ISSN 2630-0540
Journal of Technology and Innovation in Tertiary Education, Vol.2, No.2, pp. 1-24, July-December 2019

doi: ……

Thai QBE for Ad Hoc Query

Areerat Trongratsameethong
Phada Woodtikarn
Department of Computer Science, Faculty of Science
Chiang Mai University, Chiang Mai, Thailand
Email: areerat.t@cmu.ac.th

Abstract

Data for answering business questions usually query from databases. Software applications are provided for users to query these data. However, queries provided by software programs must be predefined through searching criteria specified in program user interfaces. The searching criteria are transformed to SQL statement and stored in a program. If users want data in different views or want more data, the searching criteria stored in a program must be modified or new programs must be created. This will consume more time and costs. Moreover, the ad hoc questions for real-time decision making require ad hoc query tools to generate ad hoc data and ad hoc reports. But users who want to use these tools must be trained or learned how to use them. These issues inspire the researchers/authors of this paper to develop an ad hoc query tool named *Thai QBE (Query-by-Example)* to query ad hoc data. User interfaces of Thai QBE were designed to support Thai language for ease in use. The concept used to design Thai QBE was derived from Query-by-Example (QBE) of Microsoft Access. A Thai QBE prototype was developed as a web application for users without technical skills to query ad hoc data anytime and anywhere. Two types of queries supported by Thai QBE were used: general query and statistical query. The researchers implemented and tested the Thai QBE prototype with fixed asset system database. The results prove that Thai QBE can serve as a powerful tool for querying ad hoc data. Thai QBE can be applied to query ad hoc data from other databases easily for its straightforward modification. The Thai QBE can be extended to support recursive relationship in future applications.

Keywords*: Thai QBE, Fixed Asset Database, Ad Hoc Query, Levenshtein Distance, Edit Distance Algorithm.*

1. Introduction

In business, managers or operation users want data and information to perform their tasks. Normally, these users query data and information from query programs provided in software applications, such as fixed asset system, accounting system, point of sale system, inventory system, and the like. Queries used in application programs must be predefined. That means, query results returned from predefined queries are limited to search criteria predefined in the programs. When users have other questions that cannot be answered by the existing programs, more time and costs are spent to modify programs or develop new programs to support user needs. More importantly, data used for real-time decision making are manifold and unpredictable. Therefore, ad hoc query tools are needed for these kinds of data, known as *ad hoc data*.

Currently, there are many tools designed to query ad hoc data, which are *Query-by-Example (QBE)* and business intelligence tool. However, users must learn or are

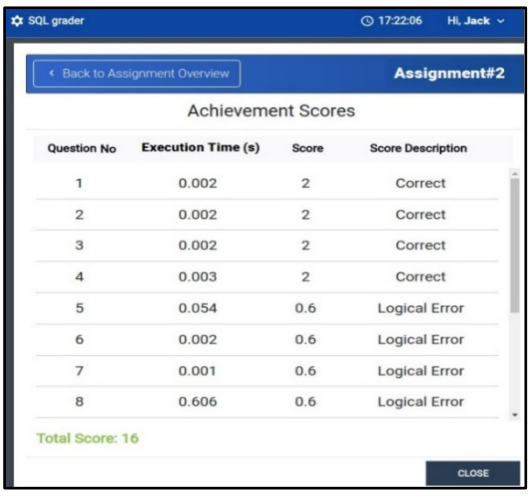
trained to use these tools. Such a limitation has prompted us to design and develop a tool to query ad hoc data without training. Our tool is designed to support Thai language and it is designed for ease in use. The tool is designed as a web application. Users can query ad hoc data anytime and anywhere. Two types of query are supported: *Thai QBE General Query and Thai QBE Statistical Query*. The concept used to design both Thai QBEs are derived from QBE of Microsoft Access. The prototype of Thai QBE was developed and tested with fixed asset system database. The experiments prove that both Thai QBEs can serve as a powerful tool. It can support routine operations and real-time decision making.

A. Predefined Query

Generally, users query data and information from programs provided in software applications. Search criteria used for searching and filtering data in any programs must be predefined. Figure 1 displays students' class scores of Automated SQL Assignment Grading System for Multiple DBMSs System (Trongratsameethong & Vichianroj, 2018). This screen queries score information with following conditions: student id = var1 (590510555) and subject id = var2 (204222) and year = var3 (2019) and semester = var4 (1). Variables var1 to var4 are entered by a user. Table names and attribute names must be predefined in the program as expressed below. It is noted that the execution time is computed by a program. If users want to search data from other search criteria, they must request a developer to modify programs or to create more programs. This will take a lot of time and cost more to develop programs. As a result, the predefined queries could turn out not appropriate for different views of data and not appropriate for real-time decision making.

```
SELECT    St.FIRSTNAME, St.LASTTNAME, Sj.SUBJECTNAME,
          Sub.QUESTIONID, Sct.SCOREPERCENT, Sct.SCOREDESC
FROM          SUBMISSIONDETAILS AS Sub, STUDENT AS St,
          SUBJECT AS Sj, SCORETYPE AS Sct
WHERE     Sub.YEAR = var3 AND Sub.SEMESTER = var4 AND
          Sub.SUBJECTID = var2 AND Sub.STUDENTID = var1 AND
          Sub.STUDENTID = St.STUDENTID AND
          Sub.SUBJECTID = Sj.SUBJECTID AND
          Sub.SCORETYPEID = Sct.SCORETYPEID;
```

Figure 1: An Example of Student Individual Scores.



B. Ad Hoc Query

An ad hoc query differs from a predefined query. The predefined query is created for routine tasks, such as searching customer information and generating daily sales summary reports. But an ad hoc query is created to obtain information as the need arises. The ad hoc query results are important and valuable for business tasks. They are usually used in analytical processes, especially for real-time decision making. Tools that are usually used by business organizations to support ad hoc query are business intelligence, such as Tableau [Tableau] and MicroStrategy [MicroStrategy]. These tools can connect to a database to query data stored in a database and the queried data are later visualized as a user need. However, these tools require knowledge in performing database connection and data visualization. Another tool that also requires knowledge in tool use is a QBE. QBE is provided in several Database Management Systems (DBMSs), such as Microsoft Access and Microsoft SQL Server. Users familiar with these tools are developers or users who have been trained in using these software tools.

C. Fixed Asset Management System

The fixed asset management system is used to manipulate company asset data. These assets are used to support companies to run their businesses, such as machineries, computers, tables, desks, and the like. Assets sold to customers are not included. Assets have a finite lifespan, because their usefulness declines over the years as a result of usage. In order to maintain these assets efficiently over their lifespan, companies must be able to track the depreciation value of these assets accurately, as well as accurately manage preventive maintenance. These features are where the fixed asset management software focuses on. There are five main features in the system: (1) managing fixed asset data, (2) tracking asset locations, (3) write off fixed assets, (4) querying asset data, and (5) generating reports related to fixed asset data. However, queries used in routine processes are predefined, such as searching fixed asset data and reporting fixed asset data in the software applications. If users want to search or filter data other than what the search criteria provided in the software application, they have to request their developer to modify programs or create new programs to support their needs. It should be noted that information used for real-time decision making cannot wait, for it must be returned immediately after queries. As a result, the predefined queries provided in the software application may not be appropriate for real-time decision making.

D. Query-By-Example

Query-by-example (QBE) is a database query language for relational databases. It is designed in a graphical user interface. Users can query data stored in a database from visual tables provided in the graphical user interface. Users can specify conditions in the interface to filter data. Many database user interfaces derive ideas from QBE. The advantage of QBE is that an SQL parser [SQL Structured Query Language] can convert the user's queries specified in a graphical user interface into SQL statements. This can minimize the burden on the users to remember the finer details of SQL statement, and it is easier and more productive for end-users and developers to select tables and their attributes by selecting them rather than typing in their names. QBE is supported in several Relational Database Management Systems (RDBMSs). An example of RDBMS implemented QBE is Microsoft Access (Wilbert, 2010). Although end-users can use QBE to query ad hoc data but they, in the first place, must be trained how to use QBE tools.

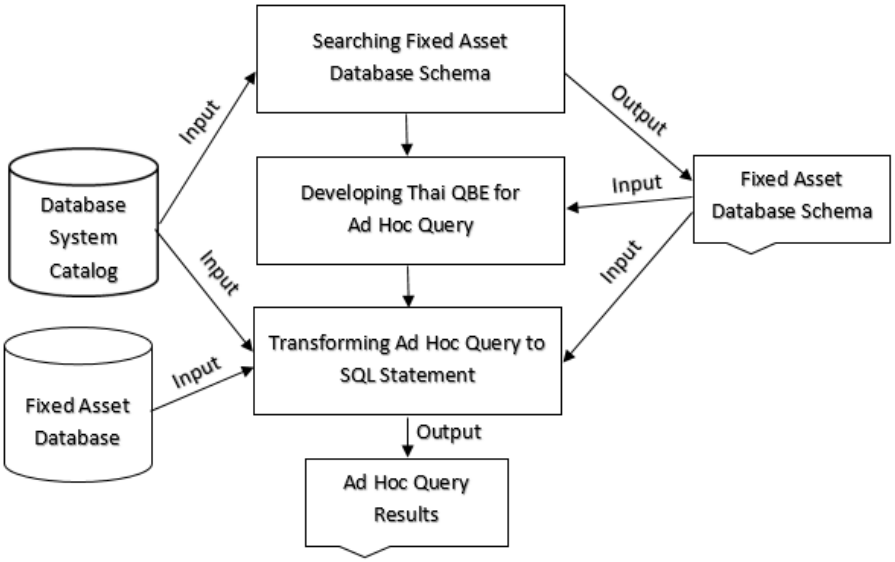
2. Research Objective

This research was to develop a Thai QBE prototype for querying ad hoc data from fixed asset system database.

