opentext™

25.1 新機能紹介

2025年1月15日 | Analytics & AI

サポートプラットフォーム

サポートが開始されるプラットフォーム

- AlmaLinux 9

インストール

Transparent hugepages設定値の確認廃止

旧サポートOS向けに行っていたTransparent hugepages設定値の確認を廃止。

データベース管理

AdminToolsを用いたログローテーションサービスへの移行

AdminToolsにin_server_logrotateツールが追加され、Linuxのlogrotateコマンドを用いたログローテーションからLogRorateサービスへの移行が行えるようになる。

```
$ admintools -t in_server_logrotate -d <database name> -p <password>
```

データベース管理

Health Watchdog

Health Watchdogの監視対象に以下が追加される。

- メモリ使用量

  メモリ使用量が以下の条件に当てはまらないことを監視。

  - general Resource Poolの空き容量の割合が設定パラメータ GeneralPoolMinFreeMemoryRatio を下回る。
  - Global poolの使用量の割合が設定パラメータ GlobalPoolMaxUtilizationRatio を超える。

```
=> SELECT check_cluster_health();
                                    check_cluster_health
-------------------------------------------------------------------------------------
 Cluster State: Not Healthy.
 Reason: MEMORY PUSHBACK
 Description: High Memory Pressure. Global pool memory utilization exceeds its allowed max
              utilization.
 Number of blocked transactions: 1
 Hint: To view all blocked transactions, check out the system table
       health_watchdog_blocked_transactions. Try to resubmit request later or Tune the value of
       the database parameter GlobalPoolMaxUtilizationRatio to proceed.
```

データベース管理

Health Watchdog

一時中断されたトランザクションの履歴を保持する dc_health_watchdog_history テーブルと、それを参照する health_watchdog_events システムビューが追加される。

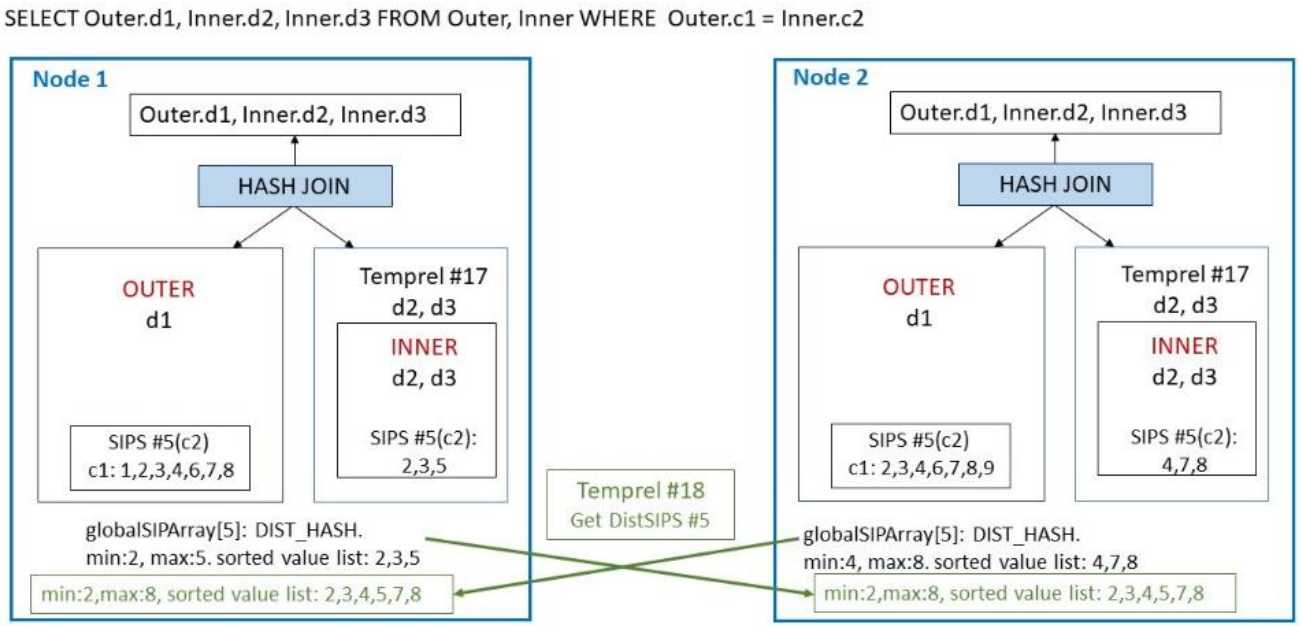
```
=> SELECT transaction_description, block_start_time, block_end_time, block_type, transaction_state
-> FROM dc_health_watchdog_history;

      transaction_description        |  block_start_time   |  block_end_time     |    block_type    |   transaction_state
-------------------------------------+---------------------+---------------------+------------------+-----------------------
 Txn: a00000000002f5 'create table …' | 2024-10-18 09:20:13 | 2024-10-18 09:20:14 | GCLX QUEUE BLOAT | Canceled while blocked
 Txn: a00000000002f5 'create table …' | 2024-10-18 09:20:13 |                     | GCLX QUEUE BLOAT | Transaction Blocked
```

