METHOD ARTICLE

REVISED A novel data storage logic in the cloud [version 3; peer review: 2 approved, 1 not approved]

Bence Mátyás¹, Gabriela Bautista², Máté Szarka^{3,4}, Gábor Járvas⁵,
Gábor Kusper⁶, István Argay^{7,8}, Alice Fialowski⁹, György Mátyás¹⁰

¹Universidad Politécnica Salesiana, Rumichaca y Morán Valverde s/n, Quito, Ecuador

²Faculty of Agricultural and Food Sciences and Environmental Management, University of Debrecen, Debrecen, Hungary

³Medical and Health Sciences Center, Research Centre for Molecular Medicine, University of Debrecen, Debrecen, Hungary

⁴Vitrolink Biotechnological Researching, Development, Servicing and Trading Limited Liability Company, Debrecen, Hungary

⁵Hungarian Academy of Sciences MTA-PE, Translational Glycomics Group, University of Pannonia, Veszprem, Hungary

⁶Computer Science Department, Eszterhazy Karoly University, Eger, Hungary

⁷Department of Obstetric and Gynecology, UD MSHC, University of Debrecen Medical Center, Debrecen, Hungary

⁸IRCAD France Laparoscopic Training center, Strasbourg, France

⁹Institute of Mathematics and Informatics, University of Pécs, Pécs, Hungary

¹⁰Statik Kontroll Ltd., H3950 Sáropatak, Rákóczi utca 33, Hungary

V3 First published: 21 Jan 2016, 5:93
<https://doi.org/10.12688/f1000research.7727.1>

Second version: 29 Mar 2016, 5:93
<https://doi.org/10.12688/f1000research.7727.2>

Latest published: 15 Aug 2017, 5:93
<https://doi.org/10.12688/f1000research.7727.3>

Abstract

Databases which store and manage long-term scientific information related to life science are used to store huge amount of quantitative attributes. Introduction of a new entity attribute requires modification of the existing data tables and the programs that use these data tables. A feasible solution is increasing the virtual data tables while the number of screens remains the same. The main objective of the present study was to introduce a logic called Joker Tao (JT) which provides universal data storage for cloud-based databases. It means all types of input data can be interpreted as an entity and attribute at the same time, in the same data table.

Keywords

Joker Tao , NoSQL , Cloud , Database , Life science , Physical data table , Virtual data table , RDBMS

Open Peer Review

Approval Status

	1	2	3
version 3 (revision) 15 Aug 2017			view
version 2 (revision) 29 Mar 2016	view
version 1 21 Jan 2016	view		view

1. **Jan Lindström**, MariaDB Corporation, Espoo, Finland

2. **Kavita Sunil Oza**, Shivaji University, Kolhapur, India

3. **Mansaf Alam**, Jamia Millia Islamia, New Delhi, India

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Bence Mátyás (bmatyas@ups.edu.ec)

Author roles: **Mátyás B:** Data Curation, Formal Analysis, Investigation, Methodology, Project Administration, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Bautista G:** Formal Analysis, Software, Supervision; **Szarka M:** Methodology, Software, Supervision, Validation; **Járvás G:** Methodology, Software, Supervision; **Kusper G:** Methodology, Software; **Argay I:** Data Curation, Formal Analysis; **Fialowski A:** Data Curation, Formal Analysis, Methodology, Software; **Mátyás G:** Conceptualization, Funding Acquisition, Methodology, Software

Competing interests: No competing interests were disclosed.

Grant information: The first version of JT is a Hungarian product which was developed in 2008 (R.number: INNO-1-2008-0015 MFB-00897/2008) thanks to an INNOCSEK European Union application.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2017 Mátyás B *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Mátyás B, Bautista G, Szarka M *et al.* **A novel data storage logic in the cloud [version 3; peer review: 2 approved, 1 not approved]** F1000Research 2017, 5:93 <https://doi.org/10.12688/f1000research.7727.3>

First published: 21 Jan 2016, 5:93 <https://doi.org/10.12688/f1000research.7727.1>

REVISED Amendments from Version 2

Taking the referees' advice:

We extended the Introduction chapter to clarify the problems our proposed method tries to solve.

Database designing rules applied by us and more examples were added to the method chapter in order to describe the method more clearly.

