Engineering and Applied Science Research
https://www.tci-thaijo.org/index.php/easr/index
Published by the Faculty of Engineering, Khon Kaen University, Thailand

A semantic approach to automated design and construction of star schemas

Non Sanprasit[1)], Taravichet Titijaroonroj[2)], and Kraisak Kesorn*[1)]

[1)]Department of Computer Science and Information Technology, Faculty of Science, Naresuan University, Phitsanulok 65000, Thailand
[2)]Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand

Received 20 December 2020
Revised 7 February 2021
Accepted 15 February 2021

Abstract

Designing a star schema is a complex and time-consuming process requiring an expert to perform several tasks such as denormalization, dimension design, and construction of fact tables. This study presents a method to automatically design and generate star schema models, or so-called multidimensional models. We first introduce a method to incorporate a novel knowledge-based framework to enable an automation system to construct dimensional and fact tables as well as measures, which are the key elements of star schema models. The proposed framework provides a capability of column name identification using the arithmetic coding approach and measures identification using a natural language processing framework (NLP), resulting in dimensions and fact tables being constructed automatically without human intervention. Although the current version of our system is limited to reading data from semi-structured datasets such as CSV files and spreadsheets, the experimental results demonstrate that our framework can generate a star schema effectively, and can support online analytical processing (OLAP) operations. The experimental results show that our method is superior to other conventional approaches, achieving 96.67% accuracy for numerical data, higher than any of the prior models used for comparison.

Keywords: Semantic approach, Knowledge-based, Star schema, Data warehouse

1. Introduction

Several schemas are used in a data warehouse (DW), such as a star and snowflake schema, to enhance the analysis of business transactions. This research focuses only on star schema, which is more popular among DW designers, with a less complicated structure compared to the snowflake model. The star schema model facilitates OLAP operations to assist policymakers in analyzing business data efficiently, whereas the traditional database (DB) system does not allow such a task to be performed, or has great difficulty.

Typically, DW design is performed manually by experienced DW designers, which is time-consuming and laborious because clients generally cannot specify their requirements precisely. Having obtained users' requirements, these experts will follow DW design models such as a top-down approach [1], in which the designer records users' needs and designs the DW schema to meet those user requirements. Therefore, this method is also known as user-driven [2]. In contrast, a bottom-up technique [3] identifies the requirements through observation the business processes, the so-called data-driven [2] approach. The main limitation of this approach is that it tends to lead to high risk of design failure because it cannot identify real business needs accurately without the involvement of users. Designing a DW is even more difficult if designers lack knowledge and understanding of the relationships among the data in the database at hand, resulting in a valid star schema modeling task being overwhelming or a non-expert or even an expert. Hence, the design of star models requires specialized techniques [4].

Our main objective is to overcome these problems by presenting a novel framework automating the design and generation of a star schema based on knowledge stored in a semantic model. This approach supports the necessary and often complex data analysis and prediction processes. Therefore, we hypothesize that an ontology model could facilitate the DW management system in designing a star schema depending on a given context. The existing semantics of the data contents are encoded in a knowledge-based representation in the web ontology language (OWL) format. This can provide crucial context knowledge essential for supporting dimensions, facts, and measures identification in multidimensional model construction for further use in OLAP operations as well as data prediction and information extraction for the variety of perspectives required.

2. Literature review

Data are often collected and stored in a semi-structured format, such as CSV and XML. These data are very useful, but their structure may be less suitable for further processing. As such, these data need to be transformed into some well-structured forms e.g., DB or DW forms, before they can be used to support decision making. In the past few years, several researchers have invested effort to create a framework to automate the generation of star schema from semi-structured data. Hansen et al. [5] proposed an approach to

*Corresponding author. Tel.: +66 5596 3267
Email address: kraisakk@nu.ac.th
doi: 10.14456/easr.2021.54

construct a star schema based on a CSV file, which is a very challenging task because of the lack of data types as well as relationships specified explicitly in the dataset. Therefore, the proposed system of Hansen et al. [5] must automatically identify the appropriate data types in the table and the relationships among the tables in the DW. The main limitation of this work is that the capability of detecting column names and data types is not sufficiently accurate. Given the limitation of CSV file which cannot embed data types within the text, some research has used XML files, allowing data types to be identified in tags. The main advantage of XML is that it is universally accessible to any software that comprehends the markup language.

Extracting user requirements is very important for the requirement-driven approaches. Typically, user requirements are collected using traditional methods such as interviewing. Recently, Bimonte et al. [6] proposed the use of a pivot table to assist in analysis and improve understanding of user requirements. The main limitation of this approach is that the requirements were not acquired automatically, and the DW schema is still constructed manually, which is a laborious task. Moukhi et al. [7] presented a hybrid method to construct a DW by integrating the data-driven method X-ETL and the requirement-driven method XCube Assist to extract user requirements in the form of natural language. However, the main limitation of this approach is that those requirements must be represented in the defined format; otherwise, the system cannot process them and is unable correctly to generate a multidimensional schema.

An ontology model was first used in DW design in 2009 by Nebot et al. [8], who integrated an ontology model into the framework to convert instance data into cubes for data analytic tasks. The presented framework is called "a multidimensional integrated ontology (MIO)". There are some limitations to this framework in that it does not support DW evolution in the future. Later, Nebot and Berlanga [9] presented a framework for constructing a DW from a semantic web in which data is represented in resource description framework (RDF) and OWL in the biomedical domain, which partially supports OLAP operations such as aggregation and navigation. However, it does not allow users to perform drill-down and roll-up operations because it cannot construct dimensional "*hierarchies*" or "*levels*." Liu and Iftikhar [10] proposed an algorithm to design a DW with high dimensionality using an ontology model describing business requirements. The data are partitioned both horizontally and vertically, which helps to reduce the response times to queries. In addition, the method is flexible and can support changing requirements by regenerating the schema using the tool. However, this work only supports a fixed framework of ontology data representation.

Table 1 Limitations of existing frameworks

	Identification capability							**Aggregation, navigation**	**Heterogeneous terminology problems**
Methods	**Column name**	**Data type**	**Measure**	**Dimension tables**	**Fact tables**	**Data hierarchy**	**Table relationships**		
Nebot et al. [8]	✓	✓	-	✓	✓	-	✓	-	-
Later, Nebot and Berlanga [9]	✓	✓	-	✓	✓	-	✓	✓	-
Khouri et al. [11]	✓	-	-	✓	✓	-	✓	✓	-
Liu and Iftikhar [10]	✓	-	-	✓	✓	-	✓	✓	-
Hansen et al. [5]	✓	✓	-	✓	✓	-	✓	✓	-
Bimonte et al.[6]	✓	-	-	✓	✓	-	✓	✓	-
Moukhi et al. [7]	✓	-	-	✓	✓	-	✓	✓	-

Note: - = does not support, ✓ = supports.

