An Evaluation Framework for Validating the Quality of a Data Vault 2.0 Data Model

Heli Helskyaho, Marko Helskyaho
Miracle Finland Oy
Helsinki, Finland
Heli.Helskyaho@miracleoy.fi
Marko.Helskyaho@miracleoy.fi

Laura Ruotsalainen, Tomi Männistö
University of Helsinki
Helsinki, Finland
Laura.Ruotsalainen@helsinki.fi,
Tomi.Mannisto@helsinki.fi

***Abstract*— Designing databases is essential to provide businesses with high-quality data for effective and correct decision-making. Data warehouses, such as Data Vault 2.0, are important storages for the data and therefore designing the data warehouse is important. The process of database designing is usually time-consuming and requires the expertise of data modeling professionals. Automating the process requires metrics to evaluate the quality of a data model. We used the newly created metrics for a Data Vault 2.0 data model and automated the evaluation process. We created a framework for the evaluation and revealed it as an application for the evaluation process to facilitate efficient and effective database design evaluations.**

I. INTRODUCTION

The more data is available for decision making the more important becomes understanding the data and storing it in a usable format. This process is called database designing. There are different kinds of databases for different purposes. For decision-making, a data warehouse (DW) is often the datastore where all data is gathered in one format or another. The Data Vault 2.0 is a methodology for building DWs and it also defines how a Data Vault 2.0 database should be designed [1], [2], [3], [4]. A raw database is a layer where all data is stored, but the users access the data via other layers of Data Vault 2.0 database. It is important that the raw database has been designed correctly and that it includes all the data needed to support the data access in other layers.

Database designing is often done manually by a human expert and it is time-consuming. Being able to design a database automatically would bring benefits and efficiency [5], [6]. One option for automating the design process would be using Generative AI. Since Generative AI is not deterministic, we cannot trust the results from one generation of a Data Vault 2.0 database to guarantee the quality of the next generation. Therefore, we need a process to verify the results, which applies also when a human expert designs a database. For the verification we created a set of metrics [7]. Using these metrics, it might be possible to automate, or at least semi-automate, the quality evaluation process.

The objective of this paper is to automate the process of evaluating the Data Vault 2.0 data model quality by utilizing our metrics [7]. First, we automated the process by creating a technique for obtaining the measures and metrics automatically. Then, we tested the automation with three different data models. Finally, we created a framework for evaluation and based on this framework an application to evaluate Data Vault 2.0 data models.

The remainder of this paper is organized as follows. Section II introduces the materials and methods used in the automation of the evaluation process. In this section, all measures and metrics used are introduced, and the method is described. Section III describes the process needed for collecting measures and metrics and tests it with three different examples. Section IV introduces the framework created for the Data Vault 2.0 data model evaluation. It starts with setting up the environment and continues explaining the framework. At the end of the section, a tool for using the framework is introduced. Finally, in Section V, we conclude the paper and set the scene for future work.

II. MATERIALS AND METHODS

This paper aims to automate the process of Data Vault 2.0 data model quality evaluation using metrics defined in our previous work [7]. We will first introduce the measures and metrics, then create the scripts for programmatically automating the Data Vault 2.0 data model quality evaluation, test the automation with three data models, and finally we will create a framework for the evaluations.

A. Measures and Metrics

Based on the predefined measures, introduced in Table I, the metrics for defining the quality of a Data Vault 2.0 data model are calculated. These measures include information about the existence of the key elements of a Data Vault 2.0 database, such as Hub, Link, or Satellite tables, or key elements of a relational model, such as primary keys (PK) and foreign keys (FK). This information is needed to be able to obtain the metrics for evaluation.

Using these measures, the metrics shown in Table II are calculated. The metrics are defined based on the Data Vault 2.0 methodology. For example, the number of Link tables cannot be larger than the number of Hub tables (RoT2), since the a Link table has been defined to be a m:n relationship between two or more Hub business keys e.g. Hub tables. Or, a Hub table can never be a child table to a Hub, Link or Satellite table, therefore it is not allowed to have any foreign keys (NoFKH).

TABLE I. DIFFERENT MEASURES NEEDED FOR CALCULATING DATA VAULT 2.0 DATA MODEL QUALITY METRICS [7]

Measure	Measure description
NoTDS	Number of tables in the data source
NoH	Number of Hub tables
NoHCNoTDS	Number of Hub tables minus Number of tables in the data source (NoH-NoTDS)
NoS	Number of Satellite tables
NoL	Number of Link tables
NoPK	Number of Primary keys (PKs)
NoFK	Number of Foreign keys (FKs)
NoFKH	Number of FKs in Hub tables
NoFKS	Number of FKs in Satellite tables
NoFKL	Number of FKs in Link tables
MaxD	Maximum number of Depth in the model
NoPKA	Number of PK columns
NoPKAM	Number of mandatory PK columns
NoFKA	Number of FK columns
NoFKAM	Number of mandatory FK columns
NoPKAH	Number of PK columns in Hub tables
NoPKAL	Number of PK columns in Link tables
NoPKAS	Number of PK columns in Satellite tables
NoFKAH	Number of FK columns in Hub tables
NoFKAL	Number of FK columns in Link tables
NoFKAS	Number of FK columns in Satellite tables
NoAH	Number of columns in Hub tables
NoAL	Number of columns in Link tables
NoAS	Number of columns in Satellite tables
NoMAH	Number of mandatory columns in Hub tables
NoMAL	Number of mandatory columns in Link tables
NoMAS	Number of mandatory columns in Satellite tables

When the specified criteria are satisfied, a score of one is allocated; conversely, if the criteria are not met, the metric is assigned a score of zero. The cumulative sum of all allocated points serves as an aggregate measure. A higher total point score indicates superior model quality.