In the Introduction and Discussion we highlighted the importance of the developed method and how it improves the state of the art.

Dataset 1 has been added to show the data storage structure in JT logic based databases.

See referee reports

Introduction

Databases which store and manage long-term scientific information related to life science are used to store huge amount of quantitative attributes. This is specially true for medical databases^{1,2}. One major downside of these data is that information on multiple occurrences of an illness in the same individual cannot be connected^{1,3,4}. Modern database management systems fall into two broad classes: Relational Database Management System (RDBMS) and Not Only Structured Query Language (NoSQL)^{5,6}. The objects in the relational database are grouped based on their type, format and number of identical attributes. The data tables in the relational databases are normalized. Join operations are used in the relational databases when combining information based on a matching value for a primary key and foreign key across multiple data tables. In several cases the query became slower in the relational database because of the larger schema and increased number of data tables that need to be joined⁷. A significant amount of time is spent by database experts developing a common structure for the data coming from different sources and not on the analysing processes themselves.

The introduction of a new entity attribute requires the introduction of additional data tables or modification of the existing data tables. In the NoSQL databases, the development such as creating simple query is more complex because there is no standard query language and there are limits to the operations (for example, there is no join operation in MongoDB)⁷. The primary goal of our developed method was to store and manage each data in a single data table while the data handling processes can be completed by SQL commands. Our method may be useful in an environment where the schema is constantly changing as a result of adding new devices or tools that generates new types of attributes. This solution contributes to the interoperability between the relational and NoSQL systems where converting application usage is unnecessary. JT can be defined as a NoSQL engine on an SQL platform.

Methods

The technical environment is Oracle Application Express (Apex) 5.0 cloud-based technology. Workstation: OS (which is indifferent) + internet browser (Chrome). Firstly, we demonstrate an example for a simplified relational database (Figure 1). Following this, the presented data tables have been modified step by step. At the end of these steps, each data from the presented database will be stored in a single data table using JT logic.

The database designing rules we applied can be summed up as follows: .

Rule 1: Creating a data table with four columns

Specification of the physical data table structure was determined with *-ID* (num) as the identifier of the entity, which identifies the entity between the data tables (not only in the given data table); *-ATTRIBUTE* (num) is the identifier of the attribute; *-SEQUENCE* (num) which is used in the case of a vector attribute; and *-VALUE* (VARCHAR2) which is used for storing values of the attributes.

Figure 1. Example for a traditional (relational) data storage structure.

Note: The codes which are stored in the *Attribute* column are also defined, sooner or later, in the *ID* column. At that time the attribute becomes an entity. In every case, the subjectivity determines the depth of entity-attribute definition in the physical data table.

Rule 2: Defining basic relationships

The first step is the technical data storage. In Figure 2, basic relationships will be defined which describe the names of the attributes (can be interpreted as columns in the usual relational databases), type of relationships (belonging to the structure) and virtual data tables (can be interpreted as “belonging to the data table” in the usual relational databases). The *Name* is defined as an entity and an attribute at the same time as the code 1 appears in both columns in the ID and in the Attribute as well. Following this two entities are defined: the “Belonging to the virtual data table”

with ID=2, and “Belonging to the structure” with ID=3. In these records the same code 1 appears in the Attribute column which means we named the entities (Name was identified with code 1). With this technique we can name the new entities in the database. Once the code appearing in the ID column is stored in the Attribute column, the data will be interpreted as an attribute (like in the case of *Name*).

Rule 3: Entity storage

An entity is defined with the records having the same ID values (Figure 3). These identifiers can be any natural number that has not already been used in the ID column. In this example we introduced the *Address*. The records with the ID=10 identify the *Address* as an entity. The Address was determined by the previously defined attributes: Firstly we named this entity

☐	<u>Id</u> ↑	<u>Attribute</u>	<u>T Sequence</u>	<u>T Value</u>
☐	1	1	1	Name
☐	2	1	1	Belonging to the virtual data table
☐	3	1	1	Belonging to the structure

row(s) 1 - 10 of 90 ▾ Next ⓘ

[Print](#)

Add Row

Figure 2. Storage of basic attributes.

