Ilkogretim Online - Elementary Education Online, 2021; Vol 20 (Issue 6): pp. 2487-2504
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.06.230

Hybrid Data Integration Architecture For Cloud Enterprise Resource Planning

Mohammed Hussain Research Scholar, Department of Computer Applications, School of Computing Sciences, Hindustan Institute of Technology and Science, Chennai, India. Email:mohammed.hussain6@gmail.com

Dr. J. Thangakumar Associate Professor, Department of CSE, Hindustan Institute of Technology & Science, Chennai, India. Email: tkumar@hindustanuniv.ac.in

ABSTRACT
Cloud-based services and implementations are rapidly growing among higher education institutions and organizations. This provides a challenge to the organisations to redirect their existing integrations, interfaces and API to get the data. The data structure of the cloud is different from the traditional systems and facing challenges for accessing resources. Also, the cloud provides the opportunity to plug and play SaaS products. It allows the institutions to move between different cloud products, but it also provides the challenge of keeping data synchronized seamlessly with the integrations between many third-party systems. The Common Data Model (CDM) encompasses many data entities based on the higher education domain data. In this research work, the Canonical Data Model and Integration Architecture is applied and compared with the direct Peer to Peer data integration from Cloud SaaS system and found this approach effective in terms of performs, maintenance cost and resources. The qualitative and quantitative metrics obtained has been discussed in the result section.

KEYWORDS: ERP, Higher Education, Cloud ERP, Data Integration Architecture, Hybrid Integration Architecture and Canonical Data Architecture

1. INTRODUCTION

The next-generation ERP are now accessible and contending with the customary ERP players. The creator depicts the distinction in implementation phases and naming standards. The creator characterizes "Again class of bundled application programming has arisen over the past decade, apparently combining under a solitary banner, a multibillion-dollar industry that incorporates the world's fourth-largest programming seller, a few other of the biggest software firms and the world's biggest administration consulting organization". ERP Systems are largest software programs which are getting adopted by the fortune organizations and leading global institutions. Highly integrated with organization business process which creates large number of transactional data during everyday business activities. The ERP system features multiple products such as

Human Capital Management [HCM], Financial and Supply Chain Management [FSCM], Customer Relationship Management [CRM], Campus Management Solutions [CS] for Higher Education institutions etc. This provides a complete suite of functionality suitable for different type of organizations such as Manufacturing, Insurance, Retail and Higher Educational institutions etc.

2. LITERATURE REVIEW

The concept of ERP systems emerged in 1990. A researcher has shown interest for developing combined software packages for standardizing the business processes to the industry best practices. The authors describe "ERP systems are like the dream come true. The system promises the complete integration of business process of all the information in the organization which includes FINANCE, HRMS, CRM etc" [1].

In 2004, the profoundly apparent review of rofecoxib concentrated on the business, public, government, and pushed the issue of medication safety. Subsequent medication security issues and reviews have kept the subject of medication well-being in the public eye and have featured very much archived weaknesses in the current medication well-being checking framework. In 2006 report, the Institute of Medicine gave an appraisal of the current framework for assessing and guaranteeing drug security post-endorsement. The discoveries archived in this report are suggestions for the development of the current medication security framework including the expanded utilization of mechanized medical services information for detailing and testing of medication well-being hypotheses [5].

The improvement of approved techniques for the foundation of a post-market hazard distinguishing proof and investigation framework to connect and examine security information from numerous sources. In May 2008, the Institute of Medicine report and Food and Drug Administration Amendments Act, the United States Food and Drug Administration (FDA) gave a report depicting its goal to set up a public, incorporated, an electronic framework for checking clinical item security utilizing different, existing electronic clinical record frameworks and data sets to increase the organization's present capacity [6]

The Sentinel Initiative was dispatched to start up this vision. The few associations have been set up to zero in the utilization of observational information for drug wellbeing research, including the Observational Medical Outcomes Partnership (OMOP). The interest in the utilization of observational information for drug security research isn't restricted to the United States. The EU-ADR and IMI-PROTECT are two European consortia presently chipping away at the advancement of creative modernized framework to distinguish security signals in observational information to enhance unconstrained detailing frameworks [7]

As of late, a great deal of examination has been done in the field of architecture planning and pattern coordinating. It is essential to recognize outline coordinating as the self-loader finding of semantic correspondences between components of two patterns and construction planning, which is the production of a change between various mappings. Additional data and assessment on coordinating and planning apparatuses for mappings and ontology can be found. A large portion of the new work in this space gives a couple of astute correlations of mappings. In our methodology, we expect to make a sanctioned information model that coordinates all blueprints. This approach is called n-way figure 3 coordinating [8].

ERP systems have to properly measure and assess for its performance, reliability and other benefits. The success evaluation model proposed by the authors can be taken further and applied for evaluating Cloud based ERP solution [9].

