

Performance Metrics on Creating Data via Custom OData API in SAP

Srihariram Chendamarai Kannan

SAP ABAP Lead Analyst, Edison, New Jersey, USA

Abstract This article provides performance testing metrics for a suggested solution that attempts to create data in SAP using a registered OData API technique. Experimented with the SAP Gateway Client, specifically transaction code /IWFND/GW_CLIENT, to assess the performance implications of bulk generation scenarios utilizing the SAP OData API. This study provides valuable insights into the numerous elements to consider while using the SAP OData API for bulk generation situations. The testing results showed that the suggested solution met the performance standards and demonstrated its ability to properly manage large amounts of data.

Keywords SAP OData, SAP API, OData V4, SAP Restful-ABAP Programming Framework

1. Introduction

Organizations rely significantly on effective and high-performing data integration technology in today's data-driven business environment to guarantee seamless operations and the best possible user experiences. OData (Open Data Protocol) has become a favoured protocol for constructing RESTful APIs and enabling data interchange. OData provides a clear and standardized methodology for developing and using queryable and interoperable APIs. Providing a machine-readable definition of API data models, its metadata facilitates the creation of flexible client applications and tools. Many SAP applications and services employ OData APIs, which have emerged as the favoured interface for exposing data held in SAP systems, hence facilitating external access. OData has a pivotal "connecting point" that streamlines interoperability between SAP and non-SAP applications. Any application or programming language that can communicate via HTTP can consume OData services because they adhere to RESTful principles. For instance, you can create a Python, Go, or Rust application or service that interacts directly with an OData service, including integration with mobile applications.

The main focuses are on exploring the performance capabilities of OData and sheds light on its potential to deliver fast, scalable, and responsive data access within SAP systems.

In SAP, data is well organized and saved within a database. Multiple layers of security are put in place to safeguard its

data. To interface with this data, specific tools in the form of modules, functions, or classes are created to allow exact interactions with the data model, whether for reading or writing. These instruments are commonly acknowledged as APIs (Application Programming Interfaces). Contemporary applications and ERPs typically provide APIs that facilitate connection with their internal data architectures. Similarly, SAP Gateway is critical in facilitating the creation and implementation of RESTful APIs for SAP systems. It functions as a primary nexus for the integration of diverse SAP components, business processes, and external systems. Using SAP Gateway, enterprises may develop seamless communication channels and provide SAP data and functionality via defined protocols such as OData.

SAP OData denotes SAP's execution of the OData protocol within its software framework. It provides an extensive array of tools, frameworks, and services that enable developers to create OData-compliant APIs and utilize the functionalities of SAP systems.

OData V4 is the most recent iteration of the OData protocol, incorporating numerous improvements and innovations compared to earlier versions. It offers augmented support for intricate data models, broadened querying functionalities, and fortified security measures.

2. Motivation and Goal

Performance plays a critical role in determining the effectiveness of any data integration solution. With the rise in data volumes and the increasing complexity of business processes, organizations are seeking ways to optimize the speed, throughput, and efficiency of their data exchange mechanisms. OData, with its standardized approach and

* Corresponding author:

srihariram@hotmail.com (Srihariram Chendamarai Kannan)

Received: Aug. 19, 2023; Accepted: Nov. 22, 2023; Published: Nov. 29, 2023

Published online at <http://journal.sapub.org/xxx>

extensive tooling support, presents an opportunity to address these performance challenges and ensure smooth data access and transfer between SAP systems and external applications.

The motivation of the study stems into the various factors that influence the performance of OData and examine best practices for optimizing its capabilities. It will explore techniques for efficient data creation and other performance-enhancing measures specific to OData implementations. By understanding these performance capabilities and techniques, organizations can leverage OData to maximize the speed and responsiveness of their data integration processes, ultimately improving overall system performance and user satisfaction.