3. Research Methodology

In this research, a Thai QBE prototype was designed to query ad hoc data from a fixed asset system database. An overview of the Thai QBE prototype is presented in Figure 2. The prototype consists of three main sections: (a) Searching Fixed Asset Database Schema (b) Developing Thai QBE for Ad Hoc Query and (c) Transforming Ad Hoc Query to SQL Statement.

Figure 2: Overview of Thai QBE Prototype for Querying Ad Hoc Data from Fixed Asset System Database



A. Searching Fixed Asset Database Schema

Normally, a database schema is designed and created by a Database Administrator (DBA). Database schemas are stored in a database system catalog. DBMS used in this research work is MySQL DBMS. It stores database schema information in a system catalog named 'information_schema'. Database names defined by a DBA must follow MySQL DBMS naming convention (Hinz et al., 2020). However, database names defined by a DBA may be vary, for examples, "Asst", "Assts", "Asset", "FixAsset", "FixAssets", "dbFixedAsset", db_FixedAsset", and so on. As a result, we search database schema having the same semantic as "Asset" or "FixedAsset" instead, which are following names: "Asset" or "FixAsset" or "FixedAsset." Here and after, these three names are called *db_candidates*. The schema having highest percent of string similarity compared to these three names, is selected. Tables, attributes, and foreign keys of the selected schema are outputs of this section. There are 5 sub processes used to search Fixed Asset database schema as explained below:

1) Find Database Schema on System Environment or Database Server: This sub process starts with searching all database names on database server using following SQL statement:

```
SELECT      SCHEMA_NAME
FROM        INFORMATION_SCHEMA.SCHEMATA;
```

Query results may be as follows:

- Contain both ‘f’/’F’ and ‘a’/’A’: For example, ‘financial’, ‘FINANCIAL’, ‘db_financial_accounting’, ‘DBFinancialAccounting’, ‘Financial_Accounting’, ‘Fixed_Asset’, ‘fix_asst’, ‘db_Fix_assets’.
- Contain ‘f’/’F’: For examples, ‘Food’, ‘FoodNutrient’.
- Contain ‘a’/’A’: For examples, ‘accounting’, ‘Accounting’, ‘db_accounting’, ‘db_assets’, ‘Assets’, ‘ASSET’.

The query results are later converted to lower case and underscore ‘_’ (if any) in each result is removed. Here and after, these query results are called *schema_names*.

2) Calculating String Similarity: *schema_names* are checked percent of string similarity compared with *db_candidates* one by one. The *schema_names* having highest percent of similarity is selected as an output of this sub process. There are two main parts for this sub process:

a. Compute Levenshtein Distance (*d*): Function named iterative_levenshtein (s, t) written in Python (Necas et al., 2014) is used to measure the similarity of two strings. The concept of string similarity algorithm used in this function is based on Edit Distance as described in (Zhang, Hu, & Bian, 2017). We use it to measure distance between strings in *schema_names* and *db_candidates*. Each string in *schema_names* is compared with all strings in *db_candidates* one by one. If a distance value is zero, it means two strings are the same. If the distance value is very high, it means two strings are very different. For examples:

d1 = iterative_levenshtein(“accounting”, “asset”) = 8
d2 = iterative_levenshtein(“asst”, “asset”) = 1

b. Compute Percent of String Similarity: The Levenshtein distance *d* from step a is converted to percent of string similarity using Equation (1).

$$\text{String Similarity (\%)} = \frac{l-d}{l} \times 100, \quad (1)$$

where *l* is number of characters of the longest string and *d* is the distance between two strings. String Similarity Examples:

String Similarity (“accounting”, “asset”) = (10 – 8)/10*100 = 20%
String Similarity (“asst”, “asset”) = (5 – 1)/6*100 = 80%

The *schema_names* having highest percent of string similarity are selected as an output of this step. If there are more outputs than one, one of them is selected as arbitrary. However, the highest percent value must be greater than 80 percent because database names having the same semantic or very close to “asset” or “fixedasset,” must have percent of string similarity greater than or equal to 80 percent as displayed in Table 1. Here and after, the output of this step is called *asset_db*, which is the name of fixed asset database.

3) Find Tables Names of *asset_db* Database Schema: The SQL statement displayed below is performed. Outputs from this step are all table names belonging to *asset_db* database schema.

```
SELECT    TABLE_NAME
FROM      INFORMATION_SCHEMA.TABLES
WHERE     TABLE_TYPE = 'BASE TABLE' AND
          TABLE_SCHEMA = asset_db
```

```
ORDER BY    TABLE_NAME;
```

4) Find Attribute Names of *asset_db* Database Schema: The SQL statement displayed below is performed. Outputs from this step are all attribute names belonging to *asset_db* database schema.

```
SELECT      TABLE_NAME, COLUMN_NAME
FROM        INFORMATION_SCHEMA. COLUMNS
WHERE       TABLE_SCHEMA = asset_db
ORDER BY    TABLE_NAME, COLUMN_NAME;
```

5) Find Foreign Keys of *asset_db* Database Schema: The SQL statement displayed below is performed. Outputs from this step are all names belonging to *asset_db* database schema.

```
SELECT      TABLE_NAME, COLUMN_NAME,
            CONSTRAINT_NAME,
            REFERENCED_TABLE_NAME,
            REFERENCED_COLUMN_NAME
FROM        INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE       REFERENCED_TABLE_SCHEMA = asset_db;
```

where COLUMN_NAME is a foreign key attribute in TABLE_NAME, CONSTRAINT_NAME is a name of foreign key, REFERENCED_TABLE_NAME is a primary table of a foreign key attribute, and REFERENCED_COLUMN_NAME is an attribute in a primary table that the foreign key attribute in TABLE_NAME refers to.

Table 1: Percent of String Similarity Examples

Database Names	asset		fixedasset	
	Distance (*d*)	String Similarity (%)	Distance (*d*)	String Similarity (%)
asst	1	80.00	6	40.00
assts	2	60.00	7	30.00
assets	1	83.33	6	40.00
fix	5	0.00	7	30.00
fixed	4	20.00	5	50.00
fixasst	4	42.86	3	70.00
fixassts	5	37.50	4	60.00
fixedasst	6	33.33	1	90.00
fixedassts	7	30.00	2	80.00
fixasset	3	62.50	2	80.00
fixassets	4	55.56	3	70.00
fixedassets	6	45.45	1	90.91

B. Developing Thai QBE for Ad Hoc Query

The concept used to design Thai QBE user interfaces is derived from QBE of Microsoft Access (Adamski & Finnegan, 2010). User interfaces of Thai QBE are designed for easy use in support of Thai language. There are two main steps to generate Thai QBE as follows:

1) Translating table names and attribute names into Thai language: table names and attribute names obtained by the previous section are translated into Thai language using Python Translate library (Yin, 2017).

2) Designing user interfaces for Thai QBE: There are two user interfaces: Thai QBE General Query and Thai QBE Statistical Query. Mockups of these two interfaces are displayed in Figures 3 and 4, respectively. Components of these Thai QBE user interfaces are explained below:

a. Table List: table names including attribute names of each table after being translated into Thai language (from step 1) are shown in left hand side frame under Table List part.

b. Query Type: There are two types of queries. If uses select general query, basic SQL SELECT (statement without aggregation function) is performed. If users select statistical query, SQL SELECT statement with aggregation function is performed.

c. Display Data: This part is for specifying attribute(s) that users want to show in query results. Users can select attributes from check boxes placed in the Table List part. For statistical query, users can specify aggregation function by pressing icon. Users can specify attribute to show statistical data from the drop down list shown below. We support following aggregation functions: MIN, MAX, SUM, AVG, and COUNT.

d. Condition: This part is used to filter records of data. There are three components of conditions:

- First Operand: Attributes that users select from the left hand side frame are filled in the "Attribute" dropdown list (the First Operand) under the Condition part.
- Comparison Operator: It can be one of the followings: =, >, >=, <, <=, !=. If the first operand is a string data type attribute, the LIKE operator will be provided more in the comparison dropdown list.
- Second Operand: The second operand contains attributes that users select from the left hand side frame or users can specify value in this part. If users specify LIKE operator, Python WordNet Library (NLTK Project, 2019) is used to find words having the same semantic in Second Operand. For example, if users specify LIKE "% คอมพิวเตอร์%", not only "คอมพิวเตอร์" [word 1] is retrieved but "คอม" [word 2] and "เครื่องคอม" [word 3] are also retrieved. Two wildcard are supported:
  - '%' represents zero, one, or multiple characters
  - '_' represents a single character.

