Hello David,

Below are the questions I have before starting the project. Please answer the below questions to ensure we are on the same track.

1. Will the provider need to support joining multiple JSON data sources? If so, what types of joins are required (e.g. inner, outer, left, right)?

I would like the provider to support inner joins. Other types of joins may be added in future work, but other types of JOINs are outside of the scope of this work.

1. Will the provider need to support grouping JSON data by one or more fields?

No. This is outside of the scope of what I’m asking here.

1. Will the provider need to support aggregate functions (e.g. COUNT, SUM, AVG, MIN, MAX) on JSON data?

Yes.

1. If aggregate functions are needed, what specific aggregate functions are required?

Support for COUNT(\*) is the only aggregate that is necessary.

1. Are there any specific performance requirements for the provider when performing joins, grouping, or aggregate functions?

Only functional verification is required. No performance requirements at all. Of course, if we see obvious inefficiencies in the code, those will be pointed out in the PR reviews.

1. Will the provider need to support sub-queries for joins, grouping, or aggregate functions?

No. Out of scope.

1. Will the provider need to support full-text search on JSON data?

No. Out of scope.

1. Will the provider need to support indexing of JSON data? If so, what types of indexes are required?

No. Out of scope. (Later, if use cases present themselves, we may consider adding some indexing support)

1. Will the provider need to support sharding of JSON data?

No. Out of scope.

1. Will the provider need to support geospatial queries on JSON data?

No. Out of scope.

1. Will the provider need to support real-time updates to JSON data?

Operations UPDATE/DELETE performed asynchronously within a loaded instance of the provider needs to (at the “database” level) take an exclusive writer lock for the database. Concurrent reader access to data, though, is required. Using something like [ReaderWriterLockSlim](https://learn.microsoft.com/en-us/dotnet/api/system.threading.readerwriterlockslim?view=net-7.0) class is ideal.

Note: If there happen to be multiple loaded instances of the provider in different processes, you can take a “write last, wins” approach. You must only ensure that for a single JSON file, that multiple writers aren’t writing at the same time (i.e. avoiding file corruption of the JSON). Therefore, in the scenario mentioned in the README where a folder represents a ‘database’ in which there is a file for each ‘table’, it is a valid scenario that a provider in one process writes to table A (for instance) and later a second provider in another process writes to the same table that the first provider still contains his cached version of the data in table A. (Of course, it would be ideal if the first provider was monitoring the file with [FileSystemWatcher](https://learn.microsoft.com/en-us/dotnet/api/system.io.filesystemwatcher?redirectedfrom=MSDN&view=net-7.0) class and refreshes its table cache when the file changes.)

Also, in that scenario, it is valid if one provider is writing to table A while the second provider is writing at the same time to table B. (Of course, this isn’t allowed in the scenario where the whole database is stored in one JSON file)

1. Will the provider need to support streaming JSON data?

No. Out of scope.

1. Will the provider need to support auditing and logging of JSON data?

No. Out of scope.

**Although**, this reminded me that I failed to mention that it is vital for the provider to have properly implemented [DbConnection.GetSchema()](https://learn.microsoft.com/en-us/dotnet/api/system.data.common.dbconnection.getschema?view=net-7.0). Eventually this provider will be used in developing an [EF Core](https://learn.microsoft.com/en-us/ef/core/) Provider that will allow for [scaffolding](https://learn.microsoft.com/en-us/ef/core/managing-schemas/scaffolding/?tabs=dotnet-core-cli).

1. Will the provider need to support backup and recovery of JSON data?

No. Out of scope.

Below is an example of an SQL query

SELECT Customers.Name, ~~SUM(Orders.Total)~~ AS TotalSales

FROM Customers

JOIN Orders ON Customers.Id = Orders.CustomerId

~~GROUP BY Customers.Name~~

~~HAVING SUM(Orders.Total) > 1000~~

NOTE: I have crossed out what isn’t needed in this SQL query. Also, the query below is needed (as mentioned above):

SELECT COUNT(\*)

FROM Customers

JOIN Orders ON Customers.Id = Orders.CustomerId

It isn’t required to support the COUNT(<column name>) syntax though, just the asterisk feature.