Personal_record_Update					Cancel	Delete	Apply Changes
☐	<u>Id</u> ↑	<u>Attribute</u>	<u>T Sequence</u>	<u>T Value</u>			
☐	1	1	1	Name			
☐	2	1	1	Belonging to the virtual data table			
☐	3	1	1	Belonging to the structure			
☐	10	1	1	Address			
☐	10	3	1	11			
☐	10	3	2	12			
☐	10	3	3	13			
☐	10	3	4	14			
☐	10	3	5	15			
☐	10	3	6	16			

row(s) 1 - 10 of 90 ▾ Next ⓘ

[Print](#)

Add Row

Figure 3. Entity storage.

with the Attribute=1 that means *Name*. Secondly the “Belonging to the structure” was clarified (with Attribute=3). We introduced some values of the Address in a list. Every record of the Address entity belongs to different structures in this example. The TSequence column is used to make a list in this concept.

Rule 4: Attribute storage

The codes which were previously introduced into the ID column, now are transferred to Attribute column (Figure 4). The values of these identifiers can be any natural number that has not already been used in the Attribute column.

Each attribute is identified in the Attribute column. In this example the following contexts can be read out related to the entity identified with ID=1001: -The value of the “belonging to the virtual data table” attribute (code 2) is Personal data table (code 31); -First name (code 32) is Richard; -Second name (code 33) is Jones; -Date of birth (code 33) is 01/02/1963; -Social security number (code 34) is 33325333; -Nationality (code 25) is American. The codes (namely 2,31,32,33,34,35) have to be stored sooner or later in ID column. At that time these attributes become entities and are defined by other attributes (eg. the “name” of the entity identified with 82 ID value is Personal insurance ID; the attribute called “name” was defined earlier in ID column see Figure 3 and now it is applied in the attribute column as an entity attribute).

Rule 5: Defining data tables

The attributes are assigned to each virtual data table using a previously introduced attribute called “belonging to the virtual data table”. In the usual relational databases, the entities are grouped by different data tables. For the same purpose, we introduced “virtual data tables” in our single big data table. The code 2 in the Attribute column identifies the belonging to the table. The different entities that belong to the same data table have to own the same value in the Tvalue column (Figure 5).

From this example, the following context can be read out: The entities identified with 1001 and 1002 ID values belong to the same virtual data table.

With these steps the developer can design one data table to store each entity, attribute and value in a relational database. Obviously, the user doesn’t see these virtual tables. These appear for the user in a form as usual data tables in the Oracle Apex platform. Only the names of the attributes appear for the user instead of their codes. A query can be created in the same way like in any relational database. In the background: the attribute of the given virtual data table is queried. This attribute is array-type and means increasing number of TSequence. The user can see only the names of the TSequence and TValue (Name of the attributes) like in the following example (Figure 6).

Personal_record_Update				
<input type="button" value="Cancel"/> <input type="button" value="Delete"/> <input type="button" value="Apply Changes"/>				
☐	<u>Id</u>	<u>Attribute</u>	<u>T Sequence</u>	<u>T Value</u>
☐	82	1	1	Personal insurance ID
☐	1001	2	1	30
☐	1001	31	1	Richard
☐	1001	32	1	Jones
☐	1001	33	1	01/02/1963
☐	1001	34	1	33325333
☐	1001	35	1	USA
☐	1002	2	1	30
☐	1002	31	1	Mary
☐	1002	32	1	Twain
<input type="button" value="Print"/> Previous row(s) 61 - 70 of 90 Next <input type="button" value="Add Row"/>				

Figure 4. Attribute storage.

Personal_record_Update

Cancel

Delete

Apply Changes

☐	Id	Attribute	T Sequence	T Value
☐	1	1	1	Name
☐	2	1	1	Belonging to the virtual data table
☐	3	1	1	Belonging to the structure
☐	10	1	1	Address
☐	10	3	1	11
☐	10	3	2	12
☐	10	3	3	13
☐	10	3	4	14
☐	10	3	5	15
☐	10	3	6	16

row(s) 1 - 10 of 90

Next

Print

Add Row

Personal_record_Update

Cancel

Delete

Apply Changes

☐	Id	Attribute	T Sequence	T Value
☐	82	1	1	Personal Insurance ID
☐	1001	2	1	30
☐	1001	31	1	Richard
☐	1001	32	1	Jones
☐	1001	33	1	01/02/1963
☐	1001	34	1	33325333
☐	1001	35	1	USA
☐	1002	2	1	30
☐	1002	31	1	Mary
☐	1002	32	1	Twain

Previous

row(s) 61 - 70 of 90

Next

Print

Add Row

Figure 5. Belonging to the virtual data table.

new entity

home

Clients

?