TABLE II. METRICS AND THEIR EQUATIONS [7]

No	Metric	Equation
1	CDTSHS	NoHCNoTDS = 0 or If NoHCNoTDS < 0, then NoS - Noh >= 1
2	RoT1	NoS / NoH >= 1
3	RoT2	NoH / NoL > 1
4	RPK	(NoH + NoL + NoS) / NoPK = 1
5	MaxD	<= 3
6	RPKH	NoPKAH / NoH = 1
7	RPKL	NoPKAL / NoL =1
8	RPKS	NoPKAS / NoS >= 2
9	NoFKH	= 0
10	RFKS	NoFKS / NoS =1
11	RFKL	NoFKL / NoL >= 2
12	RAH	NoAH / NoH >= 4
13	RAL	NoAL / NoL >= 5
14	RAS	If the Satellite table does not have the hashdiff column NoAS / NoS > 3, if the hashdiff column is used (recommended) then NoAS / NoS > 4.
15	RMPKA	NoPKAM / NoPKA = 1
16	RMFKA	NoFKAM / NoFKA = 1
17	RMAH	NoAH / NoMAH = 1
18	RMAL	NoAL / NoMAL = 1

B. Methods

We automated the process of Data Vault 2.0 data model quality evaluation in the Oracle RDBMS environment. First, we created a technique and a framework for obtaining the measures and metrics automatically. For the evaluation framework we defined three database schemas for different purposes:

- SOURCE schema for storing the objects of the source database
- DVDW schema for storing the objects of the data model to be evaluated
- DV schema to hold the data of measures and metrics in general and for a particular data model.

We used the Data Definition Language scripts (DDLs) of the source database to create the objects in a database in the SOURCE schema. This is needed to be able to obtain the measures related to the source database. Then, we used ChatGPT 3.5 to generate the DDLs for the Data Vault 2.0 objects and executed them in a database using the DVDW schema. The DDLs can be generated using any technique, including manual database designing. The only requirement is that there are DDLs available for the Data Vault 2.0 database under evaluation. Then, using a PL/SQL package [8] in a database, we acquired the values of measures, metrics and the quality score using the 18-point system [7] for the data model by querying the data dictionary.

We tested the evaluation framework with three different source databases that were chosen to be simple for easy understanding but different from each other to create a larger test base for the metrics and the process.

Finally, we created an application to evaluate Data Vault 2.0 data models. This application was created using a low-code tool called Oracle Application Express (APEX) [9] and the PL/SQL scripts created for the evaluation framework.

III. AUTOMATING THE MEASURE AND METRIC COLLECTION PROCESS

In this Section, we will discuss the setup of the environment, collect the data for measures and metrics, and finally calculate the quality value for the data model using the 18-point system [7].

A. Collecting the Measure data

We created SQL queries for collecting the data needed for measures. In Table III the SQL clauses for acquiring the data are introduced. The data is collected from the data dictionary views of an Oracle Database.

During our automation work we noticed some problems with the original measures and metrics. Compared to the original measures [7] we changed the names of MaxD to MMaxD and NoFKH to MNoFKH to avoid conflicts with the metric names. We also added one more measure, NoDiff, to accommodate the metric RAS, which should be >3 if there are no Hashdiff columns in Satellite tables, and > 4 if there are. Hashdiff is not a mandatory column according to the Data Vault 2.0 methodology, but it would be best practice to create one to

improve the performance of loading data to the Data Vault 2.0 database. The value of a Hashdiff column is compared to a hash value of the new data loaded to identify if the new data is different from the one already stored in the table. If the data has not been changed, it does not need to be stored again. Without the Hashdiff column, the data in all columns in the Satellite table must be compared to the data being loaded, column by column.

