XNorthwind: Grammar-driven Synthesis of Large Datasets for DB Applications

Abejide Ade-Ibijola Member, IAENG, and George Obaido Member, IAENG

Abstract—Relational databases often come with sample databases. One known example is the Northwind database, often used as a repository for software testing and development purposes. The Northwind database includes hypothetical records of customers, companies, products, employee and so on. The number of records in the Northwind is however considered inadequate for large applications, where a developer or user may need a lot more, possibly, millions of records. In this paper, we have used a Context-free Grammar in describing the rules for the synthesis of exponentially many hypothetical datasets that are similar to the Northwind database. We referred to the resulting database as XNorthwind (Extended Northwind). The new grammar was implemented, resulting in thousands of unique data values across the eight different Northwind Data Tables. These datasets will find applications in training and development environments. A survey of 112 participants’ perceptions showed that 94.6% agreed that the XNorthwind can be useful.

Index Terms—Northwind, Sample database, Training dataset, Synthesis of things, Formal grammars applications.

I. INTRODUCTION

With the advent of the Internet and other related technologies, various applications have emerged which have led to a high demand for data, stored in various database (DB) technologies [1]. In testing software applications before release, the higher the volume of data, the better the result derived from the system testing [2, 3, 4]. A number of applications have adopted sample DBs as practice environments for their testing and development tasks [5, 6, 7]. These sample DBs include the Sybase’s Pubs, PostgreSQL’s Sakila and Microsoft’s Northwind [8]. In this work, we are interested in the Microsoft’s Northwind DB, containing the records of a fictitious company known as the “Northwind Traders” [5]. The Northwind DB consist of hypothetical datasets that educates users with useful illustrations of a typical e-commerce scenario and has been extensively used with many software applications and research projects [8, 9]. A 2018 study conducted on querying property graphs used the Northwind DB in a tool called Gremlinator, and the authors reported good results [10]. Other tools that have used the Northwind DB are OntoGrate [11] and SPARK [5].

Despite the capabilities of the Northwind DB to support a diverse set of applications, the datasets is insufficient to meet the current demands of technologies that requires more data for their training needs [12, 13]. A comparative analysis study was conducted on the extraction and generation of Personal Data Reports (PDR) from two relational DBs (i.e. Northwind and TPC-H) [14]. Interestingly, the study showed that although, the Northwind possesses 3.7MB (3.7 · 10^3 tuples) and TPC-H with 1GB (10^9 tuples) of datasets; the TPC-H datasets achieved a better accuracy because of the larger datasets. Roger [12] opined that the Northwind DB contains fewer records than one may find in most production DBs; hence, the current limited record size is not ideal to support a full-fledged system. Taking this limitation into consideration, it has become imperative to create more datasets that can assist instructors and programmers in their training or development tasks.

In this work, we have used a Context-free Grammar (CFG) to describe the syntactic generation of tuples of hypothetical records, similar to the Northwind DB. This appears to be the first time such an approach has been extended to the automatic generation of large datasets to be used as a sample DB. The contributions are stated as follows. We have:

1) designed a CFG for the synthesis of datasets for records of the Northwind DB,
2) implemented the CFG rules and shown that it produced 100,000 tuples (and could produced more) as opposed to 3,200 of the Northwind DB, and
3) evaluated this approach and shown that developers and application users agreed that large datasets can be useful.

The organisation of this paper is as follows. Section II presents the background to this work. Section III describes the grammar for the generation of XNorthwind datasets. Section IV presents the implementation details and result of the XNorthwind DB. Section V presents possible applications of the XNorthwind DB. Section VI presents the evaluation. In Section VII, we present the conclusion and provide future work.

II. BACKGROUND AND RELATED WORK

In this section, we present the problem and a review of the relational DB model, focusing on the Northwind DB and application areas. We also presented definition of used terms.

