# DB2 for z/OS

# Limiting Access to the Database Server using the Resource Limit Facility RLF

summer 2011

# Intro

This document bases on common IBM DB2 documentation and support notes for DB2 Version 9.1 for z/OS, assembled in summer 2011.

DB2 includes a resource limit facility (RLF or „governor"), which helps control the use of system resources by DB2. Other facilities, such as z/OS Workload Manager (WLM), and the DB2 QMF governor, complement the DB2 governor.

DB2 includes a resource limit facility (governor), which helps control the use of system resources by DB2. Other facilities, such as z/OS Workload Manager (WLM), and the DB2 QMF governor, complement the DB2 governor. Because DB2 is integrated with the operating system and transaction subsystems, control definitions are used in most cases.

RLF or „governor" enables you to limit resource usage, bind operations, and parallelism modes in dynamic query processing. You can use two modes of governing to limit the amount of system resources that a dynamic „S-I-U-M-T-D[1]" query uses. These modes are called reactive and predictive. You can use either mode or both together.

You can use the resource limit facility to perform the following activities:

- Set warning and error thresholds for dynamic „S-I-U-M-T-D" statements. The resource limit facility can inform users (via your application programs) that a processing limit might be exceeded for a particular dynamic SQL statement. This is called predictive governing.
- Stop dynamic SQL statements („S-I-U-M-T-D" ) that exceed the processor limit that you specified. This is sometimes called reactive governing.
- Restrict bind and rebind activities to avoid performance impacts on production data.

Nota bene: The resource limit facility does not control static SQL statements regardless of whether they are executed locally or remotely. And, no limits apply to primary or secondary authorization IDs with installation SYSADM or installation SYSOPR authority.

# Types of Govering

## Reactive Governing

You can use the DB2 resource limit facility to set limits for reactive governing, meaning that DB2 actually stops a dynamic query that overuses system resources. When you specify reactive governing, the resource limit facility stops a currently running query that meets the conditions in a resource limit table row when the query uses more than the maximum amount of resources specified by the value of ASUTIME in that row. No limits apply to primary or secondary authorization Ids with installation SYSADM or installation SYSOPR authority.

To specify reactive governing: Specify either of the following values in the RLFFUNC column of the resource limit table:

blank Govern by plan name (RLST)

'2' Govern by package name (RLST)

'8' Govern by client information (RLMT)

DB2 resets the accumulated ASUTIME at the following events:

- After the execution completes for an INSERT or UPDATE statement that includes a parameter marker.
- During the PREPARE and after the close and reopen of the cursor for a FETCH statement.

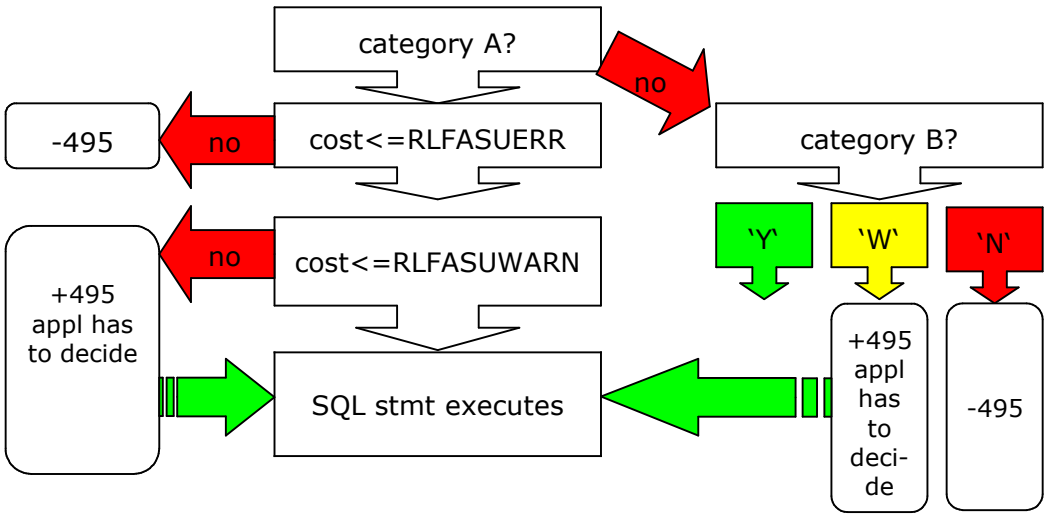
[1] Any one of a SELECT, INSERT, UPDATE, MERGE, TRUNCATE, or DELETE SQL statement.

Any statement that exceeds a limit that you set in the RLST terminates with a –905 SQLCODE and a corresponding '57014' SQLSTATE. You can establish a single limit for all users, different limits for individual users, or both. Limits do not apply to primary or secondary authorization IDs with installation SYSADM or installation SYSOPR authority. For queries that enter DB2 from a remote site, the local site limits are used.

## Predictive governing

The DB2 provides a predictive governing capability that avoids wasting processing resources by giving you the ability to prevent a query from running when it appears likely to exceed processing limits. In reactive governing, those resources are already used before the query is stopped.

The following figure provides an overview of how predictive governing works.