Q-

Go

Actions

	Row_ID	Name
	1000226	Törzsök Zoltán
	1000246	Miczán Ferencné
	1000276	Diamond Aranybánya Panzió
	1000286	Önkormányzat Telkibánya
	1000124	ABACSER Coop Kft.
	1000146	Weszelovszkiné Vajda Mária
	1000176	Takács Béla
	1000186	Sellweger Kft.
	1000135	Tóth Gellértné
	1000196	Dezsi Józsefné

Home

Application 244

Edit Page 18

Create

Session

Caching

View D

Figure 6. User screen.

Oracle Apex automatically supply each record with row IDs. The above described method can be applied manually. For the automatic conversion (for primarily non cloud-based applications) we created a Java code below 7:

```
public static String getEntityName ( )
throws Exception
{
    Connection conn = broker.getConnection ( );
    PreparedStatement pstmt =
    conn.prepareStatement ("select *from joker");
    ResultSets = pstmt.executeQuery ( );
    inti = 0;
    while (rs.next ( )) {
        i++;
    }
    System.out.println ("number of records:" + i);
    broker.freeConnection (conn);
    return "";
}

public static void insert JokerRow
(Integer GROUP_ID, Integer UNIQ_ID,
Integer FIELD_ID, Integer ARRAY_INDEX,
String SEEK_VALUE, String FIELD_VALUE)
throws Exception {
    if (GROUP_ID == null) pstmt.setNull (1, 2);
    else pstmt.setInt (1, GROUP_ID.intValue ( ));
    if (UNIQ_ID == null) pstmt.setNull (2, 2);
    else pstmt.setInt (2, UNIQ_ID.intValue ( ));
    if (FIELD_ID == null) pstmt.setNull (3, 2);
    else pstmt.setInt (3, FIELD_ID.intValue ( ));
    if (ARRAY_INDEX == null) pstmt.setNull (4, 2);
    else pstmt.setInt
    (4, ARRAY_INDEX.intValue ( ));
    if (SEEK_VALUE == null)
    pstmt.setNull (5, 12);
    else pstmt.setString (5, SEEK_VALUE);
    if (FIELD_VALUE == null) pstmt.setNull
    (6, 12); else pstmt.setString
    (6, FIELD_VALUE); pstmt.execute ( );
}

public static void readFile ( )
throws Exception
{
    File f = new File ("data.txt");
    BufferedReader br = new BufferedReader
    (new FileReader (f));
    while (br.ready ( )) {
        String line = br.readLine ( );
        int GROUP_ID = Integer.parseInt
        (line.substring (0, 10));
        int UNIQ_ID = Integer.parseInt
        (line.substring (11, 21));
        int ARRAY_INDEX = Integer.parseInt
        (line.substring (22, 32));
        String FIELD_VALUE = line.length ( ) > 32 ?
        line.substring (33, line.length ( )): " ";
        insertJokerRow (Integer.valueOf (GROUP_ID),
        Integer.valueOf (UNIQ_ID), null,
        Integer.valueOf (ARRAY_INDEX),
        null, FIELD_VALUE);
    }
    br.close ( );
}
```

Results

The resulting table structure is called JT structure (Figure 7). The result from automatic conversion is a physical data table which uses 6 columns. In cloud, Oracle Apex automatically add row IDs and we introduced “belonging to the virtual data table” attribute instead of Group IDs. In cloud we prefer to use only 4 columns to store each data in a database.

