IBM Business Process Manager database troubleshooting, Part 1: Improve IBM BPM performance with an IBM DB2 database

Solve problems with IBM DB2 for Linux, UNIX, and Windows

Stephan O. Volz
Bonnie K. Brummond

September 23, 2015

IBM Business Process Manager (BPM) typically interacts and communicates with other software products. Therefore, if you understand cross-product troubleshooting tools you can quickly understand and solve problems. This tutorial shows what you can learn from the content of the BPMDB database in IBM BPM to troubleshoot performance problems and database-related issues.

View more content in this series

IBM® Business Process Manager (BPM) is a platform for handling all kinds of enterprise business tasks. With proper planning, it is possible to prevent issues that affect performance before they are reported by process participants and other end users. This series focuses on what you can learn from the BPMDB database in IBM BPM to prevent problems and troubleshoot issues. The first two parts of this series focus on IBM DB2. Part 1 describes tips for database maintenance, statistics, and analysis for IBM DB2® for Linux®, UNIX®, and Windows®. Part 2 shows seven examples of common troubleshooting scenarios with IBM BPM and a `BPMDB` database that uses IBM DB2 for Linux, UNIX, and Windows. Then, Part 3 focuses on tips for database maintenance, statistics, and analysis for troubleshooting IBM BPM and a `BPMDB` database that uses Oracle.

First, in Part 1, learn about database maintenance and statistics, the number of process instances in the system, system tasks, the number of stored snapshots, JDBC driver versions, hardware, and environment limitations and related performance testing, a detailed troubleshooting analysis, and a summary of data that is required to debug an issue with the database that stores the runtime data that is associated with business process definition (BPD) processing (named BPMDB by default). This series addresses the BPMDB database but does not address the CMNDB (messaging and BPEL processing) and PDWDB (Performance Data Warehouse) databases that also are part of IBM BPM.

Database maintenance and statistics information

When you are facing database-related problems in IBM BPM, first focus on the database itself. In many cases, problems occur because the database maintenance failed.

For IBM DB2® for Linux®, UNIX®, and Windows®, many tools can optimize data distribution and performance. Listing 1 shows two SQL statements that give you a quick overview of what is actually stored in the database. The statements are not specific to IBM BPM. You can use them in various situations that involve a DB2 for Linux, UNIX, and Windows database.

Listing 1. SQL statements to query the database catalog

```
select tabschema, tabname, card, overflow, stats_time, npages, fpages, avgrowsize from syscat.tables;
select tabschema, tabname, indname, colnames, stats_time, indcard, lastused from syscat.indexes;
```

Table 1 shows an example of what you can learn from the output.

Table 1. Reduced sample output for the first example SQL statement

BPMDB	LSW_BPD_INSTANCE	76,531	23,000	Apr 1, 2015 08:00:26 PM 140775
BPMDB	LSW_SNAPSHOT	27,643	0	Mar 27, 2012 02:00:25 AM 240154
BPMDB	LSW_TASK	5,686,732	0	Apr 1, 2015 08:00:28 PM 993948
BPMDB	LSW_USR_XREF	-1	-1	-

In addition to the schema and the table name, first check the `stats_time` column, where the time stamp is stored when the last time statistics are collected for the associated database table. If the entry is a view, or the statistics are not collected for the database table, the column is marked with a `-` and the card and overflow display a `-1`. For example, see the last table, `LSW_USR_XREF`, in the example that is shown in Table 1. Current statistics are essential for a good performancebecause the DB2 optimizer bases the access path calculation on these numbers. Wrong statistics can lead to a bad execution path, which takes a long time to complete.

Statistics updates can be done with the `runstats` command in DB2. There is a special "with distribution and detailed indexes all" setting that can be beneficial because more details are collected on the distribution and in indexes. In many cases, only sampled statistics are collected, which can be misleading, but the previously mentioned statistics setting for the `runstats` command in DB2 considers all data.

Be aware that empty tables where statistics are collected can cause negative effects. The data distribution shows a typical production distribution during peak load. Empty tables are important to consider for cases where there are many process instances over the day, but they are automatically deleted in the evening. A `runstats` command that is run during the night would collect a data distribution that does not match the distribution during the day and would be counterproductive. In this case, consider collecting statistics during the day. An example `runstats` command is shown in Listing 2.

Listing 2. Execution of the runstats command with distribution statistics for the LSW_TASK table

```
runstats on table LSW_TASK WITH DISTRIBUTION AND DETAILED INDEXES ALL
```

In this situation, administrators normally raise the question of whether to use an automatic or manually triggered `runstats` command. There can be benefits for both options. To reduce effort, it makes sense to use the automatic statistics collection. However, consider the maintenance window so that data is not collected during peak load hours.

Also, consider the settings for the statistics collection. For example, the earlier mentioned distribution statistics are not collected by default, but they can be enforced with the db2profiles command. Also, see the Influence query optimization with optimization profiles and statistical views in DB2 9 developerWorks tutorial. Newer DB2 for Linux, UNIX, and Windows versions include more statistics features. Check the product documentation for the database version that you are using to make sure to benefit from all available features.

