A NoSQL Database about Customer Reviews

Paolino Di Felice
Dipartimento di Ingegneria Industriale e dell'Informazione, Economia
Università di L'Aquila (Italy)

Martina Marinelli, Gaetanino Paolone, Francesco Pilotti
Gruppo SI S.c.a.r.l., 64100 Teramo (Italy)

Giovanni Valenza
Comit S.r.l., 00193 Roma (Italy)

ABSTRACT

Small and medium-sized enterprises (SMEs) are the backbone of the economy of most countries. There is large evidence in the literature that digitalization improves the market performance of enterprises and, as a consequence, it helps the growth of their businesses. SMEs suffer of budget restrictions, as a consequence within them is low R&D expenditure as well as the number of employees with IT technical abilities. This lack is an objective obstacle for SMEs to compete in the online market. The aim of the present paper is to reach out a hand to SMEs by presenting in deep details how to structure, load and querying a NoSQL PostgreSQL database being part of the customer service.

General Terms

Data and Information Systems, Digital Systems

Keywords

Customer service, Complaint, Database, NoSQL, JSON, SME

1. INTRODUCTION

There is large evidence in the literature that digitalization improves the market performance of enterprises and, as a consequence, it helps the growth of their businesses. This claim is true in special way for Small and Medium-sized Enterprises (SMEs). Compared with other enterprises, SMEs are confronted with a unique set of issues when competing in the globalised market where operate, in a manner almost hegemonic, giants such as Amazon.

The term "IT readiness" has been introduced in the literature as a precondition that SMEs need to meet in order to fully exploit IT potentials (e.g., [1]). [2] defines SME readiness in terms of enterprise's financial resource to take risks and pressures to change processes, management's IT skills and project support, and employee's IT skills and attitude; while [3] suggests readiness in terms of internal IT infrastructure and number of IT employees. So, different scholars understand the term SME readiness differently; nonetheless, the definitions share the fact that to exploit IT potentials, SMEs must embed "IT-capabilities" in themselves. Unfortunately, for most SMEs this is not true, as stressed, for example, in a 2020 document of the European Commission: "SMEs often must [...] overcome structural barriers such as a lack of management and technical skills [...]" [4]. In light of the findings of [5], a low level of IT readiness within SMEs is a barrier for them to change and grow in the digital-based global market.

The present paper aims at giving an help to SMEs to cover the IT gap. In fact, the paper describes in full details how it is possible to build a database about complaining customers. All firms playing in the market know very well that no matter how hard they try, they cannot please 100% of the people 100% of the time, that's why customer complaints are inevitable. Whatever the problem might be, the way they address unhappy customers and handle their complaints can have a major impact on the company's reputation. Obviously, ignoring complaints and failing to resolve them can make the customers leave and spread negative word of mouth. On the opposite, Yilmaz et al. [6] have shown that learning from complaints influence both short- and long-term firm-level performance measures positively. In light of considerations above, SMEs that aim entering the online market of goods and/or services have to adopt a database as the one described in the present contribution.

The paper is structured as follows. Section 2 is about the related work, while Section 3 presents the structure of the database and the SQL scripts for its creation and loading. A meaningful set of query patterns are also given; SMEs can use them to carry out relevant statistics. As stated above, the contribution of this work is the database structure and the query patterns against it; that is the reason why the tables were loaded with a small number of sample data and the query results are not discussed. Section 4 concludes the paper and describes the work in progress at present, as well as the aim of the long term industrial project that has been launched.