A. The Problem

One major problem faced with most illustrative DBs such as the Northwind DB is its inability to provide enough datasets that meet the demands for testing critical applications before release [12, 13, 15]. In most cases, the datasets in the Northwind DB is insufficient to support large-scale applications. For enterprise applications, Rogers [12] identified that the Northwind DB does not qualify as a full-fledged sample DB because the datasets is only ideal for small-sized wholesale or
retail outfit mapped with procurement and order fulfillment processes. A similar study by Warren et al. [15] stated that the Northwind DB contains a small number of datasets, ideal to support a beginner learning DBs whilst inadequate to support large scale applications. Recently, an overview of problems faced while learning DBs indicated that large sample DB for training are not readily available, and most textbook examples are too oversimplified, and inadequate to cope with real-world scenario [16]. The author echoed that training students with a large sample DB would prepare them to cope with pressures at the workplace. In view of all these, synthesis of datasets in the Northwind DB is desirable. One benefit of this is that, it will provide software developers with enough datasets to use for deployment and testing applications. This work explores formal techniques using CFGs to solve this problem.

B. Relational Database

The relational DB model is one of the most simple structure for storing and organising data for easy retrieval [3, 17]. Since released in 1970s, it has found applications in large-scale commercial implementations of banking systems, airline reservation systems and in desktop computers for maintaining and storing of records [18]. The relational DB that we are interested in is the Microsoft Northwind DB. Vicknair et al [19] described the Microsoft SQL Server as a relational DB that supports both desktop and web applications. Chung [20] highlighted the benefits of Microsoft SQL Server over other popular DBs such as tremendous ROI1. Rapid Application Development; it is also good for data entry and reporting. The step by step installation guide for the Northwind DB is provided in [21]. The applications of Northwind DB is discussed in the next section.

C. Northwind Database and Applications

The Northwind DB contains eight tables and 3,200 tuples comprising of: Suppliers, Products, Orders, Shippers, Customers, OrderDetails, Categories and Employees [5]. As an illustrative DB, Northwind resemble a typical merchandising firm that undergoes sales transactions that occur between a company and its customers. This DB provides a model for table relationships, forms, queries, VBA2, data access, and manipulation functionalities [22]. Borker [23] regarded the Northwind as an “intuitive” OLTP3 system that stores and links tables by means of a primary key. Nelson [24] illustrates the Northwind DB using a schema showing entities and the relationship among them as seen in Figure 1. In the schema, orders are shipped by a Supplier with details stored in the Shippers table.

The Northwind DB have been extensively used in a number of applications such as:

Decision Support Systems Angermann et al [26] used the Northwind DB to demonstrate the efficiency of Taxo-semantics, a decision support system that was used to match an expression against other sources of knowledge. The study concluded that the Northwind DB improved the accuracy of the system. A recent study in 2018 by Runwuene et al. [7] applied the Northwind DB for a comparative study for the Extract, Transformation and Loading (ETL) data integration processes. The study aimed to assist a BI4 developer in processing data to produce useful information.

Semantic Web Applications A number of semantic web applications have used the datasets from the Northwind DB for their operations. Tools, such as SPARK [5] and OntoGrate [11] used the Northwind DB as a back-end for a keyword search and semantic web ontologies respectively.

Natural Language Systems Lumbantoruan et al [6] applied the Northwind DB in evaluating a star schema5, that automatically generates and identifies noun words. A study conducted by Gelbukh [27] used the Northwind DB in the translation of queries expressed in natural languages; using prepositions and conjunctions into formal languages.

Computer Science Education In an introductory course on IT Audit, Northwind was used as a tutorial DB for beginners [28]. The author stressed that although the Northwind DB was ideal for teaching, its datasets is inadequate for analysis in a vendor neutral environment. Similarly, Lavbić [29] proposed a system that applies hints, meant to assist students to solve SQL-related exercises. The system adopted the Northwind DB as the backend in solving problems in SQL related tasks.

Healthcare Systems Kaddoura et al [30] conducted a study that involved tracking and repairing damaged health care databases, the Northwind DB was used as the experimental db. The study showed that the Northwind DB performed better because of its data consistency. The result of this study were further replicated in similar studies [31, 32].

We have presented the application areas of the Northwind DB. It is important to note that the above-mentioned areas are some of application of this test DB. Other applications that have used the Northwind DB are discussed in [33, 34].

