УДК 004.75

Informatics and Mathematical Methods in Simulation
Vol. 7 (2017), No. 1-2, pp. 113-118

POSSIBILITY OF OBTAINING FUNCTIONAL DEPENDENCES FROM DATABASE STRUCTURE

S.I. Grishin[1], L.N. Timoshenko[2]

[1] Odessa National Maritime University,
Mechnikov 34, Odessa, 65404, Ukraine, e-mail: grishin_si@ukr.net
[2] Odesa National Polytechnic University,
1 Shevchenko Str., Odesa, 65044, Ukraine

We study the possibility of obtaining functional dependencies from the scheme of the existing database, as well as the data contained in it. Statistical methods for finding dependencies between data were used to generate probability dependencies. Software implementation of the developed algorithm for generating dependencies provides connection to various data providers. Application testing showed that the correct results can be obtained with a sufficiently large amount of data in the source.

Keywords: functional dependence of attributes, normal form, probability, class C#, SQL query

Introduction

Normalization is an obligatory stage of database design. Knowledge of the violation of certain rules of normal forms allows foresee the occurrence of anomalies and redundancy and apply appropriate measures for their preliminary elimination. Despite the fact that the normalization stage is repeated a huge number of times in various developments, in most cases it is performed manually by an experienced specialist. Normalization process is completely formalized, however it is described in terms of the functional dependences which are, in fact, mathematical representation of some rules and regularities of the real world. Since they are completely determined by the subject domain, there is no possibility to automate their search at the design stage. That is why there are no mechanisms of normalization in modern modeling tools, such as Sybase Power Designer or IBM Rational Software Architect.

At the same time, the use of functional dependencies may be required not only at the design stage of OLTP information systems. For example, to extend an OLTP system or design a business intelligence system without creating a data warehouse, you may need an existing database that is not supported by the developer. Another possible situation is operation with the ROLAP data warehouse. The dimensional model that is used for ROLAP database design, include means of denormalization and data division to enhance speed of request execution. The use of outtrigger tables in the dimensional model bring to partial or complete normalization of dimension tables. Thus, the level of normalization of ROLAP data warehouse can be different. If there are problems with the operation laboriousness in such systems, it may be necessary to check their normalization.

Objective and task

In this work, we investigated the possibility of obtaining functional dependencies from the scheme of the existing database, as well as the data contained in it.

To achieve this goal in the paper we solve such *problems*:

– software has been developed for constructing functional dependencies;
– the program is tested and its scope is determined.

Main part

Finding functional dependencies in the database is not a trivial task. First, the regularities found in particular relationship variable not always valid for whole relation. Secondly, required functional dependences are not reflected at design of the database, so the corresponding restrictions of integrity are not respected. Therefore, the studied tables can contain mistakes. The way out of this situation is the probabilistic approach in the search for functional dependencies [1].

The developed software is designed to obtain probabilistic functional dependencies from the data contained in the tables. Statistical methods intended for search of dependences between data are used for generation of probabilistic functional dependences. Such a search is carried out in each separate data source. Then obtained data are combined into a general scheme by searching for the minimum coverage for the found set of probabilistic functional dependencies. Further, results that do not meet the requirements of the obtained scheme are eliminated.

Let relation R be given, X is a subset of its attributes, and A is an attribute of this relation. A probabilistic functional dependence is the triplet $< X, A, p >$, where p indicates the probability of the existence of a functional dependence $X \to A$. It is proposed to calculate the probability p for all possible combinations X, A of the R relation. We excepted from calculation the functional dependencies that already found, as well as those sets X that are supersets of some potential keys. If probability of dependence existence is greater than some parameter $p0$, the dependence is included in the model and analyzed together with previously constructed dependencies.

The following algorithm is proposed to find the probability p of the existence of the functional dependence of the attribute A on the set of attributes X:

1. The most common value Va of attribute A is found for each unique value Vx of the set of attributes X.

2. The probability of the functional dependence $X \to A$ among the tuples with the value Vx of the X attributes is calculated. It is calculated as the ratio of the number of tuples in which Vx corresponds to the value Va, to the number of tuples with the Vx value of the X attribute:

$$P(X \to A, Vx) = \frac{|Vx, Va|}{|Vx|},$$

where $|Vx, Va|$ is the number of tuples in which the value Va corresponds to the value Vx, $|Vx|$ is the number of tuples with the Vx value of the X attribute.

3. The probability p is calculated as the average of all probabilities obtained in step 2.

Formally, if Dx – a set of unique values of a set of attributes X, then the probability of existence of the functional dependence is described as follows:

$$P(X \to A, R) = \frac{\sum_{Vx \in Dx} P(X \to A, Vx)}{|Dx|}.$$

The application for constructing functional dependencies in the above ways and dependencies analysis is performed in C # and the .NET 4.0 platform.