2. RELATED WORK

No company is so perfect in the delivery of their products/services that dissatisfaction (the source of complaints) does not exist. In this context the well-known saying: "No news, good news" is not always true. So, the correct approach from SMEs that aim playing online lengthily is handling complaints. Handling complaints includes the following three steps: (a) collecting them; (b) analyzing them, and (c) responding appropriately (i.e., overcoming the underlying issue). This paper focuses on the first two steps. Only when a complaint has been captured, the appropriate corrective action can be taken. Nobody complains without reason. No matter how absurd a complaint might sound, it is important to look at the posed issue from the customer's point of view. What might be an effective procedure for resolving customer complaints is a topic outside the scope of the present paper. [7], for example, proposes a seven-step customer complaint procedure. [8] is another interesting, and recent, source for managers to learn about how to deal with customers online complaints. In fact, the study collects best practices to online complaint management from a team of researchers with backgrounds in retailing and relationship marketing.

In [7], authors report that "for every complaint expressed, there are over 25 unregistered complaints. Many dissatisfied customers just quietly take their business elsewhere. […] Furthermore, a customer with a complaint is likely to tell others about his complaint." Organizations that are truly committed to delivering an effective customer service have to providing them opportunities to complain. A mobile app or a webpage are the easiest way to encourage customers to write their comments that, from those collecting points, can then automatically be redirected into a firm's database.

3. THE NoSQL DATABASE

The implementation of an effective customer service implies that different channels are adopted in a consistent and coordinated way [9]. A database collecting users complaints is one of those medium. Such a database is a precious intangible assets for firms, since, by querying it, it is possible to extract customers complaints and take appropriate actions to reply to them. In addition, the availability of such kind of data allows the implementation of quantitative methods as an alternative to qualitative ones. In fact, by querying the database over long periods of time it is possible to build statistics useful to get a correct vision of what is going wrong with the offered products and/or services.

PostgreSQL (ver.13) has been adopted as DBMS. The reasons for such a choice are the following. PostgreSQL is an open-source system; using a free software is a unique opportunity for SMEs to keep the costs low. Moreover, PostgreSQL offers a robust support to the `JSON` data type which, in turn, allows to implement NoSQL databases. The features of NoSQL databases have been reported in the literature (e.g., [10, 11]). To implement an effective customer service it is highly recommendable that the underlying database is able to host unstructured data.

The customer survey the paper refers to is composed of ten questions. It was taken from page: https://www.surveymonkey.com/mp/customer-satisfaction-survey-template/.

3.1 The Tables

The NoSQL database is composed of five tables: Our_Products, The_Survey_Template, Our_Customers, Sales, The_Reviews. The PostgreSQL's SQL/DDL scripts are listed below.

```
CREATE TABLE Our_Products (
P_Code serial PRIMARY KEY NOT NULL,
description JSONB);

CREATE TABLE Our_Customers (C_Code serial PRIMARY KEY NOT
NULL, Description JSONB);

CREATE TABLE Sales (
P_Code integer NOT NULL,
C_Code integer NOT NULL,
"When" date NOT NULL,
PRIMARY KEY (C_Code, P_Code, "When"),
FOREIGN KEY (P_Code) references Our_Products(P_Code) ON
UPDATE CASCADE,
FOREIGN KEY (C_Code) references Our_Customers(C_Code) ON
UPDATE CASCADE);

CREATE TABLE The_Survey_Template (
TheQuestions JSONB NOT NULL);

CREATE TABLE The_Reviews (
R_Code serial PRIMARY KEY NOT NULL,
C_Code integer NOT NULL,
P_Code integer NOT NULL,
"When" Date NOT NULL,
TheQuestions JSONB NOT NULL,
FOREIGN KEY (C_Code) references Our_Customers(C_Code) ON
UPDATE CASCADE,
FOREIGN KEY (P_Code) references Our_Products(P_Code) ON
UPDATE CASCADE);
```

3.2 A Sample Database

The database is structured as 30 products, 20 customers, and 100 sales. Five reviews are loaded into The_Reviews table; while the ten questions of the survey are stored into The_Survey_Template table. The PostgreSQL's SQL/DML scripts are listed below. Due to space limits only the INSERT of the first review is shown. Table 1 collects the records about the five reviews. Overall, four different customers wrote, on 2020, five reviews mentioning two different products/services.