TABLE III. MEASURES AND THE SQL CLAUSES FOR CALCULATING THEM

Measure	Schema	SQL Clause
NoTDS	SOURCE	select count(*) from user_tables;
NoH	DVDW	select count(*) from user_tables where upper(table_name) like 'HUB%';
NoHCNoTDS	DVDW	NoHCNoTDS:=NoH - NoTDS;
NoS	DVDW	select count(*) from user_tables where upper(table_name) like 'SAT%';
NoL	DVDW	select count(*) from user_tables where upper(table_name) like 'LINK%';
NoPK	DVDW	select count(*) from user_constraints where constraint_type = 'P';
NoFK	DVDW	select count(*) from user_constraints where constraint_type = 'R';
MNoFKH	DVDW	select count(*) from user_constraints where constraint_type = 'R' and upper(table_name) like 'HUB%';
NoFKS	DVDW	select count(*) from user_constraints where constraint_type = 'R' and table_name like 'SAT%';
NoFKL	DVDW	select count(*) from user_constraints where constraint_type = 'R' and table_name like 'LINK%';
MMaxD	DVDW	with pur as (select uc.table_name, uc.constraint_type, uc.constraint_name, uc.r_constraint_name, ucc.column_name, uc.delete_rule, max(decode(uc.constraint_type,'R',1,0)) over(partition by uc.table_name) is_r from user_constraints uc, user_cons_columns ucc, user_tables ut where uc.table_name = ucc.table_name and uc.table_name = ut.table_name and uc.constraint_name = ucc.constraint_name and uc.constraint_type in ('P', 'U', 'R')), son_mom as (select distinct s.table_name son, m.table_name mom, m.constraint_type, s.column_name son_column, m.column_name mom_column, s.constraint_name, s.delete_rule from (select * from pur where constraint_type = 'R' or is_r = 0) s left join pur m on s.r_constraint_name = m.constraint_name and s.table_name != m.table_name) select max(lvl) MaxD from (select son, mom, level lvl from son_mom where son not like '%TABLExxyy%' start with mom is null connect by nocycle mom = prior son order siblings by mom, son) ;
NoPKA	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c where c.constraint_type = 'P' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name;
NoPKAM	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c, user_tab_columns tc where c.constraint_type = 'P' and tc.nullable = 'N' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and tc.table_name = cc.table_name and tc.column_name = cc.column_name;
NoFKA	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c where c.constraint_type = 'R' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name;
NoFKAM	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c, user_tab_columns tc where c.constraint_type = 'R' and tc.nullable = 'N' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and tc.table_name = cc.table_name and tc.column_name = cc.column_name;
NoPKAH	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c where c.constraint_type = 'P' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and cc.table_name like 'HUB%';
NoPKAL	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c where c.constraint_type = 'P' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and cc.table_name like 'LINK%';
NoPKAS	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c where c.constraint_type = 'P' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and cc.table_name like 'SAT%';
NoFKAH	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c where c.constraint_type = 'R' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and cc.table_name like 'HUB%';
NoFKAL	DVDW	select count(cc.column_name) into NoFKAL from user_cons_columns cc, user_constraints c where c.constraint_type = 'R' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and cc.table_name like 'LINK%';
NoFKAS	DVDW	select count(cc.column_name) from user_cons_columns cc, user_constraints c where c.constraint_type = 'R' and c.constraint_name = cc.constraint_name and c.table_name = cc.table_name and cc.table_name like 'SAT%';
NoAH	DVDW	select count(column_name) from user_tab_columns where upper(table_name) like 'HUB%';
NoAL	DVDW	select count(column_name) from user_tab_columns where upper(table_name) like 'LINK%';
NoAS	DVDW	select count(column_name) from user_tab_columns where upper(table_name) like 'SAT%';
NoMAH	DVDW	select count(column_name) from user_tab_columns where upper(table_name) like 'HUB%' and nullable = 'N';

NoMAL	DVDW	select count(column_name) from user_tab_columns where upper(table_name) like 'LINK%' and nullable = 'N';
NoMAS	DVDW	select count(column_name) from user_tab_columns where upper(table_name) like 'SAT%' and nullable = 'N';
NoDiff	DVDW	select count(column_name) from user_tab_columns where upper(table_name) like 'SAT%' and upper(column_name) like '%DIFF%';

B. *Collecting the Metric data*

When the measures have been collected, the metrics can be defined as shown in Table IV. For each metric, a point is awarded if the criteria are met; otherwise, zero points are assigned. The cumulative sum of these points determines the overall quality score of the data model. A higher score indicates better quality.

TABLE IV. METRICS AND THEIR LOGIC

No	Metric	Logic
1	CDTSHS	IF (NoHCNoTDS = 0) or ((NoHCNoTDS < 0) and ((NoS - Noh) >= 1)) THEN CDTSHS:=1; ELSE CDTSHS:=0; END IF;
2	RoT1	IF (NoS / NoH >= 1) THEN RoT1:=1; ELSE RoT1:=0; END IF;
3	RoT2	IF (NoH / NoL > 1) THEN RoT2:=1; ELSE RoT2:=0; END IF;
4	RPK	IF (((NoH + NoL + NoS) / NoPK) = 1) THEN RPK:=1; ELSE RPK:=0; END IF;
5	MaxD	IF MMaxD <= 3 THEN MaxD=1; ELSE MaxD:=0; END IF;
6	RPKH	IF (NoPKAH / NoH = 1) THEN RPKH:=1; ELSE RPKH:=0; END IF;
7	RPKL	IF (NoPKAL / NoL =1) THEN RPKL:=1; ELSE RPKL:=0; END IF;
8	RPKS	IF (NoPKAS / NoS >= 2) THEN RPKS:=1; ELSE RPKS:=0; END IF;
9	NoFKH	IF MNoFKH=0 THEN NoFKH := 1; ELSE NoFKH := 0; END IF;
10	RFKS	IF (NoFKS / NoS =1) THEN RFKS:=1; ELSE RFKS:=0; END IF;
11	RFKL	IF (NoFKL / NoL >= 2) THEN RFKL:=1; ELSE RFKL:=0; END IF;
12	RAH	IF (NoAH / NoH >= 4) THEN RAH:=1; ELSE RAH:=0; END IF;
13	RAL	IF (NoAL / NoL >= 5) THEN RAL:=1; ELSE RAL:=0; END IF;
14	RAS	IF (NoDIFF = 0 and (NoAS / NoS > 3)) THEN RAS:=1; ELSIF (NoDIFF > 0 and (NoAS / NoS > 4)) THEN RAS:=1; ELSE RAS:=0; END IF;
15	RMPKA	IF (NoPKAM / NoPKA = 1) THEN RMPKA:=1; ELSE RMPKA:=0; END IF;
16	RMFKA	IF (NoFKAM / NoFKA = 1) THEN RMFKA:=1; ELSE RMFKA:=0; END IF;
17	RMAH	IF (NoAH / NoMAH = 1) THEN RMAH:=1; ELSE RMAH:=0; END IF;
18	RMAL	IF (NoAL / NoMAL = 1) THEN RMAL:=1; ELSE RMAL:=0; ELSE IF;

