, while at the same time writing new protocol files for the restore operation. If the restore is successful, those protocol files are deleted. For illustrative purposes, the following protocol file for a snapshot restore has been broken up into sections for each DB2 ACS API function that is a part of the operation.

db2ACSInitialize

After loading the library for DB2 ACS and querying the version of the DB2 ACS API, the database manager establishes a DB2 ACS session by calling db2ACSInitialize(). This step is required for all operations.

The flags that are of most interest for monitoring purposes are:

- EXTERNAL_SCRIPT: the name and path of the script
- DB_NAME: the database name
- INSTANCE: the DB2 instance name
- DBPARTNUM: the database partition number

```
# =====
# db2ACSInitialize(): BEGIN [2012-11-30 08:27:38]
REPOSITORY_PATH=/home/hotellnx99/jklauke/repository/
EXTERNAL_SCRIPT=/home/hotellnx99/jklauke/libacssc.sh
HANDLE=1354282058
START_TIME=1354282058
DB_NAME=SAMPLE
INSTANCE=jklaue
DBPARTNUM=0
SIGNATURE=SQL10020
EXTERNAL_OPTIONS=/home/hotellnx99/jklauke/repository
# db2ACSInitialize(): END
# =====
```

db2ACSBeginOperation

The database manager calls db2ACSBeginOperation() to begin the specified operation (indicated in the OPERATION flag).

```
# db2ACSBeginOperation(): BEGIN [2012-11-30 08:27:38]
OPERATION=restore
# db2ACSBeginOperation(): END
# =====
```

db2ACSBeginQuery

The database manager calls db2ACSBeginQuery() to determine which snapshot backup objects are available to be used for the restore operation and to prepare the restore. The protocol file also shows the command with which the script was invoked for the prepare action and the return code of the preparation.

```
# =====
# db2ACSBeginQuery(): BEGIN [2012-11-30 08:27:38]
QUERY_TYPE=snapshot
QUERY_PARTNUM=0
QUERY_DB=SAMPLE
QUERY_INSTANCE=*
QUERY_HOST=*
QUERY_OWNER=*
QUERY_TIMESTAMP=20121130082717
# cmd: /home/hotellnx99/jklauke/libacssc.sh -a prepare
#      -c/home/hotellnx99/jklauke/db2acs.SAMPLE.0.jklauke.1353421208.cfg
```

```

/home/hotel1nx99/jklauke/repository
RC_PREPARE=0
# db2ACSBEGINQuery(): END
# =====

```

db2ACSGetNextObject

The database manager calls db2ACSGetNextObject() to find an appropriate backup image for the given timestamp. The function is called iteratively and loops over the available files, giving information about each backup image to the database manager. The following output shows the looping over three protocol files:

```

# =====
# db2ACSGetNextObject(): BEGIN [2012-12-13 08:01:39]
RESULT_0_FILE=/home/hotel1nx99/jklauke/repository/db2acs.SAMPLE.
0.jklauke.1355341475.cfg
# read result object with timestamp 20121212144436
# db2ACSGetNextObject(): END [2012-12-13 08:01:39]
# =====
# db2ACSGetNextObject(): BEGIN [2012-12-13 08:01:39]
RESULT_1_FILE=/home/hotel1nx99/jklauke/repository/db2acs.SAMPLE.
0.jklauke.1355341690.cfg
# read result object with timestamp 20121212144811
# db2ACSGetNextObject(): END [2012-12-13 08:01:39]
# =====
# db2ACSGetNextObject(): BEGIN [2012-12-13 08:01:39]
RESULT_2_FILE=/home/hotel1nx99/jklauke/repository/db2acs.SAMPLE.
0.jklauke.1355341892.cfg
# read result object with timestamp 20121212145133
# db2ACSGetNextObject(): END [2012-12-13 08:01:39]
# =====

```

db2ACSRetrieveMetaData

The database manager calls db2ACSRetrieveMetaData() to retrieve all metadata about the backup image.

```

# =====
# db2ACSRetrieveMetaData(): BEGIN [2012-11-30 08:27:39]
GET_META_OBJ_ID=3
METADATA_DECODED_SIZE=9004
METADATA_CHECKSUM=14583
# db2ACSRetrieveMetaData(): END
# =====

```

db2ACSSnapshot

The database manager calls db2ACSSnapshot() to perform the restore. The protocol file shows the commands and actions used by the script and the return code of the snapshot operation. The action can be one of two options:

- DB2ACS_ACTION_READ_BY_OBJECT. This indicates that the LOGTARGET INCLUDE FORCE options were specified with the **RESTORE DATABASE** command. The script uncompresses both tar files (one for the data and one for the logs). You also need to copy the disks used for the log files.
- DB2ACS_ACTION_READ_BY_GROUP. This indicates that the LOGTARGET EXCLUDE FORCE options were specified with the **RESTORE DATABASE** command. This shows the groups, or IDs of the file systems, for the groups that have to be restored. You must not copy the disks used for the log files.

```

# =====
# db2ACSSnapshot(): BEGIN [2012-11-30 08:27:40]
OBJ_ID=3
ACTION=DB2ACS_ACTION_READ_BY_OBJECT
# cmd:/home/hotel1nx99/jklauke/libacssc.sh -a restore
-c /home/hotel1nx99/jklauke/repository/db2acs_SAMPLE_1354282058_0.cfg
/home/hotel1nx99/jklauke/repository
# cmd: tar -xf /home/hotel1nx99/jklauke/repository/SAMPLE.0.jklauke.