At prepare time for a dynamic SELECT, INSERT UPDATE, MERGE, or DELETE statement, DB2 searches the active resource limit table to determine if the processor cost estimate exceeds the error or warning threshold that you set in RLFASUWARN and RLFASUERR columns for that statement. DB2 compares the cost estimate for a statement to the thresholds that you set, and the following actions occur:

- If the cost estimate is in cost category A and the error threshold is exceeded, DB2 returns a -495 SQLCODE to the application, and the statement is not prepared or run.
- If the estimate is in cost category A and the warning threshold is exceeded, a +495 SQLCODE is returned at prepare time. The prepare is completed, and the application or user decides whether to run the statement.
- If the estimate is in cost category B, DB2 takes the action that you specify in the RLF_CATEGORY_B column; that is, it either prepares and executes the statement, does not prepare or execute the statement, or returns a warning SQLCODE, which lets the application decide what to do.
- If the estimate is in cost category B and the warning threshold is exceeded, a +495 SQLCODE is returned at prepare time. The prepare is completed, and the application or user decides whether to run the statement.

To specify predictive governing: Specify either of the following values in the RLFFUNC column of a resource limit table:

'6' Govern by plan name (RLST)

'7' Govern by package name (RLST)

'9' Govern by client information (RLMT)

## Cost categories

DB2 provides cost estimates, in service units and in milliseconds, for SELECT, INSERT, UPDATE, and DELETE statements, both static and dynamic. The estimates do not take into account several factors, including cost adjustments that are caused by parallel processing, or the use of triggers or user-defined functions.

DB2 puts its cost estimate into one of two cost categories: category A or category B. Estimates that go into cost category A are the ones for which DB2 has adequate information to make an estimate. That estimate is not likely to be 100% accurate, but is likely to be more accurate than any estimate that is in cost category B.

DB2 puts estimates into cost category B when it is forced to use default values for its estimates, such as when no statistics are available, or because host variables are used in a query.

If system administrators use these estimates as input into the resource limit specification table for governing (either predictive or reactive), they should give more allowance to statements in cost category B than those in cost category A.

Nota bene: Because the cost estimate of category B statements are unprecise, they are also good candidates for reactive governing.

You can use EXPLAIN to populate the DSN_STATEMNT_TABLE with cost estimates at the same time as your PLAN_TABLE is being populated. See section „DSN_STATEMNT_TABLE description“ at page 10.

### Cost category A

DB2 puts everything that doesn't fall into category B into category A.

### Cost category B

DB2 puts a statement's estimate into cost category B when any of the following conditions exist:

- The statement has UDFs.
- Triggers are defined for the target table:
  - The statement uses an insert operation, and insert triggers are defined on the target table.
  - The statement uses an update operation, and update triggers are defined on the target table.
  - The statement uses a delete operation, and delete triggers are defined on the target table.
- The target table of a delete statement has referential constraints defined on it as the parent table, and the delete rules are either CASCADE or SET NULL.
- The WHERE clause predicate has one of the following forms:
  - COL op constant, and the constant is a host variable, parameter marker, or special register. The operator can be >, >=, <, <=, LIKE, or NOT LIKE.
  - COL BETWEEN constant AND constant where either constant is a host variable, parameter marker, or special register.
  - LIKE with an escape clause that contains a host variable.
- The cardinality statistics are missing for one or more tables that are used in the statement.
- A subselect in the SQL statement contains a HAVING clause.

# Installation

## System Parameters

During installation of DB2 the system administrator has to specify options and parameters considered RLF on the DSNTIPB.

- RLF AUTO START field (RLF subsystem parameter): The RLF subsystem parameter specifies whether the resource limit facility (governor) is to automatically start each time DB2 is started. Acceptable values: YES, NO (default)
- RLST NAME SUFFIX field (RLFTBL subsystem parameter): The RLFTBL subsystem parameter specifies the suffix that is to be used for the default resource limit specification table (RLST). The default RLST is used when the resource limit facility (governor) is automatically started or when the governor is started without a specified suffix. Acceptable values: any 2 alphanumeric characters; national characters are not allowed, default is 01.
- RLST ACCESS ERROR field (RLFERR subsystem parameter): The RLFERR subsystem parameter specifies what DB2 is to do if the governor encounters a condition that prevents it from accessing the resource limit specification table. This setting also applies if DB2 cannot find an applicable row in the resource limit specification table. An applicable row is one that applies to the authorization ID, plan or package name, and name of the logical unit of work of the query user. Acceptable values: NOLIMIT, NORUN (default), 1 to 5000000.

  NOLIMIT: Allows all dynamic SQL statements to run without limit.

  NORUN: Terminates all dynamic SQL statements immediately with an SQL error code.

  1 to 5000000: Specifies the default limit for the number of resource limit specification table access errors. If the limit is exceeded, the SQL statement is terminated.
- RESOURCE AUTHID field (RLFAUTH subsystem parameter): The RLFAUTH subsystem parameter specifies the authorization ID that is to be used if you plan to use the resource limit facility (governor). Acceptable values: 1 to 8 characters, default is SYSIBM.
- RLST ACCESS ERROR field (RLFERRD subsystem parameter): The RLFERRD subsystem parameter specifies the action that DB2 is to take if the governor encounters a condition that prevents it from accessing the resource limit specification table. This setting also applies if DB2 cannot find an applicable row in the resource limit specification table. An applicable row is one that applies to the authorization ID, plan or package name, and name of the logical unit of work of the query user. This parameter applies to the distributed data facility. Acceptable values: NOLIMIT (default), NORUN, 1 to 5000000.

  NOLIMIT: Allows all dynamic SQL statements to run without limit.

  NORUN: Terminates all dynamic SQL statements immediately with an SQL error code.

