Esgyn Corporation

EsgynDB Control Query Default (CQD) Reference Manual

Published: March 2018
Edition: EsgynDB Release 2.4.0

Contents

Legal Notice

The information contained herein is subject to change without notice. This documentation is distributed on an “AS IS” basis, without warranties or conditions of any kind, either express or implied. Nothing herein should be construed as constituting an additional warranty. Esgyn Corporation shall not be liable for technical or editorial errors or omissions contained herein.

NOTICE REGARDING OPEN SOURCE SOFTWARE: Project Trafodion is licensed under the Apache License, Version 2.0 (the "License"); you may not use software from Project Trafodion except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/ licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Acknowledgements

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation. Java® and MySQL® are registered trademarks of Oracle and/or its affiliates. Bosun is a trademark of Stack Exchange Inc. Apache®, Hadoop®, HBase®, Hive®, Zookeeper, ORC, openTSDB®, Sqoop®, and Trafodion® are trademarks of the Apache Software Foundation. Esgyn and EsgynDB are trademarks of Esgyn Corporation.

1 About This Document

This manual describes information about CONTROL QUERY DEFAULT options supported in EsgynDB.

1.1 Intended Audience

This manual is intended for database administrators and application programmers for either tuning performance of the database, or for debugging purposes.

You should be familiar with structured query language (SQL) and with the American National Standard Database Language SQL:1999.

1.2 New and Changed Information

This edition includes the following changes

Change	Location
Added EsgynDB specific CQDs	Multiple sections

2 Introduction

Refer to the EsgynDB SQL Reference Manual for a full documentation of the CQD (CONTROL QUERY DEFAULT) statement.

The CONTROL QUERY DEFAULT statement changes the default settings for the current process. You can execute the CONTROL QUERY DEFAULT statement in a client-based tool like TrafCI or through any ODBC or JDBC application.

2.1 Syntax

```
{ CONTROL QUERY DEFAULT | CQD } control-default-option
```

`control-default-option` is:

```
attribute {'attr-value' | RESET}
```

The result of the execution of a CONTROL QUERY DEFAULT statement stays in effect until the current process terminates or until the execution of another statement for the same attribute overrides it. CQDs are applied at compile time, so CQDs do not affect any statements that are already prepared. For example:

```
PREPARE x FROM SELECT * FROM t;
CONTROL QUERY DEFAULT SCHEMA 'myschema';
EXECUTE x;                              -- uses the default schema SEABASE
SELECT * FROM t2;                       -- uses MYSCHEMA;
PREPARE y FROM SELECT * FROM t3;
CONTROL QUERY DEFAULT SCHEMA 'seabase';
EXECUTE y;                              -- uses MYSCHEMA;
```

2.2 Examples

- Change the maximum supported length of the column names to 200 for the current process:

  ```
  CONTROL QUERY DEFAULT HBASE_MAX_COLUMN_NAME_LENGTH '200';
  ```

- Reset the HBASE_MAX_COLUMN_NAME_LENGTH attribute to its initial value in the current process:

  ```
  CONTROL QUERY DEFAULT HBASE_MAX_COLUMN_NAME_LENGTH RESET;
  ```

2.3 CQD Descriptions

The following information is provided for each CQD:

Description	Describes the purpose of the CQD.
Values	Identifies this information: • Values, in the form of a character string, that specify the applicable attribute values for the CQD. • The default attribute value. • If applicable, the EsgynDB release in which the attribute values or default changed.

Usage	Describes the conditions when the CQD is helpful, and how to detect the conditions.
Production Usage	Identifies when the CQD is not safe to be used as a permanent setting in production.
Impact	Describes any positive and negative implications of using the CQD.
Level	Indicates one of these levels at which the CQD should be used: • Query • Session • Service • Any NOTE: This level indicates that the CQD can be used at the Query, Session or Service level as long as you fully understand the scope of the impact of the CQD.
Conflicts/Synergies	Describes CQDs that are in conflict with or can be used in conjunction with the CQD.
Real Problem Addressed	Describes any design or solution that the CQD may be a workaround for and how you can directly address the real problem.
Introduced In Release	Indicates the EsgynDB release when the CQD was introduced.
Deprecated In Release	Indicates the EsgynDB release the CQD was deprecated.

3 Query Plans

This section describes CQDs that are used to influence query plans.

3.1 DEFAULT_DEGREE_OF_PARALLELISM

Description	Defines the minimum size of the adaptive segment – the number of processors available for query operator parallelism. The optimizer may choose an adaptive segment size that is equal to, or the multiple of, the value of this CQD, depending on the maximum estimated resource consumed by any single operator in the query. The optimizer may also decide to run the query with no parallelism if the resource consumption estimate is very low.
Values	Unsigned Integer Default: '16'
Usage	For systems running at higher levels of concurrency with workloads that include a large number of small queries, reducing the default degree of parallelism may help achieve higher throughput. With the default of 16, for 32p systems, adaptive segmentation can use two 16p virtual segments to execute queries that do not require a degree of parallelism of 32. This default setting can, for example, be changed to 8 for a 16p system, to allow adaptive segmentation to leverage a lower degree of parallelism.
Production usage	Yes
Impact	Lowering the value of this CQD can increase the throughput of high concurrency small query workloads, but has the potential disadvantage of increasing the elapsed time for some of the longer running queries that leverage adaptive segmentation.
Level	System. There may be scenarios where you want to influence the degree of adaptive segmentation parallelism only for a certain set of queries and use it at the service level.
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.2 GEN_HSHJ_MIN_MAX_OPT

Description	Use Min Max optimization for hash joins. The smaller table of hash join is read first, min and max value of joining column computed, synthesized predicates that limit scan of the bigger table to be within this min-max range are pushed down to the scan during execution. Applies to both Trafodion and Hive tables.
Values	ON | OFF Default is OFF
Usage	Enable and check if feature has been used by looking for the term "min_max" in full explain output.
Production usage	Yes
Impact	Query performance
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

3.3 GROUP_OR_ORDER_BY_EXPRESSION

Description	Allows expressions to be specified in GROUP BY or ORDER BY clause of elements that appear in the SELECT list. The expressions do not have to match the element in the SELECT exactly, but could be monotonic functions of them.
Values	ON/OFF
Usage	Default is ON.
Production usage	Yes
Impact	Allows such syntax
Level	Session
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

3.4 HASH_JOINS

Description	Determines if Hash Join will be considered by the optimizer to generate an execution plan
Values	'ON': Hash Join is considered 'OFF': Hash Join is disabled Default: 'ON'
Usage	Use this CQD when you want to force a query plan not to use any Hash Joins.
Production usage	Hash Join is an important join implementation strategy for most BI queries. It is highly recommended not to turn Hash Join OFF. It should only be used to force a query plan for a particular query on an exception basis.
Impact	Turning Hash Join OFF may result in very inefficient query plans with expensive nested joins or sorts for merge joins.
Level	Query
Conflicts/Synergies	Avoid turning all the 3 join implementations OFF (Hash Joins, Nested Joins, and Merge Joins). This may result in the compiler failing to generate query plans.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.5 HBASE_COPROCESSORS

Description	Allow HBase coprocessors to be used when computing aggregates
Values	ON / OFF Default: 'ON'
Usage	Enables EsgynDB to use HBase coprocessors to do early aggregation and filtering at the HBase Region Server level. This attribute does not affect Transaction coprocessors used by EsgynDB. Only COUNT(*) queries will be affected by this attribute.
Production usage	Yes
Impact	Network traffic between Region Server and Trafodion processes is reduced, but Region Server can become very busy when aggregating over large tables.
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0

Deprecated in Release	

3.6 HIVE_NUM_ESPS_PER_DATANODE

Description	Determines number of ESP processes that will be used to scan a Hive table, per datanode In a mulltitenant environment, this CQD determines the number of ESP processes that will be used to scan a Hive table, per tenant Cube.
Values	Positive integer Default: '2'
Usage	Use this attribute to increase or decrease the number of scanners that will process a single Hive table. If a Hive scan is found to be the bottleneck for a particular query, increasing this attribute to say 4 or higher will help. On the other hand decreasing the attribute to 1 could help with workload management.
Production usage	Yes
Impact	Controls number of processes and therefore affects query execution time and system workload.
Level	Query
Conflicts/Synergies	The attribute HIVE_MIN_BYTES_PER_ESP_PARTITION (default = 67108864) may need to be adjusted downward when this attribute is used to increase the parallelism of scanning smaller Hive tables.
Real Problem Addressed	
Introduced in Release	EsgynDB R2.1
Deprecated in Release	

3.7 INTERSECT_PUSH_GRBY_RATIO

Description	Query transformation for INTERSECT clause that will push groupby used by intersect below the join. This is a cardinality estimate based transformation. Compare the cardinality of a 1:n join (the max of the child cardinalities) with the actual cardinality of the join representing the intersect. If the actual cardinality is more than <ratio> times higher that that of a 1:n join, then push groupby below join.
Values	A float value
Usage	< 0: Never make the transformation 0: Always make the transformation > 0: Make the transformation based on the cardinality ratio Default is -1.0
Production usage	Yes
Impact	Improve performance of certain INTERSECT queries
Level	Session
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

3.8 JOIN_ORDER_BY_USER

Description	Enables or disables the join order in which the optimizer joins the tables to be the sequence of the tables in the FROM clause of the query
Values	'ON': Join order forced

	'OFF': Join order decided by the optimizer Default: 'OFF'
Usage	When set to ON, the optimizer considers only execution plans that have the join order matching the sequence of the tables in the FROM clause
Production usage	This setting is to be used only for forcing a desired join order that was not generated by default by the optimizer. It can be used as a workaround for query plans with inefficient join order.
Impact	Since you are in effect forcing the optimizer to use a plan that joins the table in the order specified in the FROM clause, the plan generated may not be the optimal one
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.9 MAX_ESPS_PER_CPU_PER_OP

Description	Defines the number of ESPs per core and helps the computation of maximal degree of parallelism (DoP). In a multitenant environment, the maximal DoP is computed as the ceiling of CQD * cores per tenant cube[1] (4) x total number of tenant cubes per node[2] x total number of nodes in a cluster. For example, in a tenant with 6 tenant cubes per node in a 12-node system, and the CQD is set to 0.2, the maximum DoP = ceil (0.2 x 4 x 6 x 12) = 58. In a non-tenant environment, the maximal DoP is computed as the ceiling of CQD x cores per node x total number of nodes. For example, in a cluster with 40 cores per node, 12 nodes, and the CQD is set to 0.2, then the maximum DoP = ceil(0.2 x 40 x 12) = 96. If this CQD is not set, the maximum DoP is the ceiling of (2/(cores per node)) x cores per node x number of nodes. For the above 12-node cluster, maximum DoP = 2 x 12 = 24.
Values	Default value is 2 / number of cores in a physical node.
Usage	If the DoP is too small for a heavy query, specifying a larger value can help the query by allowing more degree of parallelism.
Production usage	Yes
Impact	Affects degree of parallelism for most of the queries directly.
Level	Query Performance
Conflicts/Synergies	DEGREE_OF_PARALLELISM
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

[1] Tenant cube: A tenant cube is a slice of computation capability in a cluster. A Cube is defined to be a virtual machine with 4 cores, 32 GB memory. A set of 4 Cubes together define a tenant Unit.

