Infor LN Performance, Tracing and Tuning Guide for DB2

Publication Information

Release: Infor LN 10.x

Publication date: November 14, 2012

Document code: B0077B

Contents

About this guide **5**
Intended audience 5
Related documents 5
Contacting Infor 6
Chapter 1 Tuning DB2 for Infor LN **7**
Introduction 7
Creating statistics 7
DB2 compression 7
DB2 settings 8
Database manager configuration tuning 9
Database configuration tuning 9
Call Level Interface configuration 9
DB2 environment parameters 10
Log reader and log writer priority using DB2_RESOURCE_POLICY 10
DB2_USE_FAST_PREALLOCATION 11
DB2_MINIMIZE_LISTPREFETCH 12
DB2_SKIPINSERTED 12
DB2_SKIPDELETED 13
DB2_EVALUNCOMMITTED 13
DB2_USE_ALTERNATE_PAGE_CLEANING 14
DB2_LOGGER_NON_BUFFERED_IO 14
DB2_AVOID_PREFETCH 15
DB2LOCK_TO_RB 15
DB2COMM=tcpip 15
DB2_PARALLEL_IO 16
Chapter 2 Tuning Infor LN for DB2 ..... **19**
DB2 db_resource parameters 19
db2_opt_level 19

db2_opt_rows 19
db2_max_open_handles 19
db2_retained_cursors 20
Array interface 20
Batches 21
Database storage and driver files 21
DB2 storage file 21
DB2 driver parameter file 21

Chapter 3 Tracing DB2 23

DB2 Explain facility 23
Formatting/viewing the content of explain tables 24
db2exfmt 24
Visual Explain 24
Dynamic SQL snapshot 25
To interpret the output 27
DB2 Explain 27
Dynamic SQL snapshot 27
IBM InfoSphere Optim Performance Manager 27

About this guide

This document provides guidelines to improve the performance of an Infor LN environment on a DB2 database by tracing and tuning the environment.

All information is based on the use of the Infor LN software. If you require information about other versions, read the relevant documentation.

Note: This document is a comprehensive compilation; however there may be instances wherein relevant information or procedures may have been omitted. Therefore, we strongly recommend verifying the proposed changes in a test environment before moving to production. The information provided may not hold true for future versions of the DB2 database.

Intended audience

This document is intended for intermediate to expert Infor LN and database Administrators and Technical Consultants in order to get optimal performance out of an Infor LN system.

Related documents

Certain sections in this document are described in more detail in other documents. The following documents help to extend the knowledge in particular areas.

- *Infor LN - Performance, Tracing and Tuning Guide (U9357 US)*
- *Infor LN - Sizing guide (B0045 US)*
- *Infor Enterprise Server - Technical Reference Guide for DB2 Database Driver (U7829 US)*

You can find the documents in the product documentation section of the Infor Xtreme Support portal, as described in "Contacting Infor" on page 6.

A good reference guide on DB2 documentation is important. We recommend the following site: http://pic.dhe.ibm.com/infocenter/db2luw/v10r1

Contacting Infor

If you have questions about Infor products, go to the Infor Xtreme Support portal at www.infor.com/inforxtreme.

If we update this document after the product release, we will post the new version on this Web site. We recommend that you check this Web site periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.

Chapter 1 Tuning DB2 for Infor LN

1

The performance of Infor LN depends highly on the database performance. This chapter covers the most important tuning areas of DB2.

Introduction

Every new version of a database comes with new settings, tools, and so on. Although Infor LN runs relatively well with an out-of-the box DB2 database, you should perform some tuning. DB2 strives for a self-learning and tuning database, but you must perform tuning for optimal performance. For an out-performing DB2 database, you must have DB2 database knowledge.

IBM and partners provide courses on generic DB2 tuning; the Internet also provides information. The DB2 documents at http://pic.dhe.ibm.com/infocenter/db2luw/v10r1 are a good starting point.

Creating statistics

You create statistics in DB2 so that the optimizer can select the best possible plan. Therefore, you should create a script, which will save the name of all the existing tables in a file and run runstats on this file. As data changes, we recommend that you regularly run such a script.

```
runstats on table <DB name>.<table name> WITH DISTRIBUTION AND DETAILED INDEXES ALL
SHRLEVEL CHANGE
```

To save all tables to a file, use the following statement:

```
db2 -r tables.txt "select name from sysibm.systables where creator = '<DB name>’
```

DB2 compression

Data may grow faster than expected, particularly if too much history is saved or financial logging is turned on. Therefore, some tables will be large. To reduce the data increase, you can change application parameters. To reduce existing data, save storage and probably increase performance, use DB2 row compression.

To see how much space you can save when compressing the table, use the DB2 INSPECT command. DB2 INSPECT has an option to report on the number of pages saved if you were to implement compression on a given table. The syntax is as follows:

```
DB2 INSPECT ROWCOMPESTIMATE TABLE NAME table_name RESULTS KEEP file_name
```

To convert the binary output file of DB2 inspect into a readable text file, run the following command:

```
db2inspf file_name output_file_name
```

The output file contains the estimated percentage of data pages saved from compression.