Id	Attribute	T Sequence	T Value
1	1	1	Name
2	1	1	Belonging to the virtual data table
3	1	1	Belonging to the structure
10	1	1	Address
10	3	1	11
10	3	2	12
10	3	3	13
10	3	4	14
10	3	5	15
10	3	6	16
10	3	7	17
10	3	8	18
10	3	9	19
11	1	1	Country
12	1	1	ZIP
13	1	1	House number
14	1	1	Street
15	1	1	City
16	1	1	Telephone
17	1	1	Email
18	1	1	Address type ID
19	1	1	Personal or insurance ID
20	1	1	Address type
20	3	1	21
21	1	1	Address type name
30	1	1	Personal
30	3	1	31
30	3	2	32
30	3	3	33
30	3	4	34
30	3	5	35
31	1	1	First name
32	1	1	Second name
33	1	1	Date of birth
34	1	1	Social security number
35	1	1	Nationality
40	1	1	Bank account
40	3	1	41
40	3	2	42
40	3	3	43
41	1	1	Bank account number
42	1	1	Name of the bank
43	1	1	Personal or insurance ID
50	1	1	Insurance company
50	3	1	51
51	1	1	Insurance company name
60	1	1	Personal BNO
60	3	1	61
60	3	2	62
61	1	1	BNO ID
62	1	1	Personal insurance ID
70	1	1	Personal Insurance
70	3	1	71
70	3	2	72
71	1	1	Personal ID
72	1	1	Insurance ID
80	1	1	Personal intervention
80	3	1	81
80	3	2	82
81	1	1	Intervention ID
82	1	1	Personal insurance ID
1001	2	1	30
1001	31	1	Richard
1001	32	1	Jones
1001	33	1	01/02/1963
1001	34	1	33325333
1001	35	1	USA
1002	2	1	30
1002	31	1	Mary
1002	32	1	Twain
1002	33	1	01/06/1989
1002	34	1	22255211
1002	35	1	USA
1003	2	1	30
1003	31	1	Béla
1003	32	1	Kovács
1003	33	1	04/05/1986
1003	34	1	555212333
1003	35	1	Hungarian
1004	2	1	30
1004	31	1	István
1004	32	1	Kovács
1004	33	1	08/04/1990
1004	34	1	325155453
1004	35	1	Hungarian
3001	1	1	Allianz
3001	2	1	50
10001	2	1	70
10001	71	1	1001
10001	72	1	3001

Figure 7. Physical data storage structure.

The JT logic-based databases can be defined using primitive relation scheme known as a three-tuple according to Paredaens (1989)⁹ concept:

$$PRS = (\omega, \delta, dom)$$

where

ω is a finite set of attributes, in our case, it is the set of entities from the ATTRIBUTES virtual data table.

δ is a finite set of entities, in our case, it is a set of virtual records.

$$dom : \omega \rightarrow \delta$$

is a function that associates each attribute to an entity; it can be interpreted as a predefined set of attributes called “1:N registry hive”. This function is used to maintain the entities in the virtual data tables.

A relation scheme (or briefly a relation) is a three-tuple $RS=(PRS,M,SC)$

where

PRS is a primitive relation scheme; M is the meaning of the relation. This is an informal component of the definition, since it refers to the real world and since we will describe it using a natural language. SC is a set of relation constraints. From the JT physical data table, the following definitions can be read out:

- Virtual record is set of the physical records which have the same ID value.
- Virtual data table is set of the virtual records which have the same value of the “belonging to the virtual data table” attribute⁶.

Thesis: In the JT structure, each attribute needs only one index for indexing in the database.

Proof using mathematical induction: It is obvious the statement is true for the case of one record stored in a data table (according to the RDBMS structure where the developers use more indexes to indexing more attributes). In this case the data table appears as shown in Figure 8. Index= attribute (num) + value (varchar 2) In view of entity, an ID (numerical) index is also used in JT logic-based systems. This ID does not depend (no transitive dependency) on any attribute. Thus, the entities of the virtual data tables meet the criteria of the third normal form (Figure 9).

The modes of the expansion of a data table are: -input new entity (Figure 10); -input new attribute (Figure 11); -input new virtual data table (Figure 12).

The indexing is correct in case of n+1 record expansion also. With JT logic the user is able to use only one physical data table to define each virtual data table in a database. Therefore, since

Index Name	E_CLIENT_TL_IDX1		
Index Type	NORMAL		
Table Owner	AP_301		
Table Name	E_CLIENT_TL		
Table Type	TABLE		
Uniqueness	NONUNIQUE		
Compression	DISABLED		
Prefix Length	-		
Tablespace Name	APEX_87556711811556733		
Status	VALID		
Last Analyzed	10/29/2015 05:45:38 PM		

Index Columns		
COLUMN_NAME	COLUMN_EXPRESSION	COLUMN_POSITION
ATTRIBUTE	-	1
T_VALUE	-	2

EDIT	PRIMARY_ID	ID	ATTRIBUTE	T_SEQUENCE	T_VALUE
	1000009	1000003	10024	4	Sárospatak

Figure 8. Indexing a record.