Users can specify more than one condition by pressing Add button, the logical AND and OR are provided for users to combine two or more conditions.

e. Group By: This part is used with group data having the same values. Users can group data by attributes or aggregation functions apparent in Display Data part. The Group By is only for Statistical Query.

f. Statistical Condition: This part is quite similar to Condition part but it is used for Statistical Query to filter data based on statistical information. There are three components: first operand, comparison operator, and second operand. The first operand must have numeric data types, they are from aggregation function apparent in Display Data part. The comparison operator can be: =, >, >=, <, <=, or !=. The second operand also contains aggregation function apparent in Display Data part or users can specify value in this part.

g. Order By: This part is used for sorting values of attribute or aggregation function (for Statistical Query) apparent in Display part. The sorting data will be displayed below the Order By part. They can be sorted in a descending order (DESC) or an ascending (ASC) .

h. Query Text Summary: This part is used for summarizing query specified by users in text form.

i. Translation: This part is used to display SQL statement. The methodology used for translating Thai QBE components into SQL statement is explained in the next section. After SQL Translation being performed, the SQL statement is executed and query results are generated and displayed at the end of the user interface.

C. Transforming Ad Hoc Query to SQL Statement

Data entered by a user in the right hand side frame of Thai QBE are inputs of this section. There are two types of inputs: General Query and Statistical Query. Inputs of General Query are from the following parts: Display Data, Condition, and Order By. Condition and Order By parts are optional. Inputs of Statistical Query are similar to those of General Query but have two more parts. They are Group By and Statistical Condition parts, and these two parts are optional.

The components in General Query and Statistical Query are transformed using SQL statement displayed below. Components in Thai QBE are mapped onto SQL statement in a transformation process using information displayed in Table 2. If users perform General Query, GROUP BY and HAVING will not be included.

```
SELECT      <attribute_1, attribute_N>
FROM        <table_1, table_N>
WHERE       <conditions>
GROUP BY    < attribute_1, attribute_N>
HAVING      <statistical conditions>
ORDER BY    < attribute_1, attribute_N>
```

Figure 3: Thai QBE General Query Mockup

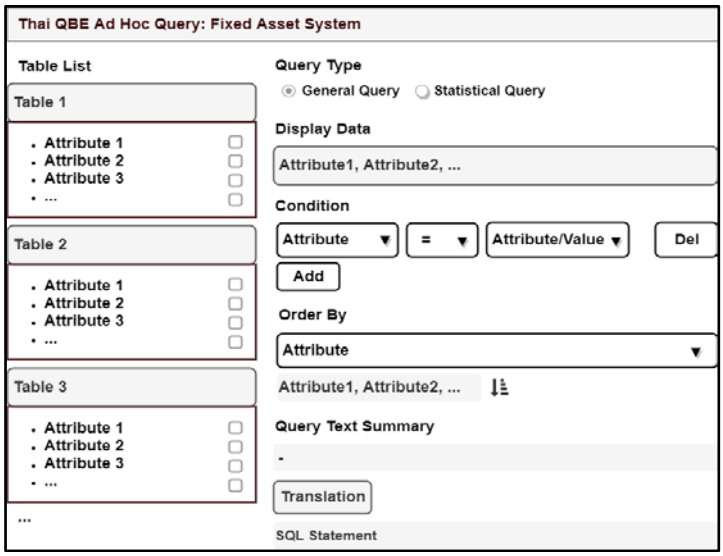


Figure 4: Thai QBE Statistical Query Mockup

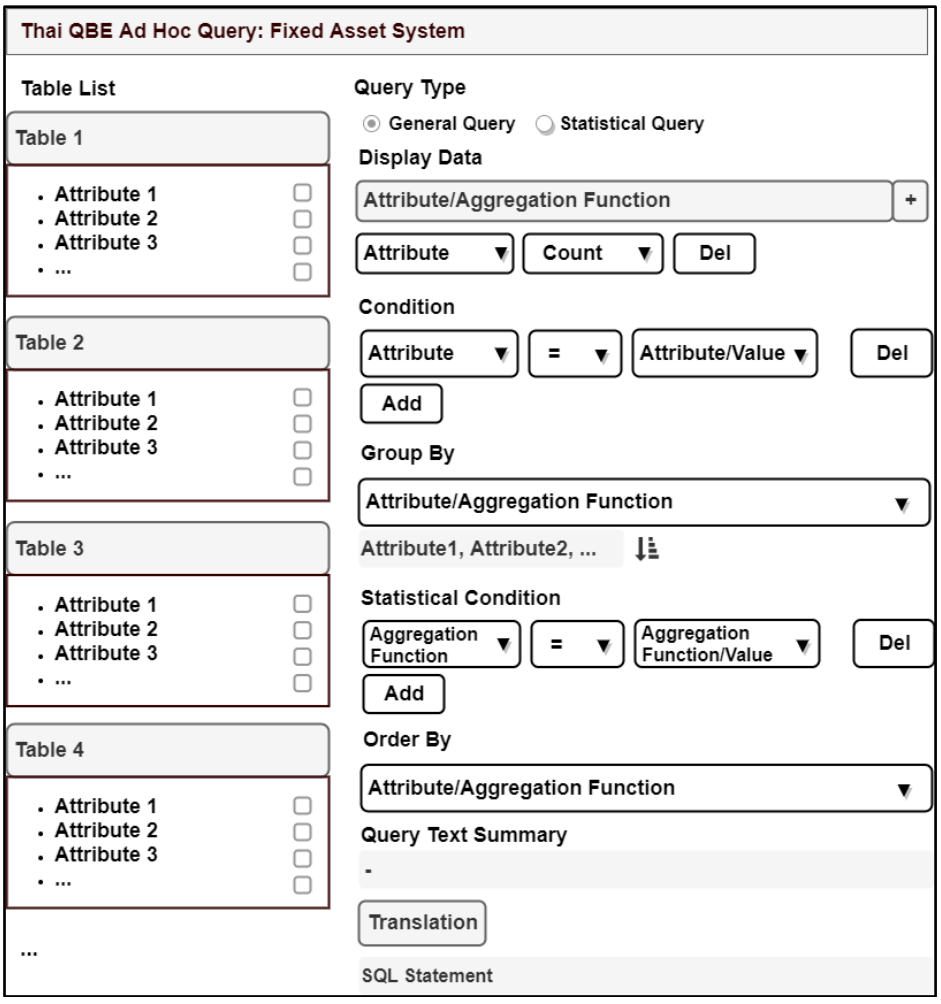


Table 2: Mapping Thai QBE Components to SQL Statement

SQL Statement		Thai QBE	
Fixed Part	**Mapping Part**	**General Query**	**Statistical Query**
SELECT	<attribute_1, attribute_N>	Display Data	Display Data
FROM	<table_1, table_N>	Table List	Table List
WHERE	<conditions>	Condition	Condition
GROUP BY	< attribute_1, attribute_N>	-	Group By
HAVING	<statistical conditions>	-	Statistical Condition
ORDER BY	< attribute_1, attribute_N>	Order By	Order By

In the case that there are more than one table after FROM keyword, join attribute(s) must be specified *n*-1 couples, where *n* is number of tables. Join attributes are from foreign keys obtained from step 5 of section A as follows:

TABLE_NAME.COLUMN_NAME =
REFERENCED_TABLE_NAME.REFERENCED_COLUMN_NAME

4. Results and Discussion

The experimental setup was performed according to step by step described in the research methodology section as follows:

1) Searching Fixed Asset Database Schema: Database schemas on the test environment used in our experiment are displayed in Figure 5. There are 5 sub processes as follows:

a. Find Database Schema on Test Environment: The SQL statement used to find all databases on test environment and its query results are displayed below.

```
SELECT   SCHEMA_NAME
FROM     INFORMATION_SCHEMA.SCHEMATA;
```

SCHEMA_NAME
accounting
car_insurance
finance
fix_asset
information_schema
mysql
performance_schema
sakila
sys
world

b. Calculate String Similarity: The query results from 1(a) are later computed for percent of string similarity using Equation (1). The percent of string similarities is displayed in Table 3. The database schema name with highest percent of string similarity, which is fix_asset, is selected. Structure of fix_asset database is from Williams (2017) as presented in Figure 6.

c. Find Tables Names of fix_asset Database Schema: The SQL statement displayed below is performed. Query results are displayed in Figure 7(a).

```
SELECT      TABLE_NAME
FROM        INFORMATION_SCHEMA.TABLES
WHERE       TABLE_TYPE = 'BASE TABLE' AND
            TABLE_SCHEMA = ″fix_asset″
ORDER BY    TABLE_NAME;
```

d. Find Attribute Names of fix_asset Database Schema: The SQL statement displayed below is performed. Query results are displayed in Figure 7(b).

```
SELECT      TABLE_NAME, COLUMN_NAME
FROM        INFORMATION_SCHEMA. COLUMNS
WHERE       TABLE_SCHEMA = ″fix_asset”
ORDER BY    TABLE_NAME, COLUMN_NAME;
```