C. *Defining the Score for a Data Model Quality*

We tested the automated quality evaluation with three data models. When generating the DDLs using ChatGPT on January 22nd 2024, there were errors which did not appear in previous experiments. Generation systematically added “, -- Add other attributes as needed” after the last column in Hubs causing an error. The constraints (PK and FK) were also added twice for the Links: first when creating the table and at the end of the DDL file. This behavior proves the need for metrics and automatic evaluation of the data model quality since Generative AI is non-deterministic; previous tests do not guarantee future results.

In Fig. 1, there is a simple example of a birth data source, including four tables: Birth, Baby, Mother, and Midwife.

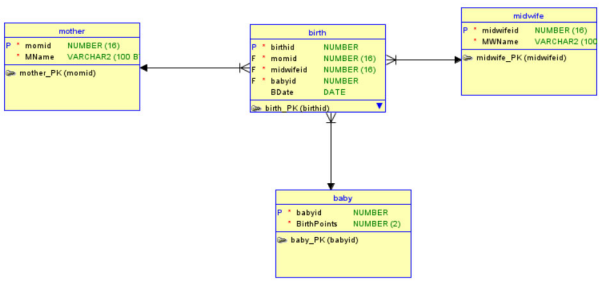

Fig. 1. Birth source database

The Data Vault 2.0 data model generated by ChatGPT is shown in Fig. 2.

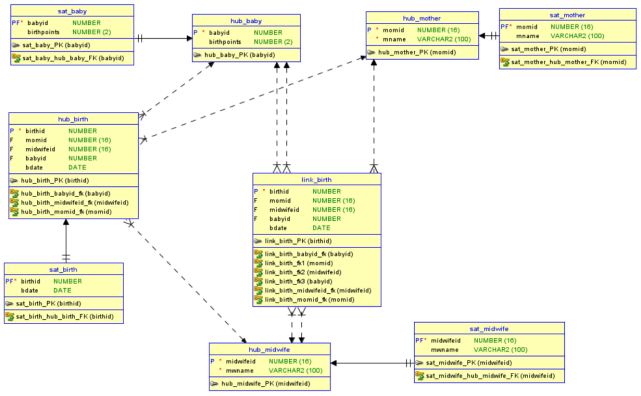

Fig. 2. The Birth model as Data Vault 2.0 generated by ChatGPT

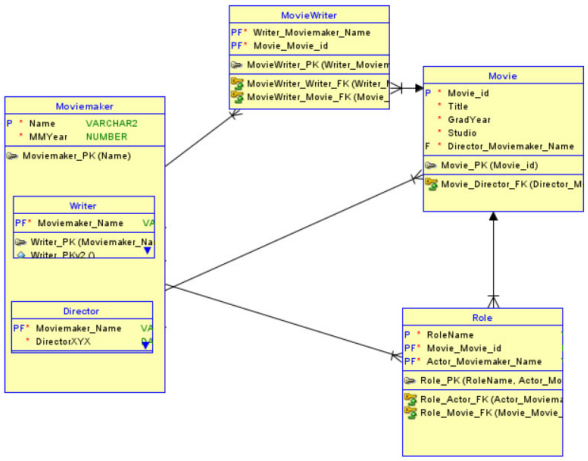

Fig. 3. Source database for a Moviemaker

Upon reviewing the model's visual depiction, it becomes apparent that ChatGPT's performance was suboptimal. Our evaluation, conducted through the designated metrics, is documented in Table V, showcasing the respective values. The Birth data model accrued a score of 12 points. The data model quality is considered unsatisfactory for scores below 17 [7]. The model fails in attributes, primary keys (PKs) and foreign keys (FKs). The primary keys in Satellites have been misdefined preventing the history data to be stored. Hubs and Satellites are missing technical attributes required by the methodology and some foreign keys are missing. Also, several attributes have been defined as non-mandatory.

TABLE V. METRICS AND THEIR VALUES FOR THE BIRTH MODEL

No	Metric	Value
1	CDTSHS	1
2	RoT1	1
3	RoT2	1
4	RPK	1
5	MaxD	1
6	RPKH	1
7	RPKL	1
8	RPKS	0
9	NoFKH	1
10	RFKS	1
11	RFKL	1
12	RAH	0
13	RAL	1
14	RAS	0
15	RMPKA	1
16	RMFKA	0
17	RMAH	0
18	RMAL	0
	TOTAL	**12**

Then, we tested a MovieMaker model with four tables: MovieMaker, MovieWriter, Movie, Role, and two sub-entities for a MovieMaker(Writer, Director). This data model is shown in Fig. 3.

The generated Data Vault 2.0 model is shown in Fig. 4.

