Disclaimer

This document is for informational purposes only and is subject to change at any time without notice. The information in this document is proprietary to Actian and no part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of Actian.

This document is not intended to be binding upon Actian to any particular course of business, pricing, product strategy, and/or development. Actian assumes no responsibility for errors or omissions in this document. Actian shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. Actian does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.
Statistics Enhancements and Mixed Query Support

Doug Inkster
Actian Fellow
Abstract

Features enabling greater flexibility in the creation of histograms for more accurate query compilation have recently been added to Ingres and Vector. Work is also underway to permit the execution of mixed queries that access both Ingres and Vector tables.

This session discusses these and other new developments in the Ingres and Vector products.
Agenda

• Query optimization 101
• Create/drop statistics
• Autostats
• New IMA object for histogram information
• Mixed queries
Query Optimization 101

- Query optimizers break queries into individually optimizable units or blocks (e.g. each select of a union)
- For each block, optimizer considers all possible shapes and orders of join trees of constituent tables
- For each join tree shape, table order, optimizer estimates cost of executing block (CPU and disk access)
- Estimations are based on row cardinalities
  - Table cardinality
  - Selectivity of restriction, join predicates
  - Cardinality of groupings in aggregate operations
- Histograms document value distribution in a column
  - Used to estimate number of rows qualified by restriction, join predicates
  - Used to estimate number of rows resulting from grouped aggregation
- Historically, histograms built using optimizedb command line utility
Create/Drop Statistics

▪ CREATE STATISTICS FOR <table1> (<column1>, <column2>, ...), <table2> (<column1>, ...), ... WITH <options>
▪ DROP STATISTICS FOR <table1> (<column1>, <column2>, ...), <table2> (<column1>, ...), ...
▪ CREATE/DROP STATISTICS FOR ALL TABLES
▪ CREATE/DROP STATISTICS FOR ALL <user> TABLES
▪ <options> can be SAMPLE = nn, MAXCELLS = nnn, NOSAMPLE

▪ First appeared in Vector 4.0 for Vector tables only
▪ In 11.0 for Ingres tables
▪ Avoids some problems with optimizedb (mixed architectures between Ingres server machine and optimizedb machine)
▪ Histograms can be built as part of a single transaction
▪ Histograms can be built on global temporary tables (greatly improving query plans involving GTTs) – Histogram cached only – not written to catalog
Create/Drop Statistics

- For Vector:
  - X100 queries composed and executed to retrieve column values and counts
  - Histograms constructed in memory
  - Histograms then written to catalog tables or loaded to server cache (for GTTs)

- For Ingres:
  - DB procedure-like query constructed to compile/execute “DECLARE GLOBAL TEMPORARY TABLE abc AS SELECT ...
    ...” for each column in CREATE STATISTICS
  - SELECT statements retrieve column values and counts
  - Histograms constructed in memory by reading from GTT
  - Histograms then written to catalog tables or loaded to server cache (for GTTs)
Autostats - Vector

- First added to Vector 5.0
- Query optimizer (OPF) requests histograms for columns in WHERE/ON clauses and GROUP BY
- Previously, if column has no histogram, OPF built “default” histogram (2 cells for min, max values of column data type)
- With “autostats” enabled, “create statistics for <table> (<column>)” is executed under the covers to build the histogram in server cache
- Not written to catalogs, but stays in cache as long as memory isn’t needed (possibly for duration of server)
- Effectively obviates need to know anything about statistics
- Eliminates need to determine which columns need histograms (only built on those that require them)
- More practically, users can run optimizedb for columns whose distributions don’t change over time. Those that do change can be left for autostats to continually rebuild histograms
- Histogram built by autostats (in memory only) can be flushed with drop statistics. Autostats will rebuild histogram on next reference to column
**Autostats - Ingres**

- Will arrive in Ingres in point release (11.1??)
- Vector histograms are fast to build and can be built inline during optimizer processing
- Ingres histograms are slow to build and are constructed by a daemon thread
- Optimizer builds list of columns requiring histograms and passes to daemon thread for CREATE STATISTICS processing
- Query that triggers the histogram compiles with default and may produce substandard query plan
- Once histograms are built, queries will speed up with better plans
- Same rationale as for Vector, though selective use of optimizedb will mitigate startup performance issues

- Controlled by CBF opf_autostats option, “set autostats on/off” query (Vector, too)
- CBF opf_autostats_rebuild option identifies proportion change in table cardinality to trigger rebuild of autostats histogram (Vector, too)
IMA histogram object

- Vector 5.0, Ingres 11.1 (??) introduce new `ima_rdf_histogram_info` object registered in imadb
- Initially enabled server wide by “set trace point rd24”, disabled by “set notrace point rd24”
- Each histogram loaded from catalog or by autostats is recorded in IMA

Columns include:
- `rdf_owner, rdf_table, rdf_column` - identify column
- `rdf_autostats` - indicates whether built by autostats (Y/N)
- `rdf_count` - number of times requested by optimizer
- `rdf_buildtime` - time taken to build histogram (in msec for Vector autostats only)
- `rdf_numcells` - number of cells in histogram
- `rdf_cellsizes` - size of value in cell
- `rdf_nunique` - number of distinct values of column
- `rdf_reptfact` - repetition factor

Very useful in determining which columns to use optimizedb
select rdf_table, rdf_column, rdf_autostats, rdf_count, rdf_bldtime, rdf_numcells, rdf_cellsiz, rdf_nunique, rdf_reptfact from ima_rdf_histogram_info

<table>
<thead>
<tr>
<th>rdf_table</th>
<th>rdf_column</th>
<th>rdf_autostats</th>
<th>rdf_count</th>
<th>rdf_bldtime</th>
<th>rdf_numcells</th>
<th>rdf_cellsiz</th>
<th>rdf_nunique</th>
<th>rdf_reptfact</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineitem</td>
<td>l_linestatus</td>
<td>Y</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3000608</td>
</tr>
<tr>
<td>lineitem</td>
<td>l_shipdate</td>
<td>Y</td>
<td>1</td>
<td>10</td>
<td>1521</td>
<td>4</td>
<td>2526</td>
<td>2376</td>
</tr>
<tr>
<td>lineitem</td>
<td>l_returnflag</td>
<td>Y</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>2000405</td>
</tr>
</tbody>
</table>

(3 rows)
Mixed Queries

▪ From the start of the Vector project, ability to mix Ingres and Vector tables in same query was always in the plan
▪ Next release (??) will support mixed queries
▪ Ingres table references will be isolated into separate query blocks for compilation – if multiple Ingres tables are joined, they’ll be compiled in a single block
▪ Query optimizer will generate a single Vector query from remaining Vector tables and already-compiled Ingres blocks
▪ Vector will support new “IngresScan()” operator that calls back to Ingres server to materialize rows from the plan fragment of an Ingres block
▪ IngresScan() will vectorize Ingres rows into individual columns for Vector processing
▪ All other joins and aggregations will be performed in Vector (even aggregations on Ingres-only blocks)
New WHERE Clause Selectivity-by-sample

- Cardinalities in query fragments are computed by using histograms with WHERE/ON clause predicates to estimate predicate selectivity
- Many approximations are involved, correlations are ignored (problem with all query optimizers)
- We’re experimenting with alternate approach for Vector queries
- Restriction predicates on a single table are sent to Vector to be executed on a sample, returning selectivity on live data which is extrapolated to the whole table
- Sample is the minmax index for each column, augmented by an extra random value for each minmax entry
- Experiments in the literature show this to be more accurate than histograms
- More testing required!
Thank you!
Coffee Break - 15 minutes

#hybriddataconference