SQL Performance Overview

Version 2024.1
2024-05-16

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

SQL Performance Overview
InterSystems IRIS Data Platform Version 2024.1 2024-05-16

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

SQL Performance Overview

InterSystems SQL provides several capabilities that improve the performance of your SQL queries. These capabilities include:

- Automatically optimized query plans that efficiently process the queries based on the table data and the runtime parameters.
- Query analysis tools for tracking down slow-running queries or monitoring new queries running in production.
- Various table indexing methods for choosing the optimal index on frequently queried columns.
- Configuration options, such as parallel query processing, that efficiently process queries on large data sets.

This topic explores the range of features available for improving performance. To quickly get started improving SQL performance on your queries, see *Best Practices for Improving SQL Performance*.

1 Examine Query Performance

The InterSystems SQL Query Optimizer automatically processes the queries you run so that they run as efficiently as possible. These processes include:

- Cached queries — InterSystems IRIS maintains a cache of prepared queries. When you run a query, if that query is in the cache and the table and indexes being queried have not changed, then InterSystems SQL runs the cached query directly, skipping the more expensive steps, such as statement preparation and the generation of query plans.
- Query plans — The order in which the conditions of query are processed can dramatically effect performance. Often, the optimal order to process conditions depends on the table being queried. The Query Optimizer generates multiple query plans based on automatically gathered table statistics and runs the plan with the lowest performance cost. The optimizer also chooses among the most efficient plans at runtime based on the query parameters being supplied.

For more details on these and other processes, see *How InterSystems IRIS Processes SQL Statements*.

To help you make additional customization to improve SQL performance, InterSystems IRIS provides a record of SQL queries and other operations for each table, including insert, update, and delete. InterSystems IRIS also provides SQL runtime statistics than enable you to monitor how fast queries are running and identify slow-running queries. For more details on these features, see *Analyze SQL Statements and Query Execution Statistics*. If you need to profile specific queries in more detail, you can use the *SQL Performance Analysis Toolkit*.

Analyzing the plans generated by the query optimizer can also help you make sure the statements are running smoothly. See *Interpret an SQL Query Plan*.

For additional help with improving SQL performance, you can generate query reports that InterSystems Worldwide Response Center team members can analyze and suggest improvements. For more details, see *Get SQL Performance Help*.

2 Define High-Performance Tables

Tables with a schema optimized for queries can improve performance significantly.

2.1 Table Statistics

InterSystems SQL uses an operation called Tune Table to periodically examines the data in tables and gathers statistics about it.

- The number of rows in the table, also called the extent size.
- The relative distribution of distinct values in each column, for outlier detection and optimization of range queries.
- The average length of values in each column.

Tune Table sends this information to the Query Optimizer so it can choose an efficient plan based on queries that operation on the table. Tune Table runs automatically, but you configure how frequently it runs. For more details, see *Table Statistics for Query Optimizer*.

2.2 Indexes

One of the cornerstones of a high-performance table is defining indexes on columns. InterSystems IRIS automatically indexes certain columns, such as the RowID column that by default acts as the primary key for the table. You can index additional columns, such as those that are frequently queried.

Key decisions around indexes are which columns to index and what type of index to define given the data. InterSystems SQL provides a variety of different index types you can specify, including:

- Standard indexes — Defines an index column to speed up queries.
- Bitmap indexes — A special index type that is faster on data that has only a few distinct values.
- Bitslice indexes — A special kind of index that enables very fast evaluation of certain expressions, such as sums and range conditions.
- Columnar indexes — A special kind of index that stores frequently queried fields of table with an underlying row storage layout in a compressed, vectorized format.

For more details on working with indexes, see *Define and Build Indexes*.

2.3 Table Definition Settings

When defining a table through DDL, a number of best practice settings that optimize performance are applied automatically. When defining a table through a persistent class, consider enabling the following features in the class definition:

- USEEXTENTSET = 1 — This organizes table storage into a more efficient set of globals.
- Define a bitmap extent index — This creates a bitmap index on the entire table, which enables efficient counting and other operations.

For more details on these and other settings, see *Define SQL-Optimized Tables Through Persistent Classes*.

2.4 Storage Layout

Relational tables in InterSystems IRIS can store their data in rows, columns, or a mixture of both. Depending on the nature of your queries and transactions, the right storage layout can increase query performance or transaction throughput by an order of magnitude. For more information about columnar and row-wise storage, see *Choose an SQL Table Storage Layout*.

2.5 Sharded Tables

If the table has high data volumes, it can be efficient to shard the table, which distributes the data across multiple servers and merges the data back upon query. For more details on sharding, see *Horizontally Scaling for Data Volume with Sharding*.

3 Configure Query Performance

Additional configuration options for improving performance are available. For example:

- Parallel query processing — For high data volumes, allow multi-processor systems to divide query execution among the processors. You can configure parallel processing per query or system wide. For more details, see *Configure Parallel Query Processing*.
- Runtime Plan Choice — Some queries may see performance improvements due to runtime parameter values. When Runtime Plan Choice is activated, the SQL optimizer will choose whether or not to execute a plan that includes outlier status. For more details, see *Configure Runtime Plan Choice Query Optimization*.
- Frozen plans — When a SQL statement is prepared, a query plan that determines how the system executes the statement is created. This plan is purged when adding an index or recompiling the class and a different query plan will be created. However, you can retain an existing query plan across compiles by creating a frozen plan. For more details, see *Configure Frozen Plans*.
- Optimization hints — Provide additional information in the queries themselves that provide “hints” to the Query Optimizer to how to generate and select its query plans. For example, if all data from a column is being included in a query, then you can specify %NOINDEX on that column to inform the optimizer to not use that index in its query plan. In general, the Query Optimizer can make these kinds of decisions based on the table statistics, but these hints provide an additional level of customization. For more details, see *Specify Optimization Hints in Queries*.