UNIVERSITY OF NAIROBI

COLLEGE OF BIOLOGICAL AND PHYSICAL SCIENCES

SCHOOL OF COMPUTING AND INFORMATICS

Auto-ETL: A Generic Methodology for Transforming Entity-Attribute-Value (EAV) Data Model into Flat Tables using Form Metadata to Optimize Report Generation. A Case Study of Kenya EMR

Ojwang' Antony
P51/85879/2016

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in Applied Computing of the University of Nairobi.

DECEMBER, 2018

DECLARATION

This project is my original work and has not been presented for a degree in any other University

Signature……. Date……..

Ojwang' Antony
P51/85879/2016

Supervisor:

This proposal has been submitted for examination with my approval as University Supervisor.

Signature……. Date……..

Prof. Peter W. Wagacha

Supervisor

DEDICATION

I would like to dedicate this research to my family for the unending support they accorded me. Special thanks to colleagues and friends who gave ideas and ensured the research achieved the set objectives.

ACKNOWLEDGEMENTS

I would first like to thank God for providing us with the strength and the will to accomplish this research project. Many people have contributed to this project in different ways right from the beginning to its completion.

My special appreciation to the project supervisor, **Prof. Peter W. Wagacha**, for his valuable guidance that led to the successful completion of the project. He consistently allowed this work to be my own, but steered me in the right direction whenever he thought I needed it. We would also like to thank the panelists who ensured that the project remained as relevant as possible.

Special appreciation also to the Palladium Group – HMIS II project for the proper study environment and support throughout the study.

I would also like to thank the experts who were involved in the testing and validation of the methodology proposed in the study. Without their participation and input, the validation exercise could not have been successfully conducted.

Finally, I must express my sincere gratitude to my parents and the family at large for the support and continuous encouragement throughout the period of my study. This accomplishment would not have been possible without them.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

DDL	Data Definition Language
DHIS2	District Health Information Systems version 2
DML	Data Manipulation Language
EAV	Entity-Attribute-Value
EMR	Electronic Medical Record
ETL	Extract-Transform-Load
HIV	Human Immunodeficiency Virus
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
OpenMRS	Open Medical Records System
SQL	Structured Query Language
XML	eXtensible Markup Language

DEFINITION OF TERMS

Data Model - is the logical structure of a database. It defines how data is connected to each other and how they are processed and stored inside the system.

Data point - a discrete unit of information. In this study, a data point refers to a variable for handling data in a form.

DHIS2 - an open source software platform for reporting, analysis and dissemination of data for all health programs, developed by the Health Information Systems Programme (HISP)

Encounter - the interaction between patient and health care provider.

Entity-Attribute-Value (EAV) model - is a database model suitable for storing heterogeneous, sparse and highly volatile data.

ETL Tables/ETL Data source/ETL Dataset - a set of flat tables generated through ETL process to support data extraction and reporting functions.

Extract-Transform-Load (ETL) - a process that comprises of three database functions (extract, transform, load) that are combined into one tool to pull data from one database into another. The process uses a set of SQL statements for the operations.

Form Metadata - the description of a form's contents and the keywords linked to the content usually expressed in the form of meta-tags denoted by angle brackets i.e. <firstName>, <lastName>, <obs>, etc.

Heterogeneous data - is data of all types i.e. structured, semi-structured, and unstructured.

Html Form Entry Module - an OpenMRS module used to manage data collection forms designed in HTML format. The module uses normal HTML tags and additional custom HTML tags to define user interfaces and how data is stored and retrieved from OpenMRS data model.

Human Immunodeficiency Virus (HIV) - a virus that attacks the body's immune system. Both the virus and the infection it causes are called HIV.

KenyaEMR - a distribution of OpenMRS used to support health care in Kenya. It is mostly used in Comprehensive Care Center (CCC) for HIV Care and has active implementations in over 350 health facilities.

Observation - an assessment of a patient's condition, or analysis of data collected on one or more patients by the investigator/staff as required by protocol. It may also refer to a discrete datum (piece of information) collected during a clinical assessment.

Open Medical Records System (OpenMRS) - an Open source electronic medical records platform used to support delivery of health care in developing countries. OpenMRS has implemented the Entity-Attribute-Value (EAV) data model to store medical observations.

Sparse data - data where while several parameters are applicable, only a few are actually recorded for a typical patient.

Volatile data - data that changes so fast. That is, new variables are frequently introduced.

ABSTRACT

The use of Entity Attribute Value (EAV) data model is strongly favored by the existing research in both medical and other domains because of its flexibility and extensible nature that makes it suitable for storing highly sparse and heterogeneous medical data/observations. The model is however slow in data extraction and reporting and as a result affects timely decisions for patient care. Research has shown that pivoting of the EAV data through ETL (Extract, Transform and Load) process is the ideal solution to the data extraction and reporting problems in the EAV based systems. The ETL process has largely been a manual process which was costly, slow, tedious, and prone to human errors. At present, a number of studies have documented approaches to automate the different parts of the ETL process which consist of generation of ETL specifications, rule composition, and SQL statement generation. With automation, the process enjoys reduced human involvement and consequently improve the accuracy and reduce the overall time and cost in the design, development, and implementation of the process. However, there has not been full automation and experts are still involved, either fully or partially, in supporting the generation of rules for the process hence the disadvantages of a manual process.

This study introduces Auto-ETL methodology that uses form metadata to fully automate the ETL rule generation and SQL generation phases and partly the less demanding ETL specifications phase of the process.

To evaluate the new methodology, we conducted a pilot experiment to compare efficiency and accuracy of the Auto-ETL methodology to a purely manual process. We also conducted analysis of triage variables (weight, height, temperature, respiratory rate) data extracted from both the EAV model and the generated tables from two facility EMR databases. Results of the pilot experiment shows that Auto-ETL is faster and more accurate in generating queries. The generated datasets also showed great simplification in SQL query complexity and performance compared to those for EAV model. For analysis, descriptive data characteristics from the source and generated tables was presented by use of median and range (minimum, maximum) for the continuous variables. Lin's concordance correlation with 95% confidence interval (CI) was used to measure the agreement between the source dataset and generated tables for the study variables and an agreement will be considered if the concordance correlation is closer to 1. The R squared was used to report the relationship between the variables form the source dataset and generated tables and considered to have a strong relationship if closer to 1.Statistically significant association will be considered when $p < 0.05$.
For descriptive characteristics, results showed same median across the two data sources and a p-value of <0.001 for R squared and CI value of 0.997 for concordance relationship. These are indications of a strong concordance between source data and ETL data and consequently informs the conclusion that form metadata can be used to support generation of SQL statements for ETL process with zero data loss.

Keywords: ETL, pivoting, Encounter, Entity-Attribute-Value, OpenMRS, KenyaEMR

CHAPTER ONE: INTRODUCTION

Effective and efficient data management practices is critical in supporting the decision-making function in any organization. The ability to quickly collect quality data, analyze and use what has been learned to help nurture better decision-making is the dream of many organizations across all sectors. But for a sector like health care that experiences numerous hurdles including data complexity, sparseness, constant evolution, and a high demand for data, it is much harder and requires proper planning and execution around data storage and preparations for reporting and analysis.

Health information has been identified as a key component in promoting health care by supporting evidence-based decisions and policy-making in the health sector. World Health Organization in its progress report "Towards Universal Access" and the Ministry of Health, Kenya identified 5 key Strategic Directions towards realizing universal access to comprehensive HIV/AIDS prevention programs, treatment, care, and support (Ministry of Health, 2013; World Health Organization, 2007). These included among others strengthening and expansion of health systems and investing in strategic information to guide decisions and timely interventions.

A functional health system is one which provides reliable and timely information on health determinants, health status, and health system performance and further analyze data to guide activities across other key strategic directions. In this regard, many developed and developing countries have adopted information technology solutions such as Electronic Medical Records (EMR) systems to manage health information and support delivery of health care services through timely data entry, reduced data entry errors, fast and accurate reporting, reduced patient waiting time, and lowered cost of medical services , and continuity of care services across different care providers/centers (Fraser, Blaya, Choi, Bonilla, & Jazayeri, 2006; Jawhari, Ludwick, Keenan, Zakus, & Hayward, 2016; World Health Organization, 2007).

1.1 Data Models in electronic information systems

There are two main data models in electronic information systems and the choice of what is adopted is wholly dependent on the nature of data that a given information system is to handle. The main data models include the Entity-Attribute-Value (EAV) and conventional relational model.

1.1.1 The Conventional Data Model

In a conventional relational model, information about attributes on an object (entity or a "thing" for instance patient) is stored as columns in a table with each attribute presented as a column (Chen et al., 2000; Dinu & Nadkarni, 2007; Nadkarni & Marenco, 2001). The data model suits objects with a fixed number of attributes/characteristics and in cases where only a few variables can be empty for a given instance.

1.1.2 The EAV Data Model

The EAV data model implements a row-model where each row records one or more related facts (object type, attribute, and attribute value) about an entity/object. The model provides a flexible and simple database schema for storage of sparse and heterogeneous data (Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Löper, Klettke, Bruder, & Heuer, 2013). The EAV data model is predominant in the health care and other sectors where highly heterogeneous, sparse and rapidly evolving data i.e. medical observations is generated.

The EAV data model, compared to the conventional relational model, supports the ability to define new attributes without additional programming and schema changes. It also provides compact storage handling of sparse data, and simple database design (Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Luo & Frey, 2016). However, the data model is complex and requires complex queries to extract population-based variables. Again, queries take a longer time to complete execution and the model is also not readily usable with existing statistical packages (Anhoj, 2003; Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Löper et al., 2013).