Table 1. The tuples in The_Reviews table

R_Code	C_Code	P_Code	When
1	1	3	'2020-03-13'
2	1	10	'2020-05-18'
3	5	3	'2020-04-19'
4	10	3	'2020-08-04'
5	14	10	'2020-09-28'

Figure 1 shows the screen dump of the created JSONB database and the five tables.

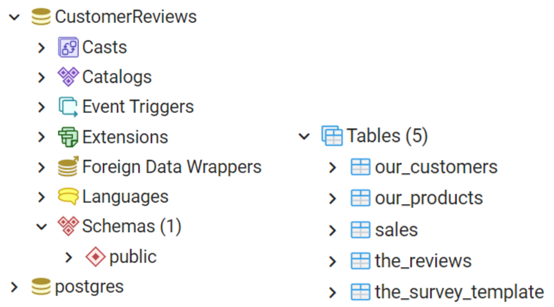


Fig 1: Two screens about the created NoSQL database

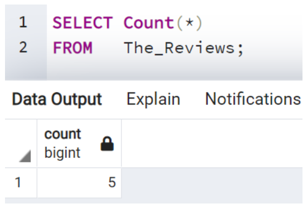
```
-- Insertion of 30 products into Our_Products
INSERT INTO Our_Products(P_Code)
  SELECT n
  FROM generate_series(1,30) n;

-- Insertion of 20 customers into Our_Customers
INSERT INTO Our_Customers(C_Code)
  SELECT n
  FROM generate_series(1,20) n;

-- Loading of 100 purchasing inside table Sales
INSERT INTO Sales (P_Code, C_Code, "When")
    SELECT
        -- Generation of a random natural in the range [1..30]
        floor(random() * 30 +1)::int,
        floor(random() * 20 +1)::int,
        (now() - (random() * (NOW()+'365 days' - NOW())))::date
        -- Generation of a random date in the last year from
        now()
    FROM generate_series(1,100) n;


INSERT INTO The_Survey_Template (TheQuestions)
VALUES (
'{"Q1":
    {"description": "Recommend it to a friend. From Not AT ALL (0)
    to EXTREMELY LIKELY (10)",
    "score": 6},
 "Q2":
    {"description": "Overall, how satisfied or dissatisfied are you
    with our company?",
    "field1": "Very satisfied",
    "field2": "Somewhat satisfied",
    "field3": "Neither satisfied nor dissatisfied",
    "field4": "Somewhat dissatisfied",
    "field5": "Very dissatisfied"},
  "Q3":
    {"description": "Which of the following words would you use to
    describe our products/services? Select all that apply.",
    "field1": "Reliable",
    "field2": "High quality",
    "field3": "Useful",
    "field4": "Unique",
    "field5": "Good value for money",
    "field6": "Overpriced",
    "field7": "Impractical",
    "field8": "Ineffective",
    "field9": "Poor quality",
    "field10":"Unreliable"},
  "Q4":
    {"description": "How well do our product/service meet your
    needs?",
    "field1": "Extremely well",
    "field2": "Very well",
    "field3": "Somewhat well",
    "field4": "Not so well",
    "field5": "Not at all well"},
  "Q5":
    {"description": "How would you rate the quality of the
    product/service?",
    "field1": "Very high quality",
    "field2": "High quality",
    "field3": "Neither high nor low quality",
    "field4": "Low quality",
    "field5": "Very low quality"},
  "Q6":
    {"description": "How would you rate the value for money of the
    product/service?",
    "field1": "Excellent",
    "field2": "Above average",
    "field3": "Average",
    "field4": "Below average",
    "field5": "Poor"},
  "Q7":
    {"description": "How responsive have we been to your
    questions or concerns about our product/service?",
    "field1": "Extremely responsive",
    "field2": "Very responsive",
    "field3": "Somewhat responsive",
    "field4": "Not so responsive",
    "field5": "Not at all responsive",
    "field6": "Not applicable"},
  "Q8":
    {"description": "How long have you been a customer of our
    company?",
    "field1": "This is my first purchase",
    "field2": "Less than six months",
    "field3": "Six months to a year",
    "field4": "1 - 2 years",
    "field5": "3 or more years",
    "field6": "I have not made a purchase yet"},
  "Q9":
    {"description": "How likely are you to purchase any of our
    products/services again?",
    "field1": "Extremely likely",
    "field2": "Very likely",
    "field3": "Somewhat likely",
    "field4": "Not so likely",
    "field5": "Not at all likely"},
  "Q10":
    "Do you have any other comments, questions, or concerns?"
} ');


-- Review 1
INSERT INTO The_Reviews (C_Code, P_Code, "When",
TheQuestions)
VALUES
(1, 3, '2020-03-13',
'{"Q1": {"score": 6},
  "Q2": {"field4": "Somewhat dissatisfied"},
  "Q3": {"field6": "Overpriced"},
  "Q4": {"field5": "Not at all well"},
  "Q5": {"field4": "Low quality"},
  "Q6": {"field5": "Poor"},
  "Q7": {"field3": "Somewhat responsive"},
  "Q8": {"field3": "Six months to a year"},
  "Q9": {"field4": "Not so likely"}
} ');
```