  1 to 5000000: Specifies the default limit for the number of resource limit specification table access errors. If the limit is exceeded, the SQL statement is terminated.

## Installation and Customization of the RLF using Control Tables

When you install DB2, installation job DSNTIJSG creates a database, table space, table, and descending index for the resource limit specification. You can tailor those statements. Resource limit tables can reside in any database; however, because a

database has some special attributes while the resource limit facility is active, it is best to create resource limit tables in their own database. Only one resource limit table can be active at any particular time, however you might create several instances of either or both types of resource limit tables and use different instances at different times.

You enable the resource limit facility by populating the resource limit tables with rows of data that describe the limits. To insert, update, merge, or delete data in the resource limit table, you need only the usual table privileges on the resource limit table. No higher authorities are required. Use INSERT, UPDATE, MERGE, and DELETE statements to populate the resource limit table.

Changes to the resource limit specification table are immediately effective for all new threads, and for those that have not issued their first dynamic „S-I-U-M-T-D" statement. However, if you change the resource limit specification table while a thread is executing, the limit that existed when the thread issued its first dynamic statement applies throughout the life of the thread until DB2 reads in the new limit. DB2 reads in a new limit in the following situations:

- the application uses a different primary authorization ID.
- the resource limit facility is stopped and started again.
- a predictively governed package or DBRM is loaded for execution.

You can create several resource limit tables and start and stop them as needed to support the type and amount of processing that occurs at different times. If the governor is active and you restart it without stopping it, any jobs that are active continue to use their original limits, and all new jobs use the limits in the new table.

While the resource limit facility (governor) is active, you cannot execute certain SQL statements on the resource limit table, or the table space and database that contain the resource limit table.



---

**DSNRLSTxx description**

RLST resource limit tables can be used to limit the amount of resource used by queries. Queries can be limited based on information about the query, including the plan name, collection ID, package name, authorization ID, and location name of the query.

The values in each row define a limit, including the function and scope of each limit. The function of a particular limit is defined by the value of the RLFUNC column. Other columns specify the scope for the limit defined by the row. For example, you can specify that a limit broadly by leaving the AUTHID column blank, which means that the row applies to all authorization IDs. Or, you can specify that the limit applies more narrowly by specifying a different row for each authorization ID for which the function applies. Some columns are blank in each row that specifies a valid limit. DB2 tries to find the most exact match when it determines which row to use for a particular function. The search order depends on the type of governing that is specified. The search order is described under each of those functions.

| Column | Description |
|---|---|
| AUTHID | The resource specification limits apply to this primary authorization ID. A blank means that the limit specifications in this row apply to all authorization IDs for the location that is specified in LUNAME. No limits apply to primary or secondary authorization Ids with installation SYSADM or installation SYSOPR authority. |
| PLANNAME | The resource specification limits apply to this plan. If you are specifying a function that applies to plans (RLFFUNC=blank or '6'), a blank means that the limit specifications in this row apply to all plans for the location that is specified in LUNAME. Qualify by plan name only if the dynamic statement is issued from a DBRM bound in a plan, not a package; otherwise, DB2 does not find this row. If the RLFFUNC column contains a function for packages ('1,' '2,' or '7'), this column must be blank; if it is not blank, the row is ignored. |

| Column | Description |
|---|---|
| ASUTIME | The number of processor service units allowed for any single dynamic „S-I-U-M-T-D" statement. Use this column for reactive governing.<br>Other possible values and their meanings are:<br>null No limit<br>0 (zero) or a negative value: No dynamic „S-I-U-M-T-D" statements are permitted.<br>A relative metric is used so that the resource limit table values do not need to be modified when processors are changed. However, in some cases, DB2 workloads can differ from the measurement averages. In these cases, resource limit table value changes might be necessary. |
| LUNAME | The LU name of the location where the request originated. A blank value in this column represents the local location, not all locations.<br>The value PUBLIC represents all of the DBMS locations in the network; these locations do not need to be DB2 subsystems. PUBLIC is the only value for TCP/IP connections. |
| RLFFUNC | blank The row reactively governs dynamic „S-I-U-M-T-D" statements by plan name.<br>'1' The row reactively governs bind operations.<br>'2' The row reactively governs dynamic „S-I-U-M-T-D" statements by package or collection name.<br>'3' The row disables query I/O parallelism.<br>'4' The row disables query CP parallelism.<br>'5' The row disables Sysplex query parallelism.<br>'6' The row predictively governs dynamic „S-I-U-M-T-D" statements by plan name.<br>'7' The row predictively governs dynamic „S-I-U-M-T-D" statements by package or collection name.<br>All other values are ignored. |
| RLFBIND | Shows whether bind operations are allowed. An 'N' implies that bind operations are not allowed. Any other value means that bind operations are allowed. This column is used only if RLFFUNC is set to '1'. |
| RLFCOLLN | Specifies a package collection. A blank value in this column means that the row applies to all package collections from the location that is specified in LUNAME. Qualify by collection name only if the dynamic statement is issued from a package; otherwise DB2 does not find this row. If RLFFUNC=blank, '1,' or '6', RLFCOLLN must be blank. |
| RLFPKG | Specifies a package name. A blank value in this column means that the row applies to all packages from the location that is specified in LUNAME. Qualify by package name only if the dynamic statement is issued from a package; otherwise DB2 does not find this row. If RLFFUNC=blank, '1', or '6', RLFPKG must be blank. |
| RLFASUERR | Used for predictive governing (RLFFUNC= '6' or '7'), and only for statements that are in cost category A. The error threshold number of system resource manager processor service units allowed for a single dynamic „S-I-U-M-T-D" statement. If the predicted processor |

