Heatwave Release Notes

Abstract

This document contains release notes for the changes in each release of HeatWave.

For additional HeatWave documentation, see MySQL HeatWave User Guide.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (https://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

For legal information, see the Legal Notices.

For help with using MySQL, please visit the MySQL Forums, where you can discuss your issues with other MySQL users.

Document generated on: 2023-11-28 (revision: 27609)
Preface and Legal Notices

This document contains release notes for the changes in each release of HeatWave.

Legal Notices

Copyright © 1997, 2023, Oracle and/or its affiliates.

License Restrictions

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD,
Heatwave Release Notes

Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Use of This Documentation

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Changes in MySQL HeatWave

Changes in HeatWave 8.2.0 (Not yet released, General Availability)

Version 8.2.0 has no release notes, or they have not been published because the product version has not been released.

Changes in HeatWave 8.1.0 (2023-07-18, General Availability)

Functionality Added or Changed

- CREATE TABLE now supports zonemaps for columns that are not defined as a primary key. (WL #15723)
• HeatWave now supports the `GET_FORMAT()`, `MAKETIME()`, `PERIOD_ADD()`, `PERIOD_DIFF()`, `SEC_TO_TIME()`, `SUBTIME()`, `SYSDATE()`, and `TIMEDIFF()` temporal functions. (WL #14958)

Changes in HeatWave 8.0.34 (2023-07-18, General Availability)

Version 8.0.34 has no release notes, or they have not been published because the product version has not been released.

Changes in HeatWave 8.0.33-u3 (2023-06-09, General Availability)

HeatWave Lakehouse

• MySQL HeatWave expands to include HeatWave Lakehouse, letting organizations process and query hundreds of terabytes of data residing in Object Storage—in a variety of file formats, such as CSV and Parquet.

The Lakehouse feature of MySQL HeatWave enables query processing on Object Storage-resident data. The source data is read from Object Storage, transformed to the HeatWave format, stored in the HeatWave persistence storage layer in OCI Object Storage, and then loaded to HeatWave Cluster memory.

• Provides in-memory query processing on Object Storage-resident data.
• Data is not loaded into the MySQL InnoDB storage layer.
• Supports structured and relational data in CSV and Parquet formats.
• With this feature, users can now analyse data in both InnoDB and an object store using familiar SQL syntax in the same query.

See HeatWave Lakehouse. (WL #15526)

Changes in HeatWave 8.0.33 (2023-04-27, General Availability)

• HeatWave AutoML

• Functionality Added or Changed

HeatWave AutoML

• HeatWave AutoML now supports a recommendation task. (WL #15416)

Functionality Added or Changed

• The new HeatWave Auto Parallel AutoUnload utility automates the process of unloading tables from HeatWave using an optimized number of parallel threads. (WL #15447)

• The following columns were added to the `performance_schema.rpd_query_stats` table:
  • `CONNECTION_ID`: The ID of the connection.
  • `STATEMENT_ID`: The global query ID.

(WL #15526)

Changes in HeatWave 8.0.32-u2 (2023-02-21, General Availability)
HeatWave AutoML

- HeatWave AutoML now supports unsupervised anomaly detection, which is the data mining task that finds unusual patterns in data. (WL #15518)

Changes in HeatWave 8.0.32-u1 (2023-02-21, General Availability)

- HeatWave AutoML
- Functionality Added or Changed

HeatWave AutoML

- HeatWave AutoML adds support for multivariate endogenous forecasting models, and exogenous forecasting models. (WL #15511)

Functionality Added or Changed

- The `rdp_tables` table now supports recovery time measurement with `RECOVERY_SOURCE`, `RECOVERY_START_TIMESTAMP`, and `RECOVERY_END_TIMESTAMP`. The `performance_schema.global_status` now includes the `rapid_recovery_time` variable. (WL #15441)
- Arithmetic operators are now supported with variable-length encoded string columns. Mathematical functions are now supported with variable-length columns. (WL #15405)

Changes in HeatWave 8.0.32 (2023-01-17, General Availability)

- Deprecation and Removal Notes
- HeatWave AutoML
- Functionality Added or Changed