In the health sector, for instance, different studies have shown growing interest in EMRs since they provide for secondary use of data for research and consequently improve safety, quality and efficiency of healthcare. Data in the EMR can be used to support research activities in areas such as population and disease, detection of adverse drug reaction in patients, assessment of treatment outcomes, shaping of health policies and guidance on effective use of resources (Fraser et al., 2006; Jawhari et al., 2016; Van Velthoven, Mastellos, Majeed, O'Donoghue, & Car, 2016; World Health Organization, 2007; World Health Organization (WHO), 2007).

To support effective and extensive medical research and analysis, EMRs should not only provide high-quality data but also promote fast, timely and accurate reporting as well as providing readily usable datasets for data mining, analytics and visualizations. Whereas EMRs with conventional relational model readily support reporting, those with the EAV data model which form the majority have inefficiencies with regard to data extraction and reporting. This greatly affects data use and meaningful use of health information in decision making.

1.1.3 Handling data extraction and reporting challenges in the EAV Data Model

Different studies have highlighted solutions to help improve data extraction in the EAV data model. The highlighted solutions include use of multiple simple queries in place of the complex ones, addition of more resources i.e. computer memory and processor speed, pivoting of EAV data into several conventional tables that meet reporting needs, use of Optimized Entity-Attribute-Value (OEAV) model and use of Dynamic Tables (Chen et al., 2000; Corwin, Silberschatz, Miller, & Marenco, 2007; Dinu & Nadkarni, 2007; Ong et al., 2017; Paul & Hoque, 2011). Most studies, however, regard ***pivoting*** of EAV data to be the solution to the data extraction problem since it provides conventional/flat tables for easy and fast reporting (Dinu & Nadkarni, 2007; Dinu, Nadkarni, & Brandt, 2006; Luo & Frey, 2016; Nadkarni & Marenco, 2001).

1.1.4 Extract-Transform-Load (ETL) Process for EAV Data Model

Pivoting involves the transformation of data in the EAV data model through Extract-Transform-Load (ETL) process into numerous conventional/flat tables that meet the reporting needs of users. It uses a set of Structured Query Language (SQL) statements for data extraction, transformation, and loading (Dinu & Nadkarni, 2007; Dinu et al., 2006; Luo & Frey, 2016; Nadkarni & Marenco, 2001).

The design and development of an ETL solution has three major phases: generation of ETL specifications, generation of transformation rules (ETL rules), and generation SQL statements from the rules and specifications. The generation of ETL specifications, the first phase, defines the basic, one-time/first-time setup configurations for the ETL process and does not require specialized skill set. The rule generation phase entails mapping of source data to target data and the definition of all transformations required for the mapping. The

SQL statement generation phase is responsible for translating ETL specifications and the generated ETL rules into a set of SQL statements that supports the extraction, transformation, and eventual loading of data from source to target datasets. The rule generation and SQL statements' generation phases are evidently complex and require specialized skill set to execute. They are also most expensive in the whole process.

A number of studies have described the manual design, development, and implementation of an ETL process in which domain experts work together with SQL programmers and other health experts to adequately define all mappings and transformation rules for the ETL process. The approach is therefore complex, tiresome, tedious, erroneous, and costly to sustain especially in cases of frequent changes that need adjustments in the existing solution. A few studies have developed methodologies to automate parts of the ETL process. In the documented automated approaches, domain experts and ETL designers formulate mappings and transformation rules for the ETL process. The generated rules are then automatically converted into SQL statements using custom applications. The automated approach, even though not complete, shifts focus of data experts from the manual SQL statement writing to the definition of information (rules) to aid the automation of SQL statements generation. It therefore saves time, cost, and human errors in the query generation phase and promotes efficiency in the development of an ETL solution (Lin & Haug, 2006; Mate et al., 2015; Nadkarni & Marenco, 2001; Ong et al., 2017).

This study's main aim was to extend existing automated methodologies and provide a generic approach of generating ETL rules and associated SQL statements from form metadata present in the information systems. The resulting queries are responsible for the initial/incremental creation/update of the generated flat datasets for reporting purposes. In the end, this study sought to provide the following:

1. Automated ETL rules and SQL statements generation based on form metadata. The solution will scan through form metadata and generate the rules and SQL statements for the ETL process.
2. Ontological/contextual data extraction and loading to avoid ambiguities and preserve meaning and contexts of data variables as used in forms.
3. Automatic incremental updates of ETL tables when form metadata changes

1.2 Problem Statement

Effective and efficient data management, and faster access to data are key functions when it comes to electronic information systems. The management of heterogeneous, sparse, and rapidly evolving data presents a number of challenges including high percentage of null values, frequent changes in schema, which requires changes in the corresponding electronic information systems. The use of Entity Attribute Value (EAV) storage model is strongly favored by the existing research in both medical and other domains because of its flexible and extensible nature which makes it suitable for storing highly sparse and heterogeneous data i.e. the medical data/observations, unlike the conventional relational model which requires system design and schema changes for every change. The EAV model, however, has known problems of data extraction that makes it very inefficient and unresponsive to data needs thereby affecting effective and timely evidence-based decisions. Research has shown that pivoting of EAV data through ETL process to a set of flat/conventional tables greatly improves data extraction and consequently report generation. Pivoting comprises of three major phases: generation of ETL specifications, generation of data transformation rules, and generation of SQL statements from specifications and rules for the extraction, transformation, and loading operations. Previously, all the three phases were purely manual and dependent on the ETL designers, SQL programmers and other health data experts who spent their entire time in the design, development of implementation of an ETL solution. Whereas the ETL specifications phase is light, mostly one-time, and requires basic knowledge, formulation of ETL transformation rules and generation of SQL statements are very complex and requires extensive human effort and knowledge. The manual process, therefore, was not only costly but also slow, tedious, and prone to human errors leading to inefficiencies in the design, development and implementation of the ETL process.

Through continued research, a number of studies have documented approaches to automate the SQL statements generation phase of the ETL process which is a great step towards reducing human involvement in the process. However, experts are still involved in the rule formulation phase which is equally complex, labor intensive, costly, tedious, demanding, and prone to errors.

1.2.1 Proposed solution

To eliminate the manual generation of ETL rules and consequently improve the accuracy and efficiency in the design and development of an ETL process, this study sought to

develop a generic methodology to pivoting that uses form metadata in the system to automate the creation of ETL rules and the SQL statements used to create flat tables for reporting.

1.2.2 Research Objectives

1. To review existing ETL methodologies for EAV data model and identify limitations
2. To design and develop a suitable ETL methodology that addresses the limitations by exploiting the metadata.
3. To evaluate and validate the new methodology for accuracy and efficiency.

CHAPTER TWO: LITERATURE REVIEW

Effective and efficient data management practices is critical in supporting the decision-making function in any organization. The ability to quickly collect quality data, analyze and use what has been learned to help nurture better decision-making is the dream of many organizations in different sectors. But for a sector like health care which experiences numerous hurdles that include data complexity, sparseness, constant evolution, and a high demand for data, it is much harder and requires proper planning and execution around data storage and preparations for reporting and analysis.

Health information has been identified as a key component in promoting health care by supporting evidence-based decisions and policy-making. The World Health Organization (WHO) in its report "***Towards Universal Access***" identified 5 key Strategic Directions towards realization of universal access to comprehensive HIV/AIDS prevention programs, treatment, care, and support. These included among others strengthening and expansion of health systems and investing in strategic information to guide decisions and timely interventions (World Health Organization, 2007).

The Kenya Ministry of Health (MOH) in its report (MOH, 2013) also prioritized health information as a key driver towards achieving "**Universal Health Coverage**" and promotes investments in the areas of data generation, validation, analysis, dissemination, and utilization of routine health information (Ministry of Health, 2013).

World Health Organization (WHO) defined a functional health system as one which provides reliable and timely information on health determinants, health status, and health system performance and further analyze data to guide activities across other key strategic directions (World Health Organization, 2007). In this regard, many developed and developing countries have adopted Electronic Medical Records (EMR) systems to manage patient data; collection, storage, analysis and dissemination of information (Fraser et al., 2006; Jawhari et al., 2016). Easy access and retrieval of electronic data and reports from EMR systems has not only reduced patient waiting time in facilities but also supported evidence-based planning, decision making, implementation and accountability of healthcare interventions and treatment programs. EMR systems reduce data entry errors through inbuilt validation functions thus ensuring high-quality health information. In

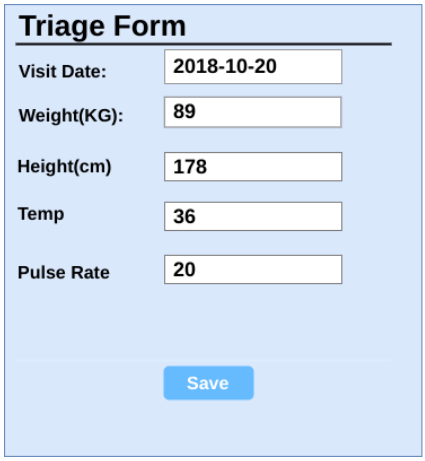
addition, in a networked environment patient information can be shared across multiple facilities thus improving continuity of care (Fraser et al., 2006; Jawhari et al., 2016).

2.1 Data Models in Information systems

A data model defines the logical structure of a database. It defines the relationship between data elements and how data is processed and stored in the system. Depending on the nature of data to be handled i.e. complexity, sparseness and volatility, different information systems adopt different data models for effective and efficient data management. The main data models include the conventional relational model and the Entity-Attribute-Value (EAV) data model.

