Informatica Multidomain MDM

Performance Whitepaper

For Informatica Multidomain MDM releases v10.1 and v10.2

Publication date: December 2017

Contents

Executive Summary

Informatica Multidomain MDM is designed to process large volumes of data. Hundreds of enterprises use Multidomain MDM to create and maintain complete, accurate views of complex business-critical master data. Since the launch of Multidomain MDM in 2002, the size of Multidomain MDM implementations has grown significantly and Multidomain MDM has scaled to handle those volumes.

The largest volumes of data managed in Multidomain MDM are typically Business-to-Individual use cases, where Individual may represent a consumer, a patient, a guest, a citizen, a traveler, or any one of the other 23 variations of persons in Business-to-Individual masters that enterprises manage with Multidomain MDM.

Multidomain MDM loads source data, then matches and consolidates records to deliver master data with the latency each enterprise needs – including real-time read-write operations, streamed message-based tasks, and batch interfaces that move master data in bulk DBMS feeds or in file loads.

Multidomain MDM large volume use cases

Hundreds of enterprises use Multidomain MDM to manage tens of millions of records, and many of those customers manage hundreds of millions of records, via a combination of batch, streamed and real- time feeds in to and out of Multidomain MDM.

Here are just a few examples[1]:

1. A well-known North American premium retailer uses Multidomain MDM to master their customer data to provide their customers with a unified experience across every channel, providing them with a seamless experience across every touch point, including social media.

   Every time a customer interacts with the retailer on any channel the customer profile is retrieved from Multidomain MDM in real-time, and updates to customer profiles flow directly back to MDM. The system has consolidated **over 100M source records** to provide the Total Customer Relationship view that the retailer needed.

   Because Multidomain MDM is at the heart of every customer interaction for this retailer, it runs in 24X7 mode using Informatica's Zero Downtime module providing the ability to upgrade any part of the system (including hardware, operating system, MDM, application server) without requiring system downtime.

2. A global leisure and entertainment company wanted to design immersive and personalized experiences for their guests to ensure that guests felt cared for and important before, during, and after their visit. The problem was that the company's transactional systems had **800M**

[1] For more customer use cases, see https://www.informatica.com/about-us/customers/customer-success-stories.html#fbid=5bT-1ZS9IWr?hashlink=filter-infa%253Aproduct.infa%253Aproduct/mdm

guests, and what they needed was a unified service that enables multiple segments of the company to securely store and share a consistent, single profile per guest.

The company created a flexible data management infrastructure powered by Multidomain MDM to help 14,000 employees identify any guest in less than 400 milliseconds, and put that information into action to provide the best, most seamless customer service possible.

As a result, the company has:

- Reduced data redundancy across 800 million transactional guests by 60%
- Gained insight into customers' needs
- Enabled employees to drive further revenue and focus on decision-making.

3. A multinational enterprise information technology company uses Multidomain MDM to master customer data for millions of organizations and consumers, totaling **350M records** after consolidation (close to 1 billion source records), in support of enterprise acquisitions and divestitures.

History of Multidomain MDM Performance

Over the years, the size of Informatica Multidomain MDM implementations has grown significantly, and Informatica has invested time and resources in ensuring that the software could scale to handle growing volumes at the required latencies. The graph in Figure 1 shows how the total run times for large Initial Data Load batches have decreased from release to release.

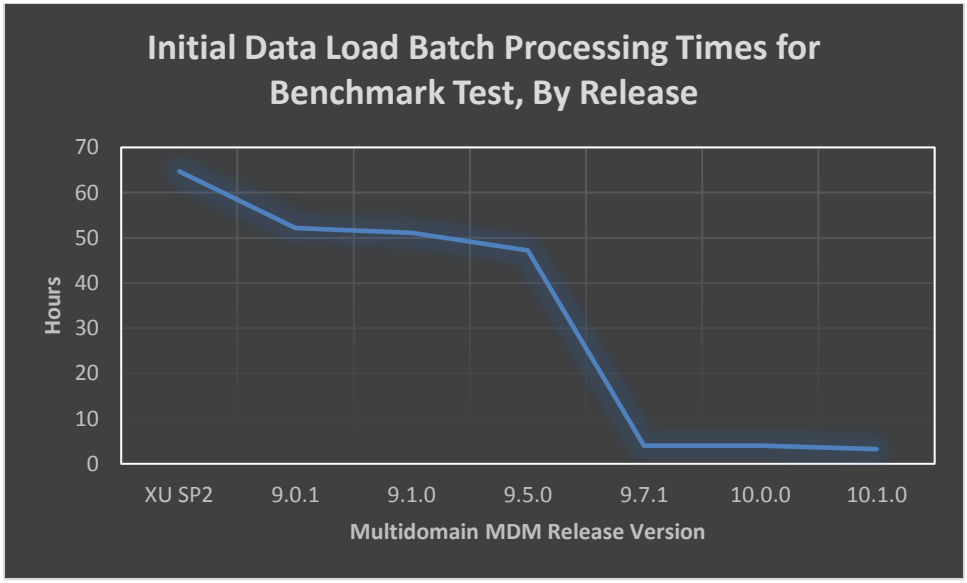


Figure 1: Initial Data Load Batch Processing Times for Oracle DBMS[2]

[2] For consistency of comparison to older releases, only the performance results for Multidomain MDM with Oracle are shown here.

The biggest performance gains in recent years were made by moving to a new architecture in 9.7.1 (note the significant dip in processing times for 9.7.1 release in Figure 1), which moved all processing out of the database tier, freeing MDM from process scalability constraints in the database tier for data onboarding, match, consolidation and outbound integration.

The performance gains resulting from the move to the database independent architecture were experienced in both the batch processes as well as the real time API (a.k.a. MDM Services Integration Framework) in the 9.7 release. In Multidomain MDM v10, composite APIs called Business Entity Services were introduced to achieve even greater performance gains. This composite API hides the underlying relational data model and provide developers with a simple RESTful API to code against, and that also reduces the overhead of making multiple calls to the more granular Services Integration Framework API.

