Infor LN Performance, Tracing, and Tuning Guide for DB2

Publication Information

Release: Infor LN 10.x

Publication date: November 25, 2014

Document code: B0077C

Contents

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 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 LN - Data compression (B0050 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.

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 on the database performance. This chapter describes the important tuning areas of DB2.

Introduction

Every new version of a database includes new settings, tools, and so on. Although Infor LN runs efficiently with an out-of-the box DB2 database, you must perform tuning. DB2 is a self-learning and tuning database, but you must perform tuning for optimal performance. For an optimal DB2 database, you must have DB2 database knowledge.

IBM and partners provide courses on generic DB2 tuning; the Internet also provides information. See DB2 documentation at http://www.ibm.com/support/publications/us/library/#letter_D.

Creating statistics

You create statistics in DB2 to ensure that the optimizer can select the best possible plan. Therefore, you must create a script, which saves the names of all the existing tables in a file and execute the runstats on this file. As data is modified, it is recommend that run a script during the weekly maintenance window:

```
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>’
```

Data Compression

Data can grow faster than expected; especially when is history stored for extended periods or if the financial logging option is used. From porting set 9.0 onwards, Infor LN supports advanced table and index compression. The benefits of data compression:

- Significant savings in disk storage space.

- Fewer page reads, because more rows can fit on a page.
- Reduced disk I/O activity, because more compressed rows than uncompressed rows are included in a page.
- Ability to compress older data that is not accessed often, such as archived tables. The frequently accessed data is stored in the uncompressed form.
- Possibility to free space no longer required for a table.
- Faster backup and restore capabilities.

See Infor LN Data compression (B0050 US).

DB2 settings

The Infor LN application performs efficiently with an out-of-the box DB2 database with standard parameters, used by benchmarks and customers. However, you can modify parameters for tuning performance related issues on DB2 systems.

Database manager configuration tuning

For every connection established with the database server, the DB2 Engine creates an AGENT. To accommodate all active applications, you must set **MAXAGENTS** appropriately. Each session in the Infor LN application virtual machine establishes a new connection. On average, the number of connections established by a user's application virtual machine is three. It is recommended 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, It is recommend you take a snapshot of the data to determine the buffer pool activity, lock escalation, sort activity, the package cache lookups, and package cache inserts information.

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

To create an environment that helps optimize perfomance, the ratio of package cache lookups to package cache inserts must be 100:1. Modify the value of **PCKCACHESZ** to reach this ratio.

It is recommended 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.

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

In a client/server environment, to assign a 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 includes 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, based on which benchmarking, that increase the performance of the application. Some parameters are optional, others are required. All parameters are explained in detail.

Parameter	Recommended Value	Importance
DB2_RESOURCE_POLICY		Optional (high-end only)
DB2_USE_FAST_PREALLOCATION	OFF	Recommended
DB2_MINIMIZE_LISTPREFETCH	YES	Recommended

Parameter	Recommended Value	Importance
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 impact other products or Infor LN sessions. Therefore, you must carry out tests before you implement these parameters.

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 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 with 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

With the 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 a faster allocation process to reserve a tablespace and also speed up the process of creating or altering large tablespaces and database restore operations. This 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 the fast preallocation provides improved runtime performance and a decrease in latch contention. However, the tablespace creation and restore times will increase.

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, it is recommended 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 the rows by page number, and then prefetching the data pages. Sometimes, the optimizer does not receive accurate information to determine whether the list prefetch is the correct access method. This problem can occur when the predicate selectivity contains parameter markers or host variables, which prevent the optimizer from using catalog statistics to determine the selectivity.

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

To minimize 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.
- Changes to this variable immediately effect all future compiled SQL statements, if the db2set command is issued with the -immediate parameter. You are not required to restart the instance.

DB2_SKIPINSERTED

The DB2_SKIPINSERTED registry variable is used to check if uncommitted data insertions can be ignored for statements that use the cursor stability (CS) or the read stability (RS) isolation level.