2.1.1 Conventional and EAV Data Models

In a **conventional relational** database model, information about **attributes** on an object (entity or a "thing" i.e. patient) is stored as **columns** in a table with each attribute presented as a column (Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008). This data model suits objects with a fixed number of attributes and in cases where only a few variables can be empty for a given instance. **EAV data model** on the other hand implements a row-model where **each row** records one or more related facts (object type, attribute, and attribute value) about an entity/object. The model offers a flexible and simple database schema for storage of sparse and heterogeneous data (Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Lin & Haug, 2006; Löper et al., 2013). **Figure 1** illustrates approaches used by the two data models in saving data.



EAV Data Model

PatientId	EncounterDate	Attribute	Value
7891	2018-10-12	Weight	89
7891	2018-10-12	Height	178
7891	2018-10-12	Temp	36
7891	2018-10-12	Pulse	20

Conventional Relational Model

PatientId	EncounterDate	Weight	Height	Temp	Pulse
7891	2018-10-12	89	178	36	20

Figure 1: EAV vs Conventional Data Models

2.1.2 Data extraction in Conventional and EAV Data Models

Compared to EAV, the performance of queries on conventional relational data model is faster and is 3 to 5 times more efficient than in EAV model. However, conventional data model does not work well for heterogeneous and sparse data which requires schema changes (programming) every time a new attribute (data element) is added. It is also prone to storage of NULL attribute values in the database which unnecessarily occupies space (Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Löper et al., 2013).

Existing studies have presented the following of EAV model over the conventional model (Anhoj, 2003; Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Lin & Haug, 2006; Löper et al., 2013).

1. It is possible to define new attributes (data elements) without additional programming and schema changes.
2. Compact storage handling of sparse data as it does not record NULL attribute values.
3. A query for finding all the data about an object (entity-centric) has to access only one table and is thus fast.
4. It has a simple database design which uses fewer tables and supports storage of different types of data in a single table. This makes it easier to maintain.

2.1.3 EAV Data Model in Healthcare Systems

Healthcare/medical data has been described to be sparse, heterogeneous, and highly volatile and as such requires a data model that is flexible and accommodative of the dynamic nature of healthcare. Many health care systems, for instance Clinical Data Repositories (CDRs), EMR systems i.e. OpenMRS, Regenstrief EMR, DHIS2, etc, have implemented the EAV data model due to its flexibility and extensibility (Anhoj, 2003; Chen et al., 2000; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Lin & Haug, 2006; Löper et al., 2013).

The model, however, has a downside of slow query performance and high query complexity thus making EAV based systems less efficient in data extraction with the inefficiency increasing as the data in the database continues to grow (Chen et al., 2000; Corwin et al., 2007; Dinu & Nadkarni, 2007; Eessaar & Soobik, 2008; Paul & Hoque, 2011).

2.1.4 Handling the EAV data extraction and reporting challenges in Healthcare Systems

Different studies have shown different solutions to help improve data extraction in the EAV data model and they include several that we discuss in the following sections:-

2.1.4.1 Use of multiple simple queries in place of one complex

Studies have shown that database engines have better execution plans for simple queries compared to complex ones. Many queries may, however, translate into many requests to the server and this can affect performance. Again, it involves a lot more programming to join together results of different queries into one (Chen et al., 2000; Dinu & Nadkarni, 2007).

2.1.4.2 Addition of more resources to boost performance

Addition of computer memory and processor speed in a resource-constrained server has always improved performance through the improved number of requests serviced, caching and buffering of query results. However, this is usually a short-term solution as long as data is small (Chen et al., 2000).

2.1.4.3 Pivoting of EAV data

This approach uses ETL process to extract, transform and load data from EAV data model to a set of flat tables for faster data extraction and reporting (Dinu & Nadkarni, 2007; Dinu et al., 2006; Lin & Haug, 2006; Luo & Frey, 2016; Mate et al., 2015; Moturi, Mburu, &

Ngaruiya, 2016; Nadkarni & Marenco, 2001; Ong et al., 2017). It has the following benefits including:

1. Separation of the transactional database from the reporting database.
2. Ability to use statistical packages i.e. STATA for in-depth analysis. Data in the EAV data model is not friendly to most statistical packages.
3. Provides ready tables for data mining, visualizations and ad hoc querying
4. It's not disruptive to existing transactional process i.e. does not require changing of the code base for normal system operations.

2.1.4.4 Use of Optimized Entity-Attribute-Value (OEAV) model

OEAV is an open schema model that removes the search inefficiency of EAV model while preserving the benefit of storing heterogeneous and highly sparse data. OEAV model has been optimized for reading thus is faster (15-70% times) in data extraction compared to the EAV model (Paul & Hoque, 2011).

Although a good alternative, implementation of OEAV data model would require at a minimum a) migration of existing data to the new model and b) re-engineering of the underlying code base to work with the new model all of which are disruptive and costly. Again, the new model has no established implementation where its strengths and weaknesses can be better analyzed.

2.1.4.5 Use of Dynamic Tables

Dynamic tables provide a search efficient model as well as the ability to handle heterogeneous and sparse clinical observations without the need for schema changes. This approach requires modifications at the database engine's level to provide a decomposed model while maintaining a logical view of the data (Corwin et al., 2007). Even though very promising, the only documented implementation works in PostgreSQL database. This requires a similar solution for other database engines like MySQL, SQL server among others.

Existing research favours pivoting as a sure way of promoting faster data extraction and reporting in EAV based systems. This is done through extraction of data from EAV data model into prepared flat tables/datasets responsive to data use requirements. Pivoting is achieved through a process known as Extract-Transform-Load (ETL) and is described in

the following section 2.1.5.

2.1.5 Extract-Transform-Load (ETL) Process

Pivoting involves the extraction and transformation of data in EAV data model, and eventual loading into conventional/flat tables/datasets that meet the reporting needs of users through **Extract-Transform-Load (ETL)** process. ETL process uses a set of Structured Query Language (SQL) statements to support data extraction, transformation, and loading into target datasets (Lin & Haug, 2006; Mate et al., 2015; Nadkarni & Marenco, 2001; Ong et al., 2017). Previously, the process of generating such queries was purely manual and domain experts i.e. ETL designers created from scratch all queries to support ETL process. The process was therefore costly, tiresome, slow, tedious and prone to human errors. Through continued research, a number of studies (Lin & Haug, 2006; Mate et al., 2015; Ong et al., 2017) have attempted to automate the process (ETL) through a series of steps that can be generally categorized under three stages: generation of ETL specifications, generation of data transformation rules, and generation of SQL statements for ETL process from specifications and rules. Out of the three stages, existing research has only automated the stage for SQL statement generation with the rest manually accomplished by ETL designers and/or data managers. The manual stages (ETL specifications and rule generation) are labour intensive as one has to do mappings for all entities and all data points in each entity. As a result, the process is still slow, costly, and prone to errors due to considerable level of human effort required.

This study therefore aimed to improve on existing research and develop a generic process to pivoting that uses form metadata present in the system to create specifications and rules from which SQL statements to create conventional/flat tables are generated automatically. This will speed up the overall speed of defining ETL process while maintaining data semantics across the generated flat tables. Automation will also reduce costs since ETL designers and/or data managers will only be verifying generated SQL statements and results.

2.2 Null Hypothesis

Data entry form metadata cannot be used to generate flat/conventional tables for reporting in the EAV data model.

2.3 The significance of the Research

This research provides automatic reporting datasets and improve the meaningful use of health data in EMR systems with EAV data model. These will be through the following:

1. Faster design and development of ETL process through automation of ETL rules generation.
2. Flat datasets for fast and timely reporting, analysis, mining, and data visualizations.
3. Use of table-per-form approach promotes ad-hoc querying since users easily relate forms to reporting tables in the system.
4. Reduced developer time in generating SQL statements for ETL process
5. Reduced human errors in defining ETL rules

2.4 Scope and Limitation

This research is based on health care systems that implement EAV data model and use a standard data dictionary to define form metadata in the HTML format and handling of structured and coded data in the data model. We have developed a prototype based on ***KenyaEMR*** to demonstrate the Auto-ETL methodology that uses form metadata to design and generate SQL statements used support the creation of a flat table for every form in the system.

2.5 Research Assumptions

The below assumptions are made for this project;

1. That the patient record in the EMR is complete, clean and of good quality.
2. That the EMR will be used after express authorization from the Palladium Group and MOH and facility leadership

2.6 Conceptual Framework

Conceptual framework is a system of concepts, assumptions, expectations, beliefs, and theories that supports and informs a research. **Figure 2** illustrates the main ideas in this research.

