MSCI ESG Metrics Database Structure Investigation

Mark Bergh

2023-02-08

The Current Structure

The current table that holds metric data queried by the MSCI ESG Data Service (esg_metrics) has the following structure:

issuerid	metrics	start_date	end_date	checksum
IID000000001789005	{JSON Blob}	2013-01-01	2013-04-08	{Some Characters}

When accessing metric data, we use MS SQL Server inbuilt JSON-parsing functionality to retrieve the relevant attributes.

The Problem

The issue is that the queries are taking way too long for a lot of common use-cases in the investment teams. This is due to the JSON-parsing step of data access taking a long time, as well as the fact that there is a lot of data redundancy in the current setup.

The Proposal

I've been investigating new table structures which might have faster read times for the types of queries that are failing at the moment. Specifically I've been looking into a "long and thin" table structure. The one I've had some success with has this structure:

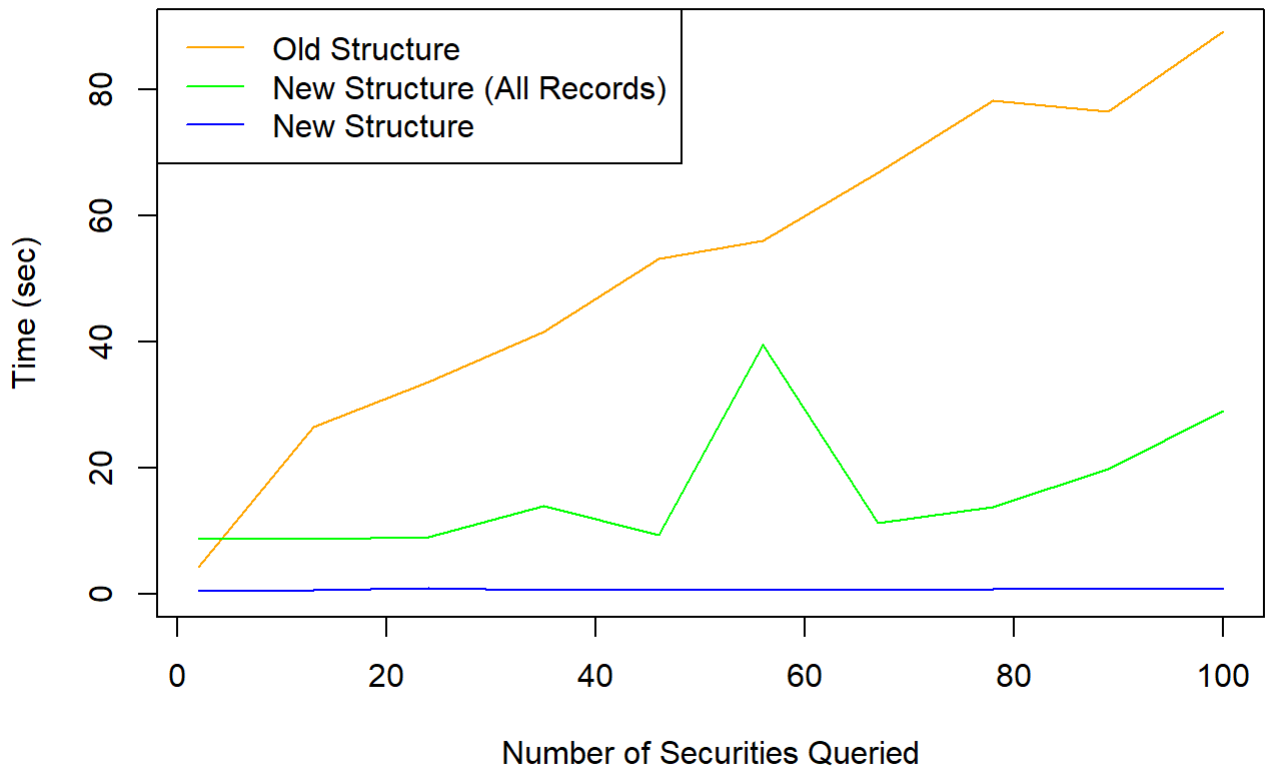
id	metric	issuerid	value	date
100000	INDUSTRY_ADJUSTED_SCORE	IID000000002638393	2.8999999999999999	2013-01-01

This structure works well. It can handle queries of the sort the Investment teams want to make in around a second in the testing environment (in production it could take longer). One change to this structure that I would recommend if we go ahead with implementing it in production would be to split the "date" column into "start_date" and "end_date". I didn't originally see the point in this, but I think it enables "as at date" queries to be made more easily.

Comparison on a Representative Subset

In order to get measurements for the time it takes to make queries to the different table structures I created two tables on the development DB: LDNPDITSQLDC1. I inserted all records associated with approximately 2% of the total IssuerIDs we have in esg_metrics into two testing tables, one with the old structure and one with the new proposed structure.

I looked at the time it took to complete a database query for the different structures (not including the time to construct the SQL query using SQLAlchemy) and plotted it against the number of securities queried:



This if for a time-series query over a 10-year time-frame.

For a range of query types, similar results were found:

	Old Structure Mean	Old Structure SD	New Structure Mean	New Structure SD
10 Sec (10 years)	2.84949	1.203014	0.4537926	0.0088687
50 Sec (10 years)	54.06430	7.083895	0.5800642	0.0369708
50 Sec (1 year)	11.57781	1.010523	0.4618854	0.0118231

I believe the substantial speed increase observed in the testing environment is due mainly to the overhead involved in the SQL Server JSON-parsing functionality. Additionally, the proposed structure has other benefits. Of particular interest will be that it is much easier to only record changes in metric values using this structure. I believe at the moment we will insert a record of all metric associated with a security to the DB if any of the metrics have changed in the intervening day. However, the majority of metrics change value infrequently. This results in data data redundancy. Since records in the new structure are stored on a per-metric basis, it is much easier to only store the relevant data, which should yield faster and more information-dense queries.