D. Definition of Terms

Noam Chomsky coined the term “Context-free Grammars” or CFGs while describing classes of formal grammars [35]. These grammars differs with their generative and recognitive capacity. Here, we define some terms used in this paper.

Definition 1. (Context-free grammar [36].) A context-free grammar or CFG is a four-tuple, \( G = (N, \Sigma, P, S) \) where:
1) \( N \) is a finite set of non-terminal symbols.
2) \( \Sigma \) represents a finite set of terminals symbols, disjoint from \( N \).
3) \( P \) is a set of productions.
4) \( S \) is the start symbol.

Each non-terminals can be replaced by a string of terminals to the right of the arrow represented as production rules. The rule of the form: \( A \rightarrow \alpha \), simply replaces A with

1Return On Investment
2Visual Basic for Applications
3Online Transaction Processing
4Business Intelligence
5A form of data warehouse modelling
α, where A is the non-terminal or a left-hand side symbol and α are strings of right-hand side symbols or terminals.

E. Related Work

Formal grammars have been used in a wide range of applications. In this section, we present applications of CFGs to research similar to that discussed in this work.

Structural 3D Designs Formal Grammars have been extensively used in the design fields such as product design [37], architecture [38] and 3D modeling [39]. Christensen [40] extended the use of CFGs in a tool named Structure Synth for creating 3D images. The Structure Synth engine uses a recursive descent parser to create and transform rules stored in 4x4 matrices.

Profile Synthesis Ade-Ibijola [41] developed a tool based on a variation of CFGs, that automatically synthesises social media profiles using the Facebook user profile page as a test case. Lin [42] presented a tool aimed at assisting a digital forensic examiner to build behavioural profile from analysis of a network traffic. This tool applied CFGs to compare behavioural patterns and reduce the volume of evidence needed to analyse a network traffic.

Multimedia Applications A study was conducted by Purdaruth et al. [43] using CFGs to automatically generate song lyrics. The lyrics generator applied grammatical rules and statistical constraints derived from a song corpus to generate lyrics. FINCHAN [44] was developed using CFGs for the automatic comprehension and summarisation of financial instant messaging.

Natural Language Processing (NLP) A recent study by Velupillai et al. [45] showed that CFGs was used to identify pathological findings in radiology reports in clinical NLP data. The study showed that integrating CFGs to state-of-the-art NLP tools will advance clinical tools in the near future. Liang [46] built a parser using CFGs for natural language understanding in a question answering system. The study concluded that the CFGs were an essential component used to parse natural languages in this system.

Protein Synthesis One notable application of CFGs in RNA\textsuperscript{6} structure prediction and detection of patterns in DNA\textsuperscript{7} was presented in [47]. Experiments in this study concluded that the CFG approach was helpful in producing human-readable descriptors for the analysis of these protein sequences.

Program Synthesis Butler [48] proposed a system that uses the CFGs with a domain-specific extension to support variable binding and a type system to construct a program. A research study in 2018 by Ade-Ibijola [49] uses CFGs for the automatic generation of procedural programs in Python. The study concluded that the CFG approach used in this research can be applied to generate

\textsuperscript{6}Ribonucleic acid
\textsuperscript{7}Deoxyribonucleic acid
programs in many procedural programming languages.

**Signal Processing** Macko [50] used CFGs to syntactically analyse a VHDL (VHSIC Hardware Description Language) model used for digital signal processing before it is visualised and simulated. In this work, CFG was used to transform the VHDL into an intermediate form that conforms with processing of digital signals. A research study by Fanaswala and Krishnamurthy [51] extended the use of a variation of CFG and the reciprocal Markov model to model long-range signal dependencies. The authors stressed that the CFG possess the added advantage because of its expressive power and ability to deal with variable-range dependencies.

Together, all these areas have applied the use of CFGs for describing the languages used in these domains. Other applications areas of CFGs include: Fuzzy systems [52, 53], Safety systems [54, 55] and Software systems [56, 57]. In the next section, we describe the grammar formalism for synthesizing large datasets for XNorthwind DB.