To use compression in DB2:

1. To use row compression in DB2, you must first set the table to be compression eligible. Then you must generate the dictionary that contains the common strings from within the table. To set the table to be eligible for compression, use one of the following:

   ```
   CREATE TABLE table_name … COMPRESS YES
   ```

   Or

   ```
   ALTER TABLE table_name COMPRESS YES
   ```

2. To enable DB2 to scan the data in the table to find the common components that it can compress out of the table and place in a dictionary, use the REORG TABLE command. The first time you compress a table, or if you want to rebuild the dictionary, you must run the REORG TABLE command:

   ```
   REORG TABLE table_name RESETDICTIONARY
   ```

3. Running this command scans the table, creates the dictionary, and performs the actual table reorganization, which compresses the data as it goes. From this point onward, any insert into this table or subsequent load of data uses the compression dictionary and compresses any new data in the table. If in the future you want to run a normal table reorganization and not rebuild the dictionary, run the following command:

   ```
   REORG TABLE table_name KEEPDICTIONARY
   ```

   Each table object has its own dictionary. Therefore, a partitioned table has a separate dictionary for each partition, which allows DB2 to adapt to changes in the data as you roll in a new partition.

From DB2 v10.1 onwards, a new type of compression called adaptive compression or deep compression is introduced. For more information, read the IBM whitepaper “Best Practices Storage Optimization with Deep Compression” available via the following link: https://www.ibm.com/developerworks/mydeveloperworks/wikis/home/wiki/Wc9a068d7f6a6_4434_aece_0d297ea80ab1/page/Storage%20optimization%20with%20deep%20compression?lang=en

DB2 settings

In most cases, Infor LN runs well with an out-of-the box DB2 database with standard parameters used by benchmarks and customers. To tune away some side effects or influence resource usage on high-end DB2 systems, you can modify parameters.

Database manager configuration tuning

For every connection established with the database server, the DB2 Engine starts an AGENT. To accommodate all active applications, you must appropriately set **MAXAGENTS**. Each session in the Infor LN application virtual machine establishes a new connection and. On average, the number of connections established by a user's application virtual machine is three. We recommend that you set the **MAXAGENTS** to three times the total number of users supported on the system.

Database configuration tuning

To estimate the performance of the DB2 environment, we recommend that you take a snapshot to determine the buffer pool activity, lock escalation information, sort activity, the package cache lookups, and package cache inserts information.

If lock escalation occurs, increase the value of **MAXLOCKS** and **LOCKLIST**. These two parameters are mutually dependent.

To ensure a well performing environment, the ratio of package cache lookups to package cache inserts must be 100:1. Modify the value of **PCKCACHESZ** to reach this ratio.

We recommend that you set the **NUM_IOCLEANERS** parameter value to the following:

```
num_iocleaners = 4 + number of containers that constitute the tablespaces
```

For example, if the table data tablespace consists of six containers on various disk drives, and another six containers are available for index data on various disk drives, you must set **NUM_IOCLEANERS** to the following:

```
num_iocleaners = 4 + 6 + 6 = 16
```

Because the page cleaners are activated asynchronously, the cleaners update the dirty pages on various disks, which make the pages free and available for the database data.

If you experience lock escalations on the catalog tables, you can change the **CATALOGCACHE_SZ** parameter to a higher value.

In a client/server environment, to give higher priority to the database agent processes, increase the value of the **AGENTPRI** parameter.

You must also increase your buffer pool size to an appropriate value for your system. You can increase the buffer pool size by using the ALTER BUFFERPOOL statement. See IBM's *DB2 Administration Guide.*

Call Level Interface configuration

The DB2 driver has a connection pooling mechanism in the call level interface (CLI), which reduces the number of database connections to one per user. This mechanism saves CPU and memory resources. To enable multiconnect, use the following command:

```
update cli cfg for section Common using MultiConnect 3
```

Note:

- The MultiConnect parameter is applicable for ALL operating systems.
- Default value: Not set.
- Possible values: <number of db connections>.
- After setting the registry variable value, restart the database.

DB2 environment parameters

The following table shows the DB2 parameters for which benchmarks showed that implementing these parameters will increase the performance. Some parameters are optional, others are required. All parameters are explained in more detail later in this chapter.

Parameter	Recommended Value	Importance
DB2_RESOURCE_POLICY		Optional (high-end only)
DB2_USE_FAST_PREALLOCATION	OFF	Recommended
DB2_MINIMIZE_LISTPREFETCH	YES	Recommended
DB2_SKIP_INSERTED	ON	Recommended
DB2_SKIPDELETED	ON	Recommended
DB2_EVALUNCOMMITTED	ON	Recommended
DB2_USE_ALTERNATE_PAGE_CLEANING	ON	Recommended
DB2_LOGGER_NON_BUFFERED_IO	ON	Recommended
DB2_AVOID_PREFETCH	ON	Recommended
DB2LOCK_TO_RB	STATEMENT	Required
DB2COMM	tcpip	Required
DB2_PARALLEL_IO	*	Recommended

Caution: Setting one or more of these parameters in the database may lead to side effects in other products or Infor LN sessions. Therefore, test before you implement.

Log reader and log writer priority using DB2_RESOURCE_POLICY

To optimize the speed of the commits, increase the CPU scheduler priority of the DB2 log reader and log writer processes by using the DB2_RESOURCE_POLICY parameter.

The DB2 log reader (db2loggr) reads the database log files during:

- Transaction processing (rollback)
- Restart recovery
- Roll forward operations

The database log writer flushes log records from the log buffer to the log files on disk.

