Extract, transform, load (ETL) MySQL to MongoDB Atlas using Data Pipeline

Tldr

Basically hierarchical to nosql data migration which involves:

- Queries to extract MySQL data into Azure Data Factory
- Pipelines to transform SQL into NoSQL
- Json data to be loaded into MongoDB

Good:

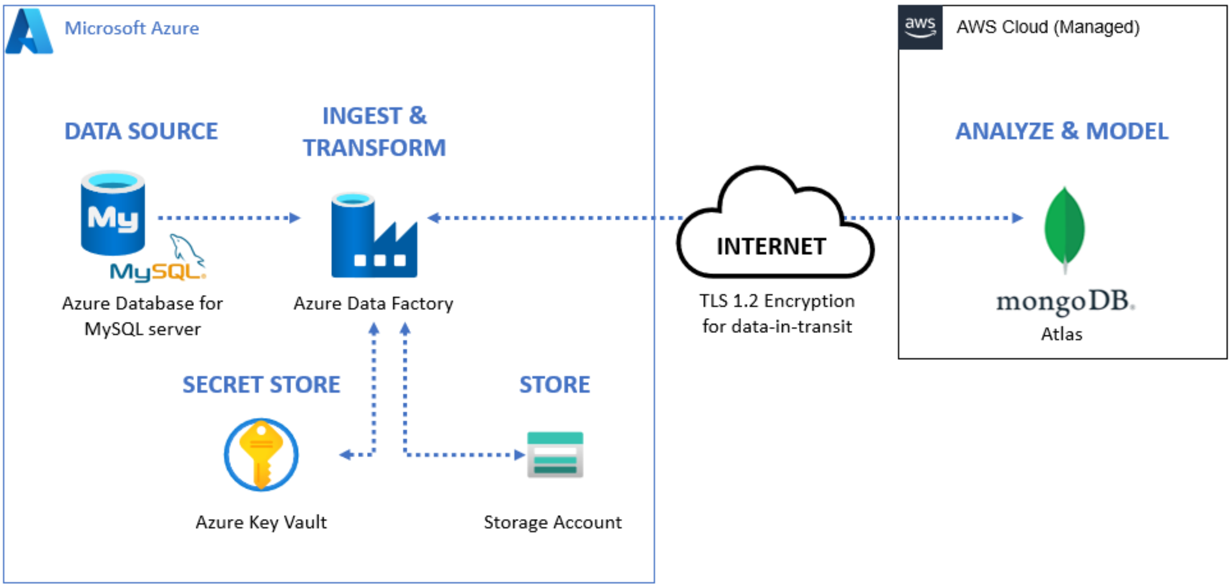
- Excellent Performance (4ms; sec.2.4)

Bad:

- Unwanted data duplication that also lead to significant update effort (sec. 3.1)
- Lack data validation (sec. 3.3)
- Lack aggregation for continuous update (sec. 3.3)
- No reverse migration (sec. 3.3)

Extract, transform, load (ETL)

Overview



Source:

ETL MySQL to MongoDB Atlas.pptx

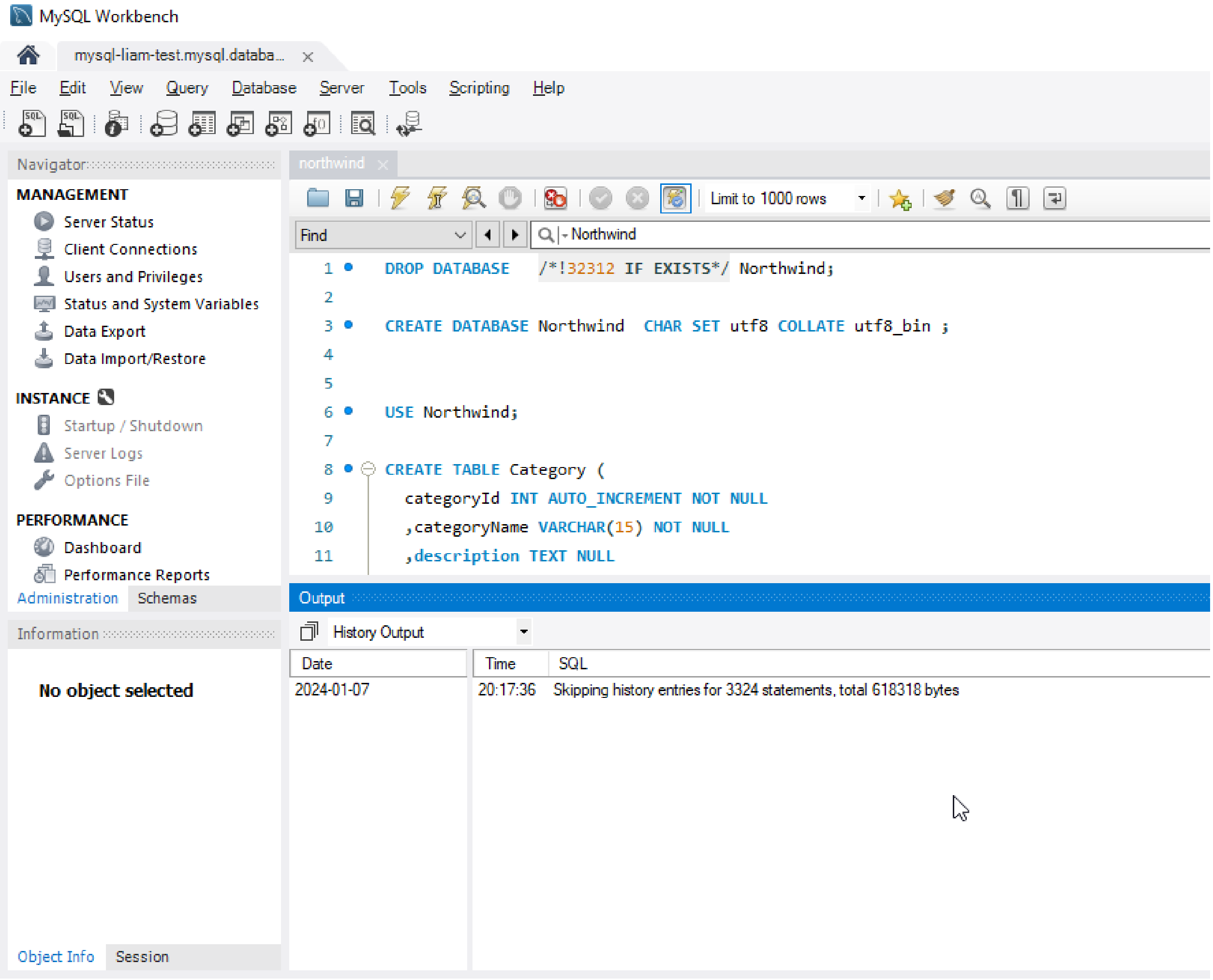
Estimate cost for 72 hours: US$16.82
https://azure.com/e/0d6d1f802abc478c9cedf6594e9a6731

Source - MySQL

Azure Database for MySQL Burstable B1s
MySQL version 8.0
Resource location: Azure East US

Configuration Procedures

1. Create a Azure Database for MySQL flexible server
2. Connect to Azure MySQL on MySQL Workbench
3. Import Northwind.sql



```
DROP DATABASE /*!32312 IF EXISTS*/ Northwind;

CREATE DATABASE Northwind  CHAR SET utf8 COLLATE utf8_bin ;


USE Northwind;

CREATE TABLE Category (
  categoryId INT AUTO_INCREMENT NOT NULL
  ,categoryName VARCHAR(15) NOT NULL
  ,description TEXT NULL
```

Date	Time	SQL
2024-01-07	20:17:36	Skipping history entries for 3324 statements, total 618318 bytes