```



```

0.20121130082717.001.tar
  && echo 0 || echo 1
# tar extracted, rc=0
# cmd: tar -xf /home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.
0.20121130082717.log.tar
  && echo 0 || echo 1
# logs extracted, rc=0
RC_RESTORE=0
# db2ACSSnapshot(): END
# =====

```

If an appropriate backup image is found, the meta data is read from the protocol file and the restore is started by invoking the customer library.

db2ACSEndOperation

The database manager calls db2ACSEndOperation() to end the operation. The return code of 0 indicates that the restore operation was successful.

```

# =====
# db2ACSEndOperation(): BEGIN [2012-11-30 08:27:41]
END_ACTION=0
# db2ACSEndOperation(): END
# =====

```

db2ACSTerminate

The database manager calls db2ACSTerminate() to terminate the session.

```

# =====
# db2ACSTerminate(): BEGIN [2012-11-30 08:27:41]
# db2ACSTerminate(): END
# =====

```

Example protocol file for a snapshot deletion

This section contains an example protocol file written for a deletion of a snapshot image which invokes the sample script. During a deletion, the protocol file for the snapshot backup is read, while at the same time new protocol files are written for the delete operation. If the deletion is successful, those protocol files are removed. For illustrative purposes, the following protocol file for the deletion of the snapshot has been broken up into sections for each DB2 ACS API function that is a part of the operation.

db2ACSInitialize

After loading the library for DB2 ACS and querying the version of the DB2 ACS API, the database manager establishes a DB2 ACS session by calling db2ACSInitialize(). This step is required for all operations. Take care that you do not accidentally delete any images by either specifying the database name or using a unique protocol file repository for each snapshot operation. In the following output, for example, all backups contained in the /home/hotellnx99/jklauke/ directory are deleted.

```

# db2ACSInitialize(): BEGIN [2012-11-20 09:10:17]
REPOSITORY_PATH=/home/hotellnx99/jklauke/
EXTERNAL_SCRIPT=/home/hotellnx99/jklauke/libacssc.sh
HANDLE=1353420617
DB_NAME=*
INSTANCE=*
DBPARTNUM=0
SIGNATURE=SQL10020
EXTERNAL_OPTIONS=/home/hotellnx99/jklauke/
# db2ACSInitialize(): END
# =====

```



```
# db2ACSBeginOperation(): BEGIN [2012-11-20 09:10:17]
OPERATION=delete
# db2ACSBeginOperation(): END
# =====
```

db2ACSBeginOperation

The database manager calls db2ACSBeginOperation() to begin the specified operation (indicated in the OPERATION flag).

```
# db2ACSBeginOperation(): BEGIN [2012-11-30 08:27:38]
OPERATION=restore
# db2ACSBeginOperation(): END
# =====
```

db2ACSBeginQuery

The database manager calls db2ACSBeginQuery() to determine which snapshot backup objects are available to be deleted and to prepare the restore. The protocol file also shows the command with which the script was invoked for the prepare action and the return code of the preparation.

```
# =====
# db2ACSBeginQuery(): BEGIN [2012-12-13 08:24:42]
QUERY_TYPE=0x0
QUERY_DBPARTNUM=-1
QUERY_DB=*
QUERY_INSTANCE=*
QUERY_HOST=*
QUERY_OWNER=*
# cmd: /home/hotellnx99/jklauke/sqllib/samples/BARVendor/libacssc.sh -a prepare
# -c /home/hotellnx99/jklauke/repository/db2acs.0.1355405082.cfg
RC_PREPARE=0
# db2ACSBeginQuery(): END [2012-12-13 08:24:42]
# =====
```

db2ACSGetNextObject

The database manager calls db2ACSGetNextObject() to find an appropriate backup image for the given timestamp. The function is called iteratively and loops over the available files, giving information about each backup image to the database manager. The following output shows the looping over three protocol files:

```
# =====
# db2ACSGetNextObject(): BEGIN [2012-11-20 09:10:17]
RESULT_0_FILE=/home/hotellnx99/jklauke/db2acs.SAMPLE.0.jklauke.1353420375.cfg
# read result object with timestamp 20121120090616
# db2ACSGetNextObject(): END
# =====
# db2ACSGetNextObject(): BEGIN [2012-11-20 09:10:17]
# db2ACSGetNextObject(): END
# =====
```

db2ACSDelete

The database manager calls db2ACSDelete() to delete recovery objects. For every backup image that matches the timestamp (retrieved during the db2ACSGetNextObject() call), the API and the script are called sequentially to delete the backup images and any dependent files.

```
# =====
# db2ACSDelete(): BEGIN [2012-12-13 08:24:44]
DELETE_OBJ_ID=5
# cmd: /home/hotellnx99/jklauke/sqllib/samples/BARVendor/libacssc.sh -a delete
# -o 5 -t 20121213051805
# -c /home/hotellnx99/jklauke/repository/db2acs.0.1355405082.cfg
# Delete old backup file and logs:
# /home/hotellnx99/jklauke/repository/SAMPLE.0.jklauke.0.20121213051805.001.tar
# Delete old configuration file:
# /home/hotellnx99/jklauke/repository/db2acs.SAMPLE.0.jklauke.1355393884.cfg
```

```

RC_DELETE=0
# db2ACSDelete(): END [2012-12-13 08:24:44]
# =====

                db2ACSEndQuery
                The database manager calls db2ACSEndQuery() to terminate the query
                session for backup images.

# =====
# db2ACSEndQuery(): BEGIN [2012-11-20 09:10:19]
# db2ACSEndQuery(): END
# =====

                db2ACSEndOperation
                The database manager calls db2ACSEndOperation() to end the operation.
                The return code of 0 indicates that the deletion was successful.

# =====
# db2ACSEndOperation(): BEGIN [2012-11-20 09:10:19]
END_ACTION=0
# db2ACSEndOperation(): END
# =====

                db2ACSTerminate
                The database manager calls db2ACSTerminate() to terminate the session.

# =====
# db2ACSTerminate(): BEGIN [2012-11-20 09:10:19]
# db2ACSTerminate(): END
# =====

```

DB2 Advanced Copy Services (ACS) user scripts

By providing your own script for snapshot operations, you can use storage hardware that does not provide a vendor library.

A script specifies the type of snapshot operation that you want performed, as well as some additional options. You specify the script name with the **-script** parameter for the appropriate command or API. The library for DB2 ACS invokes the script at various times through the operation.

You have to create the script yourself and ensure that it is executable. There is a sample script called `libacssc.sh` provided in `samples/BARVendor` for your reference. The sample script creates one tar file containing the database files and, if logs are included, a second one for the log files. You can use the sample script as a template for your own script, with the appropriate modifications that set it up for your storage device. You would probably want to remove the section that creates the tar file.