To set the DB2_RESOURCE_POLICY parameter, create a file that has the following content:

```
<RESOURCE_POLICY>
                <SCHEDULING_POLICY>
                        <POLICY_TYPE>SCHED_FIFO</POLICY_TYPE>
                        <PRIORITY_VALUE>58</PRIORITY_VALUE>

                        <EDU_PRIORITY>
                                <EDU_NAME>db2loggr</EDU_NAME>
                                <PRIORITY_VALUE>56</PRIORITY_VALUE>
                        </EDU_PRIORITY>

                        <EDU_PRIORITY>
                                <EDU_NAME>db2loggw</EDU_NAME>
                                <PRIORITY_VALUE>56</PRIORITY_VALUE>
                        </EDU_PRIORITY>
                </SCHEDULING_POLICY>
</RESOURCE_POLICY>
```

To enable the resource policy, use this command:

```
db2set DB2_RESOURCE_POLICY=<path to file>
```

Note:

- The DB2_RESOURCE_POLICY parameter is applicable for UNIX operating systems.
- Default value: Not set.
- Possible values: <path to file>.
- After setting the registry variable value, restart the database.

DB2_USE_FAST_PREALLOCATION

As of DB2 Version 9.5 Fix Pack 6 and Version 9.1 Fix Pack 7, fast preallocation is the default mechanism for creating and extending table spaces on AIX JFS2. Fast preallocation enables fast allocation to reserve a tablespace and speed up the process of creating or altering large tablespaces and database restore operations. This faster processing is implemented with a small overhead cost for rows insertion. For AIX operating systems with large volumes of inserts and selects on the same tablespace, disabling fast preallocation provides improved runtime performance and decrease latch contention, at the cost of slower tablespace creation and restore times.

To disable fast preallocation in a running environment, back up the database and set the DB2_USE_FAST_PREALLOCATION variable to OFF. After fast preallocation is disabled, restore the database.

To prevent DB2 latch contention, we recommend that you create the DB2 database files by using the following setting:

```
db2set DB2_USE_FAST_PREALLOCATION=OFF
```

Note:

- The DB2_USE_FAST_PREALLOCATION parameter is applicable for UNIX operating systems.
- Default value: ON.
- Possible values: ON, OFF.
- After setting the registry variable value, restart the database.

DB2_MINIMIZE_LISTPREFETCH

List prefetch is a special table access method that involves retrieving the qualifying Row IDs from the index, sorting them by page number, and then prefetching the data pages. Sometimes the optimizer does not have enough accurate information to determine whether list prefetch is the correct access method. This problem may occur when predicate selectivity contains parameter markers or host variables that prevent the optimizer from using catalog statistics to determine the selectivity.

This registry variable prevents the optimizer from considering list prefetch in such situations.

Caution: A list-prefetch plan requires a sort, and the sort might lead to an overflow sort which can adversely affect performance. A value of YES ensures that the optimizer uses an execution plan other than a list-prefetch plan, so that sorts can be avoided.

To enable list prefetching, use the following command:

```
DB2_MINIMIZE_LISTPREFETCH=YES
```

Note:

- The DB2_MINIMIZE_LISTPREFETCH parameter is applicable for all operating systems.
- Default value: NO.
- Possible values: YES, NO.
- After setting the registry variable value, restart the database.

DB2_SKIPINSERTED

The DB2_SKIPINSERTED registry variable controls whether uncommitted data insertions can be ignored for statements that use the cursor stability (CS) or the read stability (RS) isolation level.

Uncommitted insertions are handled in one of the following two ways, depending on the value of the DB2_SKIPINSERTED registry variable:

- When the value is ON, the DB2 server ignores uncommitted insertions, which in many cases can improve concurrency and is the preferred behavior for most applications. Uncommitted insertions are treated as though they had not yet occurred.
- When the value is OFF (the default value), the DB2 server waits until the insert operation completes (commits or rolls back) and then processes the data accordingly. This action is appropriate in certain cases. For example:

- Suppose that two applications use a table to pass data between themselves, with the first application inserting data into the table and the second one reading it. The data must be processed by the second application in the order presented, such that if the next row to be read is being inserted by the first application, the second application must wait until the insert operation is committed.
- An application avoids UPDATE statements by deleting data and then inserting a new image of the data.

To enable DB2_SKIPINSERTED, use the following command:

```
db2set DB2_SKIPINSERTED=ON
```

Note:

- The DB2_SKIPINSERTED parameter is applicable for all operating systems.
- Default value: OFF.
- Possible values: OFF, ON.
- After setting the registry variable value, restart the database.

DB2_SKIPDELETED

If the DB2_SKIPDELETED variable is enabled, statements using either Cursor Stability or Read Stability isolation levels unconditionally skip deleted keys during index access and deleted rows during table access.

If the DB2_EVALUNCOMMITTED variable is enabled, deleted rows are automatically skipped, but uncommitted pseudo-deleted keys in type-2 indexes are skipped only when the DB2_SKIPDELETED variable is also enabled.

This registry variable does not affect the behavior of cursors on the DB2 catalog tables.