SQLステートメント・ファンクション

Distributed SIPS

SIPS（Sidewise Information Passing）は、テーブル結合の際、Outer側がJoin KeyでSegmentedされておらず、データの Resegment または Broadcast が必要な場合、SIPSの適用ができない制約がある。
これを解消するため、それぞれのNodeで生成されたSIPS情報をNode間で共有し、OuterのSegmentationに関わらずSIPSを適用することでパフォーマンス向上を図るDistributed SIPSを導入。



SQLステートメント・ファンクション

Distributed SIPS

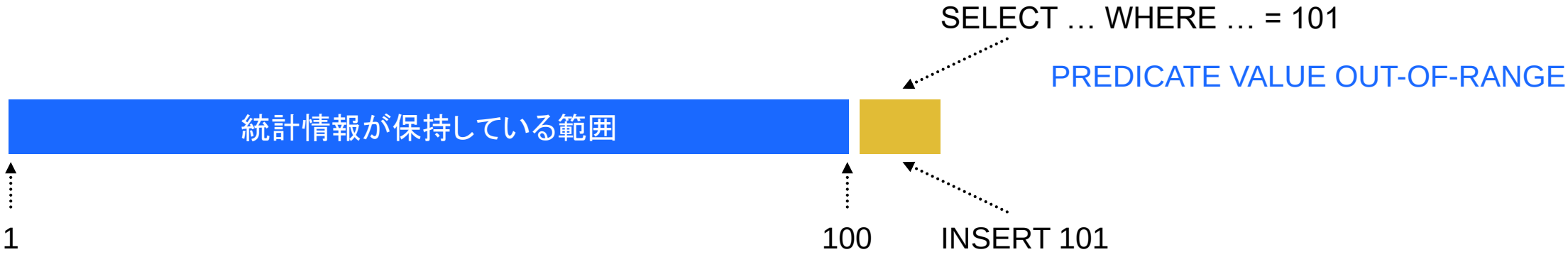
```
=> SELECT COUNT(*) FROM (WITH ct AS (SELECT * FROM bar ORDER BY a) SELECT * FROM foo JOIN ct ON foo.a = ct.a) tt;

 Access Path:
 +-GROUPBY NOTHING [Cost: 378, Rows: 1] (PATH ID: 1)
 |  Aggregates: count(*)
 |  Execute on: All Nodes
 | +---> JOIN HASH [Cost: 357, Rows: 100K] (PATH ID: 2) Outer (RESEGMENT)(LOCAL ROUND ROBIN) Inner (RESEGMENT)
 | |      Join Cond: (foo.a = ct.a)
 | |      Execute on: All Nodes
 | | +-- Outer -> STORAGE ACCESS for foo [Cost: 110, Rows: 100K] (PATH ID: 3)
 | | |      Projection: default_namespace.public.foo_super
 | | |      Materialize: foo.a
 | | |      Filter: (foo.a IS NOT NULL)
 | | |      Execute on: All Nodes
 | | |      Runtime Filter: (SIP1(DistHashJoin): foo.a)
 | | +-- Inner -> SELECT [Cost: 33, Rows: 1K] (PUSHED GROUPING) (PATH ID: 4)
 | | |      Execute on: All Nodes
 | | | +---> STORAGE ACCESS for bar [Cost: 33, Rows: 1K] (PATH ID: 6)
 | | | |      Projection: default_namespace.public.bar_super
 | | | |      Materialize: bar.a
 | | | |      Execute on: All Nodes
```