### III. Grammar Design for the XNorthwind Database

In the previous section, a wide range of applications areas of CFGs was presented. This section describes the use of CFGs for the automatic generation of large datasets in the XNorthwind DB.

\[
\begin{align*}
<\text{comma}> & \rightarrow , \\
<\text{wspace}> & \rightarrow \text{ws} \\
<\text{period}> & \rightarrow . \\
<\text{dash}> & \rightarrow - \\
<d> & \rightarrow 0 | \ldots | 9 \\
<\text{b_slash}> & \rightarrow \backslash \\
<f_slash> & \rightarrow / \\
<\text{colon}> & \rightarrow : \\
<\text{brac_o}> & \rightarrow ( \\
<\text{brac_c}> & \rightarrow ) \\
<\text{sup_id}> & \rightarrow 1 | \ldots | 5 \cdot 10^3 \\
<\text{cat_id}> & \rightarrow 1 | \ldots | 5 \cdot 10^3 \\
<\text{emp_id}> & \rightarrow 1 | \ldots | 2.1 \cdot 10^4 \\
<\text{ship_id}> & \rightarrow 1 | \ldots | 3.2 \cdot 10^5 \\
<\text{cus_id}> & \rightarrow 1 | \ldots | 2.8 \cdot 10^4 \\
<\text{ord_id}> & \rightarrow 1 | \ldots | 2 \cdot 10^4 \\
<\text{prod_id}> & \rightarrow 1 | \ldots | 2 \cdot 10^4 
\end{align*}
\]

To generate the datasets, we describe the set of productions which are rules that make up the grammar. These rules replace the nonterminal symbols that appear on the left-hand side with terminal or nonterminals symbols on the right-hand side of the productions. In Production 1-10, we present the symbols that appear in some of the rules used for other productions. The initial productions show comma, white space (wspace), period (period), hyphen (dash), digits (d), backslash (b_slash), forward slash (f_slash), colon (colon), bracket open (brac_o) and bracket close (brac_c). Productions 11-17 is used to present the ids (primary) keys for each of the eight tables, and in some cases, they appear as foreign keys in some tables. For example, the <ord_id> appears as a primary key in the Orders table. Similarly, it is a foreign key in the Orderdetails table. Productions 11 and 12 allow for random supplier and category ids ∈ [5000]. Production 13 allows for employee ids ∈ [21000]. Production 14 allows for shippers ids ∈ [1000]. Production 15 allows for customer id ∈ [28000]. Productions 16 and 17 allow for order and product ids ∈ [20000]. This amounts to 100,000 as opposed to 3,200 tuples contained in the Northwind DB.

In Productions 18-21, the orders and quantity of the tables are generated within the range presented. The quantity field is found in the Orderdetails and Products tables.

\[
\begin{align*}
<\text{unit_order}> & \rightarrow 0 | \ldots | 120 \\
<\text{units_stk}> & \rightarrow 0 | \ldots | 100 \\
<\text{reorder_l}> & \rightarrow 0 | \ldots | 30 \\
<\text{quantity}> & \rightarrow 1 | \ldots | 50 
\end{align*}
\]

Productions for names specified in the tables are presented within the range of 22 and 32. The <fname> symbol is specified as the first name of the field in the table where \( n_1 \) is the total number of first names that appear. The <lname> symbol specifies the last name of the field and \( n_2 \) is the number of last names in the field. The <cat_name> symbol represents the category names and \( n_3 \) is the total number of all category names contained in the field. In the <comp_suffix> symbol, this shows the company suffixes that may appear (e.g. Limited, Services, Agency, Consulting, Advisors, etc.) \( n_4 \) is the number of such suffixes. In Production 26, a company name is generated with first name and arbitrary company suffixes (e.g. Booyesen Consulting). Production 27 shows how a contact is generated while \( n_5 \) shows the number of generated contacts. In Production 28, arbitrary ship names are generated and \( n_6 \) shows the total number of generated ship names. The ship via symbol, <ship_via>, is generated in Production 29. \( n_7 \) is the total number of such pattern. Production 30 describes the rules for product names and \( n_8 \) shows the total number of product names that appears. The <report_to> symbol is a concatenation of the first and last names as seen in Production 31. The <cont_name> holds if the first name and last name applies, indicated in Production 32.