[2] In R2.3 and R2.4, the product of "cores per tenant cube" and "total number of tenant cubes per node" can be configured by the CQD AS_TENANT_UNIT_CORES.

3.10 MC_SKEW_SENSITIVITY_THRESHOLD

Description	Define the multi-column skew sensitivity threshold T used by multi-column skew-insensitive hash join (Skew Buster) Let f be the occurrence frequency of a skew value v, DoP be the degree of parallelism of a hash join operator, and RC be the row count of the source data (e.g., fact table) where the skew originates. The hash join will run in the anti-skew mode for v if f >= T * DoP / RC.
Values	< 0: disable the multi-column skew buster >= 0: define the threshold T Default value: 0.1.
	Use of a negative value to disable multi-column anti-skew hash joins. This may slow down query performance if multi-column skew values are present in the fact table. Use of a value 0 will treat every multi-column value as skew values. This may increase network traffic since skewed values are broadcasted from the inner side child of the hash join to all join processes. Use of a value > 0 will select those multi-column values as skewed values if their occurrence frequencies are high enough.
Usage	Please consult with community.
Production usage	Run-time performance
Impact	Compiler instance
Level	
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.11 MDAM_NO_STATS_POSITIONS_THRESHOLD

Description	This CQD effects the behavior of the query optimizer when there are no statistics available for a query having range predicates on key columns. The optimizer calculates the worst case number of seeks that the MDAM access method would do if chosen for the query. If this number is greater than the value of MDAM_NO_STATS_POSITIONS_THRESHOLD, then MDAM will not be considered for the query execution plan.
Values	Any integer greater than equal to zero. The default for this CQD is 10.
Usage	In certain situations, queries on tables lacking statistics may not be optimal because MDAM was not chosen. Increasing the value for this CQD will allow MDAM to be chosen in more cases. On the other hand, if the value is made too high and the worst case scenario actually occurs, an MDAM plan may perform poorly.
Production usage	Exercise caution when increasing this CQD in a production environment.
Impact	Table scans on tables lacking statistics may improve by varying the value of this CQD. Results will vary depending on the actual data in the table and the semantics of the

	query.
Level	Query
Conflicts/Synergies	If MDAM_SCAN_METHOD is set to 'OFF', this CQD will have no effect.
Real Problem Addressed	Perform UPDATE STATISTICS on the table (at the very least on key columns) to obtain statistics.
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.12 MDAM_SCAN_METHOD

Description	Enables or disables the Multi-Dimensional Access Method
Values	'ON': MDAM considered 'OFF': MDAM disabled Default: 'ON'
Usage	In certain situations, the optimizer might choose MDAM inappropriately, causing poor performance. In such situations you may want to turn MDAM OFF for the query it is effecting.
Production usage	
Impact	Table scans with predicates on non-leading clustering key column(s) could benefit from MDAM access method if the leading column(s) has a small number of distinct values. Turning MDAM off will result in a longer scan time for such queries.
Level	This CQD should be set mostly at the query level when MDAM is not working efficiently for a specific query. However, there may be cases (usually a defect) where a larger set of queries is being negatively impacted by MDAM. In those cases you may want to set it at the service or system level.
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.13 MERGE_JOINS

Description	Determines if Merge Join will be considered by the optimizer to generate an execution plan.
Values	'ON': Merge Join is considered 'OFF': Merge Join is disabled Default: 'ON'
Usage	Use this CQD when you want to force a query plan not to use Merge Joins. This is useful as a workaround for query plans with very expensive sorts for Merge Joins. Turning Merge Join OFF also has the advantage of reducing the query compile time.
Production usage	Merge Join is an efficient join implementation strategy if the physical schema was designed to take advantage of it (i.e. large tables are physically ordered based on the most frequently joined column(s)).
Impact	Turning Merge Join OFF may result in the optimizer not considering potentially efficient query plans, for queries with large joins on tables that are physically ordered by the join column(s). Turning Merge Join ON causes an increase in compile time since the optimizer now has to consider many more join options.
Level	This CQD should be set mostly at the query level when a Merge Join is not working efficiently for a specific query. However, there may be cases (usually a defect) where a larger set of queries is being negatively impacted by Merge Joins. In those cases you may want to set it at the service or system level.
Conflicts/Synergies	Avoid turning all the 3 join implementations OFF (Hash Joins, Nested Joins, and Merge Joins). This may result in the compiler failing to generate query plans.
Real Problem Addressed	

Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.14 NESTED_JOINS

Description	Determines if Nested Join will be considered by the optimizer to generate an execution plan.
Values	'ON': Nested Join is considered 'OFF': Nested Join is disabled Default: 'ON'
Usage	Use this CQD when you want to force a query plan not to use Nested Joins. This is useful as a workaround for query plans with very expensive Nested Joins, which may occur if the optimizer fails to estimate the cost of a Nested Join correctly.
Production usage	Nested Join is an important join implementation strategy for many BI queries. It is recommended not to turn Nested Join OFF. It should only be used to force a query plan for a particular query on an exception basis.
Impact	Turning Nested Join OFF may result in inefficient query plans for certain type of queries, such as light workloads and star join queries.
Level	Query
Conflicts/Synergies	Avoid turning all the 3 join implementations OFF (Hash Joins, Nested Joins, and Merge Joins). This may result in the compiler failing to generate query plans.
Real Problem Addressed	The problem of inefficient Nested Joins can be better handled using a higher degree of query plan robustness as set by the ROBUST_QUERY_OPTIMIZATION CQD
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.15 OPTIMIZATION_LEVEL

Description	Controls the optimizer resources and time spent for optimizing a query plan, with level 0 indicating the least amount of optimization effort and level 5 indicating the most. Lower optimization levels produce lower plan quality with minimal compile time, while higher optimization levels cause the compiler to spend more compilation time to produce better plan quality.
Values	'0', '2', '3', '5' Default: '3'
Usage	Reduce the optimization level when compile time is longer than desired and queries have relatively small execution cost and are simple in structure.
Production usage	This CQD is to be used only as a workaround for queries with unacceptable compile time or plan quality.
Impact	Lowering the optimization level below the system default may result in inefficient query execution plans. Increasing the optimization level over the system default may result in very high compile time for complex queries.
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.16 ORC_PRED_PUSHDOWN

Description	Push predicates on an ORC scan down to the ORC subsystem. The ORC subsystem will return only strides (a contiguous collection of rows, with a default size of 10,000) that

	match the predicate. On a sorted column such a pushdown can reduce the files/stripes ORC subsystem has to access significantly. On unsorted columns pushown will reduce the amount of data transferred across JNI between Orc and EsgynDB.
Values	ON | OFF Default is ON
Usage	Turn OFF only if a bug causes failures during ORC scan. It can be determined if this is necessary from Java stack shown in SQL error message.
Production usage	Yes
Impact	Improve query performance
Level	System
Conflicts/Synergies	Required for GEN_HSHJ_MIN_MAX_OPT on Orc tables
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

3.17 ORC_AGGR_PUSHDOWN

Description	Push aggregates such as count, sum, min and max to the Orc subsystem. No groupby operator is required in EsgynDB to compute the aggregate. Orc susbsystem will not read all rows to compute the aggregate.
Values	ON | OFF Default is ON
Usage	Turn OFF only if a bug causes failures during Orc scan. It can be determined if this is necessary from Java stack shown in SQL error message.
Production usage	Yes
Impact	Improve query performance
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

3.18 PARALLEL_NUM_ESPS

Description	Maximum number of parallel ESPs that work on a particular operation, like a join. In a multitenant environment, defines the maximum number of parallel ESPs for the tenant Cube, regardless of any other ESP related CQD settings.
Values	Unsigned integer: The maximum number of ESPs that should be used for a particular operation. The value must be less than the number of CPUs in the cluster. 'SYSTEM': Compiler calculates the number of ESPs to be used Default: 'SYSTEM'
Usage	Used to control the maximum degree of parallelism for a query. This could be useful to limit the number of resources (CPU and memory) any single query can use.
Production usage	Yes
Impact	Lowering the value of this CQD can increase the throughput of high concurrency small and medium query workloads, but has the potential disadvantage of increasing the elapsed time of some of the long running queries.