3. SaaS

Programming as a Service otherwise called cloud application administrations addresses the most regularly used choice for organizations in the cloud market. SaaS uses the web to convey applications which are overseen by an outsider merchant to its clients. A greater part of SaaS applications run straightforwardly through your internet browser, which implies they did not need any downloads or establishments on the customer side. Because of its web conveyance model, SaaS wipes out the need to download and introduce applications on every individual PC. With SaaS, merchants deal with all possible specialized issues like information, middleware, workers, and capacity bringing about smoothed out upkeep and backing for the business [10].

SaaS might be the most valuable alternative in a few circumstances including, new businesses or little organizations that need to dispatch eCommerce rapidly and don't possess energy for client issues or programming transient undertakings that require speedy, simple, and reasonable coordinated effort Applications that aren't required over and over again. For example, charge programming Applications that need both web and versatile access. The following are some of the limitations of SaaS when implementing ERP solution:

3.1 SAAS Limitations and Concerns

3.1.1Interoperability.

Reconciliation with existing applications and administrations can be a significant concern if the SaaS application isn't intended to keep open guidelines for joining. For this

situation, associations may have to plan their own reconciliation frameworks or lessen conditions with SaaS administrations, which may not generally be conceivable.

3.1.2 Seller lock-in.

Merchants may make it simple to join a help and hard to receive in return. For example, the information may not be convenient actually or cost-adequately across SaaS applications from different sellers without bringing about the tremendous expense or in-house designing revamp. Only one out of every odd seller adheres to standard APIs, conventions, and instruments, yet the highlights could be important for certain business assignments.

3.1.3 Absence of incorporation support.

Numerous associations require profound mixes with on-premises applications, information and administrations. The SaaS seller may offer restricted help in such manner driving associations to put inward assets in planning and overseeing combinations. The intricacy of mixes can additionally restrict how the SaaS application or other ward administrations can be utilized.

3.1.4 Information security.

Huge volumes of information must be traded to the backend server farms of SaaS applications to play out the vital programming usefulness. Moving delicate business data to a public-cloud-based SaaS administration may bring about security and consistence notwithstanding massive expenses for moving enormous information responsibilities.

3.1.5 Customization.

SaaS applications offer negligible customization abilities. Since a one-size-fits-everything arrangement doesn't exist, clients might be restricted to explicit usefulness execution, and reconciliations as offered by the merchant. Interestingly, on-premise arrangements that accompany a few programming advancement packs (SDKs) offer a serious level of customization alternatives.

3.1.6 Execution and personal time.

Since the seller controls and deals with the SaaS administration, your clients currently rely upon merchants to keep up the assistance's security and execution. Arranged and unprepared support, digital assaults or organization issues may affect the exhibition of the SaaS application notwithstanding sufficient assistance level understanding (SLA) securities set up.

3.2 Instances of SAAS

The Well-known instances of SaaS include:

- Google Workspace (previously G Suite)

- Drop box
- Sales force
- Cisco WebEx
- SAP Concur
- GoToMeeting

Cloud stage administrations otherwise called Platform as a Service (PaaS)gives cloud parts to certain product while being utilized fundamentally for applications. PaaS conveys a structure for designers that they can expand upon and use to make repeated applications. Managers ought to gauge the advantages against their necessities and figure out which arrangements will give the best outcomes. The following section discusses about important benefits of adopting cloud-based solution for ERP.

4. COMMON DATA MODEL ARCHITECTURE

The solid models utilized in our work depend on XML-information decreased (XDR) e-business outlines. Further principles for depicting figure 3s are for instance XML construction definitions (XSD) and report type definitions (DTD). Typically, a mapping definition portrays a chart of types. A model appears in Figure 1 where types are shown as boxes. An intricate kind comprises components. A component has a name.

The component's construction is given regarding another sort. A basic sort doesn't contain further sorts. The constructions we use in our work were at that point utilized as grandstands and a reason for assessment by a few creators and come from the Microsoft BizTalk worker. Components are named by the mark of their sort as a show, where edge name and resulting type name are indistinguishable.

