International Journal of Computer Science and Software Engineering (IJCSSE), Volume 8, Issue 7, July 2019
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 161-165

# Improving Entity-Attribute-Value Architecture Using Middleware Layer

**Moayed Khodairy[1] and Fahad ALqurashi[2]**

[1, 2] Computer Science Department, Faculty of Computing and Information Technology King Abdulaziz University, Jeddah, Saudi Arabia

fahad@kau.edu.sa

**ABSTRACT**

The Entity-Attribute-Value (EAV) data model, is widely used solution to represent high dimensional and sparse data it is actually provides high flexibility, but EAV does not explicitly provide the constrains as a traditional database (e.g. data type, length, default value, allow null, etc.). In this paper, we implemented middleware layer that managing EAV operations to reflect the features that applied on traditional database and more, for purpose of improve complexity and efficiency. The experiments were done using Microsoft EntityFramework ORM tool, Microsoft SQL Server database, and implementation language was C#.

Keywords: *EAV, Open Schema, Sparse Data, High Dimensional Data, Improving EAV.*

## 1. INTRODUCTION

Actually, High dimensional and sparse data require to be used in several types of application. We require an open schema data model to support dynamic schema change. In open schema data models, logical model of data is stored as data rather than as schema, so changes to the logical model can be made without changing the schema. In open schema data model, schema is kept as data. Entity- Attribute-Value (EAV) is the widely used open schema data model to handle these challenges of data representation. However, EAV suffers from complexity and not search efficient. In this paper, we have implemented an EAV middleware layer to improve complexity and efficiency. This model is search efficient as well compared to traditional data model.

## 2. BACKGROUND

### 2.1 Horizontal and Vertical Data Representation

Depending on the business requirements, various ways of storing data can be implemented. The traditional way to represent data is the horizontal schema, as outlined in Figure 1, Horizontal Schema table, where data is stored as a rows and table columns horizontally, horizontal schema is the efficient method to handle concrete attributes and well-defined business requirements. There is another way to represent data is the vertical schema as outlined in Figure 1, Vertical Schema table, where data is stored as key/value pairs vertically, vertical schema is the efficient method to handle dynamic attributes and sparse data.

Horizontal Schema

| Id | A1 | A2 | A3 |
|---|---|---|---|
| 1 | | v(1, 2) | v(1,3) |
| 2 | v(2, 1) | v(2, 2) | |
| 3 | | v(3, 2) | |
| 4 | v(4, 1) | | v(4, 3) |

Vertical Schema

| Entity | Attribute | Value |
|---|---|---|
| 1 | A2 | v(1, 2) |
| 1 | A3 | v(1, 3) |
| 2 | A1 | v(2, 1) |
| 2 | A2 | v(2, 2) |
| 3 | A2 | v(3, 2) |
| 4 | A1 | v(4, 1) |
| 4 | A3 | v(4, 3) |

Figure 1 a sparse dataset represented in the horizontal and vertical schema.

### 2.2 Entity-Attribute-Value Architecture

The Entity-Attribute-Value data model (EAV), also known with other names as open schema or vertical database, provides the dynamic storage with unlimited attributes [3]. A basic EAV architecture is consist of three tables that treated as one object, in the other words each table in the vertical schema is represented by three tables in EAV schema, as outlined in Figure 2.

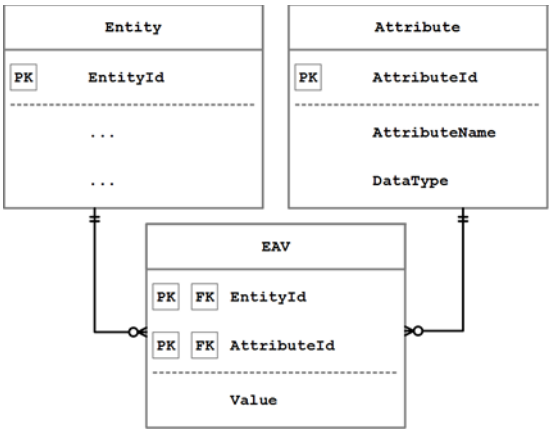


*Fig. 2. basics EAV schema.*

- The Entity table: which contains the general information about the entities, which will not change very often and has fixed attributes can and should be stored in a conventional data table, in this case the advantages of the EAV layout do not exceed its disadvantages [4].
- The Attribute table: which contains the metadata. This data represents the constraints for values of EAV table. In additional to name and data type this table may also contain constraints that could possibly be used (e.g. max value, min value, etc.).
- The EAV table: which contains the value that is stored as a string. Entity and Attribute are referenced via foreign key.