Snapshot backup

During a snapshot backup, the script extracts the information that is required for the current phase from the protocol file and runs the required actions for creating the snapshot. The script writes progress information to the protocol file for debugging reasons.

A snapshot backup script can implement the following actions, preceded by the flag **-a**:

prepare

Runs any actions that need to take place before the snapshot is performed

snapshot

Performs the snapshot

verify Verifies that the snapshot was successfully produced (that is, the vendor tools did not return any errors)

store_metadata

Specifies actions that can occur after the snapshot has been produced and all required metadata has been stored to the protocol file. For example, the script can back up the protocol file or rename the backup image.

rollback

Cleans up the image if a snapshot has failed

Snapshot restore

During snapshot restores the protocol files that were written during snapshots are read, and new protocol files are written (to the same repository) to show the progress of the restore operation. Every restore operation writes a new protocol file. If the restore is successful, the corresponding protocol file is removed. If the operation fails, the protocol file remains for debugging purposes.

A snapshot restore script can implement the following actions, preceded by the flag **-a**:

prepare

Runs any actions that need to take place before the restore is performed

restore

Restores the snapshot backup image

Snapshot management

When a snapshot backup image is deleted, the protocol files that were written during snapshots are read, and new protocol files are written (to the same repository) to show the progress of the delete operation. If the delete operation is successful, the corresponding protocol file is removed. If the operation fails, the protocol file remains for debugging purposes.

A snapshot delete script can implement the following actions, preceded by the flag **-a**:

prepare

Runs any actions that need to take place before the restore is performed

delete Deletes the snapshot backup image

Performing a snapshot backup with a script

Using a custom script allows you to perform snapshot backup operations to storage devices that are not supported by DB2 ACS

Before you begin

You must have one of the following authorities: SYSADM, SYSCTRL, or SYSMAINT.

About this task

Snapshot backups allow you to use the functionality of your underlying storage system to instantly create a copy of all database data and transaction logs without any interruptions. With a custom script, you can specify various options for the snapshot backup operation as well as utilize a wide range of storage devices that do not provide a vendor library.

During online snapshot backups, the database manager temporarily suspends all write operations to disk before creating the snapshot. This ensures that no changes occur to the data during the few seconds when the snapshot is taken.

Procedure

To perform a snapshot backup:

1. Create a script that implements the DB2 ACS API. The script must be executable. For information on custom scripts, see “DB2 Advanced Copy Services (ACS) user scripts” on page 218.
2. Optional: Create a protocol file repository. This directory will contain the protocol files for the snapshot. Ensure that the directory is readable and writable.

If you do not create the repository, the protocol files will be written to the directory that contains your script.

3. Initiate[®] the backup operation using either the **BACKUP DATABASE** command, the **ADMIN_CMD** procedure with **BACKUP DB** option, or the **db2Backup** API.

BACKUP DATABASE command

```
BACKUP DATABASE dbname ONLINE  
USE SNAPSHOT SCRIPT path-to-script  
OPTIONS 'path-to-repository additional options'
```

ADMIN_CMD procedure

```
CALL SYSPROC.ADMIN_CMD  
  (backup database dbname online  
   use snapshot script path-to-script  
   options 'path-to-repository additional options')
```

db2Backup API

```
int sampleBackupFunction( char dbAlias[],  
                          char user[],  
                          char pswd[],  
                          char workingPath[] )  
{  
    db2MediaListStruct mediaListStruct = { 0 };  
  
    mediaListStruct.locations = &workingPath;  
    mediaListStruct.numLocations = 1;  
    mediaListStruct.locationType = SQLU_SNAPSHOT_SCRIPT_MEDIA;  
  
    db2BackupStruct backupStruct = { 0 };  
  
    backupStruct.piDBAlias = dbAlias;  
    backupStruct.piUsername = user;  
    backupStruct.piPassword = pswd;  
    backupStruct.piVendorOptions = NULL;  
    backupStruct.piMediaList = &mediaListStruct;  
    db2Backup(db2Version1050, &backupStruct, &sqlca);  
  
    return 0;  
}
```

Results

The snapshot operation generates a snapshot backup image and a protocol file. Ensure that you keep the protocol file so it can be used for subsequent restore, query, delete operations.

Restoring from a snapshot backup image with a script

Using a custom script allows you to restore snapshot backup images taken using storage devices that are not supported by DB2 ACS.

Before you begin

You must have one of the following authorities: SYSADM, SYSCTRL, or SYSMAINT.

About this task

A snapshot restore operation restores a snapshot backup. You must use a custom script for that restore operation if your storage device does not provide a vendor library.

During snapshot restore operations, the protocol files that were written during the snapshot backup are read. As well, a new protocol file is written for the restore operation to show its progress. If the restore operation is successful, the protocol file is deleted; if the operation fails, you can use the protocol file to help investigate the cause of the failure.

A restore operation restores the latest image that matches the specified time stamp. For example, if there are two images for the time stamp 20121120, one taken at 201211201000 and one taken at 201211202000, the last one is chosen.

Restrictions

Procedure

To perform a snapshot restore:

1. Create a script that implements the DB2 ACS API. The script must be executable. For information on custom scripts, see “DB2 Advanced Copy Services (ACS) user scripts” on page 218.
2. Initiate the restore operation using either the **RESTORE DATABASE** command, the ADMIN_CMD procedure with RESTORE DATABASE option, or the db2Restore API.