Uncommitted insertions are processed in 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. When the value is OFF (the default value), the DB2 server waits until the insert operation is completed (commits or rolled back) and then processes the data accordingly. This action is appropriate in cases such as:
    - Two applications use a table to pass data, with the first application inserting data into the table and the data is read by the second application. The data must be processed by the second application in the order in which the data is sent, for example, if the next row to be read is 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 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 the Cursor Stability or Read Stability isolation levels; unconditionally skip deleted keys, during the index access, and deleted rows during the 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 identified to meet the requirements of the predicate evaluation. If this variable is enabled, predicate evaluation can 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 to check if deferred locking is applicable is made during the 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 if 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 the disk, to ensure there are no Log Sequence Number (LSN) gaps and proactively identifies the 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 the variable does not control page cleaner activity. This proactive method of page cleaning improves the performance of the Infor LN application.

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

Use this variable to control the usage of direct I/O (DIO) on the log file system. If DB2_LOGGER_NON_BUFFERED_IO is set to AUTOMATIC, active log windows (the primary log files) are opened using DIO, and all other logger files are buffered. If this parameter is set to ON, all log file handles are opened using DIO. If this parameter is set to OFF, all log files handles are 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 if prefetch must 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 is used to specify if lock timeouts must roll back the entire transaction, or just the current statement must be rolled back. If DB2LOCK_TO_RB is set to STATEMENT, lock timeouts only roll back the current statement. When any other value is specified for this variable, the entire transaction is 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

You can use the DB2COMM registry variable to set the communication protocols for the current DB2 instance. If the DB2COMM registry variable is undefined or set to null, 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 modify the method used by the DB2 to calculate the I/O parallelism of a table space. When I/O parallelism is enabled (by the use of multiple

containers, or 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 is broken into four requests (one extent per request) and there is a possibility that four prefetchers are used to service the requests, in parallel.

It is recommended 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 (number of operations that must be simultaneously executed) 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 is broken into two requests (each request is applicable 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, specify 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 the numeric table space ID). If the prefetch size of a table space is AUTOMATIC, you can specify 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=*, the value for all table spaces is 6, that is the number of disks per container. If you specify both an asterisk (*) and a table space ID, the table space ID setting precedes other requests. 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 important Infor LN performance settings and parameters when running on a DB2 database.

DB2 db_resource parameters

The default DB2 db_resource values are used to provide a balance between the performance and memory usage, which are optimal for customers. Therefore, for the OLTP usage in a 2-tier environment, no db_resource file is required. The following parameters are important for performance, and in certain circumstances it may be necessary to change the same. For 3-tier OLTP and batches (both 2-tier and 3-tier), some db_resource parameters 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 is used to set the query optimization level for SQL queries. The recommended value is 5, which is designed to use the important query transformations and other query optimization techniques efficiently. For possible query optimization class values and the meaning, see the DB2 documentation.

db2_opt_rows

You can use the db2_opt_rows resource to indicate to the DB2 engine, that only the specified number of rows are retrieved for a single request. Therefore, DB2 optimizes the fetch request. It is assumed that the number of rows retrieved do not exceed the number of rows. Based on this value, DB2 can determine a suitable communication buffer size to improve performance. It is recommended to set the value of this parameter equal to the db2_max_array_size.

db2_max_open_handles

The db2_max_open_handles resource limits the number of open cursors that the driver maintains for each connection. The cursor represents a type of SQL statement. A maximum of 250, and a

minimum of 1 open statement handles are allowed per connection. The default value is optimal for OLTP usage. An example for a batch setting:

```
db2_max_open_handles:200
```

db2_retained_cursors

The db2_retained_cursors resource is used to specify the number of inactive cursors that must be retained in the list for reuse. After the rows are fetched, the driver changes the status of the inactive cursors to Cancel in a cancel list, so that the cursors can be assigned to a different query. Several inactive cursors cannot be assigned to a different query because the default value for the number of inactive cursors is set to 20. If more than 20 cursors are in the cancel list, and a request for a new cursor is issued, the cursor that is inactive for the longest duration is used as the new cursor.