Deprecation and Removal Notes

- MySQL 8.0.32 deprecates the `heatwave_load_report` and `heatwave_advisor_report` tables, and replaces them with the `heatwave_autopilot_report` table in the `sys` schema. A future release will remove them. (Bug #34727481)

HeatWave AutoML

- MySQL 8.0.32 introduces a number of changes for HeatWave AutoML routines:
  - The routines: `ML_PREDICT_ROW`, `ML_PREDICT_TABLE`, `ML_EXPLAIN_ROW`, and `ML_EXPLAIN_TABLE` now include the `ml_results` column, which uses JSON format to return the results.
  - The routines: `ML_PREDICT_ROW`, `ML_PREDICT_TABLE`, `ML_EXPLAIN_ROW`, and `ML_EXPLAIN_TABLE` now allow additional columns that are not required for prediction or explanation.
  - The routines: `ML_PREDICT_ROW`, and `ML_PREDICT_TABLE` now allow an `options` parameter in JSON format.
  - The `ML_TRAIN` routine also runs the `ML_EXPLAIN` routine with the default Permutation Importance model. (WL #15420)
Heatwave Release Notes

- MySQL 8.0.32 introduces progress tracking for `ML_TRAIN`. Use a second MySQL Client window to track the progress of `ML_TRAIN` with calls to the performance schema. It also introduces the `rapid_ml_operation_count` status variable. (WL #15384)

**Functionality Added or Changed**

- HeatWave now supports the `COMPRESS()`, `MD5()`, `RANDOM_BYTES()`, `SHA()`, `SHA1()`, `SHA2()`, `UNCOMPRESS()`, and `UNCOMPRESSED_LENGTH()` encryption and compression functions. (WL #15283)

- HeatWave now supports the `CRC32()`, `LOG2()`, `LOG10()`, and `RAND()` mathematical functions. (WL #15256)

**Changes in HeatWave 8.0.31 (2022-10-11, General Availability)**

- HeatWave AutoML

- Functionality Added or Changed

**HeatWave AutoML**

- HeatWave AutoML queries are now monitored and recorded in the Performance Schema tables `rpq_query_stats` and `rpq_exec_stats`. Where a single HeatWave AutoML query contains a number of sub-queries, there is one record in `rpq_query_stats` and multiple records in `rpq_exec_stats`. (WL #15243)

- New functions have been added to HeatWave AutoML to help you manage models:
  - When you run the `ML_TRAIN` routine on a training dataset, you can now specify a model handle to use for the model instead of the generated one.
  - A new column `notes` has been added to the `MODEL_CATALOG` table, which you can use to record notes about the models in your model catalog.
  - The new column `model_metadata` in the `MODEL_CATALOG` table records metadata for models, such as the training score, training time, and information about the training dataset. If an error occurs during training or you cancel the training operation, HeatWave AutoML records the error status in this column. (WL #15243)

- HeatWave AutoML now supports the upload of pre-trained models in ONNX (Open Neural Network Exchange) format to the model catalog. You can load them using the stored procedure `ML_MODEL_IMPORT` that provides the conversion required to store the model in a MySQL table. (WL #15243)

- A new stored procedure `ML_EXPLAIN` lets you train a variety of model explainers and prediction explainers for HeatWave AutoML, in addition to the default Permutation Importance model and prediction explainers:
  - The Partial Dependence model explainer shows how changing the values of one or more columns will change the value that the model predicts.
  - The SHAP model explainer produces global feature importance values based on Shapley values.
  - The Fast SHAP model explainer is a subsampling version of the SHAP model explainer which usually has a faster runtime.
Heatwave Release Notes

- The Permutation Importance prediction explainer explains the prediction for a single row or table.
- The SHAP prediction explainer uses feature importance values to explain the prediction for a single row or table.