RESTORE DATABASE command

```
RESTORE DATABASE dbname  
USE SNAPSHOT SCRIPT path-to-script  
OPTIONS 'path-to-repository'  
TAKEN AT timestamp LOGTARGET INCLUDE
```

ADMIN_CMD procedure

```
CALL SYSPROC.ADMIN_CMD  
  (restore database dbname  
   use snapshot script path-to-script  
   options 'path-to-repository'  
   taken at timestamp logtarget include)
```

db2Restore API

```
int sampleRestoreFunction( char dbAlias[],
                           char restoredDbAlias[],
                           char user[],
                           char pswd[],
                           char workingPath[] )
{
    db2MediaListStruct mediaListStruct = { 0 };

    rmediaListStruct.locations = &workingPath;
    rmediaListStruct.numLocations = 1;
    rmediaListStruct.locationType = SQLU_SNAPSHOT_SCRIPT_MEDIA;

    db2RestoreStruct restoreStruct = { 0 };

    restoreStruct.piSourceDBAlias = dbAlias;
    restoreStruct.piTargetDBAlias = restoredDbAlias;
    restoreStruct.piMediaList = &mediaListStruct;
    restoreStruct.piUsername = user;
    restoreStruct.piPassword = pswd;
    restoreStruct.iCallerAction = DB2RESTORE_STORDEF_NOINTERRUPT;

    struct sqlca sqlca = { 0 };

    db2Restore(db2Version1050, &restoreStruct, &sqlca);

    return 0;
}
```





Part 5. Utilities

Use the IBM InfoSphere Optim Query Workload Tuner wizards and advisors and the DB2 utilities to manage new features introduced in DB2 10.5.

Chapter 23. IBM InfoSphere Optim Query Workload Tuner for DB2 for Linux, UNIX, and Windows

IBM InfoSphere Optim Query Workload Tuner improves performance and cuts costs by helping database administrators and SQL developers optimize the performance of SQL statements in applications that query DB2 databases. Built-in integration with the Data Studio client helps to find problematic queries in development. Database administrators can further tune performance by optimizing statistics, creating or modifying indexes, creating MQTs, converting tables to MDC tables, distributing data across partitions, and creating optimization profiles.

Related information:

-  [IBM InfoSphere Optim Query Workload Tuner documentation](#)
-  [IBM InfoSphere Optim Query Workload Tuner information roadmap](#)
-  [IBM InfoSphere Optim Query Workload Tuner for DB2 for Linux, UNIX and Windows product web page](#)
-  [Download IBM InfoSphere Optim Query Workload Tuner from IBM Passport Advantage](#)



Workload Table Organization Advisor

The Workload Table Organization Advisor recommends converting row-organized tables to column-organized tables.

The advisor examines all of the tables that are referenced by the statements that are in a query workload and makes recommendations for the query workload as a whole.

The advisor presents its analysis and rationales so that you can see the tables that are recommended for conversion, as well as those tables that are not. It displays the cardinalities of the tables, how many statements in the query workload reference them, the cumulative cost of running the statements that reference a table, the statements themselves, and more. You can view and save the DDL for implementing the changes that are necessary to convert the tables.

Related information:


-  [Generating and acting on recommendations for converting tables to column organization](#)
-  [Virtually testing conversion of tables to column-organization](#)

Workflow assistant

The workflow assistant can help you understand the structure of an SQL statement, understand the access path for that statement, and compare access paths

Use the Compare section of the workflow assistant for query tuning to compare two access plan graphs. The Compare section lists and highlights the differences. By comparing access plans, you can determine whether changes that you made to an SQL statement improved the access plan for that statement.

Related information:

 Comparing access plan to see the results of tuning single SQL statements

Access Plan Explorer

The Access Plan Explorer shows the operations that are in the access plan and details about each operation.

You can view the operations in a table, in which you can sort the operations by cost, or as a tree.

Use the Access Plan Explorer to find the most expensive components of an access plan and the relationships between components. The nodes that appear in representations of access plans for the SQL statements correspond to the DB2 explain operators including the new CTQ operator.

Related information:

 Browsing access plans with the Access Plan Explorer

Chapter 24. Converting row-organized tables to column-organized tables

To add columnar capabilities to your DB2 database, convert row-organized tables to column-organized tables.

Before you begin

- You must have enough authorization to run the **db2convert** command.

About this task

Restrictions

- You cannot convert the following types of row-organized tables:
 - Range clustered tables
 - Typed tables
 - Materialized query tables
 - Declared global temporary tables
 - Created global temporary tables
- You cannot convert row-organized tables that are in partitioned database environments or nonautomatic storage table spaces
- You cannot convert row-organized tables that have unique indexes on nullable columns, generated columns, columns of type BLOB, DBCLOB, CLOB, or XML.
- The REORG option is not supported.
- Restrictions for the ADMIN_MOVE_TABLE procedure also apply to row-organized tables because the **db2convert** command calls the ADMIN_MOVE_TABLE procedure.

Procedure

To convert row-organized tables to column-organized tables:

1. Ensure that you have a backup of the row-organized tables that you want to convert.
2. Invoke the **db2convert** command, specifying the **-stopBeforeSwap** parameter. If you are converting range partitioned tables, MDC tables, or ITC tables, the **-force** parameter. These types of tables are not converted by default.
3. Perform an online backup of the target table space or table spaces.
4. Invoke the **db2convert** command, specifying the **-continue** parameter.

Example

Converting one table

```
db2convert -d database_name -stopBeforeSwap -z schema_name -t row_organized_table_name
db2 BACKUP DATABASE database_name TABLESPACE (tablespace_name) ONLINE
db2convert -d database_name -continue -z schema_name -t row_organized_table_name
```

Chapter 25. Loading data into column-organized tables

Although loading data into column-organized tables is very similar to loading data into row-organized tables, you should be aware of the few exceptions and configure your database to handle the additional resources that are required.

Before you begin

- You must have an established connection to the database or be able to implicitly connect to the database into which you want to load the data.

About this task

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

In addition, statistics are collected during the load according to the profile defined by the RUNSTATS command.

Restrictions

- Check the restrictions of the **LOAD** command as they also apply to row-organized tables.

Procedure

To load data in column-organized tables:

1. Set the **blocknonlogged** (block creation of tables that allow non-logged activity) database configuration parameter to NO before you load data into a column-organized table. If this parameter is not set, the error message SQL2032N is returned.
2. Ensure that the **util_heap_sz** (utility heap size) database configuration parameter is set to at least 1,000,000 pages and AUTOMATIC to address the resource needs of the LOAD command.

If the database server has at least 128 GB of memory, set **util_heap_sz** 4,000,000 pages. If concurrent load utilities are running, increase the **util_heap_sz** value to accommodate higher memory requirements. If memory is scarce, the **util_heap_sz** value can be increased dynamically only when a load operation is running.