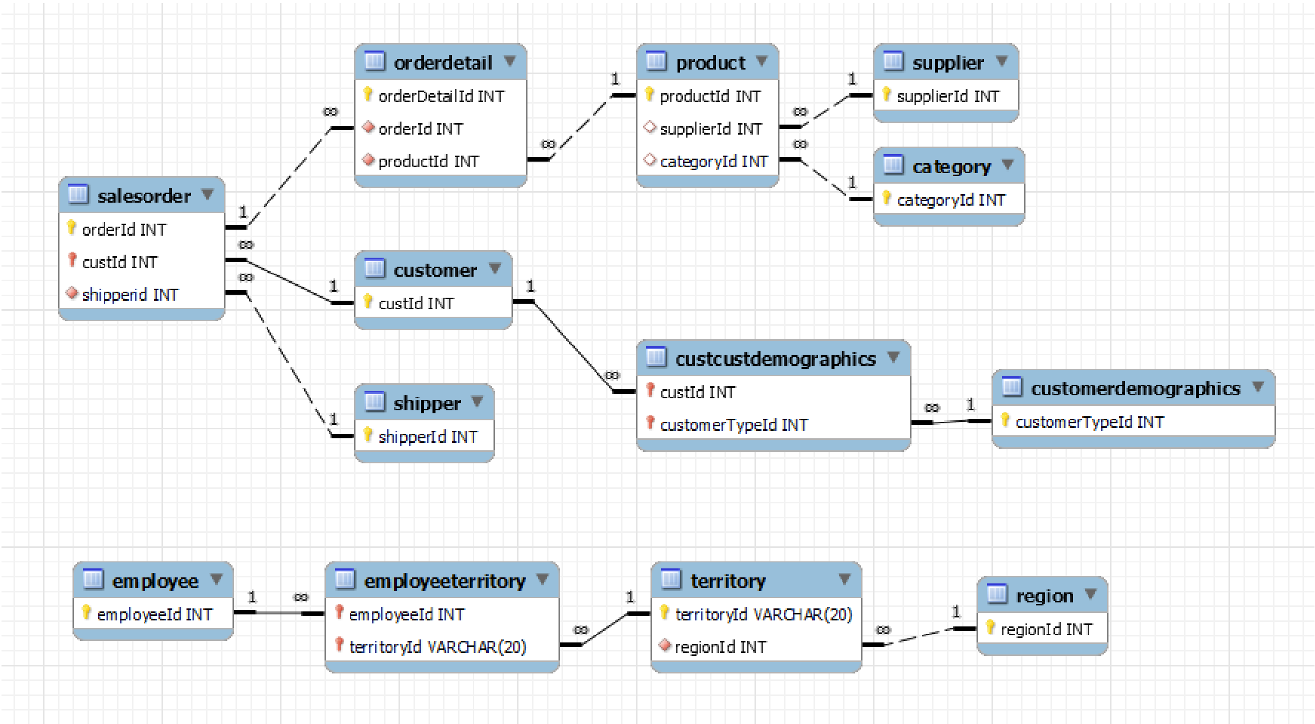
Data Reference from https://github.com/harryho/db-samples

Azure MySQL Credential

azureadmin
Z57v36VQ4sm3E2Lc5299U4mQK

Source Data Structure - ER Diagram

Procedures: Create ER Diagram using Reverse Engineer



Only Primary Keys and Foreign Keys are shown.

Destination - MongoDB

MongoDB Atlas Free tier M0 Sandbox cluster
MongoDB version 6.0
Resource location: AWS us-east-1

MongoDB Credential

liamng
Gak2QAKFi0Joq2wA

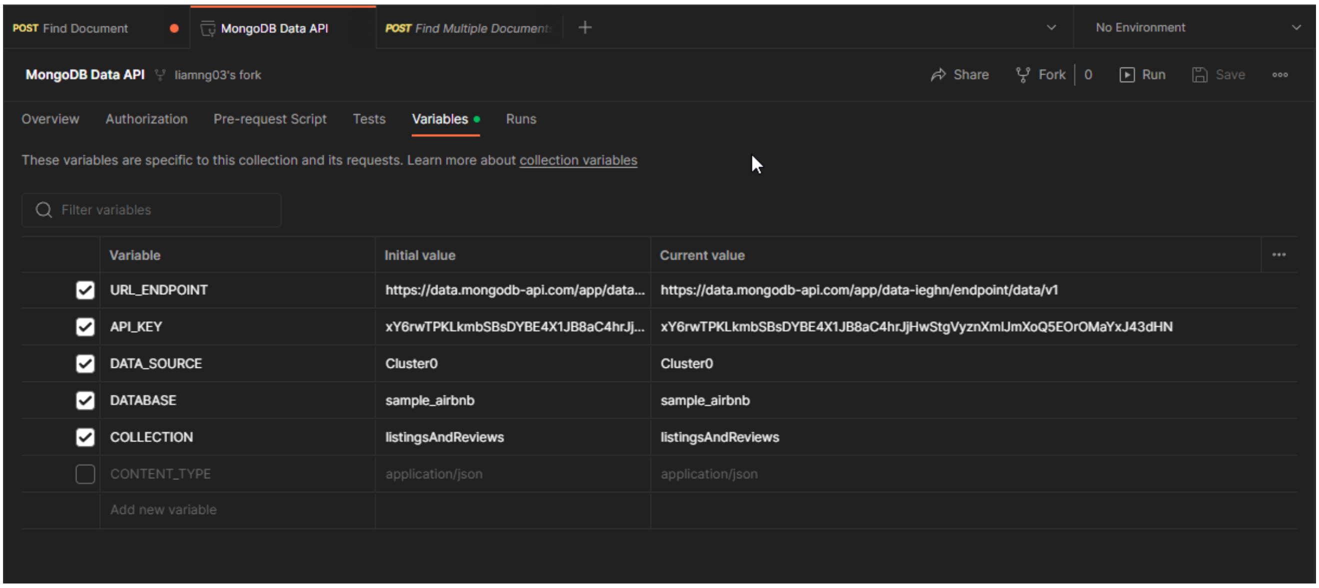
sa
NmnXthqG341zT3IR

Data API Secrets

`liam-mongo-api`
API Key: `xY6rwTPKLkmbSBsDYBE4X1JB8aC4hrJjHwStgVyznXmlJmXoQ5EOrOMaYxJ43dHN`

MongoDB API Test on Postman

Postman project variables



These variables are specific to this collection and its requests. Learn more about collection variables