When you use the `ML_EXPLAIN_TABLE` and `ML_EXPLAIN_ROW` stored procedures to generate explanations for a prediction, you can now use the SHAP prediction explainer as an alternative to the default Permutation Importance prediction explainer. SHAP produces feature importance values (explanations) based on Shapley values. (WL #15243)

- HeatWave AutoML now supports timeseries forecasting using the existing stored procedures `ML_TRAIN`, `ML_PREDICT_TABLE`, and `ML_SCORE`. You can create a forecast for a single column (a univariate endogenous variable) with a numeric data type. The forecasting task is specified as a JSON object when you call the `ML_TRAIN` stored procedure. (WL #15243)

### Functionality Added or Changed

- HeatWave uses dictionary encoding to compress string columns (CHAR, VARCHAR, TEXT). These dictionaries are built for each string column with the `RAPID_COLUMN=ENCODING=SORTED` keyword. HeatWave now supports 8.5 billion dictionary entries (up from 4 billion), which means HeatWave can now encode string columns with number of distinct value (NDV) up to 8.5 billion. (WL #14742)

- MySQL HeatWave now uses zone maps to exclude data chunks that are not relevant for a query. The zone map stores per chunk statistics for the minimum and maximum values of primary key columns. Queries using point and range filters to filter on values can now get accelerated by HeatWave by an order of magnitude. This is particularly useful for improving range queries in OLAP and mixed workloads. (WL #14713)

- A new hypergraph-based MySQL optimizer is introduced for HeatWave to provide a holistic cost model across MySQL and HeatWave, create better query plans based on statistics used in MySQL Autopilot, reduce compilation time, eliminate the need of query hints for join order, and improve join query performance. With the new optimizer, HeatWave can now run all 22 TPC-H queries without straight join hints. Before 8.0.31, a straight join hint is needed for 10 out of 22 TPC-H to reach peak performance. (WL #14449)

- DDL statements such as `ALTER TABLE`, `RENAME TABLE`, and `TRUNCATE TABLE` are now permitted on a table that has RAPID defined as the secondary engine. If a DDL operation is successfully carried out on a table that is loaded to a HeatWave Cluster at the time, HeatWave automatically reloads the table from InnoDB. Note that if the DDL operation makes the table’s structure incompatible with HeatWave, the table is unloaded from the HeatWave Cluster. (WL #15129)

### Changes in HeatWave 8.0.30-u1 (2022-09-06, General Availability)

#### Functionality Added or Changed

- HeatWave now supports the `ABS()`, `POWER()`, and `SIGN()` mathematical functions. (WL #15246)

- HeatWave now supports the `GROUP_CONCAT()` aggregation function with variable-length (VARLEN) string columns. (WL #14767)

- Change propagation memory management was enhanced to improve HeatWave shutdown time. (WL #15306)

- Query performance was optimized for compressed data. Decompression of data that is queried but does not participate in filter evaluation is delayed until after a filter is applied. With this optimization, decompression is avoided entirely for data that is filtered out. (WL #15169)
Changes in HeatWave 8.0.30 (2022-07-26, General Availability)

- **Advisor**
- **HeatWave AutoML**
- **Functionality Added or Changed**

**Advisor**

- HeatWave Advisor Auto Encoding, which recommends string column encodings, now provides encoding recommendations that optimize query performance. Recommendations are based on performance models that use query execution data. Previously, string column encoding recommendations were optimized for cluster memory usage only. A performance improvement estimate is provided with string column encoding recommendations. (Bug #34145862)

**HeatWave AutoML**

- You can now train HeatWave AutoML models on tables containing DATE, TIME, DATETIME, TIMESTAMP, and YEAR data types. (Bug #33895503)
- HeatWave AutoML now generates a model explanation when you train a machine learning model. Model explanations help identify the features that are most important to a model. For more information, see The Model Catalog.

The following columns were added to the MODEL_CATALOG table:

- **column_names**: The feature columns used to train the model.
- **last_accessed**: The last time the model was accessed. HeatWave AutoML routines update this value to the current timestamp when accessing the model.
- **model_explanation**: The model explanation generated during training.
- **model_type**: The type of model (algorithm) selected by ML_TRAIN to build the model.
- **task**: The task type specified in the ML_TRAIN query (classification or regression).

ML_PREDICT_* and ML_EXPLAIN_* routine performance was improved, resulting in faster prediction and explanation processing. (WL #15088, WL #15014)

- The following HeatWave AutoML enhancements were implemented:

  - **ML_TRAIN** options for advanced users. These options permit users to customize various aspects of the ML training pipeline including algorithm selection, feature selection, and hyperparameter optimization.
    - The model_list option permits specifying the type of model to be trained.
    - The exclude_model_list option specifies models types to exclude from consideration during model selection.
    - The optimization_metric option specifies the scoring metric to optimize for when training a machine learning model.
    - The exclude_column_list option specifies feature columns to exclude from consideration when training a machine learning model.
Heatwave Release Notes

For more information, see Advanced ML_TRAIN Options.