Object Details	Statistics	SQL
Disable	Drop	Rebuild

Index Name	E_CLIENT_IDX1		
Index Type	NORMAL		
Table Owner	AP_301		
Table Name	E_CLIENT		
Table Type	TABLE		
Uniqueness	NONUNIQUE		
Compression	DISABLED		
Prefix Length	-		
Tablespace Name	APEX_87556711811556733		
Status	VALID		
Last Analyzed	09/29/2015 10:00:41 PM		

Index Columns		
COLUMN_NAME	COLUMN_EXPRESSION	COLUMN_POSITION
ID	-	1

EDIT	PRIMARY_ID	ID	ATTRIBUTE	T_SEQUENCE	T_VALUE
------	------------	----	-----------	------------	---------

Figure 9. ID usage.

PRIMARY_ID	ID	ATTRIBUTE	T_SEQUENCE	T_VALUE
1000009	1000003	10024	4	Sárospatak
1000010	1000003	10025	5	Kossuth út 7.
1253829	1000003	10020	1	0
1037395	1000003	10665	1	987654321
1253909	1000003	10777	1	1099009
1000063	1000003	10023	1	3950
1000062	1000003	10054	1	10402726-27211485-00000000
1000004	1000003	10010	1	10000

Figure 10. New entity.

ID	ATTRIBUTE	T_SEQUENCE	T_VALUE	PRIMARY_ID
10	1	1	1:N (attributum:values)	1253269

1 rows returned in 0.00 seconds [Download](#)

Figure 11. New attribute.

ID	ATTRIBUTE	T_SEQUENCE	T_VALUE	PRIMARY_ID
10000	10	1	10019	10021
10000	10	2	10020	10022
10000	10	3	10023	10033

Figure 12. New virtual data table.

only one index is required to index each attribute, the statement of the thesis is true in every case of the JT logic-based data table according to the principle of mathematical induction below. Thesis: For $n=1$ ergo;

$$1 + 2 + \dots + n = n * (n + 1) / 2$$

substituting one into the equation we get :

$$1 = 1 * (1 + 1) / 2$$

result of the operation is $1=1$, that is, the induction base is true.

Using proof by induction we can now show that this is true for the following equation:

$n = k$ where k is a optional but fixed natural number. Therefore, we know that the following operation is true:

$$1 + 2 + \dots + k = k * (k + 1) / 2$$

Finally using $n=k+1$ we can prove our assumption to be true:

$$1 + 2 + \dots + k + (k + 1) = (k + 1) * (k + 2) / 2$$

The above induction proof shows:

$$1 + 2 + \dots + k + (k + 1) = k * (k + 1) / 2 + (k + 1)$$

Conducting the mathematical operations we obtain the following:

$$1 + 2 + \dots + k + (k + 1) = (k * ((k + 1) / 2) + 2 * (k + 1)) / 2 = (k * k + k + 2k + 2) / 2 = (k * k + 3k + 3) / 2$$

Conducting the mathematical operations on the other side we obtain the same:

$$(k+1) * (k+2) / 2 = (k * k + 2k + k + 2) / 2 = (k + k + 3k + 2) / 2$$

Thus, the induction step is true. Given that both the induction base and the induction step are true, the original statement is therefore true. In the present study, we explained the JT data storage logic. In our other study we focused on the query tests. Our previous results⁸ show that from 18000 records the relational model generates slow (more than 1 second) queries in Oracle Apex cloud-based environment while JT logic based databases can remain with the one second time frame.

Discussion and conclusions

Using the developed database management logic, each attribute needs only one index for indexing in the database. JT allows any data whether entity, attribute, data connection or formula, to be stored and managed even under one physical data table. In the JT logic based databases, the entity and the attribute are used interchangeably, so users can expand the database with new attributes

after or during the development process. Our solution provides three time saving benefits. Firstly it improves the query time over large amounts of data. Secondly it reduces development time for integrating data from various sources. Finally it removes the need for adding or modifying data tables with the addition of new attributes.

With JT logic, one physical data storage is ensured in SQL database systems for the storage and management of long term scientific information (Dataset 1).