This cursor is disassociated from the original query and assigned to a new query, which performs parsing and binding. When the original query is executed again, the driver detects the earlier link of the cursor with another query and assigns a new cursor, re-parses and binds the query again. Increasing the value of db2_retained_cursors leads to a reduced number of re-parsing and rebinding queries, which reduces usage of the CPU resources. However, the number of open cursors and memory is increased.

Increasing the retained cursors can help to increase performance, but there is only a minimal effect on the for batch sessions. Therefore, the default value for OLTP usage can be retained. An example for a batch setting:

```
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, additional memory is consumed. Array interfacing is especially 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 contain the array rows, you must set the **db2_max_array_size** resource parameter.

You can enable or disable the array fetch interface using the resource variable **db2_array_fetch**. You can enable the array insert interface using the resource variable **db2_array_insert**. Array inserts are always enabled when the data is stored in the database tables, using the bdbpost utility with the –f option.

In case of a 3-tier setup, increasing the array size increases the overall performance. The following DB2 database driver settings for 3-tier OLTP are recommended:

```
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 using the USR_DBS_RES environment variable that must be set only for the user who executes the batches. The following DB2 database driver settings for batches are recommended:

```
db2_retained_cursors:300
db2_max_open_handles:200
db2_opt_rows:10
db2_max_array_size:10
db2_array_insert:1
```

Using these parameters increases the memory usage of the bshell (incl database driver), but saves the CPU cycles, to execute the batch faster.

Chapter 3 Tracing DB2

3

DB2 provides a comprehensive tool, called Explain, with detailed optimizer information on the access plan selected for an explained SQL statement. Several methods are provided to capture and access Explain information. For 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 an in-depth analysis of an access plan, is stored in the Explain tables, which is separated from the actual access plan. There are three methods to retrieve information from the Explain tables:

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

1. Create the Explain tables for the user that executes the slow SQL query/Infor LN session. Connect to the database using the required user name and implement the EXPLAIN.DDL script from the following location: ${INSTHOME}/sqllib/misc/EXPLAIN.DDL

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

2. Activate Explain. If the offending SQL is not yet 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
```

   Note: Every user connecting to Infor LN reads this file and assumes their Explain tables exist. It is recommended that Explain data must be collected by only one user at a time. The Explain data from all the SQL performed by the session is stored in the Explain tables.

3. If the offending SQL statements is identified, you can add the “explain plan for …<sql statement>” clause and implement the same directly from the command prompt or DB2 CLP prompt. An example of a single SQL statement:

```
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

To view the content of the Explain tables you can use the db2exfmt command and the Visual Explain GUI tool.

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 to be formatted. The tool is executed from the command line and a prompt is displayed for any parameter value that is not supplied. Online help/usage is available and is documented in the DB2 UDB Administration Guide.

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

Visual Explain

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

You can double-click each operation and object in the graph to display detailed data, as shown in Figure 3-2.