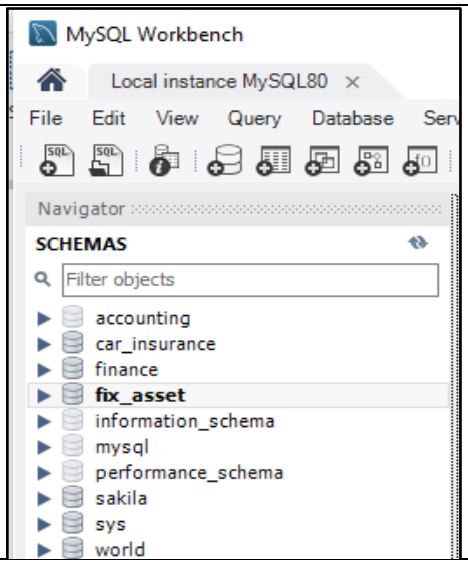
e. Find Foreign Keys of fix_asset Database Schema: the SQL statement displayed below is performed. Query results are displayed in Figure 7(c).

```
SELECT      TABLE_NAME, COLUMN_NAME,
            CONSTRAINT_NAME,
            REFERENCED_TABLE_NAME,
            REFERENCED_COLUMN_NAME
FROM        INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE       REFERENCED_TABLE_SCHEMA = ″fix_asset″;
```

2) Generating Thai QBE for Ad Hoc Query: Table names and attribute names obtained from the previous section are translated into Thai language using Python Translate library. They are displayed in the left hand side frame and searching criteria are displayed in the right hand side frame. Examples of ad hoc query results are based on data displayed in Figure 8. Examples of Thai QBE ad hoc query for General Query are displayed in Figures 9 to 12 and Statistical Query are displayed in Figures 13 to 16.

3) Transforming Ad Hoc Query to SQL Statement: Information in Table 2 are used to map components of Thai QBE onto SQL statement. If there are more tables than one specified in the Thai QBE, foreign keys obtained from the previous section shown in Figure 6(d) are used to specified join attributes. Examples of transforming Thai QBE ad hoc query to SQL basic statements and SQL statements with aggregation function are displayed in Figures 9 to 12 and Figures 13 to 16, respectively.

Figure 5: Database Schema on a Test Environment



Database Description
accounting: Accounting System Database
car_insurance: Car Insurance System Database
finance: Financial System Database
fix_asset: Fixed Asset System Database
sakila: Film System Database
world: Country Information Database
information_schema, mysql, performance_schema, sys: System databases created by MySQL DBMS

Table 3 String Similarity of Database Schemas Stored in Test Environment

Database Names	asset		fixedasset	
	Distance (*d*)	**String Similarity (%)**	**Distance (*d*)**	**String Similarity (%)**
accounting	8	20.00	10	0.00
carinsurance	10	16.67	16.67	16.67
finance	6	14.29	6	40.00
fixasset	3	62.50	2	80.00
sakila	5	16.67	9	10.00
world	5	0.00	9	10.00
informationschema	14	17.65	14	17.65
mysql	4	20.00	9	10.00
performanceschema	14	17.65	14	17.65
sys	4	20.00	9	10.00

Figure 6: Structure of fix_asset Database

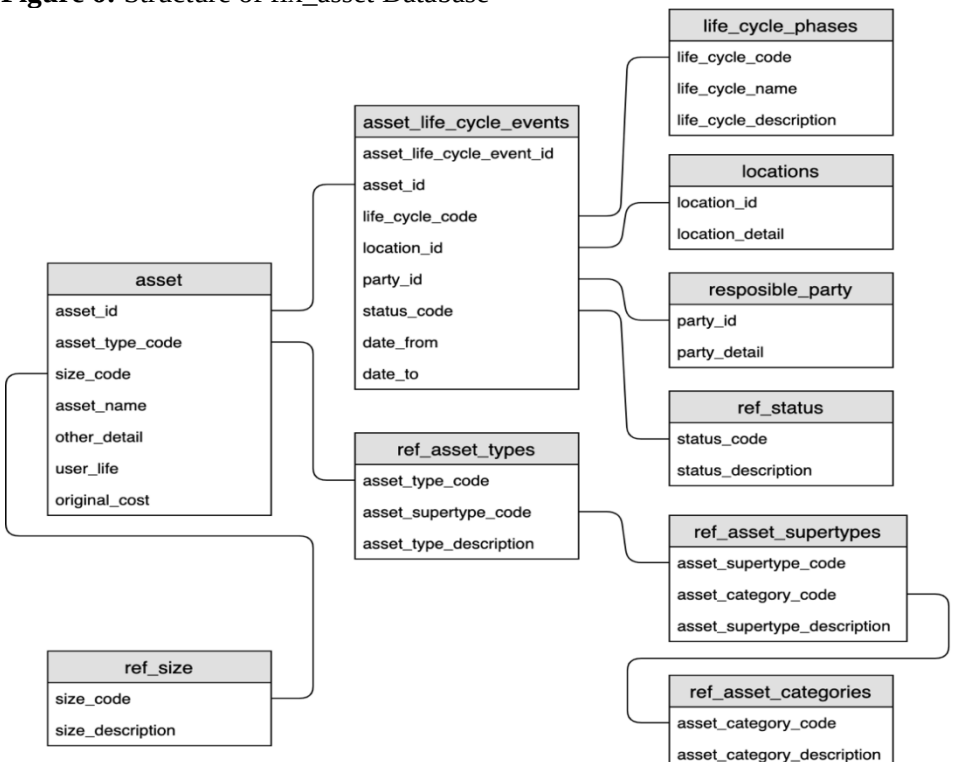


Figure 7: Tables, Attributes, and Foreign Keys of fix_asset Database

TABLE_NAME
asset
asset_life_cycle_events
life_cycle_phases
locations
ref_asset_categories
ref_asset_supertypes
ref_asset_types
ref_size
ref_status
responsible_party

TABLE_NAME	COLUMN_NAME
asset	asset_id
asset	asset_name
asset	asset_type_code
asset	original_cost
asset	other_details
asset	size_code
asset	use_life
asset_life_cycle_events	asset_id
asset_life_cycle_events	asset_life_cycle_event_id
asset_life_cycle_events	date_from
asset_life_cycle_events	date_to
asset_life_cycle_events	life_cycle_code
asset_life_cycle_events	location_id
asset_life_cycle_events	party_id
asset_life_cycle_events	status_code

TABLE_NAME	COLUMN_NAME
life_cycle_phases	life_cycle_code
life_cycle_phases	life_cycle_description
life_cycle_phases	life_cycle_name
locations	location_detail
locations	location_id
ref_asset_categories	asset_category_code
ref_asset_categories	asset_category_description
ref_asset_supertypes	asset_category_code
ref_asset_supertypes	asset_supertype_code
ref_asset_supertypes	asset_supertype_descrip...
ref_asset_types	asset_supertype_code
ref_asset_types	asset_type_code
ref_asset_types	asset_type_description
ref_size	size_code
ref_size	size_description
ref_status	status_code
ref_status	status_description
responsible_party	party_detail
responsible_party	party_id

(a) Tables of fix_asset (b) Attributes of fix_asset

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
ref_asset_supertypes	asset_category_code	ref_asset_supertypes_ibfk_1	ref_asset_categories	asset_category_code
ref_asset_types	asset_supertype_code	ref_asset_types_ibfk_1	ref_asset_supertypes	asset_supertype_code
asset	asset_type_code	asset_ibfk_1	ref_asset_types	asset_type_code
asset	size_code	asset_ibfk_2	ref_size	size_code
asset_life_cycle_events	asset_id	asset_life_cycle_events_ibfk_1	asset	asset_id
asset_life_cycle_events	life_cycle_code	asset_life_cycle_events_ibfk_2	life_cycle_phases	life_cycle_code
asset_life_cycle_events	location_id	asset_life_cycle_events_ibfk_3	locations	location_id
asset_life_cycle_events	party_id	asset_life_cycle_events_ibfk_4	responsible_party	party_id
asset_life_cycle_events	status_code	asset_life_cycle_events_ibfk_5	ref_status	status_code

(c) Foreign Keys of fix_asset

Figure 8: Experimental Data from fix_asset Database Asset

asset_id	asset_type_code	size_code	asset_name	other_details	use_life	original_cost
1	T0001	M	โน๊ตบุ๊ค	NULL	5	30000.00
2	T0004	L	โต๊ะคอมพิวเตอร์	NULL	15	5600.00
3	T0004	L	โต๊ะประชุม	NULL	15	12000.00
4	T0005	M	เก้าอี้คอมพิวเตอร์	NULL	5	2600.00
5	T0001	M	เครื่องคอมพิวเตอร์	NULL	5	16000.00
6	T0001	M	คอมAsus	NULL	5	12000.00
7	T0001	M	คอมiMac	NULL	5	60000.00

ref_asset_type

asset_type_code	asset_supertype_code	asset_type_description
T0001	ST0001	คอมพิวเตอร์
T0002	ST0001	อุปกรณ์เครือข่าย
T0003	ST0001	เครื่องพิมพ์
T0004	ST0002	โต๊ะ
T0005	ST0002	เก้าอี้

asset_life_cycle_events

asset_life_cycle_event_id	asset_id	life_cycle_code	location_id	party_id	status_code	date_from	date_to
1	1	1	1	1	S0002	2016-11-01 10:00:00	NULL
2	2	1	1	1	S0002	2016-11-01 10:00:00	NULL
3	3	1	1	1	S0002	2016-11-01 10:00:00	NULL
4	4	1	1	1	S0002	2016-11-01 10:00:00	NULL
5	5	1	1	1	S0002	2016-11-01 10:00:00	NULL
6	6	1	1	1	S0002	2016-11-01 10:00:00	NULL
8	1	2	3	2	S0002	2016-11-02 09:30:00	NULL
9	2	2	3	2	S0002	2016-11-02 09:30:00	NULL
10	3	2	3	2	S0002	2016-11-02 09:30:00	NULL
11	4	2	3	2	S0002	2019-02-15 09:30:00	NULL
12	5	2	4	3	S0002	2016-11-02 09:30:00	NULL
13	6	2	4	3	S0002	2016-11-02 09:30:00	NULL
15	1	3	3	2	S0001	2018-01-31 09:30:00	NULL
16	4	3	3	2	S0001	2016-11-02 09:30:00	NULL
17	4	4	3	2	S0003	2019-02-16 09:30:00	2019-02-16 09:30:00