All the works mentioned above used a single ontology model, which is not adequate in some cases. Therefore, a method to construct a DW design from multiple ontologies has been proposed by Khouri et al. [11] by aggregating ontologies from these heterogeneous sources into a database, with the DW being generated from those ontologies, resulting in a system supporting a larger-scale enterprise. However, both frameworks from Khouri et al. [11] and Liu & Iftikhar [10] still lack the capability to infer attribute names, data types, or measures. In addition, they cannot manage data ambiguity problems such as synonyms, polysemy, homonyms, misspellings, abbreviations, vocabulary, and terminology. Table 1 summarizes the limitations of the state-of-the-art methods surveyed.

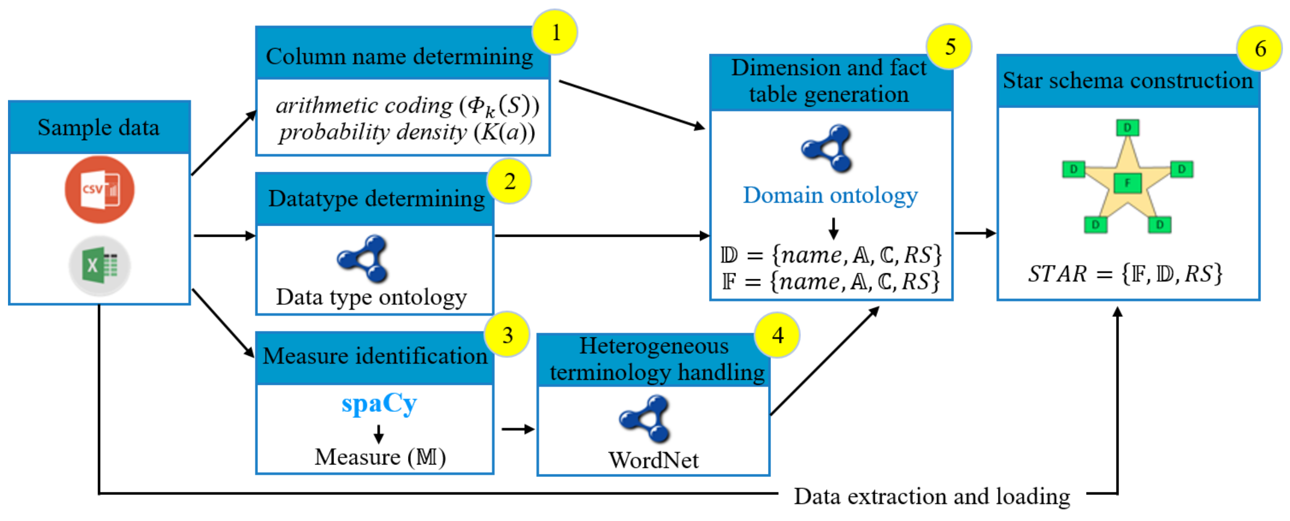

Sample data
CSV
Column name determining
1
arithmetic coding ($\Phi_k(S)$)
probability density ($K(a)$)
Datatype determining
2
Data type ontology
Measure identification
3
spaCy
Measure ($\mathbb{M}$)
Heterogeneous terminology handling
4
WordNet
Dimension and fact table generation
5
Domain ontology
$\mathbb{D} = \{name, \mathbb{A}, \mathbb{C}, RS\}$
$\mathbb{F} = \{name, \mathbb{A}, \mathbb{C}, RS\}$
Star schema construction
6
D
F
$STAR = \{\mathbb{F}, \mathbb{D}, RS\}$
Data extraction and loading

Figure 1 Overview of semantics-based star schema design frameworks

3. Semantic star schema design

This section provides an overview of the proposed framework, describing our processes for star schema construction, as shown in Figure 1. Typically, a star model contains dimensions, facts, and measures. A collection of related business data will be stored in “fact” tables consisting of “measures” representing the quantitative information in the fact tables, providing a measure of the performance of the business relevant to the given dimensions. A dimension is a collection of data that describes one business perspective. The architecture of the proposed framework consists of seven basic components including 1) column name determination, 2) data type detection, 3) measure identification, 4) heterogeneous terminology handling, 5) dimension and fact table generation, and 6) star schema construction. These components enable the proposed framework to construct dimensions, facts, measures, and relationships among tables. Figure 1 shows the major components of this framework.

This section presents an overview of the proposed system, which imitates the reasoning process of human knowledge models in the form of an ontology model. Here, we present a new strategy to construct a star schema from a CSV or spreadsheet file. The proposed method is termed “*Semantic-based Star Schema Designer (SSSD)*.” The architecture of the framework consists of six components, as shown in Figure 1.

3.1 Column name determination

This module performs column naming using the data available in the data source file. Typically, the column names in the sample data are specified in the first row of the data source file. However, if column names are absent, they will be derived from the data in the column of the spreadsheet or CSV file, which is a relatively advanced technique for this application. To overcome this issue, the *SSSD* provides a module to facilitate column name prediction for both categorical and numerical data types associated with each column. We identify an appropriate column name using a technique named “*arithmetic coding*” introduced by Witten et al. [12, 13]. Some researchers suggested the use of “*regular expression (regex)*” techniques determining the column name based on a particular form e.g., postal code, phone number, or email address. However, most column names are typically generic and do not have a particular canonical form. Consequently, the *regex* technique may not work effectively in this case. Arithmetic coding is commonly used to transform nominal data into numbers between 0 and 1, called *codes.*

Primarily, the arithmetic coding process generates a sequence of nested intervals in the form shown in (1).

$$\Phi_k(S) = |b, l\rangle_{k=0,1,2,\ldots N} \tag{1}$$

where S is the string, b is called the starting point of the interval and l is the length of the interval where $b, l \in \mathbb{R}$, $\mathbb{R}$ represents real number. The intervals used during the arithmetic coding process are defined by a set of recursive equations [14] shown in (2) and (3).

$$\Phi_k(S_k) = \begin{cases} |0,1\rangle, & k=1 \\ |b_k, l_k\rangle = |b_{k-1} + c(S_k)l_{k-1}, p(S_k)l_{k-1}\rangle, & k>1 \end{cases} \tag{2}$$

$$\hat{\gamma}(S) = \frac{b_N + l_N}{2} \tag{3}$$

where c and p are the distribution and probability of the string of a sentence, respectively. The properties of the intervals guarantee that $0 \le b_k \le b_{k+1} < 1$ and $0 \le l_{k-1} < l_k < 1$. Finally, arithmetic coding defines a code value $\hat{\gamma}(S)$ that represents string S.

To derive attribute names of the numerical from the existing data, the “*estimated probability density function*” is also deployed to predict the column name. The probability density technique learns data patterns from training data and can then predict the column name of an *unknown* attribute.

Our main hypothesis is that the column names need to be a finite set, and each attribute should have a unique probability density. To compute the probability density estimation ($\mathcal{PD}$), these codes ($\hat{\gamma}$) are later processed by the estimated probability density function to extract the pattern of the probability density to discriminate all attributes. The probability density function is widely used for such a task to estimate the distribution of data and is defined as shown in (4).