Consider that in IBM DB2 Linux, UNIX and Windows releases, follow the sequence of `runstats`, `reorg`, `runstats`, and `rebind` commands for an optimum statistics maintenance.

You can further optimize with the real-time statistics feature in DB2.

The next column in the table 1 is the cardinality, which is the number of unique values that are stored in a table. For IBM BPM, cardinality corresponds in most cases to the actual number of rows, so it is a good indicator for data volume – if statistics are up to date. Cardinality is especially important if the workload is changing, for example more users or batch jobs are added. The next section explores cardinality further, but note that it can be beneficial to keep historic values of the SQL statement output to identify changes on the system.

The overflow column lists the number of pages that are too small to store the changed content of a data row, which places a pointer at the original location and stores the content on a second page. This overflow affects performance, because each time both pages need to be accessed. Similar information can be gained by the `reorgchk` DB2 command, but its impact is larger because more data is processed.

The drawback on the SQL statement is that not all cases are covered without background knowledge. For example, when there are many inserts and deletes (which can correspond to the cleanup of process instances), then there are still many pages in use, because the data is randomly distributed. However, it is possible to compact the data on fewer pages, which is recommended when you run large delete operations. You can use the DB2 `reorg`, but the task is normally done by a database administrator, so make sure to consult the product documentation for the exact version of DB2 that you are using. See the REORG INDEXES/TABLE command topic in the IBM DB2 documentation on IBM Knowledge Center.

Number of process instances in the system

After you check the database maintenance, check the completed activities in the system. The BPMDB database is a runtime database, which means its purpose is to store data that is required immediately to handle running operations. Over time there can be a tremendous number of completed process instances and tasks that are created in the system, so it is important to have a cleanup strategy in place. The following SQL statements can help you understand what number of process instances and associated tasks are stored in the database (sorted by the snapshot ID of the application).

Listing 3. SQL statements to query the process instances and tasks sorted by snapshot ID and state

```
-- Provides process instances grouped by process instances state
select substr(snap.name,1,50) as "name",
       bpd.snapshot_id as "snapshot id",
       substr(code.NAME,1,20) as "status",
       count(bpd.EXECUTION_STATUS) as "number of instances"
from LSW_BPD_INSTANCE bpd
       left outer join lsw_bpd_status_codes code on code.STATUS_ID = bpd.execution_status
        left outer join lsw_snapshot snap on bpd.snapshot_id = snap.snapshot_id
group by snap.name, bpd.snapshot_id, code.NAME
order by snap.name, bpd.snapshot_id, code.NAME
with ur;

-- Provides task states grouped by task steps

select substr(snap.name,1,50) as "snapshot name",
       t.SNAPSHOT_ID,
       substr(t.SUBJECT,1,50) as "subject",
       substr(code.NAME,1,20) as "status",
       COUNT(t.STATUS) as "number"
from lsw_task t
       left outer join LSW_TASK_STATUS_CODES code on code.STATUS_VALUE = t.STATUS
       left outer join LSW_SNAPSHOT snap on t.snapshot_id = snap.snapshot_id
group by  snap.name, t.SNAPSHOT_ID, t.SUBJECT, code.NAME
order by snap.name, t.SNAPSHOT_ID, t.SUBJECT
with ur;
```

The example in Table 2 is the result of running the first statement of Listing 3 against a small sample database that has only one snapshot in place. You can see that there are 140 active process instances and 67583 completed instances. This example is normally a good indication that cleanup was not completed for a long time. The ratio between active instances and completed instances should not differ to this order of magnitude, because it can affect the query performance.

Table 2. Reduced sample output for the first provided SQL statement

TestCase_startProcessByName	823463b1-52fe-4ad0-bad9-3ee02fbff55d	Active	140
TestCase_startProcessByName	823463b1-52fe-4ad0-bad9-3ee02fbff55d	Completed	67583

In addition, if you want to understand the simple number of completed process instances and closed tasks, you should understand the relationship between process instances and tasks. An active process instance with many closed tasks can indicate problems in the context of loops, but also can give an indication for completed system tasks that is discussed in following sections.

For now, the queries of Listing 4 can give you a good estimate how much data is no longer required if the process design is considered. IBM teams that work with customers see cases where millions of completed tasks could be deleted while the system was running turn into issues with disk space usage or database backups. It's important to know what is stored in the system and how the large number of completed data that is still stored in the database affects the performance of the system.

If the data is no longer needed, a cleanup is required. IBM teams also see cases where the business design relied on storing the data for an number of years in the BPMDB database. This approach is normally not a good design, because the volume grows with data that is rarely used but must be considered for all query operations. If there is not sufficient high-end database hardware in place, reconsider the design.