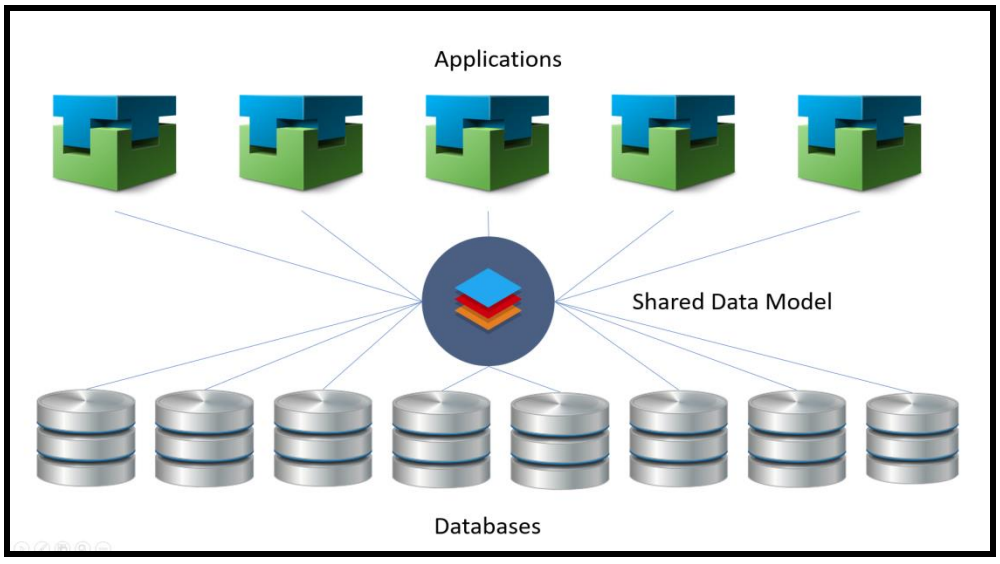

Figure 1: Common Data Model Architecture

5. CANONICAL MODEL ARCHITECTURE

Like Common Data model (CDM) the other type of data model is Canonical Data Model which is shown in Figure 2. It is one of the approaches in creating the data models in the logical manner for the developing of the data from the common sets or the databases. This is the mechanism used to share the data from one data model to the other data model for making data transferring more flexible.