Level	Service
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.19 PARQUET_NJS

Description	This CQD controls a performance feature that allows nested loop join into a Parquet fact table.
Values	ON / OFF Default: 'OFF'
Usage	Set the CQD to 'ON' to turn on the feature. Use this feature when the other join source (e.g., a dimension table) can provide limited number of rows (say in 100) to probe the Parquet fact table, the selectivity on the join column on the Parquet fact table is high, and optionally the join column on the Parquet fact table is sorted.
Production usage	Yes
Impact	The join type selected for Parquet tables.
Level	Query Performance
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.4
Deprecated in Release	

3.20 PARQUET_NJS_PROBES_THRESHOLD

Description	This CQD specifies a threshold value for the estimated maximal number of probes that a nested loop join into a Parquet fact table should be considered.
Values	1 .. n Default: 150
Usage	When the estimated number of rows is less than or equal to this threshold, the nested loop join is considered by the optimizer when join to a Parquet table. Use this CQD to disable nested joins that may have too many probes.
Production usage	Yes
Impact	The nested loop join into Parquet table.
Level	Query Performance
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.4
Deprecated in Release	

3.21 RISK_PREMIUM_NJ

Description	Influences the optimizer to choose other types of joins over nested joins, by making a nested join plan relatively more expensive
Values	Any positive fractional value Default: '1.2'
Usage	Review ROBUST_QUERY_OPTIMIZATION first before considering the use of this CQD. The default setting indicates that a nested join plan must be 20% cheaper before it

	is allowed to win over competing safer (hash) join plans. A setting of 1.0 means no handicap for nested joins. A setting of 5.0 means a nested join must be 400% cheaper before it is allowed to win over competing hash join plans. If it is determined that the optimizer is using nested joins often enough where these plans are resulting in poor performance, this CQD may be used to influence the optimizer to consider another join instead, such as a hash join, in some of those cases. Certainly, NESTED_JOINS OFF could turn nested joins off completely. However, there are many cases where nested joins do provide better performance than hash joins, and turning them off completely may negatively impact the performance of queries that can do a lot better with nested joins.
Production usage	
Impact	Specifying a risk premium buys insurance against nested joins being chosen when they should not have been. However, this can also result in nested joins not being chosen where the cardinality estimation was in fact accurate and a nested join could have performed better. So this setting should be used with care in order to get robustness with a net gain in performance.
Level	Any. There may be cases where there are different applications or workloads that might benefit from this CQD more than other workloads. In such cases this could be used at the Service level.
Conflicts/Synergies	ROBUST_QUERY_OPTIMIZATION is a CQD that provides a robust query setting across the board, influencing the nested join risk premium as well. Customers are advised to use that setting instead to influence plans, unless they are specifically addressing nested join issues and need to use this setting independent of that CQD.
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

3.22 RISK_PREMIUM_SERIAL

Description	Influences the optimizer to choose a parallel plan over a serial plan, by making a serial plan relatively more expensive
Values	Any positive fractional value Default: '1.2'
Usage	Review ROBUST_QUERY_OPTIMIZATION first before considering the use of this CQD. The default setting means that a serial plan must be 20% cheaper before it is allowed to win over competing parallel plans. A setting of 1.0 means no handicap for serial plans. A setting of 2.0 means a serial plan must be 100% cheaper before it is allowed to win over competing parallel plans. If it is determined that the optimizer is using serial plans often enough where these plans are resulting in poor performance, this CQD may be used to influence the optimizer to consider parallel plans instead in some of those cases. Certainly, ATTEMPT_ESP_PARALLELISM MAXIMUM could turn serial plans off completely. However, there are many cases where serial plans do provide better performance than parallel plans, and turning them off completely may negatively impact the performance of queries that can do a lot better with serial plans.
Production usage	
Impact	Specifying a risk premium buys insurance against serial plans being chosen when they should not have been. However, this can also result in serial plans not being chosen where the cardinality estimation was in fact accurate and a serial plan could have performed better. So this setting should be used with care in order to get robustness

	with a net gain in performance.
Level	Any. There may be cases where there are different applications or workloads that might benefit from this CQD more than other workloads. In such cases this could be used at the Service level.
Conflicts/Synergies	ROBUST_QUERY_OPTIMIZATION is a CQD that provides a robust query setting across the board, influencing the serial plan risk premium as well. Customers are advised to use that setting instead to influence plans, unless they are specifically addressing serial plan issues and need to use this setting independent of that CQD.
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

3.23 RISK_PREMIUM_SERIAL_SCALEBACK_MAXCARD_THRESHOLD

Description	Define the minimal estimated max cardinality or row count of any relational operators in a query above which the risk premium for serial plan will be applied. A serial query plan is favored by the compiler when it estimates the query reads and processes small amount of data. The estimation error could become large when some operator is calculated to produce many rows, and thus the serial plan is not optimal. This CQD helps prevent utilizing serial plan in such cases.
Values	An unsigned integer value.
Usage	Adjust this CQD only when necessary. Use of a value smaller than the default (10,000) to penalize more serial plans or favor more parallel plans for operators produce less number of rows. Use of a larger value otherwise.
Production usage	Please consult with community.
Impact	Plan quality
Level	Compiler instance
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

3.24 ROBUST_QUERY_OPTIMIZATION

Description	Is intended to provide a simpler way to influence the optimizer's choice of query plans. The optimizer chooses query plans based on cardinality estimates (the number of result rows estimated at each step of a query execution plan). Actual cardinalities encountered at query execution often differ from estimates. Just like an insurance actuary, the optimizer considers risky choices and exacts risk premiums before it chooses a plan that is inherently sensitive to cardinality estimation errors. Risky choices include: • Nested joins: Can be excellent data reducers but they can also result in extremely long running queries when their outer table cardinality is grossly underestimated. • Serial plans: Are an excellent choice because they use the least resources when processing low data volumes. But, they can also result in very long running queries when actual cardinalities greatly exceed estimates. • Complete sharing of ESP partitioning schemes: Correct parallel processing requires

	partitioning the data across ESP instances using a partitioning scheme usually based on the largest table's partitioning keys, join columns, and group by columns. Complete sharing of ESP partitioning schemes minimizes the overhead of runtime repartitioning. But, it can also result in very long running queries if the "least common denominator" partitioning scheme results in a few active ESPs doing most of the work. This can happen when repartitioning is being done on a very low unique entry count attribute e.g. gender. ROBUST_QUERY_OPTIMIZATION can be used to influence the premiums associated with these risky plans and thereby overall plan quality and performance for your specific workloads.
Values	'MIN': No risk premium 'HIGH' & 'MAXIMUM': Higher levels of risk premium 'SYSTEM': Safe risk premium Default: 'SYSTEM'
Usage	MAXIMUM tells the optimizer to make the safest choice of query plans. This means: • RISK_PREMIUM_NJ is set to 5.0 -- nested join must be 400% cheaper before it can win over competing (hash) join plans • RISK_PREMIUM_SERIAL is set to 2.0 -- serial plan must be 100% cheaper before it can win over competing parallel plans • PARTITIONING_SCHEME_SHARING is set to 2 -- no partition scheme sharing between adjacent ESP fragments HIGH tells the optimizer to make a safer choice of query plans. This means: • RISK_PREMIUM_NJ is set to 2.5 -- nested join must be 150% cheaper before it can win over competing (hash) join plans • RISK_PREMIUM_SERIAL is set to 1.5 -- serial plan must be 50% cheaper before it can win over completing parallel plans • PARTITIONING_SCHEME_SHARING is set to 1 -- subset sharing of partition schemes between adjacent ESP fragments SYSTEM tells the optimizer to make a safe choice of query plans. This means: • RISK_PREMIUM_NJ is set to 1.2 -- nested join must be 20% cheaper before it can win over competing (hash) join plans • RISK_PREMIUM_SERIAL is set to 1.2 -- serial plan must be 20% cheaper before it can win over completing parallel plans • PARTITIONING_SCHEME_SHARING is set to 1 -- subset sharing of partition schemes between adjacent ESP fragments MIN tells the optimizer to believe its cardinality estimates will always be correct when choosing query plans i.e. don't apply any risk premium for risky operations. This means: • RISK_PREMIUM_NJ is set to1.0 -- nested join can win over competing (hash) join plans purely based on cost & cardinality estimates • RISK_PREMIUM_SERIAL is set to1.0 -- serial plan can win over completing parallel plans purely based on cost & cardinality estimates • PARTITIONING_SCHEME_SHARING is set to 0 -- complete sharing of partition schemes between adjacent ESP fragments If histograms are accurate and the queries are relatively simple then you could choose a lower robustness setting. Invariably in ad hoc complex query environments where queries could end up processing large amounts of data, you should consider higher settings. If you notice that when queries are not performing well it is due to either nested join

	plans, serial plans, or reduced parallelism, then you could consider increasing risk premiums to see if you can get overall better performance.
Production usage	It is best to try out different options for best overall performance in a Development instance before rolling changed settings out to production.
Impact	Specifying a risk premium buys insurance against nested joins or serial plans being chosen when they should not have been. However, this can also result in such plans not being chosen where the cardinality estimation was in fact accurate and such plans could have performed better. So this setting should be used with care in order to get robustness with a net gain in performance.
Level	Any. There may be cases where there are different applications or workloads that might benefit from this CQD more than other workloads. In such cases this could be used at the Service level.
Conflicts/Synergies	This conflicts with the RISK_PREMIUM_NJ, RISK_PREMIUM_SERIAL, and PARTITIONING_SCHEME_SHARING settings. Use this CQD when possible. Use the risk premium settings rarely, when specific premiums need to be set differently to address specific issues. If overall this CQD is working well but you have outliers e.g. poor nested join plans or inappropriate serial plans, you could use the individual CQDs at a finer granularity, such as at a query level, to get better plans.
Real Problem Addressed	Sometimes the cardinality underestimation, compared to the actual row counts, resulting in a nested join or serial plan being chosen when it shouldn't have been, may be due to not enough, or inaccurate, histogram statistics information available to the optimizer. So, first and foremost, histogram statistics should be kept up to date along with the multi- column statistics that the optimizer may warn about. However, cardinality underestimations may still happen at higher levels of an execution plan.
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

3.25 SKEW_EXPLAIN

Description	Turns on the reporting of anti-skew join plan details in EXPLAIN or EXPLAIN options 'f
Values	'OFF': Disables the use of skew explain 'ON': Enables skew information in explain Default: 'OFF'
Usage	
Production usage	
Impact	Does not change query plans
Level	Any
Conflicts/Synergies	This CQD allows additional information to be displayed in explain plans. It has no impact on query plans
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.26 SKEW_ROWCOUNT_THRESHOLD

Description	This row count threshold tells the optimizer to look for skewed values and address that skew, only if the number of rows in the table exceeds this threshold.
Values	'n': where n is the number of rows Default: '1000000'
Usage	Skew can occur either in a nested join or a hash join. Currently such a skew is typically handled for the outer table of a join. The outer table is identified by the Explain plan.