Figure 2 illustrates how using Business Entity Services instead of the Services Integration Framework reduces the number of API calls that need to be made from a client application to Multidomain MDM. In this example, a Customer business entity includes tables for Customer, Contact, Contact Preference, Roles, Address as well as an Address Relationship table for a many-to-many relationship between Customer and Address. To use Business Entity Services to fully create a new Customer that has five Contact records, five Contact Preference records, five Roles and five Address records, a single call to the Business Entity Service for Customer is required. To use Services Integration Framework to do the same would require 26 API calls (1 for Customer and five for each of the five related tables).

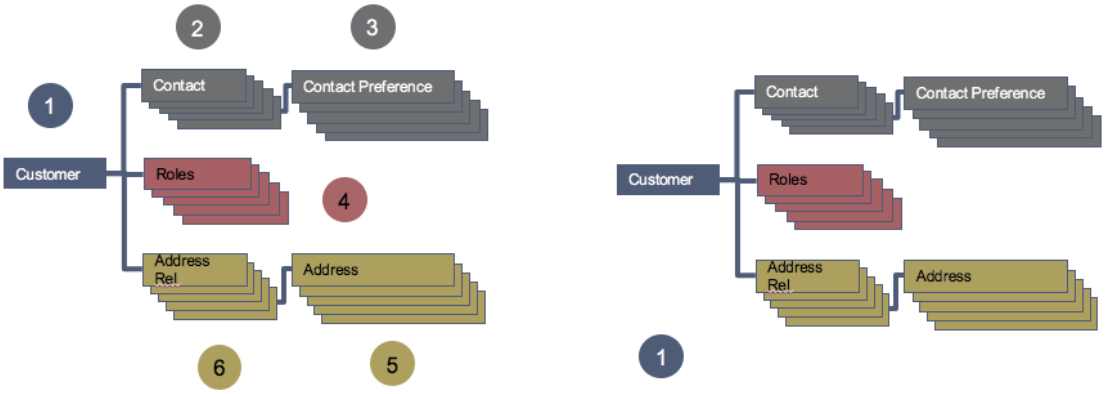


Figure 2: Difference in number of calls required to create a Customer using Services Integration Framework (*left*) compared to Business Entity Services (*right*).

Scalability

Informatica Multidomain MDM allows for horizontal (*add more servers*) and vertical (*multi-threaded to leverage more CPUs*) scalability. Multidomain MDM also includes built-in load balancing capabilities to distribute batches of data cleansing and matching work over multiple servers[3].

For more information on deploying Multidomain MDM for scalability and high availability, see the Infrastructure Planning Guide at https://kb.informatica.com/proddocs/Product%20Documentation/5/MDM_102_InfrastructurePlanningGuide_en.pdf .

High Availability

High availability is the ability of a system to continue functioning after the failure of one or more of the servers. You can achieve high availability of Multidomain MDM either with multiple standalone application server instances or with application server clusters.

For more information on deploying Multidomain MDM for scalability and high availability, see the Infrastructure Planning Guide at https://kb.informatica.com/proddocs/Product%20Documentation/5/MDM_102_InfrastructurePlanningGuide_en.pdf .

Zero Downtime

Zero Downtime is an add-on capability for Multidomain MDM that supports the continuous operations of Multidomain MDM during planned downtime activities, such as the upgrade or modification of the MDM infrastructure, including hardware, software and/or data models. The requirement of a zero-downtime solution is to ensure that there is no disruption of services for the end users of the applications built on top of Multidomain MDM.

For more information on the Zero Downtime add-on option, see the Zero Downtime Upgrade Guide at https://kb.informatica.com/proddocs/Product%20Documentation/4/MDM_1010_ZeroDowntime[ZDT]UpgradeGuide_Oracle_en.pdf .

[3] For best practices on tuning for performance, see https://kb.informatica.com/proddocs/Product%20Documentation/5/MDM_102_PerformanceTuningGuide_en.pdf

Appendix A: Multidomain MDM 10.x Performance Benchmark Summary with Oracle DBMS[4]

Approach to MDM performance tests

Informatica's performance benchmarking for Multidomain MDM is typically carried out as part of each major release and provides release-on-release comparisons to ensure that successive revisions and optimizations in the product continue to build on the successes of previous releases. This also provides companies that have an older version of Multidomain MDM with an indication of the sorts of performance improvements they should expect when they upgrade to the latest release of Multidomain MDM.

All performance tests use the same data set, data model and rules. The benchmark for the newest release of the software and the previous version are run on the same hardware with the same DBMS, application server, operating system and network.

Performance testing environment

The testing environment has two application servers, one database server and one machine that drives the tests. Figure 3 illustrates this.

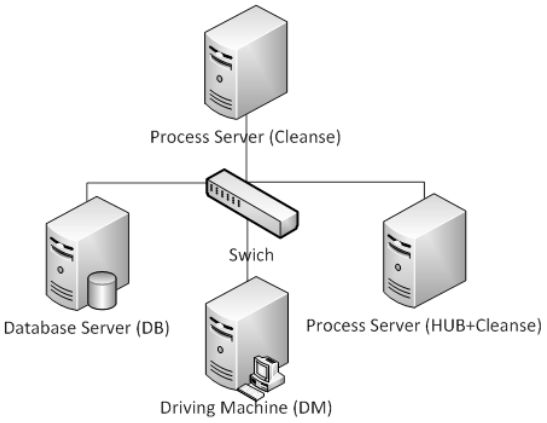


Figure 3: Batch performance testing machines

All the machines are interconnected via a 10G switch as private network that isolates the traffic from the other machines from office network. Net speed for the application servers was set to 1G for all batch testing, and the database server has 10G network speed.