\[
\begin{align*}
<\text{fname}> & \rightarrow fn_1 | \ldots | fn_{n_1} \\
<\text{lname}> & \rightarrow ln_1 | \ldots | ln_{n_2} \\
<\text{cat_name}> & \rightarrow cat_1 | \ldots | cat_{n_3} \\
<\text{comp_suffix}> & \rightarrow cf_1 | \ldots | cf_{n_4} \\
<\text{comp_name}> & \rightarrow <\text{fname}> <\text{wspace}> \\
<\text{contact}> & \rightarrow con_1 | \ldots | con_{n_5} \\
<\text{ship_name}> & \rightarrow shp_1 | \ldots | shp_{n_6} \\
<\text{ship_via}> & \rightarrow shv_1 | \ldots | shv_{n_7} \\
<\text{report_to}> & \rightarrow <\text{fname}> <\text{fname}> <\text{wspace}> \\
\end{align*}
\]

(Advance online publication: 20 November 2019)
Productions 33-36 is used for titles. The contact title symbol, \(<cont\_title>\), is generated in Production 33 (e.g. Purchasing Manager, Sales Manager, Owner, etc.). \(n_9\) is the total number of contact titles that may appear. The title symbol \(<title\_court>\) as seen in Production 34 holds if the Production 35 is satisfied. The different title types that we have in this list are: Dr, Mrs, Mr and Miss followed by a period. Production 36 shows the generation for employee titles that we may appear (e.g. Sales Representative, Vice President, Chairman etc.) and \(n_9\) shows the total number of such titles.

\[
<cont\_title> \rightarrow cont_1 | \ldots | cont_{n_9} \\
<text>\_court> \rightarrow <title\_types> \\
<text>\_types> \rightarrow Dr | Mrs | Mr | Ms \\
<emp\_title> \rightarrow emp_1 | \ldots | emp_{n_{10}}
\]

Productions for supplier, category, description and notes is shown in 37-40. Hence, \(n_{11}\) to \(n_{14}\) show the total number of suppliers, categories, description and notes names.

\[
<supp> \rightarrow sp_1 | \ldots | sp_{n_{11}} \\
<categ> \rightarrow ctg_1 | \ldots | ctg_{n_{12}} \\
<desc> \rightarrow desc_1 | \ldots | desc_{n_{13}} \\
<notes> \rightarrow not_1 | \ldots | not_{n_{14}}
\]

Productions 41-42 holds if either of the entries in the symbols are generated. Production 41 shows either an individual’s gender is a male, a female or other as indicated in Production 41. The \(<discontinue>\) symbol shows either if a product should be continued or not as seen in Production 42.

\[
<gender> \rightarrow male | female | other \\
<discontinue> \rightarrow yes | no
\]

The \(<price>\) symbol satisfies the Productions 43-44. In this case, we opted for the South African currency symbol - Rands denoted as \(R\) (e.g R25.00). The \(<freight>\) shows a price if Productions 43 is satisfied.

\[
<price> \rightarrow R<d>^+ <period><d><d> \\
<freight> \rightarrow <price>
\]

Productions 45-49 show country, ship country, region, city and ship city. The \(<ship\_country>\) symbol is satisfied depending on the list of countries specified in Production 45. The countries that were generated in this production are: UK, USA, Germany, Australia, South Africa, Nigeria etc. \(n_{15}\) show the number of countries that appear in this list. Production 47 and 48 describe rules for the formulation of regions and cities. Here, \(n_{16}\) and \(n_{17}\) are the number of region and city names respectively. We enforced rules to ensure that this concatenation exists. For example, the city, Melbourne, matches with the Australian Victoria region. Production 49 holds if a city is generated, as seen in Productions 50-57 specifies the symbol for phone, fax, extension and ship code. Productions 50-52 generates a phone number where \(C\) is a 7-digit pseudorandom number. The \(<s\_code>\) symbol shows the prefixes that are used by service providers in South Africa (e.g. 061, 082, 084, 072, etc.). Production 53 shows the rules for a fax number. Every fax number is equivalent to a phone number. Production 54 holds if prefixes are satisfied in Production 52. Production 55 specify the rules for postal codes and \(n_{18}\) is the total number of postal code names. Productions 56 to 57 are recursively defined that allows more occurrences of values.