Figure 2: Canonical Data Model Architecture

5.1 DESIGN AND IMPLEMENTATION OF CDM BASED ARCHITECTURE

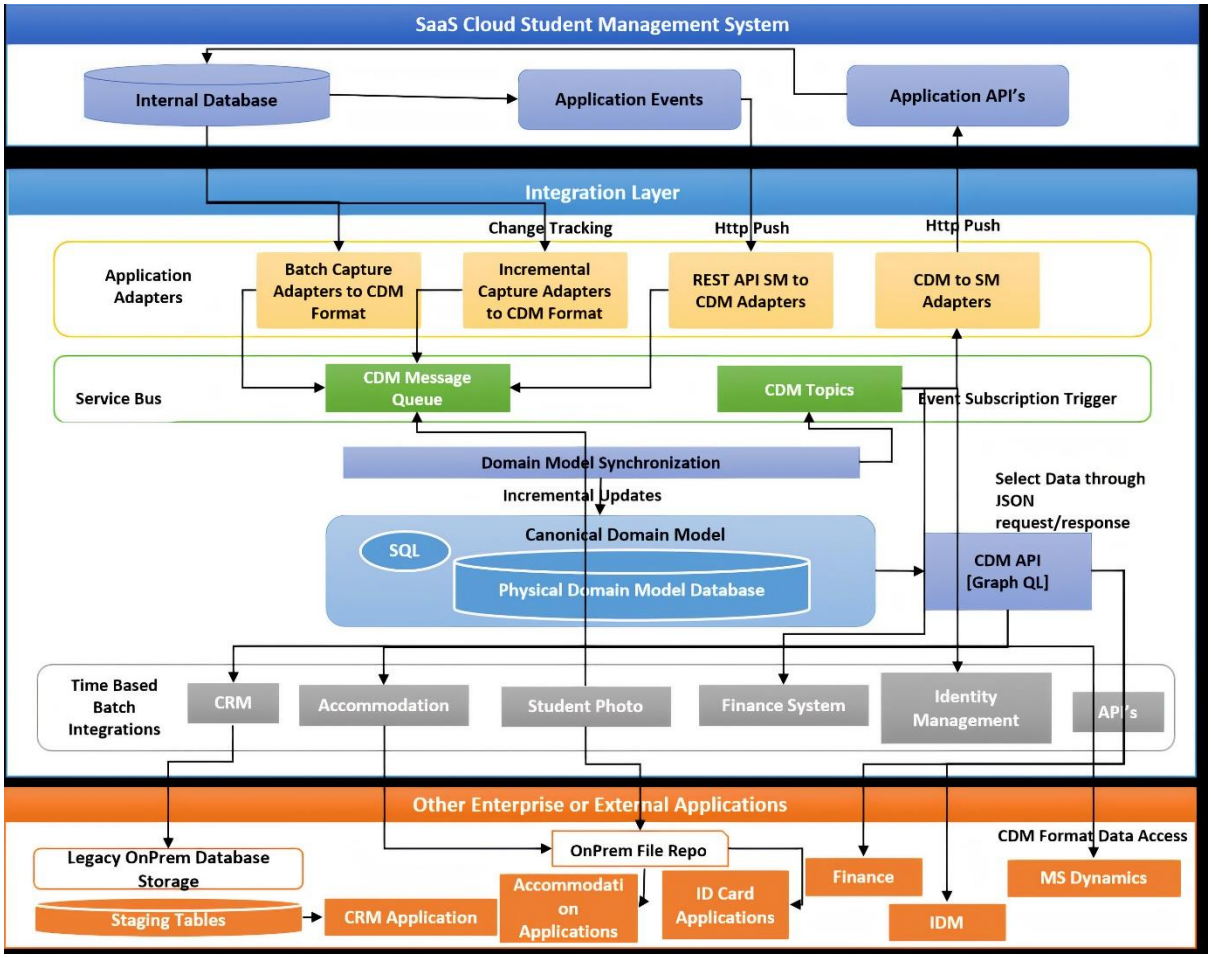

Figure 3: CDM-Based Cloud Integration Architecture

The architecture shown in Figure 3 is based on an enterprise architecture where a new cloud Student Management (SM) system has been procured. In this case the new student management system has its own system specific database structure and data formats which are often not consistent with the way in which the business sees its own data. Since nowadays many organisations procure or use external ERP or SaaS products rather than building in-house application. The challenge is to integrate with the new system but continue to persist the canonical format of the business data within the enterprise at large. The principal of Separation of Concerns within different systems in the architecture is very important to maintain in order to ensure that overall costs are kept to a minimum and that changes in the architecture in the future is not burdened with an expensive decoupling exercise.

The architecture needs to provide adapters at each integration point with all systems to convert from the system specific data format to the canonical data format. Each adapter should be self-contained and should not service any other requirement than the conversion of data to/from the CDM format. This allows for future change to be accommodated easily by switching out the adapters as and when required.

The CDM format encompasses many data entity models that may or may not have a physical representation in the Physical Domain Model shown in the figure 3. Each model is a logical representation of a category of data items. The category and the data items are void of any system specific descriptors or identifiers and collectively the data should be meaningful to the business as a whole and their associated processes.
In an ideal world, all data would be transient within the integration layer, but this would require that all systems within the architecture can integrate using the following methods:

a) Publication of event data to the integration layer that happens within the systems in near real time.
b) Provide a Rest API to enable the integration layer to query the system and receive information.
c) Receive data change event notifications (and associated related data) in near real time from the integration layer.
d) Requesting context specific data through an integration API.

Note that depending on the data flow requirements, it is not necessary that all systems can implement all the methods. In the real world, which the architecture figure 3 is attempting to simulate, systems across the enterprise are a combination of ERPs and locally/externally developed software implementations, many of which are legacy and do not have the required capabilities. They often provide only fixed integration methods such as daily batch file or database dumps. In order to service the requirements of these systems, it often becomes practical and more cost effective to implement a physical manifestation of selected CDM model entities within database tables in the integration layer to be able to provide large datasets to a system within a reasonable time period. The architecture must provide multiple integration patterns to meet the limitations of legacy systems in addition to providing partial/full transience of data flows to more modern and capable systems.

5.2 SM SYSTEM SYNCHRONISATION AND ADAPTERS

In this architecture example, the new SM system is an ERP that offers data exchange through its internal APIs, but it has no capacity to publish data change events that occur in its application layer. The APIs are limited to core data structures in the SM system, and they are a direct representation of underlying database tables and from the format of the CDM model entities understood by the business. Additionally, since not all data required by other systems in the architecture is exposed through APIs, the implementation of adapters to convert the data to the CDM format would be very complex, slow to execute and would result in data gaps. However, the SM system does offer direct access to its database, so theoretically the data gaps could be plugged where it is not available through an API.

A key problem however, is that neither the database nor the APIs provide any internal mechanism for determining exactly what has changed over a given period of time. Since other systems need to receive business events, such as the matriculation of a student or a change to the application status for an applicant, it is not possible to accurately determine this from either the API or the database. The database is hosted on SQL Server, so change tracking can be used to provide row level changes on the internal database entities, but since the SRM application always deletes and then inserts rows of data across multiple tables whenever any change on an application page is made it is only ever possible to detect that a change of some nature may have occurred on a given database entity row. This would result in sending notifications to other systems of an event that has not really occurred.

As a result of these deficiencies, a different approach is required to accurately identify field level changes and publish event messages to other systems in the architecture. The approach must compare data that may have changed with an external database to determine if a real change has taken place to data and also to determine which field values have changed. Within the architecture, the Physical Domain Model database is implemented as a direct reflection of the CDM model structures and contains all the row data for valid records in the SM system.

The comparison process is carried out, not by any bulk SQL snapshot comparison, but by the "Domain Model Synchronisation" message queue subscriber component shown in the figure 3 [referred to as the DMS from this point forward]. It expects JSON data on an Azure Service Bus queue for each specific change or group of changes. The JSON must conform to the CDM model formats. Where the message data is different to the data in the domain model database, the database will be updated, and the message will be passed to the Azure Service Bus CDM 'topic' to trigger subscriber components to push the data to other systems in the architecture. The message will be supplemented with properties that define the specific changes that have occurred. If no change is detected, the message will be discarded.

This approach will effectively overcome the issues with SQL Server change tracking on the SM system, but since the data in the domain database is a point in time replica of selected data in the SM system, there remains the issue of future effective data in the SM system for which there will no change tracking event when the data becomes current at the effective date. To cater for this, a full comparison of all rows in the domain database entities is required on a scheduled basis.

Two of the SM adapters shown in the figure 3 implement the expected functionalities. The following section describes in detail about those two adapters.

