Central Trade Repository

Solution Logical Architecture

Prepared By:

**Dheeraj Marwaha**

27-Feb-2020

Contents

[1 Introduction 3](#_Toc31283007)

[1.1 Assumptions 3](#_Toc31283008)

[2 Solution Design – Logical Architecture 4](#_Toc31283009)

[2.1 Market Segment 4](#_Toc31283010)

[2.2 Requirements for Central Trade Repository 4](#_Toc31283011)

[2.2.1 Functional Requirements 4](#_Toc31283012)

[2.2.2 Non-functional Requirements 4](#_Toc31283013)

[2.2.3 Extended requirements. 5](#_Toc31283014)

[2.3 Constraints 5](#_Toc31283015)

[2.3.1 Usage Data constraints 5](#_Toc31283016)

[2.3.2 Technology Boundaries 5](#_Toc31283017)

[2.3.3 Environment Constraints 5](#_Toc31283018)

[2.4 Logical Architecture Details 6](#_Toc31283019)

[2.4.1 Front-office 7](#_Toc31283020)

[2.4.2 Central Trade Repository 8](#_Toc31283021)

[2.5 Open questions 11](#_Toc31283022)

1. Introduction

Central Trade Repository is a data consolidation platform to fetch the data from front-office data centers located across multiple geographical locations

Consolidation process includes operations – Insert, Update, Delete, Transform the records along with water marking for incremental and full pulls.

Finally, it has the capabilities to expose the data to downstream applications.

* 1. Assumptions

1. Instead of cloud-based data consolidation system, service company wishes to have in-house solution.
2. Application instance deployed on different geo-locations has same code base and structure of database schema.
3. All data centers are independent to each other and don’t have any interaction.
4. All the data centers can connect to the central trade repository system over the internet.
5. The data source used in front-office applications is SQL Server.
6. All the front office applications are using the same instance of data source.
7. Back-office applications can process the data in batch mode as well as single record processing.
8. Team has adequate exposure to modern technologies.
9. Data sanity check is possible after cut off time in the end of the day trading session.
10. Data from front office applications are “single source of truth”.
11. The solution is designed to handle the daily generated data (delta) in facility.
12. The solution has mechanism to archive the processed data.
13. All the applications are following the common coding standards.
14. Solution Logical Architecture
    1. Market Segment

To beat the network latency and to achieve higher availability, Service company has established front offices across the multiple geographical regions/facility/tenant.

Each region has set of applications which can run independently to book the trades.

* 1. Requirements for Central Trade Repository

The purpose of trade repository is to consolidate the trading data happened across the branches.

* + 1. Functional Requirements
* Central repository system should allow the data sync tool at a facility to migrate data from front office facility to central repository.
* It should store and archive the data from each data center.
* Downstream applications should be able to fetch the data in batch mode as well as through REST API’s.
  + 1. Non-functional Requirements
* Reliability, Availability

The system should be designed for 99.9% availability. The system should have the ability to continue operating with graceful degradation when unexpected faults occurs in some parts of the system.

* Security
* Quick Troubleshooting and support

The system should common logging platform to easily troubleshoot and solve the issues very fast.

* Performance
* Logging

System should provide centralized mechanism to host the logs from windows, SQL server, events, applications and services.

* Monitoring

To support fast failures, system should be equipped to provide the analytics and dashboards for the health probe of the different components in system.

* Automated Testing
  + 1. Extended requirements.
* Data sanity check should be triggered by end of the day after trading session cut off time.
* System should provide mechanism to re-run the data migration to pull the data again in case of data sanity failure.
  1. Constraints
     1. Usage Data constraints

The constraints in this section derive the capacity of the planned system. These are stated in Table 1.