[4] Performance benchmarks with DB2 and SQLServer are available on request

Hardware configuration

The hardware configuration in this section is what the performance benchmark tests are run on

DB Server	
CPU	2 * Xeon E5-2630 2.30GHz, 12 cores/24 threads
Physical Memory	64G (8*8Gb 800 GHz)
Storage Disk0(System)	223GB: 2*240GB SSD RAID1 (Areca ARC-1882IX RAID controller)
Storage Disk1(DB Data)	2.18TB: 10*240G SSD RAID0 (Areca ARC-1882IX RAID controller)
OS	Windows 2008 Server R2 (64-bit) Ent, SP1, V6.1.7601
Installs	Oracle 11.2.0.3, DB2 9.7 HF6, SQL Server 2012 SP1
Testing Schema	Standard 50 million records Customer base object
NETWORK (for testing management)	1Gb (integrated Intel 4 port Network Adapter)
NETWORK (for testing itself)	10Gb (Intel Ethernet Converged Network Adapter X540-T2)

App Servers (HUB/Process Server) – 2 app server machines	
CPU	Each: 2 * Xeon E5-2430 2.20GHz, 12 cores/24 threads
Physical Memory	Each: 64G (4*16Gb 1333 GHz)
Storage Disk0(System)	Each: 1*1Tb TOSHIBA MG03ACA100 ATA (non-RAID)
OS	Each: Windows 2008 Server R2 (64-bit) Ent, SP1, V6.1.7601
Installs	Each: JBoss Server 6.1.0 EAP with JDK 1.7.0_45
NETWORK (for testing management)	Each: 1Gb
NETWORK (for testing itself)	Each: 1Gb (Broadcom BCM57810 NetXtreme II 10 GigE)

This does not imply that the hardware shown above is the right hardware configuration for your implementation. There are many factors that go into determining the optimal hardware for each MDM implementation, including number of source records, volume of delta loads, data model, the rules you implement, hardware, network and more. Your Informatica account team will help you size hardware for your specific implementation of Multidomain MDM.

Software configuration

Schema definition including schema, business objects, cleanse mappings, match rules, validation rules and trust columns are described in detail in *Appendix B – MDM software configuration for performance benchmark testing*.

The application server used for the testing is JBoss. The application server and the client configuration information are listed below.

JBOSS configuration for performance tests

JBoss Application Server Configuration

Data source configuration: Initial connection Pool capacity: 300, Maximum Capacity: 300
Heap size: Minimum: -Xms49152m; Maximum: -Xmx49152m
Garbage collection Settings: -XX:MaxPermSize=512M -Dsun.rmi.dgc.server.gcInterval=3600000
Bind server address to local not clustered: -Djgroups.bind_addr=localhost
Use of JDBC logging: -Djava.util.logging.manager=java.util.logging.LogManager -Djava.util.logging.config.file=C:\PTF\DS_SHARE\workdir\logging.properties
Prefer IPv4: -Djava.net.preferIPv4Stack=true
Byteman visible in all module loader: -Djboss.modules.system.pkgs=org.jboss.byteman
Debug level for Java classes is INFO
Default parameter files from MDM installation are used

DB Server Configuration

Memory configuration total for database: 56 GB
Default DB parameters are used

Batch performance testing results, with Oracle as the DBMS

Initial data load test

Interoperability is OFF (API_BATCH_INTEROPT_IND=0)

Process	Records	Time to process (Minutes)
Cleanse	82.5M	24.78
Load (inserts)	75.0M	63.51
Tokenize	15.0M	41.73
Match	4.9M	26.14
Automerge	4.9M	39.74
Totals		**195.89**

100 source systems test

Interoperability is ON (API_BATCH_INTEROPT_IND=1)

Process	Records	Time to process (Minutes)
Cleanse	249.8K	8.77
Load (inserts, updates)	249.8K	4.21
Tokenize	12.4K	0.11
Match	12.4K	1.03
Automerge	12.4K	0.64
Totals		**14.67**

Services Integration Framework (SIF) API performance test

SIF API performance test approach

1. To achieve the main goal of getting the best performance result for each SIF API measured by Transactions Per Second (TPS), we needed to put enough SIF API load based on the number of threads available on the driving machine. We used 50 threads of API to each app server to create a balanced workload. We also monitored to make sure that the workload did not overload the servers and that resource consumption remained 80% and under.
2. On each SIF API run, the ramp up time was 400 seconds to reach 100 threads and keep the constant peak load for 10 minutes. Each thread was continuously sending requests and getting responses without waiting.
3. We collected TPS results for each of SIF API run and used the following equation to check if the results were correct (**Number of threads = Transaction per Second (TPS) X Average Transaction Response Time in Second**). The response time in the result is not the optimized one, but just the reference one when the system is reaching the optimized TPS value.

Services Integration Framework performance testing results, with Oracle as the DBMS

API Name	TPS (tx# /sec)	Latency (ms)	Threads (#)
GET	7009	10	70
SEARCHQUERY	7460	9	70
CLEANSE	8714	8	70
PUT	1920	36	70
CLEANSEPUT	1810	38	70
TOKENIZE	3253	21	70
MERGE	762	91	70
MULTIMERGE	251	273	70
UNMERGE	555	125	70