SQLステートメント・ファンクション

PREDICATE VALUE OUT-OF-RANGEの廃止

クエリを実行する際には取得済みの統計情報を使用する。
統計情報は各カラムの最小値・最大値を保持しており、例えば WHERE句に指定される条件の値がこの範囲を外れている場合、Optimizerは PREDICATE VALUE OUT-OF-RANGE と判断し、統計情報が使用できない場合と同等の手順で実行プランを作成するため、最適ではない。



IDや日付など値が増加し続けるカラムは容易に範囲外となることが多く、かつ、大規模テーブルの場合は都度統計情報の収集を行うことが難しい。これを回避するため、範囲外の値を条件に用いられた場合、この値を最小値・最大値と見なして既存の統計情報を使用して実行プランを作成する。

SQLステートメント・ファンクション

PREDICATE VALUE OUT-OF-RANGEの廃止

統計情報の確認:

```
=> SELECT export_statistics('', 'online_sales.online_sales_fact', 'sale_date_key');

<?xml version="1.0" encoding="utf-8"?>
<schema>
  <tables>
    <table name="online_sales_fact" schema="online_sales">
      <columns>
        <column name="sale_date_key" attrNum="1" dataPages="9773" dataType="Integer" encoding="NONE">
          <intStats>
            <distinct value="1811" />
            <buckets value="100" />
            <rows value="80000000" />
            <minValue>1</minValue>
            <maxValue>1826</maxValue>
            .....
```

※ maxValue = 1826 は2007年12月31日を表す。

SQLステートメント・ファンクション

PREDICATE VALUE OUT-OF-RANGEの廃止

統計情報の確認:

```
=> SELECT export_statistics('', 'public.date_dimension', 'date_key');

<?xml version="1.0" encoding="utf-8"?>
<schema>
  <tables>
    <table name="date_dimension" schema="public">
      <columns>
        <column name="date_key" attrNum="1" dataPages="1" dataType="Integer" encoding="NONE">
          <intStats>
            <distinct value="1826" />
            <buckets value="100" />
            <rows value="1826" />
            <minValue>1</minValue>
            <maxValue>1826</maxValue>
            .....
```

※ maxValue = 1826 は2007年12月31日を表す。

SQLステートメント・ファンクション

PREDICATE VALUE OUT-OF-RANGEの廃止

［以前：統計情報が使用できるケース］

```
=> SELECT d.calendar_year_quarter, SUM(o.sales_dollar_amount) sales_dollar_sum
-> FROM online_sales.online_sales_fact o JOIN public.date_dimension d ON o.sale_date_key = d.date_key
-> WHERE d.calendar_year = 2007 GROUP BY 1 ORDER BY 1;

 +-GROUPBY HASH (SORT OUTPUT) [Cost: 384K, Rows: 21] (PATH ID: 2)
 |  Aggregates: sum(o.sales_dollar_amount)
 |  Group By: d.calendar_year_quarter
 | +---> JOIN HASH [Cost: 375K, Rows: 15M] (PATH ID: 3)
 | |      Join Cond: (o.sale_date_key = d.date_key)
 | |      Materialize at Output: o.sales_dollar_amount
 | | +-- Outer -> STORAGE ACCESS for o [Cost: 188K, Rows: 80M] (PATH ID: 4)
 | | |      Projection: online_sales.online_sales_fact_super
 | | |      Materialize: o.sale_date_key
 | | |      Runtime Filter: (SIP1(HashJoin): o.sale_date_key)
 | | +-- Inner -> STORAGE ACCESS for d [Cost: 102, Rows: 347] (PATH ID: 5)
 | | |      Projection: public.date_dimension_super
 | | |      Materialize: d.date_key, d.calendar_year_quarter
 | | |      Filter: (d.calendar_year = 2007)
```