Figure 9: Example 1 - Thai QBE Ad Hoc Query for General Query

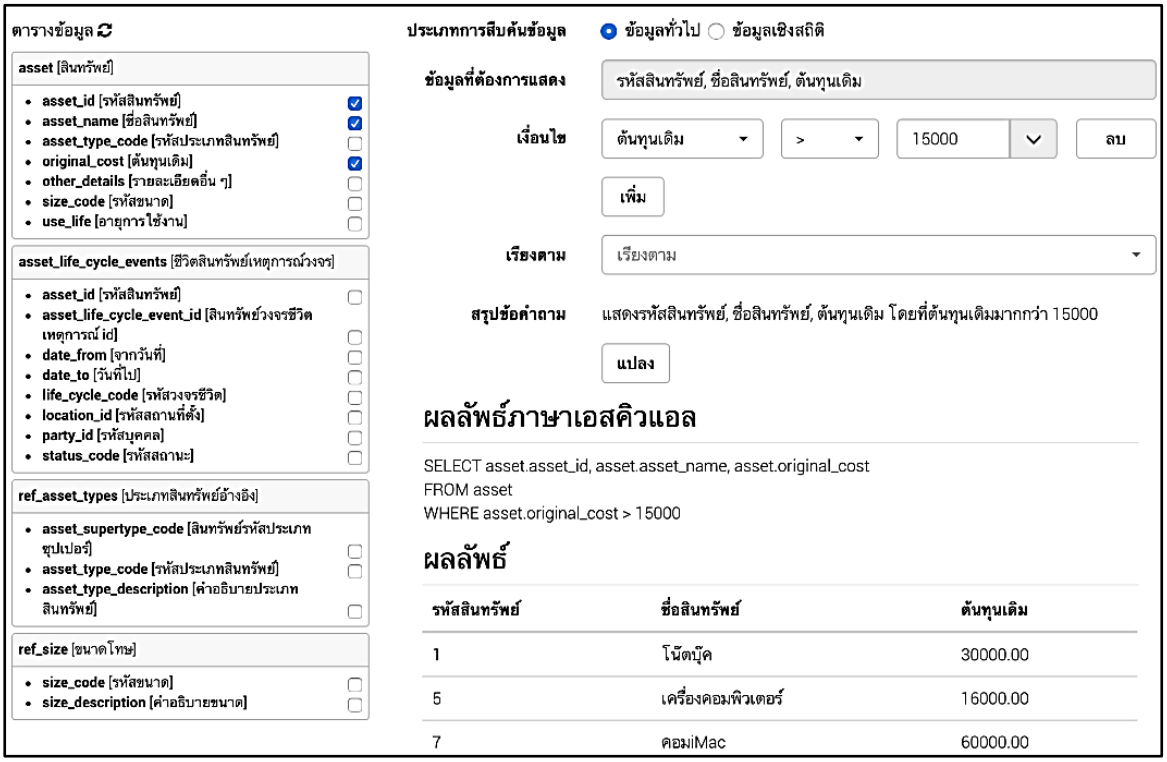


Figure 9 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = General Query (ข้อมูลทั่วไป)
- Display Data = asset_id (รหัสสินทรัพย์), asset_name (ชื่อสินทรัพย์), original_cost (ต้นทุนเดิม)
- Condition (เงื่อนไข):
    - First Operand (Attribute) = original_cost (ต้นทุนเดิม)
    - Comparison Operator = ″>″
    - Second Operand (Value) = 15000
- Order By (เรียงตาม): -
- Query Text Summary (สรุปข้อคำถาม) = “Display asset_id, asset_name, original_cost where original_cost > 15000”. The query text summary in Thai is displayed under the “สรุปข้อคำถาม” part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed onto SQL statement using information specified in Table 2. The SQL statement result is displayed under the “ผลลัพธ์ภาษาเอสคิวแอล” part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named “asset.”

Figure 10: Example 2 - Thai QBE Ad Hoc Query for General Query

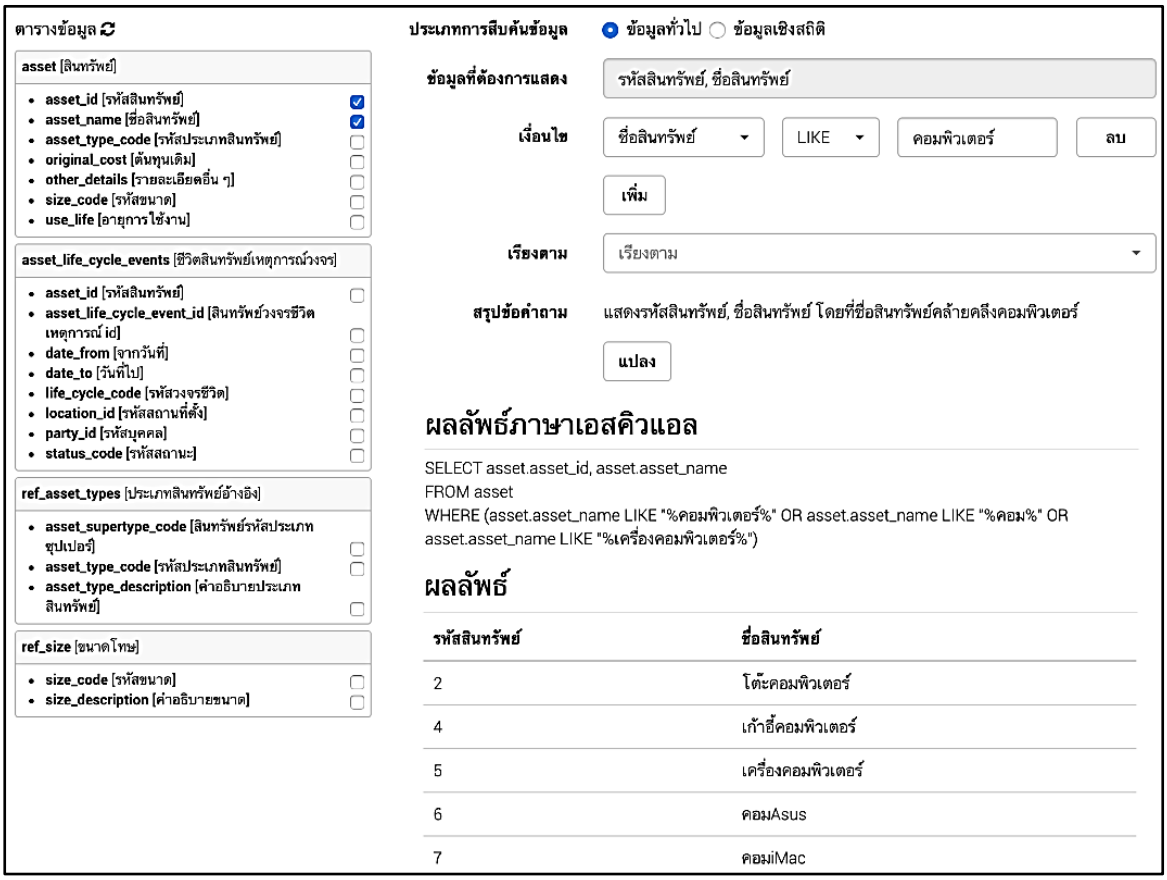


Figure 10 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = General Query (ข้อมูลทั่วไป)
- Display Data = asset_id (รหัสสินทรัพย์), asset_name (ชื่อสินทรัพย์)
- Condition (เงื่อนไข): If user specifies “LIKE” operator, the Python Wordnet Library is used to find the same semantic data as second operand and these semantic data are combined with OR logical operator.
    - First Operand (Attribute) = asset_name (ชื่อสินทรัพย์)
    - Comparison Operator = LIKE
    - Second Operand (Value) = “Computer” (“คอมพิวเตอร์”)
- Order By (เรียงตาม): -
- Query Text Summary (สรุปข้อคำถาม) = “Display asset_id, asset_name where asset_name LIKE computer”. The query text summary in Thai is displayed under the “สรุปข้อคำถาม” part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed onto SQL statement using information specified in Table 2. The SQL statement result is displayed under the “ผลลัพธ์ภาษาเอสคิวแอล” part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named “asset.”