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{4}$$

where $x_1, x_2, \ldots, x_n$ are the values of each attribute, n is the sample size (number of records), h is the bandwidth, and $K(\cdot)$ is the probability density function of the normal distribution, as computed in (5).

$$K(a) = \frac{1}{\sigma\sqrt{2\pi}} a^{-\frac{(a-\mu)^2}{2\sigma^2}} \tag{5}$$

where a is the input of $K(\cdot)$ in (5), σ is the standard deviation and μ is the mean of x. Bowman and his colleague [15] presented a bandwidth (h) estimation technique from the sample size by using (6).

$$h = \sigma \times \left(\frac{4}{3\times n}\right)^{\frac{1}{5}} \tag{6}$$

where σ and n are similar variables in (4) and (5).

This technique works effectively for any generic attribute regardless of the data type or the order of the attributes, and can resolve the problems addressed previously. The similarity score between objects p and q can be computed using the Euclidean distance, as shown in (7).

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{7}$$

3.2 Datatype determination

It remains a very challenging task to identify the most appropriate data size and type of an attribute because this information does not exist in unstructured or semi-structured data sources. To examine the attribute's data type and size, we take inspiration from [5, 16]. The hierarchical structure of an ontology model is exploited to determine the suitable data type and size of an attribute by traversing through the ontology structure. The lowest valid node found in the ontology model is the appropriate data type for the current attribute. This ontology is named as "*Data Type Ontology (DTO)*" as shown in Figure 2. To clarify, we demonstrate this process using the following example. Consider the value in the attribute entitled "Cases" with the value of '30'. The search method for this attribute is as follows. First, the *SSSD* considers "30" as numeric data, so the Numeric path is chosen and traversed. Since "30" is an Integer, the 'Integer' hierarchy is traversed, stopping at the most likely specific type. The value is then parsed in the hierarchy as a BigInt, to an Int, to a SmallInt, and finally to a TinyInt, which is the lowest node. Therefore, TinyInt would be considered an appropriate data type. If data is not numeric e.g. 'Fred', it is tested as a DateTime, which fails. Then, it is tested as a Boolean, which fails, and is then tested in the string. As the value is being 'data typed' in the ontology, the data type must be appropriate to all values in the data column, so pre-processing to find the highest value is necessary. Strings are restricted only by length, which creates no problems. In the special case of multiple data types being found in a column, the most fitting data type such as char, int, real that can encompass all values in the column will be chosen.

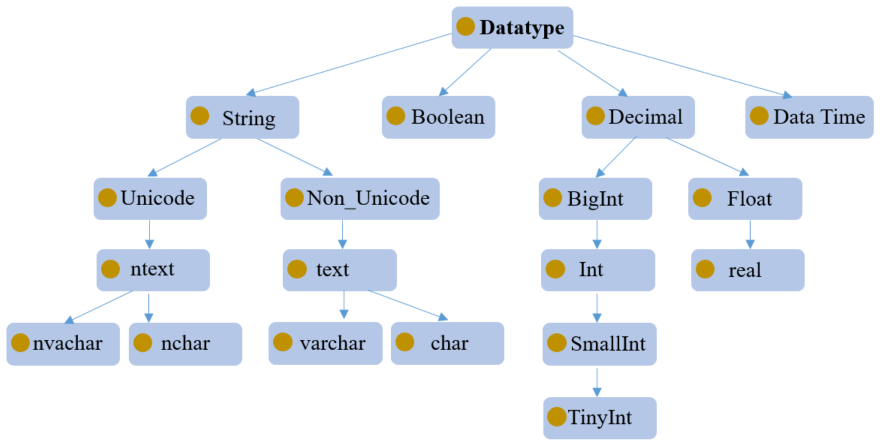


Figure 2 An example of a data type ontology (DTO) where the higher node is the more general data type the lower node and the lowest level being the most specific

3.3 Measure identification

User business requirements are the key information that can be used to support decision-making processes. This information is also required for a star schema, particularly a fact table. The user business requirements are often expressed in natural language in the form of queries such as "*Summarize the dengue cases by time and location.*" We can extract "*measure*" and "*dimension*" from the query and obtain measure is "*dengue cases*" and dimensions are "*time*" and "*location*." Here we define a measure as follows.

Definition 1. Numeric attributes that represents business dimensions are referred to as measures ($\mathbb{M}$). Typically, they are non-key attributes and can be validly aggregated across all dimensions. Equation (8) defines this measure.

$$\mathbb{M} = \left| \begin{Bmatrix} a_i | a_i \in \mathbb{A} \wedge a_i.dtp \in \mathbb{R}^n \wedge a_i \; is \; non-key \wedge a_i \in \{\mathbb{M}\} \\ \wedge \; a_i \; can \; validily \; be \; aggregated \; across \; all \; dimensions \end{Bmatrix} \right| \tag{8}$$

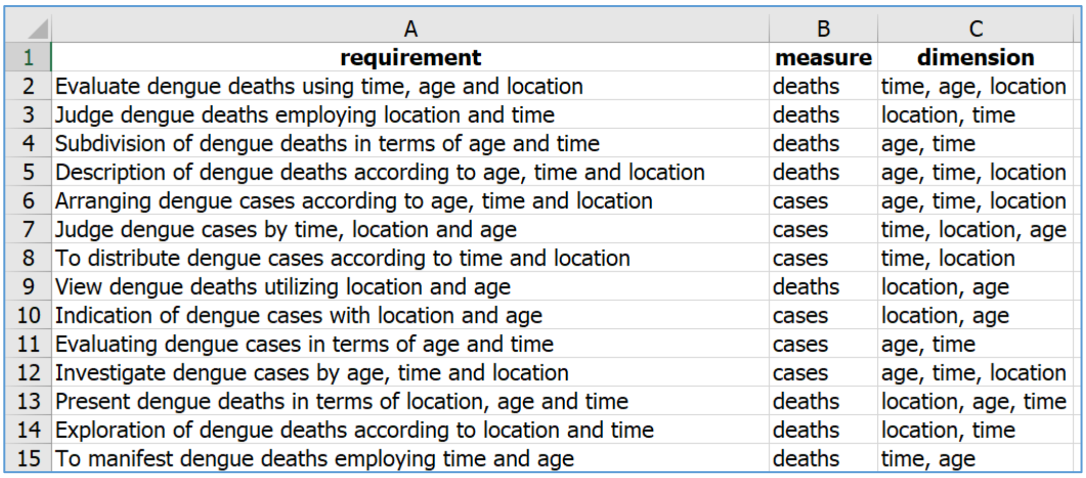
where a_i is an attribute ($\mathbb{A}$) in the sample data, *dtp* is the data type of a_i $i \in \{1, 2, \dots, n\}$, and $\{\mathbb{M}\}$ is a list of measures.