SQLステートメント・ファンクション

PREDICATE VALUE OUT-OF-RANGEの廃止

［以前：条件の値が範囲外のため統計情報が使用できないケース］

- 2008年1月1日のデータを追加

```
=> SELECT d.calendar_year_quarter, SUM(o.sales_dollar_amount) sales_dollar_sum
-> FROM online_sales.online_sales_fact o JOIN public.date_dimension d ON o.sale_date_key = d.date_key
-> WHERE d.calendar_year = 2008 GROUP BY 1 ORDER BY 1;

 +-GROUPBY HASH (SORT OUTPUT) [Cost: 375K, Rows: 20 (PREDICATE VALUE OUT-OF-RANGE)] (PATH ID: 2)
 |  Aggregates: sum(o.sales_dollar_amount)
 |  Group By: d.calendar_year_quarter
 | +---> JOIN HASH [Cost: 375K, Rows: 44K (PREDICATE VALUE OUT-OF-RANGE)] (PATH ID: 3)
 | |      Join Cond: (o.sale_date_key = d.date_key)
 | |      Materialize at Output: o.sales_dollar_amount
 | | +-- Outer -> STORAGE ACCESS for o [Cost: 188K, Rows: 80M] (PATH ID: 4)
 | | |      Projection: online_sales.online_sales_fact_super
 | | |      Materialize: o.sale_date_key
 | | |      Runtime Filter: (SIP1(HashJoin): o.sale_date_key)
 | | +-- Inner -> STORAGE ACCESS for d [Cost: 52, Rows: 1 (PREDICATE VALUE OUT-OF-RANGE)] (PATH ID: 5)
 | | |      Projection: public.date_dimension_super
 | | |      Materialize: d.date_key, d.calendar_year_quarter
 | | |      Filter: (d.calendar_year = 2008)
```

SQLステートメント・ファンクション

PREDICATE VALUE OUT-OF-RANGEの廃止

［今後：条件の値が範囲外であっても統計情報を使用するケース］

- 2008年1月1日のデータを追加

```
=> SELECT d.calendar_year_quarter, SUM(o.sales_dollar_amount) sales_dollar_sum
-> FROM online_sales.online_sales_fact o JOIN public.date_dimension d ON o.sale_date_key = d.date_key
-> WHERE d.calendar_year = 2008 GROUP BY 1 ORDER BY 1;

 +-GROUPBY HASH (SORT OUTPUT) [Cost: 375K, Rows: 20] (PATH ID: 2)
 |  Aggregates: sum(o.sales_dollar_amount)
 |  Group By: d.calendar_year_quarter
 | +---> JOIN HASH [Cost: 375K, Rows: 44K] (PATH ID: 3)
 | |      Join Cond: (o.sale_date_key = d.date_key)
 | |      Materialize at Output: o.sales_dollar_amount
 | | +-- Outer -> STORAGE ACCESS for o [Cost: 188K, Rows: 80M] (PATH ID: 4)
 | | |      Projection: online_sales.online_sales_fact_super
 | | |      Materialize: o.sale_date_key
 | | |      Runtime Filter: (SIP1(HashJoin): o.sale_date_key)
 | | +-- Inner -> STORAGE ACCESS for d [Cost: 52, Rows: 1] (PATH ID: 5)
 | | |      Projection: public.date_dimension_super
 | | |      Materialize: d.date_key, d.calendar_year_quarter
 | | |      Filter: (d.calendar_year = 2008)
```

Management Console

新しくサポートされるAWS EC2インスタンスタイプ

Management Consoleからデータベースを構成する際に選択できるAWS EC2インスタンスタイプに以下のタイプが追加となる。

- i4i.12xlarge
- i4i.24xlarge
- i4i.32xlarge


opentext™