3. Approach

The focus of this study is to explore the slim and lightweight protocol provided by the OData and SAP gateway for efficient data consumption. While create operations are typically performed for single entities, there are valid scenarios where the creation of multiple entities simultaneously becomes necessary. To evaluate the performance and scalability of producing data in SAP through custom OData APIs, a series of experiments were conducted using SAP S4 HANA (2021 Version). The OData V4 protocol was implemented in SAP HANA through various approaches. Two commonly used methods for creating OData V4 were explored:

1. Using the RAP (Restful-ABAP Programming and CDS) Framework: The RAP Framework offers both managed and unmanaged scenarios for creating OData V4. The managed scenario provides a structured and standardized approach for generating OData V4 entities, ensuring consistency and ease of maintenance. On the other hand, the unmanaged scenario allows more flexibility and customization but may require additional effort for development and maintenance.
2. Code-Based Implementation for OData V4: This approach involves writing custom code to implement

the OData V4 protocol. It provides greater flexibility and control over the implementation, allowing for fine-tuning and customization according to specific requirements. However, it may also involve more complex development and maintenance processes compared to using the RAP Framework.

Since most of the requirements in SAP need custom development, so will consider unmanaged RAP V4 API and Code based V4 API. By creating these objects in SAP S4 HANA (2021 Version) and implementing OData V4 through different approaches, the performance and scalability of staging data in SAP can be thoroughly tested. This research aims to provide insights into the effectiveness and efficiency of custom OData APIs for managing and manipulating data in SAP environments.

4. Technical Details

4.1. Prerequisite

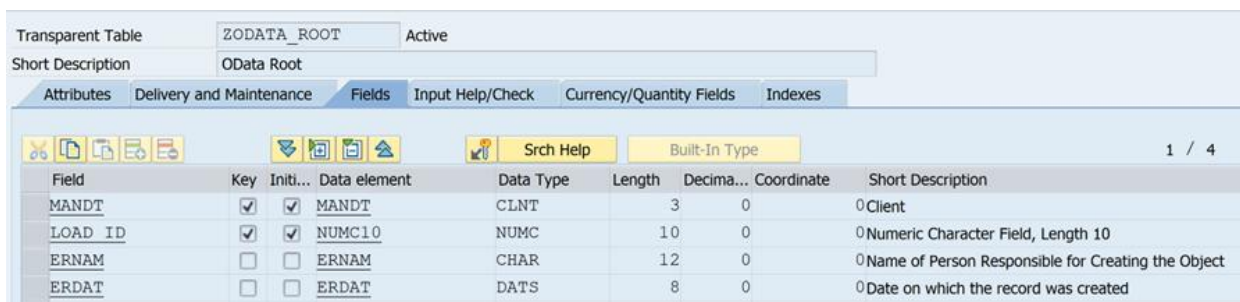
- ABAP Programming: You should have a good understanding of SAP ABAP programming, including class and object-oriented programming (OOP) concepts. Additionally, you should be familiar with OData development and gateway configurations in SAP.
- CDS Development: You should have a fundamental understanding of SAP Core Data Service SQL queries and annotations.
- S4HANA: You should be familiar with SAP S4HANA newer versions and Restful-ABAP programming concepts.

By mastering these prerequisites, you can start creating your own OData objects in SAP.

4.2. Technical Design

In this document, it will be examined which approach is better for creating or loading data using OData V4 service.

- Step 1: Create database tables and CDS views to load data.



The screenshot shows the SAP 'Transparent Table' ZODATA_ROOT. The 'Fields' tab is active, displaying a list of fields with their properties. The fields are MANDT, LOAD_ID, ERNAM, and ERDAT. Each field has a 'Key' checkbox, an 'Initial' checkbox, a 'Data element', a 'Data Type', 'Length', 'Decimal places', 'Coordinate', and a 'Short Description'.

Field	Key	Initial	Data element	Data Type	Length	Decima...	Coordinate	Short Description
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		Client
LOAD_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NUMC10	NUMC	10	0		Numeric Character Field, Length 10
ERNAM	<input type="checkbox"/>	<input type="checkbox"/>	ERNAM	CHAR	12	0		Name of Person Responsible for Creating the Object
ERDAT	<input type="checkbox"/>	<input type="checkbox"/>	ERDAT	DATS	8	0		Date on which the record was created

