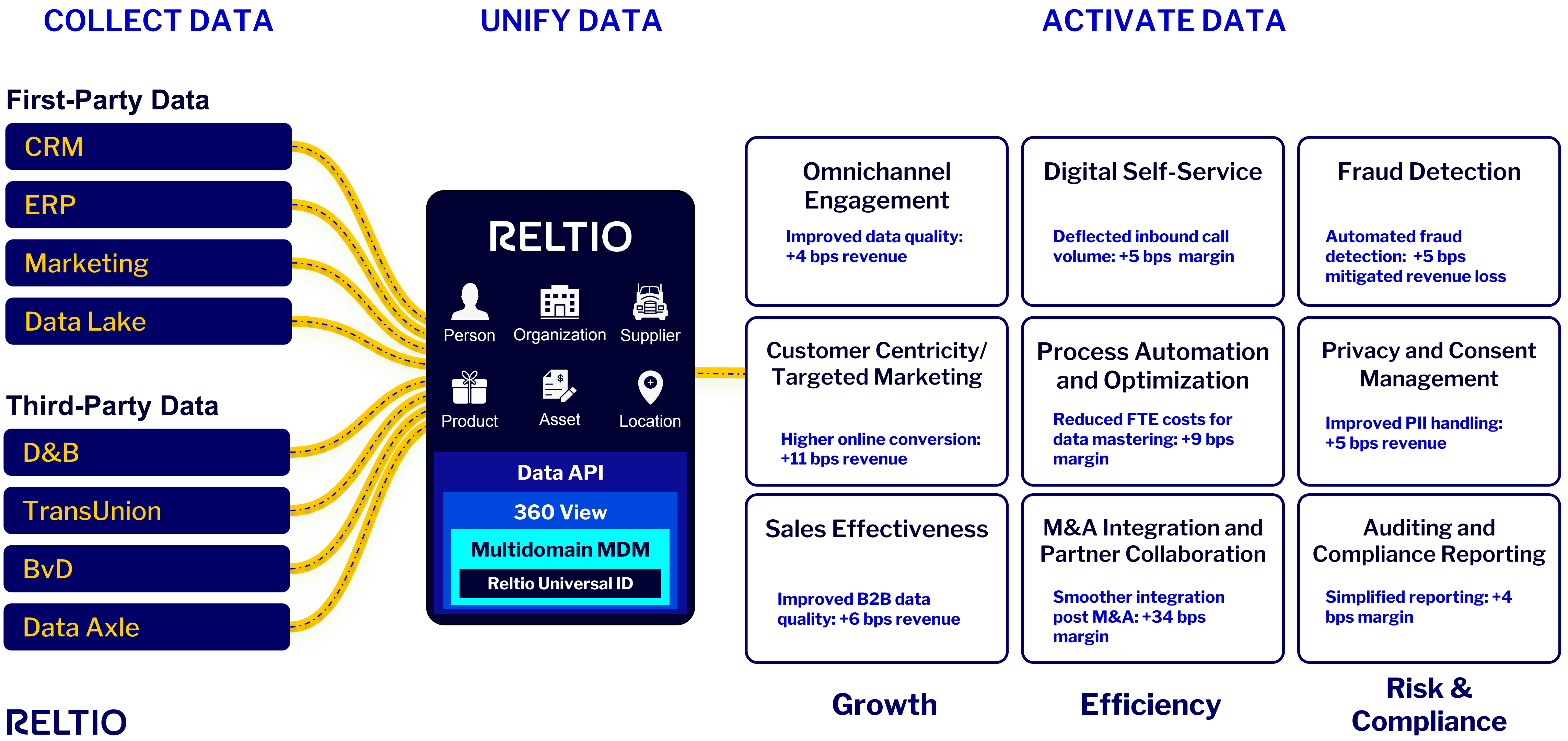
RELTIO

Operational MDM

About the Presenters

Mike Frasca serves as Field CTO at Reltio, responsible for overseeing strategic technical guidance to all customer facing Reltio teams and assisting with long term product strategy and innovation at Reltio. Prior to joining Reltio in 2016, Mike led the technical side of MDM implementation teams at multiple consulting firms and was a principal architect at IBM and Initiate Systems. He has worked on some of the most complex MDM implementations and led data architecture designs used by Fortune 100 companies today.