Listing 4. SQL statements to query the process instances and associated tasks sorted by snapshot ID and state

```
 -- Provides completed process
                    instances and associated tasksselect substr(snap.name,1,50) as
                    "snapshot
                name",
           bpd.snapshot_id as "snapshot id",
           count(distinct bpd.bpd_instance_id) as "number of completed process instances",
           count(ta.status) as "number of closed tasks"
from LSW_BPD_INSTANCE bpd
        left outer join lsw_snapshot snap on bpd.snapshot_id = snap.snapshot_id
        left outer join lsw_task ta on bpd.bpd_instance_id = ta.bpd_instance_id
where bpd.bpd_instance_id = ta.bpd_instance_id and bpd.execution_status = 2
group by snap.name, bpd.snapshot_id with ur;

– Provides active process instances and associated closed tasks
select  substr(snap.name,1,50) as "snapshot name",
        bpd.snapshot_id as "snapshot id",
        count(distinct bpd.bpd_instance_id) as "number of running process instances",
        count(ta.status) as "number of closed tasks"
from LSW_BPD_INSTANCE bpd
        left outer join lsw_snapshot snap on bpd.snapshot_id = snap.snapshot_id
        left outer join lsw_task ta on bpd.bpd_instance_id = ta.bpd_instance_id
where bpd.bpd_instance_id = ta.bpd_instance_id and bpd.execution_status = 1 and ta.status = 32
group by snap.name, bpd.snapshot_id with ur;
```

Table 3 shows an example output for the first SQL statement in Listing 4. These example numbers give you a good understanding how many of the completed tasks are associated with completed process instances. The reduced example output lists completed process instances with associated closed tasks grouped by the `SNAPSHOT_ID` value.

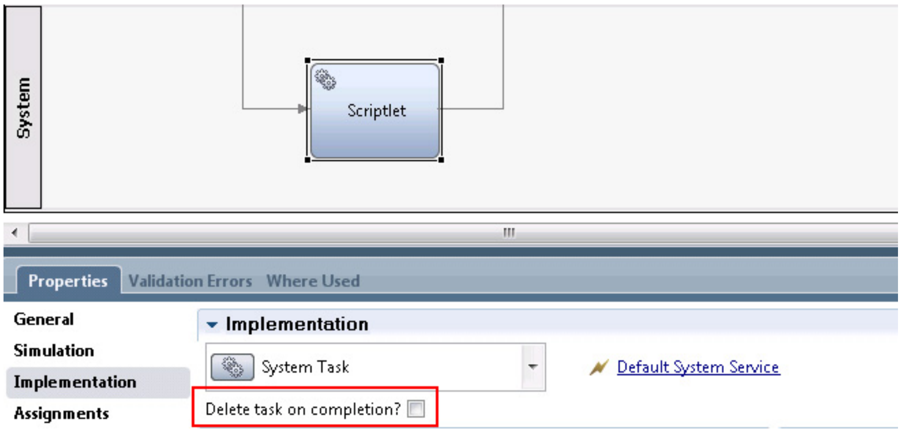
Table 3. Example output for the first SQL statement in Listing 4

`TestCase_StartProcessByName`	`823463b1-52fe-4ad0-bad9-3ee02fbff55d`	453	1359

System tasks

In addition to human tasks, the system tasks in IBM BPM are run directly by the IBM BPM system. You can modify the system task behavior in Process Designer with the **Delete task on completion** property, as shown in Figure 1. If you want the system tasks to be deleted when they are completed, select the check box.

Figure 1. Delete task on completion property for system tasks



Delete task on completion? ☐

Use the setting to delete system tasks immediately after their completion. Otherwise, they are deleted only at the end of the process lifecycle when the process instance is deleted. In some cases, these completed system task entries attributed to more than 40% of the stored data in the `LSW_TASK` table. The default behavior of the setting was changed to be enabled with IBM BPM V8.5, but for earlier releases it was disabled. When you see a large number of completed system tasks, you can change the application and slowly decrease the number of completed system tasks by doing normal housekeeping tasks for completed process instances. The housekeeping process deletes all associated objects, including system tasks.

To identify the affected database entries, use the following SQL statements to identify potential system tasks and their status: `select user_id, activity_name, status, count(*) from LSW_TASK group by user_id, activity_name, status with ur;`.

Check the associated description for limitations.

If a task is really a system task, you can see it only from the application model in Process Designer. However, there are some indications from the data. System tasks are run under the `tw_admin` user. In IBM BPM, every user is assigned an internal ID named `user_id`, which is used to assign tasks, for example. In versions earlier than IBM BPM V8.5, the `tw_admin` user is connected by default to the `user_id` that is set to the value of `1`. Starting in IBM BPM V8.5 deployment environments, the `user_id` for the `tw_admin` user can be different.

Therefore, the SQL statement in the previous example queries all user information, and it is possible to identify the `user_id` with the most associated tasks. It is also possible to sort the result and fetch only the top number of entries by adjusting the SQL statement. In addition to system tasks, the query can also give some indication of tasks in the context of looping.

There are cases when the `tw_admin` user can also be used to work on tasks, and activities can be assigned with a RunAs role for this user ID. This situation is rare, but you should consider it when you analyze the output of the SQL statements.