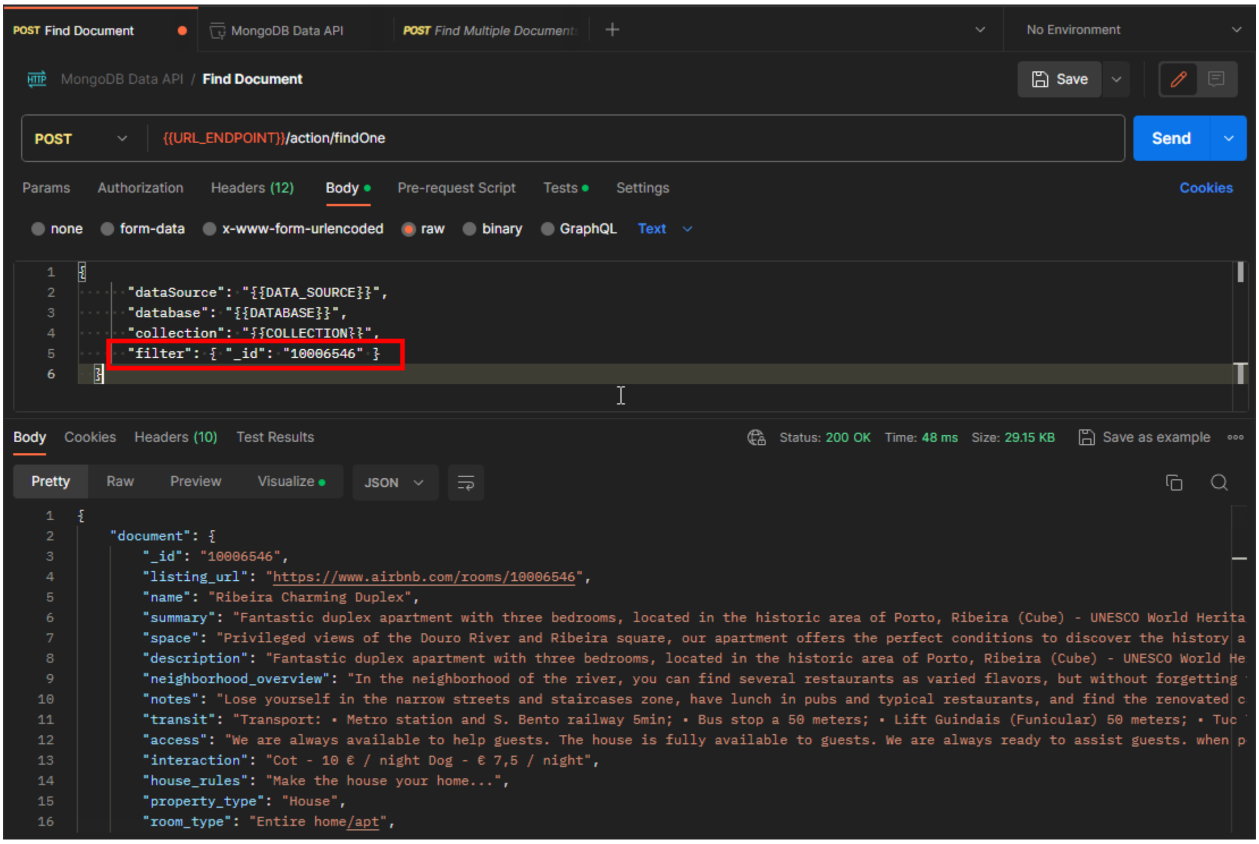
Filter variables

	Variable	Initial value	Current value
☑	URL_ENDPOINT	https://data.mongodb-api.com/app/data...	https://data.mongodb-api.com/app/data-ieghn/endpoint/data/v1
☑	API_KEY	xY6rwTPKLkmbSBsDYBE4X1JB8aC4hrJj...	xY6rwTPKLkmbSBsDYBE4X1JB8aC4hrJjHwStgVyznXmIJmXoQ5EOrOMaYxJ43dHN
☑	DATA_SOURCE	Cluster0	Cluster0
☑	DATABASE	sample_airbnb	sample_airbnb
☑	COLLECTION	listingsAndReviews	listingsAndReviews
□	CONTENT_TYPE	application/json	application/json
	Add new variable		

Error

Enable `CONTENT_TYPE` may causes error `mime: unexpected content after media subtype`

1. Open Find Document
2. Update `filter` in `Body` to one of the records
3. Verify json content



{{URL_ENDPOINT}}/action/findOne

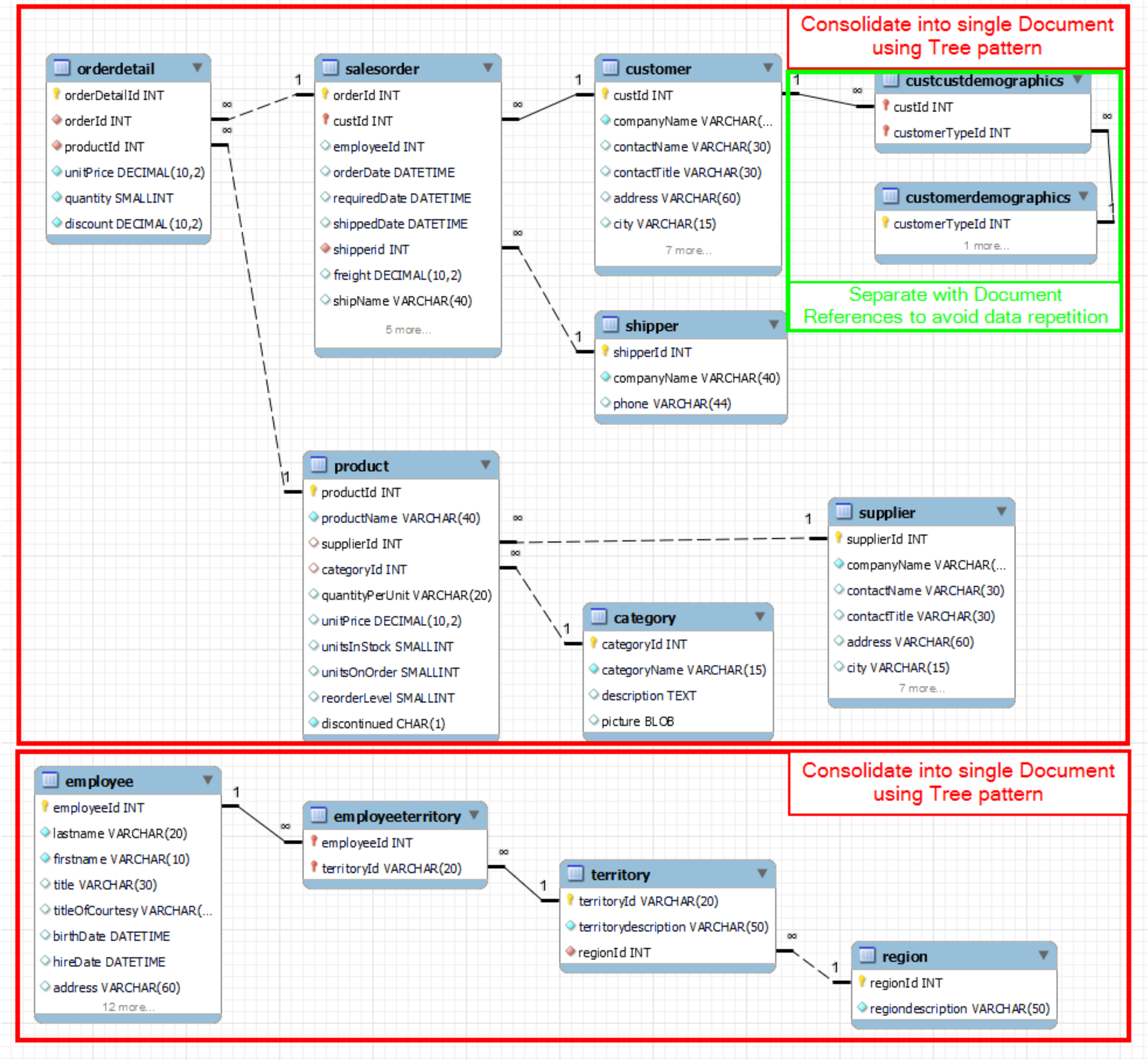
```
{
    "dataSource": "{{DATA_SOURCE}}",
    "database": "{{DATABASE}}",
    "collection": "{{COLLECTION}}",
    "filter": { "_id": "10006546" }
}
```

Status: 200 OK Time: 48 ms Size: 29.15 KB

```
{
    "document": {
        "_id": "10006546",
        "listing_url": "https://www.airbnb.com/rooms/10006546",
        "name": "Ribeira Charming Duplex",
        "summary": "Fantastic duplex apartment with three bedrooms, located in the historic area of Porto, Ribeira (Cube) - UNESCO World Herita
        "space": "Privileged views of the Douro River and Ribeira square, our apartment offers the perfect conditions to discover the history a
        "description": "Fantastic duplex apartment with three bedrooms, located in the historic area of Porto, Ribeira (Cube) - UNESCO World He
        "neighborhood_overview": "In the neighborhood of the river, you can find several restaurants as varied flavors, but without forgetting
        "notes": "Lose yourself in the narrow streets and staircases zone, have lunch in pubs and typical restaurants, and find the renovated c
        "transit": "Transport: • Metro station and S. Bento railway 5min; • Bus stop a 50 meters; • Lift Guindais (Funicular) 50 meters; • Tuc
        "access": "We are always available to help guests. The house is fully available to guests. We are always ready to assist guests. when p
        "interaction": "Cot - 10 € / night Dog - € 7,5 / night",
        "house_rules": "Make the house your home...",
        "property_type": "House",
        "room_type": "Entire home/apt",
```

Future development for my PayPal API website (Node.js 18)