Key business initiatives we activate across industries



COLLECT DATA

First-Party Data

- CRM
- ERP
- Marketing
- Data Lake

Third-Party Data

- D&B
- TransUnion
- BvD
- Data Axle

UNIFY DATA

RELTIO

Data API

360 View

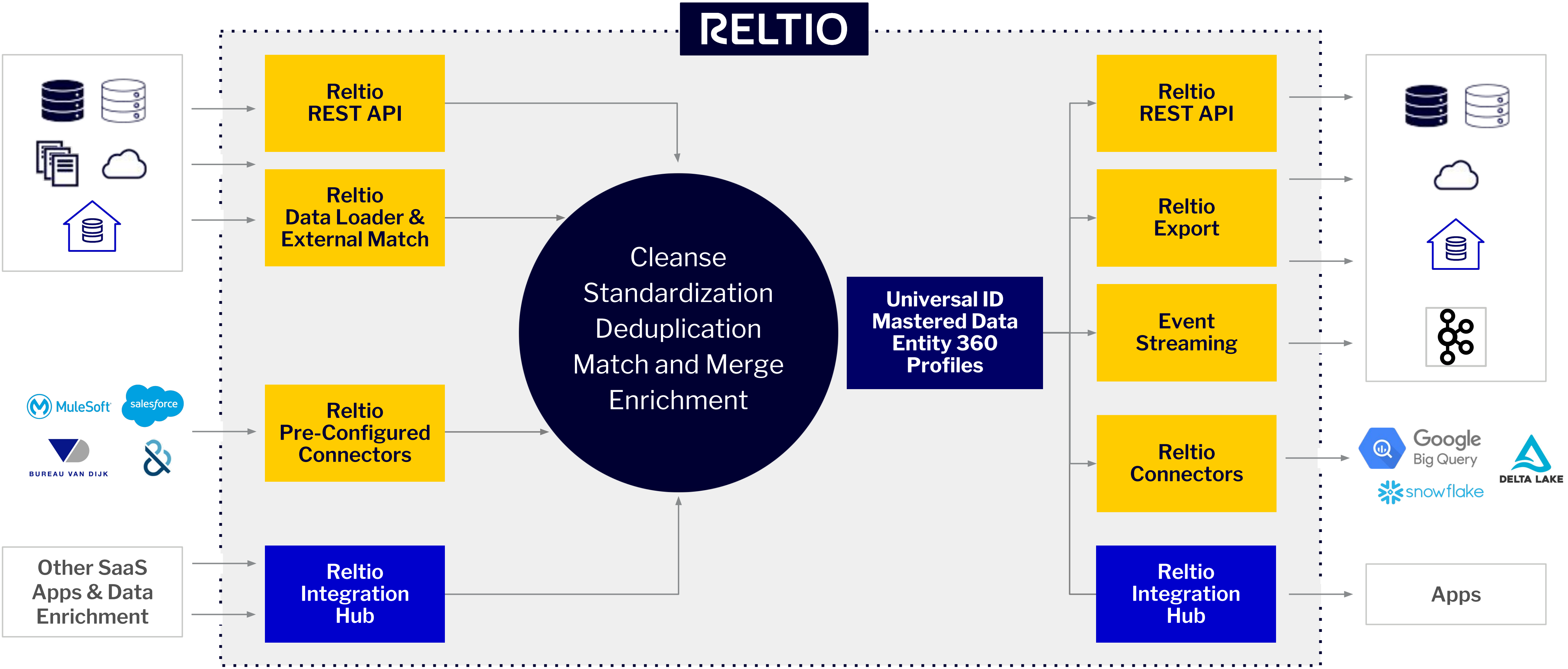
Multidomain MDM

Reltio Universal ID

ACTIVATE DATA

Growth	Efficiency	Risk & Compliance
Omnichannel Engagement Improved data quality: +4 bps revenue	**Digital Self-Service** Deflected inbound call volume: +5 bps margin	**Fraud Detection** Automated fraud detection: +5 bps mitigated revenue loss
Customer Centricity/ Targeted Marketing Higher online conversion: +11 bps revenue	**Process Automation and Optimization** Reduced FTE costs for data mastering: +9 bps margin	**Privacy and Consent Management** Improved PII handling: +5 bps revenue
Sales Effectiveness Improved B2B data quality: +6 bps revenue	**M&A Integration and Partner Collaboration** Smoother integration post M&A: +34 bps margin	**Auditing and Compliance Reporting** Simplified reporting: +4 bps margin