To enable DB2_SKIPDELETED, use the following command:

```
db2set DB2_SKIPDELETED=ON
```

Note:

- The DB2_SKIPDELETED parameter is applicable for all operating systems.
- Default value: OFF.
- Possible values: OFF, ON.
- After setting the registry variable value, restart the database.

DB2_EVALUNCOMMITTED

If the DB2_EVALUCOMMITTED variable is enabled, where possible, scans defer or avoid row locking until the data is known to satisfy predicate evaluation. If this variable is enabled, predicate evaluation might occur on uncommitted data.

DB2_EVALUNCOMMITTED is applicable only to statements using either Cursor Stability or Read Stability isolation levels. For index scans, the index must be a type-2 index. Deleted rows are

skipped unconditionally on table scan access. Deleted keys are not skipped for type-2 index scans unless the registry variable DB2_SKIPDELETED is also set.

The decision as to whether deferred locking is applicable is made at statement compile or bind time.

To enable DB2_EVALUNCOMMITTED, use the following command:

```
db2set DB2_EVALUNCOMMITTED=ON
```

Note:

- The DB2_EVALUNCOMMITTED parameter is applicable for all operating systems.
- Default value: NO.
- Possible values: YES, NO.
- After setting the registry variable value, restart the database.

DB2_USE_ALTERNATE_PAGE_CLEANING

This variable specifies whether a DB2 database uses the alternate method of page cleaning algorithms or the default method of page cleaning. If this variable is set to ON, the DB2 system writes changed pages to disk, keeping ahead of Log Sequence Number (LSN) gaps and proactively finding victims. Therefore, the page cleaners utilize available disk I/O bandwidth. If this variable is set to ON, the chngpgs_thresh database configuration parameter is no longer relevant because it does not control page cleaner activity. This proactive method of page cleaning improves performance for Infor LN.

To enable DB2_USE_ALTERNATE_PAGE_CLEANING, use the following command:

```
db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON
```

Note:

- The DB2_USE_ALTERNATE_PAGE_CLEANING parameter is applicable for all operating systems.
- Default value: OFF.
- Possible values: OFF, ON.
- After setting the registry variable value, restart the database.

DB2_LOGGER_NON_BUFFERED_IO

By using this variable you can control whether direct I/O (DIO) is used on the log file system. If DB2_LOGGER_NON_BUFFERED_IO is set to AUTOMATIC, active log windows (namely, the primary log files) will open with DIO, and all other logger files will be buffered. If this parameter is set to ON, all log file handles will open with DIO. If this parameter is set to OFF, all log files handles will be buffered.

To enable DB2_LOGGER_NON_BUFFERED_IO, use the following command:

```
db2set DB2_LOGGER_NON_BUFFERED_IO=ON
```

Note:

- The DB2_LOGGER_NON_BUFFERED_IO parameter is applicable for all operating systems.
- Default value: AUTOMATIC.
- Possible values: AUTOMATIC, OFF, ON.
- After setting the registry variable value, restart the database.

DB2_AVOID_PREFETCH

The DB2_AVOID_PREFETCH variable specifies whether prefetch should be used during crash recovery. If DB2_AVOID_PREFETCH=ON, prefetch is not used.

To enable DB2_AVOID_PREFETCH, use the following command:

```
db2set DB2_AVOID_PREFETCH=ON
```

Note:

- The DB2_AVOID_PREFETCH parameter is applicable for all operating systems.
- Default value: OFF.
- Possible values: OFF, ON.
- After setting the registry variable value, restart the database.

DB2LOCK_TO_RB

The DB2LOCK_TO_RB variable specifies whether lock timeouts will cause the entire transaction to be rolled back, or only the current statement. If DB2LOCK_TO_RB is set to STATEMENT, lock timeouts cause only the current statement to be rolled back. When any other value is specified for this variable, the whole transaction will be rolled back.

To enable DB2LOCK_TO_RB, use the following command:

```
db2set DB2LOCK_TO_RB=STATEMENT
```

Note:

- The DB2LOCK_TO_RB parameter is applicable for all operating systems.
- Default value: NULL.
- Possible values: STATEMENT.
- After setting the registry variable value, restart the database.

DB2COMM=tcpip

By using the DB2COMM registry variable you can set communication protocols for the current DB2 instance. If the DB2COMM registry variable is undefined or set to null, no protocol connection managers are started when the database manager is started.

To set the communication protocol for the DB2 instance, use the following statement:

```
db2set DB2COMM=tcpip
```

Note:

- The DB2COMM parameter is applicable for all operating systems.
- Default value: NULL.
- Possible values: tcpip, ssl.
- After setting the registry variable value, restart the database.

DB2_PARALLEL_IO

The DB2_PARALLEL_IO registry variable is used to change the way DB2 calculates the I/O parallelism of a table space. When I/O parallelism is enabled (either implicitly, by the use of multiple containers, or explicitly, by setting DB2_PARALLEL_IO), the parallelism is achieved by issuing the correct number of prefetch requests. Each prefetch request is a request for a set of pages. For example, a table space has two containers and the prefetch size is four times the extent size. If the registry variable is set, a prefetch request for this table space will be broken into four requests (one extent per request) and there is a possibility that four prefetchers will service the requests in parallel.

You may want to set the registry variable if the individual containers in the table space are striped across multiple physical disks or if the container in a table space is created on a single RAID device that is composed of more than one physical disk.