Number of stored snapshots

Another area that can be a potential bottleneck is the number of snapshots that are stored in a system. (Make sure to check toolkit snapshots, which are frequently neglected.) In this context, it is easy to see why a Process Server is recommended for a production system load instead of a Process Center, where many snapshots can coexist. Many IBM BPM fixes were published to improve snapshot performance.

In addition, consider the tuning of snapshot context and branch context cache. The cache has a default value of 64. If the number of snapshots or branches is more than 64, review the following developerWorks tutorial: Tuning branch and snapshot cache sizes in IBM Business Process Manager.

Of course, the best strategy is to avoid any unneeded data. Deleting snapshots is important, but before you delete, check the Issues with BPMDeleteSnapshot and BPMSnapshotCleanup commands in IBM Business Process Manager (BPM) support document for your version. Make sure to understand the known issues with the **BPMDeleteSnapshot** and **BPMSnapshotCleanup** commands and corresponding required interim fixes.

JDBC driver versions

The JDBC driver version is in most cases not an issue, but it is recommended to use the latest version and the same driver version as the database server. To match these versions, check the matrix in the DB2 JDBC Driver Versions and Downloads support document.

Hardware, environment limitations, and performance testing

IBM teams that work with customers see many systems where system scaling and load testing weren't completed. Proper system scaling and load testing of a production load are required for preventing unnecessary outages on production systems. For more information about these types of issues, see the IBM Redbooks publications about IBM BPM Performance (in the Related topics section).

It is essential for a production system to monitor the general system performance, including CPU, memory, disk input and output, and network throughput. Many corresponding tools are on the market to achieve this goal. For troubleshooting on UNIX systems the `nmon` tool is an option.

Monitoring is not a one-time event. It needs to be a permanent task, and someone needs to check the data regularly. Otherwise, it is not valuable for problem prevention. Also, monitoring data also can make troubleshooting easier. When monitoring tasks are done, predictions can be made for system utilization and potential bottlenecks (for example, a full hard disk, load increase). A system might be scaled perfectly today, but within the next month, a new batch job might be introduced with 5 million new process instances!

Especially for environments where different teams are involved for IBM BPM, the database, and the network, it is essential that all teams are aware on the availability for a required service. If your database is not running, do not wonder why there are no process instances processed. Also, problems with a Lightweight Directory Access Protocol (LDAP) system can completely block the IBM BPM system. A small change such as a port alteration or adding a firewall can block processing.

When the connection to the database is not working well or many connections are established and the processing is slow, you can end up with out of memory (OOM) situations. You face the same memory problems when the connection pool is completely used up. The database connections show up in the `heapdump` and `javacore` files, and you must figure out the origin of the problem.

Also, keep in mind that large memory usage for IBM BPM can be caused by imprecise queries that request millions of rows as a result set. During the IBM BPM product lifecycle, improvements were made to limit the data that is transferred between the IBM BPM system and the database. Use a newer release to benefit from the most recent improvements.

Another thing to consider when you run performance testing is the connection pool size. If you are performing load tests on an application or system and running out of connections, you might need to change the values to a higher value. The IBM Redbooks publications about IBM BPM performance that are listed in the Resources section provide valuable tips for starting values.

For a well performing database system, the I/O speed for the transaction logs storage for the database is important. Consider using a solid-state drive (SSD).

If you follow all the previous information, you solve most performance problems. In some cases you might need to collect further data. The following sections the general required data for the database and for IBM BPM. Iin most cases, data must be collected in parallel to correlate both sides.

Detailed analysis: Solving performance problems with the DB2 database

You can use the following DB2 tools for a deeper understanding of database problems in IBM BPM.

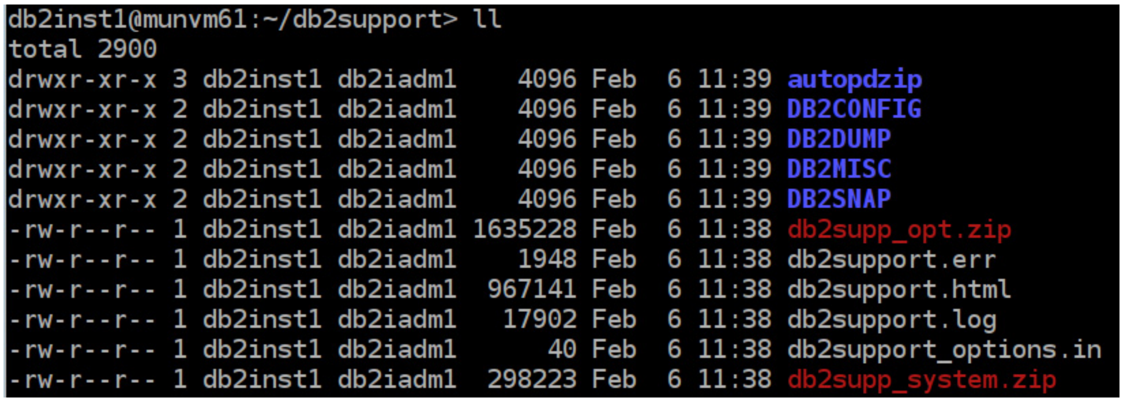
Collect db2support output

From the database side, it is always good to collect output with the **db2support** tool, as described in the Collecting Data for DB2 Compiler Issues IBM Support document.