Figure 1. Header Table

Transparent Table

ZODATA_LOAD

Active

Short Description

OData Load

Attributes

Delivery and Maintenance

Fields

Input Help/Check

Currency/Quantity Fields

Indexes

Srch Help

Built-In Type

1 / 13

Field	Key	Initi...	Data element	Data Type	Length	Decima...	Coordinate	Short Description
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0		0Client
LOAD ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NUMC10	NUMC	10	0		0Numeric Character Field, Length 10
UNIQUE ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NUMC10	NUMC	10	0		0Numeric Character Field, Length 10
NAME LAST	<input type="checkbox"/>	<input type="checkbox"/>	BP NAMLAST	CHAR	40	0		0Last Name
NAME FIRST	<input type="checkbox"/>	<input type="checkbox"/>	BP NAMFRST	CHAR	40	0		0First Name
NATION	<input type="checkbox"/>	<input type="checkbox"/>	BP CNTR N	CHAR	3	0		0Nationality
STATE	<input type="checkbox"/>	<input type="checkbox"/>	BP CNTR ST	CHAR	3	0		0Citizenship
BIRTH DATE	<input type="checkbox"/>	<input type="checkbox"/>	BP BIRTHD	DATS	8	0		0Date of Birth of Business Partner
BIRTH PLAC	<input type="checkbox"/>	<input type="checkbox"/>	BP BIRTHPL	CHAR	40	0		0Birthplace
MARITAL ST	<input type="checkbox"/>	<input type="checkbox"/>	BP MARITAL OLD	NUMC	2	0		0Marital Status
CHILDREN	<input type="checkbox"/>	<input type="checkbox"/>	BP CHILDNRN	DEC	2	0		0Number of Children Belonging to the Business Partner
ERNAM	<input type="checkbox"/>	<input type="checkbox"/>	ERNAM	CHAR	12	0		0Name of Person Responsible for Creating the Object
ERDAT	<input type="checkbox"/>	<input type="checkbox"/>	ERDAT	DATS	8	0		0Date on which the record was created

Figure 2. Item Table

```
1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 @EndUserText.label: 'OData Root Unmanaged'
3 define root view entity ZCDS_I_UMROOT
4     as select from zodata_root
5     composition [0..*] of ZCDS_I_UMLOAD as _Item
6 {
7
8     key load_id,
9         ernam,
10        erdat,
11
12        _Item
13 }
```

Figure 3. CDS View for Header

```

1 @AbapCatalog.viewEnhancementCategory: [#NONE]
2 @AccessControl.authorizationCheck: #NOT_REQUIRED
3 @EndUserText.label: 'OData Load data Un Managed'
4 @Metadata.ignorePropagatedAnnotations: true
5 @ObjectModel.usageType:{
6     serviceQuality: #X,
7     sizeCategory: #S,
8     dataClass: #MIXED
9 }
10 define view entity ZCDS_I_UMLOAD
11     as select from zodata_load
12     association to parent ZCDS_I_UMROOT as _Header on $projection.load_id = _Header.load_id
13 {
14     key load_id,
15     key unique_id,
16     name_last,
17     name_first,
18     nation,
19     state,
20     birth_date,
21     birth_plac,
22     marital_st,
23     children,
24     ernam,
25     erdat,
26
27     _Header
28 }

```

Figure 4. CDS View for the Item

- Step 2: Create a Restful-ABAP Unmanaged V4 OData API.
 - a. Create Unmanaged Restful-ABAP Objects

```

1  unmanaged implementation in class zbp_cds_i_umroot unique;
2  strict;
3
4  define behavior for ZCDS_I_UMROOT //alias <alias_name>
5  //late numbering
6  lock master
7  authorization master ( instance )
8  //etag master <field_name>
9  {
10   create;
11   update;
12   delete;
13   association _Item { create; }
14 }
15
16 define behavior for ZCDS_I_UMLOAD //alias <alias_name>
17 //late numbering
18 lock dependent by _Header
19 authorization dependent by _Header
20 //etag master <field_name>
21 {
22   update;
23   delete;
24   field ( readonly ) load_id;
25   association _Header;
26 }

```

Figure 5. Unmanaged behavior definition

```

[ED5] ZSD_ODATA_UMLOAD X
1  @EndUserText.label: 'OData UnManaged Load'
2  define service ZSD_ODATA_UMLOAD {
3    expose ZCDS_C_UMROOT;
4    expose ZCDS_I_UMROOT;
5    expose ZCDS_I_UMLOAD;
6  }

```

Figure 6. Unmanaged Service Definition

Service Binding:

General Information
This section describes general information about this service binding

Binding Type: OData V4 - Web API

Service	Version	API S...	Service Definition
▼ ZSD_OD	1.0.0	Not ...	ZSD_ODATA_UMLOAD

Service Version Details
View information on selected service version

Service Information
Service URL: /sap/opu/odata4/sap/zsb_odata_umload_v4/srvd_a2x/sap/zsd_odate

Entity Set and Association

- ▼ ZCDS_C_UMROOT
 - _Item
- ▼ ZCDS_I_UMLOAD
 - _Header
- ▼ ZCDS_I_UMROOT
 - _Item

Figure 7. Unmanaged Service Binding

- Step 3: Create a Code based V4 OData classes using SAP provide interfaces /IWBEP/IF_V4_MP_BASIC and /IWBEP/IF_V4_DP_ADVANCED.