To extract measure from user business requirements, we deploy an off-the-shelf named entity recognizer (NER) framework called "*SpaCy*" pipeline which is the open source library for natural language processing (NLP) in Python used for several tasks such as POS Tagging, Named Entity Recognition, Dependency Parsing [17]. The implementation used a transition-based framework [18]. A named entity refers to a "*real-world object*" which is assigned a name, so called "*annotation*," for example, a location, a product, or a person. SpaCy can recognize various types of named entities in a sentence by training a prediction model using a sample dataset, and it uses statistics and a machine learning model requiring parameter tuning. SpaCy processes the business requirements, as shown in Algorithm 1. When the training dataset is input into Spacy, each query is tokenized ti the system begins the NER process. To train Spacy, we provide a training dataset containing business requirements with annotated data specifying "*measure*" and "*dimension*", as shown in Figure 3. The SpaCy pipeline works based on a deep learning algorithm using a convolutional neural network (CNN) and long short-term memory (LSTM) architectures to learn patterns of each requirement and identify entity labels. Each label is used as an input for the next iteration. Spacy processes all training datasets and outputs annotated data in the format required to train the SpaCy models. This step is repeated 30 times for each entity label to obtain the final output. This is an optimal number to train SpaCy; higher values do not improve accuracy. Finally, we used this model to process the *unknown* data called the testing dataset. After the measures and dimensions were extracted, we use them as terms to identify measure and dimension columns in the dataset and later use them for fact and dimension table construction.

Algorithm 1 User business requirements processing using SpaCy

```
//Training Data
Input :Training Data (TR)
Output :Trained Model
Read all requirements {r}
create an empty model
Add the new entity label
FOR each requirement ($r_i$) where $r_i \in TR$ and i $\leq$ 30
    call the spacy's minibatch
      if prediction was wrong
     adjusts weights
     updating label
  end if
END FOR
save Trained Model

//Test Data
Input :Test Data (TS)
Output :A list of possible measures {M} and dimensions{D}
Read all requirements {r}
FOR each requirement ($r_i$) where $r_i \in TS$
        Call prediction
        Add m to a list of measures{ML}
        Add dim to a list of dimensions {DL}
END FOR
RETURN {M}, {D}
```



	A	B	C
1	**requirement**	**measure**	**dimension**
2	Evaluate dengue deaths using time, age and location	deaths	time, age, location
3	Judge dengue deaths employing location and time	deaths	location, time
4	Subdivision of dengue deaths in terms of age and time	deaths	age, time
5	Description of dengue deaths according to age, time and location	deaths	age, time, location
6	Arranging dengue cases according to age, time and location	cases	age, time, location
7	Judge dengue cases by time, location and age	cases	time, location, age
8	To distribute dengue cases according to time and location	cases	time, location
9	View dengue deaths utilizing location and age	deaths	location, age
10	Indication of dengue cases with location and age	cases	location, age
11	Evaluating dengue cases in terms of age and time	cases	age, time
12	Investigate dengue cases by age, time and location	cases	age, time, location
13	Present dengue deaths in terms of location, age and time	deaths	location, age, time
14	Exploration of dengue deaths according to location and time	deaths	location, time
15	To manifest dengue deaths employing time and age	deaths	time, age

Figure 3 An example of business requirements as a training dataset for Spacy.

3.4 Heterogeneous terminology handling

When the data types are determined, the column names specified in the first row of the sample data are retrieved. However, heterogeneous terms from different data sources may be represented in the same attribute, and those terms could, therefore, vary from the terms in the knowledge base. Therefore, these terms must be disambiguated. We use WordNet [19] to solve the term vagueness problem of column names. With the assistance of *synsets* in WordNet, the system can match terms based on their semantics, rather than only simple string matching.

3.5 Dimension and fact table generation

Because information about primary keys, foreign keys, and relationships among data are not available in semi-structured data sources, our approach needs to provide this capability to construct a star schema with relationships between tables. Our approach differs from the existing methods in that a star schema is generated from the information in an ontology model, which is intended to be similar to human cognitive knowledge architectures. We apply sample data over the *domain ontology* and *data type ontology* to detect measures to form a fact table for analysis and derive related attributes to form dimension tables containing a hierarchy of levels representing different granularities of the sample data. This concept is exploited by multidimensional operators, such as drill-down and roll-up.

Dimension tables usually contain descriptive attributes, such as textual information or numerical data that behave like text. The advantage of using an ontology model is that it enables our *SSSD* to find semantic-related attributes that should be grouped into the same table by using relationships explicitly specified in the ontology structure. For instance, country, region, province, city, and district should be put together in a table because those attributes are semantically relevant: they are in the same path and under the same "*Location*" class in our domain ontology. In other words, the domain ontology can facilitate the normalization and denormalization of dimension tables as well as generate data hierarchies, both of which are expected to be useful in OLAP operations.

In addition, some surrogate key columns are inserted into dimension tables to enable foreign key connections to a fact table other than those based on the actual primary key of the dimensional tables. Also, some additional attributes are derived or calculated from available attributes, such as a date attribute (20/02/2020) being used to create a day name, day number of a week, or month name to facilitate data analysis regarding temporal aspects of ground truth. The result of this process is a system of logically associated dimension and fact tables.

Definition 2. A dimension table is a set of contextual data describing measures in a fact table from various perspectives. The dimension table ($\mathbb{D}$) is defined as follows in (9).

$$\mathbb{D} = \{name, \mathbb{A}, \mathbb{C}, \mathcal{RS}\} \tag{9}$$

where *name* is the name of a table derived from a class name in the domain ontology. Attribute ($\mathbb{A}$) is the set of columns within a table (ti), and t_i is a table in $\mathbb{D}$ where $t_i \in \mathbb{D}$. The attribute set of t_i is denoted as, $\mathbb{A}_i \subseteq t_i$ and ai is an attribute in $\mathbb{A}_i, a_i \in \mathbb{A}_i$, and a_i has a data type (dtp_i), where $dtp_i \in Types$ and *Types* is a set of all available data types. A dtp_i has a $length \in \mathbb{I}+$ which is the maximum number of characters that can be accepted as input or the maximum number of bytes in the case of numeric or binary data types. *Constraint* describes some characteristic of a_i *e.g.*, primary key and not null. *Relationship* ($\mathcal{RS}$) describes how a fact table relates to other dimension tables and is defined as follows in (10).

$$\mathcal{RS} = \{(a_{dim_1}, a_{fact}), \ldots, (a_{dim_n}, a_{fact}), cardinality\} \tag{10}$$

where a_{dim} and a_{fact} are the attributes of a dimension table and a fact table, respectively, and cardinality $\in\{one\text{-}to\text{-}many\}$. We assume $\mathbb{A}$ and $\mathbb{D} \in \phi$. That is, there exists at least one table, and each table contains at least two attributes (primary key and non-key attributes).

Definition 3. A fact table ($\mathbb{F}$) is the central table in a star schema storing business items or transactions, which are usually quantitative information, so-called measure, or analytical context data and are often denormalized. The fact table is defined as (11):

$$\mathbb{F} = \{name, \mathbb{A}, \mathbb{C}, \mathcal{RS}\} \tag{11}$$

where *name*, *constraint*, and *relationship* definitions are similar to $\mathbb{D}$ but *attributes* ($\mathbb{A}$) of $\mathbb{F}$ is the set of columns within the table (ti), $t_i \in \mathbb{F}$. The attribute set of t_i is denoted as, $\mathbb{A}_i \subseteq t_i$ and a_i is an attribute in $\mathbb{A}_i$ and $a_i \in \mathbb{A}_i$. There are two different types of a_i: foreign keys (FK) and measures. An a_i has a data type (dtp_i), where $dtp_i \in Types$ and *Types* is a set of all available data types. A dtp_i of a measure can only be numeric data type and thus a value v_i where $v_i \in V_i$ and $V_i \in \mathbb{R}$ and $length \in \mathbb{I}^+$.