The \(<address>\) symbol satisfies the Productions 58-60. To generate an address, we specify a house number \(<d>\) followed by a street name \(<add\_list>\) and a city \(<city>\) (e.g. 54, Klein Street, Johannesburg). The \(<add\_list>\) symbol holds a street name with \(n_{19}\) specifying the total number of street names. The \(<ship\_add>\) holds if Production 58 is satisfied.

The \(<date>\) symbol satisfies Productions 61-70, and is composed of the terminal symbol: day of the week \(<d\_wk>\), days of the month \(<d\_mnth>\), month of the year \(<mnth\_y>\) and year \(<yr>\). The \(<birth\_d>\) symbol as indicated in Production 66 holds, if an individual is between the ages of 18 to a retirement age of 65, according to the Gregorian calendar. Productions 67 to 70 holds if a date is satisfied.
The productions for the Order Details table in Production 81 with fields — OrderID, ProductID, UnitPrice, Quantity are 16, 17, 43, 21 respectively.

\[ \text{<orddetails_tb>} \rightarrow \text{<ord_id>} \text{<prod_id>} \text{<quantity>} \text{<price>} \]

The Categories table yields the fields — CategoryID, CategoryName, Description with Productions (12, 24, 39) respectively. The formalism as seen in Production 82 for this table produces:

\[ \text{<categories_tb>} \rightarrow \text{<cat_id>} \text{<cat_name>} \text{<desc>} \]

The formalism for the Orders table; with fields such as OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Freight, ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, and ShipCountry as seen in Production 83.

\[ \text{<orders_tb>} \rightarrow \text{<ord_id>} \text{<cus_id>} \text{<emp_id>} \text{<order_d>} \text{<req_d>} \text{<ship_d>} \text{<ship_via>} \text{<freight>} \text{<ship_name>} \text{<ship_add>} \text{<ship_city>} \text{<region>} \text{<ship_pcode>} \text{<ship_country>} \]

The Customer table is formalised using its fields — CustomerID, CompanyName, ContactName, ContactTitle, Address, Region, Postalcode, Country as presented in Production 84.

\[ \text{<customer_tb>} \rightarrow \text{<cus_id>} \text{<comp_name>} \text{<cont_name>} \text{<cont_title>} \text{<address>} \text{<region>} \text{<p_code>} \text{<country>} \text{<phone>} \]

The formalism for the Product table with fields — ProductID, ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitsInStock, UnitsOnOrder, ReorderLevel and Discontinued as displayed in Production 85.

\[ \text{<product_tb>} \rightarrow \text{<prod_id>} \text{<prod_name>} \text{<sup_id>} \text{<cat_id>} \text{<quantity>} \text{<units_stk>} \text{<units_order>} \text{<reorder_l>} \text{<discontinue>} \]

The Employee table is formalised in Production 86 with fields — EmployeeID, Lastname, Firstname, Title, TitleofCourtesy, BirthDate, Hiredate, Address, City, Region, Postalcode, Country, Homephone, Extension, Photo, Notes, ReportsTo. The production for photo is beyond the scope of this paper.

The <homepage> symbol satisfies Productions 72-78. This rule basically specify use a protocol, followed by a colon and a double front slash with a host and domain name. This is followed by a period, a suffix, a single front slash and a folder, a single front slash and a file name. An example of the <homepage> symbol specify a complete web url address (e.g http://www.mydomain.com/folder/image.png). Production 79 describe a given name as described in Production 71 with a period and a file suffix.

Within Productions 1 to 78, we have defined the elements that are used to create the rules for the tables. Productions 80 to 87 specify the rules for the tables. The complete formalism for the Shippers table in Production 80 with fields — ShipperID, CompanyName and Phone is derived from the Productions (14, 26, 50) and presented below:

\[ \text{<shippers_tb>} \rightarrow \text{<ship_id>} \text{<comp_name>} \text{<phone>} \]
The formalism for the Supplier table is derived from the fields — SupplierID, CompanyName, ContactName, ContactTitle, City, Region, PostalCode, Country, Phone, Fax, Homepage, as presented in Production 87.