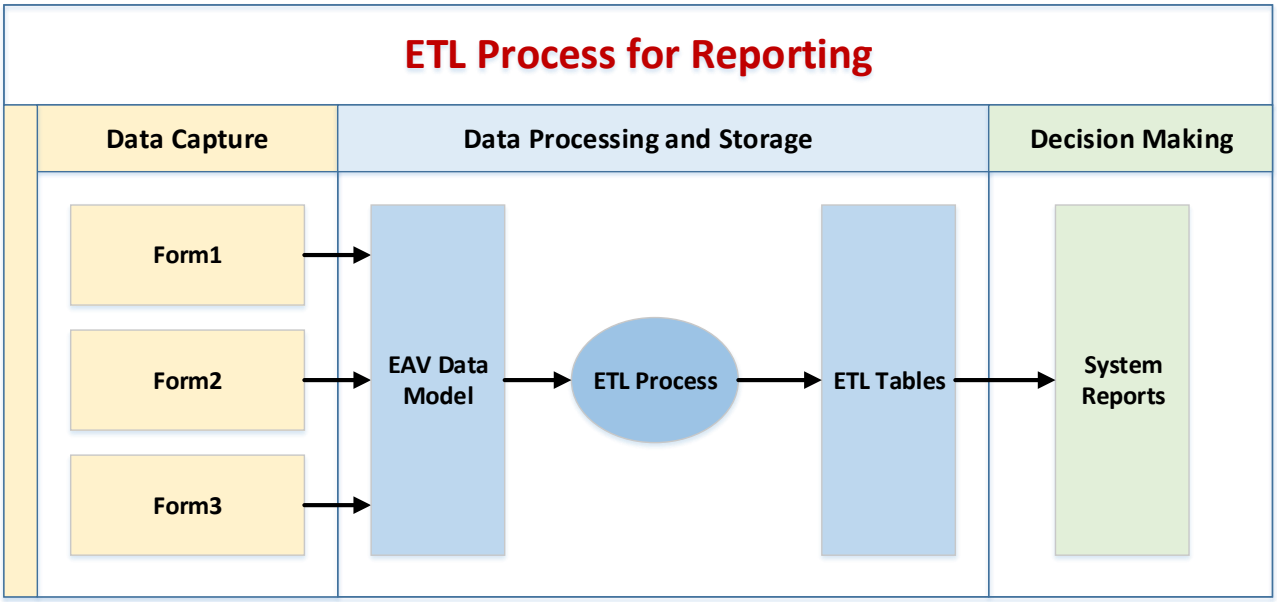


Figure 2- Conceptual Framework

CHAPTER THREE: METHODOLOGY

3.1 Introduction

In the sections below, we discuss the methodology used to address each research question.

This study has used Information Technology (IT) product design and development strategy, and an agile methodology that provides for iterative development and testing. The study has developed a working add-on module for ***Kenya EMR*** as software prototype.

3.2 To review existing ETL methodologies for EAV data model and identify limitations

3.2.1 Dynamic ETL: a hybrid approach for health data extraction

Describes a methodology for integrating data from various sources to a common data repository using a hybrid rule-based ETL work flow with the following components **(Ong et al., 2017)**.

1. ETL specifications written in plain narrative text and diagrams and describes the plan for the entire ETL process. It contains information about the scope of data for extraction, target datasets, input and output data files.
2. D-ETL rule design where mappings required to accurately extract, transform and load data from source to target datasets are composed. It requires basic information on source and target data sources (database, table, and field) as well as transformation formula required for variables to be extracted. **Figure 3** shows more information about D-ETL rule design.
3. ETL engine that generates SQL statements from ETL rules. The SQL statements are the queries responsible to extract, transform, conform and load data from source tables to target datasets.
4. Testing and debugging of generated queries by health data domain experts.

Table 2 Example of a D-ETL rule that loads data into the Care_site table in OMOP from a claims-based source CSV file

Rule Order	Rule Description	Target Table	Target Column	Map Type	Map Order	Source Table	Source Value
1	Medical_claims to Care_site	Care_site		PRIMARY	1	Medical_claims	medical_claims.billing_provider_id, medical_claims.place_of_service_code, provider.provider_organization_type
1	Medical_claims to Care_site	Care_site		JOIN	2	provider	medical_claims.billing_provider_id = provider.provider_id
1	Medical_claims to Care_site	Care_site		WHERE	3		provider.provider_organization_type in ('1', '2')
1	Medical_claims to Care_site	Care_site	care_site_source_value	VALUE	4		medical_claims.billing_provider_id || '-' ||medical_claims.place_of_service_code||'-' ||provider.provider_organization_type
1	Medical_claims to Care_site	Care_site	organization_source_value	VALUE	5		NULL
1	Medical_claims to Care_site	Care_site	place_of_service_source_value	VALUE	6		medical_claims.place_of_service_code
1	Medical_claims to Care_site	Care_site	care_site_address_1	VALUE	7		provider.provider_address_first_line
1	Medical_claims to Care_site	Care_site	care_site_address_2	VALUE	8		provider.provider_street
1	Medical_claims to Care_site	Care_site	care_site_city	VALUE	9		provider.provider_city
1	Medical_claims to Care_site	Care_site	care_site_state	VALUE	10		provider.provider_state
1	Medical_claims to Care_site	Care_site	care_site_zip	VALUE	11		provider.provider_zip
1	Medical_claims to Care_site	Care_site	care_site_county	VALUE	12		NULL

Figure 3- Screenshot of CSV file showing D-ETL rule design (Ong et al., 2017)

The Dynamic-ETL methodology automates the process of generating SQL statements from D-ETL rules and this improves the speed and accuracy of SQL statement generation process. It also improves transparency of generated SQL statements since it provides ETL designers the ability to test and debug generated queries and this helps with building ownership of the generated datasets. However, the methodology requires ETL designers

and other domain experts in the design of D-ETL rules which involves filling of CSV templates, similar to the one shown in **Figure 3**, for every data source to be pivoted. This manual process is slow, tedious, costly, and prone to human errors.

3.2.2 Easing the transition between EAV and conventional databases for scientific

Describes a process and a bi-directional tool for converting data between the Entity-Attribute-Value with Classes and Relationships (EAV/CR) and the conventional data models. The methodology has defined the following steps **(Nadkarni & Marenco, 2001)**:

1. **Schema extraction:** involves creation of schema metadata showing class/table definitions and relationships, and how tables, columns and rows of a conventional database map to classes, attributes and objects of EAV/CR. The generated metadata for a particular source is stored for subsequent pivoting operations.
2. **Transferring data:** Makes use of mapping metadata created by domain experts. The mapping metadata specifies mappings between tables/columns in the source/destination data and classes/attributes in EAV/CR models.

Even though a good ETL methodology, the process is still heavily dependent on human effort in generating metadata mapping document. In addition, for complex conversion expressions, there is a need for programming skills since the constructs used in the study are for a Visual Basic programming language. Again, the methodology applies to EAV/CR model that is a different flavor of EAV and still has no documented wide implementations.

3.2.3 Ontology based data integration between clinical and research systems

Describes an ontology-based approach to ETL for complex data extraction and transformation and at the same time maintaining data semantics. The study uses ontologies to organize and describe medical concepts of source and target systems. The ontologies are as described below **(Mate et al., 2015)**:

1. **Source ontology:** describes all data elements in the source system and their hierarchical organization.
2. **Target ontology:** describes the collection of concepts to be loaded into the target system/dataset. It contains the syntax and semantics linked to every data element and is used to generate metadata information for the target system.
3. **Mapping ontology:** provides linkages between the source and the target

ontologies. It contains information that maps every data element in the source ontology to the corresponding data element in the target ontology including all transformation rules.

Using the ontologies, SQL statements for the entire ETL process are generated to extract, transform, and load data from source to target databases. Figure shows the process of mapping source to target ontologies.

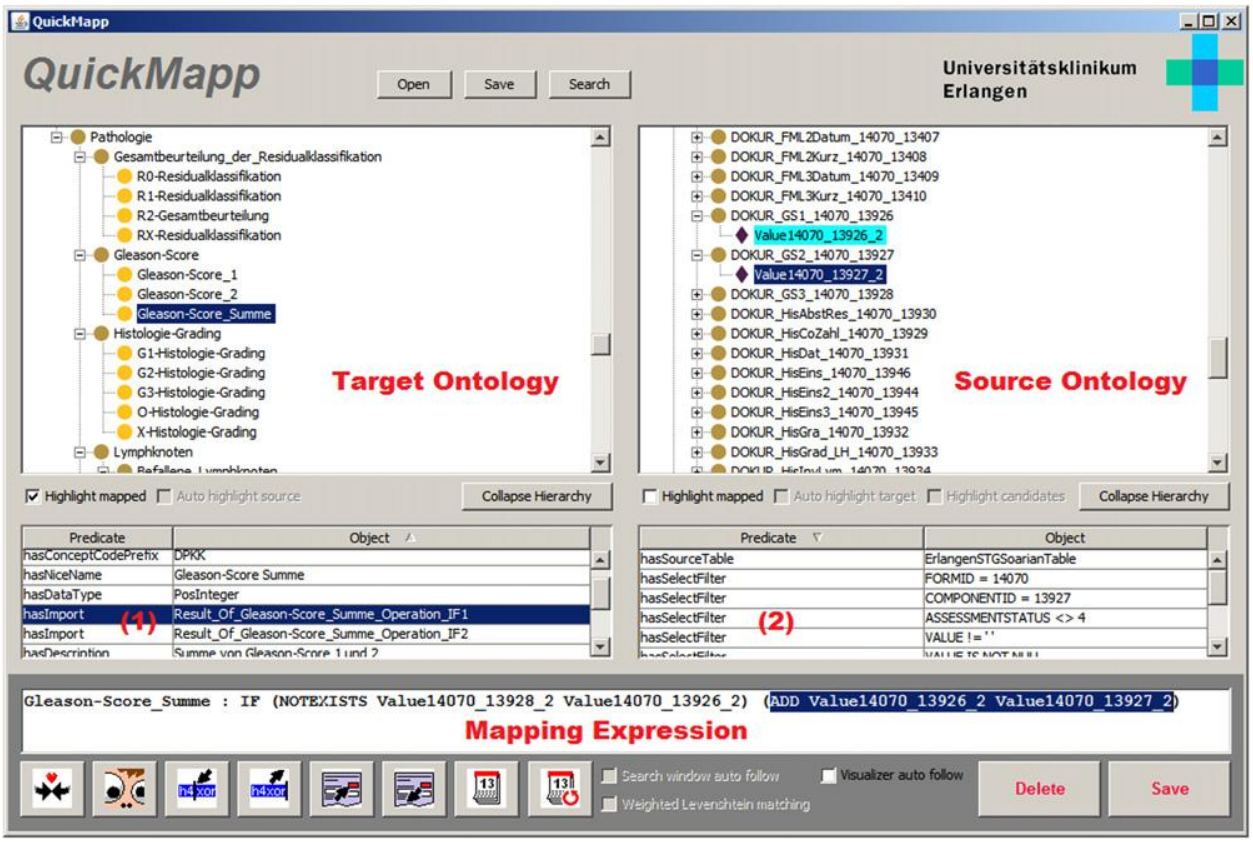


Figure 4 - Graphical Mapping tool for ontologies (Mate et al., 2015)

This work has the benefits of generating SQL statements from rules defined in the different ontologies thereby eliminating the need for SQL programmer. It has also used ontologies to define relationships in the source data thus retaining data semantics in the generated datasets. However, the definition of ontologies requires human expertise because semantic relationships have to be manually created as shown in Figure . The work is also documented to only support structured non-coded data thus limiting its applicability.