Advantages of the EAV design:

- Flexibility: the attributes can increase to unlimited numbers without need to redesign schema.
- Space: in traditional database design, need to reserve additional storage space for attributes whose values would be null. In contrast EAV design, empty attribute does not need to be reserved.

Disadvantages of EAV design:

- Retrieval Data: actually, in EAV design when retrieving many entities is needed to implement multiple joins which leads to an increase of execution time compared with traditional design that need less joins which perform in less time [5].
- Constraints Checking: compared with traditional design there is not explicitly constraint checking in EAV design which should be implemented in the user interface [3].

## 3. METHODOLOGY

Our solution to overcome the defects associated with the EAV data model, is to use middleware layer that integrated with Object- Relational Mappers (ORMs) component in Data Layer, as outlined in Figure 3, for purpose of handle all operations between the front-end layer and the persistence layer that directly associated with the EAV tables. Actually, we depend on ORM components in order to improve the complexity and data retrieving.

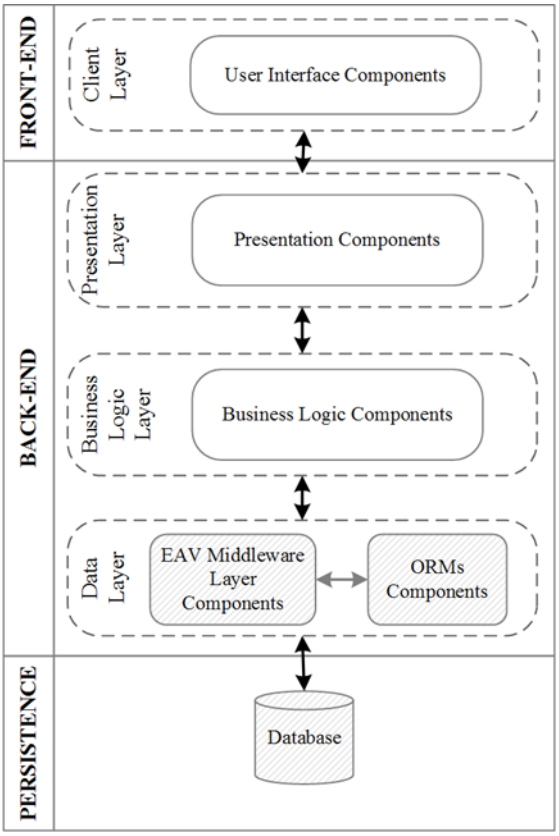


*Fig. 3. middleware layer architecture for EAV data model.*

### 3.1 EAV Middleware Layer

The EAV middleware layer contains the components that organize and manage EAV operations through the use of ORMs components as integrated component. There are many functions are done in EAV middleware layer such as constraints checking, converting values between persistence layer and business logic layer, CRUD operations for EAV tables and more.

### 3.2 Middleware Layer Interfaces

As we mentioned earlier, one table in the traditional design corresponds to three tables in the EAV design. Accordingly, we have designed three interfaces each interface of them is considered as a contract or signature

and all its terms must be implemented when designing EAV tables at database or classes at Business Layer in order to ensure that the middleware layer working smoothly at any EAV design.