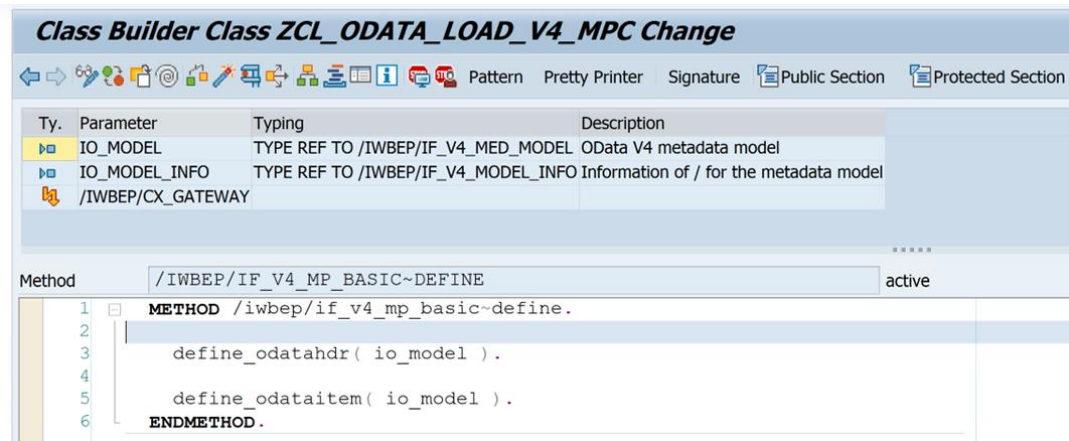


Figure 8. Model provider class for V4 OData

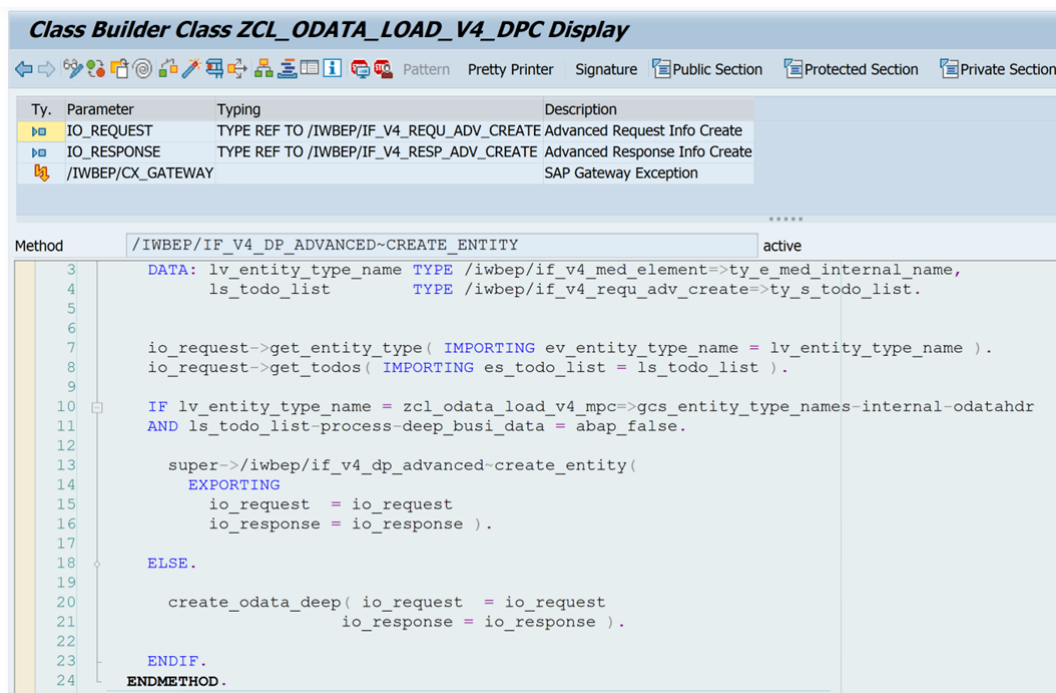


Figure 9. Data provider class for V4 OData

5. Testing and Performance Metrics

To evaluate the impact of different approaches, we conducted tests with and without utilizing changesets in the OData and SAP Gateway frameworks and tried to examine the performance and efficiency of alternative methods. To gauge the performance of the create process, the SAP Gateway framework provides comprehensive performance statistics in the HTTP response header. These statistics are structured in the following format [3]:

- HTTP header name: sap-statistics
- HTTP header value as below,
 - a. total: Total processing time.
 - b. fw: Framework.
 - c. app: Application.
 - d. gwttotal: Total processing time of the OData request.

- e. gwhub: Processing time in SAP Gateway hub system.
- f. gwrfoh: RFC and network overhead for communication between the hub and backend system.
- g. gwbe: Processing time in the SAP Gateway framework in the backend system (without application time).
- h. gwapp: Processing time in the application (data provider).
- i. gwnongw: Processing time of applications called (referred to as non-SAP Gateway since this processing time is not related to the SAP Gateway framework).

5.1. Testing RAP Unmanaged V4 OData API

I tried to create data in SAP using Restful ABAP programming and an unmanaged OData API with multiple sets of records.

Sample Payload

The below given JSON payload represents data for SAP OData API request. It includes information about a header and associated item details. Multiple items can be added in the “_item” section. Used similar payloads for both RAP OData and code based V4 OData APIs.

```
{
  "load_id" : "0001",
  "ernam" : "Test",
  "erdat" : "20230101",
  "_Item": [
    {
      "load_id" : "0001",
      "unique_id" : "0000000001",
      "name_last" : "Configure",
      "name_first" : "Test",
      "nation" : "USA",
      "state" : "NJ",
      "birth_date" : "19900101",
      "birth_plac" : "EDISON",
      "marital_st" : 02,
      "children" : 0,
      "ernam" : "Test",
      "erdat" : "20230101"
    }
  ]
}
```

Average Times (in milliseconds)						
No. of Reque...	Processing TL...	SAP GW Hub	RFC and Netw...	SAP GW Back	Application	Non-GW
1	3,109	243	85	125	2,654	2

Detail (Time in milliseconds)						
Ex.	Namespa	Location	Operation Name	Level	Processing TL	SAP GW Hub
	/SAP/	Hub System	BATCH	1	3,109	243
	/SAP/	Hub System	PROCESS_BATCH	2	0	0
	/SAP/	Backend	PROCESS_CHANGESET	3	0	0
	/SAP/	Backend	CHANGESET_BEGIN	4	0	0
	/SAP/	Backend	CHANGESET_PROCESS	4	0	0
	/SAP/	Backend	CHANGESET_END	4	0	0
	/SAP/	Backend	PROCESS_CHANGESET	3	0	0
	/SAP/	Backend	CHANGESET_BEGIN	4	0	0
	/SAP/	Backend	CHANGESET_PROCESS	4	0	0
	/SAP/	Backend	CHANGESET_END	4	0	0
	/SAP/	Backend	PROCESS_CHANGESET	3	0	0
	/SAP/	Backend	CHANGESET_BEGIN	4	0	0
	/SAP/	Backend	CHANGESET_PROCESS	4	0	0
	/SAP/	Backend	CHANGESET_END	4	0	0
	/SAP/	Backend	PROCESS_CHANGESET	3	0	0
	/SAP/	Backend	CHANGESET_BEGIN	4	0	0
	/SAP/	Backend	CHANGESET_PROCESS	4	0	0
	/SAP/	Backend	CHANGESET_END	4	0	0

Figure 10. Evidence, Mutli Changeset approach calls multiple time to commit

All operations within a change set must be treated as a logical unit of work. This means all or nothing. Therefore, a provider must not issue COMMIT WORK or ROLLBACK WORK during change set processing. Otherwise, the framework will abandon the change set processing. If the

change set contains only one operation, the check for commit or rollback is deactivated [2]. The Multi Changeset option for the create process will be considered the least favorable due to the repeated invocation of the CREATE_ENTITY method in the DPC Extension class and subsequent multiple COMMIT WORK operations. The Multi Changeset approach can result in performance degradation and increased costs. However, to optimize data staging, it is beneficial to commit multiple records simultaneously. Using the payload, The OData API was executed from the transaction code /IWFND/GW_CLIENT and measurements were taken using transaction /IWFND/TRACES.