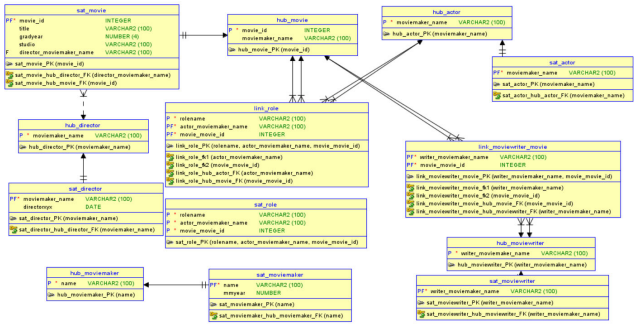

Fig. 4. Data Vault model for the Moviemaker data source

The metrics and their values for the Moviemaker data model can be seen in Table VI. The total score for the Moviemaker data model is 9. The score indicates very poor quality for the model. The model fails in several ways. The most visible flaw is a strange relationship between the SatMovie and HubDirector, the wrongly defined PKs for Satellites, several FKs to Links, and missing Hubs and relationships. The evaluation shows flaws in missing Hubs, the number of PK attributes in Link and Satellite tables is wrong as well as the number of FK attributes in Satellites. The number of technical attributes required by the methodology are wrong in Hubs, Links and Satellites. Also, the number of mandatory FK attributes in general is too low as well as mandatory attributes in Hubs.

TABLE VI. METRICS AND THEIR VALUES FOR THE MOVIEMAKER MODEL

No	Metric	Value
1	CDTSHS	0
2	RoT1	1
3	RoT2	1
4	RPK	1
5	MaxD	1
6	RPKH	1
7	RPKL	0
8	RPKS	0
9	NoFKH	1
10	RFKS	0

11	RFKL	1
12	RAH	0
13	RAL	0
14	RAS	0
15	RMPKA	1
16	RMFKA	0
17	RMAH	0
18	RMAL	1
	TOTAL	**9**

We also tested the Orders model [7]. The source model can be seen in Fig. 5. The model consists of four tables: Customers, Orders, Orderlines and Product. The inconsistency of the naming is on purpose, since in real life the models are of different quality and the naming conventions are often forgotten.

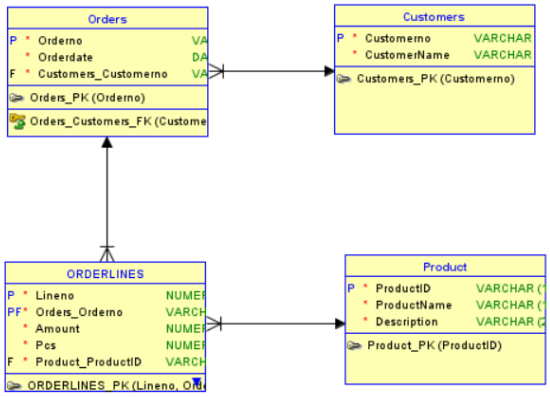

Fig. 5. Order source database

In Fig. 6, you can find the generated Data Vault 2.0 data model. We removed the double FKs of the Link table from the DDL generated by ChatGPT.

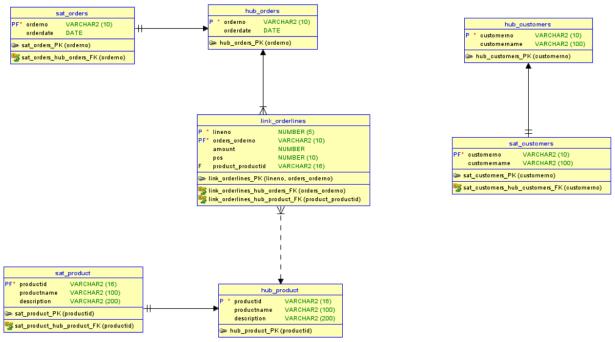

Fig. 6. Data Vault 2.0 data model for Orders model generated by ChatGPT

The metrics and their values for the Order model can be seen in Table VII. The total score for the model is 10. The model is missing tables and FKs and the FKs in Satellites are defined wrong. The evaluation shows missing Hubs since the dependent child concept has been solved wrongly. This can be seen in metric CDTSHS's else clause where the number of Satellites should be greater than the number of Hubs, which in this case is untrue. The number of PK attributes in Links and Satellites is incorrect. Also, the number of attributes in Hubs and Satellites is incorrect. The number of mandatory FK attributes is incorrect in general. The number of mandatory attributes in Hubs and Links is incorrect.

TABLE VII. METRICS AND THEIR VALUES FOR THE ORDER MODEL

No	Metric	Value
1	CDTSHS	0
2	RoT1	1
3	RoT2	1
4	RPK	1
5	MaxD	1
6	RPKH	1
7	RPKL	0
8	RPKS	0
9	NoFKH	1
10	RFKS	1
11	RFKL	1
12	RAH	0
13	RAL	1
14	RAS	0
15	RMPKA	1
16	RMFKA	0
17	RMAH	0
18	RMAL	0
	TOTAL	**10**

The manually created Data Vault 2.0 data model for the Orders data source can be found in Fig. 7.