5.3 Incremental Capture Adapter

This process executes once a minute to look for changes in specific tables in the SM database where integration data is required. If a change is reported by Change Tracking, the models in the CDM that are related to the nature of the change are created in JSON format and populated with the associated SM data. All transformation from the SM format to the CDM format takes place at this point. The JSON messages are then published to the CDM message queue in the integration layer.

5.4 Batch Capture Adapter

For future effective changes that become current, no trigger is provided that the Incremental Capture Adapter would be able to detect. Where time-based events are needed to be captured, a supplementary process is required to perform snapshot analysis and publish identified changes based on the differences found. The Batch Capture Adaptor uses the same data format conversion processes as the Incremental Capture Adapter to create a virtual full model set for all records in the SM systems and then compares these with the Physical Domain Model entities in the integration layer. Each change in a CDM model is converted to JSON and published to the CDM message queue in the same way as the Incremental Capture Adapter. Although row changes are identified using this process, the specific field changes are still identified in the DMS process.

As with the application adapters servicing the SM to CDM model conversion requirements, all other adapters servicing other systems will also always publish any change to the queue that the DMS process reads. The process has no knowledge about any specific system in the architecture. It is solely concerned with identifying/capturing changes and routing messages to the CDM topic in the Azure Service Bus. In order for data to be consumed by the SM system another dedicated adapter is implemented to subscribe to the relevant model changes in the CDM topic and convert the format to the API format of the SM system. The CDM to SM Adapter performs these subscriptions and conversion tasks and then submits the data to the SM system through its APIs.

6. DOMAIN MODEL SYNCHRONIZATION PROCESS

It's important to understand that the fields in the physical domain model entities are a direct representation of the properties of the CDM models. In addition, the CDM models are independent of any one system and can contain data in different properties/fields that are sourced from different systems. For example, the CDM application model may contain 30 properties, 20 properties are associated with data owned by the SM system, 8 properties are associated Microsoft Dynamics and 2 are associated with the Finance System. The DMS must only allow changes to specific field values to be communicated to other systems and/or updated in the Physical Domain Model where the ownership of the property/field matches the designated source of the message. This is managed through attributes assigned to the CDM model definitions. Each CDM model must contain an

attribute that determines the primary owning system where a property on the model represents data owner by another system, the property must be assigned its own attribute to override the owning system.
The primary owning system must own the overarching subject matter associated with the model. For example, the SM system is where applicant data is added and managed. Therefore, only the SM system can be the primary owner of the application CDM model and be allowed to add new rows in the application table in the physical domain model. However, if an application message is passed to the DMS that contains values for properties for which it is not the owner, the DMS will not include the values in any inserts or updates when synchronising the physical model. If the Finance System (or it's adapter) publishes a CDM application message to the DMS queue, the subject of the data change (e.g. the applicant and their application) must already exist in the application table in the physical domain model. Only then will the specific properties designated as being owned by the Finance System be updated in the application table. Change detection is always limited to the ownership scope of the data values associated with the properties owned by the originator of the message.

7. PROGRAMMING COMPONENTS FOR CDM

The CDM API component shown in the figure 3 exposes the CDM model structures defined in the integration layer and enables JSON queries to be constructed and executed to return data to architecture systems that are capable of consuming data through APIs. The data is provided to consumers in the CDM format and is the preferred approach for systems integration. External systems should perform their own conversion from the CDM format to their internal formats thus ensuring Separation of Concerns. Similarly, if an external system needs to push data into the integration layer, it should convert from its internal format into the CDM format. In this example architecture, only the Microsoft Dynamics integration point is capable of consuming and publishing data in the CDM format. All other system does not have the native capacity to do the conversion and/or communicate over HTTP.

7.1System Adapters

As with the SM system, if an external system is incapable or unwilling to perform the conversion, a dedicated adapter should be implemented to encapsulate the conversion logic from/to the CDM format. All of the adapters depicted in the architecture figure 3 are conversion proxy adapters that service the external system. By encapsulating the logic in dedicated adapters in this manner, it is simple to remove a system from the architecture without affecting any other system. Similarly, if the system undergoes any changes, only the adapter code must be updated. If the external system is unable or unwilling to perform the conversion, a dedicated adapter should be implemented to encapsulate the conversion logic from/to the CDM format. By encapsulating the logic in dedicated adapters in this manner, it is simple to remove a system from the architecture without

affecting any other system. Similarly, if the system undergoes any changes, only the adapter code must be updated.

7.2 CRM Adapter

This is a daily scheduled task that queries the CDM API to convert the data to the required format for CRM and then inserts the data into a staging table. The CRM system collects and stores the data in its internal database. Similar to the CRM Adapter, the Accommodation Adapter runs once a day and converts data to the required format, but this time it outputs a file to an on-premises file repository.

7.3 Student Photographs Adapter

Student photographs are captured in the ID Card Online Application and the photographs are stored in a file repository. The Student Photographs adapter executes four times a day and looks for new or updated photographs in the repository. Any changes are pulled into the data and converted to the Contacts CDM model format and then published to the CDM Message Queue in the internal Azure Service Bus.