3.2.4 Data preparation framework for preprocessing clinical data in data mining

This is a contribution by (Lin & Haug, 2006) and it describes a framework that creates ETL rules based on statistical characteristics (occurrence, distinct values, mean, and variance) of the data, the metadata that describes the source database, and the medical knowledge provided by experts and/or in medical knowledge base. The rules can then be used for data preparation, attribute selection, and data transformation for loading into data-mining ready datasets.

The methodology presented is indeed a valuable contribution but still require human effort for the definition of ETL rules more so where the data dictionary does not provide adequate description of codes used in the system. Human effort is also required in defining semantic linkages between data elements and target datasets.

3.3 Design and develop a suitable ETL methodology that addresses identified limitations in the existing solutions by exploiting the metadata

ETL methodologies discussed in section 3.2 have identified the following main phases of an ETL process.

1. Creation of ETL specifications
2. Rule generation
3. Generation of SQL statements to support extraction transformation and loading operations.

ETL specifications describe high level configurations for pivoting process and may include defining the target database name, process schedule/interval, and selecting the list of forms for pivoting. This phase requires basic understanding of ETL process and has no dependency on domain knowledge. Rule generation involves creation of ETL rules that provides mapping information of source and target datasets. The rules also describe all preparatory operations to be performed on source data elements before loading into target dataset(s). The third phase generates all Data Definition Language (DDL) and Data Manipulation Language (DML) SQL statements from rules. The generated DDL statements are used to support initial creation and future modifications of target datasets' structure while DML statements are used for extraction, transformation, and loading of data from source to the target datasets. It is in the second phase (rule generation) where extensive domain expertise is required for accurate mappings and creation of relevant datasets.

Whereas great automation has been achieved in the third phase (SQL statements generation) of ETL process, rule ***generation***, the second phase, still requires considerable amount of human effort and manual process. Domain experts still have to create mappings between source and target datasets in a manual (Mate et al., 2015; Nadkarni & Marenco, 2001; Ong et al., 2017) or semi-automated process (Lin & Haug, 2006) using data and metadata of the source system. This makes the process slow, tedious, error prone, and costly since an expert has to be engaged. It becomes more unsustainable if changes are to be done more frequently like in health care where medical observations are highly volatile and has very high data demand. Keeping up with such frequent changes in data collected requires a flexible ETL methodology that is capable of automatically updating the structure of target datasets as new data elements are introduced. In this study, we discuss a methodology named "**Auto-ETL**" which uses metadata of forms in the system, an enhancement to use of metadata described by (Lin & Haug, 2006) work, to generate ETL rules for all data elements on a form and consequently apply **table-per-form** approach, similar to ontologies (Mate et al., 2015), to create flat dataset for every form marked for pivoting. Auto-ETL has used the **table-per-form** approach for the following reasons (Mate et al., 2015):

1. It is easy to maintain data semantics and variable-form relationships in the resulting flat datasets. This makes it easy for users to navigate the flat datasets and design ad hoc queries that support their ad hoc needs since data collected by a form reside in the same flat table.
2. For systems with data dictionaries, it is possible to use a generic code/concept multiple times in multiple contexts/forms and still retain its meaning across generated flat datasets.

We have developed a prototype for Auto-ETL using the following open source technologies:

1. Java programming language (java 1.7)
2. Groovy templates for interface design
3. HTML and javascript for user interface design
4. MySQL community edition database (version 5.6)
5. KenyaEMR

3.4 To evaluate and validate the new methodology for accuracy and efficiency

In order to evaluate effectiveness and accuracy of Auto-ETL, the following will be used to show how well the methodology solves the problem.

1. Unit tests
2. Pilot experiment to compare speed, accuracy and efficiency of Auto-ETL to a pure manual process
3. Analysis of data extracted from EAV data model and generated datasets. The results to include median for descriptive characteristics and correlation index for concordance relationship. These will show accuracy of the Auto-ETL methodology.

3.5 Data processing and analysis

Data was extracted from both EAV and ETL datasets and was analyzed using Stata version 14.2 for Windows©. The data characteristics from the source and generated tables was presented by use of median and range (minimum, maximum) for the continuous variables. Further presentation of the distribution was presented by use of box plots that showed the minimum, lower quartile, median, upper quartile and maximum. Lin's concordance correlation with 95% confidence interval (CI) was used to measure the agreement between the source dataset and generated tables for the study variables. A strong agreement will be considered if the concordance correlation is closer to 1. This was supported by use of scatter plots that had a regression line to show the relationship between the selected variables from the source data and generated tables. The R squared was used to report the relationship between the variables from the source dataset and generated tables and considered to have a strong relationship if closer to 1. Statistically significant association will be considered when $p < 0.05$.

3.6 Data Presentation

The analyzed data were presented using descriptive statistics such as tables, graphs, pie charts, and box plots.

CHAPTER FOUR: SYSTEM ANALYSIS, DESIGN, IMPLEMENTATION AND TESTING

4.1 Introduction

This chapter presents the Auto-ETL prototype that will help generate ETL tables automatically from form metadata. The goal of the system is to have a transparent and highly customizable solution for end users which should address the following design objectives:

1. ETL process configuration/customization
2. Role-based access to Auto-ETL
3. Incremental updates of ETL tables
4. Storage of generated ETL queries for reuse
5. Review, testing, and optimization of generated SQL queries
6. Ease of use

4.1.1 Setting and context

This study has used ***KenyaEMR***, a distribution of Open Medical Records System (OpenMRS) with over 350 active implementations in Kenya. OpenMRS is the most widely used open source EMR system in the sub-Saharan Africa. It was developed in Kenya in 2004 and has since been adopted worldwide to support delivery of care in both HIV and non-HIV programs. OpenMRS implements the EAV (Entity-Attribute-Value) data model which makes it suitable for handling sparse, heterogeneous and volatile clinical information (Akanbi et al., 2014; Mamlin, Biondich, Wolfe, & Fraser, 2006).

Clinical data in OpenMRS system is encoded using CIEL concept dictionary. The dictionary is comprehensive and provides all important attributes of a concept including id, UUID, names (short, long, preferred, locale preferred), description, data type (numeric, boolean, coded, text, drug and complex), possible questions that can be answered by the concept, and possible answers/values if the concept can be used as a question. Concepts in the CIEL dictionary have mappings to internationally recognized dictionaries like SNOMED and LOINC.

OpenMRS uses ***HTML form entry module*** to manage form metadata. The module processes metadata to render HTML forms to users for data entry. Form metadata comprises of standard HTML tags and additional custom OpenMRS specific tags used to render and record data about OpenMRS objects. Examples of custom tags include <encounterDate>, <encounterProvider>, <obs> and others. Figure shows metadata for triage form in KenyaEMR. The <**obs**> tag denotes a data point for recording a medical observation and it has an attribute **conceptId** whose value is a concept id/UUID in the CIEL concept dictionary.

```
<div class="ke-form-content">
    <fieldset>
        <legend>Reason</legend>

        <table>
            <tr>
                <td>Reason for visit</td>
                <td>
                    <obs id="rsn" conceptId="160430AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" rows="2" cols="80"/>
                </td>
            </tr>
        </table>
    </fieldset>

    <fieldset>
        <legend>Vital Signs</legend>

        <table>
            <tr>
                <td>Temp</td>
                <td>
                    <obs conceptId="5088AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"/>
                    &#176;C
                </td>
            </tr>
            <tr>
                <td>Pulse Rate</td>
                <td>
                    <obs conceptId="5087AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"/>
                </td>
            </tr>
            <tr>
                <td>BP</td>
                <td>
                    <obs conceptId="5085AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"/>
                    /
                    <obs conceptId="5086AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"/>
                    mmHg
                </td>
            </tr>
```

Figure 5 - Sample HTML form showing metadata for Triage form

4.2 Auto-ETL Methodology

The Auto-ETL methodology provides for the design and implementation of ETL solutions in an efficient and convenient way with very minimal and basic one-time specifications covering mainly metadata and the overall ETL process. The methodology uses mainly metadata of forms available in the system and the one-time specifications from ETL designers to automatically generate ETL rules and SQL statements that are executed to create a flat ***table-per-form*** (table for every form).