3.6 Star schema construction

To generate a star schema from the dimension and fact tables from sample data, which usually contain several attributes ($\mathbb{A}$), the condition $\mathbb{A} \neq \phi$ is required. It is important to take into account the user requirements (business keys) to design multidimensional structures.

Definition 4. A star schema contains fact tables ($\mathbb{F}$) and dimension tables ($\mathbb{D}$) that are connected together via foreign keys through $\mathcal{RS}$ which describes how $\mathbb{D}$ and $\mathbb{F}$ are connected to each other. We formalize the generated star schema $\mathcal{STAR}$ as a collection of tables and attributes defined in (12).

$$\mathcal{STAR} \equiv \{\mathbb{F}, \mathbb{D}\ \mathcal{RS}\} \tag{12}$$

where $\mathbb{F}$ is a set of generated fact tables and $\mathbb{D}$ is a set of generated dimension tables. After the fact table generation, all measure attributes and primary keys from all dimension tables are added to the fact table to link to all dimension tables.

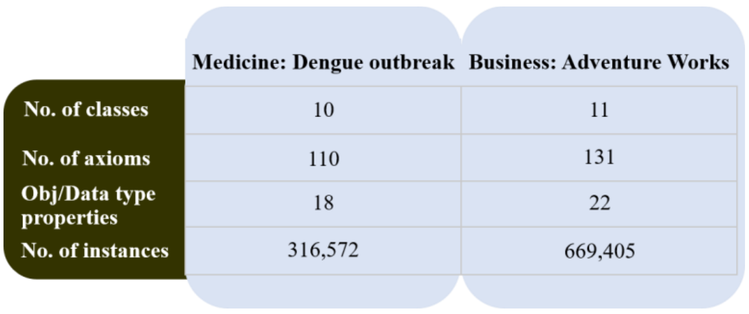
4. Experimental results and discussion

To validate our framework, two datasets from different domains were used to test *SSSD*. The datasets include Medicine (dengue information from https://ddc.moph.go.th and Vector Biology and Vector Borne Disease Research Unit, Chulalongkorn University) containing 13,764 records with 23 attributes and Business (AdventureWorksDW from https://docs.microsoft.com) containing 60,855 records with 15 attributes. The experiments were conducted to evaluate our hypotheses from three perspectives: 1) column name and data type inferencing performance, 2) measure identification performance, and 3) traditional approaches compared to our method (*SSSD*). Experiments and evaluations were set up to verify the *SSSD* performance, which was measured using an accuracy value and can be computed using (13).

$$Accuracy = \frac{\mathbb{P}}{\mathbb{Q}} \times 100 \tag{13}$$

where $\mathbb{P}$ is the amount of data collected and $\mathbb{Q}$ is the amount of data in the dataset.

The ontologies used in this research to assist the data warehouse construction were designed and constructed using Hozo [20]. All data are in OWL format. Figure 4 shows the details of the ontologies in the two domains.



	Medicine: Dengue outbreak	Business: Adventure Works
No. of classes	10	11
No. of axioms	110	131
Obj/Data type properties	18	22
No. of instances	316,572	669,405

Figure 4 Details of the two ontologies used in our research

4.1 Column name identification performance

This section evaluates the column name inferencing capability of *SSSD*. The main challenge of this task arises when the data have column names that are explicitly addressed. We used the framework shown in Figure 1 to handle this uncertainty using probability density ($\mathcal{PD}$). The sample dataset was separated into training and testing sets (70:30) with five-fold cross-validation to prevent biased results and avoid overfitting. Both categorical and numerical data were examined by $\mathcal{PD}$ and the results were compared with four other state-of-the-art methods: naïve Bayes (*NB*), decision tree (*DT*), logistic regression (*LR*), and an artificial neural network (*ANN*) optimized using grid search by two significant parameters: the hidden layers (3, 4, 5…, 7) and learning rate from [0.01, 0.02, 0.03, …, 0.07]. Figure 5 illustrates the accuracy of four different techniques for performing column name inferencing when these data were not available in the data source. The results have three sets: categorical, numerical, and average. Figure 5 shows that the accuracy of $\mathcal{PD}$ for the categorical data is the lowest, at 96.31% ,while other techniques obtain an accuracy higher than 98.00%. This is because each attribute contains significantly different values of data, which allows the machine learning approaches to easily discriminate the data values and obtain the attribute names with an inferencing accuracy higher than $\mathcal{PD}$.

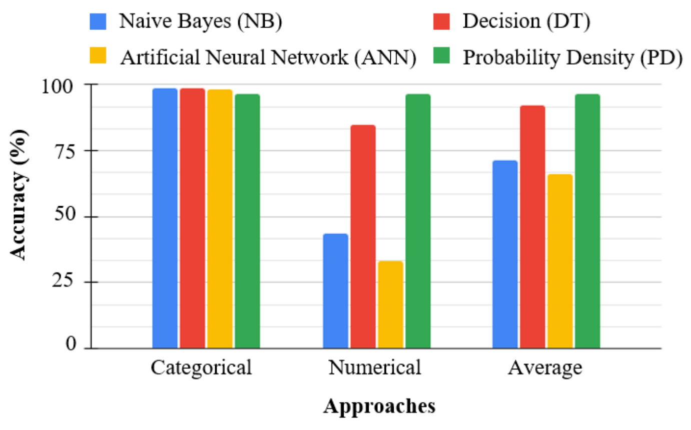


Figure 5 Comparison of the accuracy of traditional classification approaches, and the probability density technique

On the contrary, $\mathcal{PD}$ achieves the highest accuracy for numerical data at 96.67%, which is significantly higher than the accuracy of the other approaches, where 43.56%, 84.79%, and 33.33%, respectively. This is because these machine learning approaches do not work effectively with continuous values ($\mathbb{R}$), which are difficult to discriminate. Therefore, it is noticeable that the approach works more robustly than the baseline approaches for both types of data, whereas machine learning methods seems to work effectively only on nominal data. As a result, $\mathcal{PD}$ obtains the highest average accuracy of 96.49% compared to other approaches, as shown in Figure 5.