Reltio integrations



RELTIO

Reltio REST API

Reltio Data Loader & External Match

Reltio Pre-Configured Connectors

Reltio Integration Hub

MuleSoft

salesforce

BUREAU VAN DIJK

Other SaaS Apps & Data Enrichment

Cleanse
Standardization
Deduplication
Match and Merge
Enrichment

Universal ID
Mastered Data
Entity 360
Profiles

Reltio REST API

Reltio Export

Event Streaming

Reltio Connectors

Reltio Integration Hub

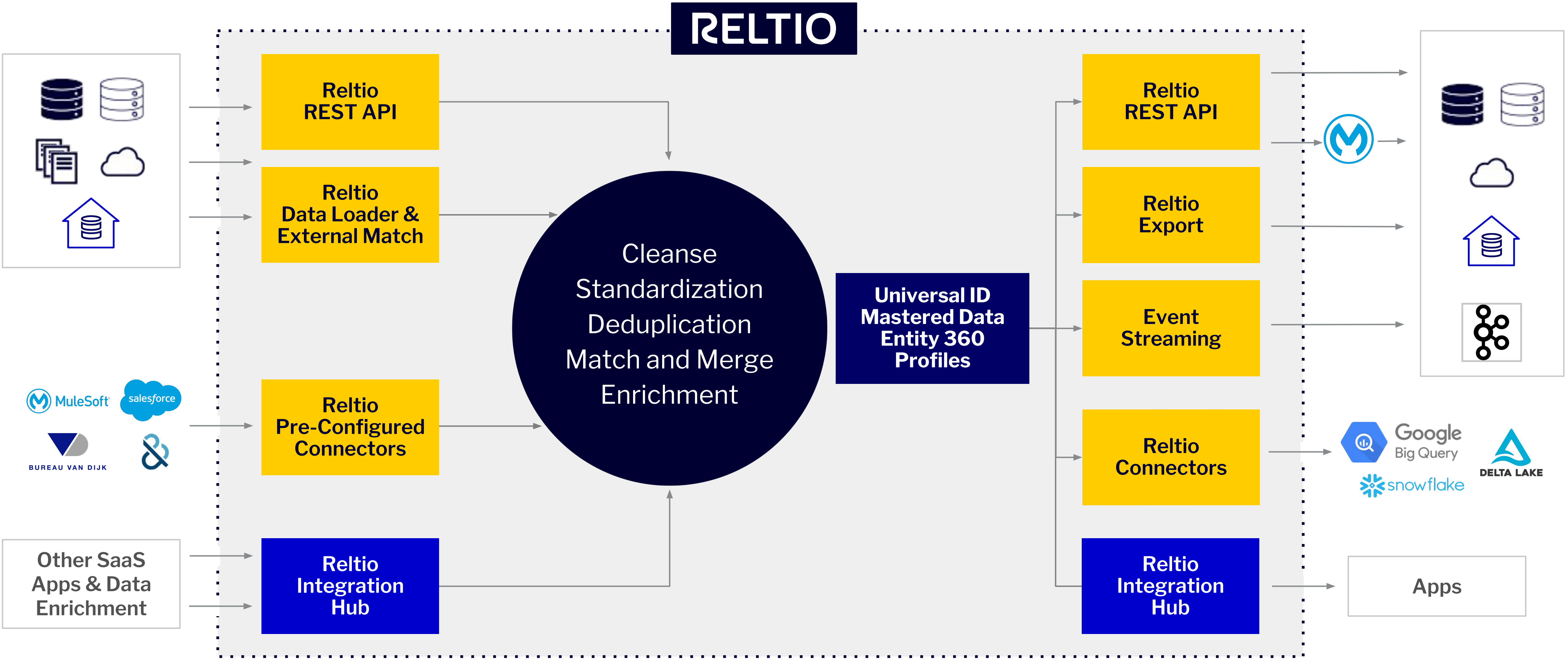
Google Big Query

snowflake

DELTA LAKE

Apps

Reltio integrations



What makes a GET API important?

Accessing Data

Reltio does a lot with data from customers. We are organizing and cleansing it. We are matching and standardizing it. We associate relationships and remap reference values across a customer's enterprise. This isn't the only value that customers are deriving though. Often their customers need this data to improve business and transactional applications. They need to get this data after Reltio has done all this heavy lifting and utilize it in the applications and analytics. This means data retrieval!! This means GETs!

Integrations

We are going to talk about the common ways to get data from Reltio:

- ID based lookup
- Search via filters
- Synchronous matching
- Synchronous multi-entity retrieval

RELTIO

Real World Metrics - Single Day transaction stats - Week of 5-1-2023

Customer #1

Total Entities: 55M

Number of Updates: 1,034,827

Average: 283 ms

P90: 321 ms

P95: 373 ms

Number of Search Connections: 1,122,655

Average: 105 ms

P90: 126 ms

P95: 158 ms

Customer #2

Total Entities: 63M

Number of Updates: 1,005,135

Average: 310 ms

P90: 425 ms

P95: 508 ms

Number of Search transactions: 653,990

Average: 97 ms

P90: 187 ms

P95: 256 ms

Customer #3

Total Entities: 1.1B

Total Number of Search API transactions: 1,597,844

Average: 142 ms

P90: 253 ms

P95: 367ms

ID Based lookup

GET by ReltioID, EntityID, CrosswalkID