Table 1 - Usage data constraints

| **Details** | **Values** |
| --- | --- |
| Peak day | Any week day |
| Peak Hour | 9:30AM-3:30 PM |
| Least Activity Hour | Remaining hours of peak hours |
| Average Number of Hits per day on Weekdays | Application to be benchmarked for 6000 concurrent users. These parameters can be determined at performance testing stage |
| Total Number of Hits on a Peak day |
| Total Number of Hits for Peak hour on a peak day |
| Average number of Hits/secs on a peak day |
| Average number of Hits/secs for peak hour on a peak day |
| Maximum number of Hits/Sec on a peak day |
| Number of transactions on a peak day | 5,000,000 |
| Total Size of the content per day | 5 GB (OLTP Data)/per day (1kb record and log, archive and other data extra). |

* + 1. Technology Boundaries

On-premise deployment has below constraints.

* Auto-scaling on-premise needs to be handled manually or with help of some custom solutions.
* DR and HA needs to be planned for all components.
* Database size needs to be monitored constantly.
* For backup strategy, it is important to decide the limit on the backup governance size say 1 TB or 500GB.
  + 1. Environment Constraints
* Central trade system runs on windows server on premise.
* Sync must happen between central and connected facility.
  1. Logical Architecture Details

This section describes logical architecture of the solution as represented in Fig.1.



**Fig.1 Logical Architecture Details**

The solution is logically separated into Front-office, Central trade repository, cross cutting services and Back office application layer. The data will migrate periodically from the data centre hosting front-office to central repository using push mechanism. Central repository provides a comprehensive view of the trade data which took place in all the data centres across multiple geographical locations. It provides the data in batches to back-office for further processing. The polyglot persistence will be used across data layers for high availability.

* + 1. Front-office

This layer consists of a trade booking system having front-office applications deployed in a specific data center. The trade booking system is spread across multiple data centers as Tenants to beat the network latency and make the application high available.

Each app

* Works as independent application.
* Shall be deployed independently, scaled out independently.
* Has own data/storage schema. (It is assumed that application is using SQL server as data source.)
* Has own business logic and UI.
* Uses data aggregator to push data/retrieve to/from central repository periodically.
* Can fetch tenant specific information like country, currency, time zone locally and embed in transaction record.

Each trade transaction consists of the fields shown in Table 2.

Table 2- Trade Booking Transactions

| Parameter | Used for |
| --- | --- |
| * [Country](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#app-service-limits) | * Customer base country |
| * [Assets Type](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#azure-redis-cache-limits) | * Customer is interested in which product like Forex |
| * [Trade Data](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#backup-limits) | * Transaction date and time in UTC |
| * [Currency](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#cloud-services-limits) | * Currency in which trade is executed. |
| * Trade Value | * Amount |
| * Tenant | * Data center details |

* + - 1. Change Tracking in Front-office application

The challenge is to find out the modified records in data store used by front office. The data store -SQL Server (As the assumption) provides two ways to track the changes. These are:

1. Change data capture – In case of enterprise license, user needs to enable the CDC for designated tables to capture the DML activities. CDC works by scanning the transaction log for a designated table’s ‘captured columns’ whose content has changed without disturbing the OLTP tables.
2. In case of absence of CDC as in standard edition of SQL server, SSIS should be used for ETL process.
   * + 1. Data sync – Data Aggregator

Data aggregator is a layer which picks up the incremental transactional data in a specific time window from the front-office application data source and pushes it to central repository in required denormalized format.

Data aggregator has the mechanism of water marking (read position or offset) and can run in two mode for data migration

1. Full mode – initial or full backup
2. Incremental mode – fetch only the modified records as delta

Major challenges for Data sync

* To build real-time streaming data pipelines.
* Reduce multiple copies of transactional data in system.
* Execution of SSIS packages to avoid deadlock in tables.

Modern Frameworks

In case of increase load on the central repository, below optimizations can also be possible.

* Push data into Kafka at the same time you put in the database in front office application. Using Spark streaming to move the transactional data into central repository in almost real time.
  + 1. Central Trade Repository

Due to multiple tenants/ facility/ data centers, the trade book details are scattered in different geographical locations. To support the Back office, there should be some system which can provide the comprehensive view of the trade data – this system is called **Central Trade repository**.

The primary responsibility for Central Trade Repository is to provide the data persistence.

It consists of windows failover clusters and SQL server with AlwaysOn Availability groups for high availability and core services to expose the trade data to downstream over a secure channel.