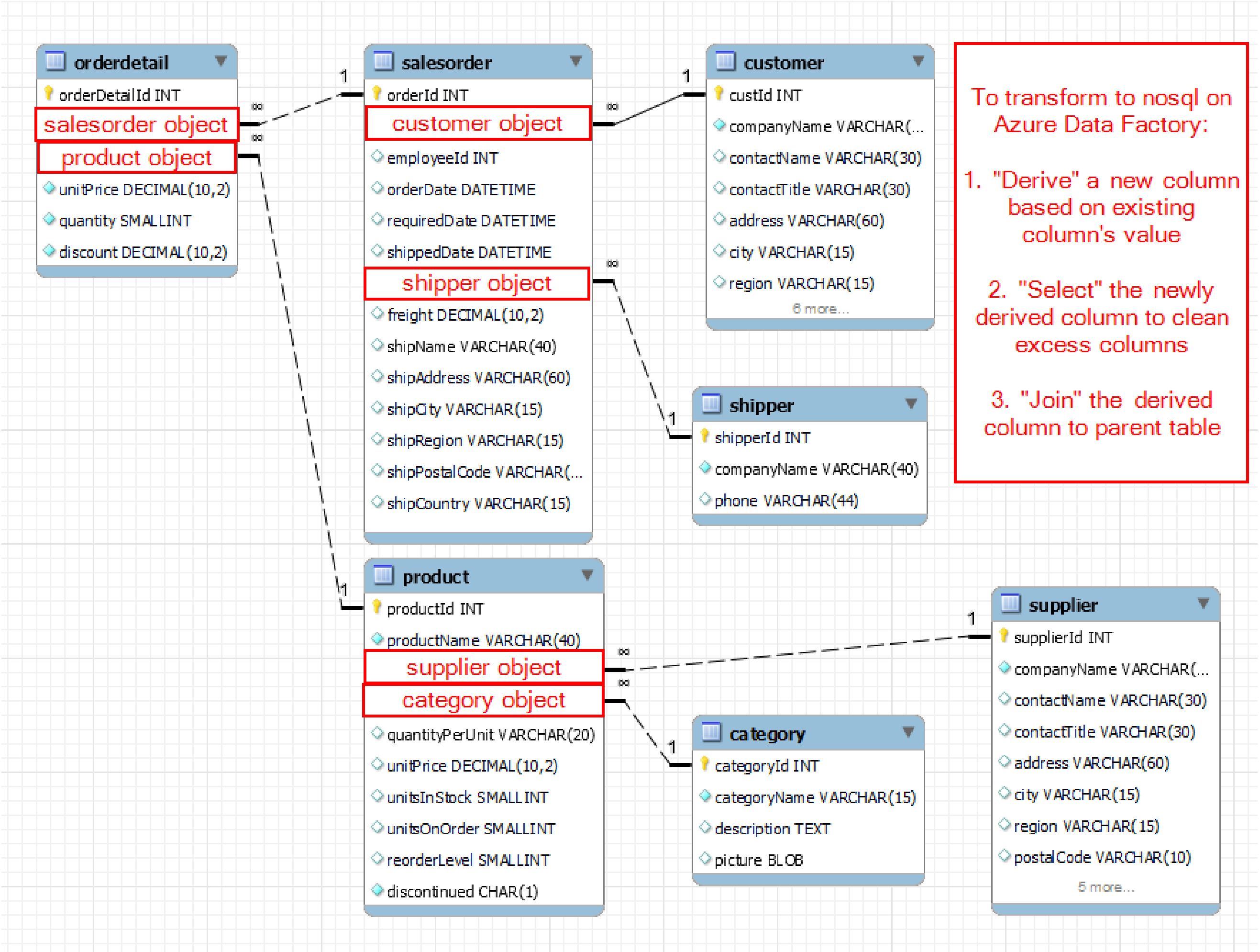
Document database schema designs



Consolidate into single Document using Tree pattern

Separate with Document References to avoid data repetition

Consolidate into single Document using Tree pattern



To transform to nosql on Azure Data Factory:

1. "Derive" a new column based on existing column's value

2. "Select" the newly derived column to clean excess columns

3. "Join" the derived column to parent table

Considerations

- Maximize number of Many-to-One (n:1) relationships to embed data in single document.
- Use references and keep the `customer demographics` information in a separate collection from the `customer` collection to avoid repetition of the demographics data,.
- Other patterns are not implemented in this project due to complexity and require more in-depth research on data modeling.

SQL-MongoDB Queries Conversion Reference

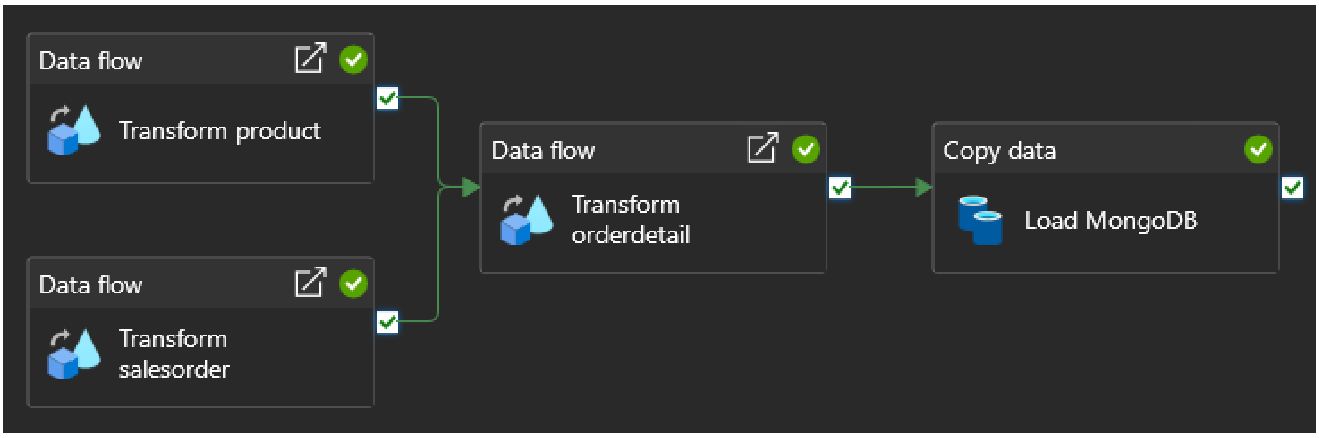
SQL query	Equivalent MongoDB query
`SELECT * FROM users`	`db.users.find({})`

More on

- Microsoft KB: Copy data from or to MongoDB using Azure Data Factory
- MongoDB KB: SQL to MongoDB Mapping Chart

Azure Data Factory (ETL Tool)

High-level ETL Steps



1. Query MySQL data into Azure Data Factory
2. Transform each table (**hierarchical**) into collection (Nosql, **Tree** pattern) in 3 steps:
   - Derive a new column based on existing column's value
   - Select the newly derived column to clean excess columns
   - Join the derived column to parent table
3. Store the transformed document into a sink (`.json`)
4. Copy the document to MongoDB

Configuration Procedures

⚠ **Pre-requisites**

- Add client address to Azure Key Vault exception
- Allow trusted Microsoft services to bypass this firewall on Key Vault
- Add Azure RBAC (**Key Vault Administrator**) for yourself/ Access Policies in Azure Key Vault
- Add Azure RBAC (**Key Vault Secrets User**) for Azure Data Factory Managed identity in Azure Key Vault
- Add Azure RBAC (**Reader**) to Azure Data Factory Managed identity in Azure Database for MySQL
- Add Azure RBAC (**Storage Blob Data Contributor**) to Azure Data Factory Managed identity in Storage Account
- SSL in Azure Data Factory is not supported
- You may need to add 0.0.0.0/0 into MongoDB `Network Access` > `IP Access List`, given Azure Data Factory (ADF) in this example is a public shared runtime, although it is possible to retrieve the public endpoints of ADF, it is unrealistic to add the extensive list of IP ranges which has over 50 entries and is also dynamic.

Note

Secret Name	Value/Connection String
mysql-conn	`Server="mysql-liam-test.mysql.database.azure.com";UserID="azureadmin";Password="Z57v36VQ4sm3E2Lc5299U4mQK";Database="northwind";`

Secret Name	Value/Connection String
mongodb-conn	`mongodb+srv://liamng:Gak2QAKFi0Joq2wA@cluster0.f67f2bk.mongodb.net/`

1. Create Azure Data Factory
2. Configure Azure Data Factory Linked Services
3. Extract and Transform tables into document-like format and repeat for all tables
4. Load data into MongoDB
5. Connect to MongoDB with VS code to verify result

Important

For Step 3, to prevent potential data loss, `left join` (outer) is preferred in queries.

Note

For step 3, as an alternative, it is possible to prepare a **sample schema** (with Tree pattern) for data sink, the columns will be mapped automatically. However, upon testing, if there is missing column, errors are expected and no fixes were found for the mapping afterwards. Further study is required.