Synchronous ID based lookups are the most performant read requests within the Reltio platform. Whenever possible, ID based (crosswalkID or entityID) lookups should be utilized. These requests will identify a single entity object and return it.

CrosswalkID and EntityID based lookups bypass the Reltio Index entirely. This provides a small improvement in query performance since it can read directly from the primary data storage environment. Only crosswalkID and entityID based requests have this behavior.

Highlights:

- This only applies to IDs which are stored as crosswalkID or EntityIDs. Identifiers stored as attributes do not follow this pattern and are considered search requests.
- crosswalkID and entityID based lookups are the most performant read requests within the Reltio platform. They should be utilized whenever possible.

Typical integration scenarios for this pattern are when a source system is doing a lookup to enrich it’s copy of an entity

Search

GET with filter (Search)

Synchronous search queries are some of the most common requests hitting the Reltio platform. Searches are focused on exploration and will result in the return of all entities within the platform that meet the criteria provided. They may not be ideal in identifying a specific record or performing queries where the goal is to see what would this data match / merge with if loaded into Reltio.

A search request is any GET entity request that leverages a filter component except for GET by crosswalkID or entityID. Identifiers that are stored as attributes (including sequence generated identifiers) are searches.

Searches will have higher read latency than direct crosswalkID and entityID based lookups. Searches rely on queries against the Reltio search index prior to the full retrieval of the entity objects. Additionally, search results have a significant chance of being broader and returning more objects which will also increase read latency due to the size of the returned payload.