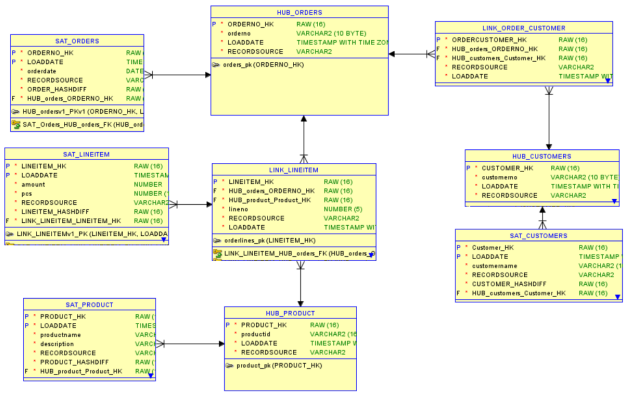

Fig. 7. Manually designed Data Vault 2.0 data model for the Orders model

In Table VIII, you can see the scores for the manually created Data Vault 2.0 data model for the Orders source database. This model scores 18, passing all the metrics.

TABLE VIII. METRICS AND THEIR VALUES FOR THE MANUALLY CREATED DATA VAULT 2.0 DATA MODEL FOR THE ORDER MODEL

No	Metric	Value
1	CDTSHS	1
2	RoT1	1
3	RoT2	1
4	RPK	1
5	MaxD	1
6	RPKH	1
7	RPKL	1
8	RPKS	1
9	NoFKH	1
10	RFKS	1
11	RFKL	1

12	RAH	1
13	RAL	1
14	RAS	1
15	RMPKA	1
16	RMFKA	1
17	RMAH	1
18	RMAL	1
	TOTAL	**18**

While evaluating the models with our scripts, we also evaluated them manually to verify the metrics and the scripts reflect the real quality of a model. The Data Vault 2.0 data models evaluated herein were found to exhibit suboptimal quality, both using the metrics and the evaluation of a human expert. This observation implies that the proficiency of ChatGPT in generating the Data Vault 2.0 model has declined relative to its previous state, resulting in a diminished capability to produce DDL statements of adequate quality. This observation underscores the necessity for easily applicable quality metrics to assess the quality of generated Data Vault 2.0 data models.

IV. Creating a Framework

In this section, we will create the framework for automatically obtaining the quality evaluation for a Data Vault 2.0 data model. Finally, we will create an application for the framework.

A. Setting up the environment

We inserted all the necessary data for quality assessment into an Oracle Database. We then retrieved the required measures by querying the data dictionary. Following this, we utilized the obtained measures to perform the calculations necessary for deriving the metrics. Details regarding measures and metrics, along with the corresponding data for each data model, is stored in dedicated tables described in Fig 8. Tables *Measures* and *Metrics* include the general data about them. Tables *MeasureValues* and *MetricValues* include data about each individual model evaluated.

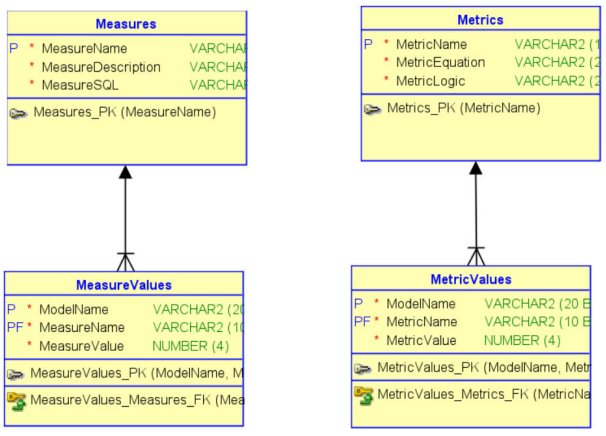

Fig. 8. Tables for measures and metrics in DV schema

First, we created three database schemas with the needed content:

- DV schema stores the measures (*Measures*) and metrics (*Metrics*) used for scoring the quality. The values of measures by model are stored in the *MeasureValues* table and the values of metrics in the *MetricValues* table. The tables needed are shown in Fig 8. The process is explained in Algorithm 1.
- SOURCE schema holds the original source data model. The process is explained in Algorithm 2.
- DVDW schema holds the Data Vault 2.0 data model objects that will be evaluated. The process is explained in Algorithm 2.

Algorithm 1 Setting up the environment, create DV schema

0: Connect to the database using *admin* privileges
1: **If** *DV* user does not exist
2: **then** create *DV* user with privileges needed
3: **end if**
4: **If** *DV* user does not have table *MEASURES*
5: **then** create table *MEASURES* and insert *values*
6: **end if**
7: **If** *DV* user does not have table *METRICS*
8: **then** create table *METRICS* and insert *values*
9: **end if**
10: **If** *DV* user does not have table *MEASUREVALUES*
11: **then** create table *MEASUREVALUES*
12: **end if**
13: **If** *DV* user does not have table *METRICVALUES*
14: **then** create table *METRICVALUES*
15: **end if**

Algorithm 2 Setting up the environment, SOURCE and DVDW

0: Connect to the database using *admin* privileges
1: **If** *SOURCE* user exists
2: **then** drop all *tables* owned by *SOURCE* and
3: purge *recyclebin*
4: **else**
5: create user *SOURCE*
6: with privileges needed
7: **end if**
8: **If** *DVDW* exists
9: **then** drop all *tables* owned by *DVDW* and purge *recyclebin*
9: **else**
10: create user *DVDW*
11: with privileges needed
12: **end if**

B. *Using the framework*

When the environment was created, we executed the DDLs in the database. The source database DDLs were executed as explained in Algorithm 3 and the Data Vault 2.0 DDLs as explained in Algorithm 4.