Figure 11: Example 3 - Thai QBE Ad Hoc Query for General Query

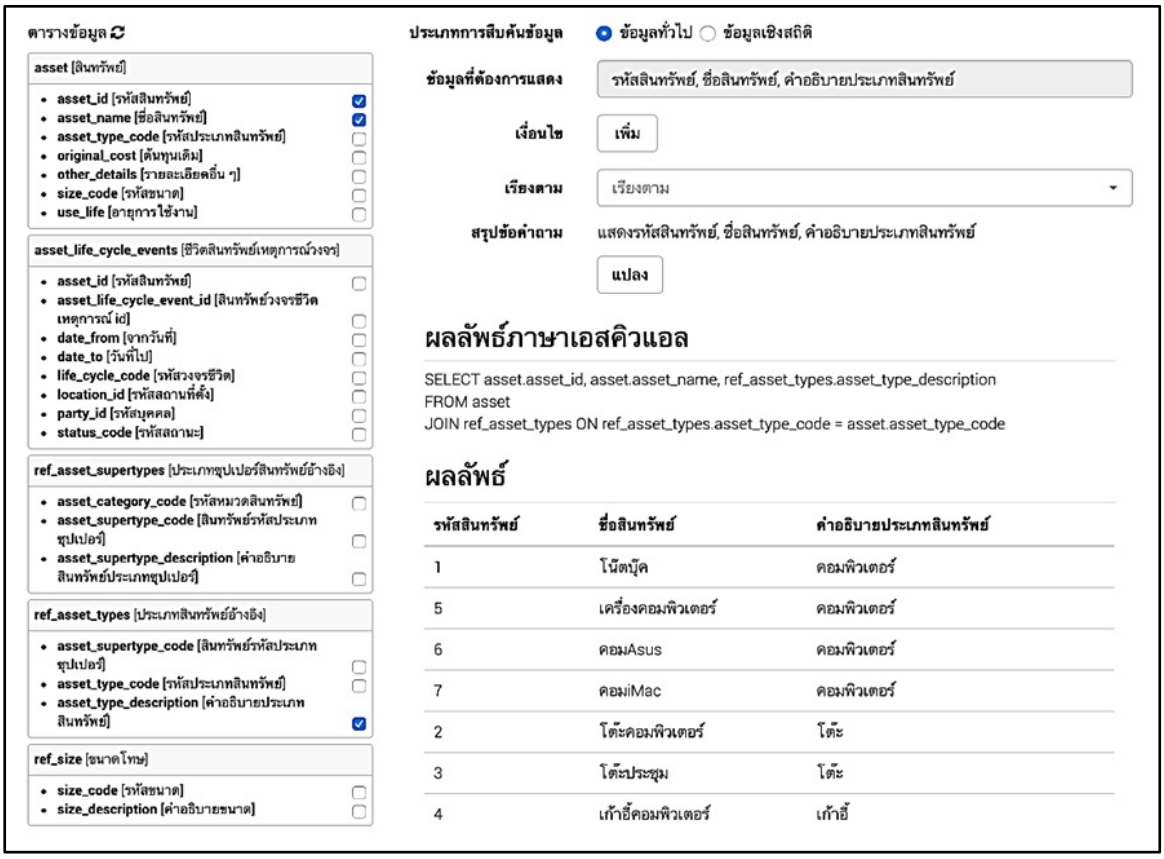


Figure 11 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = General Query (ข้อมูลทั่วไป)
- Display Data = asset_id (รหัสสินทรัพย์), asset_name (ชื่อสินทรัพย์), asset_type_description (คำอธิบายประเภทสินทรัพย์)
- Condition (เงื่อนไข): -
- Order By (เรียงตาม): -
- Query Text Summary (สรุปข้อคำถาม) = "Display asset_id, asset_name, asset_type_description". The query text summary in Thai is displayed under the "สรุปข้อคำถาม" part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed onto SQL statement using information specified in Table 2. The SQL statement result is displayed under the "ผลลัพธ์ภาษาเอสคิวแอล" part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named "asset" and "ref_asset_types". A join attribute named "asset_type_code" is used to join data from "asset" and "ref_asset_types."

Figure 12: Example 4 - Thai QBE Ad Hoc Query for General Query

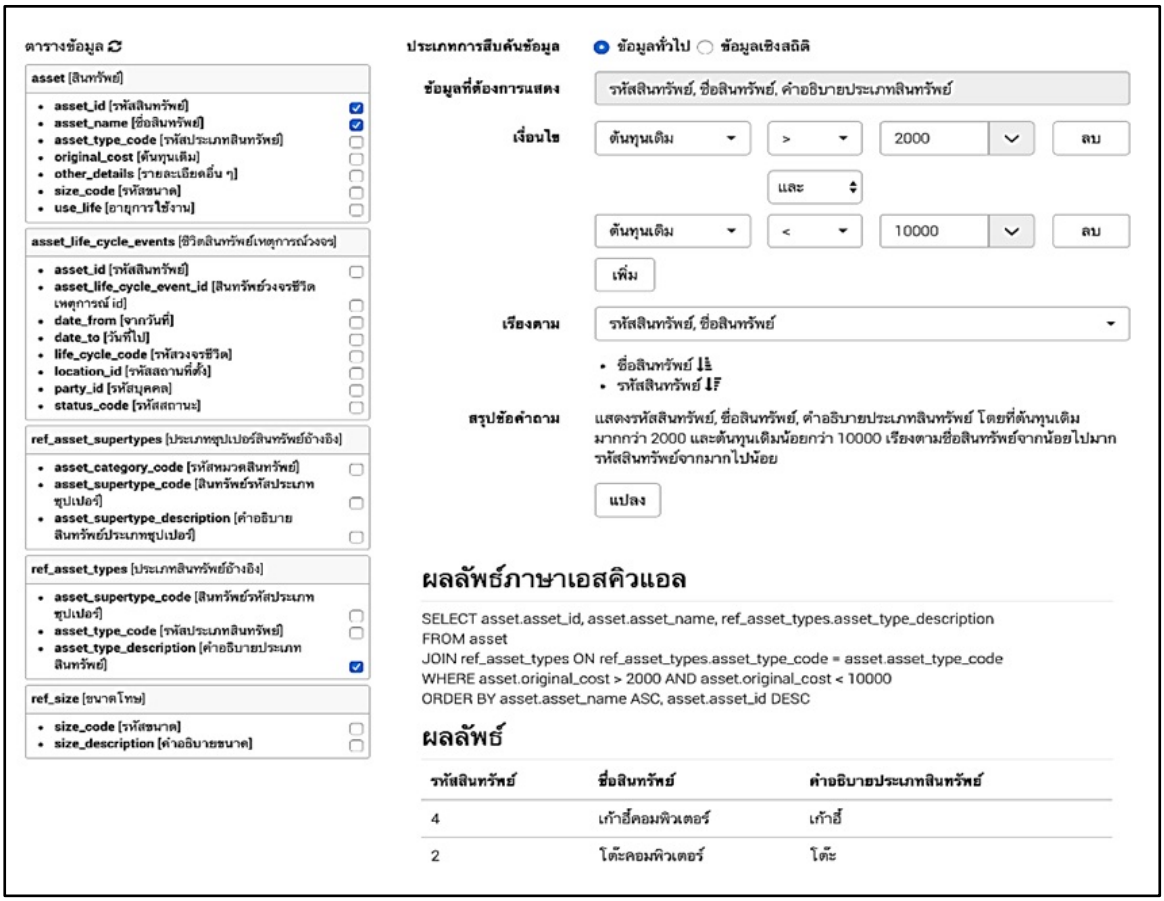


Figure 12 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = General Query (ข้อมูลทั่วไป)
- Display Data = asset_id (รหัสสินทรัพย์), asset_name (ชื่อสินทรัพย์), asset_type_description (คำอธิบายประเภทสินทรัพย์)
- Condition (เงื่อนไข): There are two conditions. These two conditions are combined with “AND” logical operators.
    - First Condition:
        - First Operand (Attribute) = original_cost (ต้นทุนเดิม)
        - Comparison Operator = ″>″
        - Second Operand (Value) = 2000
    - Second Condition:
        - First Operand (Attribute) = original_cost (ต้นทุนเดิม)
        - Comparison Operator = ″<″
        - Second Operand (Value) = 10000
- Order By (เรียงตาม) = asset_name (ชื่อสินทรัพย์) as in an ascending order and asset_id (รหัสสินทรัพย์) as in a descending order.

- Query Text Summary (สรุปข้อคำถาม) = “Display asset_id, asset_name, asset_type_description where original_cost > 2000 AND original_cost < 10000 order by asset_name as ascending asset_id as descending”. The query text summary in Thai is displayed under the “สรุปข้อคำถาม” part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed onto SQL statement using information specified in Table 2. The SQL statement result is displayed under the “ผลลัพธ์ภาษาเอสคิวแอล” part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named “asset” and “ref_asset_types”. A join attribute named “asset_type_code” is used to join data from “asset” and “ref_asset_types.”