Search requests can limit the amount of data returned in order to limit payload size.

Highlights:

- Easy lookup for any attribute persisted within Reltio
- No need for additional indexing build out for any attribute. Indexing is automatic
- Indexing is asynchronous to data load, so data is not immediately available when loaded.
- Number of returned objects can be high depending on search criteria

Synchronous Matching

What would this match with?

Synchronous matching is a way of performing “what if” analysis to identify any existing entities in Reltio that would match / merge to a record if it was POSTed to the tenant. This API interaction is not intended for data exploration or search use cases. Its use should be limited to when a system needs to do a tight non ID based lookup to identify if the platform already knows this entity. This is not a low latency request and will require additional matching engine overhead before responding.

When this request is processed, the data submitted gets submitted to the Reltio matching engine which compares it against the data within the tenant to identify matches. Only entities that return true for 1 or more match rules are returned as part of this request.

Highlights:

- This is NOT a search request. This API pattern is used to identify matches that already exist within a tenant.
- This request does not utilize the Reltio search index, but does require the submitted payload to go through the Reltio matching engine which adds significant latency on each request.
- The number and design of match rules will have a significant impact on the performance of this operation. Tenants that have a single simple match rule will respond to these requests faster than tenants that have 20 complex match rules.
- All match rules are evaluated as part of this request and any entities which result as “true” for 1 or more match rules will be returned in the request response.

Multi-Entity Search

I want to return all objects which are related to the thing I searched for. _searchConnections

Synchronous multi entity / relationship retrieval is built into the composite API _searchConnections. This API functions as a single API that searches a specific Entity Type in the platform it then retrieves all relationships connected to the search results and retrieves entities at the other end of those relationships.

Search entity1 -> GET relationships where entity1 is a party -> GET entityN at the other end of these relationships.

Performance of this API is directly related to both the number of search results returned at the beginning of the API processing AND the cardinality of the searched entities with their related entities. As the number of entities returned in the search increase, so does the response latency. Additionally, even if only 1 entity is identified in the initial search, but that entity contains a high number of relationships to other entity types, API latency will increase. If both a high number of search results are identified AND a high amount of cardinality to related entities is found for the search results, API performance will be slow.

This means that API performance is VERY data dependent and prediction of API performance will be difficult to predict without very explicit searches to start this request, and a limit to the cardinality persisted within the platform.

Multi entity retrieval with the _searchConnections API can and should leverage the entityID or crosswalkID as the starting point whenever possible. This removes the need to perform a search. This results in a 2 fold performance benefit.

- ID based lookups are faster than search based lookups as they do not need to retrieve information from the search index.
- ID based lookups limit the amount of objects identified in the first part of the composite API to a single entity. This can have dramatic impact on the performance of this API.

Documentation

API Documentation

Entity Search: https://docs.reltio.com/en/explore/get-going-with-apis-and-rocs-utilities/reltio-rest-apis/model-apis/entities-api/entity-search

Swagger doc: https://developer.reltio.com/private/swagger.htm?module=Data%20Operation#/

Swagger Developer Portal: https://developer.reltio.com/index.htm

Swagger Developer Portal:

https://developer.reltio.com/private/swagger.htm?module=Data%20Operation#/Match

Synchronous Matching

POST https://{{environmentURL}}/reltio/api/{{tenantID}}/entities/_scoredmatches?options=ovOnly

Body example:

```
[
 {
   "type": "configuration/entityTypes/Individual",
   "attributes": {
    "FirstName": [{"value": "Mitchell"}],
    "LastName": [{"value": "Burgan"}],
     "Email": [{"value":
      {
        "Email": [{"value": "mitchell_burgan@cox.net"}]
      }
     }],
     "Phone": [{"value":
      {
        "Number": [{"value": "212-940-2291"}]
      }
     }]
   }
  }
]
```

RELTIO