	So the first indication would be that there is a performance problem and it is due to a skew that is not being addressed by the compiler. A skew could be detected by observing the imbalanced use of CPUs during query execution, or a suspicion based on the knowledge that a severe skew does exist in the table that could be causing this problem and is not being dealt with. The setting of the CQD has been chosen to handle most skew values that are worth worrying about. That is, in other cases there may be skew but the impact on total query execution may be minimal. However, there could be cases where this is not true. If such a case is detected you would need to look at the cardinality of the table where such skew may be existing (typically the outer table in the join) and see if the table has less rows than defined by this CQD (1 million by default). If that is the case, set this CQD to a value smaller than the number of rows in that table. If that addresses the performance problem then the skew has been addressed. SKEW_EXPLAIN can be turned on before an EXPLAIN of the plan, to see if the skew is in fact being dealt with by the optimizer where it was not before. If this CQD needs to be set to a value other than its default value, a defect should be filed with the information on why this had to be done, in order for development to get the appropriate feedback on the default in order to take any corrective action if necessary.
Production usage	
Impact	A lower setting would allow more skews to be detected and addressed. However, it will increase compile time.
Level	Query, session, or service
Conflicts/Synergies	SKEW_EXPLAIN can be used as described in Usages. Also, SKEW_SENSITIVITY_THRESHOLD is relevant only if this threshold allows a skew to be detected.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.27 SKEW_SENSITIVITY_THRESHOLD

Description	Allows users to specify a threshold that determines whether a value in the join column is considered skewed
Values	'-1': Disables the use of skew buster 'n': where n is a floating-point value ≥0
	Default: '1'
Usage	A value is considered skewed if its occurrence frequency is greater than the threshold value multiplied by the average number of rows per processing node (CPU). That is, if average row count = row count / number of processing nodes, then frequency of the value is greater than threshold * average row count A setting of n, where n ≥ 0, indicates that the value should be considered as skewed if its occurrence frequency is greater than n times the average number of rows per processing node If some small skew is suspected for hash joins during query execution (detected by observing spiked CPU busy usage), try to lower this setting. A default setting of 0.1 should eliminate most skews. Setting the CQD to a very large value (e.g., 10) is not recommended, as it effectively turns off skew buster.
Production usage	
Impact	
Level	Any
Conflicts/Synergies	This CQD is only relevant if the SKEW_ROWCOUNT_THRESHOLD has been met. That

	CQD controls the row count of the table at which the optimizer looks for a skew. To change that setting you need to have Esgyn review and change it.
Real Problem Addressed	Skew is quite common in a real BI application, and is effectively addressed by skew buster. However, there may be design opportunities that could help address the problem as well.
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

3.28 SUBQUERY_UNNESTING

Description	Allows correlated subqueries in a SQL statement to be unnested, so that they can be executed efficiently.
Values	ON / OFF Default: 'ON'
Usage	Turn this attribute OFF if in certain rare cases when unnesting a correlated subquery causes performance to degrade. If this attribute has to be turned OFF that could indicate a bug in the optimizer.
Production usage	Yes
Impact	Turn OFF with caution at a system level, as other queries which rely on un-nesting could be adversely impacted.
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

3.29 SUBQUERY_UNNESTING_EARLY_GROUP

Description	A flavor of subquery unnesting query transformation where the grouping is done before the join that implements the correlation of subquery with outer query. Early grouping is beneficial when either join increases the cardinality several fold or when the group by reduces the cardinality significantly.
Values	ON | OFF Default is OFF.
Usage	Certain types of subqueries perform better with this transformation enabled. The default method of unnesting places the groupby above the join. It is more widely applicable. If this flavor of unnesting is enabled, it will always be chosen over the default flavor, if it can produce a valid plan.
Production usage	Yes
Impact	Improve performance of certain correlated subqueries
Level	Session
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

3.30 TRAF_ALLOW_ESP_COLOCATION

Description	Enable or disable ESP processes to be running on the same HBase region servers to

	minimize the inter-node network traffic between the ESP processes and the region servers.
Values	'ON' to enable the feature, and 'OFF' to disable. The default setting is 'OFF'.
Usage	Enable the feature when each region server serves approximately equal amount of data, and/or reducing network traffic is important.
Production usage	Yes
Impact	Plan quality
Level	Compiler instance
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.1
Deprecated in Release	

3.31 TRAF_UPSERT_MODE

Description	Controls the behavior of UPSERT against Trafodion tables when a column with default value is omitted.
Values	MERGE/REPLACE/OPTIMAL Default: 'MERGE'
Usage	When one or more columns with default value are omitted MERGE' – Either behaves like merge or is converted into MERGE statement 'REPLACE' – The omitted columns are replaced with default values 'OPTIMAL' – Aligned row format behaves like 'REPLACE' Non-aligned row format behaves like 'MERGE'
Production usage	Choose the appropriate value depending upon your application needs
Impact	
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.1
Deprecated in Release	

3.32 UPD_ORDERED

Description	Controls whether rows should be inserted, updated, or deleted in clustering key order
Values	ON: The optimizer generates and considers plans where the rows are inserted, updated, or deleted in clustering key order OFF: The optimizer does not generate plans where the rows must be inserted, updated, or deleted in clustering key order Default: 'ON'
Usage	Inserting / updating / deleting rows in the clustering key order is most efficient and highly recommended. Turning this CQD OFF may result in saving the data sorting cost but at the expense of having less efficient random I/O Insert/Update/Delete operations.
	If you know that he data is already sorted in clustering key order, or is mostly in clustering key order, so that it would not result in random I/O, you could set this CQD to OFF.
Production usage	
Impact	If turned OFF, the system may perform large number of inefficient Random I/Os when performing Insert/Update/Delete operations

Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

4 Query Execution

This section describes CQDs that are used to influence query execution.

4.1 BMO_MEMORY_LIMIT_PER_NODE_IN_MB

Description	Restricts the memory allocated to the Big Memory Operators(BMOs) of the query. Sets the limit of memory allocated for the BMO operator instances of all ESP processes hosted in a node for a query. This is a not hard limit, but it determines the memory quota assigned to the operator instances based on the memory estimate of the operator at the time of compilation of the query. When the calculated memory estimate falls below a certain value, a default minimum memory quota is assigned. When it is above a certain value, a default maximum memory quota is assigned. The default min. and max. value is dependent on the type of BMO and is configurable by CQDs.
Values	1024 (10 GB)
Usage	Set this value based on concurrency and the amount of physical memory available
Production usage	Yes, after validating in a Dev environment.
Impact	A smaller value may result in overflow to scratch files and thus impact the performance of the query. A higher value may result in unstable environment with rogue queries due to increased memory usage causing memory pressure.
Level	
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

4.2 DOP_ADJUST

Description	This is a performance feature to optimize the number of ESP processes, for concurrent execution of queries. Past R&D experiences in compiler tell us that the compiler may overestimate cardinality and thus over-assign esps. We may also optimize for performance in a single stream test in a PoC, by maximizing on the level of parallelism and carry the same settings to the concurrent runs. In either case, too many esps can be selected which can lead to system resource contentions on physical memory and # of ports. The feature uses the run-time stats data to refine the number of ESPs, for all relational operators, by adjusting the DoP based on actual data processed when the same query is seen again. When enabled, the 1st execution of a query writes the run-time stats to HDFS. The subsequent compilation of the same query in the same or different stream looks up the run-time stats and makes the DoP adjustment.
Values	CQD DOP_ADJUST '2' -- enable CQD DOP_ADJUST '0' -- disable The default value is '0'.
Usage	Use a value of 0 to disable when running a single stream Use a value of 2 when running concurrent and repeated queries
Production usage	Yes

Impact	Adjust the degree of parallelism for concurrent and repeated queries.
Level	Query Performance
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

4.3 HBASE_ASYNC_OPERATION

Description	Allows index maintenance to be performed concurrently with base table operation.
Values	ON / OFF Default: 'ON'
Usage	HBase put operations are blocking. When the table has one or more indexes, the IUD operation response time is improved by executing the index maintenance operations concurrently with the base table operation by executing the put operations to these HBase tables in different threads.
Production usage	It is 'ON' by default. This feature can be disabled by setting it to 'OFF'
Impact	IUD operations on tables with one or more indexes can become slower.
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

4.4 HBASE_CACHE_BLOCKS

Description	Influences HBase to retain the data blocks in memory after they are read.
Values	ON / OFF/SYSTEM Default: 'SYSTEM'
Usage	HBase maintains the block cache structure to retain the data blocks in memory after they are read. In LRU block cache configuration, the amount of block cache retained in memory is proportional to the amount of reserved maximum Java heap size of the region server. LRU Block Cache is the default in HBase. Trafodion optimizer determines if a sequential scan of the HBase table in a query would cause the full eviction of the data blocks cached earlier impacting the performance of the random reads. The cache blocks option is turned off for the table in such a case. Set the CQD HBASE_REGION_SERVER_MAX_HEAP_SIZE value to reflect the amount of java heap size reserved for the region servers. It is used by the Trafodion optimizer to evaluate if the block cache to be turned off.
Production usage	Leave the setting to be 'SYSTEM' when HBase is configured to use LRU block cache. If needed, the user override this settings to 'ON' or 'OFF'. With other configurations, the user needs to set HBASE_CACHE_BLOCKS on or off based on their application needs.
Impact	Automatically retains the random read performance
Level	Query
Conflicts/Synergies	
Real Problem Addressed	