If this registry variable is not set, the degree of parallelism of any table space is the number of containers of the table space. For example, if DB2_PARALLEL_IO is set to NULL and a table space has four containers, four extent-sized prefetch requests are issued; or, if a tablespace has two containers and the prefetch size is four times the extent size, the prefetch request for this table space will be broken into two requests (each request will be for two extents).

If this registry variable is set, and the prefetch size of the table is not AUTOMATIC, the degree of parallelism of the table space is the prefetch size divided by the extent size. For example, if DB2_PARALLEL_IO is set for a table space that has a prefetch size of 160 and an extent size of 32 pages, five extent-sized prefetch requests are issued.

If this registry variable is set, and the prefetch size of the table space is AUTOMATIC, DB2 automatically calculates the prefetch size of a table space.

To enable parallel I/O for all tablespaces with six disks (default) per container, enter the following command:

```
db2set DB2_PARALLEL_IO=*
```

Note:

- The DB2_PARALLEL_IO parameter is applicable for all operating systems.
- Default value: NULL.
- Possible values: TablespaceID:[n],… – a comma-separated list of defined table spaces (identified by their numeric table space ID). If the prefetch size of a table space is AUTOMATIC,

you can indicate to the DB2 database manager the number of disks per container for that table space by specifying the table space ID, followed by a colon, followed by the number of disks per container, n. If n is not specified, the default is 6.

You can replace TablespaceID with an asterisk (*) to specify all table spaces. For example, if DB2_PARALLEL_IO=*, all table spaces use 6 as the number of disks per container. If you specify both an asterisk (*) and a table space ID, the table space ID setting takes precedence. For example, if DB2_PARALLEL_IO =*,1:3, all table spaces use 6 as the number of disks per container, except for table space 1, which uses 3.

- After setting the registry variable value, restart the database.

Chapter 2 Tuning Infor LN for DB2

2

This chapter describes the most important Infor LN performance settings and parameters when running on a DB2 database.

DB2 db_resource parameters

From porting set 8.6a onwards, the default DB2 db_resource values are optimal for most customers. Therefore, for OLTP usage in 2-tier, no db_resource file is required. The following parameters are important for performance, and in certain circumstances it may be necessary to change them. For 2-tier OLTP and batches (both 2-tier and 3-tier), some db_resource parameters will increase the performance. See the *Infor Enterprise Server Technical Reference Guide for DB2 Database Driver (U7829 US).*

db2_opt_level

The db2_opt_level resource sets the query optimization level for SQL queries. The default and recommended value is 0, which means that index scans are used. For possible query optimization class values and their meaning, refer to DB2 documentation.

db2_opt_rows

You can use the db2_opt_rows resource to specify to DB2 that no more than the specified number of rows will be retrieved in one request. Therefore, DB2 optimizes the fetch request. It is assumed that the number of rows retrieved will not exceed the number of rows. Based on this value, DB2 determines a suitable communication buffer size to improve performance. The default and recommended value is 5.

db2_max_open_handles

The db2_max_open_handles resource limits the number of open cursors that the driver maintains on a per connection basis. Each cursor represents one type of SQL statement. A maximum of 250, and

a minimum of one open statement handles are allowed per connection. The default and recommended value is 100.

db2_retained_cursors

The db2_retained_cursors resource sets the number of inactive cursors that must be retained in the list for reuse. After all rows have been fetched, the driver has a facility to put inactive cursors in cancel state in a cancel list, so that they become candidates for being assigned to a different query. Several inactive cursors in this list are not available for this, and are defined by the db2_retained_cursors resource, which defaults to 20. If more than 20 cursors are in the cancel list, and a request for a new cursor is issued, the least recently inactivated cursor is used for this new cursor. This cursor is disassociated from the original query and assigned to a new query, which performs parsing and binding on this cursor. When the original query is doing a re-execute, the driver detects the cursor has been associated with another query and will get a new cursor and re-parse and bind the query again. Increasing the value of db2_retained_cursors leads to less re-parsing and rebinding of queries, which reduces CPU resources. However, the number of open cursors and memory is increased.

Increasing retained cursors can help to increase performance, but only a small effect has been seen for batch sessions. Therefore, this variable can be left to the default value for OLTP usage. An example for a batch setting is as follows:

```
db2_retained_cursors:300
```

Array interface

The Infor LN DB2 database driver can use the DB2 array interface for array fetches and array inserts. With the array interface, communication between the Infor LN DB2 driver and DB2 is more efficient as multiple rows are fetched or inserted simultaneously. However, because multiple rows must be stored in a buffer in the Infor LN database driver, more memory is consumed. Array interfacing is useful if you access a remote database (3-tier setup), because this helps to reduce the number of network round-trips.

To adjust the size of the buffers that hold array rows, you must set the **db2_max_array_size** resource parameter. You can set the array buffer size on a per-table basis using the **ARR_SIZE** storage parameter in the db2_storage_param file.

You can enable/disable the array fetch interface with the resource variable **db2_array_fetch**. You can enable the array insert interface with the resource variable **db2_array_insert**. By default, array inserts are disabled, but the array inserts are enabled when data is placed in the database tables using the bdbpost utility with the –f option.