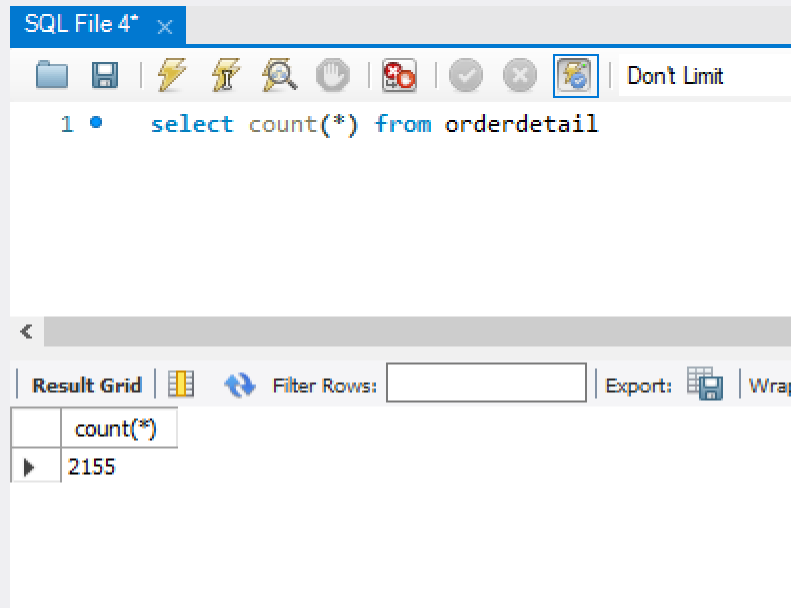
Error

- Clean `.json` file before running pipeline
- **Output to single file** must be selected for flat namespace. As a result, only Single partition can be set and this will make data flow execution longer.

Test Case

Verify total number of records

- MySQL

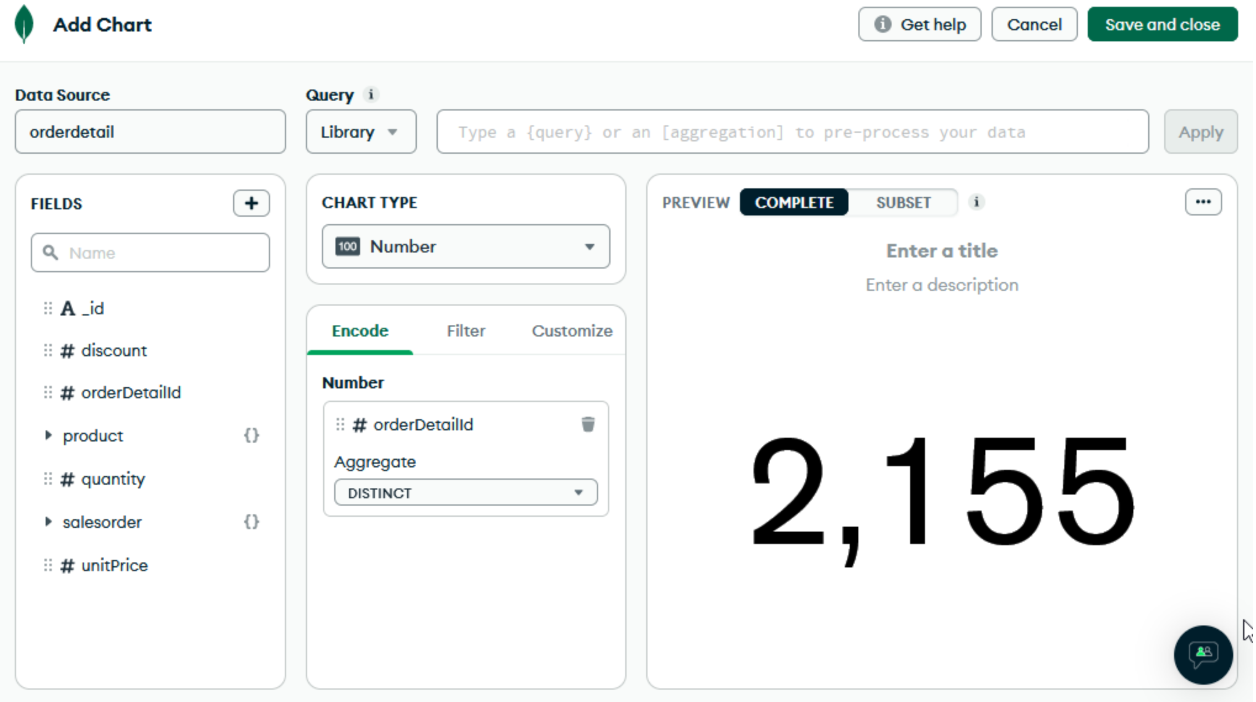


```
select count(*) from orderdetail
```

```
count(*)
2155
```