- Support was added for Support Vector Machine SVC and LinearSVC classification and regression models. For a complete list of supported model types, see Model Types.

- The ML_TRAIN routine now reports a message if a trained model does not meet expected quality criteria.

- ML_EXPLAIN_ROW and ML_EXPLAIN_TABLE routines now provide information to help interpret explanations. The routines also report a warning when a model quality issue is detected, enabling users to revisit their data in order to improve model quality.

(WL #15089)

Functionality Added or Changed

- The amount of heap memory allocated on the MySQL node for each table loaded into HeatWave was reduced, increasing the maximum number of tables that can be loaded. For MySQL.HeatWave.VM.E3.Standard shapes, the maximum was raised from 100k tables to 400k tables. For MySQL.HeatWave.BM.E3.Standard shapes, the maximum number was raised from 400k tables to 1600k tables. The actual number of tables that can be loaded is dependent on the table's data. (Bug #33951708)

- The performance_schema.rpd_column_id table was modified to remove redundant data. The NAME, SCHEMA_NAME, TABLE_NAME columns were removed, and a TABLE_ID column was added. (Bug #33899183)

- Support was added for the FROM_DAYS() date and time function, and GREATEST() and LEAST() comparison functions now support DATE, DATETIME, TIME, and TIMESTAMP columns. (WL #14956)

- Support was added for built-in server-side data masking and de-identification to help protect sensitive data from unauthorized uses by hiding and replacing real values with substitutes. Data masking and de-identification operations are performed on the server, and queries involving data masking and de-identification functions are accelerated by HeatWave. The following data masking and de-identification functions are supported:
  - gen_range()
  - gen_rnd_email()
  - gen_rnd_pan()
  - gen_rnd_ssn()
  - gen_rnd_us_phone()
  - mask_inner()
  - mask_outer()
  - mask_pan()
  - mask_pan_relaxed()
  - mask_ssn()

See Data Masking and De-Identification Functions. (WL #15143)
• Optimizations were implemented to improve performance for `JOIN` and `GROUP BY` queries with execution plans involving multiple consecutive rounds of data partitioning. (WL #15143)

• `expr IN (value,...)` comparisons, where the expression is a single value and compared values are constants of the same data type and encoding, have been optimized. For example, the following `IN()` comparison has been optimized:

```
SELECT * FROM Customers WHERE Country IN ('Germany', 'France', 'Spain');
```

(WL #14952)

Changes in HeatWave 8.0.28-u3 (2022-04-19, General Availability)

Functionality Added or Changed

• Tables that have become stale due to a change propagation failure resulting from an out-of-code error are now automatically reloaded. A check for stale tables is performed periodically when the HeatWave Cluster is idle. Previously, identifying and reloading stale tables was a manual process. See Change Propagation. (WL #14914)

Changes in HeatWave 8.0.28-u2 (2022-03-29, General Availability)

• HeatWave AutoML

• Functionality Added or Changed

HeatWave AutoML

• HeatWave customers now have access to HeatWave AutoML, which is a fully managed, highly scalable, cost-efficient, machine learning solution for data stored in MySQL. HeatWave AutoML provides a simple SQL interface for training and using predictive machine learning models, which can be used by novice and experienced ML practitioners alike. With HeatWave AutoML, you can train a model with a single call to an SQL routine. Similarly, you can generate predictions with a single `CALL` or `SELECT` statement which can be easily integrated with your applications.