Figure 13: Example 5 - Thai QBE Ad Hoc Query for Statistical Query

Figure 13 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = Statistical Query (ข้อมูลเชิงสถิติ)
- Display Data = asset_id (รหัสสินทรัพย์), asset_name (ชื่อสินทรัพย์), MAX(date_from) (MAX(จากวันที่))

- Condition (เงื่อนไข): -
- Group By (แบ่งตาม) = asset_id (รหัสสินทรัพย์)
- Statistical Condition: -
- Order By (เรียงตาม): -
- Query Text Summary (สรุปข้อคำถาม) = “Display asset_id, asset_name, MAX (date_from) grouped by asset_id.” The query text summary in Thai is displayed under the “สรุปข้อคำถาม” part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed onto SQL statement using information specified in Table 2. The SQL statement result is displayed under the “ผลลัพธ์ภาษาเอสคิวแอล” part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named “asset” and “asset_life_cycle_events.” A join attribute named “asset_id” is used to join data from “asset” and “asset_life_cycle_events.”

Figure 14: Example 6 - Thai QBE Ad Hoc Query for Statistical Query

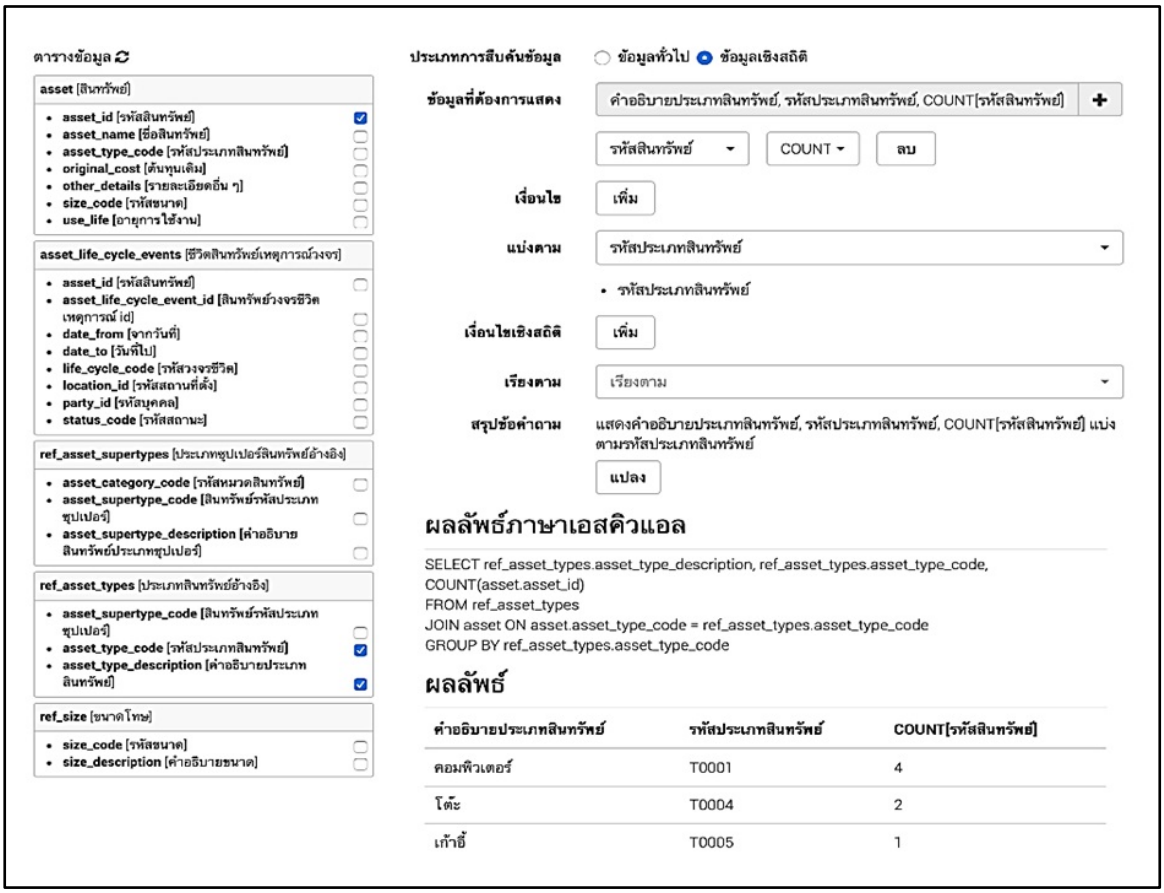


Figure 14 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = Statistical Query (ข้อมูลเชิงสถิติ)
- Display Data = asset_type_description (คำอธิบายประเภทสินทรัพย์), asset_type_code (รหัสประเภทสินทรัพย์), COUNT(asset_id) (COUNT(รหัสสินทรัพย์))

- Condition (เงื่อนไข): -
- Group By (แบ่งตาม) = asset_type_code (รหัสประเภทสินทรัพย์)
- Statistical Condition: -
- Order By (เรียงตาม): -
- Query Text Summary (สรุปข้อคำถาม) = "Display asset_type_description, asset_type_code, COUNT(asset_id) grouped by asset_type_code." The query text summary in Thai is displayed under the "สรุปข้อคำถาม" part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed onto SQL statement using information specified in Table 2. The SQL statement result is displayed under the "ผลลัพธ์ภาษาเอสคิวแอล" part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named "asset" and "ref_asset_types." A join attribute named "asset_type_code" is used to join data from "asset" and "ref_asset_types."

Figure 15: Example 7 - Thai QBE Ad Hoc Query for Statistical Query

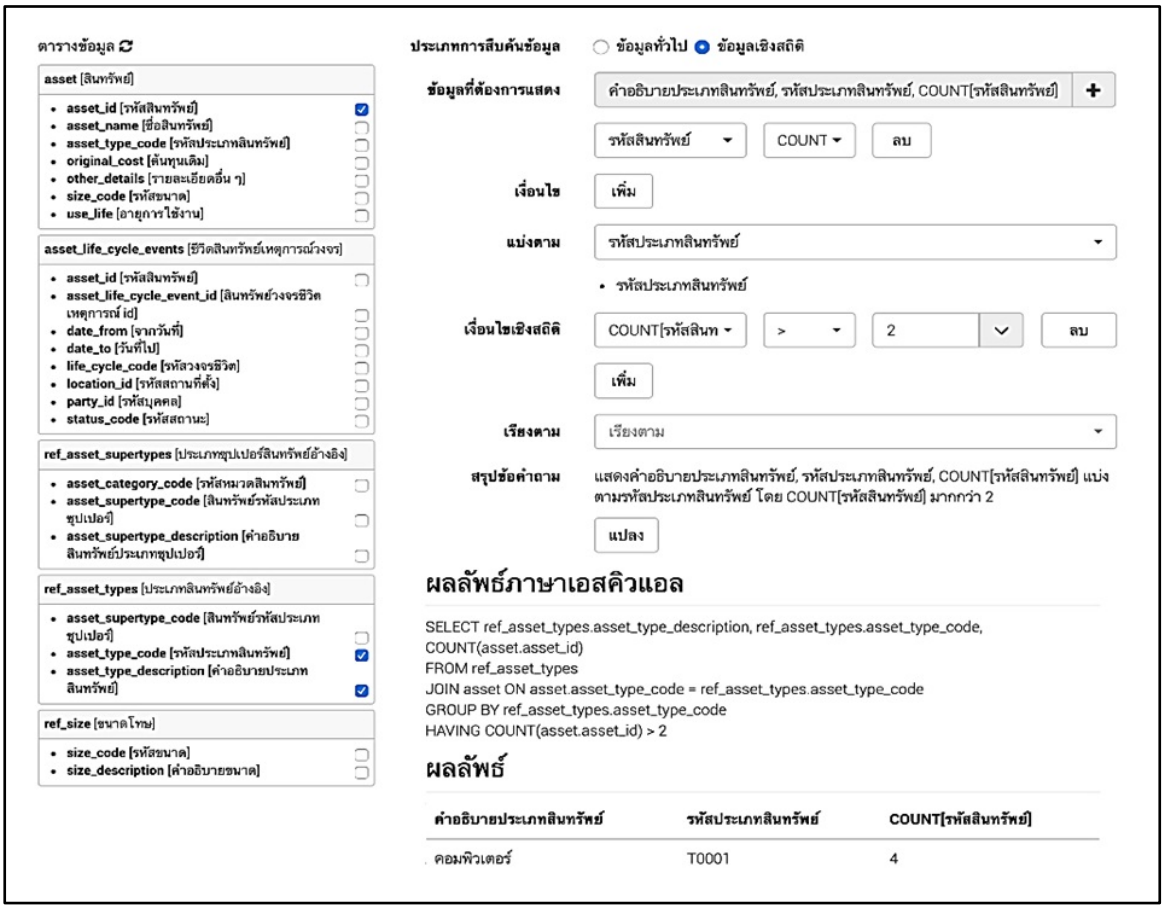


Figure 15 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = Statistical Query (ข้อมูลเชิงสถิติ)
- Display Data = asset_type_description (คำอธิบายประเภทสินทรัพย์), asset_type_code (รหัสประเภทสินทรัพย์), COUNT (asset_id) (COUNT(รหัสสินทรัพย์))

- Condition (เงื่อนไข): -
- Group By (แบ่งตาม) = asset_type_code (รหัสประเภทสินทรัพย์)
- Statistical Condition:
  - First Operand (Aggregation Function) = COUNT (asset_id)
  - Comparison Operator = ″>″
  - Second Operand (Value) = 2
- Order By (เรียงตาม): -
- Query Text Summary (สรุปข้อคำถาม) = “Display asset_type_description, asset_type_code, COUNT(asset_id) grouped by asset_type_code having COUNT(asset_id) > 2.” The query text summary in Thai is displayed under the “สรุปข้อคำถาม” part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed onto SQL statement using information specified in Table 2. The SQL statement result is displayed under the “ผลลัพธ์ภาษาเอสคิวแอล” part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named “asset” and “ref_asset_types.” A join attribute named “asset_type_code” is used to join data from “asset” and “ref_asset_types.”