| Column | Description |
|---|---|
| | cost (in service units) is greater than the error threshold, an SQLCODE -495 is returned to the application.<br>Other possible values and their effects are:<br>null No error threshold<br>0 (zero) or a negative value: All dynamic „S-I-U-M-T-D" statements receive SQLCODE -495. |
| RLFASUWARN | Used for predictive governing (RELFFUNC= '6' or '7'), and only for statements that are in cost category A. The warning threshold number of processor service units that are allowed for a single dynamic „S-I-U-M-T-D" statement. If the predicted processor cost (in service units) is greater than the warning threshold, an SQLCODE +495 is returned to the application.<br>Other possible values and their effects are:<br>null No warning threshold<br>0 (zero) or a negative value: All dynamic „S-I-U-M-T-D" statements receive SQLCODE +495.<br>Important: Make sure the value for RLFASUWARN is less than that for RLFASUERR. If the warning value is higher, the warning is never reported. The error takes precedence over the warning. |
| RLF_CATEGORY_B | Used for predictive governing (RLFFUNC='6' or '7'). Tells the governor the default action to take when the cost estimate for a given statement falls into cost category B, which means that the predicted cost is indeterminate and probably too low. You can tell if a statement is in cost category B by running EXPLAIN and checking the COST_CATEGORY column of the DSN_STATEMNT_TABLE.<br>Possible values are:<br>blank By default, prepare and execute the SQL statement.<br>Y Prepare and execute the SQL statement.<br>N Do not prepare or execute the SQL statement. Return SQLCODE -495 to the application.<br>W Complete the prepare, return SQLCODE +495, and allow the application logic to decide whether to execute the SQL statement or not. |

**DSNRLMTxx description**

Resource limit tables can be used to limit the amount of resources used be dynamic queries that run on middleware servers. Queries can be limited based on client information, including the application name, user ID, workstation ID, and IP address of the client.

The values in each row define a limit, including the function and scope of each limit. The function of a particular limit is defined by the value of the RLFUNC column. Other columns specify the scope for the limit defined by the row. For example, you can specify that a limit applies broadly by leaving the RLFEUAN column blank, which means that the row applies to all end user IDs, or you can specify limits narrowly by specifying a different row for each end user ID for which the function applies. DB2 tries to find the most exact match when it determines which row to use for a particular function. The search order depends on whether reactive or predictive governing is specified. The search order is described under each of those functions.

| Column | Description |
|---|---|
| RLFFUNC | Specifies how the row is used. The values that have an effect are: |

| Column | Description |
|---|---|
| | '8' The row reactively governs dynamic „S-I-U-M-T-D" statements by client information (RLEUID, RLFEUAN, RLFEUWN, and RLFIP).<br>'9' The row predictively governs dynamic „S-I-U-M-T-D" statements by client information (RLEUID, RLFEUAN, RLFEUWN, and RLFIP).<br>All other values are ignored. |
| RLFEUAN | Specifies an application name. A blank value in this column means that the row applies to all application names from the location specified in RLFIP. |
| RLFEUID | Specifies an end user's user ID. A blank value means that the limit specifications in this row apply to every user ID for the location that is specified in RLFIP. |
| RLFEUWN | Specifies an end user's workstation name. A blank value in this column means that the row applies to all workstation names from the location that is specified in RLFIP. |
| RLFIP | The IP address of the location where the request originated. A blank value in this column represents all locations. |
| ASUTIME | The number of processor service units allowed for any single dynamic „S-I-U-M-T-D" statement. Use this column for reactive governing.<br>Other possible values and their meanings are:<br>null No limit<br>0 (zero) or a negative value: No dynamic „S-I-U-M-T-D" statements are permitted.<br>A relative metric is used so that the resource limit table values do not need to be modified when processors are changed. However, in some cases, DB2 workloads can differ from the measurement averages. In these cases, resource limit table value changes might be necessary. |
| RLFASUERR | Used for predictive governing (RLFFUNC= '9'), and only for statements that are in cost category A. The error threshold number of system resource manager processor service units allowed for a single dynamic „S-I-U-M-T-D" statement. If the predicted processor cost (in service units) is greater than the error threshold, an SQLCODE -495 is returned to the application<br>Other possible values and their effects are:<br>null No error threshold<br>0 (zero) or a negative value: All „S-I-U-M-T-D" statements receive SQLCODE -495. |
| RLFASUWARN | Used for predictive governing (RELFFUNC= '9'), and only for statements that are in cost category A. The warning threshold number of processor service units that are allowed for a single dynamic „S-I-U-M-T-D" statement. If the predicted processor cost (in service units) is greater than the warning threshold, an SQLCODE +495 is returned to the application.<br>Other possible values and their effects are:<br>null No warning threshold<br>0 (zero) or a negative value: All dynamic „S-I-U-M-T-D" statements receive SQLCODE +495. |

| Column | Description |
|---|---|
| | Important: Make sure the value for RLFASUWARN is less than that for RLFASUERR. If the warning value is higher, the warning is never reported. The error takes precedence over the warning. |
| RLF_CATEGORY_B | Used for predictive governing (RLFFUNC='9'). Tells the governor the default action to take when the cost estimate for a given statement falls into cost category B, which means that the predicted cost is indeterminate and probably too low. You can tell if a statement is in cost category B by running EXPLAIN and checking the COST_CATEGORY column of the DSN_STATEMNT_TABLE.<br>The acceptable values are:<br>blank By default, prepare and execute the SQL statement.<br>Y Prepare and execute the SQL statement.<br>N Do not prepare or execute the SQL statement. Return SQLCODE -495 to the application.<br>W Complete the prepare, return SQLCODE +495, and allow the application logic to decide whether to execute the SQL statement or not. |