In case of a 3-tier setup, increasing the array size will increase the performance. We recommend the following DB2 database driver settings for 3-tier OLTP:

```
db2_opt_rows:10
db2_max_array_size:10
db2_array_insert:1
```

Batches

For batches, 2-tier and 3-tier, a separate db_resource file can be used to increase the performance of batches, without affecting the OLTP performance. This can be done via the environment variable USR_DBS_RES to be set only for the user who executes the batches. We recommend the following DB2 database driver settings for batches:

```
db2_retained_cursors:300
db2_max_open_handles:500
db2_opt_rows:10
db2_max_array_size:10
db2_array_insert:1
```

Using these parameters will increase the memory usage of the bshell (incl database driver), but will save CPU cycles, in order to execute the batch faster.

Database storage and driver files

The storage parameter file provides a way to specify the distribution of table and index data in various segments. The database driver uses storage parameters whenever you execute a DDL statement, such as a create table or create index statement. Only performance topics on this are discussed. For more information about these files, consult the database driver manuals.

DB2 storage file

Usually, the storage parameters indicate the tablespace used for storage. Because Locally Managed tablespaces are common, nothing more than specifying a tablespace is used for this file:

```
*:*:*:T_SPACE dataspace I_SPACE indexspace
```

From a performance point of view, it does not matter if tables and indexes are stored in the same or different tablespaces. The disk layout is more important: is the I/O evenly spread across the disks, and do we have no I/O bottleneck?

DB2 driver parameter file

We recommend the following settings in the db2_driver_param:

```
*:*:*::400::
```

This will use the iterative method of table/index optimization, which will perform optimal for most customers. However, for bad performing queries, it is possible to use the nested (200) or filtered (1000) optimization technique. The file will be read from top to bottom, so you have to specify the specific tables first in the db2_driver_param. The following example will use the nested optimization technique for table ttaad100 in company 090 and will use iterative for all other tables and indexes:

```
ttaad100:090:T::200::
*:*:*::400::
```

Chapter 3 Tracing DB2

3

DB2 provides a comprehensive tool, called Explain, with detailed optimizer information on the access plan chosen for an explained SQL statement. Several methods are provided to capture and access Explain information. For more tracing facilities of Infor LN on DB2, see the I*nfor LN Performance, Tracing and Tuning Guide (U9357 US)*.

DB2 Explain facility

Detailed optimizer information, which allows for in-depth analysis of an access plan, is kept in Explain tables separate from the actual access plan itself. There are three ways to get information from the Explain tables:

- Write your own queries based on the Explain table descriptions as shown in the DB2 documentation.
- Use the db2exfmt tool (Explain Tables formatting tool).
- Use Visual Explain to view Explain information in graphic format. To enable and collect Explain plan information, complete the following steps:

1 Create the Explain tables for the user that will be running the slow SQL/Infor LN session. To do this, connect to the database using that user name and implement the EXPLAIN.DDL script located at the following location: ${INSTHOME}/sqllib/misc/EXPLAIN.DDL

```
db2 connect to inforln user <user name>
db2 –tvf ${INSTHOME}/sqllib/misc/EXPLAIN.DDL
```

2 Activate the Explain. If the offending SQL has not yet been identified, DB2EXPLAIN=3 must be added in the COMMON section of the db2cli.ini file.
To activate the Explain, run the following command:

```
db2 update cli cfg for section common using DB2EXPLAIN 3
```

Care must be taken because every user connecting to Infor LN will read this file and expect that their Explain tables exist. It is recommended that one user collect Explain data at a time. The Explain data from all the SQL performed by the session will be stored in the Explain tables.

3 Alternatively, if the offending SQL statements have already been identified, you can add the clause “explain plan for …<sql statement>” and implement it directly from the command prompt or DB2 CLP prompt. An example of explaining a single SQL statement is shown in the following figure:

```
explain plan for
SELECT a.t_acti,a.t_ardt,a.t_astk,a.t_blck, a.t_bpid,a.t_btsp,a.t_cdck,
```

```
        a.t_clot,a.t_conf,a.t_cwar,a.t_fire,a.t_idat,a.t_item,a.t_itxt,
        a.t_loca,a.t_lsel,a.t_lsta,a.t_oorg,a.t_orno,a.t_oset,a.t_pkdf,
        a.t_pmsk,a.t_pono,a.t_prdt,.t_psno,a.t_psnr,a.t_psqu,a.t_qadv,
        a.t_qput,a.t_qrel,a.t_qstk,a.t_qstr,a.t_rcno,a.t_rcun,.t_revi,
        a.t_rstk,a.t_seqn,a.t_shid,a.t_stka,a.t_stkr,a.t_stun,a.t_txtn,
        c.t_cuni,b.t_item
FROM ( ( inforln.twhinh210570 AS a LEFT JOIN inforln.twhwmd400570 AS b
ON b.t_item = a.t_item) LEFT JOIN inforln.ttcibd001570 AS c ON c.t_item = b.t_item )
WHERE a.t_blck = ? AND (MOD(a.t_seqn,2)) = ? AND a.t_fire = ? AND (b.t_item = a.t_item OR
b.t_item IS NULL OR a.t_item IS NULL) AND (c.t_item = b.t_item OR c.t_item IS NULL OR
b.t_item IS NULL) AND (a.t_oorg > ?)
ORDER BY 18,19,23,37 ;
```

Figure 3-1: Example of explaining a single SQL statement

Formatting/viewing the content of explain tables

Two of the widely used tools to view the content of the Explain tables are the db2exfmt command and the GUI Visual Explain.

db2exfmt

The db2exfmt tool is located in the misc subdirectory of the instance sqllib directory. To use the tool, you require read access to the Explain tables being formatted. The tool is executed from the command line and will prompt for any parameter value not supplied. Online help/usage is available and it is documented in the DB2 UDB Administration Guide.