Figure 16: Example 8 - Thai QBE Ad Hoc Query for Statistical Query

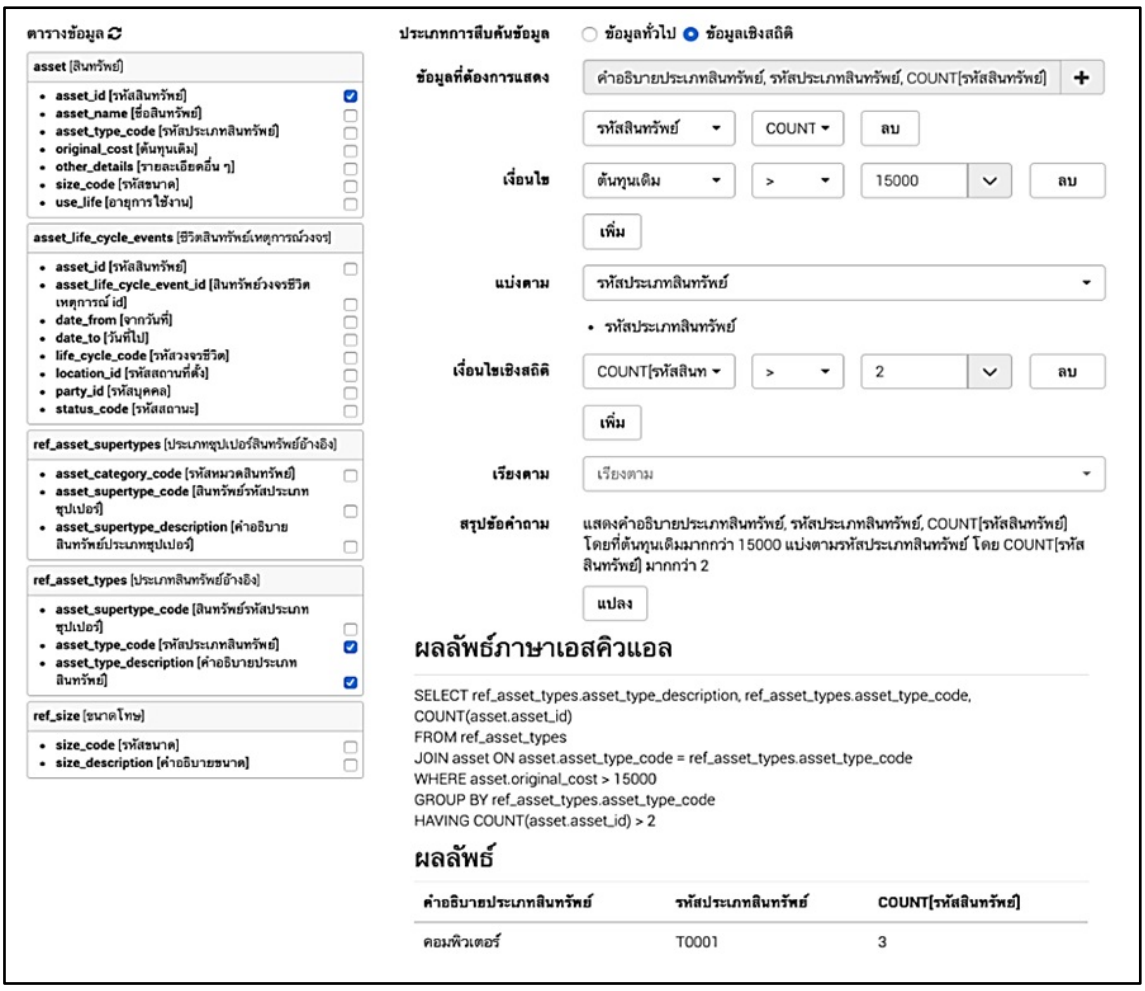


Figure 16 displays ad hoc query using the following searching criteria:

- Query Type (ประเภทการสืบค้นข้อมูล) = Statistical Query (ข้อมูลเชิงสถิติ)
- Display Data = asset_type_description (คำอธิบายประเภทสินทรัพย์), asset_type_code (รหัสประเภทสินทรัพย์), COUNT(asset_id) (COUNT(รหัสสินทรัพย์))
- Condition (เงื่อนไข):
    - First Operand (Attribute) = original_cost (ต้นทุนเดิม)
    - Comparison Operator = ″>″
    - Second Operand (Value) = 15000
- Group By (แบ่งตาม) = asset_type_code (รหัสประเภทสินทรัพย์)
- Statistical Condition:
    - First Operand (Aggregation Function) = COUNT (asset_id)
    - Comparison Operator = ″>″
    - Second Operand (Value) = 2
- Order By (เรียงตาม): -
- Query Text Summary (สรุปข้อคำถาม) = “Display asset_type_description, asset_type_code, COUNT(asset_id) grouped by asset_type_code having COUNT(asset_id) > 2”. The query text summary in Thai is displayed under the “สรุปข้อคำถาม” part.
- SQL Statement (ผลลัพธ์ภาษาเอสคิวแอล): Search criteria in the left hand side frame are transformed to SQL statement using information specified in Table 2. The SQL statement result is displayed under the “ผลลัพธ์ภาษาเอสคิวแอล” part and ad hoc query results are displayed under the output (ผลลัพธ์) part. Attributes in search criteria are from table named “asset” and “ref_asset_types.” A join attribute named “asset_type_code” is used to join data from “asset” and “ref_asset_types.”

5. Conclusion and Future Work

As reported in this paper, a prototype of Thai QBE for ad hoc query was designed and implemented as a web application. The concept used to design Thai QBE was derived from QBE of Microsoft Access DBMS. User interfaces of Thai QBE were designed to support Thai language for easy use.

Thai QBE consists of two main features: Thai QBE General Query and Thai QBE Statistical Query. The Thai QBE General Query is for querying general data and Thai QBE Statistical Query is for querying statistical data. The prototype was implemented and tested with fixed asset database. The experiments in the study prove that the Thai QBE could be a powerful tool to query ad hoc data. Thai QBE can be applied to query ad hoc data from other databases. A developer can just change fixed asset database to a new database to perform ad hoc query, with Section A of research methodology modified as seen functional for a specific purpose.

It should be noted that the Thai QBE is not applicable for database schema containing recursive relationship. The system can be further designed to cover a recursive relationship.

6. The Authors

Areerat Trongratsameethong is a faculty staff member of Computer Science Department, Faculty of Science, Chiang Mai University. She received her Ph.D. in Computer Science from the Mahidol University, Bangkok in 2010. She worked in a business company and software house for many years and gained experiences in software development, system analysis and design, and software requirement analysis. Her teaching and research interest lie in the areas of Fundamentals of Programming, Database System, Object-Oriented Design, and Ontology Design and Development.

Phada Woodtikarn is a senior programmer at Totoit Co., Ltd. He received a master's degree in Computer Science from Chiang Mai University, Chiang Mai, Thailand. His research interest is in the area of Thai QBE for Ad Hoc Query.

7. References

Adamski, J. J. & Finnegan, T. K. (2010). New Perspectives on Microsoft Office Access 2007: Introductory. Florence: Cengage Learning, 105-144.

Hinz et al. (2020). MySQL 8.0 Reference Manual: Schema Object Names. (Online). https://dev.mysql.com/doc/refman/8.0/en/identifiers.html, November 2, 2019.

Necas et al. (2014). python-Levenshtein. (Online). https://pypi.org/project/python-Levenshtein, November 10, 2019.

NLTK Project. (2019). WordNet Interface. (Online). https://www.nltk.org/data.html, November 15, 2019.

Trongratsameethong, A. & Vichianroj, P. (2018). ASQLAG - Automated SQL Assignment Grading System for Multiple DBMSs. *Journal of Technology and Innovation in Tertiary Education*. *1*(1), 41-59.

Wilbert, A. (2010). Navigate & use the Query by Example (QBE) interface in Microsoft Access 2007. (Online). https://ms-office.wonderhowto.com/how-to/navigate-use-query-by-example-qbe-interface-microsoft-access-2007-398173, November 2, 2019.

Williams, B. (2017). Data Model for Asset Management. (Online). http://www.databaseanswers.org/data_models/asset_management/index.htm, November 25, 2019.

Yin, T. (2017). Python-translate (Online). https://pypi.org/project/translate, November 13, 2019.

Zhang, S., Hu,Y., & Bian, G. (2017). Research on String Similarity Algorithm based on Levenshtein Distance. The 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 25-26 March 2017, Chongqing, China. DOI: 10.1109/IAEAC.2017.8054419.