After connecting to the database, the user is asked to select the tables he wants to analyze.

Next, he has to specify the minimum probability value, which is significant for probabilistic functional dependence existence.

The following information is displayed for each table:

1. Name of the table being analyzed.
2. A list of all the probabilistic functional dependencies found and their corresponding probabilities.
3. Graph of functional dependencies.
4. The arrows with solid lines in this graph show the functional dependencies obtained as a result of the analysis of the database schema. The dotted lines correspond to probabilistic functional dependencies calculated on the basis of the data contained in the table.
5. Estimated normal form.

The following classes were created to represent information about the structure of the database:

Database – represents the database. Contains its name and a list of tables.

DatabaseTable – represents the database table. Contains the name of the table, the name of the schema that contains the table, and lists of potential and foreign keys.

Column – represents the table attribute. Contains the name and type of the attribute.

Constraint – base class for database integrity constraints. Contains the name and type of the constraint (potential or foreign key).

ForeignKey – represents the foreign key of the table. Contains the referencing attribute, as well as information about the table and the potential key, to which this attribute refers. This class inherits the Constraint class.

CandidateKey – represents the potential key of the table. Contains a list of attributes that are part of this potential key. Inherits the Constraint class.

An abstract class called DatabaseProxy was developed to obtain information about the structure of the database. It contains the connection string to the database under investigation and the following methods for obtaining the data required for analysis:

List <DatabaseTable> GetDatabaseTables() – returns a list of database tables, including only information about table name and schema name.

Database GetDatabaseSchema(List <DatabaseTable> tables) – the method returns full information about the database and those tables that are specified in the tables parameter. It includes a list of attributes and their types, primary and foreign keys, unique indexes.

bool CheckIfColumnIsUnique(DatabaseTable table, Column column) – checks whether the values of the specified attribute are unique. Used to search for potential keys not listed in the conceptual diagram.

double GetFDProbability(DatabaseTable table, List<Column> determinant, Column dependentColumn) – receives the probability of the existence of a functional relationship with the determinant and the dependent part specified in the determinant and dependentColumn parameters.

Various implementations of this abstract class can be used to study databases presented in different DBMSs. We use Microsoft SQL Server. The appropriate class is called MSSQLServerProxy. Let us consider its implementation in more detail.

The queries to the service tables of the INFORMATION_SCHEMA schema were created to get information about the database schema. After receiving necessary data on a database structure, we pass to the analysis of its contents. The DatabaseDataAnalyzer class is responsible for it. This class realizes search of all probable functional dependences having only one attribute as a determinant. The main analysis algorithm of one table is shown below:

1. The list of all non-key table attributes is formed.

2. Attributes which all values are unique are deleted from this list. These attributes are marked as possible potential keys that are not reflected in the database schema.

3. All possible probable functional dependences are formed. The determinant of each of them will be the element of the set obtained on step 2. Dependent part will be any attribute of the table, other than a determinant.

4. For each of the probabilistic functional dependencies obtained from the data contained in the database, the probability of their existence is calculated.

5. Probable functional dependences are selected if probability exceeds some value of $p0$. This value is algorithm parameter.

It is necessary to be able to quickly check the set of attribute values for uniqueness, and also to calculate the probability of existence of functional dependencies between some sets of attributes for effective operation of the algorithm.

To implement such checks, it is proposed to generate an SQL-query that solves the task and transfer it for execution to an SQL server. This approach allows you to efficiently examine remote databases. Import database content to local storage can take a long time. At an execution stage we don't know the schema of the database under investigation. It will be necessary to generate a separate query for check of uniqueness of values or computation of probable functional dependence for each set of attributes

Here are some examples of SQL queries that solve the set tasks. The request to verify the uniqueness of the values of the attribute A of the TABLE table of the SCHEMA schema will look like this

```
SELECT [A]
FROM [SCHEMA].[TABLE]
GROUP BY [A]
HAVING COUNT(1) >=2
```

The calculation of probabilistic functional dependencies requires a series of calculations. Let it be necessary to calculate the probability of the existence of a functional dependence of the attribute B on the attribute A in the table 1. The SQL query that solves this problem has the following syntax:

```
select AVG(probability)
from
   (select
      (select MAX(count)/CAST(SUM(count) As REAL)
                  from
            (select COUNT(y.B) as count
                  from
                              (select t1.B
                           from TABLE as t1
                              where t1.A = t.A and
                  A is not null and
                  B is not null ) as x
                        join
                              (select t1.B
                               from TABLE as t1
                               where t1.A = t.A and
                  A is not null and
                  B is not null) as y
                              on x.B = y.B
                        group by x.B
         ) as tempTable
```

```
    ) as probability
from TABLE as t
where A is not null and B is not null
group by A
) as distinctProbabilities.
```

The Microsoft sample database Adventure Works [2] was used for testing of the created application. Examples of testing are given in the table 1.