7.4 Finance System Adapters

Integration with the Finance System is based on a notification push pattern and is required in near real time. Two adapters service the Finance System, one is a subscriber to events that are published to the CDM topics and responds to the events by converting the CDM data to the finance required XML format and then publishing the XML over HTTPS to a remote Url. The other is a Rest API that exposes an integration point over HTTPS that the remote finance system uses to push data change notification into the integration layer. This adapter converts the finance JSON message to the CDM JSON message format.

7.5 Identity Management Adapters

An external system manages applicant, student and staff access to the University network and needs to receive data changes relating to their status to grant and revoke access accordingly. Two adapters are used. One provides a Rest API that the Identity Management System uses to request delta changes to applicant, student and staff status twice a day. The adapter retrieves the data on demand through the CDM API, converts it to the required format, and then responds over HTTP. The other adapter is also a RESTAPI and allows the Identity Management System to send messages to the integration layer to communicate system access level changes to other systems in the architecture. Again, the adapter converts the system format to the CDM format and then publishes the CDM message to the Azure Service Bus CDM message queue.

8. RESULTS

By implementing the concept of Common Data models (CDM) and the integration architecture, we can observe that there is maintenance of information in the clear-cut format in the datasets. The domain model implementation improves the data quality and provides the single source of truth for the data integration. The architecture also provides different adapters such as http, File, ODBC, Graph QL, and JSON to integrate with heterogeneous set of internal and external third-party systems. The architecture provides the faster data access and high reliability than accessing the SaaS cloud database. The data transformations are performed, and the canonical data is stored in the Data Mart database which reduces the transformations required by the integrations leads to performance improvement and high data accuracy. The full Synchronization performance snap shots are shown in Figure 4, Incremental Synchronization Performance is shown in Figure 5 and finally the datasyn chronization performance is shown in Figure 6.