This methodology obtains all mapping requirements from form metadata and doesn't require any manual mappings by ETL designers. It, therefore, eliminates the manual mapping process between source and target datasets which is costly, slow, tedious, and prone to human errors. The table-per-form approach to the generation of flat tables, similar to ontologies defined by (Mate et al., 2015), provides a sure way of ensuring form-variable relationship (data semantics) is retained even in the generated tables since data captured by a particular form resides in the same table. Auto-ETL still provides ETL designers a chance to test, debug, and optimize generated rules and SQL statements as need be although this may not be a requirement.

4.2.1 Auto-ETL Components

Auto-ETL is based on five key components: form metadata, ETL specifications, rule generation, ETL engine, and testing and debugging. **Figure 6** shows Auto-ETL's process flow with additional description in

Table *1*.

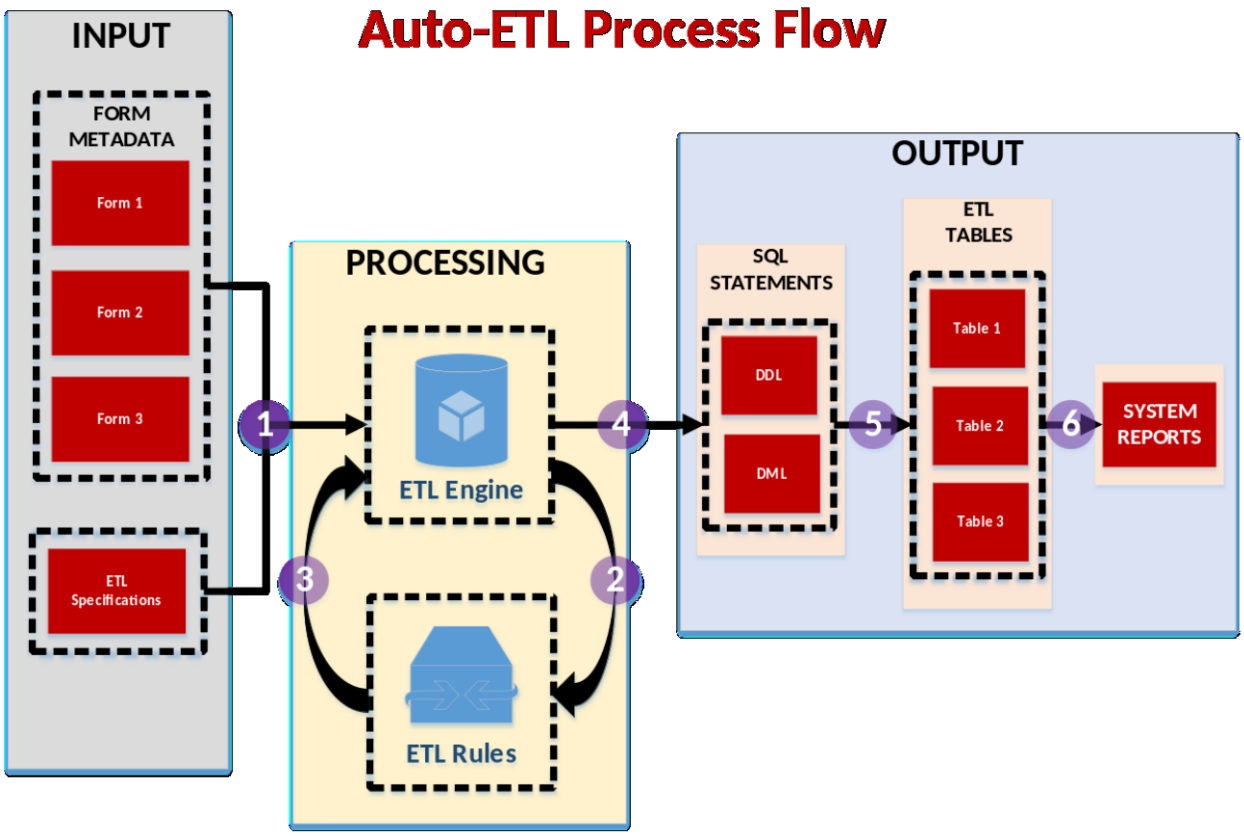


Figure 6 - Auto-ETL process flow

Table 1 - Description of Auto-ETL Process flow

Process No	Description
1	The ETL engine extracts form metadata and ETL specifications
2	The ETL engine uses information from form metadata and ETL specifications to generate rules for the ETL process. The generated rules are stored to allow verification and any modifications by ETL designers as need be.
3	The ETL engine extracts generated/stored ETL rules in readiness for SQL statements generation.
4	The ETL engine uses ETL rules to generate SQL statements to support the entire ETL process. These include DDL and DML statements.
5	The ETL engine executes the generated SQL statements to pivot data from EAV data model into flat tables.
6	Data extraction from flat tables to support system reports, data mining, and visualizations.

4.2.1.1 ETL Specifications

These are one-time configurations provided by ETL designers, similar to those described by (Ong et al., 2017), to support the ETL process. Auto-ETL specifications, however, does not require any specialized expertise since most configurations are general concepts. The specifications are any configurations that may vary from one implementation to another and may include the following.

1. Preferred name of the target database where all ETL tables will be created.
2. The scope of data to be processed in the form of a list of forms whose data needs pivoting.
3. ETL process cycle/interval.
4. Metadata descriptions including tags used for data points i.e. <obs> in KenyaEMR.

4.2.1.2 Form Metadata

This provides detailed description of a form's content and keywords linked to the content. The Auto-ETL supports metadata in the HTML format usually expressed in the form of meta-tags denoted by angle brackets. The meta-tags may consist of standard HTML tags i.e. <html>, <td>, and special custom tags local to a system and used for a specific purpose i.e. <firstName>, <lastName>, <obs>, etc. A form metadata therefore describes the general layout and variables collected by the form. The custom tags in a form metadata are used as **data points** for collecting specific data during data entry. **Table 2** shows examples of

special tags used in KenyaEMR form metadata.

Table 2 - Examples of custom tags in KenyaEMR

Tag	Description	Sample data collected
<encounterProvider>	Records information about a provider during patient-provider interactions	John Doe
<encounterDate>	Records date of a patient encounter	2018-11-10 10:50:00
<obs conceptId=123>	Records medical information during an encounter. ConceptId 123 points to the code for the specific observation i.e. height in cm.	180

The form metadata is processed and useful information relating to data points i.e. code/concept id, values' list, etc are extracted. Additional information about data points which are not explicitly defined in metadata can be found in the data dictionary.
Auto-ETL requires declaration of data point tags in the ETL specifications and this provides for the ability to handle generic forms with different tags for data points.

4.2.1.3 ETL rules

As already described by (Lin & Haug, 2006; Mate et al., 2015; Ong et al., 2017), rules provide mappings between source and target datasets which in this case refers to EAV data model and generated flat tables. At minimum, an ETL rule should define a source database, source table, generated table, source variable, transformation operations on data, and column attributes in the generated table.
Auto-ETL automates ETL rule generation for every data point defined in the form metadata. It uses information extracted from metadata, ETL specifications, and data dictionary to put together information about data points. **Code sample 1** shows a section of a clinical encounter form's metadata with data points for different variables.

Code Sample 1:

```
 <!-- provides for specified answers and with the specified labels -->
<obs conceptId="1234" labelText="Primary Diagnosis" answerConceptIds="123,7,912"
answerLabels="Malaria,Tuberculosis,Diabetes"/>
<!-- free text box (constrained to legal numbers as defined in the dictionary) -->
<obs conceptId="5497" labelText="Most recent CD4"/>
```

```
<!-- Text variable -->
<obs conceptId="345" labelText="Clinical Notes"/>

<!-- Date variable -->
<obs conceptId="456" labelText="Next Appointment"/>
```

Table 3 illustrates how Auto-ETL uses information from form metadata, ETL specifications and data dictionary to generate rules for data points in **code sample 1**.

Table 3 - Sample ETL Rule generated from metadata in KenyaEMR

Source database	Source Table	Target database	Variable Code	Variable data type	Target Table	Target Table column name	Target table column data type	Value list
openmrs	Obs	reporting_etl	1234	Coded	clinical_encounter	primary_diagnosis	INT	123, 7, 912
openmrs	Obs	reporting_etl	5497	Numeric	clinical_encounter	most_recent_cd4	DOUBLE	As defined in data dictionary
openmrs	Obs	reporting_etl	345	Text	clinical_encounter	clinical_notes	VARCHAR	open
openmrs	Obs	reporting_etl	456	Date	clinical_encounter	next_appointment	DATE	Valid date

Information about the source and the target databases are extracted from ETL specifications. The source table may be implicitly defined in the data point's tag or explicitly defined in the ETL specifications. In KenyaEMR, for instance, <obs> denotes obs table and therefore implicitly defines the source table. Information on variable data type is obtained from the data dictionary where data types for all codes/concepts are defined. The name of the target table and the generated column are derived from form name and variable name (as defined in metadata labelText or data dictionary) respectively. In addition, the data type for the generated column is derived from the variable data type as defined in the data dictionary while value list is extracted from provided list (answerConceptIds) if explicitly defined or obtained from the data dictionary and this wholly depends on the variable data type.

Auto-ETL applies standard transformation expressions when generating rules. This is informed by the ***table-per-form*** approach that does not require complex transformation operations for pivoting. Expressions commonly used by Auto-ETL include MAX, IF, MIN, and SUM.

4.2.1.4 ETL engine

The ETL engine is the core of Auto-ETL methodology. It automatically generates SQL statements for the creation of the target database and tables, data extraction from source tables, transformation, and loading into target database tables. The engine performs the following operations in the order specified in the Auto-ETL process flow.