Table 1.

Examples of Adventure Works database testing

Table name	Number of records	Amount of dependences	Normalization assessment
CreditCard	19118	0	3NF
Address	19614	2	2NF
EmployeePayHistory	316	3	2NF

The program found no probabilistic functional dependencies, except dependencies from the primary key. It correctly defined the normal form of the CreditCard table.

There are dependencies of the city and the state ID from the zip code in the Address table. That is quite logical – the same zip code always points to a certain state and city, while in one city there can be many different postal codes. The program successfully defined both dependences, displayed them on the chart and in specified that the table is only in the second normal form.

For the EmployeePayHistory table, all the dependencies obtained were found to be incorrect. The first functional dependence in terms of the subject area indicates that each employee is inclined to receive a salary with a certain frequency convenient for him. Although this statement is rather plausible, it is not a restriction on the integrity of the database. In the process of work, an employee or an employer may revise an employment contract. Thus, a false result was obtained though it was based on the correct data. Unfortunately, there is no way to identify such cases programmatically – you need a manual analysis by an expert. The second probabilistic functional dependence (the periodicity of payment from its hourly rate), unlike the first one, has no semantic meaning. The error is due to insufficient data in this case. The number of accidental coincidence of two attributes values increases in case of a small amount of records. That is shown especially strongly in this case as the “periodicity of payment” attribute accepts only 2 different values in all tables. This error can be eliminated by some modification of the calculation method for the case of a small number of records, or by searching for a more suitable data source. There is the dependence of the date of changing the line on the frequency of payment in the third case. This dependence also does not make any sense. The detailed analysis of basic data showed that the high percent of identical values of “date of change” attribute is a cause of error in this case. 277 records of 316 have identical values. This situation can also be resolved programmatically by means of a preliminary analysis of the researched data.

Conclusion

A large amount of data is needed to obtain a qualitative result using the probabilistic approach in the search for functional dependencies.

In case of dominance of the correct data the considered algorithms are capable to process even the sources containing erratic or inexact information.

References

1. Wang, D.Z. Functional dependency generation and applications in pay-as-you-go data integration systems [Electronic Resource]**.** / D.Z. Wang, L. Dong, A.D. Sarma, M. Franklin, A. Halevy. - Mode of access: http://webdb09.cse.buffalo.edu/papers/Paper18/webdb09.pdf.
2. AdventureWorks. Data Dictionary. Dataedo. 2017-05-30 [Electronic Resource]**.** - Mode of access: https://dataedo.com/download/AdventureWorks.pdf

МОЖЛИВІСТЬ ОТРИМАННЯ ФУНКЦІОНАЛЬНИХ ЗАЛЕЖНОСТЕЙ ЧЕРЕЗ СТРУКТУРУ БАЗИ ДАНИХ

С.І. Гришин[1], Л.М. Тимошенко[2]

[1] Одеський національний морський університет,
Мечникова 34, Одеса, 65404, Україна, e-mail: grishin_si@ukr.net
[2] Одеський національний політехнічний університет,
просп. Шевченка, 1, Одеса, 65044, Україна

Наведено результати дослідження можливості отримання функціональних залежностей зі схеми існуючої бази, а також даних, що містяться в ній. Для генерації ймовірнісних функціональних залежностей використані статистичні методи пошуку залежностей між даними. Програмна реалізація розробленого алгоритму генерації залежностей забезпечує підключення до різних постачальників даних. Тестування додатка показало, що коректні результати можна отримати при досить великому обсязі даних в джерелі.
Ключові слова: функціональна залежність атрибутів, нормальна форма, ймовірність, клас C#, запит SQL

ВОЗМОЖНОСТЬ ПОЛУЧЕНИЯ ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ ИЗ СТРУКТУРЫ БАЗЫ ДАННЫХ

С.И. Гришин[1], Л.Н. Тимошенко[2]

[1] Одесский национальный морской университет,
Мечникова 34, Одесса,65404, Украина, e-mail: grishin_si@ukr.net
[2] Одесский национальный политехнический университет,
просп. Шевченко, 1, Одесса, 65044, Украина

Приведены результаты исследования возможности получения функциональных зависимостей из схемы существующей базы, а также данных, содержащихся в ней. Для генерации вероятностных функциональных зависимостей использованы статистические методы поиска зависимостей между данными. Программная реализация разработанного алгоритма генерации зависимостей обеспечивает подключение к различным поставщикам данных. Тестирование приложения показало, что корректные результаты можно получить при достаточно большом объеме данных в источнике.
Ключевые слова: функциональная зависимость атрибутов, нормальная форма, вероятность, класс C#, запрос SQL