Table 1. Performance metric (msec) for RAP Unmanaged Multi changeset

Records	gwttotal	gwhub	gwbe	gwapp	Gwrfcoh
1	139	20	18	46	52
10	368	111	19	159	76
100	2503	1010	39	1406	44
1000	25061	10442	197	14062	359
5,000	143568	50248	1824	90862	634
10,000	317927	106322	5787	204565	1225

Prepared the similar payloads for the single changeset and executed in transaction code /IWFND/GW_CLIENT, and the processing time was reduced by approximately 50%.

Table 2. Performance metric (msec) for RAP Unmanaged single changeset

Records	gwttotal	gwhub	gwbe	gwapp	Gwrfcoh
1	212	31	24	71	86
10	349	112	29	95	92
100	1873	975	31	151	713
1000	11432	10024	203	1035	157
5,000	59567	51967	1823	5127	638
10,000	127300	110473	5753	10052	1058

Although the change set approach offers advantages in terms of committing data, it is not without its limitations. One significant drawback is its potential slowness, as each changeset requires its own set of processing steps. This can create a bottleneck in the overall data staging process, impacting performance.

Another approach is testing the payload with single create requests and no changeset in the RAP unmanaged V4 OData API.

Table 3. Performance metric (msec) for RAP Unmanaged V4 OData API

Records	gwttotal	gwhub	gwbe	gwapp	gwrfoch
1	230	15	21	98	96
10	161	19	11	71	60
100	322	48	33	201	40
1000	1458	349	13	1062	34
5,000	7336	1695	44	5465	132
10,000	14129	3359	48	10515	207

5.2. Testing Code Based V4 OData API

Executed the payload of data with deep create request on V4 OData code based implemented API too. And it was much faster than the RAP Framework.

Table 4. Performance metric (msec) for code based V4 OData API

Records	gwttotal	gwhub	gwbe	gwapp	gwrfoh
1	218	35	59	39	85
10	150	36	17	26	71
100	119	52	5	27	35
1000	495	266	28	135	66
5,000	1893	1232	28	552	81
10,000	3786	2443	86	1117	140

5.3. Comparison

Performance improvement with Code Based Implementation as compared to RAP API is 75%.



Figure 11. Performance comparison between Code based and Unmanaged RAP API

By deferring the changeset process and adapting the deep create approach in code-based development (Figures 8 and 9) helped to improve the performance. Also, OData V4 has more advantages, like supporting both XML and JSON formats, metadata query on service level only. It can be used to query both root and expanded entities and supports deep expand and query options in expanded entities.

6. Conclusions

The results of this study demonstrate the significant advantages of a code-based approach for achieving enhanced performance outcomes. Consequently, prioritizing a code-based solution is advisable if the project timeline permits. Organizations can improve the efficacy and agility of their data staging processes, resulting in enhanced overall performance. Findings of this study highlight the significant advantages of a code-based approach in achieving superior performance results. Therefore, if project timelines permit, transitioning to a code-based implementation should be given a higher priority. By doing so, organizations can maximize the efficiency and responsiveness of their data staging processes, resulting in improved overall performance.

It is essential to acknowledge that the suitability of this approach may vary based on specific use cases. Different scenarios and requirements might influence the decision-making process. As such, a careful assessment of individual project needs is recommended before finalizing the choice of implementation.

All things considered, our study emphasizes the need of choosing the most suitable method for data staging considering both without ChangeSet and with ChangeSet alternatives. It also stresses the significance of, whenever practical, implementing a code-based approach for performance optimization. Knowing this helps companies make wise decisions to improve data staging processes and reach better degrees of efficiency and production.

REFERENCES

- [1] <https://community.sap.com/topics/abap/rap>.
- [2] https://help.sap.com/doc/saphelp_nw74/7.4.16/en-us/94/a126519eff236ee10000000a445394/content.htm?no_cache=true.
- [3] SAP Performance Statistics. (n.d.). https://help.sap.com/doc/saphelp_nw75/7.5.5/de-DE/40/93b81292194d6a926e105c10d5048d/content.htm?no_cache=true.