If you set **DB2_WORKLOAD** to ANALYTICS before the creation of your database, the **util_heap_sz** value is automatically configured during database creation.

3. Ensure that the path for load temporary files has sufficient storage space (equivalent to the raw size of the input data) to build the column compression dictionary.

If a column compression dictionary must be built, the input data source (such as a pipe or a socket) is processed twice. If the input source can be reopened, it is read twice. If the input source cannot be reopened, its contents are temporarily cached in the load temporary file directory.

The default path for load temporary files is located under the instance directory, or in a location that is specified by the `TEMPFILES PATH` option on the **LOAD** command.

4. Ensure that you have enough memory.

Memory requirements temporarily increase when the column compression dictionary is being created.

For optimal load performance, additional cache memory is required to write column-organized data in extent-sized amounts, rather than one page at a time, therefore reducing I/O costs.

Part 6. Appendixes

Appendix A. DB2 technical information

DB2 technical information is available in multiple formats that can be accessed in multiple ways.

DB2 technical information is available through the following tools and methods:

- Online DB2 documentation in IBM Knowledge Center:
 - Topics (task, concept, and reference topics)
 - Sample programs
 - Tutorials
- Locally installed DB2 Information Center:
 - Topics (task, concept, and reference topics)
 - Sample programs
 - Tutorials
- DB2 books:
 - PDF files (downloadable)
 - PDF files (from the DB2 PDF DVD)
 - Printed books
- Command-line help:
 - Command help
 - Message help

Important: The documentation in IBM Knowledge Center and the DB2 Information Center is updated more frequently than either the PDF or the hardcopy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 documentation in IBM Knowledge Center.

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks® publications online at [ibm.com](http://www.ibm.com). Access the DB2 Information Management software library site at <http://www.ibm.com/software/data/sw-library/>.

Documentation feedback

The DB2 Information Development team values your feedback on the DB2 documentation. If you have suggestions for how to improve the DB2 documentation, send an email to db2docs@ca.ibm.com. The DB2 Information Development team reads all of your feedback but cannot respond to you directly. Provide specific examples wherever possible to better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use the db2docs@ca.ibm.com email address to contact DB2 Customer Support. If you have a DB2 technical issue that you cannot resolve by using the documentation, contact your local IBM service center for assistance.

DB2 technical library in PDF format

The DB2 technical library is available in PDF format free of charge.

You can download English and translated DB2 Version 10.5 manuals in PDF format from DB2 database product documentation at www.ibm.com/support/docview.wss?rs=71&uid=swg27009474.

The DB2 documentation online in IBM Knowledge Center is updated more frequently than the manuals in PDF format.

Starting July 29th 2013, the IBM Publications Center no longer supports ordering of printed manuals in accordance to e-Business strategy.

The following tables describe the DB2 library available in PDF format. The form number increases each time a manual is updated. Ensure that you are reading the most recent version of the manuals, as listed in the tables.

Table 23. DB2 technical information

Name	Form Number	Availability date
<i>Administrative API Reference</i>	SC27-5506-00	July 28, 2013
<i>Administrative Routines and Views</i>	SC27-5507-00	July 28, 2013
<i>Call Level Interface Guide and Reference Volume 1</i>	SC27-5511-00	July 28, 2013
<i>Call Level Interface Guide and Reference Volume 2</i>	SC27-5512-00	July 28, 2013
<i>Command Reference</i>	SC27-5508-00	July 28, 2013
<i>Database Administration Concepts and Configuration Reference</i>	SC27-4546-00	July 28, 2013
<i>Data Movement Utilities Guide and Reference</i>	SC27-5528-00	July 28, 2013
<i>Database Monitoring Guide and Reference</i>	SC27-4547-00	July 28, 2013
<i>Data Recovery and High Availability Guide and Reference</i>	SC27-5529-00	July 28, 2013
<i>Database Security Guide</i>	SC27-5530-00	July 28, 2013
<i>DB2 Workload Management Guide and Reference</i>	SC27-5520-00	July 28, 2013
<i>Developing ADO.NET and OLE DB Applications</i>	SC27-4549-00	July 28, 2013
<i>Developing Embedded SQL Applications</i>	SC27-4550-00	July 28, 2013
<i>Developing Java Applications</i>	SC27-5503-00	July 28, 2013
<i>Developing Perl, PHP, Python, and Ruby on Rails Applications</i>	SC27-5504-00	July 28, 2013
<i>Developing RDF Applications for IBM Data Servers</i>	SC27-5505-00	July 28, 2013
<i>Developing User-defined Routines (SQL and External)</i>	SC27-5501-00	July 28, 2013
<i>Getting Started with Database Application Development</i>	GI13-2084-00	July 28, 2013
<i>Getting Started with DB2 Installation and Administration on Linux and Windows</i>	GI13-2085-00	July 28, 2013
<i>Globalization Guide</i>	SC27-5531-00	July 28, 2013
<i>Installing DB2 Servers</i>	GC27-5514-00	July 28, 2013
<i>Installing IBM Data Server Clients</i>	GC27-5515-00	July 28, 2013
<i>Message Reference Volume 1</i>	SC27-5523-00	July 28, 2013
<i>Message Reference Volume 2</i>	SC27-5524-00	July 28, 2013

Table 23. DB2 technical information (continued)

Name	Form Number	Availability date
<i>Net Search Extender Administration and User's Guide</i>	SC27-5526-00	July 28, 2013
<i>Partitioning and Clustering Guide</i>	SC27-5532-00	July 28, 2013
<i>pureXML Guide</i>	SC27-5521-00	July 28, 2013
<i>Spatial Extender User's Guide and Reference</i>	SC27-5525-00	July 28, 2013
<i>SQL Procedural Languages: Application Enablement and Support</i>	SC27-5502-00	July 28, 2013
<i>SQL Reference Volume 1</i>	SC27-5509-00	July 28, 2013
<i>SQL Reference Volume 2</i>	SC27-5510-00	July 28, 2013
<i>Text Search Guide</i>	SC27-5527-00	July 28, 2013
<i>Troubleshooting and Tuning Database Performance</i>	SC27-4548-00	July 28, 2013
<i>Upgrading to DB2 Version 10.5</i>	SC27-5513-00	July 28, 2013
<i>What's New for DB2 Version 10.5</i>	SC27-5519-00	July 28, 2013
<i>XQuery Reference</i>	SC27-5522-00	July 28, 2013

Table 24. DB2 Connect-specific technical information

Name	Form Number	Availability date
<i>Installing and Configuring DB2 Connect Servers</i>	SC27-5517-00	July 28, 2013
<i>DB2 Connect User's Guide</i>	SC27-5518-00	July 28, 2013

Displaying SQL state help from the command line processor

DB2 products return an SQLSTATE value for conditions that can be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

Procedure

To start SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.