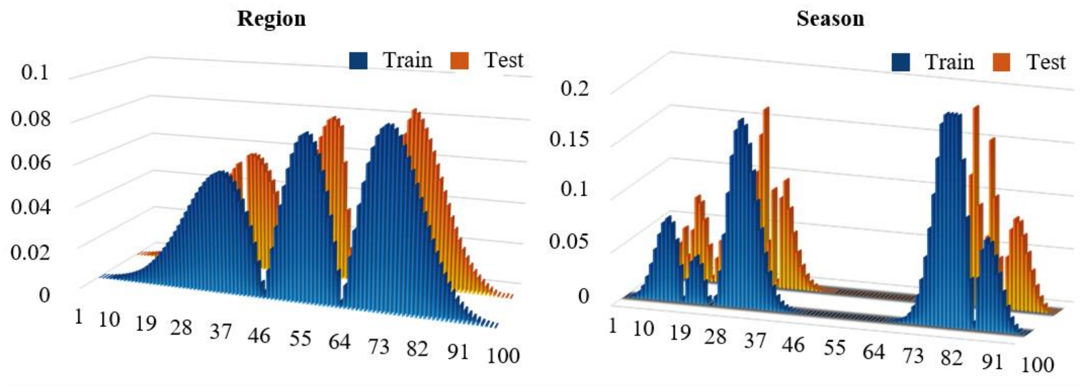


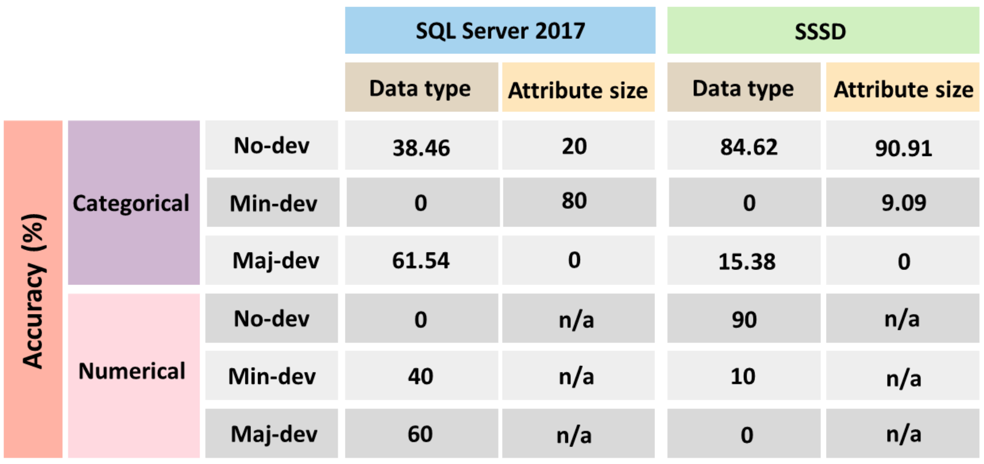
Figure 6 Some examples of probability density patterns of some attributes. The same column will have similar data pattern representing with line charts

The performance of the probability density to identify the column names is shown in Figure 6 (a-b). We note that the pattern of columns in both training and testing sets have similar density patterns, e.g., Region (Figure 6-a) and Season (Figure 6-b). In addition, the similarity of both attributes can be obtained using the Euclidean distance shown in Eq. (7), where the smallest score of Euclidean distance between two attributes indicates that they have the highest similarity and will be selected for the unknown attribute.

4.2 Data type inferencing performance

We compared the performance of data type inference of *SSSD* against a reputed and well-known off-the-shelf software, Microsoft SQL Server 2017, because it is a widely used software for DW construction available in the market. In this experiment, we used the dengue dataset, which included 23 attributes (13 nominal attributes and 10 numerical attributes). Our evaluation is based on three measures [5]: *no deviation, minor deviation, and major deviation* "*No deviation (No-dev)*" indicates that the predicted attribute type and size exactly matches the attribute that would be designed by an expert. "*Minor deviation (Min-dev)*" refers that the attribute type is mismatched between the system and the expert but has no negative effect on the result. "*Major deviation* (*Maj-dev*)" implies that the data type mismatch between the system and the expert has a significant negative effect on the result and is therefore very relevant and will significantly affect the results.

For categorical data, only 5 out of 13 (38.46%) were predicted correctly (*no deviation*) by SQL Server, as illustrated in Figure 7. In many cases, we found that most errors came from categorical attributes containing numerical data, resulting in SQL Server inferring them as floating point numbers.



			SQL Server 2017		SSSD	
			Data type	Attribute size	Data type	Attribute size
Accuracy (%)	Categorical	No-dev	38.46	20	84.62	90.91
		Min-dev	0	80	0	9.09
		Maj-dev	61.54	0	15.38	0
	Numerical	No-dev	0	n/a	90	n/a
		Min-dev	40	n/a	10	n/a
		Maj-dev	60	n/a	0	n/a

Figure 7 Accuracy comparison of datatype inferencing approaches

However, in this case, the varchar data type was more appropriate .Therefore, this case is considered as a major deviation, as were the majority of error with the SQL Server approach. *SSSD* outperforms SQL server with 84.62 %accuracy for data type prediction because it uses the data type ontology (*DTO*) which provides additional knowledge to efficiently determine the appropriate data types for each attribute .These results are consistent with the performance of the *SSSD* in correctly predicting the attribute size, which achieves 90.91% (*no deviation*) and 0 % for major deviation.

For numerical data type inferencing, the attribute data type was predicted correctly (*no deviation*) by the *SSSD* in 90% of cases, whereas the SQL Server predicts the incorrect data type in 60% (*major deviation*) because it always assigns Float (a default data type) to all numerical attributes, but, in fact, some of them should be TinyInt or varchar. *SSSD* uses the knowledge-based method to derive the most appropriate data type for each attribute. It is clear that the *SSSD* is more effective than the SQL Server, which does not have a knowledge base to facilitate the inference task.

Table 2 Accuracy comparison in measure and dimension identification using SpaCy in two different domains

Class	Accuracy(%)		
	Simple structure	Complex structure	Complex + heterogeneous terminology
Measures	100.00	100.00	100.00
Dimensions	100.00	94.73	66.67
Average	100.00	97.50	83.34

4.3 Measure identification performance

We intend to test the efficiency of the NLP tool SpaCy to determine the measures and dimensions from a user business requirement. In this experimental process we have 45,936 different business requirements forms that we have separated into three groups: (1) Simple structure, which refers to a requirement with a simple structure stated as a sentence e.g. "Analyze dengue cases by location, time and age," (2) Complex structure, which indicates a more complex requirement structure which has additional words in the sentence such as "Analyze top 5 serious dengue patients by different period, position and generation," and (3) Complex structure + heterogeneous terminology, which refers to a sentence that has had some words replaced with synonyms. This sentence structure is specifically defined to challenge the ability of SpaCy to identify and process synonyms.

Table 2 shows the performance of SpaCy for extracting measures and dimensions from user business keys. SpaCy achieved 100% accuracy in identifying measures and dimensions from the business requirements in the simple structure case. This is because SpaCy uses a deep learning process with a CNN to learn the pattern of natural language used in user business requirements; therefore, the accuracy of measure and dimension identification in both datasets was very high. However, SpaCy's performance in performing this task varied substantially when requirements were couched in more complex terms. We found that for measure identification, SpaCy achieved 100% accuracy even when more complex structures were applied because terminologies used for measures do not vary significantly, making it easy for SpaCy to detect measures. However, the performance of dimension identification was reduced substantially when a complex structure was used, with only 94.73% accuracy achieved. The main reason for this loss of accuracy was incorrect identification due to the complex structure having more words that could be interpreted as noise, impeding SpaCy in learning the patterns of sentences effectively.

Our results indicate that SpaCy cannot determine measures and dimensions with greater than 97.50% accuracy in a sentence with a highly complex structure. This trend is similar for datasets with complex structures and heterogeneous terminology, where only 66.67% accuracy was achieved. Changing the terminology in the text by using synonyms was problematic for SpaCy, which cannot effectively learn data semantics and patterns from a training dataset. To solve this problem, we need a greater number of highly complex structures of business requirements to train the Spacy model.