* + - 1. Data Ingestion

To reduce the coupling between data center and trade repository, data ingestion mechanism is implemented as an agent/crawler in data centers along with front office applications.

For the current design, this data sync happens through SSIS ETL packages which will read the changed data from the trade warehouse and push them in to the staging area of central repository periodically.

* + - 1. Data Transformation

Central Trade repository provides an optional data cleaning mechanism during moving the data from staging database to warehouse database.

* + - 1. Data Storage

The consolidated data in central trade repository will be saved in SQL Server databases.

The infrastructure consists of windows server failover cluster with configured primary and secondary replicas nodes for SQL server.

The data sync between primary and secondary replica happens using AlwaysOn feature of SQL Server using sync/async mode.

* + - 1. High Availability and Reliability

Central Trade repository is using SQL server Enterprise Edition with AlwaysOn which consists of two technologies

* AlwaysOn Failover Clustering Instances
* AlwaysOn Availability Groups.

This feature provides in-built support to failover to healthy machine manually/automatically and syncing of data from primary to secondary node. There is voting mechanism to promote secondary node as primary replica in case primary node is down.

It is recommended to use primary replica for write operations and secondary replica’s for read operations. Which can be controlled at connection level.

* + - 1. Disaster Recover

One of the secondary nodes in WSFC can be used as Data recovery node

* + - 1. Core Services

Core services are responsible to expose the central trade repository data to external world.

Batch processing

In case of batch processing of data, back-office application can use SSIS packages to fetch the data from central repository and process it as per business rules.

For this, their service account needs to be whitelisted in CTS.

Web API

For web applications like monitoring, dashboards, log insights, the data is exposed through Web APIs. These APIs should be hosted in containerized environment along with orchestration.

Service locator – API Gateway

All the services in central trade repository are onboarded into API Gateway which is the single-entry point for all the web clients.

* + - 1. Logging

Logging is fundamental to troubleshoot any application problem. There are two driving technologies in the Central trade repository.

SSIS

The SQL Server Integration Service includes the Logging service (Logs). It can be used in SSIS packages, data flow tasks. It provides different log provides to write to text file, SQL server, Events.

Web API – Services

The logs generated by programming language specific libraries/applications should write to REST APIs provided by common logging framework.

Common Logging Framework

To troubleshoot, identify debug and fix performance issues, health and to support fail fast, common logging framework is required.

Log management is a tricky thing in modular applications which consists of different components in different technologies.

Possible ways to emit application logs

* Output logs to the console

Application should write to console using native log provider. There should be some hook to read the logs from console and stream to a log management service.

* Log to a file

It will be useful if the file is available on some shared location like blob.

* Steaming logs

REST APIs or listeners are exposed by log management service which take care of optimization and analytics top of logs.

Example of some log types

* Programming language specific libraries/applications across servers
  + Web request details
  + Full stack trace
  + Related log message
* Web server logs
* Syslogs
* Windows events

CLF should support log monitoring, alerts, structured logging and mechanism to correlate the log messages.

CLF can be implemented in-house or based on 3rd party solutions like

* AppInsight
* Stackify Prefix

Specialized C# logging frameworks

* Native logging providers - TrackeSource
* Apache log4net
* Serilog

In case of logging to file, there is one challenge - how to move file logs to central platform. To solve this, we need to come up with monitoring agents to tail each log file and stream them to central platform. You can configure a Serilog sink to stream logs via TCP directly to your account.

Table 3- Minimum log fields

| Parameter |
| --- |
| * [Date](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#app-service-limits) |
| * [Logger Name](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#azure-redis-cache-limits) |
| * [Thread](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#backup-limits) |
| * [Log](https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/#cloud-services-limits) Level |
| * Event Id |
| * User Id |
| * Message |
| * Method |
| * Line |

* + - 1. Other major components of the solution

The following are the other main components of the central trade system.

* Authorization
* Globalization/Localization
* Audit/ Activity logging
* Reporting tools
* Notification Framework
* Watch dog – Health probe
* Data dog – steam logs to central platform.
  + - 1. On-boarding new Data center

Each data center can function independently for booking trade. Next thing is to move trade data from facility data center to central trade repository. This is achieved by data sync tool which comprise of SSIS packages deployed in facility to push modified data to central repository.