For example, ? 08003 displays help for the 08003 SQL state, and ? 08 displays help for the 08 class code.

Accessing DB2 documentation online for different DB2 versions

You can access online the documentation for all the versions of DB2 products in IBM Knowledge Center.

About this task

All the DB2 documentation by version is available in IBM Knowledge Center at <http://www.ibm.com/support/knowledgecenter/SSEPGG/welcome>. However, you can access a specific version by using the associated URL for that version.

Procedure

To access online the DB2 documentation for a specific DB2 version:

- To access the DB2 Version 10.5 documentation, follow this URL:
http://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.kc.doc/welcome.html.
- To access the DB2 Version 10.1 documentation, follow this URL:
http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.kc.doc/welcome.html.
- To access the DB2 Version 9.8 documentation, follow this URL:
http://www.ibm.com/support/knowledgecenter/SSEPGG_9.8.0/com.ibm.db2.luw.kc.doc/welcome.html.
- To access the DB2 Version 9.7 documentation, follow this URL:
http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.kc.doc/welcome.html.
- To access the DB2 Version 9.5 documentation, follow this URL:
http://www.ibm.com/support/knowledgecenter/SSEPGG_9.5.0/com.ibm.db2.luw.kc.doc/welcome.html.

Terms and conditions

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability: These terms and conditions are in addition to any terms of use for the IBM website.

Personal use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the previous instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Trademarks: IBM, the IBM logo, and ibm.com[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at www.ibm.com/legal/copytrade.shtml

Appendix B. Notices

This information was developed for products and services offered in the U.S.A. Information about non-IBM products is based on information available at the time of first publication of this document and is subject to change.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements, changes, or both in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to websites not owned by IBM are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

Such information may be available, subject to appropriate terms and conditions, including, in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle, its affiliates, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

- access plan diagrams
 - description 48
 - example 48
 - setting preferences 49
- access plans
 - diagramming 46
- actual parameter 104
- adaptive compression
 - details 85
 - dictionaries 91
- ADC (automatic dictionary creation)
 - details 92
- ADMIN_COPY_SCHEMA procedure
 - overview 225
- alert log 104
- ALTER TABLE statement
 - enabling compression 89
- analytic workloads
 - database configuration 3
- analytics
 - column-organized tables scenario 63
- archive log 104
- archive log mode 104
- automatic dictionary creation (ADC)
 - details 92
- automatic reorganization
 - enabling 12

B

- backups
 - compression 98
- bdump directory 104
- binding
 - database utilities 198
- BLU Acceleration
 - column-organized tables
 - overview 57
 - synopsis tables
 - overview 58

C

- CEAL
 - description 173
- CECL
 - description 173
- chains
 - job manager 42
- classic row compression
 - details 83
 - dictionaries 91
- column compression
 - dictionaries 91
- column compression dictionaries
 - after load or insert operation 95
- column-organized tables
 - converting row-organized tables 227
 - creating 59

- column-organized tables (*continued*)
 - database configuration 3
 - DELETE statement restrictions 61
 - enabling parallelism 4
 - explain information 125
 - high-speed analytics 63
 - INSERT statement restrictions 61
 - loading data 62, 229
 - MERGE statement restrictions 61
 - monitoring metrics 117
 - overview 57
 - query concurrency 13
 - space reclamation 11
 - synopsis tables 58
 - system and database configurations 58
 - UPDATE statement restrictions 61
- command management
 - Administration Explorer 29, 36
 - Object List 29, 36
- commands
 - db2pd
 - details 140
 - invoking 31
 - running 31
 - support in an Object List 29, 36
 - support in the Administration Explorer 29, 36
 - types of database administration 29, 36
- committed code level
 - see CECL 173
- compatibility
 - features summary 99
- compression
 - adaptive 85
 - backup 98
 - classic row 83
 - default system values 82
 - estimating storage savings 86
 - index
 - details 95
 - NULL values 82
 - overview 81
 - row
 - adaptive 85
 - classic 83
 - overview 83
 - tables
 - changing 90
 - column values 82
 - creating 87
 - disabling 90
 - enabling 89
 - overview 81
 - temporary tables 83, 85
 - value 82
- compression dictionaries
 - adaptive compression 85
 - automated creation 92
 - classic row compression 83
 - column 95
 - overview 91

- concurrency
 - management 13
- configuration parameters
 - date_compat 104
 - number_compat 104
 - varchar2_compat 104
- constraints
 - creating
 - overview 77
 - expression-based 65
 - informational 75, 76
 - modifying 77
- constructs
 - multiple query blocks 48
- converting tables 36
- current effective architecture level
 - see CEAL 173
- current effective code level
 - see CECL 173
- cursor sharing 104

