Top 12 Data Integration Challenges

Integration of data into a data lake or data warehouse is one of the core functions of any modern data analytics or reporting solution. In this article, we look at the top 12 considerations when implementing data integration for an Azure based data solution.

1. Being able to implement new data integrations rapidly

A common challenge for data teams is to be able to implement new data into their platform rapidly. Where a source system has hundreds of tables, it can be very time-consuming to implement integration pipelines for each. The use of a meta-data driven integration solution such as our Azure Data Factory and Lakehouse accelerators can rapidly improve the time to value for data solutions. Data can be ingested from source systems and loaded into the data lake and data warehouse staging area automatically. The step to implement data lake gold tables or data warehouse tables is the only step that requires specific coding, and this can typically be applied from template solutions.

2. Detecting incremental changes in the source system data

When implementing data integration solutions, it is generally preferable to load data incrementally into the data platform – so you only load data that has changed since the previous integration run. This is often challenging to implement reliably as it requires a method for detection of changed records in the source systems. Implementation of a Change Data Capture solution is the most reliable approach for this, however it is not always practical or affordable to implement. The most common approach to incremental data extracts is to use a watermark column – such as LastUpdateDate – on the source tables to detect changes. However, this is sometimes not reliable – for example if the support team needs to manually update data – and does not pick up records that have been deleted from the database. Other approaches can also have reliability challenges – for example the use of audit tables – or suffer from performance issues – such as using database triggers or checksums to determine what has changed. In some cases, a hybrid approach is required where a watermark extract is used to pick up most changes then once a week/month a full extract is undertaken to ensure that all changes have been picked up.

It is important that a thorough analysis of the source system and the data required in the data platform is undertaken to make sure that the most appropriate integration approach is being used.

3. Real time data integration using change data capture

It is essential that real-time data integration solutions are robust and reliable in integrating data into the data platform. The solution will need to handle unexpected events – for example networking issues – cleanly and need to ensure that no data is ever lost or duplicated in the data platform. We believe that the best approach for this is to use a data integration tool that has been designed for this specific task and has been proven in use over many years. These tools can be costly but can be a wise investment.

4. Managing dependencies

The data warehouse data model that is implemented for reporting will typically utilise a dimensional data model that is optimised for reporting requirements. Tables in this design may require data from several source system datasets. Management of dependencies when implementing these tables can become complex. It is important that requirements such as these are thoroughly analysed, designed, and implemented in the most optimal and reliable way possible.

5. Managing business logic

Where integration solutions span multiple technologies, business logic may be implemented in several technologies – integration technologies, data warehouse, reporting tools. It is important that clear rules are defined and followed regarding how business logic should be implemented to avoid future confusion for support teams. It is generally preferable to implement business logic in a single component of the architecture where possible.

6. Managing performance

All transformation logic that is implemented in the integration solution can have an impact on the performance of and how rapidly data can be ingested into the platform for reporting. Particularly where data is required for near real time reporting and there are a large number of changes applied regularly – for example telemetry data sources. It is important that the minimal number of transformations are implemented in these cases. The data warehouse design should take this into account – potentially having an untransformed table for near real time reporting and a transformed table for more analytical reporting.

7. Managing cost

When data is being imported into the data platform on a very frequent basis, it is important to keep an eye on costs of the solution. Running complex Data Factory jobs every few minutes can have costs which mount up over time. This is particularly true when using Data Factory Data Flows to perform transformation. We normally recommend against the use of Data Flows for transformations - Databricks can be used to implement them more efficiently and cost effectively.

8. Reconciliation of data with source systems

It is essential that the data platform being implemented is viewed as being a reliable data store and that users of reports have full trust in the information that they are being presented with. Data in the platform therefore needs to fully match that in the source system. This can be challenging with non-CDC integrations as noted previously. It can often be useful to develop reconciliation processes that perform counts/sums of data in the source – for example, count of customers or sum of account balance - and compares these with the counts/sums of data in the data platform.

9. Handling schema drift

Schema drift occurs when the schema of source system tables/files changes. In an ideal world, application teams would notify the data team in advance of this happening so that data ingestion pipelines and data models can be adjusted accordingly. However, in practice this often does not happen and the first time the data team are aware of the change is when a pipeline fails. The integration processes should be designed to handle schema drift as cleanly as possible. This can typically occur once the data hits the bronze zone of the data lake which does not have schema applied. The integration processes can check that the received data complies with the expected schema and if it doesn't then it can be moved to a malformed zone, and the support team alerted to the issue. Once the pipelines and data model have been updated, the data that was in the malformed zone can be re-processed and ingested into the platform. In some cases, schema drift can be handled automatically, however this should be used with extreme care as if corrupted data is somehow received from a source, it can impact the data lake and data warehouse design and reporting figures.

10. Getting Data from Software as a Service applications

Software as a Service (SaaS) applications have become very popular over the past few years as they allow organisations to rapidly implement business solutions that are fully managed – so there is no need to establish and maintain infrastructure or to manage patching and enhancement of the applications. However, SaaS applications generally do not allow access to their internal database to allow organisations to extract their data using data integration tools. Most applications enable data access through APIs and some larger ones provide operational data store databases that can be extracted from. Each of these may have challenges when it comes to integrating data into a data platform – for example, it may not be possible to get all data, and it may not be possible to detect changes to records. Integrating data from SaaS applications may require more analysis up front to understand what is required and may also require more development time – e.g. extraction of data from APIs is not as easy to automate as getting data from database tables. It is often necessary to work closely with SaaS vendors to understand the best ways of integrating their data into a data platform.

11. Being able to replay data loads

There can often be occasions where data in a data platform table needs to be reloaded. This can occur due to an error or a design change to the data warehouse (or Lakehouse Gold) tables. This can be simplified where data is stored in a well-structured data lake that records the raw data that was received (e.g. in Bronze tables) as well as data that is structured and easily accessible (e.g. in Silver tables). Integration solutions should initially load these tables and then load the data warehouse or gold tables from these. If it is necessary to reload a table in the data warehouse, then it can be done from the data lake rather than going back to source system. Even with a well-designed system though there is still likely to be some manual work required – such as ensuring that data is reloaded in the correct order.

12. Ensuring resilience and reliability of integration processes

It's essential that data integration processes are reliable and resilient. Pipelines should be designed to handle most expected data conditions and if they do fail due to an unexpected condition then they should alert the support teams and should be able to be restarted quickly and cleanly. The restart processes should guarantee that no data is lost, and no data is duplicated.

Versor are data integration experts. We have years of experience of implementing successful data integration solutions for our customers. We have also developed Azure Data Factory and Databricks Lakehouse accelerators that speed up the implementation process and ensure all data is integrated using a reliable, robust and extensible framework. For more information, please contact a Fujitsu Data & AI specialist now.

Contact
Fujitsu Data & AI
+61 3 9924 3000