3.3 Query Patterns

This subsection collects ten query patterns useful to build statistics on actual data that likely covers a large time interval. For each query the PostgreSQL's screenshot showing the result is also given.

Query 1
Count the number of reviews in the DB (Figure 2).

```
SELECT Count(*)
FROM   The_Reviews;
```



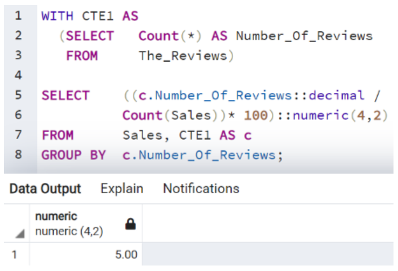
```
SELECT Count(*)
FROM   The_Reviews;
```

```
  count
  bigint
1      5
```

Fig 2: Query 1 and its output

Query 2
Compute the percentage of reviews with respect to the total number of sales (Figure 3).

```
WITH CTE1 AS
      (SELECT Count(*) AS Number_Of_Reviews
       FROM   The_Reviews)

SELECT ((c.Number_Of_Reviews::decimal /
            Count(Sales)) * 100)::numeric(4,2)
FROM     Sales, CTE1 AS c
GROUP BY c.Number_Of_Reviews;
```



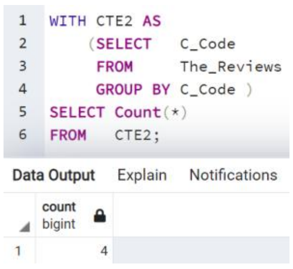
```
WITH CTE1 AS
   (SELECT   Count(*) AS Number_Of_Reviews
    FROM     The_Reviews)

SELECT    ((c.Number_Of_Reviews::decimal /
           Count(Sales))* 100)::numeric(4,2)
FROM       Sales, CTE1 AS c
GROUP BY   c.Number_Of_Reviews;
```

```
  numeric
  numeric (4,2)
1          5.00
```

Fig 3: Query 2 and its output

Query 3
Compute the number of distinct complaining customers (Figure 4).

```
WITH CTE2 AS
      (SELECT C_Code
      FROM   The_Reviews
      GROUP BY C_Code )

SELECT Count(*)
FROM CTE2;
```



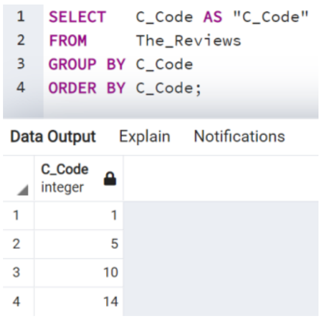
```
WITH CTE2 AS
    (SELECT   C_Code
     FROM     The_Reviews
     GROUP BY C_Code )
SELECT Count(*)
FROM   CTE2;
```

	count bigint
1	4

Fig 4: Query 3 and its output

Query 4
Show the C_Code of the complaining customers (Figure 5).

```
SELECT C_Code  AS "C_Code"
FROM   The_Reviews
GROUP BY C_Code
ORDER BY C_Code;
```