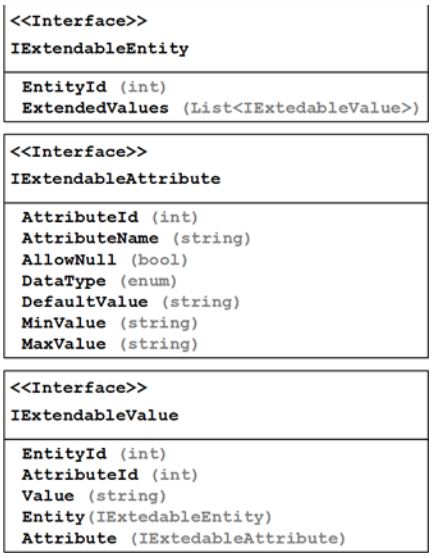


*Fig. 4. middleware layer interfaces.*

As outlined in Figure 4 middleware layer interfaces., we see interfaces and its elements that need to be defined at EAV data model in order to be used EAV middleware layer.

IExtendableEntity: this interface must be implemented by the Entity table in EAV data model and this interface contains two required properties as following:

- EntityId: integet type which represents a primary key for Entity table.
- ExtendedValues: list of IExtendableValue type which represents the related data of EAV table for the entity object.
- IExtenableAttribute: this interface must be implemented by the Attribute table in EAV data model and this interface contains several required properties as following:
- AttributeId: integer type which represents a primary key.
- AttributeName: string type which represents the name of attribute corresponds to the name of column in the horizontal table.
- AllowNull: boolean type which determines whether the value field at EAV table can contain a null value or not.
- DataType: represents the data type which desire to be converted the value to through the converting operation.
- DefaultValue: represents the default value which will be added to value field at EAV table for new records in case no other value is specified.
- MinValue: represents the minimum valid value for value field at EAV table, when specify value less than MinValue then will throw an exception.
- MaxValue: represents the maximum valid value for value field at EAV table, when specify value larger than MaxValue then will throw an exception.

- IExtendableValue: this interface must be implemented by the EAV table in EAV data model and this interface contains several required properties as following:
- EntityId: integer type which represents a foreign key of Entity table.
- AttributeId: integer type which represents a foreign key of Attribute table.
- Value: string type which represents a field value, this value will be converted to specific DataType through the converting operation.
- Entity: IExtendableEntity which represents an entity object data.
- Attribute: IExtendableAttribute which represents an attribute object data.

## 3.3 Constraints Checking

Unlike the traditional model where constraint checking at database level, in EAV model constraint checking at middleware layer. Change of constraints is flexible in EAV model than traditional model, because changing constraints in traditional model requires redesign and recheck values, in contrast EAV model where no need to redesign or recheck values. Constraints checking will be performed during add a new value into the entity object which checked whether the added value corresponds the constraints or not.

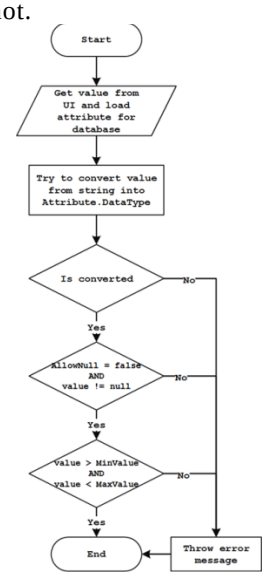


*Fig. 5. constraints checking processes*

As outlined in Figure 5, we see four processes which performed through constraints checking as following:

- Prepare parameters: in this process, the middleware layer loads the required parameters that need to be used in checking processes.
- Converting process: in this process, the middleware layer tries to convert the value from string type into type which specified in Attribute.DataType, If the conversion successfully done will be moved to the next process; otherwise; throw conversion error message.
- Check nullability: in this process, the middleware layer checks the value, in case equals null and the Attribute.AllowNull equals false, will be thrown null error message; otherwise; move to the next process.
- Check range: in this process, the middleware layer compares whether the value in range or out of range corresponds to the Attribute.MinValue and Attribute.MaxValue, if the value in range will be ended the checking operation, otherwise; throw out of range error message.

### 3.4 Retrieval Data

Unlike the traditional model where the data as being organized traditionally in tables and columns, EAV middleware layer loads and stores the related values for specific entity from EAV table into ExtendedValues property in entity object as list of IExtendableValue.
This approach gives us more control and efficient performance to deal with the dynamic data, because not necessary transform the EAV data into the traditional layout (columns and rows) when displaying data, only we need to read the dynamic data as object of extended list of values. As outlined in Figure 4, we can be found all data we need when iterating through the ExtendedValues we can get entity data and attribute data for each value.