Nonetheless, an average accuracy of 97.5% is acceptable for the model used in our framework (SSSD) for the complex structure of requirements, as humans usually use simple natural language phrases for queries.

After performing steps 1-5 as shown in Figure 1, our proposed mode SSSD, generates a star schema and becomes ready to perform OLAP operations. Figure 8 demonstrates the star schemas generated from both domains, dengue fever and AventureWorks. The constructed star schemas were stored in Microsoft SQL Server 2019.

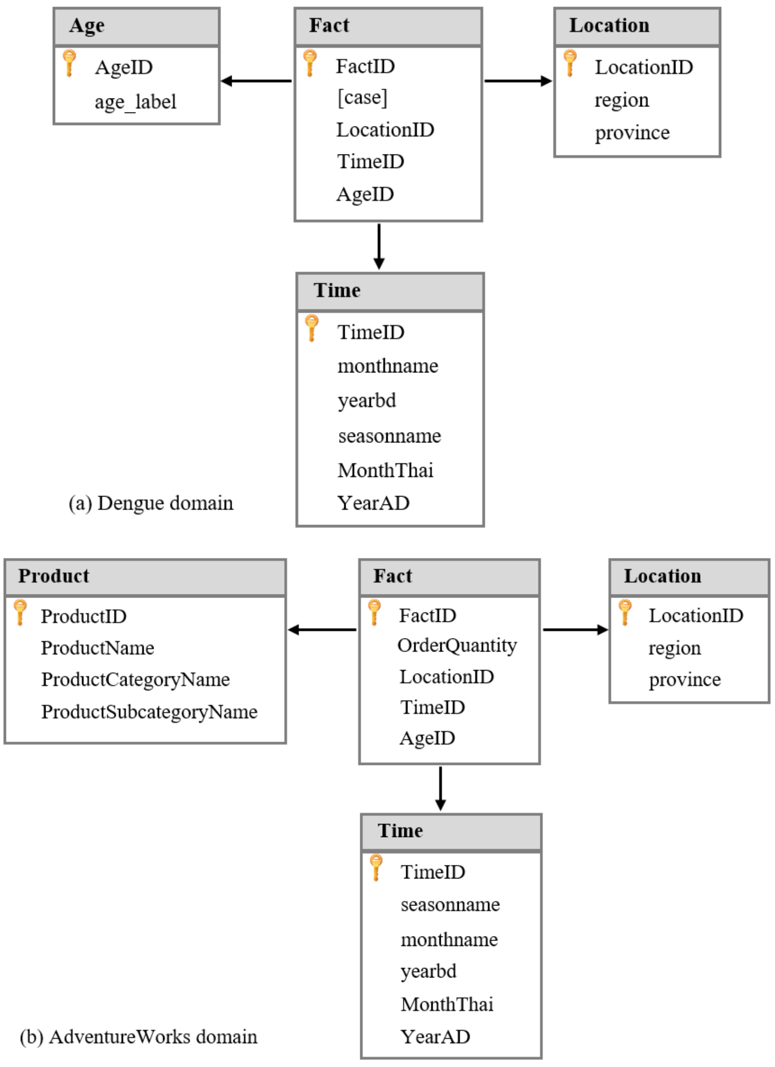
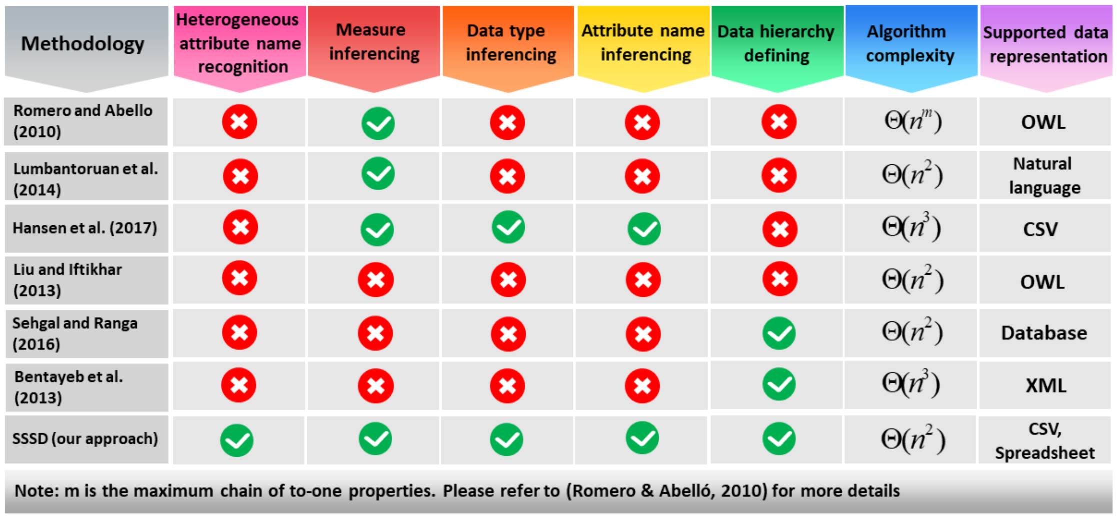


Figure 8 Generated star schemas from Dengue and Adventure Works domain

4.4 Comparison with traditional methods

This section presents our evaluation of the performance of the proposed approach by comparing it with four other state-of-the-art methods. However, it is difficult to implement the other methods because some details are absent, and often the intellectual property of the researchers. Therefore, we compare *SSSD* to others from some perspectives that are explicitly mentioned in their papers, as shown in Figure 9. First, several state-of-the art methods cannot deal with some uncertainties such as synonym problems [3, 5, 10, 21-23] and attribute name prediction when they are not available in the data source [3, 10, 21-24] resulting in failure to generate a star model. These systems cannot provide a valid data hierarchy when they do not know the attribute names or the various terminologies. These uncertainties simply, but importantly, mean that the hierarchies cannot be validly constructed, and so those systems do not work effectively.

A measure in a fact table is arguably the most important element for OLAP operations. Some works [10, 22, 23] did not provide the measure identification module and, thus, they cannot manage the uncertainty effectively when measures are not available in the data source, leading to conventional understanding of the difficulty of generating a fact table, and they may then fail to construct a multidimensional model. However, our *SSSD*, and similarly [3, 5, 21, 24], have a module to facilitate the detection of measures which can automate the determination of measures and, consequently, there is a lower chance of failure to construct a star schema for the DW.