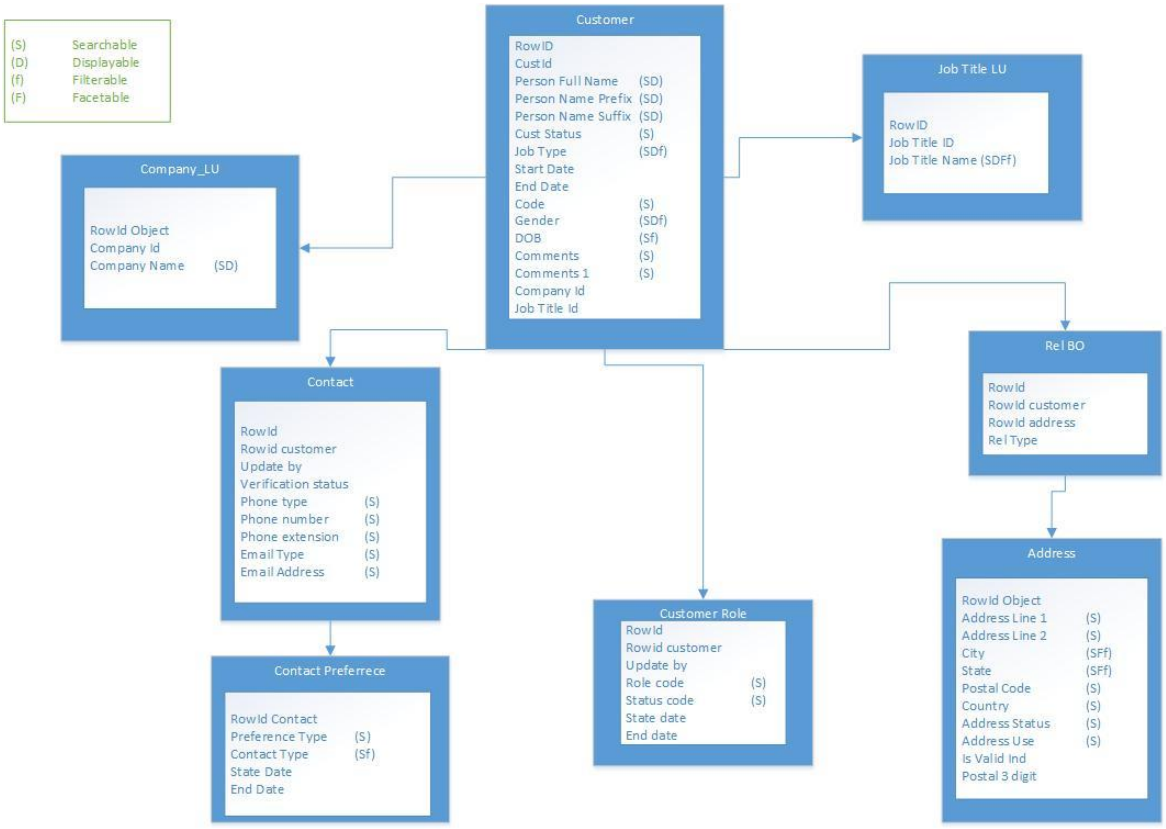
Business Entity Services performance testing

Business Entity Services performance testing was done as a comparison against Services Integration Framework in the same release. The tests emulate Business Entity Services API calls in a heavy load environment. The tests were run in 2 different scenarios:

- Business Entity Services API calls as a set of tests, each set up to 200,000 single calls or up to 800 seconds duration.
- SIF API calls issued to emulate functionally equivalent Business Entity Services API calls.

Business Entity configuration

The Customer base object is used as the root level for business entity. The following diagram shows all the base objects involved in the business entity:



Description of the Business Entity Services performance test:

API Name	Business Entity Services API description	SIF API Description (set of tests compatible to BES call)
Write Insert customer with 20 children	Insert customer business entity with all its children: 1. One customer record 2. 5 records of contacts 3. 5 records of contact preferences 4. 5 records of customer roles 5. 5 records of addresses	Issue the set of SIF API calls equal to 1 write Insert business entity API call: 1. 1 PUT insert for customer 2. 5 PUT insert for contacts 3. 5 PUT insert for of contact preferences 4. 5 PUT insert for customer roles 5. 5 PUT insert for addresses 6. 5 PUT inserts for REL_BO
Write Update customer with 20 children	Update Customer business entity with all its children: 1. One customer record 2. 5 records of contacts 3. 5 records of contact preferences 4. 5 records of customer roles 5. 5 records of addresses	Issue the set of SIF API calls equal to 1 Write Update business entity API call: 1. 1 PUT update for customer 2. 5 PUT update for contacts 3. 5 PUT update for of contact preferences 4. 5 PUT update for customer roles 5. 5 PUT update for addresses
Write Update customer	Update customer CO without its children	One single PUT update for customer
Read customer by rowid_object value with 20 children	Read Customer business entity with its children 21 records of base objects for each business entity API call	Issue the set of SIF API calls equal to 1 Read by rowid_object business entity API call: 1. 1 GET for customer using rowid_object value 2. SEARCH for job title 3. SEARCH for company 4. SEARCH for contacts 5. SEARCH for of contact preferences 6. SEARCH for customer roles 7. SEARCH for REL_BO 8. GET for addresses

API Name	Business Entity Services API description	SIF API Description (set of tests compatible to BES call)
Read customer by pkey_src_object value with 20 children	Read customer business entity with its children using pkey_src_object value plus source system name 21 record of base objects for each BES API call	Issue the set of SIF API calls equal to 1 Read by pkey_src_object plus source system name BE API call: 1. 1 GET for customer using pkey_src_object value plus system name 2. SEARCH for job title 3. SEARCH for company 4. SEARCH for contacts 5. SEARCH for contact preferences 6. SEARCH for customer roles 7. SEARCH for REL_BO 8. GET for addresses

Detailed description for APIs (excluding "Write Update customer with 20 children" API call).

All APIs listed below are done and measured as a single atomic call:Write Insert customer with 20 children:

1. Write Insert customer
2. Write Update customer
3. Read customer by rowid_object value with 20 children
4. Read customer by pkey_src_object value with 20 children

Business Entity Services (BES) performance testing results, with Oracle as the DBMS[5]

	SIF API	BES API	BES improvement over SIF API (%)	SIF API	BES API	BES improvement over SIF API (%)	Threads SIF API (#)	Threads BES API(#)
API Name	**TPS (tx#/sec)**			**Latency (ms)**				
Write Insert customer with 20 children	95.1	107.8	13.3	468	413	13.3	45	45
Write Insert customer	99	149	51.03	450	298	51.14	45	45
Write Update customer with 20 children	5.38	20.6	282	7361	1924	283	40	40
Read customer by rowid_object value with 20 children	15.3	40.6	166	2597	976	166	40	40
Read customer by pkey_src_object value with 20 children	15.1	40.3	167.8	2637	981	168.8	40	40

[5] Description of formulas used for calculations of "BES improvement over SIF API":

1. BES TPS improvement = (TPS BES - TPS SIF)/TPS SIF
2. BES Latency improvement = (Latency SIF – Latency BES) / Latency BES

Appendix B – MDM software configuration for performance benchmark testing

Testing Schema Definition

This section describes the schema configuration information that was used for testing.