Algorithm 3 Insert data (DDLs) for SOURCE

0: Connect to the database using *SOURCE* credentials
1: Execute *DDLs* for the source data model

Algorithm 4 Insert data (DDLs) for DVDW

0: Connect to the database using *DVDW* credentials
1: Execute *DDLs* for the data model under evaluation

Now, all the information regarding database objects is accessible within the database for further investigation. First, we obtained measures, as explained in Algorithm 5, then we calculated the metrics based on the measures, as explained in Algorithm 6.

Algorithm 5 Obtaining Measures

0: Connect to the database using *DV* credentials
1: Define a unique name for the *model*
2: **For** all measures in *MEASURES* **do**
3: select a *measure name* from the MEASURES table
4: execute the *SQL clause* for the measure
5: insert the *model's name,* the *measure name* and the *SQL-query result* to *DV.MeasureValues*
6: **end for**

Algorithm 6 Obtaining Metrics

0: Connect to the database using *DV* credentials
1: **For** all metrics in *METRICS* **do**
2: select a *metric name* from the table
3: execute the *Logic clause* for the metric
4: insert the *model's name,* the *metric name* and the *Logic query result* to *DV.MetricValues*
5: **end for**

The total score for each model can be calculated from the *Metricvalues* table as shown in Algorithm 7.

Algorithm 7 Obtaining Total Score

0: Connect to the database using *DV* credentials
1: Set TotalScore := 0;
2: **For** all metrics in *METRICVALUES* where *modelname* = :Model **do**
3: **select** *metric value*
4: *TotalScore*:= *TotalScore*+*metric value*
5: **end for**
6: Printout *TotalScore*

C. *Introducing the application*

In order to automate the process, we developed an APEX [9] application. The application consists of a user interface for inserting the source database DDL, the model's name and the DDL for creating a Data Vault 2.0 database. Every time a new source or target DDL is inserted, the database objects created by a previous DDL are deleted from the database. Using this technique, we are able to query the data catalog views without the previous experiments interfering with the results.

When a DDL has been uploaded, a PL/SQL package executes the DDL in Source or DVDW schema, depending which one has been chosen, and calculates the measures and metrics using the data in the data dictionary. Finally, the application shows the results in a user interface. In Fig. 9, the menu of the application is shown. It consists of five functionalities: inserting the source DDLs, inserting the target (Data Vault 2.0 model to be evaluated) DDL, measure and metric values for the data models, and administration functionalities. The measures and metrics for each experiment are stored in the database using the model's name defined by a user. These measures and metrics can be seen and compared if needed. The user interface also includes a functionality for updating the measure and metric definitions if they need to be changed or fine-tuned. The application uses the logic defined in database tables; the logic is not hardcoded.

The process starts with the DDLs of the source data model as shown in Fig 9. The name of the source database schema and its DDLs are inserted. By pressing the "Run DDL" button the schema objects of the source schema are created in the database. By separating the source database DDL management from the target DDL management. we allow experimenting with several target models against one source model.

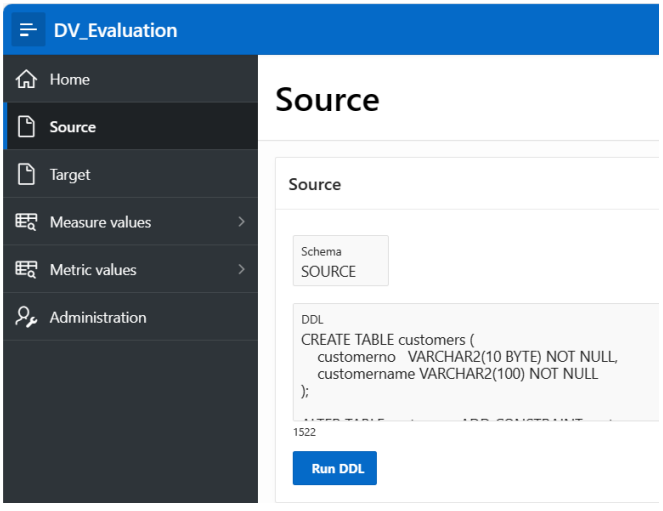

Fig. 9. The application menu and the database object creation for the source data model

Then, by selecting "Target" from the menu and inserting the schema name, the model a name, and the DDLs of the Data Vault 2.0 data model evaluated, as shown in Fig. 10, and pressing the "Run DDL" button, the data model objects are

created to the target schema defined. To calculate the measures for the data model "Measures" button is pressed. And, to calculate the metrics, the "Metrics" button is pressed.