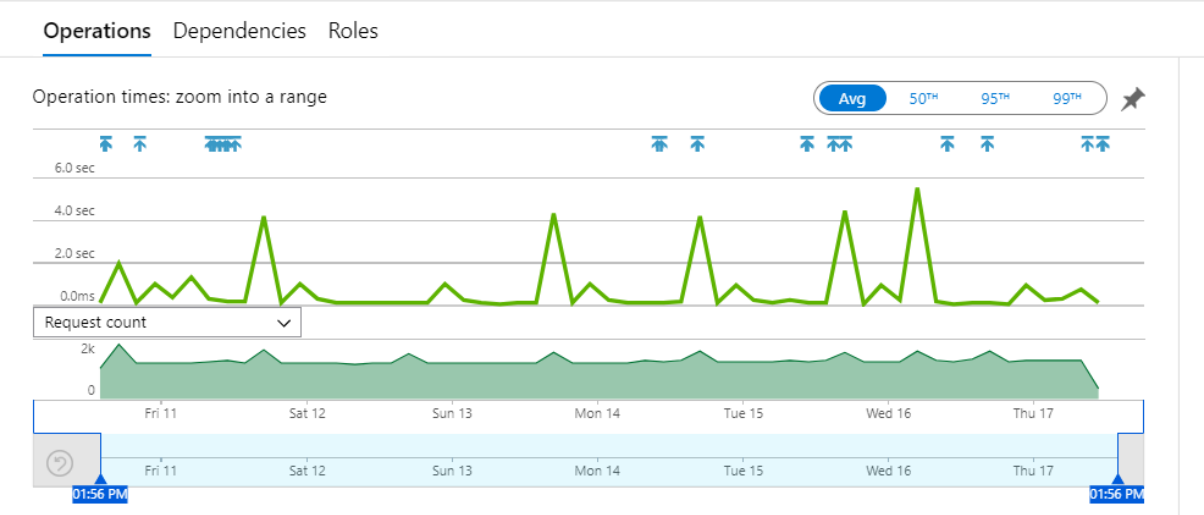

Figure 4: Full Synchronization Performance

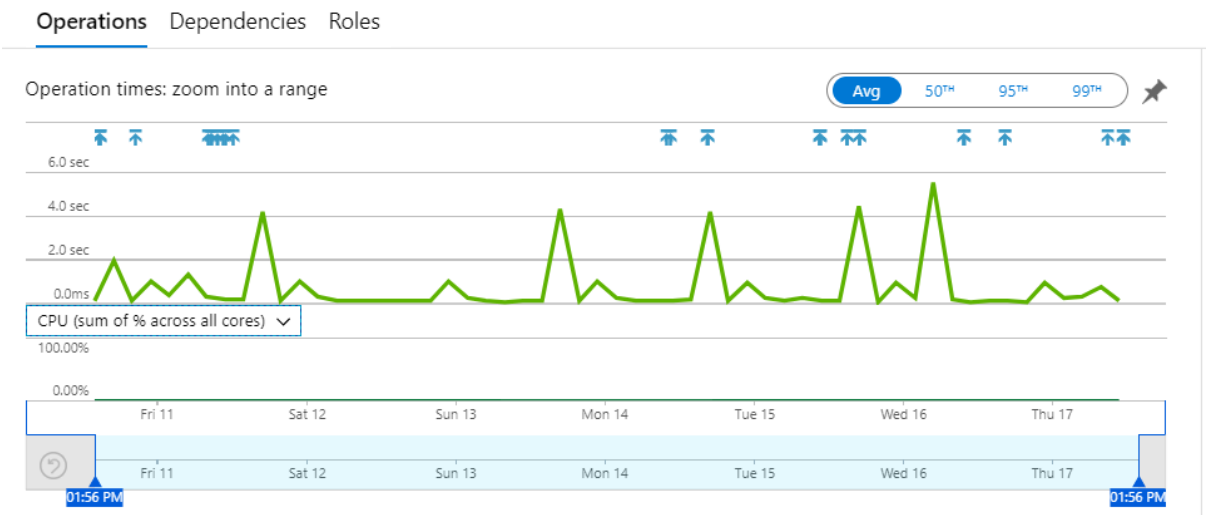

Figure 5: Incremental Synchronization Performance

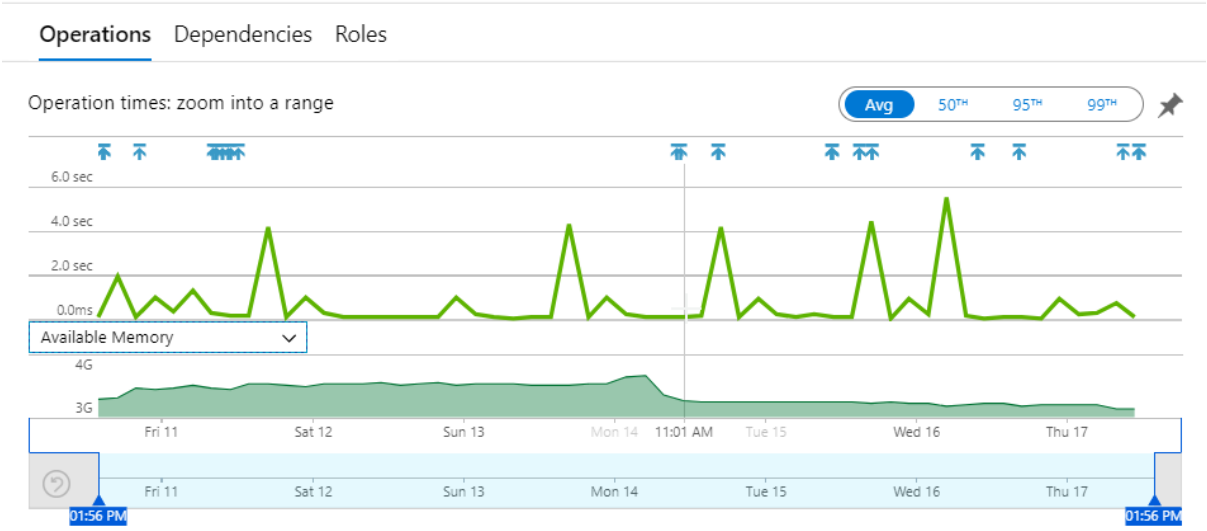

Figure 6: Data Synchronization Performance

8.1 Evaluation Metrics:

The following metrics can provide multiple benefits for the organizations and some of the key metrics which can be obtained are listed below:

1. Performance
2. Functionality
3. Usability

The following provides the list of quality metrics which are helpful to assess the implementation of the integration framework and success of the overall ERP system integration in the cloud environment.

8.2 FUNCTIONALITY METRICS

Table 1 represent the list of functional metrics

Function Number	**Description**	Qualitative and quantitative numbers – option Availability
F1.	DATA_ACCESSIBILITY	**Yes**
F2.	DATA_CONTROL	**Yes**
F3.	SYSTEM_CONTROL	**Yes**
F4.	EASY_TO_INTEGRATE	**Yes**
F5.	SECURITY_OF_DATA	**Yes**
F6.	RELIABILITY_OF_SYSTEM	**Yes**
F7.	INTEGRATION_OF_REALTIME_ DATA	**Yes**
F8.	DEMAND_FORCAST_ACCURACY _PERCENTAGE	**Only** quantitative **number option**
F9.	SCHEDULE_ADHERENCE_PERC ENTAGE quantitative (number)	**Only** quantitative **number option**
F10.	IMPROVED_CUSTOMER_SATISF ACTION yes/no - using feedback/survey	**Yes /No feedback /survey**

8.3 USABILITY METRICS

Table 2 represents the list of usable metrics in the database

Function Number	**Description**	feedback/survey **– option Availability**
U1.	EASY_TO_USE_INTERFACE	**Yes**
U2.	EASY_TO_MIGRATE_TO_CLOU D	**Yes**
U3.	EASY_TO_UPLOAD	**Yes**
U4.	EASY_TO_ACCESS_FROM_MOB ILE	**Yes**
U5.	EASY_TO_INTEGRATE	**Yes**
U5.	EASY_TO_MODIFY_DATA_SOU RCES	**Yes**