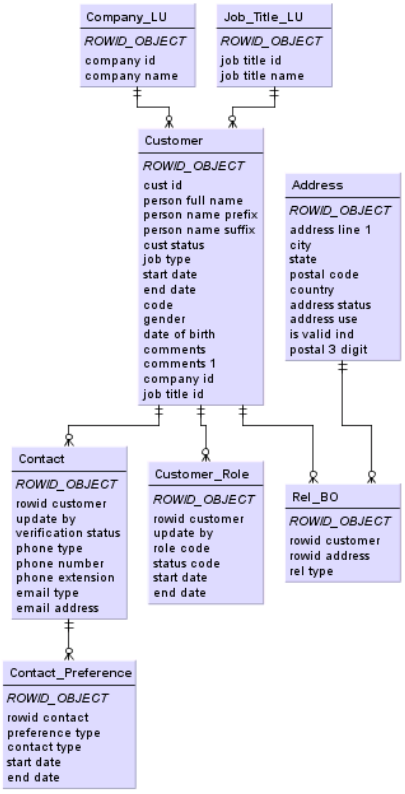
Schema

The schema used for testing has 354M source records in total.

Base Objects

The Batch performance test was run against a 50 million customer schema, which consists of 8 base objects[6]. Interoperability was OFF for Onboarding:

- CUSTOMER (Main base object – 50 million records). History disabled.
- ADDRESS (Parent base object - about 69.5 million records). History disabled.
- REL_base object (Relationship base object – about 99.5 million records). History disabled.
- COMPANY_LU (lookup base object- about 2.5 million records). History disabled.
- JOB_TITLE_LU ((lookup base object - about 5 thousand records). History disabled.
- CONTACT (child base object - 50 million records). History disabled.
- CONTACT_PREFERENCE (child base object - about 17 million records). History disabled.
- CUSTOMER_ROLE (child base object - about 65.5 million records). History disabled.



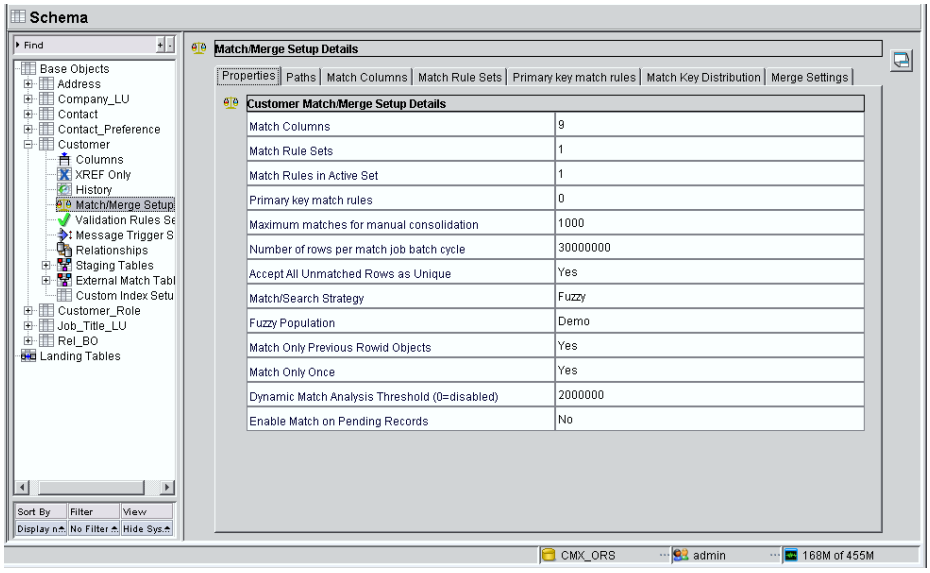
Match Rules

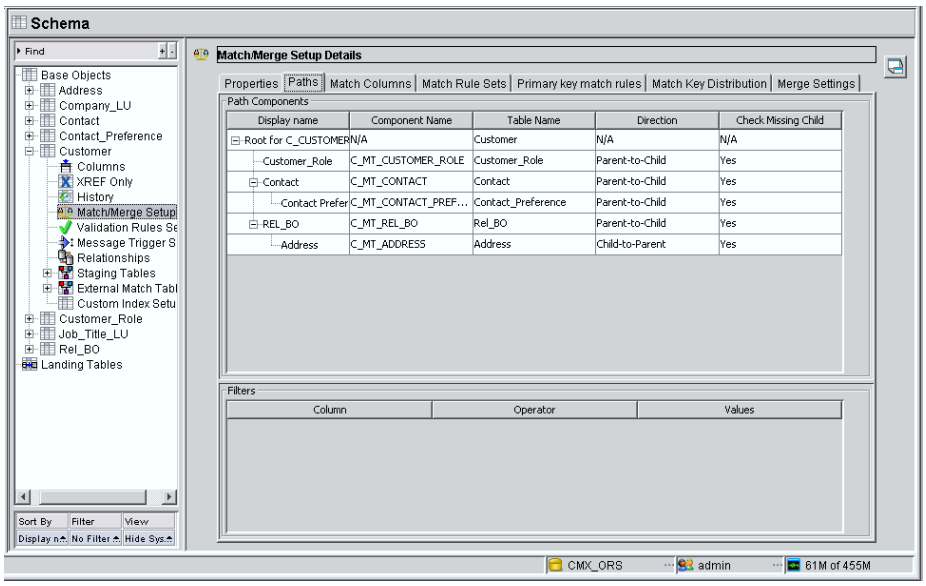
The Customer base object is configured for match rules. This is a fuzzy match base object which contains 9 match columns. The match rule contains the following match columns:

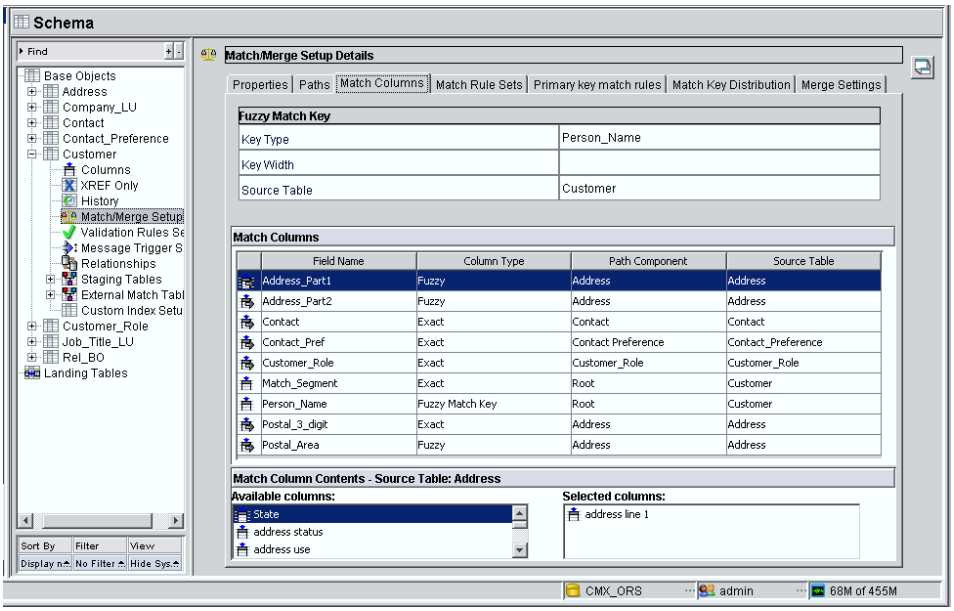
- Person_Name. Fuzzy Match Key. Based on C_CUSTOMER.PERSON_FULL_NAME.
- Address_Part1. Fuzzy Match Rule. Based on C_ADDRESS.ADDRESS_LINE_1.
- Address_Part2. Fuzzy Match Rule. Based on C_ADDRESS.CITY, STATE and POSTAL_CODE.
- Postal_Area. Fuzzy Match Rule. Based on C_ADDRESS.POSTAL_CODE.
- Postal_3_digit. Exact Match Rule. Based on C_ADDRESSS.POSTAL_3_DIGIT.

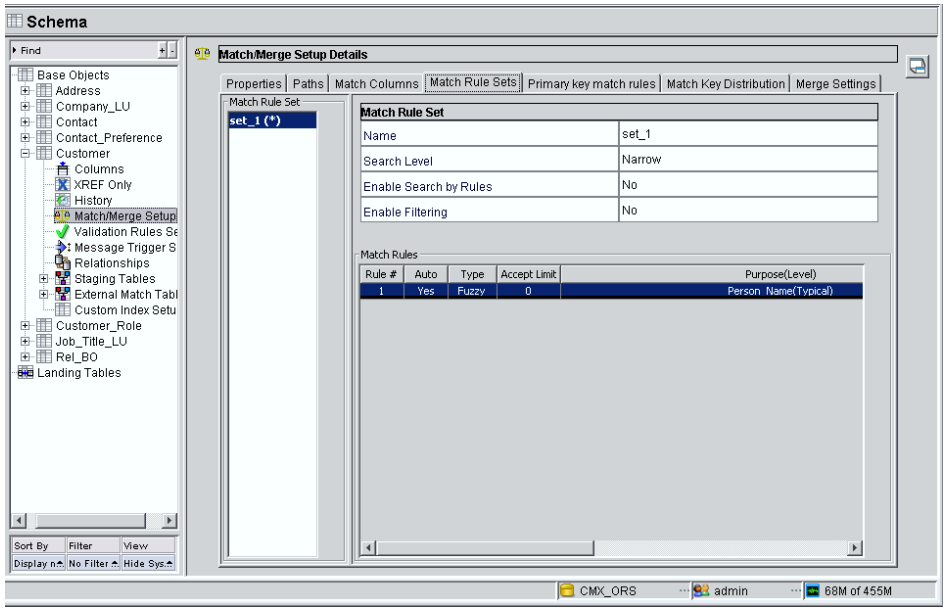
[6] For details of what a base object is, please see https://kb.informatica.com/proddocs/Product%20Documentation/5/MDM_102_ConfigurationGuide_en.pdf

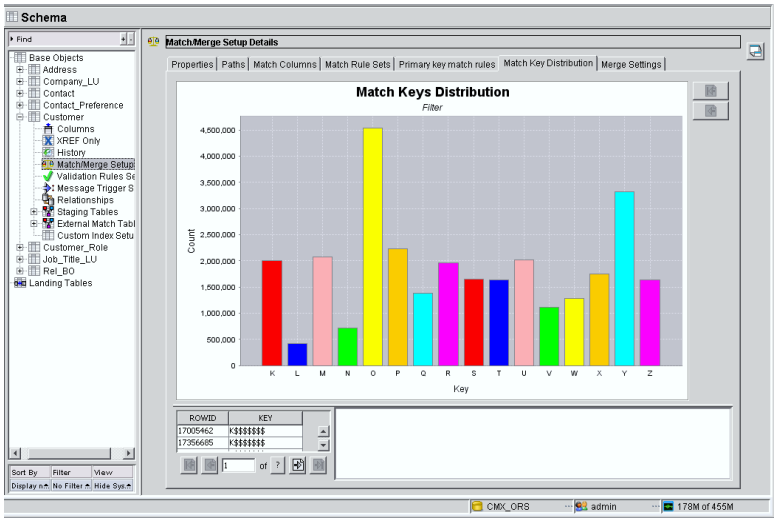
Additional Settings: Key Width=Limited; Search Level=Narrow; Match Purpose=Person_Name