With HeatWave AutoML, data and models never leave the MySQL Database Service, saving you time and effort while keeping your data and models secure. HeatWave AutoML is optimized for HeatWave shapes and scaling, and all HeatWave AutoML processing is performed on the HeatWave Cluster. ML computation is distributed among HeatWave nodes, taking advantage of HeatWave's scalability and massively parallel processing capabilities. For more information about HeatWave's machine learning capabilities, see HeatWave AutoML. (WL #14661, WL #14836, WL #15014)

Functionality Added or Changed

• HeatWave now compresses data as it is loaded, which permits HeatWave nodes to store more data. More data per node reduces costs by minimizing the size of the HeatWave Cluster required to store your data. Data compression is enabled by default but can be disabled at runtime using the `rapid_compression` session variable. For more information, see Data Compression. (WL #14868)

Changes in HeatWave 8.0.28-u1 (2022-02-15, General Availability)

Functionality Added or Changed

• HeatWave now supports up to 1017 columns for base relations (tables as loaded into HeatWave), and up to 1800 columns for intermediate relations (intermediate tables used during query processing). The
maximum column width for base relations and intermediate relations was increased to 65532 bytes. See Column Limits. (WL #14918)

Changes in HeatWave 8.0.27-u3 (2021-12-15, General Availability)

Functionality Added or Changed

• Support was added for the `CONVERT_TZ()` and `LAST_DAY()` functions, which are used to manipulate temporal values. (WL #14768)

• The HeatWave Cluster recovery process was optimized to avoid applying a large volume of changelogs during recovery. Snapshots are now taken when the volume of changelogs and the time required to apply those changes exceed specific thresholds, and recovery from Object Storage is performed using those snapshots. (WL #14615)

• HeatWave now supports automatic data reload when the HeatWave Cluster is restarted. Previously, when a HeatWave Cluster was stopped by a stop or restart action, data had to be reloaded manually after the cluster was restarted. Now, when starting or restarting a HeatWave Cluster, data that was previously loaded is reloaded automatically. Data changes that occur on the DB System while the HeatWave Cluster is offline are included in the reloaded data.

Automatic data reload does not occur if the HeatWave Cluster was stopped as a result of a stop or restart action performed on the DB System to which the HeatWave Cluster is attached. In this case, data loaded in the HeatWave Cluster must be reloaded manually after the HeatWave Cluster is restarted. (WL #14729)

Changes in HeatWave 8.0.27-u2 (2021-12-07, General Availability)

Functionality Added or Changed

• HeatWave now supports querying views. See Using Views. (WL #13568)

• Bloom filter join optimizations were introduced. For HeatWave queries that join large and small relations, bloom filters reduce the amount of data processed by early filtering and the amount of memory used during query processing. (WL #14752)

• The `rpd_query_stats` table, which stores HeatWave query history (compilation and execution statistics), now stores data for the last 1000 executed queries. Previously, data was stored for the last 200 queries.

The following columns were added to the `performance_schema.rpd_tables` table:

• `SIZE_BYTES`: The amount of data loaded per table, in bytes.

• `QUERY_COUNT`: The number of queries that referenced the table.

• `LAST_QUERIED`: The timestamp of the last query that referenced the table.

• `LOAD_END_TIMESTAMP`: The load completion timestamp for the table.

The following column was added to the `performance_schema.rpd_columns` table:

• `DICT_SIZEBYTES`: The dictionary size per column, in bytes.

The following column was added to the `performance_schema.rpd_nodes` table:

• `BASEREL_MEMORY_USAGE`: The base relation memory footprint per node.
The rapid_query_stats and rpd_exec_stats tables are now synchronized. If a query record is removed from the rapid_query_stats table, it is also removed from the rpd_exec_stats table. (WL #14759)

Changes in HeatWave 8.0.26-u2 (2021-09-21, General Availability)

Functionality Added or Changed

• The following function support was added:
  • YEARWEEK(date), YEARWEEK(date, mode)
  • The mode argument for the two-argument form of the WEEK() function: WEEK(date[, mode])
  • MAKEDATE()
  • “Zero” handling for dates such as ‘2001-11-00’ was implemented for WEEK(), YEARWEEK(), and MAKEDATE() functions.
  • CAST() of FLOAT and DOUBLE values to DECIMAL
    (Bug #33163625, Bug #33138534, WL #14714)
  • The new hw_data_scanned global status variable tracks the total cumulative megabytes scanned by successfully executed HeatWave queries.

The number of megabytes scanned by an individual HeatWave query can be obtained by querying the performance_schema.rpd_query_stats table.

An estimated number of megabytes scanned by an individual query can be obtained by running the query with EXPLAIN and querying the performance_schema.rpd_query_stats table.