**DSN_STATEMNT_TABLE description**

The statement table, DSN_STATEMNT_TABLE, contains information about the estimated cost of specified SQL statements. The EXPLAIN process populates the table where QUERYNO is the query number of the SET argument (for dynamic EXPLAIN) or the SQL Statement number of a program/package (for static EXPLAIN while BINDing the package).

Your subsystem or data sharing group can contain more than one table of following schema names (i.e. owners):

- SYSIBM - One instance of each EXPLAIN table can be created with the SYSIBM qualifier. SQL optimization tools use these tables. You can find the SQL statement for creating these tables in member DSNTESC of the SDSNSAMP library.
- userID - You can create additional instances of EXPLAIN tables that are qualified by user ID. These tables are populated with statement cost information when you issue the EXPLAIN statement or bind, or rebind, a plan or package with the EXPLAIN(YES) option. SQL optimization tools might also create EXPLAIN tables that are qualified by a user ID. You can find the SQL statement for creating an instance of these tables in member DSNTESC of the SDSNSAMP library.
- DB2OSCA - SQL optimization tools, such as Optimization Service Center for DB2 for z/OS, create EXPLAIN tables to collect information about SQL statements and workloads to enable analysis and tuning processes. You can find the SQL statements for creating EXPLAIN tables in member DSNTIJOS of the SDSNSAMP library. You can also create this table from the Optimization Service Center interface on your workstation.

| Column | Description |
|---|---|
| QUERYNO | A number that identifies the statement that is being explained. The origin of the value depends on the context of the row:<br>• For rows produced by EXPLAIN statements: The number specified in the QUERYNO clause, which is an optional part of the SELECT, INSERT, UPDATE, MERGE, and DELETE statement syntax. |

| Column | Description |
|---|---|
| | • For rows not produced by EXPLAIN statements: DB2 assigns a number that is based on the line number of the SQL statement in the source program.<br>When the values of QUERYNO are based on the statement number in the source program, values that exceed 32767 are reported as 0. However, in a very long program, the value is not guaranteed to be unique. If QUERYNO is not unique, the value of EXPLAIN_TIME is unique. |
| APPLNAME | The name of the application plan for the row. Applies only to embedded EXPLAIN statements that are executed from a plan or to statements that are explained when binding a plan. A blank indicates that the column is not applicable. |
| PROGNAME | The name of the program or package containing the statement being explained. Applies only to embedded EXPLAIN statements and to statements explained as the result of binding a plan or package. A blank indicates that the column is not applicable. |
| COLLID | The collection ID for the package. Applies only to an embedded EXPLAIN statement that is executed from a package or to a statement that is explained when binding a package. A blank indicates that the column is not applicable. The value DSNDYNAMICSQLCACHE indicates that the row is for a cached statement. |
| GROUP_MEMBER | The member name of the DB2 that executed EXPLAIN, or blank. See the description of the GROUP_MEMBER column in PLAN_TABLE for more information. |
| EXPLAIN_TIME | The time at which the statement is processed. This time is the same as the BIND_TIME column in PLAN_TABLE. |
| STMT_TYPE | The type of statement being explained. |
| COST_CATEGORY | Indicates if DB2 was forced to use default values when making ist estimates. Possible values:<br>A Indicates that DB2 had enough information to make a cost estimate without using default values.<br>B Indicates that some condition exists for which DB2 was forced to use default values. See the values in REASON to determine why DB2 was unable to put this estimate in cost category A. |
| PROCMS | The estimated processor cost, in milliseconds, for the SQL statement. The estimate is rounded up to the next integer value. The maximum value for this cost is 2147483647 milliseconds, which is equivalent to approximately 24.8 days. If the estimated value exceeds this maximum, the maximum value is reported. |
| PROCSU | The estimated processor cost, in service units, for the SQL statement. The estimate is rounded up to the next integer value. The maximum value for this cost is 2147483647 service units. If the estimated value exceeds this maximum, the maximum value is reported. |
| REASON | A string that indicates the reasons for putting an estimate into cost category B.<br>HAVING CLAUSE: A subselect in the SQL statement contains a HAVING clause. |

| Column | Description |
|---|---|
| | HOST VARIABLES: The statement uses host variables, parameter markers, or special registers.<br>REFERENTIAL CONSTRAINTS: Referential constraints of the type CASCADE or SET NULL exist on the target table of a DELETE statement.<br>TABLE CARDINALITY: The cardinality statistics are missing for one or more of the tables that are used in the statement.<br>TRIGGERS: Triggers are defined on the target table of an insert, update, or delete operation.<br>UDF: The statement uses user-defined functions.<br>MATERIALIZATION: Statistics are missing because the statement uses materialized views or nested table expresses. |
| STMT_ENCODE | Encoding scheme of the statement. If the statement represents a single CCSID set, the possible values are:<br>A ASCII<br>E EBCDIC<br>U Unicode<br>If the statement has multiple CCSID sets, the value is M. |
| TOTAL_COST | The overall estimated cost of the statement. This cost should be used only for reference purposes. |



# Customization and Use

## Managing the RLF using Control Tables

You can use resource limit tables to specify resource usage limits for dynamic queries that run in DB2. Resource limit tables allow you to limit the amount of processor resources, in service units, used by a dynamic „S-I-U-M-T-D" SQL statement. You can specify reactive or predictive governing, or use a combination of reactive and predictive limits in the same resource limit table.