D

- data block 104
- data buffer cache 104
- data dictionaries
 - DB2-Oracle terminology mapping 104
 - Oracle
 - compatible views 109
- data dictionary cache 104
- data file 104
- data management tools
 - Data Studio 23
 - InfoSphere Optim Query Tuner 225
- data movement
 - tools 225
- Data Studio
 - key tasks 23
- Data Studio web console
 - overview 28
- database administration
 - commands 29, 36
 - invoking commands 31
 - running commands 31
 - support in an Object List 29, 36
 - support in the Administration Explorer 29, 36
- database administration commands
 - cluster caching facilities 30
 - Database Partitioning Facility (DPF) 30
 - DB2 pureScale Feature 30
 - DB2 pureScale members 30
 - partitioned databases 30
- database configuration
 - analytic workloads 3
- database creation
 - analytic workloads 3
- database links
 - syntax 110
 - terminology mapping 104
- database objects
 - overview 51
- database operations
 - online fix pack updates 176
- databases
 - backing up 29, 36
 - configuring 29, 36
 - creating 29, 36
- databases (*continued*)
 - dropping 29, 36
 - forcing applications off 29, 36
 - High Availability Disaster Recovery (HADR) 29, 36
 - monitoring
 - overview 115
 - quiescing 29, 36
 - recovering 29, 36
 - restarting 29, 36
 - restoring 29, 36
 - rolling forward log files 29, 36
 - starting 29, 36
 - stopping 29, 36
 - unquiescing 29, 36
- date_compat database configuration parameter
 - overview 104
- DB2 Advanced Copy Services (ACS)
 - protocol file
 - description 207
 - usage 207
 - scripted interface 207
 - user scripts
 - description 218
 - usage 218
- DB2 documentation
 - available formats 233
- DB2 documentation versions
 - IBM Knowledge Center 235
- DB2 pureScale environments
 - EHL
 - details 203
 - fix packs
 - installing 176
 - HADR
 - adding members 162
 - overview 155
 - preferred replay member 160, 161
 - removing members 163
 - replay member 160
 - standby replay 160
 - topology changes 161, 162, 163
 - self-tuning memory 201
- DB2 pureScale Feature
 - database administration commands 30
- DB2 pureScale instances
 - installing
 - online fix pack updates 182
 - online fix pack updates in a GDPC environment 191
 - online fix pack updates 173
 - updating 197
- DB2 pureScale members
 - cluster caching facilities
 - database administration commands 30
 - database administration commands 30
- DB2 servers
 - overview 1
- DB2 technical library
 - PDF files 234
- DB2 workload management
 - concurrency control 16
 - default objects for concurrency control 16
- DB2_COMPATIBILITY_VECTOR registry variable
 - details 100
- db2iupdt command
 - DB2 pureScale environments
 - updating instance to higher level within release 197

- db2move command
 - overview 225
- db2pd command
 - hadr changes 137
 - details 140
- db2relocatedb command
 - overview 225
- deep compression
 - See adaptive compression 85
 - See classic row compression 83
- default objects for concurrency control 16
- DELETE statement
 - column-organized tables 61
- dictionaries
 - column compression 95
 - compression 91
- documentation
 - PDF files 234
 - terms and conditions of use 236
- DPF
 - database administration commands 30
- dynamic performance views 104

E

- EHL
 - details 203
 - use case 204
- enabling parallelism
 - column-organized tables 4
- explain facility
 - column-organized tables 125
- explicit hierarchical locking
 - performance 205
 - See EHL 203
- export utility
 - overview 225
- expression-based indexes
 - statistics
 - automatic collection 65
 - manual collection 66
 - overview 65
 - RUNSTATS command 66
 - statistics profiles 67

F

- fix packs
 - DB2 pureScale instances
 - online update 173
 - downloading 179
 - installing
 - DB2 pureScale environments 176
 - prerequisites 177
 - updating DB2 pureScale instance 194
 - post-installation
 - binding manually 198
 - tasks (Linux) 195
 - tasks (UNIX) 195
 - prerequisites 178, 180
 - uncompressing 180
- formal parameter 104
- functions
 - table
 - MON_GET_HADR 137
 - MON_GET_ROUTINE 135

G

- global index 104

H

- HADR 29, 36
 - DB2 pureScale environments
 - adding members 162
 - CFs 156
 - member subsetting 156
 - overview 155
 - preferred replay member 160
 - replay member 160
 - restrictions 155
 - scenario 165
 - setup 156
 - standby members 156
 - standby replay 160
 - topology changes 161, 162
 - failover
 - pureScale environment 164
 - installing
 - online fix pack updates 188
 - monitoring
 - DB2 pureScale environment 157
 - takeover
 - pureScale environment 164
- help
 - SQL statements 235
- heterogeneous instances
 - online fix pack updates 176
- high availability
 - designing 149
 - outages
 - overview 149
- high availability disaster recovery (HADR)
 - DB2 pureScale environment
 - preferred replay member 161
 - removing members 163
 - topology changes 163
 - preferred replay member 161
- High Availability Disaster Recovery (HADR) 29, 36
- history
 - job manager 42
- homogeneous instances
 - online fix pack updates 176

I

- IBM Data Studio
 - key tasks 23
 - overview 23
- IBM Knowledge Center
 - DB2 documentation versions 235
- import utility
 - overview 225
- inactive log 104
- index compression
 - details 95
 - restrictions 95
- index reorganization
 - automatic 12
- indexes
 - expression-based
 - statistics 65
 - reorganizing 29, 36

- informational constraints
 - designing 76
 - details 75
- InfoSphere Optim Query Tuner
 - overview 225
- ingest utility
 - overview 225
- init.ora 104
- inplace table reorganization
 - overview 151
- insert operations
 - column compression dictionaries 95
- INSERT statement 61
- installation
 - fix packs
 - bind files 198
 - DB2 pureScale instance 194
 - obtaining 179
 - post-installation tasks 195
 - pre-installation tasks 177
 - prerequisites 178, 180
 - uncompressing 180
- installing
 - fix packs
 - DB2 pureScale environments 176
 - online fix pack updates 182
 - HADR 188
 - online fix pack updates in a GDPC environment 191
- instance operations
 - online fix pack updates 176
- instances
 - configuring 29, 36
 - quiescing 29, 36
 - starting 29, 36
 - stopping 29, 36
 - unquiescing 29, 36
- intrapartition parallelism
 - enabling 5

J

- job manager
 - chains 42
 - create jobs 42, 43
 - history 42
 - manage jobs 42
 - notifications 42
 - schedules 42
- job type
 - DB2 CLP scripts
 - SSH 40
 - executable/shell scripts
 - ssh 40
 - SQL-only scripts 40
- jobs
 - job manager 42
 - job type 40