Introduced in Release	EsgynDB R2.0
Deprecated in Release	

4.5 HBASE_FILTER_PREDS

Description	Allows push down of predicates to region servers using HBASE filters and optimize the columns retrieved from region servers. Only supported for NON ALIGN FORMAT tables. When set to OFF, predicates are never pushed down. When set to 1 or ON, a first implementation targeted for deprecation is enabled. Only support simple predicate formed by a combination of AND, and could be counter-productive when applied on nullable columns. When set to '2', full feature is on. An explain plan can show if predicates are successfully pushed down to region server and what columns are really retrieved.
Values	Default: 'OFF' 'OFF' or '0', 'ON' or '1','2'
Usage	
Production usage	
Impact	Performance of table scan is improved by limiting the columns retrieve to a strict minimum, and filtering out rows as early as possible. The flip side is that it increases work done at Region Server level.
Level	System or Session
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

4.6 HBASE_HASH2_PARTITIONING

Description	Treat salted Trafodion tables as hash-partitioned on the salt columns.
Values	ON/OFF. Default: ON
Usage	If for some reasons there are issues with parallel plans on salted tables, especially with data skew, try setting this CQD to OFF.
Production usage	
Impact	
Level	
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

4.7 HBASE_NUM_CACHE_ROWS_MAX

Description	Determines the number of rows obtained from hbase in one RPC call to the region server in a sequential scan operation
Values	Numeric value

	Default: ’10000’
Usage	This CQD can be used to tune the query to perform optimally by reducing the number of interactions to region servers during a sequential scanning of a table. The user needs to consider how soon the maximum number of rows will be materialized on the region server. When filtering is pushed down to region servers, it can take a longer time depending upon the query and the predicates involved. This can result in HBase scanner timeouts.
Production usage	Use the default setting and reduce the value to avoid HBase scanner timeouts.
Impact	
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

4.8 HBASE_ROWSET_VSBB_OPT

Description	Allows IUD operations to be performed as an HBase batch put operation.
Values	ON / OFF Default: ‘ON’
Usage	When IUD operation involves multiple tuples, Trafodion optimizer evaluates if these operations can also be done in a batch manner at HBase level to reduce the network interactions between the client applications and the region servers. If so, the query plan involves VSBB operators. The Virtual Sequential Block Buffer(VSBB) name is retained in Trafodion though it is unrelated to HBase.
Production usage	Use the default setting. This feature can be disabled by setting it to ‘OFF’
Impact	IUD operations can become slower if turned off.
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

4.9 HBASE_ROWSET_VSBB_SIZE

Description	Determines the maximum number of rows in a batch put operation to hbase.
Values	A numeric value Default: ‘1024’
Usage	The default size of 1024 is the maximum number of rows in a batch. Trafodion execution engine already adjusts the number of rows in a batch depending upon how fast the queue to IUD operator gets filled up in the data flow architecture of Trafodion. User can adjust the maximum size to suit their application needs and thus tune it to perform optimally.
Production usage	Use the default setting. This feature can be disabled by setting HBASE_ROWSET_VSBB_OPT to ‘OFF’
Impact	IUD operations can become slower if turned off.
Level	Query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0

Deprecated in Release	

4.10 HBASE_SMALL_SCANNER

Description	Leverage HBase small scanner optimization that is using 3 times less IO, and non-blocking reads for higher concurrency support. When a scan is known to require less than an "HBASE BLOCK SIZE" (default is 64K), enabling HBase small scan is optimal.
Values	Default: 'OFF' 'OFF', 'SYSTEM', 'ON'
Usage	OFF: never use small scanner SYSTEM: enable small scanner only if compiler evaluates that the scan size will fit in the table's HBASE BLOCK SIZE ON: will enable small scanner no matter the size of scan.
Production usage	
Impact	Performance of small scan increase by 1.4X. Can be very useful for MDAM scans.
Level	System or Session
Conflicts/Synergies	With MDAM and correctly picking HBASE block size to make each MDAM scan operation fit within a HBASE BLOCK SIZE boundary, performance of MDAM can we boosted by 1.4X. Wrongly enabling small scanner on large size scan will result in a 6% performance decrease, but still return correct results.
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

4.11 HIVE_LOCALITY_BALANCE_LEVEL

Description	Controls the use of locality and the degree of balancing work between ESPs to give each the same load. Locality here means assigning an HDFS block to an ESP that has a local replica.
Values	-1 | 0 | 1 | 2 | 3 Default is 3
Usage	-1: Don't split HDFS files (no locality) 0: Don't use this method at all (no locality) 1: Try to keep ESPs within 10 % of the average load (locality) 2: Try to keep ESPs within 2.5 % of the average load (locality) 3: Try to make all ESPs completely even, if possible (locality) In the highest selected level, we do the following: ▪ Try to make all ESPs completely even, if possible ▪ Split scan infos to achieve a better balance, if allowed ▪ Sacrifice locality for a more even balance
Production usage	Yes
Impact	Query performance
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

4.12 HIVE_STATS_CACHING

Description	For ORC and Parquet, file statistics are required to correctly organize (Stripe or Rowgroup distribution per ESP) and calibrate (number of ESP) the scan operator. However, reading these statistics require opening and reading all files to access their metadata portion. This is time consuming and increase first time query compile. HIVE_STATS_CACHING is a feature that consolidate these hive files statistics into a single file per table. It uses it when we have detected no changes based on file timestamps on the source table. This way, high cost first time compile is seen only once, on the first query against the table that just got updated. This first query can be part of the table loading process, so that it is hidden from users.
Values	ON | OFF Default is OFF
Usage	
Production usage	Yes
Impact	
Level	Session
Conflicts/Synergies	
Real Problem Addressed	First time query compile per session
Introduced in Release	EsgynDB R2.3
Deprecated in Release	-

4.13 ORC_READ_STRIPE_INFO

Description	Read stripe header for all files that are part of an ORC table during query compilation. This will guarantee a better cardinality estimate and the most balanced allocation of ESPs to scan the default. The default method is to read stripes from a few files and use that to estimate information about remaining files.
Values	ON | OFF Default is OFF
Usage	The default sampling method is faster and reduces compile time. Reading all stripes could result in better execution time. This is necessary when file or stripe size for a single table are uneven.
Production usage	Yes
Impact	Increases compile time.
Level	System
Conflicts/Synergies	Use with HIVE_STATS_CACHING to manage growth in compile time.With stats caching, this cqd with cause all orc stripes of a table to be read only once till the table has a data change. For ost compiles we will see no change in compile time even though we are using all stripe information rather than a sampled subset.
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

5 Manage Histograms

This section describes CQDs that are used to manage histograms.

5.1 CACHE_HISTOGRAMS_REFRESH_INTERVAL

Description	Time interval after which timestamps for cached histograms are checked to be refreshed
Values	Unsigned integer value in seconds Default: '3600' (1 hour)
Usage	Histogram statistics are cached so that the compiler can avoid access to the metadata tables, thereby reducing compile times. The timestamp of the tables are checked against those of the cached histograms at an interval specified by this CQD, in order to see if the cached histograms need to be refreshed. You can increase the interval to reduce the impact on compile times as long as you do not need to pick fresh statistics more frequently in order to improve query performance. It may be that the default interval is too long and you would rather refresh the statistics more frequently than the default one hour, in order to improve query performance at the cost of increased compile times. This setting depends on how frequently you are updating statistics on tables. There is no point in refreshing statistics frequently when statistics are not being updated during that time. On the other hand if you are updating statistics, or generating them for the first time on freshly loaded tables frequently enough, and you want these to be picked up immediately by the compiler since you have seen this to have a dramatic impact on plan quality, then you can make the refresh more frequent.
Production usage	
Impact	Longer histogram refresh intervals can improve compile times. However, the longer the refresh interval the more obsolete the histograms. That could result in poor performance for queries that could leverage recently updated statistics.
Level	System or Service
Conflicts/Synergies	Frequency of update statistics run either using MAINTAIN or using Update Statistics Automation Server.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

5.2 HIST_MISSING_STATS_WARNING_LEVEL

Description	This CQD controls the level of missing statistics warnings that should be displayed. The warnings impacted are 6007, 6008, 6010 and 6011.
Values	'0': Display no warnings. Update Statistics Automation Server still continues to record missing column statistics. '1': Display only missing single column statistics warnings. These include 6008 and 6011. '2': Display all missing single and multi-column statistics warnings for scans only. '3': Display all missing single and multi-column statistics warnings for scans and join operators only. Update Statistics Automation Server will still continue to record missing column statistics. '4': Display all missing single and multi-column statistics warnings Default: '4'
Usage	You could just change this setting to '0' if you don't want to see these warnings. If you want to track the warnings, then you have a choice of which warnings you want to track.