Methodology	Heterogeneous attribute name recognition	Measure inferencing	Data type inferencing	Attribute name inferencing	Data hierarchy defining	Algorithm complexity	Supported data representation
Romero and Abello (2010)	✗	✓	✗	✗	✗	$\Theta(n^m)$	OWL
Lumbantoruan et al. (2014)	✗	✓	✗	✗	✗	$\Theta(n^2)$	Natural language
Hansen et al. (2017)	✗	✓	✓	✓	✗	$\Theta(n^3)$	CSV
Liu and Iftikhar (2013)	✗	✗	✗	✗	✗	$\Theta(n^2)$	OWL
Sehgal and Ranga (2016)	✗	✗	✗	✗	✓	$\Theta(n^2)$	Database
Bentayeb et al. (2013)	✗	✗	✗	✗	✓	$\Theta(n^3)$	XML
SSSD (our approach)	✓	✓	✓	✓	✓	$\Theta(n^2)$	CSV, Spreadsheet

Note: m is the maximum chain of to-one properties. Please refer to (Romero & Abelló, 2010) for more details

Figure 9 Comparison of automatic star schema design methods against various perspectives

5. Conclusion

We proposed a technique to automate design and construction of star schema models with a knowledge-based representation in the form of an ontology model. We defined six main phases to perform such a task, which is significantly different from methods presented in prior research in which the domain ontology model is leveraged to assist all the processes to effectively determine column names, measures, and dimensions. We use a novel probability density technique to determine column names of tables in a star schema model, where this information is missing in the data sources. Our proposed method can effectively and efficiently resolve existing problems in the state-of-the-art approaches, such as heterogeneity of terms, data type inferencing, and measure identification.

In future research, we aim to develop a novel technique to automate the generation of OLAP based on natural language user queries. This idea is challenging but highly likely to be possible because, in this study, we have shown an approach with the capacity to obtain key information from a user query. Therefore, this key information can be useful to our framework to formulate an SQL query to retrieve all desired information and present it in the form of OLAP.

6. Credit authorship contribution statement

K. Kesorn supervised, conceived of, and designed the study conceptualization. N. Sanprasit conducted data curation, experiments, and collected the experimental results. N. Sanprasit and K. Kesorn analyzed the data and developed the algorithm for the semantic approach for star schema construction. T. Titijaroonroj conducted an experiment to validate the attribute-name inferencing approach. K. Kesorn wrote the manuscript and provided a formal analysis of the results and validation. K. Kesorn wrote the original draft. N. Sanprasit and K. Kesorn conducted the visualization. All authors have read and approved the final manuscript, and declare that no competing interests exist. The funder had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

7. Acknowledgments

This research was supported by the Computer Science and Information Technology Department, Science Faculty, Naresuan University (Grant no: R2564C024), Health Systems Research Institute (Grant No. 63-017), the Program Management Unit for Human Resources & Institutional Development, Research and Innovation - CU (rant number B16F630071), and Thailand Science Research Innovation (TSRI)-CU (Grant No. FRB640001). The funder had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript. All authors have read and approved the final manuscript and declare that no competing interests exist.

8. References

[1] Elamin E, Feki J. Toward an ontology based approach for data warehousing state of the art and proposal. The international Arab conference on information technology (ACIT2014); 2014 Dec 9-11; University of Nizwa, Oman. p. 170-9.
[2] Abai NHZ, Yahaya JH, Deraman A. User requirement analysis in data warehouse design: a review. Proc Tech. 2013;11:801-6.
[3] Romero O, Abello A. A framework for multidimensional design of data warehouses from ontologies. Data Knowl Eng 2010;69:1138-57.
[4] Pardillo J, Mazon JN. Using ontologies for the design of data warehouses. Int J Database Manag Syst. 2011;3:73-87.

[5] Hansen JB, Jensen S, Tarp M, Thomsen C. DWStar - automated star schema generation. Aalborg: Aalborg University; 2017.
[6] Bimonte S, Antonelli L, Rizzi S. Requirements-driven data warehouse design based on enhanced pivot tables. Requir Eng. 2021;26(4):1-23.
[7] Moukhi NE, El I, Mouloudi A, Elmounadi A. Merge of X-ETL and XCube towards a standard hybrid method for designing data warehouses. Int J Adv Comput Sci Appl. 2019;10(10):130-9.
[8] Nebot V, Berlanga R, Perez JM, Aramburu MJ, Pedersen TB. Multidimensional integrated ontologies: a framework for designing semantic data warehouses. J Data Semant XIII. 2009;5530:1-36.
[9] Nebot V, Berlanga R. Building data warehouses with semantic web data. Decis Support Syst. 2012;52:853-68.
[10] Liu X, Iftikhar N. Ontology-based big dimension modeling in data warehouse schema design. In: Abramowicz W, editor. International conference on business information systems; 2013 Jun 19-21; Poznan, Poland. Berlin: Springer; 2013. p. 75-87.
[11] Khouri S, Boukhari I, Bellatreche L, Sardet E, Jean S, Baron M. Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. Comput Ind. 2012;63:799-812.
[12] Witten IH, Neal RM, Cleary JG. Arithmetic coding for data compression. Commun ACM. 1987;30:520-40.
[13] Howard PG, Vitter JS. Analysis of arithmetic coding for data compression. Inf Process Manag. 1992;28:749-63.
[14] Rubin F. Arithmetic stream coding using fixed precision registers. IEEE Trans Inf Theory. 1979;25:672-5.
[15] Bowman AW, Azzalini A. Applied smoothing techniques for data analysis. New York: Oxford University Press; 1997.
[16] Armbrust M, Xin RS, Lian C, Huai Y, Liu D, Bradley JK, et al. Spark SQL: Relational data processing in Spark. Proceedings of the 2015 ACM SIGMOD international conference on management of data; 2015 May 31 – Jun 4; Melbourne, Australia. New York: ACM; 2015. p. 1383-94.
[17] Partalidou E, Xioufis ES, Doropoulos S, Vologiannidis S, Diamantaras K. Design and implementation of an open source Greek POS tagger and entity recognizer using spaCy. WI '19: IEEE/WIC/ACM International conference on web intelligence; 2019 Oct 14-17; Thessaloniki, Greece. New York: Association for Computing Machinery; 2019. p. 337-41.
[18] Tarcar AK, Tiwari A, Dhaimodker V, Rebelo P, Desai R, Rao D. Healthcare NER models using language model pretraining. Health Search and Data Mining Workshop (HSDM 2020) in the 13th ACM International WSDM Conference (WSDM 2020); 2020 Feb; Houston, USA. p. 12-8.
[19] Miller GA. WordNet: a lexical database for English. Commun ACM. 1995;38:39-41.
[20] Mizoguchi R, Sunagawa E, Kozaki K, Kitamura Y. The model of roles within an ontology development tool: Hozo. Appl Ontol 2007;2(2):159-79.
[21] Lumbantoruan R, Sibarani EM, Sitorus MV, Mindari A, Sinaga SP. An approach for automatically generate star schema from natural language. TELKOMNIKA (Telecommun Comput Electron Control). 2014;12:501-10.
[22] Sehgal S, Ranga KK. Translation of entity relational model to dimensional model. Int J Comput Sci Mob Comput. 2016;5: 439-47.
[23] Bentayeb F, Maiz N, Mahboubi H, Favre C, Loudcher S, Harbi N, et al. Innovative approaches for efficiently warehousing complex data from the web. Data Min Concepts Methodol Tools Appl. 2013:1422-48.
[24] Song IY, Khare R, Dai B. SAMSTAR: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. DOLAP 2007, ACM 10th International workshop on data warehousing and OLAP; 2007 Nov 9; Lisbon, Portugal. New York: Association for Computing Machinery; 2007. p. 9-16.