1. Extraction of useful information from form metadata and ETL specifications. Additional information is obtained from the data dictionary.
2. Generation of ETL rules as described extensively in section 4.2.1.3.
3. Generation of SQL statements based on ETL rules. Using the ***table-per-form*** approach, every form has two sets of SQL statements: Data Definition Language (DDL) and Data Manipulation Language (DML).
    a. The DDL SQL statements are used to create models for the target database and all target tables. These are created using a set of DROP table <table name> IF EXISTS and CREATE table <table name> SQL constructs. Column data types are obtained from the resulting data type of the associated ETL rule as shown under "Target table column data type" in **Table 3.**

    b. The DML SQL statements are used for extraction, transformation, and loading of data from source to target datasets. The ETL engine uses INSERT INTO tableName <fieldList>...SELECT <Transformed fieldList>...FROM <tableList> SQL template to construct the SQL statements. Every form marked for pivoting has a DML query generated which when executed populates the corresponding flat table with data from source tables. These are also used to incrementally update target tables whenever there are changes in the source tables. Changes that trigger incremental updates include new records, updated and deleted (voided) records.
4. Creation of ETL database using generated SQL statements. The engine creates all flat tables in a separate database using the target database name as specified in the ETL specifications. This approach makes it possible to have dedicated transactional and reporting databases for system efficiency in terms of performance and data extraction speed.

4.2.1.5 Testing and debugging

Even though there exists code-level testing and debugging through unit tests, the ETL engine still stores the generated rules and SQL statements for review and enhancements by ETL designers if need be. This is important for buy-in and high trust level in the generated tables and data.

4.3 System Testing

4.3.1 Integration Testing

The prototype was deployed in KenyaEMR and operations/behavior of other modules including HTML form entry monitored. This was to ensure that the prototype does not affect normal operations of other modules and consequently of the whole system.

4.3.2 Functional Testing

ETL rules and SQL statements were reviewed and tested by a team of SQL developers to confirm their accuracy. Sample generated datasets were also reviewed by the same team and the results were positive.

4.3.3 Usability Testing

The datasets generated by Auto-ETL were used to design ad-hoc reports using KenyaEMR ad-hoc query tool. The main focus was to ensure that the generated datasets are easy to understand, navigate, and used for desired output.

4.4 Pilot Experiment

In order to compare Auto-ETL to an entirely manual process, we conducted a pilot experiment at our office using two (2) software developers who were at the point writing ETL SQL statements for PMTCT (Prevention of Mother to Child Transmission) program forms provided in KenyaEMR. We identified six forms with varied complexity levels for the experiment and timed how long it would take the developers, working at their normal rate (8 hours a day), to complete.

The manual process involved writing DDL and DML SQL statements, testing and debugging, and executing them to produce flat datasets. For the Auto-ETL, we specified the six forms, source and target databases in the ETL specifications and timed how long it would take to generate DDL and DML SQL statements for pivoting the identified forms. The generated SQL statements were reviewed by the developers for accuracy and executed to generate a database with flat datasets. The datasets were reviewed against the ETL specifications and metadata of identified forms.

An ad-hoc query tool (Kenya EMR Data Tool) was then used to test the datasets generated using the two approaches.

CHAPTER FIVE: RESULTS

5.1 Introduction

Evaluation and validation of an ETL process involves comparing values of selected data variables between original data model (EAV) and generated datasets from ETL process (Denney, Long, Armistead, Anderson, & Conway, 2016). This section discusses the results of a pilot experiment to compare Auto-ETL to a purely manual process. It also discusses results of analyzing data abstracted from original and generated datasets for accuracy and concordance.

For data analysis, EMR databases for two health facilities, Kapenguria District Hospital and Makongeni Health Center, were used. 14682 records from EAV and ETL datasets each were abstracted from Kapenguria District Hospital EMR database and 1000 records from Makongeni Health Center EMR database. The data variables included were weight, height, temperature, and respiratory rate. The records were then evaluated for concordance across the two data sources.

5.1.1 Pilot experiment results

While it took the Auto-ETL approximately 40 seconds to generate all queries, the manual process took 4 days to complete all queries. The team attributed the delay to the repetitive, tiresome, and complex process of mapping and the actual writing of queries. In situations where the values' list was not explicitly specified, the team had to check in the data dictionary to get the complete list. Other issues with the manual process included typos in table and column names, mismatch in data types resulting in rounding up/down of numeric values, and the overall tedious and repetitive process one has to perform.

On Auto-ETL, the team confirmed the accuracy of generated rules and SQL statements. It was however noted that for data points with multiple values, the methodology could provide for multiple columns with the same name. This was however not a show-stopper since such variables are not commonly used in reporting although still needs to be accurately handled by the methodology. A check on the generated tables confirmed accuracy of table structure and data contained in the tables. We also carried tests to show whether the generated tables are optimized for data extraction. We used the database

engine's EXPLAIN statement which provides information on how MYSQL executes SQL statements. The EXPLAIN statement can be used to show how simple or complex a SQL statement is. **Figure 7** and **Figure 8** shows the results of **EXPLAIN** statement used with EAV and ETL based SELECT queries for triage variables.

```
mysql>  explain select e.patient_id,
    ->  date(e.encounter_datetime) as visit_date,
    ->  max(if(o.concept_id=5088,o.value_numeric,null)) as temperature__c_,
    ->  max(if(o.concept_id=5089,o.value_numeric,null)) as weight__kg_,
    ->  max(if(o.concept_id=5090,o.value_numeric,null)) as height__cm_,
    ->  max(if(o.concept_id=5242,o.value_numeric,null)) as respiratory_rate,
    ->  max(if(o.concept_id=160430,o.value_text,null)) as reason_for_visit
    ->  from openmrs.encounter e inner join
    ->  ( select form_id from openmrs.form where uuid in('37f6bd8d-586a-4169-95fa-5781f987fe62')) f
    ->  on f.form_id=e.form_id
    ->  left outer join openmrs.obs o on o.encounter_id=e.encounter_id and o.concept_id in (5088,5089,5090,1427,5092,5242,160430,5085,5086,5087,1343)
    ->  where e.voided=0 group by e.patient_id, e.encounter_id, visit_date;
+----+-------------+-----------+-------+---------------------------------+----------------------+---------+----------------------+-------+----------------------------------------------+
| id | select_type | table     | type  | possible_keys                   | key                  | key_len | ref                  | rows  | Extra                                        |
+----+-------------+-----------+-------+---------------------------------+----------------------+---------+----------------------+-------+----------------------------------------------+
|  1 | PRIMARY     | <derived2> | system | NULL                           | NULL                 | NULL    | NULL                 |     1 | Using temporary; Using filesort              |
|  1 | PRIMARY     | e         | ref   | encounter_form                  | encounter_form       | 5       | const                | 29546 | Using where                                  |
|  1 | PRIMARY     | o         | ref   | obs_concept,encounter_observations | encounter_observations | 5    | openmrs.e.encounter_id |     5 | Using where                                  |
|  2 | DERIVED     | form      | const | form_uuid_index                 | form_uuid_index      | 114     | const                |     1 | Using index                                  |
+----+-------------+-----------+-------+---------------------------------+----------------------+---------+----------------------+-------+----------------------------------------------+
4 rows in set (0.00 sec)
```

Figure 7- Query efficiency in EAV data model

```
mysql> explain select patient_id, visit_date, temperature__c_, weight__kg_, height__cm_, respiratory_rate, reason_for_visit
    -> from reporting_etl.triage
    -> where patient_id=27
    -> order by patient_id, visit_date;
+----+-------------+--------+------+---------------+------+---------+------+-------+-----------------------------+
| id | select_type | table  | type | possible_keys | key  | key_len | ref  | rows  | Extra                       |
+----+-------------+--------+------+---------------+------+---------+------+-------+-----------------------------+
|  1 | SIMPLE      | triage | ALL  | NULL          | NULL | NULL    | NULL | 15054 | Using where; Using filesort |
+----+-------------+--------+------+---------------+------+---------+------+-------+-----------------------------+
1 row in set (0.01 sec)
```

Figure 8- Query efficiency in generated flat table

The results show a simplified query plan for the ETL based queries after Auto-ETL. This is demonstrated by the halved number of rows to be processed and the "SIMPLE" select type as shown in the EXPLAIN results. A simplified execution plan denotes faster processing of data from ETL based datasets.