Learn background information on more advanced options in the Choosing options for collecting optimizer db2support data IBM Support document.

The output structure for the **db2support** tool looks like the example in Figure 2.

Figure 2. Example db2support output structure



```
db2inst1@munvm61:~/db2support> ll
total 2900
drwxr-xr-x 3 db2inst1 db2iadm1    4096 Feb  6 11:39 autopdzip
drwxr-xr-x 2 db2inst1 db2iadm1    4096 Feb  6 11:39 DB2CONFIG
drwxr-xr-x 2 db2inst1 db2iadm1    4096 Feb  6 11:39 DB2DUMP
drwxr-xr-x 2 db2inst1 db2iadm1    4096 Feb  6 11:39 DB2MISC
drwxr-xr-x 2 db2inst1 db2iadm1    4096 Feb  6 11:39 DB2SNAP
-rw-r--r-- 1 db2inst1 db2iadm1 1635228 Feb  6 11:38 db2supp_opt.zip
-rw-r--r-- 1 db2inst1 db2iadm1    1948 Feb  6 11:38 db2support.err
-rw-r--r-- 1 db2inst1 db2iadm1  967141 Feb  6 11:38 db2support.html
-rw-r--r-- 1 db2inst1 db2iadm1   17902 Feb  6 11:38 db2support.log
-rw-r--r-- 1 db2inst1 db2iadm1      40 Feb  6 11:38 db2support_options.in
-rw-r--r-- 1 db2inst1 db2iadm1  298223 Feb  6 11:38 db2supp_system.zip
```

The main output directory includes the `db2support.html` file, which gives access to the main configuration option and can be opened in any browser. In Figure 3 all the subcategories that are available are shown.