Dataset 1. Data storage structure in JT logic based databases

<https://dx.doi.org/10.6084/m9.figshare.3119086.v1>

For the unique identification more attributes (columns) are applied in rational databases.

In practice, attributes within a type are defined in same data table.

Introducing new entity attributes requires the modification of the existing data tables and applications that use these data tables.

In JT logic based databases, records with the same ID values identify one entity called virtual record.

Virtual records with the same value of the "belonging to the virtual data table" attribute form a data table called virtual data table.

Data availability

Figshare: Data storage structure in JT logic based databases. doi: [10.6084/m9.figshare.3119086.v1](https://dx.doi.org/10.6084/m9.figshare.3119086.v1)¹⁰

Author contributions

BM, MSZ, GJ, IA, GB, GyM conceived the study. MSZ, GJ, IA, GK, AF, GyM tested the developed method. GK developed the mathematical proof related to indexing. GK and AF made the mathematical description of JT database model. BM prepared the first draft of the manuscript. All authors were involved in the revision of the draft manuscript and have agreed to the final content.

Competing interests

No competing interests were disclosed.

Grant information

The first version of JT is a Hungarian product which was developed in 2008 (R.number: INNO-1-2008-0015 MFB-00897/2008) thanks to an INNOCSEK European Union application.

I confirm that the funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

The corresponding author is thankful to György Mátyás, the idea owner of JT framework. The authors are thankful to the Call Tec Consulting Ltd. (the first one with highest Oracle certification in Hungary) and the first one that validated JT.

References

1. Goldacre M, Kurina L, Yeates D, *et al.*: **Use of large medical databases to study associations between diseases.** *QJM*. 2000; **93**(10): 669–675.
[PubMed Abstract](#) | [Publisher Full Text](#)
2. Kumar R, Sharma Y, Pattnaik PK: **Privacy preservation in vertical partitioned medical database in the cloud environments**. *IEEE, Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, International Conference on, Noida, 2015; 236–241.
[Publisher Full Text](#)
3. Simon GE, Unützer J, Young BE, *et al.*: **Large medical databases, population-based research, and patient confidentiality.** *Am J Psychiatry*. 2000; **157**(11): 1731–1737.
[PubMed Abstract](#) | [Publisher Full Text](#)
4. Delgado M, Sánchez D, Martín-Bautista MJ, *et al.*: **Mining association rules with improved semantics in medical databases.** *Artif Intell Med*. 2001; **21**(1–3): 241–245.
[PubMed Abstract](#) | [Publisher Full Text](#)
5. Leavitt N: **Will NoSQL databases live up to their promise?** *Computer*. 2010; **43**(2): 12–14.
[Publisher Full Text](#)
6. Pereira D, Oliveira P, Rodrigues F: **Data warehouses in MongoDB vs SQL Server: A comparative analysis of the query performance.** *Information Systems and Technologies (CISTI)*. 10th Iberian Conference on, 2015; 1–7.
[Publisher Full Text](#)
7. Parker Z, Poe S, Vrbsky SV: **Comparing NoSQL MongoDB to an SQL DB.** *ACMSE '13, Proceedings of the 51st ACM Southeast Conference Article No. 5*, New York, NY, USA, 2013.
[Publisher Full Text](#)
8. Mátyás B, Mátyás G, Horváth J, *et al.*: **Data storage and management related to soil carbon cycle by a NoSQL engine on a SQL platform - Joker Tao.** *J Agr Inform*. 2015; **6**(3): 67–74.
[Publisher Full Text](#)
9. Paredaens J, Bra PL, Gyssens M, *et al.*: **The structure of the relational database model.** Springer-Verlag. 1989; **17**: x 233.
[Publisher Full Text](#)
10. Mátyás B, Szarka M, Járvas G, *et al.*: **Dataset 1: Data storage structure in JT logic based databases.** *Figshare*. 2016.
[Data Source](#)

Open Peer Review

Current Peer Review Status:

Version 3

Reviewer Report 31 August 2017

<https://doi.org/10.5256/f1000research.13330.r25012>

© 2017 Alam M. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Mansaf Alam

Department of Computer Science, Jamia Millia Islamia, New Delhi, India

I have gone through entire manuscript; I feel it is a good work. The manuscript is written very well. This manuscript will be helpful to the researcher who is working in the field of cloud computing. The result presented in this work is good. The methodology is also explained in good manner. The materials supplied in this work are sufficient. But I feel lack of reference of 2016 and 2017. The author should add some more reference of year 2016 and 2017. Over all paper is good.