Schema
Find
Base Objects
Address
Company_LU
Contact
Contact_Preference
Customer
Columns
XREF Only
History
Match/Merge Setup
Validation Rules Se
Message Trigger S
Relationships
Staging Tables
External Match Tabl
Custom Index Setu
Customer_Role
Job_Title_LU
Rel_BO
Landing Tables
Sort By
Filter
View
Display n
No Filter
Hide Sys
Match/Merge Setup Details
Properties
Paths
Match Columns
Match Rule Sets
Primary key match rules
Match Key Distribution
Merge Settings
Fuzzy Match Key
Key Type
Person_Name
Key Width
Source Table
Customer
Match Columns
Field Name
Column Type
Path Component
Source Table
Address_Part1
Fuzzy
Address
Address_Part2
Contact
Exact
Contact
Contact_Pref
Contact Preference
Contact_Preference
Customer_Role
Match_Segment
Root
Person_Name
Fuzzy Match Key
Postal_3_digit
Postal_Area
Match Column Contents - Source Table: Address
Available columns:
State
address status
address use
Selected columns:
address line 1
CMX_ORS
admin
68M of 455M

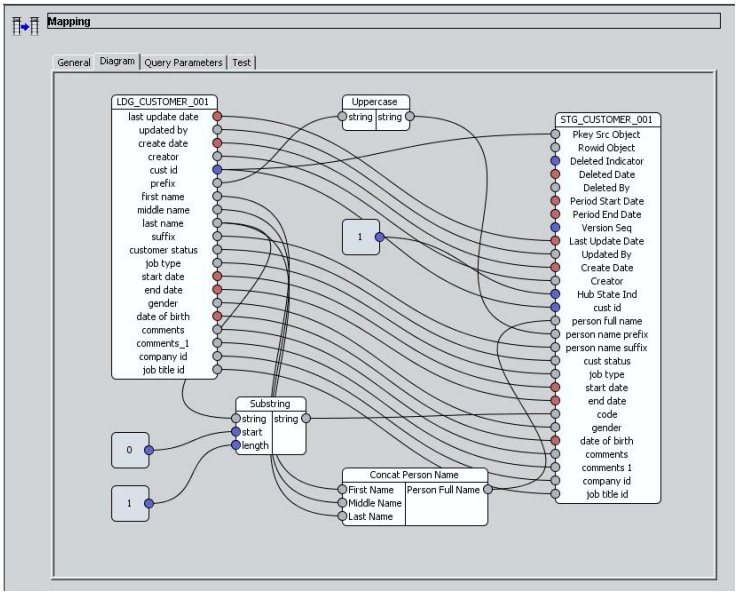

Schema
Find
Base Objects
Address
Company_LU
Contact
Contact_Preference
Customer
Columns
XREF Only
History
Match/Merge Setup
Validation Rules Se
Message Trigger S
Relationships
Staging Tables
External Match Tabl
Custom Index Setu
Customer_Role
Job_Title_LU
Rel_BO
Landing Tables
Sort By
Filter
View
Display n
No Filter
Hide Sys
Match/Merge Setup Details
Properties
Paths
Match Columns
Match Rule Sets
Primary key match rules
Match Key Distribution
Merge Settings
Match Rule Set
set_1 (*)
Match Rule Set
Name
set_1
Search Level
Narrow
Enable Search by Rules
No
Enable Filtering
No
Match Rules
Rule #
Auto
Type
Accept Limit
Purpose(Level)
1
Yes
Fuzzy
0
Person_Name(Typical)
CMX_ORS
admin
68M of 455M



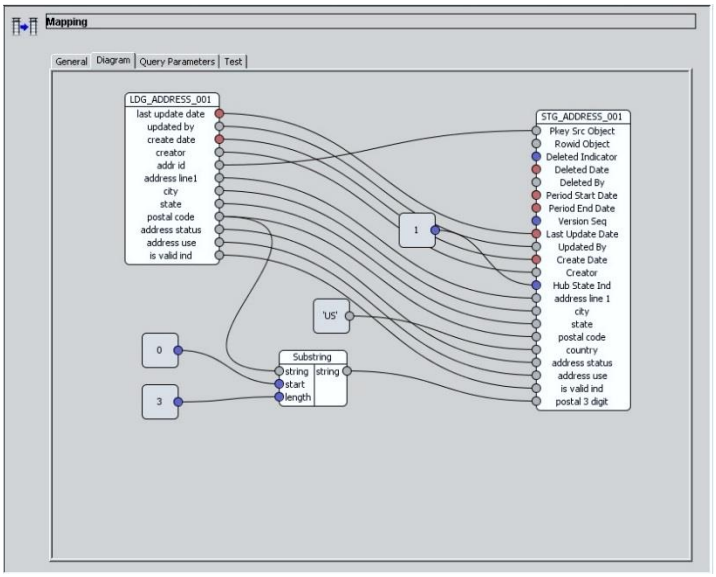
Cleanse Mapping

The following are the mapping details for each of the base objects used.

- C_JOB_TITLE_LU
    - Direct mapping without Functions
- C_COMPANY_LU
    - Direct mapping without Functions
- C_CUSTOMER
    - String Function: Concatenate – Concatenate 3 columns into another column
    - String Function: Substring – Take the first character of the last name and insert it into another column
    - String Function: Uppercase – Change the prefix to upper case