- MongoDB Chart

  Number of *Distinct* Document



Data Source: orderdetail

Query: Library

Type a {query} or an [aggregation] to pre-process your data

Apply

Chart type: Number

Number: # orderDetailId

Aggregate: DISTINCT

2,155

Number of Document

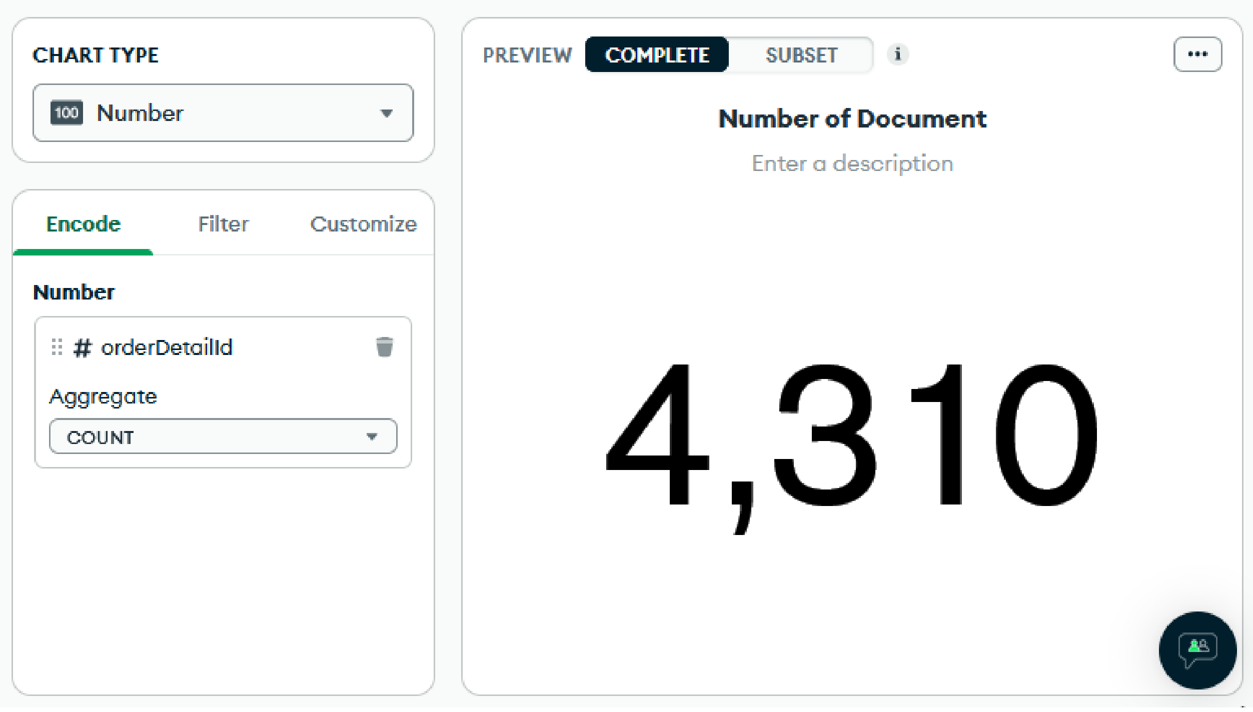


CHART TYPE

Number

Encode | Filter | Customize

Number

orderDetailId

Aggregate

COUNT

Number of Document

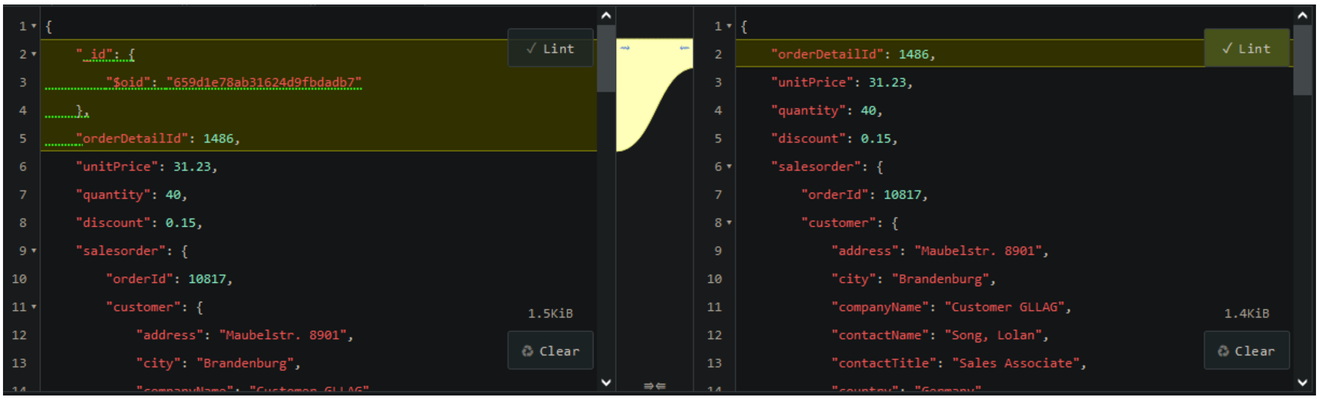
Enter a description

4,310

Since the pipeline has been run multiple times (although `Upsert` is selected), there are duplicated records in MongoDB.
To aggregate these records, see Aggregation Commands details at MongoDB KB

Schema Validation

Compare Azure Data Factory output and MongoDB record



```
{
    "_id": {
        "$oid": "659d1e78ab31624d9fbdadb7"
    },
    "orderDetailId": 1486,
    "unitPrice": 31.23,
    "quantity": 40,
    "discount": 0.15,
    "salesorder": {
        "orderId": 10817,
        "customer": {
            "address": "Maubelstr. 8901",
            "city": "Brandenburg",
```

```
{
    "orderDetailId": 1486,
    "unitPrice": 31.23,
    "quantity": 40,
    "discount": 0.15,
    "salesorder": {
        "orderId": 10817,
        "customer": {
            "address": "Maubelstr. 8901",
            "city": "Brandenburg",
            "companyName": "Customer GLLAG",
            "contactName": "Song, Lolan",
            "contactTitle": "Sales Associate",
```

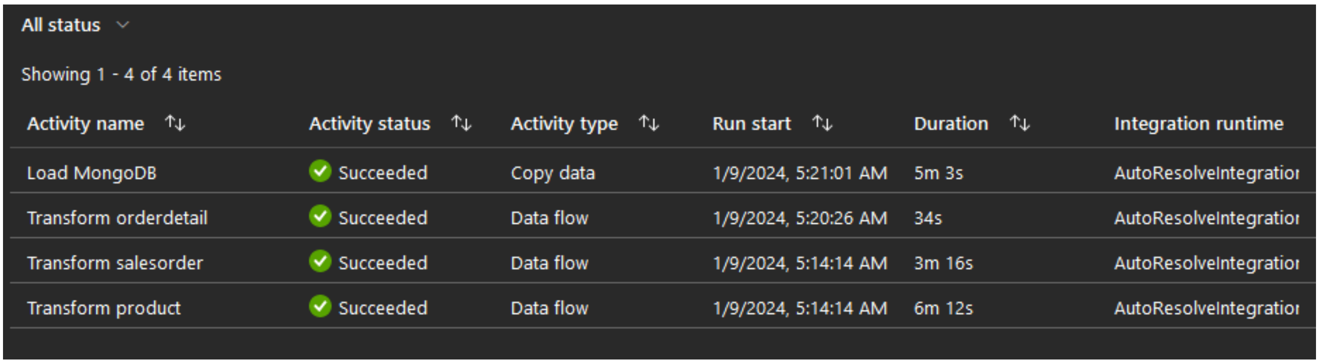
The number of column in MySQL also matches number of fields in MongoDB (except `_id`). To prevent loss of null columns in Data Factory, `'null'` text is used (also due to product limitation).

But still, due to the transformation, comparing the schema of both DB is difficult.

Azure Data Factory ETL Process Performance

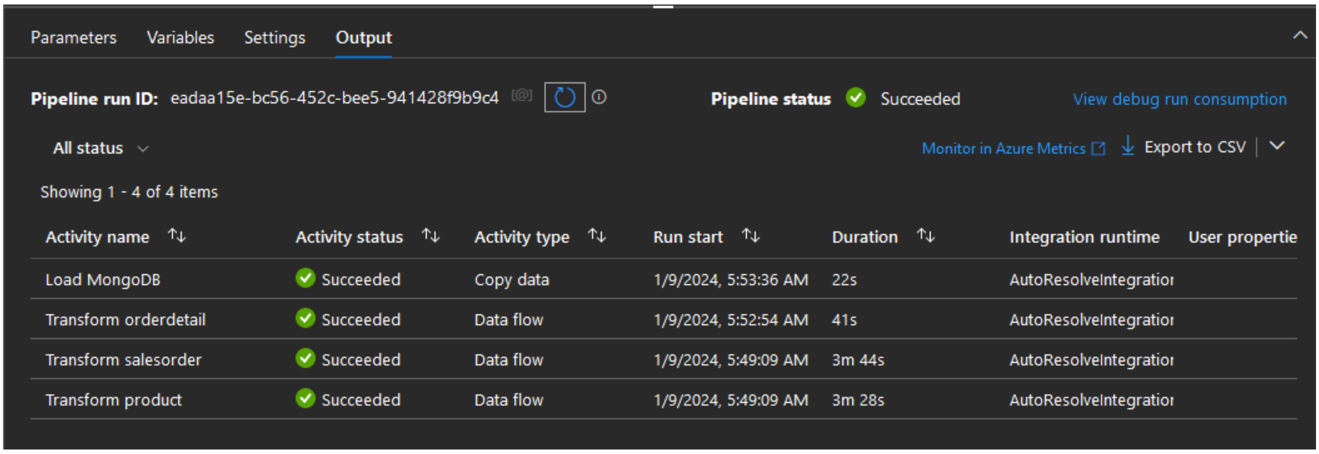
Enabled incremental column and change data capture:

First-time:



Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
Load MongoDB	✓ Succeeded	Copy data	1/9/2024, 5:21:01 AM	5m 3s	AutoResolveIntegratio
Transform orderdetail	✓ Succeeded	Data flow	1/9/2024, 5:20:26 AM	34s	AutoResolveIntegratio
Transform salesorder	✓ Succeeded	Data flow	1/9/2024, 5:14:14 AM	3m 16s	AutoResolveIntegratio
Transform product	✓ Succeeded	Data flow	1/9/2024, 5:14:14 AM	6m 12s	AutoResolveIntegratio

Subsequent:



Pipeline run ID: eadaa15e-bc56-452c-bee5-941428f9b9c4

Pipeline status ✓ Succeeded

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User propertie
Load MongoDB	✓ Succeeded	Copy data	1/9/2024, 5:53:36 AM	22s	AutoResolveIntegratio	
Transform orderdetail	✓ Succeeded	Data flow	1/9/2024, 5:52:54 AM	41s	AutoResolveIntegratio	
Transform salesorder	✓ Succeeded	Data flow	1/9/2024, 5:49:09 AM	3m 44s	AutoResolveIntegratio	
Transform product	✓ Succeeded	Data flow	1/9/2024, 5:49:09 AM	3m 28s	AutoResolveIntegratio	

Improvement in ETL process time: 40+%
However, the scale and number of pipeline execution are trivial at this moment.