The following is the formatted output from the execution of the db2exfmt command after explaining the above sample query. Detailed statistics of the objects (tables and indexes) used in the access plan are always given at the end of the db2exfmt output.

Visual Explain

Visual Explain allows for the analysis of access plan and optimizer information from the explain tables through a graphical interface. Static and dynamic SQL statements can be analyzed using this tool. Visual Explain is typically invoked from within the Control Center; the Control Center is available from the command line by typing **db2cc**. Also, Visual Explain can be invoked directly from the command line for a single SQL statement using the **db2vexp** command.

Each operation and object in the graph can be double-clicked to display detailed data, as shown in Figure 3-2.

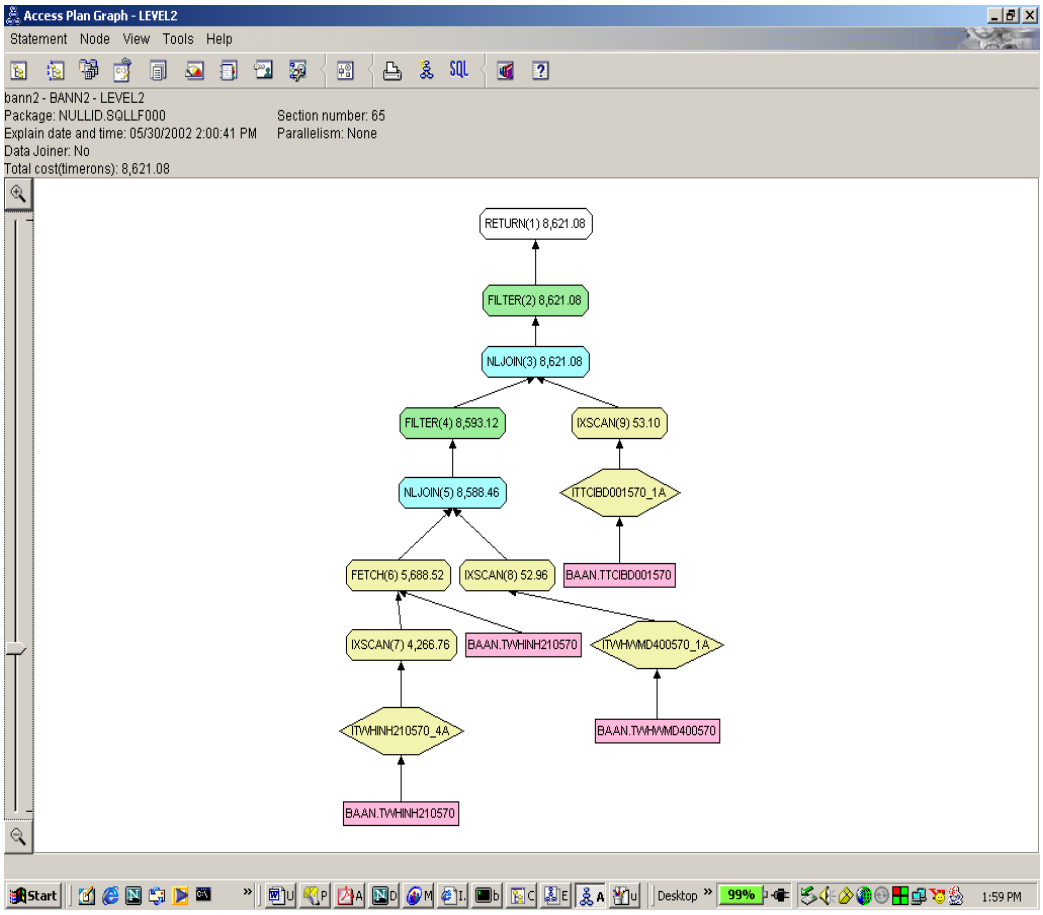

Figure 3-2: db2vexp Access Plan Graph

Dynamic SQL snapshot

Although the information provided by the Dynamic SQL snapshot tool does not contain a great deal of detail, it does provide an easy and effective way of detecting and isolating offending queries: it often indicates the reason for the performance problem.

Because of its ease of use, this must be one of the first tools used in the tuning process.