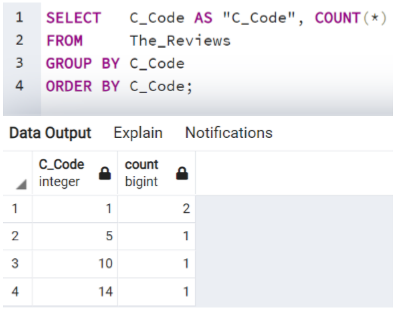
```
SELECT   C_Code AS "C_Code"
FROM     The_Reviews
GROUP BY C_Code
ORDER BY C_Code;
```

	C_Code integer
1	1
2	5
3	10
4	14

Fig 5: Query 4 and its output

Query 5
Show the C_Code of each complaining customer and the number of reviews he/she wrote (Figure 6).

```
SELECT C_Code AS "C_Code", COUNT(*)
FROM   The_Reviews
GROUP BY C_Code
ORDER BY C_Code;
```



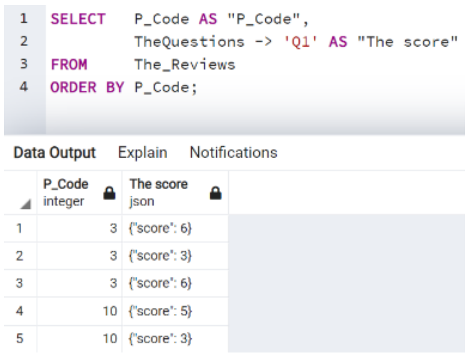
```
SELECT   C_Code AS "C_Code", COUNT(*)
FROM     The_Reviews
GROUP BY C_Code
ORDER BY C_Code;
```

	C_Code integer	count bigint
1	1	2
2	5	1
3	10	1
4	14	1

Fig 6: Query 5 and its output

Query 6
Show the overall score of the products mentioned in the reviews (Figure 7).

```
SELECT P_Code AS "P_Code", TheQuestions -> 'Q1' AS "The score"
FROM   The_Reviews
ORDER BY P_Code;
```



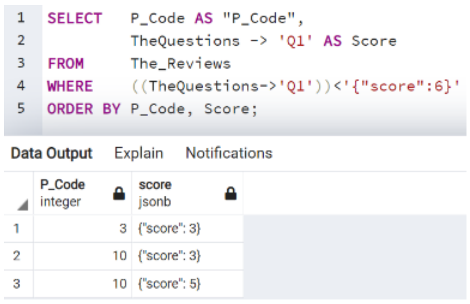
```
SELECT   P_Code AS "P_Code",
         TheQuestions -> 'Q1' AS "The score"
FROM     The_Reviews
ORDER BY P_Code;
```

	P_Code integer	The score json
1	3	{"score": 6}
2	3	{"score": 3}
3	3	{"score": 6}
4	10	{"score": 5}
5	10	{"score": 3}

Fig 7: Query 6 and its output

Query 7
Show the P_Code of the products with overall score below 6 (Figure 8).

```
SELECT  P_Code AS "P_Code", TheQuestions -> 'Q1' AS Score
FROM    The_Reviews
WHERE  ((TheQuestions -> 'Q1' )) < '{"score": 6}'
ORDER BY P_Code, Score;
```