For more details, see Chapter „Installation".

You enable the resource limit facility by populating the resource limit tables with rows of data that describe the limits. To insert, update, merge, or delete data in the resource limit table, you need only the usual table privileges on the resource limit table. No higher authorities are required. Use INSERT, UPDATE, MERGE, and DELETE statements to populate the resource limit table.

Changes to the resource limit specification table are immediately effective for all new threads, and for those that have not issued their first „S-I-U-M-T-D" statement. However, if you change the resource limit specification table while a thread is executing, the limit that existed when the thread issued its first dynamic statement applies throughout the life of the thread until DB2 reads in the new limit. DB2 reads in a new limit in the following situations:

- When the application uses a different primary authorization ID.
- When the resource limit facility is stopped and started again.
- When a predictively governed package or DBRM is loaded for execution.

## Calculating the resource limits

The processing time for a particular SQL statement varies according to the processor that executes it, but the number of service units required, a relative metric, remains roughly constant.

The resource limit facility samples processing time in a relative measurement called service units.

A relative metric is used so that you don't have to modify the resource limit table values when processors are changed. The number of service units consumed is not exact between different processors because the calculations for service units are dependent on performance averages measured before the release of a new processor. In some cases, DB2 workloads can differ from the measured averages. In these cases, resource limit table value changes might be necessary.

The easiest way to get SU times for the ASUTIME, RLFASUWARN, or RLFASUERR columns is to use the value in the PROCSU column of DSN_STATEMNT_TABLE as your starting point. You can also get the value from the IFCID 0022 record.

However, if you do not have statement table information, or if you have queries for which you have no information, you can use the following formula to calculate SU time:

```
SU time = processor time × service units per second value
```

The value for service units per second depends on the processor model. You can find this value for your processor model in z/OS MVS Initialization and Tuning Guide, where SRM is discussed.

For example, if processor A is rated at 900 service units per second and you do not want any single dynamic SQL statement to use more than 10 seconds of processor time, you could set ASUTIME as follows:

```
ASUTIME time      = 10 seconds × 900 service units/second
                  = 9000 service units
```

Later, you could upgrade to processor B, which is rated at 1000 service units per second. If the value you set for ASUTIME remains the same (9000 service units), your dynamic SQL is only allowed 9 seconds for processing time but an equivalent number of processor service units:

```
ASUTIME           = 9 seconds × 1000 service units/second
                  = 9000 service units
```

As this example illustrates, after you establish an ASUTIME (or RLFASUWARN or RLFASUERR) for your current processor, you do not need to modify it when you change processors.



## Limiting plans or packages

You can specify limits for the amount of processor resources that are used by a specific group of dynamic queries. You can use a DNRLSTxx resource limit table with values that specify processor limits that are based on the collection, plan name or package name, authorization ID, and the location that is associated with the dynamic queries, and you can specify reactive and predictive governing.

1. Insert rows into a DNRLSTxx resource limit table with values that identify the context of the governed statements, the type of governing, and thresholds for the limits. Governing by plan name and package name are mutually exclusive, meaning that all columns cannot be populated in a valid row:
   - When you specify a plan name, the resource limit facility governs the DBRMs in the MEMBER list specified on the BIND PLAN command. The RLFFUNC, RLFCOLLN, and RLFPKG columns must contain blanks.
   - When you specify a package name, the resource limit facility governs the packages used during the execution of the SQL application program. PLANNAME must contain blank, and RLFFUNC must contain '2'.

2. Issue the START RLMIT ID=xx command, where xx is the two character identifier that you specified when you created the table. You can start and stop different resource limit tables at different times. However, only one resource limit table of each type (DNSRLMTxx or DSNRLSTxx) can run at any one time.

DB2 uses the following search order:

1. Exact match
2. Authorization ID
3. Plan name, or collection name and package name
4. LU name
5. No row match

When no row in the resource limit table matches the currently executing statement, DB2 uses the default value that is set in the RLST ACCESS ERROR field of installation panel DSNTIPO (for queries that originate locally) or DSNTIPR (for queries that originate remotely). This default limit applies to reactive governing only. For predictive governing, when no row matches, no predictive governing occurs.

## Limiting resource usage of clients and middleware servers

You can limit the amount of processor resources that are used by a specific group of dynamic queries from middleware servers. You can use a DSNRLMTxx resource limit tables to specify processor limits that are based on client information, such as the application name, end user ID, workstation ID, and IP address, associated with dynamic queries that run on middleware servers, and you can specify reactive and predictive governing.

1. Insert rows into a DSNRLMTxx resource limit table with values that identify the statements to limit, the type of governing, and the thresholds of the limits.
2. Issue the START RLMIT ID=xx command, where xx is the two character identifier that you specified when you created the table. You can start and stop different resource limit tables at different times. However, only one resource limit table of each type (DNSRLMTxx or DSNRLSTxx) can run at any one time.