References

1. Warth B, Levin N, Rinehart D, Teijaro J, et al.: Metabolizing Data in the Cloud. *Trends Biotechnol.* 2017; **35** (6): 481-483 [PubMed Abstract](#) | [Publisher Full Text](#)
2. Khan S, Liu X, Shakil K, Alam M: A survey on scholarly data: From big data perspective. *Information Processing & Management.* 2017; **53** (4): 923-944 [Publisher Full Text](#)
3. Alam M, Shakil KA: Cloud Database Management System Architecture. *UACEE International Journal of Computer Science and its Applications.* 2013; **3** (1): 27-31
4. Juels A, Oprea AM, van Dijk ME, Stefanov EP: Remote verification of file protections for cloud data storage. *US 9230114 B1.* 2016.

Is the rationale for developing the new method (or application) clearly explained?

Yes

Is the description of the method technically sound?

Yes

Are sufficient details provided to allow replication of the method development and its use by others?

Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 2

Reviewer Report 06 April 2016

<https://doi.org/10.5256/f1000research.8995.r13081>

© 2016 Lindström J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Jan Lindström

MariaDB Corporation, Espoo, Finland

In my first review I requested full and significant rewrite of the paper. This has not happened. Authors did add some new material and video, both useless for validating the correctness and usefulness of the proposed method.

Firstly, research question is missing, what are the problems the proposed method tries to solve?

Secondly, the transformation of the traditional relational model to proposed model is not described clearly enough. It seems that there are some rules how relations and their attributes are stored to new structure, but this is not described clearly enough. Paper should list clearly set of rules that are used and give examples how these rules are applied.

Third, usefulness of the proposed method is not clear. Sure you can have only one index, but how you do simple queries like select first_name,street from Personal p, Address a where a.id = p.id is executed? How user could know what ID some attribute now has? How the created one index can be used to perform simple primary key or foreign key queries. How constraints are enforced ?

Finally, what are the use cases for JT logic and how the proposed method improves the state-of-the-art i.e. compared to relational model or object oriented model? This question remain fully open based on this paper.

This paper does not successfully fulfill requirements of the scientific paper. At its current form, this looks more like a marketing material.

Is the rationale for developing the new method (or application) clearly explained?

Yes

Is the description of the method technically sound?

Yes

Are sufficient details provided to allow replication of the method development and its use by others?

Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

Version 1

Reviewer Report 25 February 2016

<https://doi.org/10.5256/f1000research.8321.r12373>

© 2016 Oza K. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Kavita Sunil Oza

Department of Computer Science, Shivaji University, Kolhapur, Maharashtra, India

Work demonstrated in the paper is good and well explained. Complexity of work is not mentioned (algorithmic complexity) but this is not necessary as we already have high speed processors and time complexity may not matter much. Some more references should have been added but not mandatory as number of references are sufficient.

Is the rationale for developing the new method (or application) clearly explained?

Yes

Is the description of the method technically sound?

Yes

Are sufficient details provided to allow replication of the method development and its use by others?

Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 15 February 2016

<https://doi.org/10.5256/f1000research.8321.r12375>

© 2016 Lindström J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Jan Lindström

MariaDB Corporation, Espoo, Finland

In this paper authors introduce a new logic called Joker Tao (JT) which provides universal data storage for cloud-based databases. However, the paper is very poorly written. Firstly, the proposed logic is not presented detailed enough for the reader to understand and validate the method. Authors should research how relational model is presented and based on rigorous relational calculus and algebra. Based on this research, this paper should be rewritten based on rigorous mathematical foundation and give clear examples. Secondly, one table based example is far from convincing and provided Java-program is unnecessary. Length of the paper should be greatly increased to contain detailed description of JT method and give examples. Lastly, presentation is so poor that is not even clear how queries to resulting JT structure can be executed. To be honest, currently paper looks more like computer generated rubbish than a real scientific paper.

Is the rationale for developing the new method (or application) clearly explained?

Yes

Is the description of the method technically sound?

Yes

Are sufficient details provided to allow replication of the method development and its use by others?

Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research