L

- large pool 104
- library cache 104
- load utility
 - overview 225
- loading data
 - column-organized tables 62, 229

- loads
 - column compression dictionaries 95
- local index 104

M

- materialized view 104
- MERGE statement 61
- metrics
 - column-organized tables 117
- MON_GET_HADR
 - changes 137
- MON_GET_HADR table function
 - details 137
- MON_GET_ROUTINE table function 135
- monitoring
 - column-organized tables 117
 - databases 115
 - db2pd command 140
 - high availability disaster recovery (HADR)
 - DB2 pureScale environment 157
- multiple query blocks 48

N

- new features
 - highlights 53
- noarchive log mode 104
- nodes
 - setting preferences 49
- notices 239
- notifications
 - job manager 42
- number_compat database configuration parameter
 - overview 104

O

- online DB2 documentation
 - IBM Knowledge Center 235
- online fix pack updates
 - database operations 176
 - DB2 pureScale instances 173
 - instance operations 176
- online table reorganization
 - concurrency 152
 - details 151
 - locking 152
 - performing 152
- operators
 - TQ 128
- Oracle
 - application enablement 111
 - data dictionary--compatible views 109
 - database link syntax 110
 - DB2 terminology mapping 104
- Oracle Call Interface (OCI) 104
- ORACLE_SID environment variable 104
- overview
 - Data Studio web console 28

P

- pages
 - sizes
 - extended rows 71
- parallelism
 - intra-partition
 - enabling 5
- partitioned databases
 - database administration commands 30
- performance
 - improving with indexes 65
 - indexes
 - improving performance 65
- PL/SQL
 - Oracle application enablement 111
- program global area (PGA) 104

Q

- query concurrency
 - default management 13

R

- reclaimable storage
 - compressed tables 83, 85
- records
 - length enhancement 71
- redo log 104
- registry variables
 - DB2_COMPATIBILITY_VECTOR 100
- reorganization
 - indexes
 - automatic 12
 - tables
 - automatic 12
 - online (concurrency) 152
 - online (details) 151
 - online (locking) 152
 - online (procedure) 152
- restore utility
 - GENERATE SCRIPT option 225
 - REDIRECT option 225
- restores
 - snapshot backup
 - with script 221
- routines
 - monitoring
 - table functions 135
- row compression
 - estimating storage savings 86
 - overview 83
 - See also classic row compression 83
- row sizes
 - enhancement 71
- row-organized tables
 - converting to column-organized tables 227
- rows
 - extended size 71
- RUNSTATS command
 - expression-based indexes 66

S

- scenario
 - create jobs 43
- scenarios
 - column-organized data 63
 - high-speed analytics with column-organized data 63
- schedules
 - job manager 42
- segment 104
- self-tuning memory
 - DB2 pureScale environments 201
- Server Parameter File (SPFILE) 104
- service subclasses
 - default 16
- session 104
- snapshot backups
 - performing
 - with script 219
 - restoring
 - with script 221
- space reclamation
 - column-organized tables 11
- split mirrors
 - overview 225
- SQL statements
 - diagramming access plans 46
 - help
 - displaying 235
 - invoking 31
 - running 31
 - support in an Object List 29, 36
 - support in the Administration Explorer 29, 36
- ssh
 - DB2 CLP scripts 40
 - executable/shell scripts 40
- startup nomount 104
- storage
 - compression
 - classic row 83
 - indexes 95
 - reclaiming storage freed 83, 85
 - row 85
 - table 81
 - estimating savings offered by compression 86
- synonyms
 - DB2-Oracle terminology mapping 104
- system and database configurations
 - column-organized tables 58
- system global area (SGA) 104
- SYSTEM table space 104

T

- table compression
 - changing 90
 - creating tables 87
 - disabling 90
 - enabling 89
 - overview 81
- table functions
 - MON_GET_ROUTINE 135
 - monitoring
 - routines 135
- table spaces
 - backing up 29, 36
 - restoring 29, 36

- table spaces (*continued*)
 - rolling forward log files 29, 36
- tables
 - adaptive compression 85
 - classic row compression 83
 - column-organized
 - overview 57
 - compression
 - adaptive 85
 - changing 90
 - classic row 83
 - disabling 90
 - value 82
 - decompressing 90
 - exporting data 29, 36
 - extended row size 71
 - importing data 29, 36
 - loading data 29, 36
 - online reorganization
 - details 151
 - organization
 - configuring 9
 - reorganization
 - automatic 12
 - online 152
 - setting integrity 29, 36
 - synopsis
 - overview 58
 - unloading data 29, 36
- temporary tables
 - adaptive compression 85
 - classic row compression 83
- terminology mapping
 - DB2-Oracle 104
- terms and conditions
 - publications 236
- thresholds
 - default 16
- TQ operator 128
- troubleshooting
 - db2pd command 140

U

- update DB2 pureScale instances
 - online fix pack updates
 - installing 182
 - online fix pack updates in a GDPC environment
 - installing 191
- update HADR DB2 pureScale instances
 - online fix pack updates
 - installing 188
- UPDATE statement
 - column-organized tables 61
- updates
 - DB2 pureScale instances 197
- user global area (UGA) 104
- utilities
 - invoking 31
 - running 31
 - support in an Object List 29, 36
 - support in the Administration Explorer 29, 36
- utility management
 - Administration Explorer 29, 36
 - Object List 29, 36

V

- value compression 82
- varchar2_compat database configuration parameter
 - overview 104
- views
 - Oracle data dictionary compatibility 109
- Visual Explain
 - appearance 49
 - constructs 46
 - diagramming access plans 46
 - explain data 49
 - nodes
 - appearance 49
 - purpose 46
 - running traces 46
 - setting preferences 49
 - special registers 46, 49
 - terminator 46
 - working directory 46

W

- work action sets
 - default 16
- work actions
 - default 16
- work class sets
 - default 16
- work classes
 - default 16

X

- XPATH statements
 - diagramming access plans 46



Printed in USA

SC27-5575-00



Spine information:

Preparation Guide for DB2 10.5 DBA for LUW Upgrade from DB2 10.1 Exam 311