## 4. EXPERIMENT

In our experiment, we used EntityFramework ORM tool and SQL Server database, and implementation language was C#. The dataset from excel with 10,000 rows and 4 attributes was carefully converted into the two data models to represent the same data while ensuring no data is lost or misrepresented. Furthermore, both the experiments were done using a PC virtual machine with an Intel quad-core i7 processor with a clock rate of 4.0 GHz and 8GB of physical memory. The operating systemwas Microsoft Windows 10 Professional 64-bit.

## 5. RESULT

As outlined in Figure 6, shows the insertion and retrieval time required by each of the two models to represent the same dataset used in the experiment.

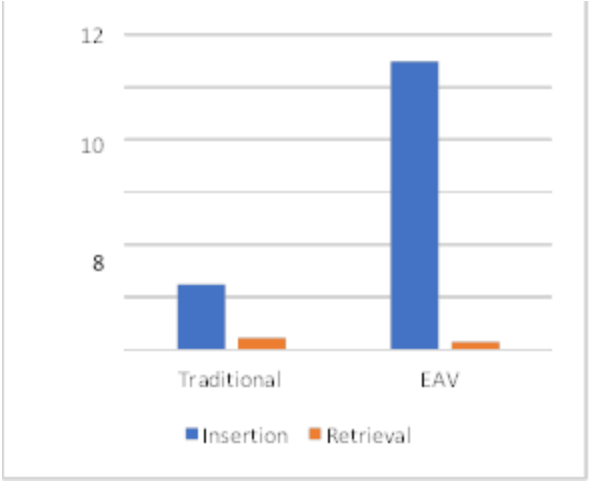


*Fig. 6. insertion and retrieval speed for EAV and traditional data model.*

The traditional data model required (2 seconds and 469 milliseconds) for insertion data and the EAV model required (10 seconds and 973 milliseconds).
The traditional data model required (30 milliseconds) for retrieval data and the EAV model required (438 milliseconds), both data model approximately retrieves data at same time.
From the results obtained above, it is apparent that the both data models, needed roughly the same time to retrieve the same set of data.

## 6. CONCLUSION

EAV data model is most widely used solution in application which needs frequently schema change, high dimensional and sparse data representation, but EAV suffers from complexity and not search efficient. This paper has presented technique to improve EAV data model drawbacks by using middleware layer at data layer that organize and improve EAV operations. Section 2 describes the horizontal and vertical data representation and Entity-Attribute- Value architecture. EAV middleware layer, middleware layer interfaces, constraints checking, and retrieval data in section 3. Experiment in section 4. Section 5 offers the result. Section 6 is the conclusion.

## REFERENCES

[1] Paul, R., et al. Optimized Entity Attribute Value Model: A Search Efficient Representation of High Dimensional and Sparse Data. IBC 2011, 3:9, 1-6. doi: 10.4051/ibc.2011.3.3.0009

[2] Kamau, Augustus, and Waweru Mwangi. "An enhanced entity-attribute- value data model for representing high dimensional and sparse healthcare data." IST-Africa Conference and Exhibition (IST-Africa), 2013. IEEE, 2013.

[3] Jastrow, Torben, and Thomas Preuss. "The Entity-Attribute-Value Data Model in a Multi-tenant Shared Data Environment." P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015 10th International Conference on. IEEE, 2015.

[4] Anhøj, Jacob. "Generic design of web- based clinical databases." Journal of Medical Internet Research 5.4 (2003): e27.

[5] Nadkarni, Prakash M., et al. "Organization of heterogeneous scientific data using the EAV/CR representation." Journal of the American Medical Informatics Association 6.6 (1999): 478-493.

[6] Beckmann, Jennifer L., et al. "Extending RDBMSs to support sparse datasets using an interpreted attribute storage format." Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on. IEEE, 2006.