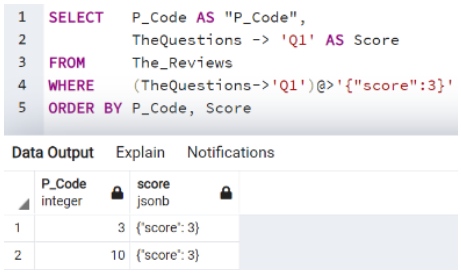
```
SELECT   P_Code AS "P_Code",
         TheQuestions -> 'Q1' AS Score
FROM     The_Reviews
WHERE    ((TheQuestions->'Q1'))<'{"score":6}'
ORDER BY P_Code, Score;
```

	P_Code integer	score jsonb
1	3	{"score": 3}
2	10	{"score": 3}
3	10	{"score": 5}

Fig 8: Query 7 and its output

Query 8
Show the P_Code of products with overall score equal to 3 (if any) (Figure 9).

```
SELECT  P_Code AS "P_Code", TheQuestions -> 'Q1' AS Score
FROM    The_Reviews
WHERE  (TheQuestions -> 'Q1' ) @> '{"score": 3}'
ORDER BY P_Code, Score
```



```
SELECT   P_Code AS "P_Code",
         TheQuestions -> 'Q1' AS Score
FROM     The_Reviews
WHERE    (TheQuestions->'Q1')@>'{"score":3}'
ORDER BY P_Code, Score
```

	P_Code integer	score jsonb
1	3	{"score": 3}
2	10	{"score": 3}

Fig 9: Query 8 and its output

Query 9
Show all the fields of all the reviews in the DB mentioning a given product (e.g., P_Code=10) (Figure 10).

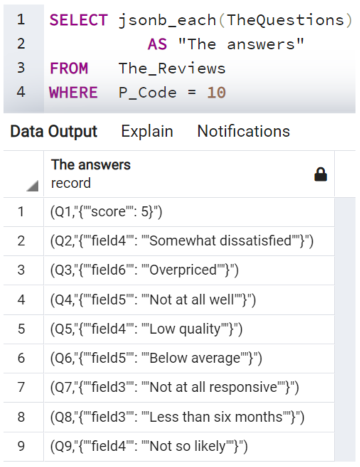
```
SELECT jsonb_each(TheQuestions) AS "The answers"
FROM   The_Reviews
WHERE P_Code = 10
```

The output of this query is composed of as many rows as the number of questions in the survey times the number of tuples which have P_Code=10.

```
SELECT to_jsonb(TheQuestions) AS p
```

```
FROM   The_Reviews
WHERE P_Code = 10
```

The output of this query is composed of as many rows as the number of tuples which have P_Code=10. Each row contains all the fields in the survey.



```
SELECT jsonb_each(TheQuestions)
          AS "The answers"
FROM    The_Reviews
WHERE   P_Code = 10
```

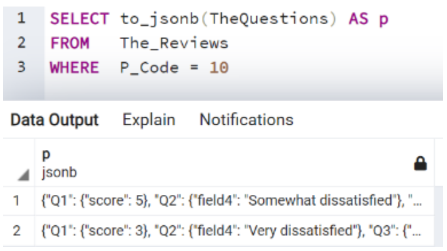
Data Output   Explain   Notifications

```
  The answers
  record
1 (Q1,"{""score"": 5}")
2 (Q2,"{""field4"": ""Somewhat dissatisfied""}")
3 (Q3,"{""field6"": ""Overpriced""}")
4 (Q4,"{""field5"": ""Not at all well""}")
5 (Q5,"{""field4"": ""Low quality""}")
6 (Q6,"{""field5"": ""Below average""}")
7 (Q7,"{""field3"": ""Not at all responsive""}")
8 (Q8,"{""field3"": ""Less than six months""}")
9 (Q9,"{""field4"": ""Not so likely""}")
```

Fig 10: Query 9 and its output

Query 10
Show the P_Code mentioned in the highest number of complaints (Figure 11).

```
WITH CTE3 AS
      (SELECT P_Code, COUNT(*) AS c
      FROM    The_Reviews
      GROUP BY P_Code)

SELECT P_Code AS "P_Code", c AS "Number of Negative Scores"
FROM   CTE3
WHERE c >= ALL
   (SELECT c
   FROM    CTE3)
```



```
SELECT to_jsonb(TheQuestions) AS p
FROM   The_Reviews
WHERE  P_Code = 10
```