IV. IMPLEMENTATION AND RESULTS

We have implemented the productions as described in Section 3 and presented a hypothetical DB called the XNorthwind (or Extended Northwind). XNorthwind was implemented using the .Net framework and the synthesized datasets were stored in Microsoft SQL Server. The synthesiser produced 100,000 iterations of datasets as opposed to 3,200 tuples of the Northwind DB. We have presented the results of two tables: Shippers and Customers table. Figure 2 shows the datasets in the Shippers table with 1,000 tuples as opposed to three tuples in the Northwind DB. This is described in Production 80 in Section III. Figure 3 shows the datasets in the Customers table of the first 10,000 tuples as opposed to 91 tuples in the Northwind DB. We have described the Customers table in Production 84 in Section III.

V. APPLICATIONS OF XNORTHWIND

In this section, we present possible applications of the XNorthwind DB that was presented in Section IV. Possible applications of the XNorthwind DB are:

1) new products and services can be tested using this database,
2) given its volume, it can be widely used in CRM\textsuperscript{8} and ERP\textsuperscript{9} applications, and
3) used in ITS\textsuperscript{10} as a practice DB for teaching database concepts to students.

VI. EVALUATION

We obtained the results of the evaluation through an online survey. This survey was carried out to obtain feedback from respondents on their perceptions of the generated datasets and its usefulness. The respondents were mostly educators and students’ in the information systems and computer sciences disciplines from two South African universities: the University of Johannesburg and the University of the Witwatersrand. We received a total of 112 responses from the respondents. The respondents were initially asked to rate their knowledge with DBs on a rating scale (for example: one (1) indicating no experience at all and ten (10) for strongly experienced). We noticed that they all had knowledge with databases (See Figure 4(a)). Furthermore, we asked them if they have used the Northwind DB. 44.6% acknowledged that they have used the Northwind DB. We can agree that this number may represent students who may have only used the Microsoft DB for data storage without a clue of hypothetical datasets (See Figure 4(b)). Furthermore, we asked the respondents about the XNorthwind, and suggested if they think this DB can be useful to have. 94.6% agreed that the XNorthwind DB can be useful to have, 5.4% stayed indifferent and no respondent indicated that this DB is unusable (See Figure 4(c) – a combination of participants who ‘strongly agreed’ and ‘agreed’).

In addition, we asked the respondents if they think that the XNorthwind has wider application than the original Northwind. About 95.5% strongly believed that XNorthwind has wider application over the Northwind owing to the extra datasets. 4.5% stayed indifferent and no respondent agreed that XNorthwind is disadvantageous to have (See Figure 4(d)). Lastly, we asked the respondents to suggest an application of large datasets (XNorthwind). 41.1% suggested that large datasets can be used extensively. 58% had no idea and 0.9% stayed indifferent. We believe that majority of our respondents are students and may not have used sample DB (See Figure 4(e)). Given these feedback, we conclude that the generation of large datasets can be useful.
This paper has described a new approach for the generations of datasets using CFGs. The CFG rules were implemented, and large hypothetical data records were injected into an SQL Server database called the XNorthwind. The synthesized datasets were stored in Microsoft SQL Server. We have shown that this approach can be used to synthesize large datasets. Evaluation results obtained through a survey showed that majority of the participants agreed that large datasets can be useful.

In future, we will extend this tool to automatically generate picture fields as seen in the Category table.

ACKNOWLEDGMENT

This work is based on research supported by the National Research Foundation (NRF) of South Africa (Grant Number: 119041). Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the NRF does not accept liability in regard thereto.

REFERENCES

(a) Rate your experience with databases? (1 for no experience and 10 for strongly experienced)

(b) Usage of the Northwind DB

(c) Usefulness of the datasets of XNorthwind

(d) Wider application of XNorthwind

(e) Think of any application of large dataset

Fig. 4. The result of the evaluation