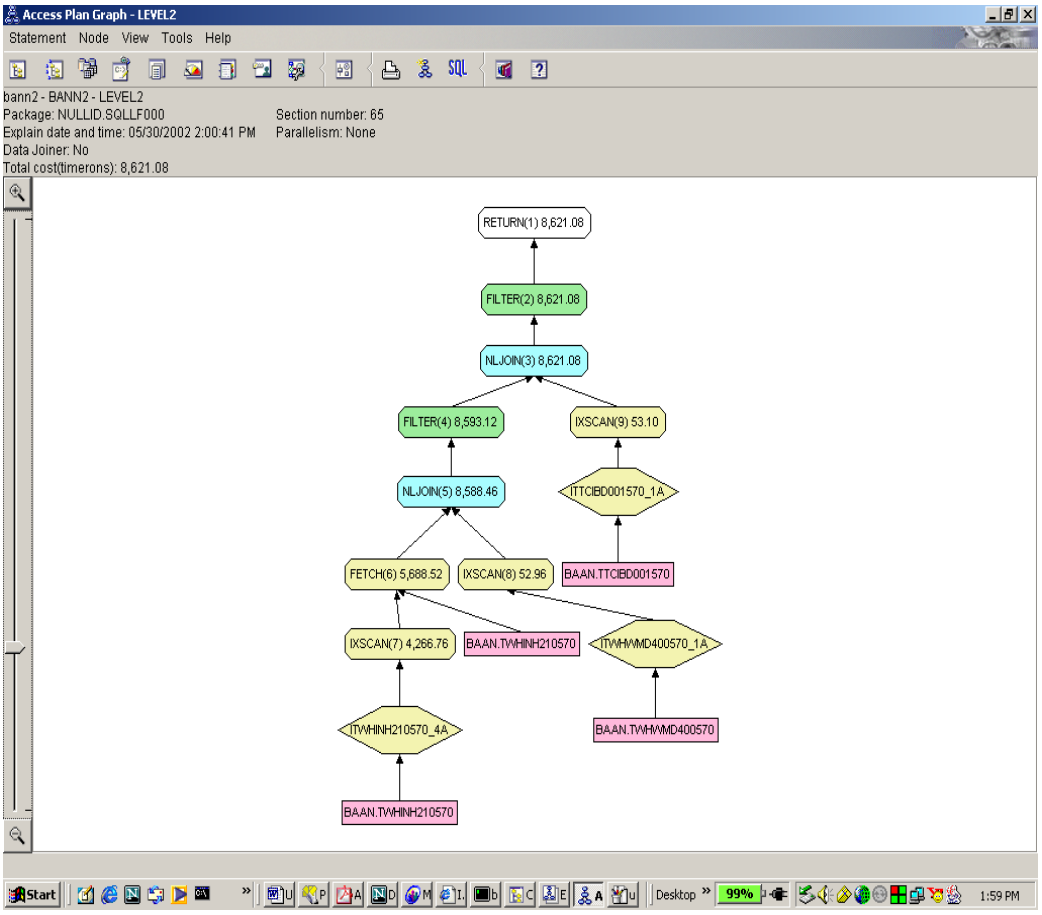

Figure 3-2: db2vexp Access Plan Graph

Dynamic SQL snapshot

The information provided by the Dynamic SQL snapshot tool does not contain detailed information, but the tool provides an easy and effective way of detecting and isolating offending queries and also indicates the reason for the performance problem.

It is recommended that you use the Dynamic SQL snapshot tool early in the tuning process.

To get a Dynamic SQL snapshot:

1 Connect to the database as a user with privileges.

```
db2 connect to inforln
```

2 Enable the database manager switch for statements.

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

3 Execute the Infor LN session(s) that are causing the issues.

4 Get a snapshot for dynamic SQL redirecting the output to a file:

```
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 complex part in the tuning exercise. There are general guidelines that are applicable. When these guidelines are followed, you eliminate a great deal of the complexity from the analysis process.

DB2 Explain

The following performance areas can be investigated using 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 are selected to retrieve the data.
- Verify that the database objects used (table and indexes) contain 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 cause issues.
- Isolate statements that perform large number of sorts. Sorts must be minimized.
- Statements that perform selects and have rows written > 0 and perform sorts include sort overflows (sorts on disk).
- Statements reading excessive number of rows are probably executing 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 Package Cache configuration parameter.

IBM InfoSphere Optim Performance Manager

The IBM InfoSphere Optim Performance Manager can be used to monitor the database performance and identify performance bottlenecks. See www.ibm.com/software/products/en/performance-manager