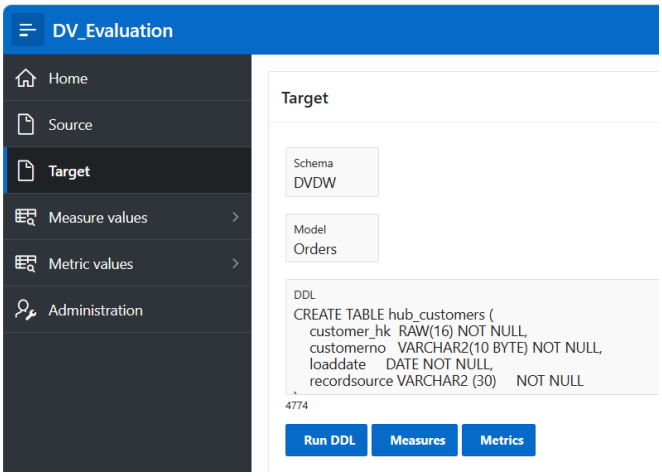

Fig. 10. Creating the Data Vault 2.0 model objects to the database and calculating the measures and metrics

When the measures and metrics have been calculated, they can be seen by selecting "Measure values" or "Metric values" from the menu. Measure values for the Orders model are shown in Fig. 11, and the Metric values in Fig. 12.

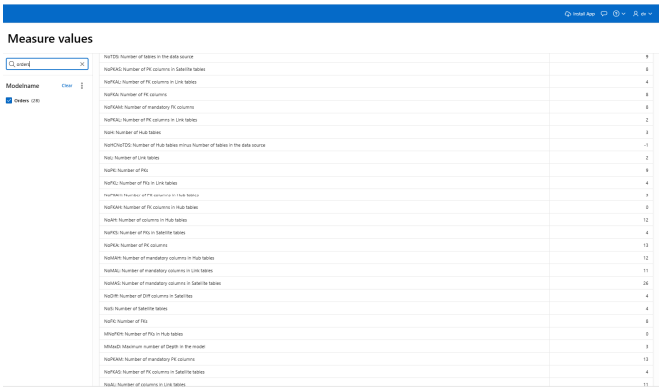

Fig. 11. Measure values for the Orders data model

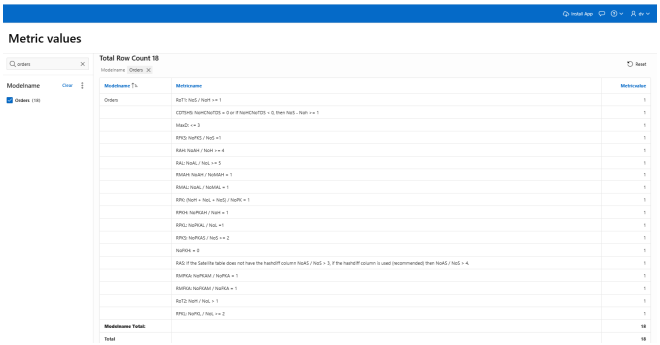

Fig. 12. Metric values and the total score for the Orders data model

V. Conclusion

The framework and application we created support the quality evaluation of a Data Vault 2.0 model using metrics defined in our previous work [7]. The evaluation process requires the DDLs of the source database and the DDLs for the Data Vault 2.0 database under evaluation. In our tests we used ChatGPT 3.5 to generate the Data Vault 2.0 DDLs based on the source data model DDLs.

While the automation of the quality measuring functions as intended, the data models' quality does not meet expectations. Regrettably, none of the generated DDLs met the standard required for utilization as a Data Vault 2.0 database. The future work should include tools and techniques for improving the quality of the generated DDLs. It could, for example, use prompt engineering to improve the quality of the Generative AI generated DDLs by adding Data Vault 2.0 governance to the process. For example, generating correct business key definitions instead of assuming the primary key is always the business key. Additionally, Retrieval-Augmented Generation (RAG) could facilitate the introduction of novel metrics, for example, via naming conventions, or it could, for example, create the possibility of adding knowledge about the hash rules to the generation process.

For future research, it would also be valuable to investigate the metrics when the DW already exists and new data sources are added. It would be interesting to see if the same metrics still apply.

References

[1] Data Vault Alliance official website, Data Vault 2.0 Data Modeling Specification v2.0.4, Web: https://datavaultalliance.com/news/data-vault-2-0-data-modeling-specification-v2-0-4/ (accessed on 28 02 2024).

[2] D. Linstedt, and M. Olschimke, *Building a scalable data warehouse with Data Vault 2.0*. Morgan Kaufmann, 2015.

[3] W.H. Inmon, and D. Linstedt, *Data architecture: a primer for the data scientist: big data, data warehouse and data vault*. Chapters 4.1-4.5. Morgan Kaufmann, 2015.

[4] W.H. Inmon, D. Linstedt, and M. Levins, *Data Architecture: A Primer for the Data Scientist: A Primer for the Data Scientist. 2nd Edition*. Chapters 6.1-6.5. Academic Press, 2019

[5] H. Helskyaho, "Towards Automating Database Designing". 34th Conference of Open Innovations Association (FRUCT) 2023, pp. 41-48

[6] V.C. Storey, C.B. Thompson, and S. Ram. "Understanding database design expertise", Data & Knowledge Engineering, 16(2), 1995. pp.97-124.

[7] H. Helskyaho, L. Ruotsalainen, and T. Männistö, "Defining Data Model Quality Metrics for Data Vault 2.0 Model Evaluation", Inventions, 9(1), 2024, p.21.

[8] Oracle Database PL/SQL Language Reference, Web: https://docs.oracle.com/en/database/oracle/oracle-database/23/lnpls/preface.html#GUID-9952D28D-0310-4869-8002-4462BDC679BB (accessed on 28 02 2024).

[9] Official APEX website, Web: https://apex.oracle.com/en/ (accessed on 28 02 2024).