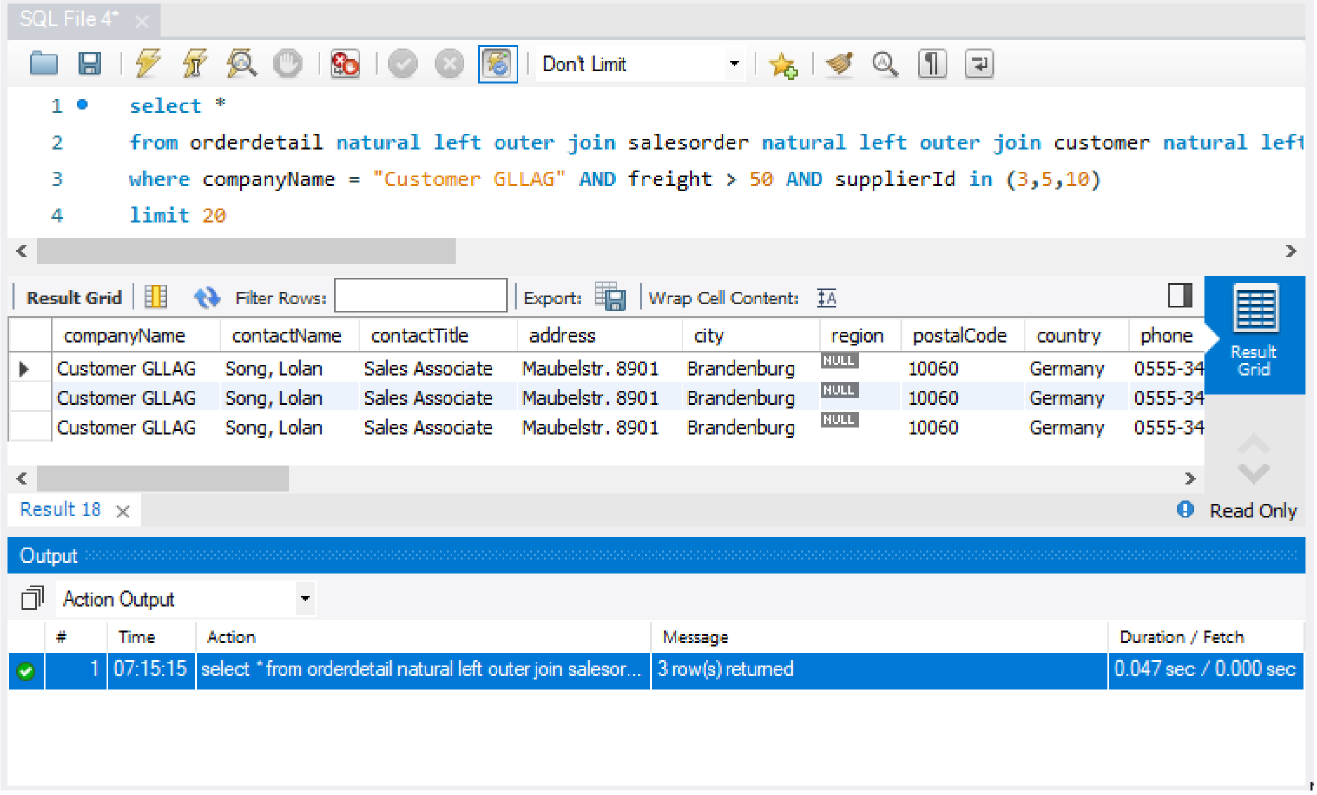
Query Performance Difference between MySQL and MongoDB

- MySQL
  - Query takes 47ms

```
Select *

From orderdetail natural left outer join salesorder natural left outer join customer
natural left outer join product natural left outer join supplier

Where companyName = "Customer GLLAG" AND freight > 50 AND supplierId in (3,5,10)
```



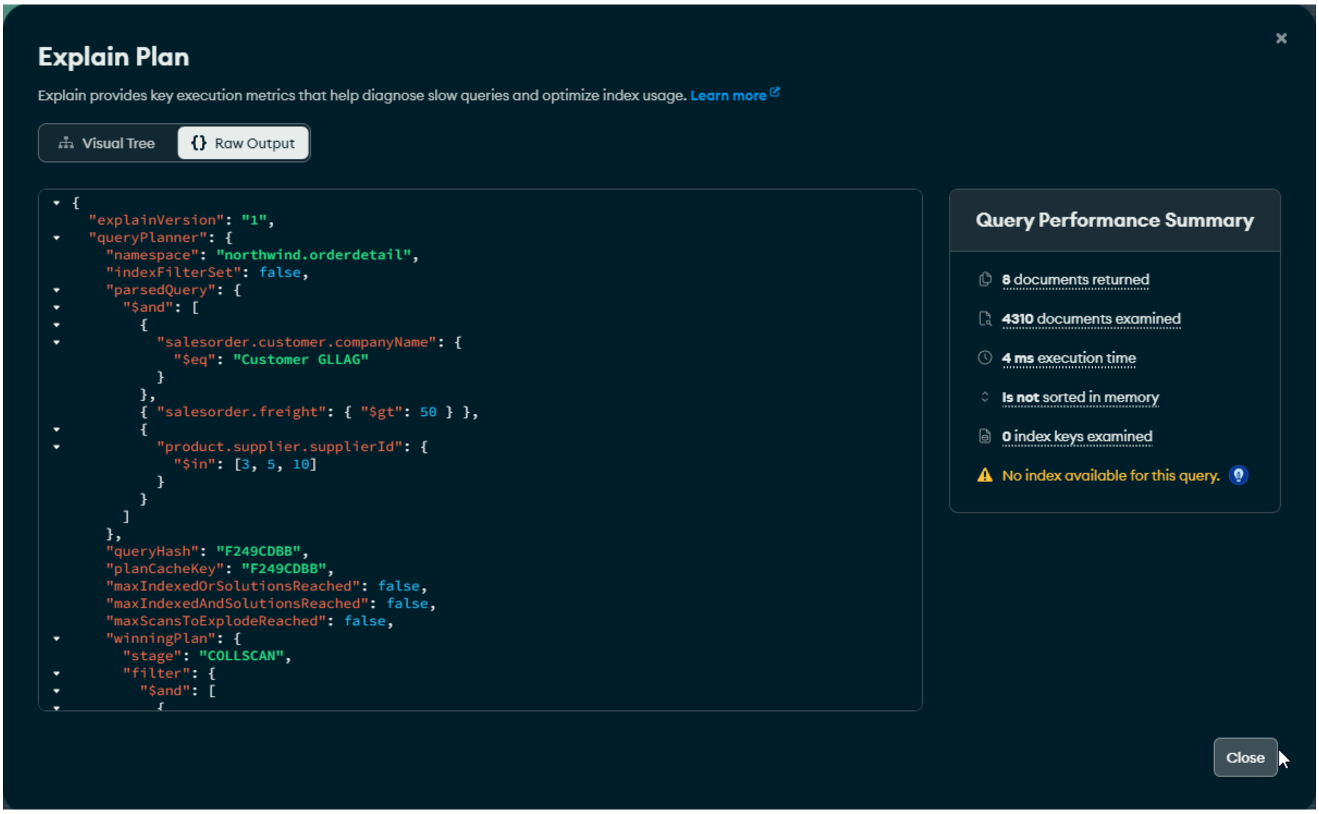
```sql
select *
from orderdetail natural left outer join salesorder natural left outer join customer natural left
where companyName = "Customer GLLAG" AND freight > 50 AND supplierId in (3,5,10)
limit 20
```

companyName	contactName	contactTitle	address	city	region	postalCode	country	phone
Customer GLLAG	Song, Lolan	Sales Associate	Maubelstr. 8901	Brandenburg	NULL	10060	Germany	0555-34
Customer GLLAG	Song, Lolan	Sales Associate	Maubelstr. 8901	Brandenburg	NULL	10060	Germany	0555-34
Customer GLLAG	Song, Lolan	Sales Associate	Maubelstr. 8901	Brandenburg	NULL	10060	Germany	0555-34

#	Time	Action	Message	Duration / Fetch
1	07:15:15	select * from orderdetail natural left outer join salesor...	3 row(s) returned	0.047 sec / 0.000 sec

- MongoDB
  - 4ms to search through 4.3k documents/ records
  - 10 times faster than MySQL

Query string `{"salesorder.customer.companyName": "Customer GLLAG", "salesorder.freight": {"$gt": 50}, "product.supplier.country": {"$in": ["Japan", "Germany", "USA"]}}`



Explain Plan

Explain provides key execution metrics that help diagnose slow queries and optimize index usage. Learn more

```
{
  "explainVersion": "1",
  "queryPlanner": {
    "namespace": "northwind.orderdetail",
    "indexFilterSet": false,
    "parsedQuery": {
      "$and": [
        {
          "salesorder.customer.companyName": {
            "$eq": "Customer GLLAG"
          }
        },
        { "salesorder.freight": { "$gt": 50 } },
        {
          "product.supplier.supplierId": {
            "$in": [3, 5, 10]
          }
        }
      ]
    },
    "queryHash": "F249CDBB",
    "planCacheKey": "F249CDBB",
    "maxIndexedOrSolutionsReached": false,
    "maxIndexedAndSolutionsReached": false,
    "maxScansToExplodeReached": false,
    "winningPlan": {
      "stage": "COLLSCAN",
      "filter": {
        "$and": [
          {
```

Query Performance Summary

- **8** documents returned
- **4310** documents examined
- **4 ms** execution time
- **Is not** sorted in memory
- **0** index keys examined
- ⚠ No index available for this query.