There should be one whitelisted or authorized GMSA account which can be used to access the facility databases as well as staging database in central repository.

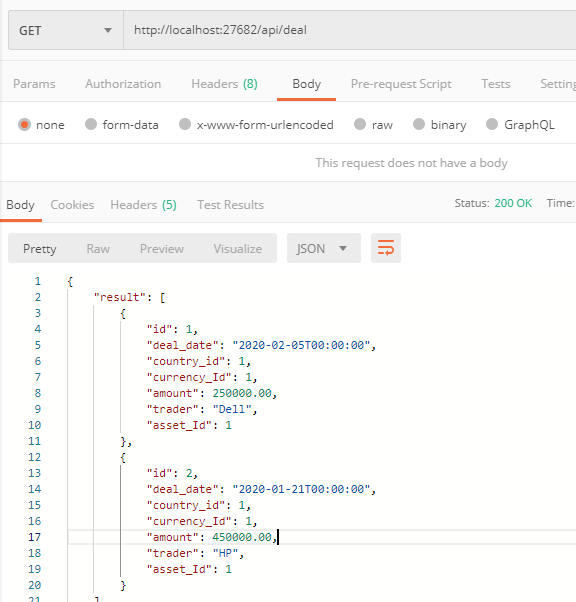
1. Central Repository Service API

Below table represents public API

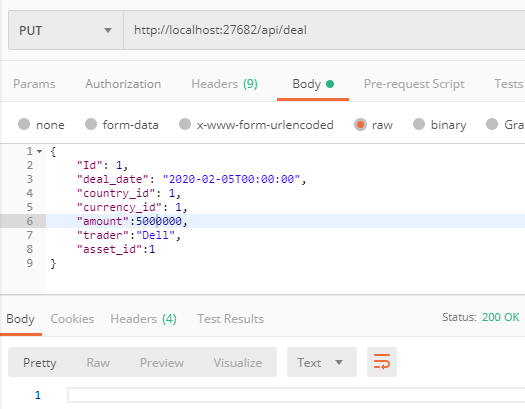
|  |  |  |
| --- | --- | --- |
| Resource | Method | Description |
| /deals | GET | Gets a list of all deals |
| /deals/{id} | GET | Gets details for a single deal |
| /deals/{country\_id} | GET | Gets a list of all deals happen for a country |
| /deals/date | GET | Gets the list of all deals happened after the bookmark date |
| /deals/{amount} | GET | Get a list of all deals more than given amount |
| /deals/{id} | PUT | Update deal properties |
| /deals/{id} | DELETE | Delete an entire deal |
|  |  |  |
|  |  |  |

1. API Testing

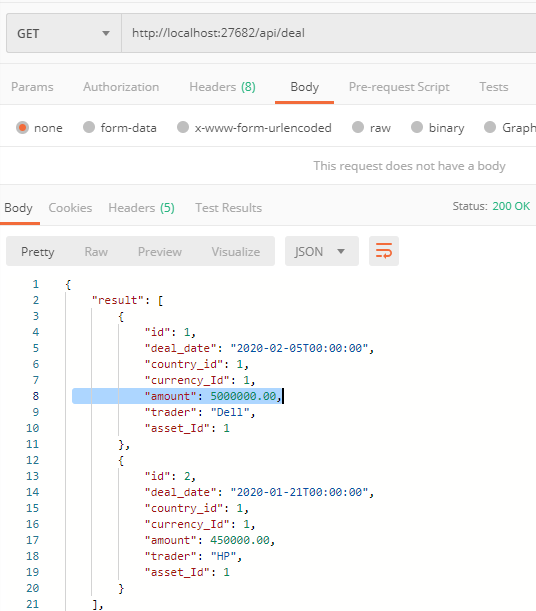
Check the current deals



Update the fields of a deal



Recheck the status



1. Open questions
2. Do we need a mechanism/Sync where changes made by back office need to be reflected in front office?