Figure 3. The table of contents as shown in the `db2support.html` file

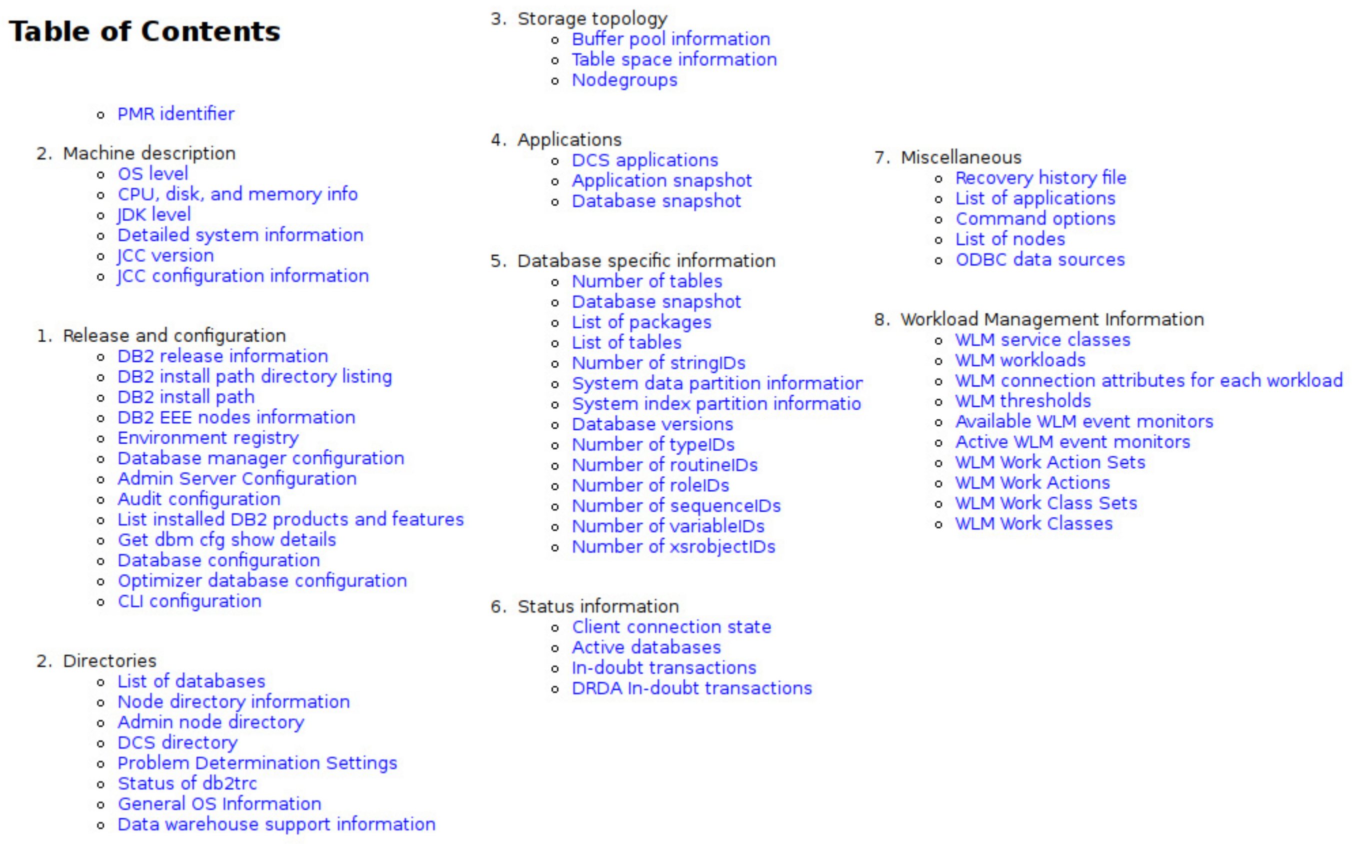


Table of Contents

- PMR identifier

2. Machine description
   - OS level
   - CPU, disk, and memory info
   - JDK level
   - Detailed system information
   - JCC version
   - JCC configuration information

1. Release and configuration
   - DB2 release information
   - DB2 install path directory listing
   - DB2 install path
   - DB2 EEE nodes information
   - Environment registry
   - Database manager configuration
   - Admin Server Configuration
   - Audit configuration
   - List installed DB2 products and features
   - Get dbm cfg show details
   - Database configuration
   - Optimizer database configuration
   - CLI configuration

2. Directories
   - List of databases
   - Node directory information
   - Admin node directory
   - DCS directory
   - Problem Determination Settings
   - Status of db2trc
   - General OS Information
   - Data warehouse support information

3. Storage topology
   - Buffer pool information
   - Table space information
   - Nodegroups

4. Applications
   - DCS applications
   - Application snapshot
   - Database snapshot

5. Database specific information
   - Number of tables
   - Database snapshot
   - List of packages
   - List of tables
   - Number of stringIDs
   - System data partition informatior
   - System index partition informatio
   - Database versions
   - Number of typeIDs
   - Number of routineIDs
   - Number of roleIDs
   - Number of sequenceIDs
   - Number of variableIDs
   - Number of xsrobjectIDs

6. Status information
   - Client connection state
   - Active databases
   - In-doubt transactions
   - DRDA In-doubt transactions

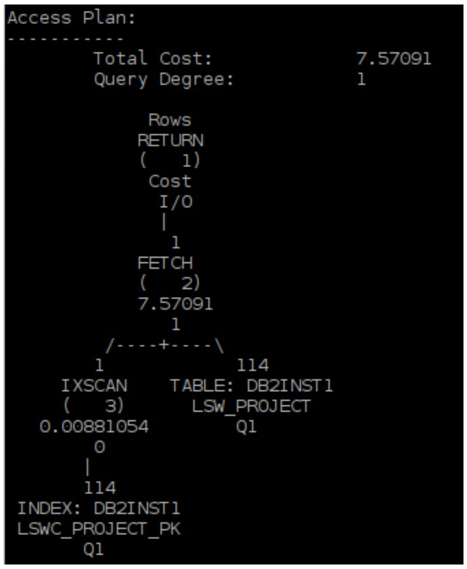
7. Miscellaneous
   - Recovery history file
   - List of applications
   - Command options
   - List of nodes
   - ODBC data sources

8. Workload Management Information
   - WLM service classes
   - WLM workloads
   - WLM connection attributes for each workload
   - WLM thresholds
   - Available WLM event monitors
   - Active WLM event monitors
   - WLM Work Action Sets
   - WLM Work Actions
   - WLM Work Class Sets
   - WLM Work Classes

The **db2support** tool also collects available log files from the IBM DB2 system and from the `db2exfmt` output (if requested by the `-cl 1` option). The `db2exfmt` output is stored in the `db2supp_opt.zip` archive and includes additional information for the execution, for example, the statement that was analyzed. To access the data, you must extract the `db2supp_opt.zip` archive.

Example `db2exfmt` output is shown in Figure 4. The access plan that is visualized in a `db2exfmt` output can quickly provide indications at which execution steps delays are seen. For example, if an index is missing, the **db2advis** tool can also provide guidance on missing indexes. In addition, the output provides details on the used statistics for the access plan calculation and on the number of rows that are involved in the execution.

Figure 4. Example `db2exfmt` output for a statement that is run against the BPMDB database



```
Access Plan:
-----------
        Total Cost:                     7.57091
        Query Degree:                   1

             Rows
            RETURN
            (   1)
             Cost
              I/O
              |
               1
            FETCH
            (   2)
            7.57091
               1
         /----+----\
        1            114
     IXSCAN     TABLE: DB2INST1
     (   3)       LSW_PROJECT
   0.00881054        Q1
        0
       |
       114
 INDEX: DB2INST1
 LSWC_PROJECT_PK
       Q1
```

To collect `explain` information in real time for a DB2 Linux or UNIX system, use the **db2top** command. See the DB2 problem determination using the db2top utility developerWorks tutorial.