DB2 uses the following search order:

1. Exact match
2. Application name
3. User ID
4. Workstation name
5. IP address
6. No row match

When no row in the resource limit table matches the currently executing statement, DB2 uses the default set on the RLST ACCESS ERROR field of installation panel DSNTIPO (for queries that originate locally) or DSNTIPR (for queries that originate remotely). This default limit applies to reactive governing only. For predictive governing, if no row matches, no predictive governing occurs.

## Limiting resource usage of Remote Requestors

You can create an RLMT to govern by client information (RLFFUNC=8 and RLFFUNC=9) such as application name, user ID, workstation ID, and IP address.

If you use an RLST, you must govern by package name (RLFFUNC=2 or RLFFUNC=7), which means that PLANNAME must be blank.

Specify the originating system's LU name in the LUNAME column, or specify PUBLIC for all remote LUs. If you leave LUNAME blank, DB2 assumes that you mean the local location only and none of your incoming distributed requests qualify.

Important:

- If dynamic statements come from requesters that use TCP/IP, you cannot specify the LU name. You must use PUBLIC.
- If no row is present in the resource limit table to govern access from a remote location: The limit is the default set on the RLST ACCESS ERROR field of installation panel DSNTIPR (see sectio below).

## Using the resource limit facility to disable parallelism

Using the resource limit facility to disable parallelism: If you use the resource limit facility (RLF), you can disable I/O parallelism, CP parallelism, or Sysplex parallelism for static or dynamic SQL statements by specifying the RLST values shown in Table 35. You should use RLF to disable parallelism for static statements only for diagnosis.

| RLFFUNC | RLFBIND | Effect |
|---|---|---|
| 3 | blank | Disable I/O parallelism for dynamic statements |
| 3 | S | Disable I/O parallelism for static plans or packages |
| 3 | A | Disable I/O parallelism for static and dynamic statements |
| 4 | blank | Disable CP parallelism for dynamic statements |
| 4 | S | Disable CP parallelism for static plans or packages |
| 4 | A | Disable CP parallelism for static and dynamic statements |
| 5 | blank | Disable Sysplex parallelism for dynamic statements |
| 5 | S | Disable Sysplex parallelism for static plans or packages |
| 5 | A | Disable Sysplex parallelism for static and dynamic statements |

These RLST values do not affect the way RLF governs the processor time for dynamic statements. To use RLF to change the parallelism mode for a plan or package, you must rebind the plan or package while RLF is active. For data sharing, RLF must be active on the DB2 member where the rebind occurs.

## SQL Codes

### +495

```
ESTIMATED PROCESSOR COST OF estimate-amount1 PROCESSOR SECONDS (estimate-
amount2 SERVICE UNITS) IN COST CATEGORY cost-category EXCEEDS A RESOURCE
LIMIT WARNING THRESHOLD OF limit- amount SERVICE UNITS
```

Explanation

The prepare of a dynamic INSERT, UPDATE, MERGE, DELETE, or SELECT SQL statement resulted in a cost estimate that exceeded the warning threshold value specified in the resource limit specification table (RLST). This warning is also issued if DB2's cost category value was "B", and the default action specified in the RLF_CATEGORY_B column in the RLST is to issue a warning.

*estimate_amount1*

The cost estimate (in processor seconds) if the prepared INSERT, UPDATE, MERGE, DELETE, or SELECT statement were to be executed.

*estimate_amount2*

The cost estimate (in service units) if the prepared INSERT, UPDATE, MERGE, DELETE, or SELECT statement were to be executed.

*cost-category*

DB2's cost-category for this SQL statement. The possible values are A or B.

*limit-amount*

The warning threshold (in service units) specified in the RLFASUWARN column of the RLST. If you entered any negative number for the RLFASUWARN column, the value for limit-amount defaults to zero.

System action

The prepare of the dynamic INSERT, UPDATE, MERGE, DELETE, or SELECT statement was successful. An SQLCODE -905 might be issued if the execution of the prepared statement exceeds the ASUTIME value specified in the RLST.

User response

If the warning is caused by an SQL statement that is consuming too much processor resource, attempt to rewrite the statement to perform more efficiently. Another option is to ask the administrator to increase the warning threshold value in the RLST.

Programmer response

Ensure that there is application logic to handle the warning to either allow the statement to execute or to stop the statement from being executed. If this SQLCODE was returned because the cost category value is "B", it might be that the statement is using parameter markers or that some statistics are not available for the referenced tables and columns. Make sure the administrator has run the utility RUNSTATS on the referenced tables. It might also be that UDFs will be invoked when the statement is executed, or for INSERT, UPDATE, MERGE, or DELETE statements that triggers are defined on the changed table. Check the DSN_STATEMNT_TABLE or the IFCID 22 record for this statement to find the reasons this SQL statement has been put in cost category "B".

SQLSTATE

01616

---

**-495**

```
ESTIMATED PROCESSOR COST OF estimate-amount1 PROCESSOR SECONDS (estimate-
amount2 SERVICE UNITS) IN COST CATEGORY cost-category EXCEEDS A RESOURCE
LIMIT ERROR THRESHOLD OF limit- amount SERVICE UNITS
```

Explanation

The prepare of a dynamic INSERT, UPDATE, MERGE, DELETE, or SELECT SQL statement resulted in a cost estimate that exceeded the error threshold value specified in the resource limit specification table (RLST). This error is also issued if DB2's cost category value was "B", and the default action specified in the RLF_CATEGORY_B column in the RLST is to issue an error.