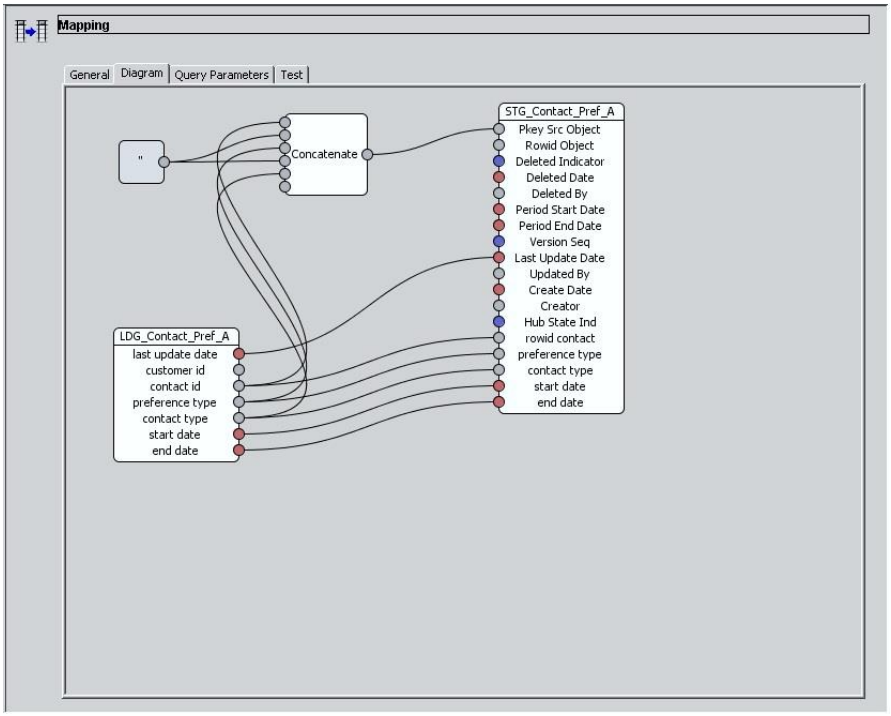


- C_ADDRESS
    - String Function: Substring – Take the first 3 characters of the postal code and insert it into another column

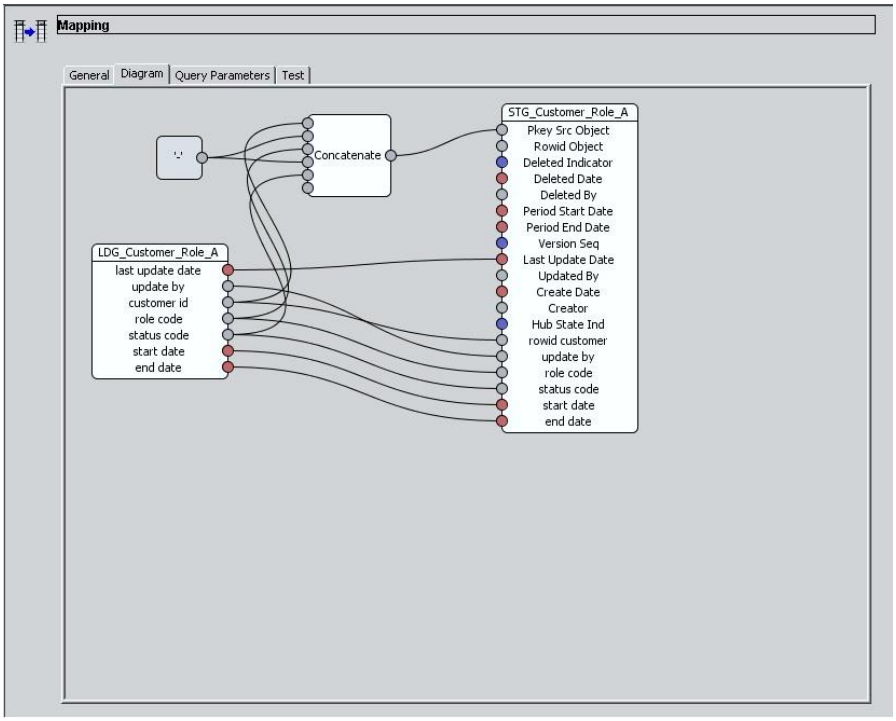


- C_REL
    - Direct mapping without Functions
- C_CONTACT
    - Direct mapping without Functions

- C_CONTACT_PREF
    - Concatenate – Concatenate 3 columns into another column



- C_CUSTOMER_ROLE
    - Concatenate – Concatenate 3 columns into another column



Validation Rules

The following are the validation rules for C_CUSTOMER and C_ADDRESS base objects.

- C_CUSTOMER has 15 validation rules:
    - downgrade cust_id if cust_id is null, downgrade percentage 100%.
    - downgrade full name if full name is null, downgrade percentage 100%.
    - downgrade name suffix if name suffix is null, downgrade percentage 10%.
    - downgrade name prefix if name prefix is null, downgrade percentage 20%.
    - downgrade cust status if cust status is null, downgrade percentage 50%.
    - downgrade job type if job type is null, downgrade percentage 60%.
    - downgrade start date if start date is null, downgrade percentage 60%.
    - downgrade end date if end date is null, downgrade percentage 20%.
    - downgrade code if code is null, downgrade percentage 50%.
    - downgrade gender if gender is null, downgrade percentage 89%.
    - downgrade DOB if DOB is null, downgrade percentage 100%.
    - downgrade comments if comments is null, downgrade percentage 10%.
    - downgrade comments1 if comments1 is null, downgrade percentage 1%.
    - downgrade company id if company id is null, downgrade percentage 90%.
    - downgrade job title id if job title id is null, downgrade percentage 1%.

- C_ADDRESS has 3 validation rules:
    - downgrade Addressline 1 if addressline 1 is null, downgrade percentage 100%.
    - downgrade postal code if postal code is null, downgrade percentage 100%.
    - downgrade is valid ind if is valid ind is null, downgrade percentage 100%.

Trust Columns

The following are the trusted columns for C_CUSTOMER and C_ADDRESS base objects.

- C_CUSTOMER has 15 trust columns including CUST_ID, PERSON_FULL_NAME, PERSON_NAME_PREFIX, PERSON_NAME_SUFFIX, CUST_STATUS, JOB_TYPE, START_DATE, END_DATE, CODE, GENDER, DATE_OF_BIRTH, COMMENTS, COMMENTS_1, COMPANY_ID, and JOB_TITLE_ID.
- C_ADDRESS has 2 trust columns including ADDRESS_LINE_1, POSTAL_CODE