Data Output   Explain   Notifications

```
  p
  jsonb
1 {"Q1": {"score": 5}, "Q2": {"field4": "Somewhat dissatisfied"}, "...
2 {"Q1": {"score": 3}, "Q2": {"field4": "Very dissatisfied"}, "Q3": {"...
```

Fig 11: Query 10 and its output

4. CONCLUSIONS AND FUTURE WORK

The paper presented a NoSQL database thought to be part of an advanced customer service of a *network* of collaborating SMEs physically distributed over a territory (for instance, a region, a province or a state), which share the objective of selling goods or services to potential consumers through a *digital* platform (therefore called *Digital Network* - DN). The next step of the work will concern the implementation of a Twitter monitoring component that scans the network for brand mentions, captures those tweets and, then, copies them into a dedicated NoSQL database.

The ongoing industrial research project aims at developing a generator of DNs. The proponents of the project, recently have completed the development and release of a tool (`xGenerator` [12]) that performs the transformations across the levels of the Model Driven Architecture up to the Java code of business Web applications. Both projects implement the emerging *low-code* paradigm. In the case of the generator of DNs, by making recourse to the generator, interested SMEs will be able to instantiate by themselves the DN that best respond to the needs of their businesses.

5. REFERENCES

[1] Dyerson, R. and Spinelli, R. 2011. Balancing Growth: A Conceptual Framework for Evaluating ICT Readiness in SMEs. International Journal of Online Marketing 1 (Apr. 2011), 43-56.

[2] Haug, A., Pedersen, S.G., and Arlbjørn, J.S. 2011. IT readiness in small and medium-sized enterprises. Journal of Industrial Management & Data Systems 111 (Apr. 2011), 490-508.

[3] Hajli, N., Sims, J., and Shanmugam, M. 2014. A practical model for ecommerce adoption in Iran. Journal of Enterprise Information Management 27 (Oct. 2014), 719-730.

[4] User Guide to the SME Definition. Publications Office of the European Union, Luxembourg 2020.

[5] Gerber, A., le Roux, P., and van der Merwe, A. 2020. Enterprise Architecture as Explanatory Information Systems Theory for Understanding Small- and Medium-Sized Enterprise Growth. Sustainability 12 (Oct. 2020), 8517.

[6] Yilmaz, C., Varnali, K., and Kasnakoglu, B.T. 2016. How do firms benefit from customer complaints? Journal of Business Research 69 (Feb. 2016), 944-955.

[7] Farnsworth, D., Clark, J.L., Wysocki, A., Kepner, K., and Glasser, M.W. 2019. Customer Complaints and Types of Customer. Report HR005, University of Florida.

[8] Stevens, J.L., Spaid, B.I., Breazeale, M., and Jones, C.L.E. 2018. Timeliness, transparency, and trust: A framework for managing online customer complaints. Business Horizons 61 (May-Jun. 2018), 375-384.

[9] Stone, M., Hobbs, M., and Khaleeli, M. 2002. Multichannel customer management: The benefits and challenges. Journal of Database Marketing 10 (Sept. 2002), 39-52.

[10] Kumbhar, H., Kinny, E., Fernandes, K., and Maitra, S. 2019. Benefits of NoSQL Databases. In IJCA Proceedings on Leveraging Information Technology for Inter-Sectoral Research ICAIM 2017, no. 1.

[11] Petković, D. 2020. Implementation of JSON Update Framework in RDBMSs. International Journal of Computer Applications 177 (Feb. 2020), 35-39.

[12] Paolone, G., Marinelli, M., Paesani, R., and Di Felice, P. 2020. Automatic Code Generation of MVC Web Applications. Computers 9, 56 (Jul. 2020).