Check the `db2diag.log` file

The main log file for an IBM DB2 system is the `db2diag.log` file, which records all significant events for the database system. Find the file in the db2support output under the `DB2DUMP` directory. Sometimes it is necessary to log informational messages, which can be done by changing the `diaglevel` value to `4`. It requires restarting the server but can be removed after your problem is resolved. To increase the logging level, follow the steps in Listing 5. The default value is 3.

Listing 5. Changing the DB2 diaglevel (use required values for instance-name and value)

```
db2 attach to <instance-name>
db2 update dbm cfg using DIAGLEVEL <value>
db2 detach
```

Check the LOCKTIMEOUT value

One critical parameter that is frequently set incorrectly is `LOCKTIMEOUT`. It defines how long a transaction that requests a lock on a database entry waits before it rolls back, if the database entry is already locked by a different transaction.

The BPMDB is an online transaction processing database with recommended values for the `LOCKTIMEOUT` as 30 - 60 seconds (see

DB2 Tuning Tips for OLTP Applications). If the database system can handle quicker queries, the `LOCKTIMEOUT` values are sometimes lowered further. The default setting is `-1`, which means that the second transaction waits forever, which can lead to further lock contention because all other locks are kept during this time.

Sometimes when administrators find lock timeouts, they attempt to increase the `LOCKTIMEOUT` value. However, that approach does not actually solve the problem – it just hides it. Lock timeouts can be triggered by long running operations, so keep execution times low.

A common example for a lock timeout is a lock that is held on the `LSW_BPD_INSTANCE` table when many email notifications are sent and the email server is processing slowly. An interim fix in IBM BPM allows a `100Custom.xml` property to switch to asynchronous email, which releases the lock much earlier. See the fixes available for JR46424.

Because the `LSW_BPD_INSTANCE` table is one central table for processing, many other operations can lock the table. Troubleshoot on a case-by-case basis. Also, tracing can help this situation, which is described in a following section.

Check the default isolation level

In some cases, administrators attempt to change the default isolation level. This approach is not a good idea because it can lead to inconsistent data processing and the data cannot be found. For an IBM BPM system with a DB2 database, the isolation level on the WebSphere Application Server side should be set to `4` (a `TRANSACTION_REPEATABLE_READ` JDBC isolation level, which is the default, corresponding to the read stability for the DB2 server).

Detailed analysis: Tracing the IBM BPM system

To correlate what is sent by the IBM BPM system and what is observed on the database side, a trace is really a requirement to know what is happening. There are many trace settings available.

The following example shows a very basic trace, which covers some basic SQL statement information:

```
WAS.clientinfopluslogging=all:org.springframework.jdbc.*=all
```

The `WAS.clientinfopluslogging` is required to log the SQL statements itself and the transaction boundaries. Sometimes there are different versions of `org.springframework.jdbc`, for example limitations to `org.springframework.jdbc.core`. The core part traces the values that are passed to the SQL statement when you use the Spring Framework, an application framework for the Java™

platform. However, the database trace subsection tracks when a connection from the data source is acquired and returned, which is valueable information, especially to track execution. From this perspective, always use the `org.springframework.jdbc.*=all trace` setting.

To include the background from where the statement was originally triggered you also need to add the following IBM BPM product tracing (which can affect performance, but it is the only way to analyze problems efficiently): `WLE.*=all:com.ibm.bpm.*=all:WAS.clientinfopluslogging=all:org.springframework.jdbc.*=all`

It is important to check the number of trace files and the size of each trace file. A good trace should cover more than a few minutes and include the problem in question. If an activity lasts 15 minutes, the trace should at least cover 20 minutes, to be sure that the complete navigation history is covered.

In some situations it is necessary to trace the following additional information:

- Database connection pool usage can be tracked by the additional settings of `WAS.j2c=all:RRA=all:Transaction=all`
- Security-related problems in combination with the database are best addressed with the additional setting of VMM and security traces:
  `com.ibm.websphere.wim.*=all: com.ibm.ws.wim.*=all: com.ibm.wsspi.wim.*=all:`
  `com.ibm.ws.security.*=all:`
  Reduced settings for security traces are published so that there is less of an effect on the trace if performance is an issue.
- There is an additional trace setting from the WebSphere Application Server side: `com.ibm.ws.rsadapter.jdbc.*=all`, which normally collects data that is not frequently required from an IBM BPM perspective. Therefore, the reduced set for tracking only prepared SQL statement can be a good alternative: `com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement.*=all`. Be aware that not all statements need to be run as a prepared statement.
- For a deeper analysis, you can use JDBC driver traces. For IBM DB2 Linux, UNIX and Windows, the JDBC driver can be enabled with the following setting: `com.ibm.ws.db2.logwriter=all`.
  Be careful: by default all trace classes are enabled on the data source that corresponds to `traceLevel -1`, which generates huge data logs, even on a low-volume system. From this perspective, full logging is not recommended, but a reduced subclass set works, if the earlier mentioned traces are not already sufficient. Corresponding `traceLevel` options for DB2 are included in the following list. You can combine individual `traceLevel` settings by adding corresponding `traceLevel` values. For example, if you want to enable `TRACE_CONNECTION_CALLS` and `TRACE_STATEMENT_CALLS`, add `1` and `2`, and the `traceLevel` setting is `3`.
  Choose from the following DB2 JDBC driver trace levels:
  `TRACE_ALL = -1`
  `TRACE_NONE = 0`
  `TRACE_CONNECTION_CALLS = 1`

```
TRACE_STATEMENT_CALLS = 2
TRACE_RESULT_SET_CALLS = 4
TRACE_DRIVER_CONFIGURATION = 16
TRACE_CONNECTS = 32
TRACE_DRDA_FLOWS = 64
TRACE_RESULT_SET_META_DATA = 128
TRACE_PARAMETER_META_DATA = 256
TRACE_DIAGNOSTICS = 512
TRACE_SQLJ = 1024
TRACE_XA_CALLS = 2048
TRACE_META_CALLS = 8192
TRACE_DATASOURCE_CALLS = 16384
TRACE_LARGE_OBJECT_CALLS = 32768
TRACE_SYSTEM_MONITOR = 131072
TRACE_TRACEPOINTS = 262144
```

A server restart is required after trace level changes.

Depending on the nature of the problem, you need to decide which trace level is required. For example, as a quick solution, try the `traceLevel 6`, if you are interested in a combination of statement calls and results.

Summary of data that is required to debug a BPMDB database issue

In summary, the following data is required to debug a BPMDB database issue, and can be shared with the IBM Support team:

- Output of the SQL statements, like the examples available in the `code_sample.zip` file in the Download section. (From the Download section of this tutorial, download the `code_sample.zip` and extract the file.) (The latest versions are also available on Github at http://github.com/stephan-volz/database-scripts/blob/master/DB2_ProcessInstances_Tasks.txt.)
- A **db2support** output. If there is already a performance problem for a specific statement that is identified, also include **db2exfmt** output.
- A trace, if you can re-create the problem. The following trace specification works best: `WLE.*=all:com.ibm.bpm.*=all:WAS.clientinfopluslogging=all:org.springframework.jdbc.*=all`

Conclusion

This tutorial discussed various tools and ways to troubleshoot your BPMDB database in IBM BPM. Nearly all of these tools can be used in other situations as well.

The statements in this tutorial are for IBM BPM releases through IBM BPM V8.5.6, which is the current release at time of publication. Future releases might require adjustments to the SQL statements, although no major changes are expected.

When you observe a problem in the BPMDB database, check the following items:

- Database maintenance and statistics (See the `code_sample.zip` and extract the file for examples.)
- Process instances in the system (See the `code_sample.zip` and extract the file for examples.)
- System tasks, including delete on completion (See the `code_sample.zip` and extract the file for examples.)
- Snapshot information, including delete and cleanup commands for snapshots
- Current JDBC drivers
- Hardware environment limitations
- Database configuration, including the **db2support** tool and the `LOCKTIMEOUT` value.
- The data that is required to debug a complex BPMDB issue, including database tools and IBM BPM trace settings.

See Part 2 for examples of database troubleshooting in IBM BPM.

To learn about troubleshooting in environments with an Oracle database, see Part 3.

Acknowledgements

The authors would like to thank Torsten Wilms and Richard Metzger for their review and contributions to this tutorial.

Downloadable resources

Description	Name	Size
Code sample	code_sample.zip	749KB

Related topics

- Lock events for DB2 for Linux, UNIX, and Windows, Part 3: Use the lock event monitor in DB2 9.7 to solve concurrency issues
- Mining your package cache for problem SQL in DB2 for Linux, UNIX and Windows
- DB2 Tuning Tips for OLTP Applications
- Influence query optimization with optimization profiles and statistical views in DB2 9
- Purging data in IBM Business Process Manager
- IBM Performance Analysis Suite (developerWorks community)
- 5 Things to Know About IBM BPM Performance Tuning (developerWorks blog)
- Collecting Data for DB2 Compiler Issues (IBM Support document)
- Reading and decoding instrumentation files for WebSphere Lombardi Edition (WLE), and IBM Business Process Manager (BPM) products (IBM Support document)
- Tuning connection pools (IBM WebSphere Application Server documentation on IBM Knowledge Center)
- ILRT - Instrumentation Log Report Tool for IBM BPM on GitHub (contributed by Andy Fedotov)
- BPM system performance evaluation basics (BP3 website)
- IBM Business Process Manager V8.5 Performance Tuning and Best Practices: An IBM Redbooks publication
- IBM Business Process Manager V8.0 Performance Tuning and Best Practices: An IBM Redbooks publication
- IBM Business Process Manager V7.5 Performance Tuning and Best Practices: An IBM Redbooks publication