U6.	EASY_TO_USE_ERP_INFORMATION	**Yes**

8.4 PERFORMANCE METRICS

Table 3 list of performance metrics available in the database

Function Number	**Description**	per day/per week/per month – **option Availability**
P1.	REDUCED_IT_SPENDING	**Yes**
P2.	DATA_REFRESH_CYCLE_TIME	**Yes**
P3	DOWN_TIME_MINUTES	**Yes**
P4.	REJECT_RATIO	**Yes**
P5.	NUMBER_OF_CONCURRENT_USERS	**Yes**
P6.	NUMBER_OF_PARALLEL_UPLOADS	**Yes**
P7.	NUMBER_OF_PARALLEL_UPDATES	**Yes**
P10.	NUMBER_OF_REPORTS_GENERATED	**Yes**
P11.	DATA_GROWTH_PERCENTAGE	**Yes**

9. CONCLUSION AND FUTURE DIRECTIONS

ERP is one of the solutions for managing Enterprises; Most of the ERP's suitable for the Manufacturing and IT industries. Even though there are ERP solutions available for educational institutions, still challenges considering the differences between the functionality of enterprises and institutions. The proposed research focuses on analyzing the existing frameworks and interfaces which are suitable for higher learning institutions. This research aims to develop conceptual, technical frameworks and integration architecture for higher learning institutions implementing Cloud ERP considering the dynamic nature of requirements for integrating the third-party solutions like library, accommodation, biometric and attendance and other third-party systems. The outcome of the proposed study promotes developing data Integration Architecture and an evaluation methodology, metrics to find the efficiency or the effectiveness of the ERP solution in the Higher Educational Domain. The standard Peer-To-Peer data integration between the cloud ERP and the results of the third-party system from the integration depends upon the developers or implementers queries or access patterns. There won't be a single source of truth, as the developer's translation of data may vary. Any changes to the data structure in cloud ERP during the cloud upgrade would pose an

integration problem. This will impact all the integrations in the peer-to-peer data integration structure. In CDM architecture, only the domain model needs to be updated which be easier instead of updating all integrations individually. Therefore, it is important to provide a canonical data model for data integration that would seamlessly integrate from cloud ERP system to any third party internal/external system.

The Hybrid model including features for supporting newer cloud databases and integrating the data in multiple formats like JSON, XML and others. The main advantage of this model architecture provides less maintenance overhead and faster integration implementation to other systems, which is very crucial in large scale ERP implementation projects. It also gives the developer more freedom to be technically creative. The future work should also add additional factors and measures which enhances on-the-fly transformations of the model and security to perform end-to-end encryption of the data during integrations.

The future work should also add additional factors and measures which enhances on the fly transformations of the model and security, by using right and flexible trust model like CDM which is very much needed for the ERP system considering the security challenges with the cloud systems and third-party solutions used in the academic institutions.

REFERENCES

[1]. Davenport, T. H. (1998). Putting the enterprise into the enterprise system. Harvard business review, 76(4).

[2]. Tjoa, A. M., Xu, L., & Chaudhry, S. (2006). Research and practical issues of enterprise information systems. In IFIP TC 8 International Conference on Research and Practical Issues of Enterprise Information Systems (CONFEINS 2006).

[3]. Daniel T. Brooks, Brandon Becker and Jerry R. Marlatt, "Computer Applications in Particular Industries: Securities appearing in Bige- low, Computers & The Law", American Bar Association, Section of Science and Technology, Third Edition 1981 at 250, 253.

[4]. Nagpal, S., Khatri, S. K., & Kumar, A. (2015, May). Comparative study of ERP implementation strategies. In 2015 Long Island Systems, Applications and Technology (pp. 1-9). IEEE.

[5] Ray, S. (2013). Big data in education. Gravity, the Great Lakes Magazine, 20, 8-10.

[6] Clarke, J., Dolado, J., Harman, M., Jones, B., Lumkin, M., Mitchell, B., Mancoridis, S., Rees, K., Roper, M., Shepperd, M.: Reformulating software engineering as a search problem. IEEE Proceedings on Software 150(3), 161–175 (2003).

[7]Gable, G. G., Chan, T., & Tan, W. G. (2001). Large packaged application software maintenance: a research framework. Journal of software maintenance and evolution: research and practice, 13(6), 351-371.

[8]. Tjoa, A. M., Xu, L., & Chaudhry, S. (2006). Research and practical issues of enterprise information systems. In IFIP TC 8 International Conference on Research and Practical Issues of Enterprise Information Systems (CONFEINS 2006).
[9] Kronbichler, S. A., Ostermann, H., & Staudinger, R. (2010). A comparison of ERP-success measurement approaches. JISTEM-Journal of Information Systems and Technology Management, 7, 281-310.
[10] Subramanian, D. V., Hussain, M. K., & Geetha, A. (2011, November). Measurement process and multi-dimensional metric model for evaluating KM systems. In 2011 International Conference on Research and Innovation in Information Systems (pp. 1-6). IEEE.