For more information, see Scanned Data Monitoring. (WL #14738)

Changes in HeatWave 8.0.26-u1 (2021-08-10, General Availability)

• HeatWave Network Layer
  • HeatWave Data Management Layer
  • Functionality Added or Changed

HeatWave Network Layer

• HeatWave network layer optimizations have improved scalability and network performance. (WL #14513)

HeatWave Data Management Layer

• Data loaded into HeatWave, including propagated changes, are now persisted to OCI Object Storage for recovery in case of a HeatWave node or cluster failure. Previously, data was recovered from the MySQL DB System. Loading data from OCI Object Storage is faster because data does not need to be converted to the HeatWave storage format, as is required when loading data from the MySQL DB System. If data recovery from OCI Object Storage fails, HeatWave falls back to recovering data from the MySQL DB System. Data removed from HeatWave when a table is unloaded is removed from OCI
Object Storage in a background operation. For related information, see HeatWave Cluster Failure and Recovery. (WL #14478, WL #14046, WL #14541)

Functionality Added or Changed

- HeatWave now supports `COUNT(NULL)`, except in cases where it is used as an input argument for non-aggregate operators. (Bug #33005146)

- Full support was added for the `DISTINCT` modifier. Previously, multiple instances of `(DISTINCT value)` expressions in a query were only permitted if the same `value` was specified. (Bug #32865043, Bug #33007714, WL #14574)

- HeatWave now supports the `WITH ROLLUP` modifier in `GROUP BY` clauses. (WL #14533)

- HeatWave now supports window functions. For optimal performance, window functions in HeatWave utilize a massively parallel, partitioning-based algorithm. For more information, see Window Functions. (WL #14674)

Changes in HeatWave 8.0.26 (2021-07-23, General Availability)

- Advisor
- Auto Parallel Load
- Auto Scheduling
- Functionality Added or Changed

Advisor

The new HeatWave Advisor provides string column encoding and data placement key recommendations based on machine learning models, data analysis, and HeatWave query history. Implementing HeatWave Advisor recommendations can improve query performance and reduce the amount of memory required on HeatWave nodes.

The HeatWave Advisor also provides a Query Insights feature, which provides runtimes for successfully executed queries, and runtime estimates for EXPLAIN queries, queries canceled using Ctrl+C, and queries that fail due to out of memory errors. Runtime data is useful for query optimization, troubleshooting, and estimating the cost of running a particular query or workload.

The HeatWave Advisor is implemented as a stored procedure named `heatwave_advisor`, which resides in the MySQL `sys` schema. Running Advisor involves issuing a CALL statement for the stored procedure with optional arguments.

```
CALL sys.heatwave_advisor (options);
```

For more information about the HeatWave Advisor, see Workload Optimization using Advisor. (WL #14328, WL #14510, WL #14431, WL #14328, WL #14651)

Auto Parallel Load

- The new HeatWave Auto Parallel Load utility automates the process of preparing and loading tables into HeatWave and loads data using an optimized number of parallel load threads.

The HeatWave Auto Parallel Load utility is implemented as a stored procedure named `heatwave_load`, which resides in the MySQL `sys` schema. Running Auto Parallel Load involves issuing a CALL statement for the stored procedure, which takes a list of schemas and options as arguments.
CALL sys.heatwave_load (db_list, options);

For more information about the HeatWave Auto Parallel Load utility, see Loading Data Using Auto Parallel Load. (WL #14149)

Auto Scheduling

• The HeatWave query scheduling algorithm was improved. The revised algorithm prioritizes queries based on estimated cost and wait time in the queue, which enables dynamic, workload-aware query prioritization. Previously, queries were prioritized using a static cost-based prioritization model. (WL #14608)

Functionality Added or Changed

• `DATE_ADD()` and `DATE_SUB()` functions now support precision `INTERVAL` values (`DECIMAL`, `DOUBLE`, and `FLOAT`). (Bug #32725985, Bug #32438123)

• Support was added for multiple instances of `COUNT(DISTINCT)` in a query. (Bug #32422984)

• Query compilation and processing was improved to permit combining aggregate operators into a single task in the physical query plan, which avoids fully materializing intermediate result sets. This enhancement reduces memory allocation and deallocation operations, memory usage, and execution time for affected queries. (WL #14614)