To get a Dynamic SQL snapshot, complete the following steps:

1 Connect to the database as a user with privileges, as shown below:

```
db2 connect to inforln
```

2 Enable the database manager switch for statements, as shown below:

```
db2 update monitor switches using statement ON
```

3 Execute the problematic Infor LN session(s).

4 Get a snapshot for dynamic SQL redirecting the output to a file, as shown below:

```
db2 get snapshot for dynamic sql on inforln > dynamic_snap.out
 Dynamic SQL Snapshot Result

 Data base name                       = INFORLN
 Data base path                       = /app/db2inst1/NODE0000/SQL00001/
 Number of executions                = 1
 Number of compilations              = 1
 Worst preparation time (ms)         = 44
 Best preparation time (ms)          = 44
 Internal rows deleted               = 0
 Internal rows inserted              = 0
 Rows read                           = 4
 Internal rows updated               = 0
 Rows written                        = 0
 Statement sorts                     = 0
 Total execution time (sec.ms)       = 0.289055
 Total user cpu time (sec.ms)        = 0.020000
 Total system cpu time (sec.ms)      = 0.000000
 Statement text                      = SELECT a.t_acti,a.t_ardt,a.t_astk,a.t_blck,
a.t_bpid,a.t_btsp,a.t_cdck, a.t_clot,a.t_conf,a.t_cwar,a.t_fire,a.t_idat,a.t_item,a.t_itxt,
a.t_loca,a.t_lsel,a.t_lsta,a.t_oorg,a.t_orno,a.t_oset,a.t_pkdf, a.t_pmsk,a.t_pono,a.t_prdt,
a.t_psno,a.t_psnr,a.t_psqu,a.t_qadv, a.t_qput,a.t_qrel,a.t_qstk,a.t_qstr,a.t_rcno,a.t_rcun,
a.t_revi, a.t_rstk,a.t_seqn,a.t_shid,a.t_stka,a.t_stkr,a.t_stun,a.t_txtn, c.t_cuni,b.t_item
FROM ( ( inforln.twhinh210570 AS a LEFT JOIN inforln.twhwmd400570 AS b ON b.t_item =
a.t_item) LEFT JOIN inforln.ttcibd001570 AS c ON c.t_item = b.t_item ) WHERE a.t_blck = 1 AND
(MOD(a.t_seqn,2)) = 2 AND a.t_fire = 3 AND (b.t_item = a.t_item OR b.t_item IS NULL OR
a.t_item IS NULL) AND (c.t_item = b.t_item OR c.t_item IS NULL OR b.t_item IS NULL) AND
(a.t_oorg > 4) ORDER BY 18,19,23,37

 Number of executions                = 1
 Number of compilations              = 1
 Worst preparation time (ms)         = 767
 Best preparation time (ms)          = 767
 Internal rows deleted               = 0
 Internal rows inserted              = 0
 Rows read                           = 42
 Internal rows updated               = 0
 Rows written                        = 74
 Statement sorts                     = 0
 Total execution time (sec.ms)       = 0.803617
 Total user cpu time (sec.ms)        = 0.090000
 Total system cpu time (sec.ms)      = 0.050000
 Statement text                      = explain plan for SELECT
a.t_acti,a.t_ardt,a.t_astk,a.t_blck, a.t_bpid,a.t_btsp,a.t_cdck,
a.t_clot,a.t_conf,a.t_cwar,a.t_fire,a.t_idat,a.t_item,a.t_itxt,
a.t_loca,a.t_lsel,a.t_lsta,a.t_oorg,a.t_orno,a.t_oset,a.t_pkdf, a.t_pmsk,a.t_pono,a.t_prdt,
a.t_psno,a.t_psnr,a.t_psqu,a.t_qadv, a.t_qput,a.t_qrel,a.t_qstk,a.t_qstr,a.t_rcno,a.t_rcun,
a.t_revi, a.t_rstk,a.t_seqn,a.t_shid,a.t_stka,a.t_stkr,a.t_stun,a.t_txtn, c.t_cuni,b.t_item
FROM ( ( inforln.twhinh210570 AS a LEFT JOIN inforln.twhwmd400570 AS b ON b.t_item =
a.t_item) LEFT JOIN inforln.ttcibd001570 AS c ON c.t_item = b.t_item ) WHERE a.t_blck = ? AND
(MOD(a.t_seqn,2)) = ? AND a.t_fire = ? AND (b.t_item = a.t_item OR b.t_item IS NULL OR
a.t_item IS NULL) AND (c.t_item = b.t_item OR c.t_item IS NULL OR b.t_item IS NULL) AND
(a.t_oorg > ?) ORDER BY 18,19,23,37
```

Figure 4-3: Sample output Dynamic SQL snapshot

To interpret the output

Interpreting the output is the most complex part in the tuning exercise. There are some general guidelines that must apply to most cases. When these guidelines are followed correctly, they can eliminate a great deal of complexity from the analysis process.

DB2 Explain

The following performance areas can be investigated with Visual Explain and db2exfmt:

- Look for operations that must be avoided, such as SORT and TBSCAN. Indexes must always be used to retrieve data.
- Check whether the correct indexes have been selected to retrieve the data.
- Verify that the database objects used (table and indexes) have updated statistics.

Dynamic SQL snapshot

With Dynamic SQL snapshot, the following performance areas can be investigated:

- Statements that consume large amounts of execution time can be problematic.
- Isolate statements that perform large number of sorts. Sorts must be minimized.
- Statements that perform selects and have rows written > 0 and perform sorts are likely to have sort overflows (sorts on disk).
- Statements reading excessive number of rows are probably doing sequential table scans and must be isolated.
- Ideally, for every statement, the number of executions > number of compilations. If the number of compilations is > 1, check the configuration parameter Package Cache.

IBM InfoSphere Optim Performance Manager

From DB2 version 9.7 onwards, the IBM InfoSphere Optim Performance Manager can be used to monitor the database performance and identify performance bottlenecks. More information can be found on http://www.ibm.com/software/data/optim/optimize-performance