Article
Benjamin De Boe · Jan 10, 2023 4m read

Open Exchange

Columnar Storage in 2022.3

As you may well remember from Global Summit 2022 or the 2022.2 launch webinar, we're releasing an exciting new capability for including in your analytics solutions on InterSystems IRIS. Columnar Storage introduces an alternative way of storing your SQL table data that offers an order-of-magnitude speedup for analytical queries. First released as an experimental feature in 2022.2, the latest 2022.3 Developer Preview includes a bunch of updates we thought were worth a quick post here.

A quick recap

If you're not familiar with Columnar Storage, please take a look at this brief intro video or the GS2022 session on the subject. In short, we're encoding table data in chunks of 64k values per column using a new $vector datatype. $vector is an internal-only (for now) datatype that leverages adaptive encoding schemes to allow efficient storage of both sparse and dense data. The encoding is also optimized for a bunch of dedicated $vector operations, such as for calculating aggregates, groupings and filters of entire chunks of 64k values at a time, leveraging SIMD instructions where possible.

At SQL query time, we take advantage of these operations by building a query plan that also operates on these chunks, which as you can image yields a massive reduction in the amount of IO and number of ObjectScript instructions to execute your query, compared to classic row-by-row processing. Of course, the individual IOs are bigger and the $vector operations are a little heavier than than the single-value counterparts from the row-oriented world, but the gains are massive. We use the term *vectorized* query plans for execution strategies that deal with $vector data, pushing entire chunks through a chain of fast individual operations.

Just faster

Most importantly, things got faster. We've expanded the optimizer's understanding of columnar indices and now you'll see more queries use columnar indices, even if some of the requested fields are not stored in a columnar index or data map. Also, you'll see it combine columnar and bitmap indices in a number of cases, which is great if you're starting off by adding columnar indices to an existing schema.

The new kit also includes a bunch of changes across the stack that improve performance, ranging from optimizations to those low-level $vector operations over some query processing enhancements and a broader set of vectorized query plans that can be parallelized. Certain ways of loading data, such as through INSERT .. SELECT statements, will now also employ a buffered model we already used for building indices and now enable really high performance building of entire tables.

Vectorized JOINs

The most exciting capability we added in this release is support for JOINing columnar data in a vectorized fashion. In 2022.2, when you wanted to combine data from two tables in a query, we'd still resort to a robust row-by-row JOIN strategy that works on columnar and row-organized data alike. Now, when both ends of the JOIN are stored in columnar format, we use a new kernel API to JOIN them in-memory, retaining their $vector format. This is another important step towards fully vectorized query plans for even the most complex queries.

Here's an example of a query that takes advantage of the new function, doing a self-JOIN of the New York Taxi dataset we used in earlier demonstrations:

```
SELECT
   COUNT(*),
   MAX(r1.total_amount - r2.total_amount)
FROM
   NYTaxi.Rides r1,
   NYTaxi.Rides r2
WHERE
   r1.DOLocationID = r2.PULocationID
   AND r1.tpep_dropoff_datetime = r2.tpep_pickup_datetime
   AND r2.DOLocationID = r1.PULocationID
   AND r1.passenger_count > 2
   AND r2.passenger_count > 2
```

This query looks for pairs of trips with more than 2 passengers, where the second trip started where the first one ended, at the exact same time and where the second trip took one back to where the first one started. Not a super-useful analysis, but I only had one real table in this schema and the composite JOIN key made this a little less trivial. In the query plan for this statement, you'll see snippets like Apply vector operation %VHASH (for building the composite JOIN key) and Read vector-join temp-file A, which indicate our new vectorized joiner at work! This may sound like a small and trivial nugget in a long-ish query plan, but it involves a lot of smart engineering on the internals, and there's quite a few high-profile columnar database vendors out there that simply don't allow any of this and put severe constraints on your schema layout, so please JOIN us in enJOINing this! :-)

When the query plan goes on to read that tempfile, you may notice there's still some row-by-row processing in the post-join work, which brings us to...

What's next?

Columnar Storage is still marked as "experimental" in 2022.3, but we're getting closer to production readiness and having that full end-to-end vectorization for multi-table queries. That includes the post-join work mentioned above, broader support in the query optimizer, even faster loading of columnar tables and further enhancements to the joiner such as shared memory support. In short: now's a great time to give this all a first try with the New York Taxi dataset (now on IPM or with docker scripts) using the 2022.3 Community Edition, so you only have to press "Run" by the time we release 2023.1!

If you are interested in more tailored advice on how to leverage columnar storage with your own data and queries, please reach out to me or your account team directly, and maybe we'll meet at Global Summit 2023 ;-).

#Columnar Storage #SQL #Vector Search #InterSystems IRIS
Check the related application on InterSystems Open Exchange