	Each setting gives you the ability to filter the warnings seen for missing single or multi-column statistics for join or scan operations. This controls the resulting number of warning messages. If poor query plans are being caused by cardinality estimations that seem to be off, you can check the histogram statistics to see if statistics are being collected for those columns and how accurate they are. If you don't find statistics being collected, you could look for the warnings by setting this CQD to the appropriate setting. Based on that you could take appropriate action to find out if update statistics is being run to generate those statistics.
Production usage	Many tools, like Microstrategy, divide a query into several steps. During the first phases volatile tables are created and populated, the last phase usually joins all the volatile tables created in the previous steps. Usually statistics are not needed for those volatile tables because the final join is straight forward and the optimizer has no big choices. Nevertheless EMS will be flooded with –in this case useless- warnings if you don't set the warning level to 0. If possible, try to direct queries from those tools to a dedicated service where you set the warning level to 0.
Impact	Though the warnings give information about all statistics that are missing, it could be overwhelming for the user to get several warnings. Not all warnings may contribute to plan improvements. The optimizer issues multi-column statistics warnings based on the search path, some of which may not even impact the plan quality. Also, the cost of gathering statistics on those columns may not bring commensurate benefit to a large number of queries.
Level	System
Conflicts/Synergies	Update Statistics Automation Server does capture these warnings and will gather statistics felt appropriate for query performance. If you are using USAS then you may set the warning level to low or 0.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

5.3 HIST_NO_STATS_REFRESH_INTERVAL

Description	Defines the time interval after which the fake histograms in the cache should be refreshed unconditionally i.e. without checking time-stamps.
Values	Integer value in seconds Default: '3600' (1 hour)
Usage	Histogram statistics are "fake" when update statistics is not being run, but instead the customer is updating the histogram tables directly with statistics to guide the optimizer. This may be done if say the data in the table is very volatile (such as for temporary tables), update statistics is not possible because of constant flush and fill of the table occurring, and statistics are manually set to provide some guidance to the optimizer to generate a good plan. If these fake statistics are updated constantly to reflect the data churn, this default can be set to 0. This would ensure that the histograms with fake statistics are not cached, and are always refreshed. If these fake statistics are set and not touched again, then this interval could be set very high.
Production usage	Yes
Impact	Setting a high interval improves compilation time. However, if statistics are being updated, the compiler may be working with obsolete histogram statistics, potentially resulting in poorer plans.
Level	Service

Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

5.4 HIST_PREFETCH

Description	Influences the compiler to pre-fetch the histograms and save them in cache
Values	'ON': Pre-fetches the histograms 'OFF': Does not pre-fetch histograms Default is 'ON'
Usage	You may want to turn this off if you don't want to pre-fetch a large number of histograms, many of which may not be used.
Production usage	Yes
Impact	Though it makes compilation time faster, it may result in the histogram cache to be filled with histograms that may never be used.
Level	System or Service
Conflicts/Synergies	This CQD is used along with CACHE_HISTOGRAMS. If CACHE_HISTOGRAMS is OFF, then this CQD has no effect.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

5.5 HIST_ROWCOUNT_REQUIRING_STATS

Description	Specifies the minimum row count for which the optimizer needs histograms, in order to compute better cardinality estimates. The optimizer will not issue any missing statistics warnings for tables whose size is smaller than the value of this CQD.
Values	Integer Default: '50000'
Usage	You can use this CQD to reduce the number of statistics warnings
Production usage	
Impact	Missing statistics warnings are not displayed for smaller tables, which in most cases don't impact plan quality much. However, there may be some exceptions where missing statistics on small tables could result in sub-optimal plans
Level	System
Conflicts/Synergies	This CQD is used with HIST_MISSING_STATS_WARNING_LEVEL. If the warning level CQD is 0, then this CQD does not have any effect. Also, for tables having fewer rows than set in this CQD, no warnings will be displayed irrespective of the warning level.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

5.6 HIST_USE_SAMPLE_FOR_CARDINALITY_ESTIMATION

Description	The CQD enables the Compile Time Stats feature. Compile Time Stats are produced during query plan generation by executing a subset of the query on a subset of data to gather more accurate cardinality estimations.

Values	'ON': Compile Time Statistics is enabled 'OFF': Compile Time Statistics is disabled Default: 'ON'
Usage	The feature is very helpful for cases when the query contains complex predicates on a table. These predicates include - LIKE, CASE, any other expressions or more than one range predicates and equality on large character columns. It can be disabled if most of the queries are single table or at most 2-way joins. It can also be disabled if the extra collection of statistics seems to be adversely affecting the total query compile and execution time.
Production usage	Yes
Impact	The feature improves cardinality estimates for Scan operators thus improving the plan quality. On the other hand it can also increase the compiletime.
Level	Any
Conflicts/Synergies	In order to use the feature in its default form, sample tables should exist in public_access_schema.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

6 Transaction Control and Locking

This section describes CQDs that are used for transaction control and locking.

6.1 ISOLATION_LEVEL

Description	Specifies the default transaction isolation level that queries will use
Values	'READ UNCOMMITTED', 'READ COMMITTED', 'REPEATABLE READ', 'SERIALIZABLE' Default: 'READ COMMITTED' (ANSI)
Usage	If you are fine with uncommitted access, i.e. reading dirty data when queries are accessing data that is being simultaneously updated, you can set the default isolation level as READ UNCOMMITTED. The default isolation level of READ COMMITTED can cause concurrency issues since reads would wait on locked rows. If rows are locked by long running transactions with infrequent commits, this can cause severe concurrency issues for SELECT queries. Please see Synergies.
Production usage	
Impact	Has an implication on locking and concurrency. If set to READ UNCOMMITTED then select queries will read through locks and don't have to wait on locks. But they won't see committed consistent data. If set to READ COMMITTED (the default setting) then the reads will wait on locked rows before they proceed with the scan. The read can proceed only when the rows locked by another transaction are released after that transaction commits. The reader will not lock rows. If set to REPEATABLE READ or SERIALIZABLE it will have severe implications on concurrency since every row read is also locked.
Level	While one can use this at a query or a service level, the most common use is a system-wide setting. If query tools are being used, then the query level setting cannot be used. A service level setting may provide uncommitted access to certain users while providing the default committed access to the other users, depending which users need to see consistent data. If however, access to tables during updates is well controlled and read uncommitted is acceptable, this can be set at the system level.
Conflicts/Synergies	The problem with using READ UNCOMMITTED as the isolation level default value is that in a SET TRANSACTION statement, the only possible access mode is READ ONLY. Any query that attempts to update the database would fail. To facilitate updates and DDL statements while the isolation level is set to READ UNCOMMITTED, a new default attribute ISOLATION_LEVEL_FOR_UPDATES is provided. This default attribute specifies the isolation level for update and DDL statements. If not specified, or if not present in the SYSTEM_DEFAULTS table, the default value will be the same as the ISOLATION_LEVEL default attribute. READ UNCOMMITTED, a new default attribute ISOLATION_LEVEL_FOR_UPDATES is provided. This default attribute specifies the isolation level for update and DDL statements. If not specified, or if not present in the SYSTEM_DEFAULTS table, the default value will be the same as the ISOLATION_LEVEL default attribute. However, if specified or present in the SYSTEM_DEFAULTS table, its value is used as the isolation level for updates and DDL statements. UPDATE in ISOLATION_LEVEL_FOR_UPDATES refers to INSERT, UPDATE, and DELETE statements.
Real Problem Addressed	

Introduced in Release	EsgynDB R1.0
Deprecated in Release	

6.2 ISOLATION_LEVEL_FOR_UPDATES

Description	Specifies the default transaction isolation level for update operations – INSERT, UPDATE, DELETE
Values	'READ UNCOMMITTED', 'READ COMMITTED', 'REPEATABLE READ', 'SERIALIZABLE' Default: 'READ COMMITTED' (ANSI)
Usage	Set this CQD to READ UNCOMMITTED if you want to prevent users from performing any updates.
Production usage	
Impact	When set, it prevents users from doing any updates – INSERT, UPDATE, DELETE operations.
Level	Service
Conflicts/Synergies	Works with the ISOLATION_LEVEL setting. Both settings are READ COMMITTED by default. ISOLATION_LEVEL can be set to READ UNCOMMITTED. This CQD still remains READ COMMITTED. You can change it to READ UNCOMMITED to prevent queries running at the service level to not perform any updates.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

7 Runtime Controls

This section describes CQDs that are used for runtime controls.

7.1 LAST0_MODE

Description	Ensures that all parts of the query plan will be executed but no rows will be returned by the query
Values	'ON' | 'OFF' Default: 'OFF'
Usage	This setting provides a realistic measure of the query's performance, minus the cost/time of returning the rows to the client. It is especially useful for testing the plans and performance of queries that return large result sets.
Production usage	This should only be used to assess the performance of a query
Impact	The query runs completely but no rows are returned
Level	Query
Conflicts/Synergies	Not to be confused with SELECT [LAST 0] which behaves the same way but does not guarantee that the plan will be the same as when you do not use the [LAST 0] clause in the query.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

7.2 QUERY_LIMIT_SQL_PROCESS_CPU

Description	This default is used to limit the amount of CPU time that a query is allowed to use in any one server process (ESP) including the DCS server. If a query exceeds the limit, an error is raised and the query is terminated. This is a way to limit the impact on the system of a poorly written or badly optimized query.
Values	The default value is '0' (zero), meaning that there is no limit. Other values greater than zero set a limit, in seconds, to how much CPU time a query is allowed to use. The maximum value is 2,147,483,583 seconds
Usage	This setting will help customers with queries that are poorly written or are badly optimized. A poorly-written query does not use predicates to limit the number of rows processed. A query that joins large tables without a predicate can have a severe impact on the system. A badly- optimized query can result from failure to maintain histograms. Typically, these are ad-hoc queries. These types of queries seldom run to completion, and are instead stopped after the problems that they cause to other users of the system are noticed.
Production usage	
Impact	Use of this default can prevent any one query from using an unlimited amount of CPU time. However, if the default is set too low, then even well-behaved, useful queries will fail to complete.
Level	Service
Conflicts/Synergies	
Real Problem Addressed	End-users can be educated or tools optimized in ways to write good queries. Procedures can ensure that table histograms are kept current.
Introduced in Release	EsgynDB R1.0

Deprecated in Release

8 Schema Controls

This section describes CQDs that are used for schema controls.