*estimate_amount1*

The cost estimate (in processor seconds) if the prepared INSERT, UPDATE, MERGE, DELETE or SELECT statement were to be executed.

*estimate_amount2*

The cost estimate (in service units) if the prepared INSERT, UPDATE, MERGE, DELETE or SELECT statement were to be executed.

*cost-category*

DB2's cost-category for this SQL statement. The possible values are A or B.

*limit-amount*

The error threshold (in service units) specified in the RLFASUERR column of the RLST. If you entered any negative number for the RLFASUERR column, the value for limit-amount defaults to zero.

System action

The prepare of the dynamic INSERT, UPDATE, MERGE, DELETE, or SELECT statement was unsuccessful.

User response

If the warning is caused by an SQL statement that is consuming too much processor resource, attempt to rewrite the statement to perform more efficiently. Another option is to ask the administrator to increase the error threshold value in the RLST.

Programmer response

If this SQLCODE was returned because the cost category value is "B", it might be that the statement is using parameter markers or that some statistics are not available for the referenced tables and columns. Make sure the administrator has run the utility RUNSTATS on the referenced tables. It might also be that UDFs will be invoked when the statement is executed, or for INSERT, UPDATE, MERGE, or DELETE statements that triggers are defined on the changed table. Check the DSN_STATEMNT_TABLE or the IFCID 22 record for this statement to find the reasons this SQL statement has been put in cost category "B". If the program cannot be changed, or if statistics cannot be obtained, ask the administrator to change the value in the RLF_CATEGORY_B column in the RLST to "Y" which allows the statement to execute or "W" which returns a warning instead of an error.

SQLSTATE

57051

---

**-905**

```
UNSUCCESSFUL EXECUTION DUE TO RESOURCE LIMIT BEING EXCEEDED, RESOURCE NAME
= resource-name LIMIT = limit-amount1 CPU SECONDS (limit-amount2 SERVICE
UNITS) DERIVED FROM limit-source
```

Explanation

The execution of the SQL statement was terminated because a resource limit was exceeded.

*resource-name*

The name of the resource whose limit was exceeded. It is also the name of the column in the DB2® table from which the limit was derived. The resource-name can be ASUTIME, which is the number of CPU seconds permitted for each SQL statement.

*limit-amount1*

The maximum number of CPU seconds permitted

*limit-amount2*

The maximum number in service units permitted

*limit-source*

The source used to derive the limit-amount: the name of a resource limit specification table, a system parameter, or the SYSIBM.SYSROUTINES catalog table. If the source is a system parameter, the resource limit specification table did not contain an applicable entry or an error occurred while accessing the table.

System action

If the limit-source was a resource limit specification table or a system parameter, the execution of this SQL statement is terminated. A trace record containing more detailed information about this failure is generated. If an SQL cursor is associated with the failed instruction, its position is unchanged and a CLOSE or PREPARE statement can be issued. If any other operation is attempted with the cursor, it cannot be executed and SQLCODE -905 is returned. If there is no cursor, this statement was rolled back.

Programmer response

Determine why this SQL statement or stored procedure took so long and take appropriate action. Consider simplifying the SQL statement, restructuring tables and indexes, or contacting the installation group responsible for maintaining the resource limit specification tables.

If the limit-source was a resource limit specification table or a system parameter, the application program that receives this return code can execute additional SQL statements.

SQLSTATE

57014

## Examples

(A) Combination of restrictive and predictive governing:

Limit local and remote access to 7.5 MSU's, issue a +495 warning with 3.5 MSU's or more, allow category B queries but issue +495 warning. Terminate requests with more than 7.5 MSU's.

```
AUTHID  PLANNAME  ASUTIME  LUNAME    RLFFUNC  ...  RLFASUERR  RLFASUWARN  RLF_CATEGORY_B
------  --------  -------  --------  -------  ...  ---------  ----------  --------------
USER1             7500000  PUBLIC    2        ...  (null)     (null)
USER1             (null)   PUBLIC    7        ...  7500000    3500000     W
```

(B) Combination of restrictive and predictive governing, allow category B:

Limit local and remote access to 7.5 MSU's, issue a +495 warning with 3.5 MSU's or more, allow category B queries but terminate every request with more than 7.5 MSU's.

```
AUTHID  PLANNAME  ASUTIME  LUNAME    RLFFUNC  ...  RLFASUERR  RLFASUWARN  RLF_CATEGORY_B
------  --------  -------  --------  -------  ...  ---------  ----------  --------------
USER2             7500000  PUBLIC    2        ...  (null)     (null)
USER2             (null)   PUBLIC    7        ...  7500000    3500000     Y
```

(C) Predictive governing:

Limit local and remote access to 7.8 MSU's. For remote users using packages of NULLID collection (i.e. DB2 Connect) issue a +495 warning with 7.5 MSU's or more and terminate statements which would you more then 7.8 MSU's.  Allow category B statements of NULL packages but issue a +495 warning.

```
AUTHID    ASUTIME  LUNAME    RLFFUNC  RLFCOLLN      RLFASUERR  RLFASUWARN  RLF_CATEGORY_B
--------  -------  --------  -------  ------------  ---------  ----------  --------------
USER3     (null)   PUBLIC    7                      7842900    (null)
          (null)   PUBLIC    7        NULLID        7842900    7500000     W
```