Discussion

Data Model

For One-to-Many relationship, embedding is preferred BUT putting all into single document is very likely to be considered anti-pattern, in terms of data duplication, administrative burden from updating documents, and data privacy/ security due to lack of segregation;

e.g. Every time we need to update information about customer, we'll need to update the document for every orderDetail

This is unlikely for these information to be frequently displayed or updated together every time.

Good:

- Excellent Performance (4ms)

Bad:

- Data duplication
- Large update effort

There are 3 approaches to this problem:

1. Embed (single document)
2. Reference (separate collections)
3. **Extended Reference** (separate collections but with duplicated data)

Fix:
Balance of storage cost and performance

- Separate `product`, `customer`, `shipper` from `orderdetail`
- Keep frequently access information in `orderdetail` document, such as, `productName`, `companyName`, `contactName`
- Reference new collections with `$lookup` operations

Data Transformation

- Tradeoff between simplicity and performance.
  - In the case of the tree pattern, you get better performance by avoiding multiple joins, however, you will need to manage the updates to your graph. Plus, excess/ duplicated data across many documents.

Combined with conclusion in previous section, these are the rule of thumbs:

- **1**: Favor embedding unless there is a compelling reason not to.
- **2**: Needing to access an object on its own is a compelling reason not to embed it.
- **3**: Avoid joins and lookups if possible, but don't be afraid if they can provide a better schema design.

- 4: Arrays should not grow without bound. If there are more than a couple of hundred documents on the many side, don't embed them; if there are more than a few thousand documents on the many side, don't use an array of ObjectID references. High-cardinality arrays are a compelling reason not to embed.

As always, with MongoDB, how you model your data depends entirely on your particular application's data access patterns. You want to structure your data to match the ways that your application queries and updates it.

Data Consistency and Replication

- Lack Data Validation, very difficult to compare apple-to-apple… need a solution
- Lack Aggregation on MongoDB side? every time the pipeline run (copy operation) will duplicate data set. No continuous/ incremental update to MongoDB.
- Potential improvement with `Trigger` for pipeline execution when data / schema update is performed on source MySQL DB. (ADF CDC is not yet supported for MongoDB)
    - Timer trigger for pipeline execution in regular basis
    - Event-driven trigger, and if Data Lake is used, could benefit streaming data
- No Reverse Migration native function on Azure Data Factory (from MongoDB back to MySQL)

Others

1. Scalability and Performance
    1. Hierarchical namespace would allows optimized partition + parallel process and make data flow execution faster
2. Security Compliance and Privacy

Appendix

Source Files

In chronological build order

Key Vault

ExportedTemplate-kv-liam-test.zip

Storage Account (Blob)

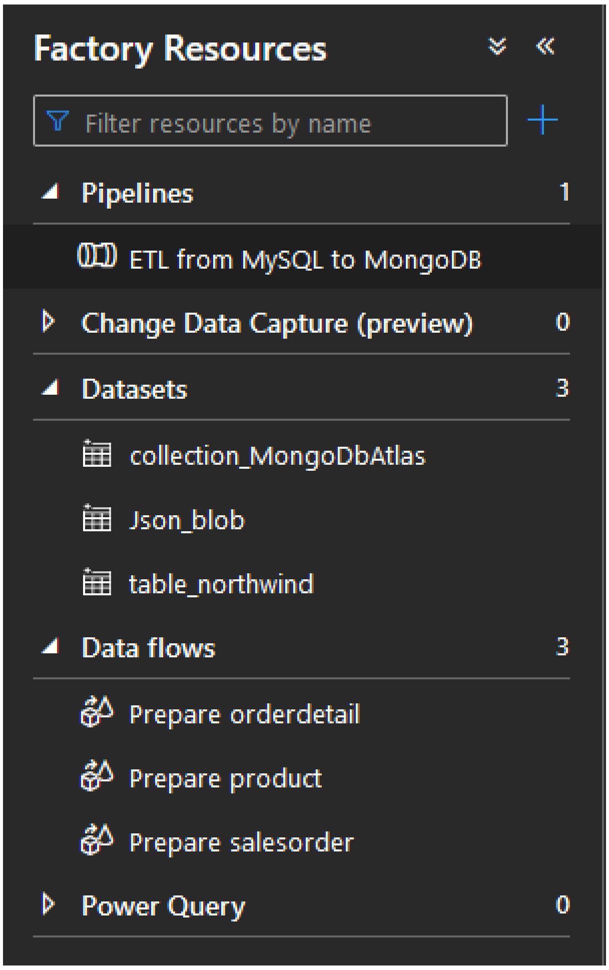
ExportedTemplate-stliamtestdf.zip

Azure Database for MySQL

ExportedTemplate-mysql-liam-test.zip

Azure Data Factory

arm_template_adf-liam-test.zip



Factory Resources

Filter resources by name

Pipelines 1

ETL from MySQL to MongoDB

Change Data Capture (preview) 0

Datasets 3

collection_MongoDbAtlas

Json_blob

table_northwind

Data flows 3

Prepare orderdetail

Prepare product

Prepare salesorder

Power Query 0

dataset_table_northwind.json

dataset_Json_blob.json

dataset_collection_MongoDbAtlas.json

dataflow_Prepare product.json

dataflow_Prepare salesorder.json

dataflow_Prepare orderdetail.json

ETL from MySQL to MongoDB.json

MongoDB

Sample Schema

```
{
  "_id": {
    "$oid": "659d1e78ab31624d9fbdadb7"
  },
  "orderDetailId": {
    "$numberInt": "1486"
  },
  "unitPrice": {
    "$numberDouble": "31.23"
  },
  "quantity": {
    "$numberInt": "40"
  },
  "discount": {
    "$numberDouble": "0.15"
  },
  "salesorder": {
    "orderId": {
      "$numberInt": "10817"
    },
    "customer": {
      "address": "Maubelstr. 8901",
      "city": "Brandenburg",
      "companyName": "Customer GLLAG",
      "contactName": "Song, Lolan",
      "contactTitle": "Sales Associate",
      "country": "Germany",
      "custId": {
        "$numberInt": "39"
      },
      "phone": "0555-34567",
      "postalCode": {
        "$numberInt": "10060"
      }
    },
    "employeeId": {
      "$numberInt": "3"
    },
    "orderDate": "2008-01-06 00:00:00",
    "requiredDate": "2008-01-20 00:00:00",
    "shippedDate": "2008-01-13 00:00:00",
    "shipper": {
        "companyName": "Customer GLLAG",
        "phone": "0555-34567",
        "shipperid": {
          "$numberInt": "2"
        }
      },
      "freight": {
        "$numberDouble": "306.07"
      },
      "shipName": "Destination RMBHM",
      "shipAddress": "Maubelstr. 1234",
      "shipCity": "Brandenburg",
      "shipRegion": "null",
      "shipPostalCode": {
        "$numberInt": "10209"
      }
    },
    "product": {
      "productId": {
        "$numberInt": "26"
      },
      "productName": "Product HLGZA",
      "supplier": {
        "address": "Tiergartenstraße 3456",
        "city": "Berlin",
        "companyName": "Supplier ZPYVS",
        "contactName": "Jain, Mukesh",
        "contactTitle": "Sales Manager",
        "country": "Germany",
        "phone": "(010) 3456789",
        "postalCode": {
          "$numberInt": "10016"
        },
        "supplierId": {
          "$numberInt": "11"
        }
      },
      "category": {
        "categoryId": {
          "$numberInt": "3"
        },
        "categoryName": "Confections",
        "desc": "Desserts, candies, and sweet breads"
      },
      "quantityPerUnit": "null",
      "unitPrice": {
```

```
      "$numberDouble": "31.23"
    },
    "unitsInStock": "null",
    "unitsOnOrder": "null",
    "reorderLevel": "null",
    "discontinued": "null"
  }
}
```