8.1 CATALOG

Description	Specifies the default catalog name for all DDL and DML statements
Values	Any valid ANSI name, including delimited names. Default: 'TRAFODION'
Usage	Trafodion tables must be in a catalog called TRAFODION. If primary usage will be for Hive access or native HBase table access then the catalog could be changed to HIVE or HBASE respectively. The default will be overridden by any catalog name specified in a SQL statement.
Production usage	Yes
Impact	
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

8.2 SCHEMA

Description	Sets the default schema for the session
Values	SQL identifier Default: Default ROLE of the user
Usage	A SET SCHEMA statement, or a control query default SCHEMA statement, can be used to override the default schema name.
Production usage	It is a convenience so users do not have to type in 2 part names
Impact	
Level	Any
Conflicts/Synergies	Alternately you can use the SET SCHEMA statement. There is a CATALOG default also that allows you to set the catalog, e.g. CATALOG can be NEO, the system catalog, or MANAGEABILITY. Again it is a convenience to avoid typing in long names.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

9 Table Definition

This section describes CQDs that are used for table definition.

9.1 ALLOW_NULLABLE_UNIQUE_CONSTRAINT

Description	Allow Trafodion tables to be created with NULLABLE columns in the PRIMARY or STORE BY key.
Values	ON / OFF Default: 'OFF'
Usage	Allows NULLABLE columns to be included in the PRIMARY or STORE BY key for Trafodion tables. Attribute must be set prior to creating the table. It is not necessary to specify this attribute during DML operations on the table. When performing UPDATE STATISTICS with SAMPLE option on such tables, the cqd must again be set so that an appropriate sample table can be created.
Production usage	
Impact	Two bytes are added to the key for each nullable column
Level	Table
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

9.2 HBASE_BLOCK_SIZE

Description	Allow Trafodion tables to be created with specified HBase block size.
Values	Positive integer Default: '65536'
Usage	The value of this attribute is passed on to HBase when a Trafodion table is created in HBase. Please see https://hbase.apache.org/book.html for usage information
Production usage	Yes
Impact	Depends on type of access on table. Choose block size that is appropriate for how the table will be primarily accessed.
Level	Table
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

9.3 HIVE_DEFAULT_CHARSET

Description	This default tells Trafodion what character set the columns of Hive tables should have.
Values	ISO88591, UTF8. Default: UTF8
Usage	Set this to ISO88591 when reading from Hive tables with ISO8859-1 data.
Production usage	
Impact	
Level	
Conflicts/Synergies	HIVE_FILE_CHARSET

Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

9.4 HIVE_FILE_CHARSET

Description	For certain character sets that are not supported in Trafodion, you can specify the character set here and Trafodion will automatically convert the data to the character set specified in the HIVE_DEFAULT_CHARSET default. This is currently supported only for GBK, and only if HIVE_DEFAULT_CHARSET is set to UTF8.
Values	empty, GBK. Default: empty
Usage	Leave this blank, unless you want to access GBK data in Hive tables.
Production usage	
Impact	
Level	
Conflicts/Synergies	HIVE_DEFAULT_CHARSET
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

9.5 HIVE_MAX_STRING_LENGTH_IN_BYTES

Description	Hive columns of type STRING have a maximum length in Trafodion, which you can specify with this default. Note that for UTF-8 data, this length is specified in bytes, not UTF-8 characters.
Values	1-n. Default: 32000.
Usage	Set this to the lowest possible value to improve system performance.
Production usage	Yes
Impact	
Level	
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

9.6 TRAF_DEFAULT_COL_CHARSET

Description	This default tells Trafodion what character set the columns of Trafodion tables should have.
Values	ISO88591, UTF8. Default: ISO88591
Usage	
Production usage	
Impact	
Level	
Conflicts/Synergies	
Real Problem Addressed	

Introduced in Release	EsgynDB R1.0
Deprecated in Release	

10 Update Statistics

10.1 USTAT_MAX_READ_AGE_IN_MIN

Description	When performing update statistics with the NECESSARY keyword, this is the number of minutes that are allowed to have elapsed since a histogram was marked as read for it to be regenerated. Histograms that were marked more than USTAT_MAX_READ_AGE_IN_MIN minutes ago will not be regenerated.
Values	0 to max unsigned integer value Default 5760 (4 days).
Usage	Influences how frequently the histograms for a table are regenerated. If a table is being used frequently then chances are that its histograms will also be considered for update frequently. However, if a table is not used frequently, this CQD influences how frequently the histograms for that table are updated. A smaller setting reduces the number of histograms being updated if there are many tables that have not been used within that interval. A larger setting will update histogram for many more tables that are not being accessed that often.
Production usage	
Impact	Influences the number of histograms that need to be regenerated and therefore the time it takes for update statistics automation to regenerate histograms for all the tables that so qualify.
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

10.2 USTAT_MIN_ROWCOUNT_FOR_SAMPLE

Description	Sets the minimum rows that need to be in a table before sampling is used to update statistics for that table. If a table has a fewer rows than the value of this CQD, the SAMPLE option will be silently ignored when performing update statistics.
Values	1 to max unsigned integer value Default: 10000
Usage	Influences for what tables sampling is used for histogram statistics generation. If the setting is smaller, then more tables will qualify for sampling. If the setting is larger, then fewer tables will qualify for sampling. Sampling can result in faster update statistics run times. But for smaller tables, it can also result in poor histogram accuracy which could result in poor query plans.
Production usage	
Impact	Setting this CQD to a smaller value means that sampling will be used for tables with fewer rows, when the SAMPLE option is specified as part of update statistics. This can result in less accurate histograms and poor query plans, since the sample size may be too small to generate good estimates for histograms. Setting this CQD to a larger value can result in sampling not being used for many tables and therefore longer update statistics run times. However, these tables may also have more accurate histograms.
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0

Deprecated in Release	

10.3 USTAT_MIN_ROWCOUNT_FOR_LOW_SAMPLE

Description	This CQD effects the behavior of the UPDATE STATISTICS utility. It places a lower limit on the number of sample rows that will be used when sampling. If the number of rows in the table is less than this value and sampling is used, then the sample size used will be that specified by the HIST_DEFAULT_SAMPLE_MIN (which defaults to 10,000).
Values	This CQD may take on any integer value greater than zero. The default for this CQD is 1,000,000.
Usage	Prevents accidental use of too-small samples when generating statistics on tables. If sample sizes are too small, then histogram statistics will be less accurate, leading to potentially less efficient query plans.
Production usage	
Impact	Setting this CQD to lower values may result in smaller sample sizes for small tables. This may slightly improve UPDATE STATISTICS run time, but at the cost of potentially less efficient queries.
Level	System
Conflicts/Synergies	As mentioned in the Description above, the value of HIST_DEFAULT_SAMPLE_MIN effects the behavior of this CQD.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

11 Operational Controls

This section describes CQDs that are used for operational controls.

11.1 AUTO_QUERY_RETRY_WARNINGS

Description	Indicates whether a warning should be issued when a query is retried, in case a failed query is automatically retried.
Values	ON / OFF Default: 'OFF'
Usage	There are certain cases, such as when a CPU failure occurs, where a query may fail midstream. In many of such failure scenarios, if the query has not returned any data, it will be retried automatically. When such retries happen, you may want to see a warning that an automatic retry took place. That would be a reason to turn this on.
Production usage	
Impact	You will get a warning message every time a query is automatically retried due to a failure. When there is a CPU failure, a large number of queries may be impacted. So you need to assess if you want to see a flood of warnings. The warning is returned after the query completes.
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

11.2 EXPLAIN_DESCRIPTION_COLUMN_SIZE

Description	Specifies maximum length of DESCRIPTION column for EXPLAIN virtual table
Values	Positive integer > 10,000 Default: -1, indicating a maximum size of 10,000
Usage	Query plan information for a SQL DML statement is stored temporarily in the Explain virtual table. For large queries, or queries with complex predicates the default size of 10 KB may be insufficient to describe certain nodes in the query plan. Specifying a larger value for this cqd will allow more bytes to be stored in the description column. Change this setting only if you see explain plan being undesirably truncated.
Production usage	Yes
Impact	
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

11.3 FIRSTN_PREDICATE_SAFETY_FACTOR

Description	This is a performance feature. For queries that have FIRST N or LIMIT clause, and if tables involved have had UPDATE STATISTICS performed on them, then this feature will use compiler estimates to synthesize a predicate that will limit the output of

	SELECT to N*factor rows. “factor” here is the setting of this cqd. The synthesized predicate will be pushed down to a scan thereby limiting dataflow for the query. If compiler estimates are incorrect and we get less than N rows, then we will discard results and retry using the Auto Query Retry feature. Retry will be without user intervention.
Values	An integer
Usage	0 denotes OFF Positive integer is enabled with the integer specifying factor by which we wish to compensate for any under estimation by the compiler Negative integer denotes that predicate will not used in the query. However, a predicate will be synthesized by the compiler and displayed in a WARNING message with SQLCODE 2997. The user could explicitly add the predicate to the query text to improve performance. Default is 0.
Production usage	Yes
Impact	Reduces resource utilization and response time for applicable queries
Level	Session
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

11.4 HBASE_REGION_SERVER_MAX_HEAP_SIZE

Description	Specify to EsgynDB the maximum Java heap size (-Xmx option) that HBase Region Server been assigned, in MB units
Values	Positive integer Default: ‘1024’
Usage	Set this attribute so that EsgynDB is aware of the maximum heap size of Region Server processes. This will enable EsgynDB to use HBase block cache in an optimal manner.
Production usage	Yes
Impact	HBase block cache will be used optimally, with small scans being cached and larger scans which may cause cache trashing to be not cached.
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

11.5 HIVE_CTAS_TABLETYPE

Description	Storage format in Hive of target table for Create Table As statement in EsgynDB that uses a Hive table as target
Values	TEXT | ORC | PARQUET Default is ORC
Usage	
Production usage	Yes
Impact	
Level	

Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.4
Deprecated in Release	

11.6 HIVE_DATA_MOD_CHECK

Description	Used to determine if a Hive table has had data modified through Insert, Update and Delete activity, between the time query plan was created and the time when this plan starts to execute. Data mod check will compare time stamps of partitions and data files of Hive table in HDFS with what is stored in the plan, at the start of query execution. If a mismatch is detected, the query will be recompiled using Auto Query Retry. With query caching the time interval between query compile and execute can be large.
Values	ON/OFF
Usage	Default is ON
Production usage	Yes
Impact	Necessary when there is concurrent Select and Insert activity against Hive tables
Level	System
Conflicts/Synergies	Performance will be slightly lower when enabled, can be disabled in Read only environments
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

11.7 HIVE_DDL

Description	Use EsgynDB SQL interfaces to execute a Hive DDL statement against the Hive service
Values	ON | OFF Default is OFF
Usage	Typically in porting existing Hive applications to EsgynDB
Production usage	Consult with Esgyn first
Impact	Allows Hive DDL syntax to be used through Esgyn in existing applications
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.4
Deprecated in Release	

11.8 HIVE_METADATA_REFRESH_INTERVAL

Description	Controls the meta-data cache for hive tables
Values	-1: never invalidate any cache entries; 0: always check the latest meta-data from Hive; > 0: the cached hive meta-data is valid only for n (=CQD value) seconds
Usage	Use of value -1 when the hive tables are read-only and no updates are possible to avoid reading the hive meta-data when the same table is referenced in queries again. Use of value 0 when frequent update of hive tables are possible. Use of a positive value when when infrequent update of hive tables can take place

	and the value can be set such that it is impossible for the hive tables to be updated within next n seconds.
Production usage	
Impact	Compilation time
Level	Compiler instance
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

11.9 METADATA_CACHE_SIZE

Description	Size of in memory cache used by compiler to store metadata information for both Hive and Trafodion tables. It is units of MB.
Values	0 disables metadata caching. A positive integer denotes maximum cache size. Some unused entries will be replaced on a cache miss.
Usage	Default is 20 (MB). For workloads that access large numbers of tables or tables with large number of partitions or files, values upto 256 MB are reasonable.
Production usage	Yes.
Impact	Compile time could increase significantly if this cache is disabled
Level	System. Must be specified in DEFAULTS table. Ineffective if used as a CQD.
Conflicts/Synergies	select * from table(natablecache('user', 'local')) ; This statement from a sql prompt can be used to monitor cache usage.
Real Problem Addressed	
Introduced in Release	EsgynDB R2.1
Deprecated in Release	

11.10 ORC_READ_NUM_ROWS

Description	This CQD controls the read of the number of rows per stripe statistics for each ORC table data file during compilation. The statistics is useful to help load balance the parallel read of all data files for a table.
Values	ON / OFF Default: 'ON'
Usage	Set the CQD to 'ON' to read the number of rows statistics.
Production usage	Yes
Impact	The compilation time may be impacted a little bit when the CQD is turned on, since it takes time to open and read the statistics from each data file.
Level	Compile Time and Query Performance
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

11.11 PARQUET_LEGACY_TIMESTAMP_FORMAT

Description	Support for Hive legacy timestamp format. If data is populated in a timestamp column from hive shell, then it gets stored in legacy hive timestamp format in Hive 1.1. (96 bit

	number). If a recent Hive version or EsgynDB is used, then the value will be a 64 bit number.
Values	ON | OFF Default is ON
Usage	With this feature enabled, the old format of timestamp used by Hive 1.1 will be read correctly by EsgynDB
Production usage	Yes
Impact	Read data from Hive correctly
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.3
Deprecated in Release	

11.12 PRIV_SENTRY_RECHECK_INTERVAL

Description	Retry interval to check if a Hive/Sentry GRANT or REVOKE has occurred. Apache Sentry implements its own GRANT/REVOKE mechanism, but does not have a notification service to inform clients. The only way a client subsystem learns of a change is by polling Sentry.
Values	Integer, in seconds Default is 60 seconds
Usage	
Production usage	Yes, if privilege is not expected to change often
Impact	A low value could result in more frequent polling, which will impact performance. Depending on how frequently such GRANTs or REVOKEs are done in Apache Sentry, use an appropriate value.
Level	System
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.2
Deprecated in Release	

11.13 QUERY_CACHE

Description	Query caching attempts to reduce compilation times by storing and reusing previously compiled query plans. It maximizes the chances of plan reuse by parameterizing literals in equality predicates. Two equality predicates "col = val1", "col = val2" are considered to match if their selectivity match. A query cache setting of '16384' means a maximum of 16,384 KB of compiler memory can be used for keeping previously compiled plans before evicting the oldest unused plan(s) to make room for the latest cacheable plan.
Values	Accepted values: Kilobytes of memory allocated to query cache, up to 4294967295. '0': Turns off query plan caching Default: '16384' (16 MB)
Usage	To choose the appropriate size for the query cache, examine your applications. Applications that use a PREPARE statement to pre-compile queries once and then EXECUTE the prepared plan, should turn off plan caching. Ad hoc query applications can specify a size that can hold most of the frequently processed queries. For example, if an application processes 40 classes of queries frequently with an average plan size of 100 KB per query, a cache size of 4000 KB

	might be optimal. (Plan size is not the same as the size of the SQL statement and is not easy to assess.) There may be applications that are operational in nature, with many small queries, and others that are analytical in nature with large complex queries. Cache size can be set differently for different service levels handling such workloads based on the classes /types of queries, size of the queries, and propensity to get cache hits. Another consideration is how frequently the cache is getting flushed due to the compiler being shutdown and a new one started by an MXOSRVR (ODBC/Connect server), in order to run queries on behalf of a different role than the role that was using the compiler before. If this happens often and not enough static servers can be started to reduce this from happening, then creating a large cache may not be useful, since it has to be flushed and filled too often. After taking the above into account the best way to really assess whether caching is effective, and tune it for your specific applications, is to understand the cache hit statistics, how many queries are forced to be removed from cache (on a least recently used basis), and a number of other statistics about the efficiency of query plan caching for your applications. Currently a lot of this information is not exposed to customers. Esgyn support / services can help you get information on how caching works
Production usage	
Impact	A larger cache size allows more query plans to be cached. This increases the probability of finding a plan in cache that can be reused for a query, thereby reducing compile time. It does mean that the compiler will use more memory, but since there are usually not that many compilers running in a CPU, the negative effects may be minimal. However, you do need to know the amount of physical memory available on each node and the number of compilers that will run on a node (influenced by the number of concurrent connections configured to run on the cluster). If the cache size is disproportionally large, it is likely to result in reduced performance as the operating system may repeatedly swap the compiler (bloated by a huge cache) in and out of physical memory.
Level	Service
Conflicts/Synergies	You should be aware that the cache allocated is divided into text caching and template caching. Text caching gets approximately 25% of the cache memory. Query plan caching occurs prior to parsing (text-based caching) and after parsing (template-based caching). The compiler caches same- text queries as text cache hits. Same-text queries are queries whose SQL texts are identical in everything, including case and white space. By caching text-based queries, the compiler avoids redundant re- computation of previously compiled queries and improves performance by reducing compile times and increasing compiler throughput. The text cache is always searched first for a query. If the plan object is not produced due to a text cache miss, the plan is stored in the template cache if it meets the criteria for template caching.
Real Problem Addressed	
Introduced in Release	EsgynDB R1.0
Deprecated in Release	

11.14 TRAF_LOAD_ALLOW_RISKY_INDEX_MAINTENANCE

Description	Allow incremental index maintenance during bulk load.

Values	ON / OFF Default: 'OFF'
Usage	When this attribute is ON, during a bulk load, any indexes on a table are maintained incrementally. New rows are added to the base table and all the indexes in HFiles and then during LOAD COMPLETE phase all new files are moved to HBase. Indexes are not offline. However it will cause the index to be inconsistent with the base table if any of the new rows have the same key value as an existing. Change the default to ON only when certain that new rows will not have a conflict with existing rows in table.
Production usage	Yes, with previous caution.
Impact	Bulk load into tables with index is faster, when the attribute is set to ON.
Level	Load query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

11.15 TRAF_LOAD_FLUSH_SIZE_IN_KB

Description	Specify flush size used by bulk load when writing to HFiles.
Values	Positive integer Default: '1024'
Usage	If the system is not memory constrained specifying a larger value may make the write phase of LOAD statement to proceed faster. Consider using lower values if the table has several indexes and is memory constrained.
Production usage	Yes
Impact	Affects memory usage patterns and write performance of LOAD.
Level	Load query
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

12 Debugging

This section describes CQDs that are used for debugging EsgynDB processes.

12.1 UDR_DEBUG_FLAGS

Description	Use this only for debugging UDFs.
Values	0-n. Default: 0
Usage	See https://cwiki.apache.org/confluence/display/TRAFODION/Tutorial%3A+The+object-oriented+UDF+interface#Tutorial:Theobject-orientedUDFinterface-DebuggingUDFcode
Production usage	
Impact	
Level	
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

12.2 UDR_JVM_DEBUG_PORT

Description	Use this only for debugging UDFs.
Values	0-n. Default: 0
Usage	See https://cwiki.apache.org/confluence/display/TRAFODION/Tutorial%3A+The+object-oriented+UDF+interface#Tutorial:Theobject-orientedUDFinterface-DebuggingUDFcode
Production usage	
Impact	
Level	
Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	

12.3 UDR_JVM_DEBUG_TIMEOUT

Description	Use this only for debugging UDFs.
Values	0-n. Default: 0
Usage	See https://cwiki.apache.org/confluence/display/TRAFODION/Tutorial%3A+The+object-oriented+UDF+interface#Tutorial:Theobject-orientedUDFinterface-DebuggingUDFcode
Production usage	
Impact	
Level	

Conflicts/Synergies	
Real Problem Addressed	
Introduced in Release	EsgynDB R2.0
Deprecated in Release	