• The cost model that estimates HeatWave query runtimes can now use statistics from previously executed queries, which improves the accuracy of query runtime estimates. (WL #14546)

• HeatWave now supports `CREATE TABLE ... SELECT` statements where the `SELECT` query is offloaded to HeatWave and the table is created on the MySQL Database Service instance. This feature improves `CREATE TABLE ... SELECT` performance in cases where the `SELECT` portion of the statement is a long running, complex query. For more information, see CREATE TABLE ... SELECT Statements. (WL #14516)

• Support was added for `REGEXP_REPLACE()` and `REGEXP_SUBSTR()` regular expression functions, and error messaging was improved for `REGEXP()` function syntax mismatches, expression errors, and input argument errors. (WL #14641)

Changes in HeatWave 8.0.25 (2021-05-11, General Availability)

Functionality Added or Changed

• Support was added for `CAST()` of `ENUM` column values to `CHAR` or `VARCHAR` where the `ENUM` value is cast to a `FLOAT` value, as in the following example:

```sql
SELECT CAST(CAST(enum_col AS FLOAT) AS CHAR(3)) FROM tbl_name;
```

(Bug #32618454)

• Support was added for `SELECT DISTINCT` queries that order the result set by a column that is not defined in the `SELECT` list. For example, the following query can now be offloaded to HeatWave for execution:

```sql
SELECT DISTINCT a FROM t1 ORDER BY c DESC;
```

(Bug #32583856)

• Query plan statistics are now collected and stored in a statistics cache when a query is executed in HeatWave. When a new query shares query execution plan nodes with previously executed queries,
the actual statistics collected from previously executed queries are used instead of estimated statistics, which improves query execution plans, cost estimations, execution times, and memory efficiency.

The statistics cache is an LRU structure. When cache capacity is reached, the least recently used entries are evicted from the cache as new entries are added. The maximum number of entries permitted in the statistics cache is defined by the `rapid_stats_cache_max_entries` setting. The number of entries permitted by default is 65536, which is enough to store statistics for 4000 to 5000 unique queries of medium complexity. (WL #14503)

- Support was added for:
  - `CAST() AS YEAR`. Both variable-length and dictionary-encoded string columns values are supported.
  - The `FORMAT()` function. Variable-length-encoded string columns are not supported.

(WL #14511)

Changes in HeatWave 8.0.24 (2021-04-20, General Availability)

Functionality Added or Changed

- Comparison of different temporal type values is now supported. For example, a query that compares `DATE` values to `TIMESTAMP` values can now be offloaded to HeatWave. (Bug #32420986)

- Range operators on `VARLEN`-encoded string columns are now supported. For example, the following query, where `L_LINESTATUS` is a `VARLEN`-encoded string column, can now be offloaded to HeatWave:

```sql
SELECT COUNT(*) FROM lineitem WHERE L_LINESTATUS >= 1 and L_LINESTATUS <= 10;
```

(Bug #31721399)

- HeatWave now supports `INSERT ... SELECT` statements where the `SELECT` query is offloaded to HeatWave and the result set is inserted into a table on the MySQL Database Service instance. This feature improves `INSERT ... SELECT` performance in cases where the `SELECT` portion of the statement is a long running, complex query. For more information, see INSERT ... SELECT Statements. (WL #14299)

- `VARLEN`-encoded columns are now supported as data placement keys. For information about the data placement feature, see Defining Data Placement Keys. (WL #14491)

- Failure handling was improved for queries involving unsupported internal data types. Such queries now exit with an error indicating that the internal data type of the query is not supported. (WL #14483)

Changes in HeatWave 8.0.23-u2 (2021-03-15, General Availability)

Functionality Added or Changed

- Support was added for the following aggregate functions:
  - `STD()`
  - `STDDEV()`
  - `STDDEV_POP()`
  - `STDDEV_SAMP()`
  - `VAR_POP()`
Heatwave Release Notes

• VAR_SAMP ()
• VARIANCE ()

See Aggregate Functions. (WL #14479)

• HeatWave now uses a priority-based scheduling mechanism based on query cost estimates to schedule queries for execution. Previously, queries were executed in the order of arrival. The scheduling mechanism prioritizes short running queries over long running queries to reduce overall query execution wait times. For more information, see Auto Scheduling. (WL #14423)

Changes in HeatWave 8.0.23-u1 (2021-02-09, General Availability)

Functionality Added or Changed

• String column encoding support was added for TEXT-type columns. See Encoding String Columns. (WL #14430)

• UNION and UNION ALL support was extended. The clauses are now supported at any location in a query that is permitted by MySQL. (WL #14455)

• The following date and time functions are now supported:
  • TO_SECONDS ()
  • UNIX_TIMESTAMP ()
  • FROM_UNIXTIME ()
  • TIME_TO_SEC ()

See Date and Time Functions.

The following date and time functions are now supported with VARLEN-encoded columns:

• TO_DAYS ()
• DAYOFYEAR ()
• QUARTER ()
• TO_SECONDS ()

See Date and Time Functions.

The following string functions are now supported with VARLEN-encoded columns:

• ORD ()
• ASCII ()

See String Functions and Operators.

SET timezone = timezone with the timezone value specified as an offset from UTC in the form of [H]H:MM and prefixed with a + or – is now supported with the UNIX_TIMESTAMP () and FROM_UNIXTIME () functions. (WL #14345)
Offset is now supported with the `LIMIT` clause:

```
SELECT * FROM tbl LIMIT offset, row_count;
```

The PostgreSQL syntax is also supported:

```
SELECT * FROM tbl LIMIT row_count OFFSET offset;
```

(WL #14341)

New Performance Schema tables provide access to query and execution statistics:

- `performance_schema.rpd_exec_stats`
- `performance_schema.rpd_query_stats`

Changes to HeatWave Performance Schema tables:

- An `NDV` (Number of Distinct Values) column was added to the `performance_schema.rpd_columns` table.
- A `ROWS` column that shows the total number of rows in a table was added to the `performance_schema.rpd_tables` table.
- A `MEMORY_USAGE` column that shows node memory usage was added to the `performance_schema.rpd_columns` table.
- The `performance_schema.rpd_nodes DRAM` column was renamed to `MEMORY_TOTAL`. The `MEMORY_TOTAL` column shows the total memory allocated to a HeatWave node.

See [HeatWave Performance Schema Tables](#). (WL #14386)
D
data compression, 7, 10
data placement, 15
data types, 15
DATE_ADD(), 13
DATE_SUB(), 13
DAYOFYEAR(), 16
DISTINCT, 12

F
FORMAT(), 14
FROM_DAYS(), 8
FROM_UNIXTIME(), 16

G
GREATEST(), 8
GROUP BY, 12
GROUP_CONCAT(), 7

H
heap segment size, 8

I
IN(), 8
INSERT ... SELECT, 15

L
LAST_DAY, 11
LEAST(), 8
LIMIT, 16
loading data, 8

M
machine learning, 4, 4, 5, 5, 6, 8, 10
MAKEDATE(), 12

N
networking, 12

O
Object Storage, 11
OFFSET, 16
ORD(), 16

P
partitioning, 8
Performance Schema, 8, 11
planned shutdown, 11
POWER(), 7

Q
QUARTER(), 16
queries, 12, 15
query cost model, 13
query scheduling, 13, 15

R
rapid_stats_cache_max_entries, 14
recovery, 11, 12
REGEXP_REPLACE(), 13
REGEXP_SUBSTR(), 13
rpd_cloumns table, 16
rpd_column_id table, 8
rpd_exec_stats table, 16
rpd_nodes table, 16
rpd_query_stats, 11
rpd_query_stats table, 4, 16

S
scalability, 12
SELECT DISTINCT, 14
SET timezone, 16
SIGN(), 7
statistics, 14
string column encoding, 15

temporal data types, 8
temporal type comparison, 15
TEXT, 16
TIME_TO_SEC(), 16
TO_DAYS(), 16
TO_SECONDS(), 16

U
UNION, 16
UNION ALL, 16
UNIX_TIMESTAMP(), 16

V
variable-length encoding, 15
VARLEN encoding, 16
views, 11

W
WEEK(), 12
window functions, 12
WITH ROLLUP, 12

Y
YEARWEEK(), 12