Table 4 - List of generated tables with information on data points and rows inserted

Form Name	Generated table name	No of data points	No of rows inserted	Time taken to populate flat table
Triage	triage	11	14681	2.41 sec
Clinical Encounter	clinical_encounter	19	9187	0.74 sec
HIV Enrollment	hiv_enrollment	30	4111	0.44 sec

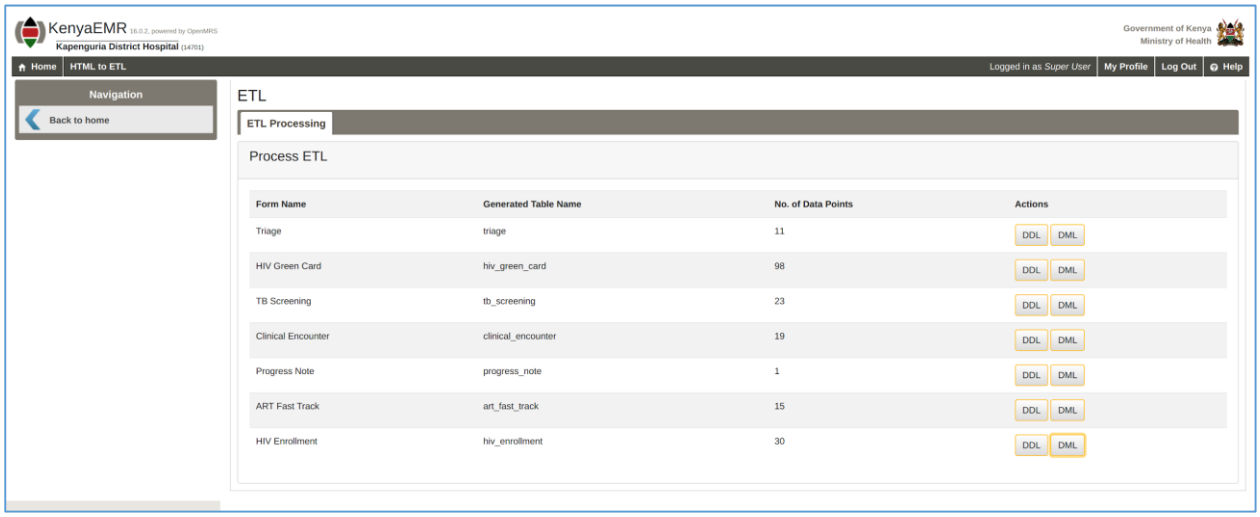


Figure 9 - Screenshot showing details of generated datasets in Kenya EMR

```
mysql> select e.patient_id,
    ->  date(e.encounter_datetime) as visit_date,
    ->  max(if(o.concept_id=5088,o.value_numeric,null)) as temperature__c_,
    ->  max(if(o.concept_id=5089,o.value_numeric,null)) as weight__kg_,
    ->  max(if(o.concept_id=5090,o.value_numeric,null)) as height__cm_,
    ->  max(if(o.concept_id=5242,o.value_numeric,null)) as respiratory_rate,
    ->  max(if(o.concept_id=160430,o.value_text,null)) as reason_for_visit
    ->  from encounter e inner join
    ->  ( select form_id from form where uuid in('37f6bd8d-586a-4169-95fa-5781f987fe62')) f
    ->  on f.form_id=e.form_id
    ->  left outer join obs o on o.encounter_id=e.encounter_id and o.concept_id in (5088,5089,5090,1427,5092,5242,160430,5085,5086,5087,1343)
    ->  where e.voided=0 and e.patient_id=27 group by e.patient_id, e.encounter_id, visit_date;
+------------+------------+----------------+------------+------------+-----------------+-----------------+
| patient_id | visit_date | temperature__c_ | weight__kg_ | height__cm_ | respiratory_rate | reason_for_visit |
+------------+------------+----------------+------------+------------+-----------------+-----------------+
|         27 | 2013-02-22 |           NULL |         40 |        158 |            NULL | NULL            |
|         27 | 2013-05-15 |           NULL |         44 |        158 |            NULL | NULL            |
|         27 | 2016-07-27 |           36.8 |       46.7 |        158 |            NULL | revisit         |
|         27 | 2016-11-02 |           36.3 |       47.5 |        158 |              22 | revisit         |
|         27 | 2017-01-25 |           36.3 |       44.2 |        158 |              18 | NULL            |
|         27 | 2017-04-19 |             37 |       43.3 |        158 |              18 | NULL            |
|         27 | 2017-07-26 |             36 |       42.8 |        158 |              18 | NULL            |
|         27 | 2018-03-07 |             36 |         43 |        158 |              18 | NULL            |
|         27 | 2018-06-27 |           37.5 |       49.3 |        158 |              18 | NULL            |
+------------+------------+----------------+------------+------------+-----------------+-----------------+
9 rows in set (0.00 sec)
```

Figure 10 - Sample query for extracting triage data from EAV data model in Kenya EMR

```
mysql> select patient_id, visit_date, temperature__c_, weight__kg_, height__cm_, respiratory_rate, reason_for_visit
    -> from reporting_etl.triage
    -> where patient_id=27
    -> group by patient_id, visit_date;
+------------+------------+-----------------+-------------+-------------+------------------+------------------+
| patient_id | visit_date | temperature__c_ | weight__kg_ | height__cm_ | respiratory_rate | reason_for_visit |
+------------+------------+-----------------+-------------+-------------+------------------+------------------+
|         27 | 2013-02-22 |            NULL |          40 |         158 |             NULL | NULL             |
|         27 | 2013-05-15 |            NULL |          44 |         158 |             NULL | NULL             |
|         27 | 2016-07-27 |            36.8 |        46.7 |         158 |             NULL | revisit          |
|         27 | 2016-11-02 |            36.3 |        47.5 |         158 |               22 | revisit          |
|         27 | 2017-01-25 |            36.3 |        44.2 |         158 |               18 | NULL             |
|         27 | 2017-04-19 |              37 |        43.3 |         158 |               18 | NULL             |
|         27 | 2017-07-26 |              36 |        42.8 |         158 |               18 | NULL             |
|         27 | 2018-03-07 |              36 |          43 |         158 |               18 | NULL             |
|         27 | 2018-06-27 |            37.5 |        49.3 |         158 |               18 | NULL             |
+------------+------------+-----------------+-------------+-------------+------------------+------------------+
9 rows in set (0.02 sec)
```

Figure 11 - Sample query for extracting triage data from generated flat table

5.1.2 Descriptive characteristics using median

From analysis, the descriptive characteristics of the source and generated tables in the two facility databases were of similar nature as shown in **Table 5**. This results indicate zero data loss with Auto-ETL methodology.

Table 5 - Median values for triage variables

	Kapenguria median (min, max)		Makongeni median (min, max)	
Variable	**ETL**	**EAV**	**ETL**	**EAV**
Temperature	36.2 (27, 40.4)	36.2 (27, 40.4)	36.2 (36, 36.8)	36.2 (36, 36.8)
Weight	56 (1.9, 167)	56 (1.9, 167)	61.1 (4.6, 122)	61.1 (4.6, 122)
Height	163 (10.5, 199)	163 (10.5, 199)	164 (16.7, 187)	164 (16.7, 187)
Respiratory rate	18 (1, 1895)	18 (1, 1895)	18 (12, 22)	18 (12, 22)

The graphical distribution of the extracted variables is as shown in Figure and Figure **.**

i. Height

Figure 12 - Distribution of Height in ETL and EAV data models

ii. Weight

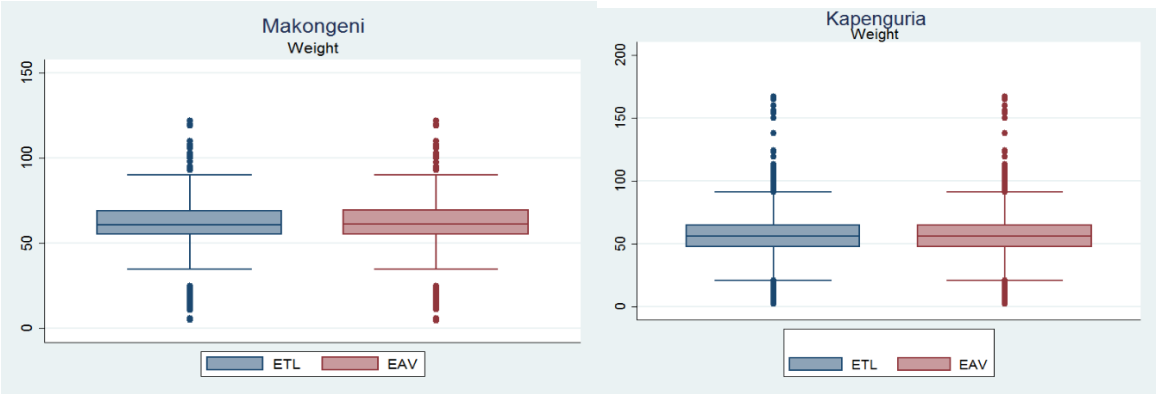


Figure 13 - Distribution of Weight in ETL and EAV data models

5.1.4 Concordance correlation

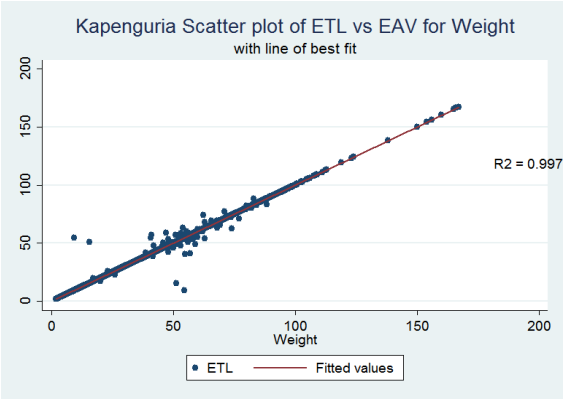
Concordance correlation was performed to measure the agreement between variables from the source and generated tables. All the variables had a strong correlation closer to 1 and they were statistically significant. The results are as shown in table 6.

Table 6 - Concordance Correlation values

Variable	Kapenguria			Makongeni		
	Concordance correlation	95% CI	*p* value	Concordance correlation	95% CI	*p* value
Temperature	0.995	0.995-0.996	<0.001	1	1.0 - 1.0	<0.001
Weight	0.999	0.999-0.999	<0.001	1	1.0 - 1.0	<0.001
Height	0.998	0.998-0.998	<0.001	1	1.0 - 1.0	<0.001
Respiratory rate	1	1.00- 1.00	<0.001	1	1.0-1.0	<0.001

Scatter plots with regression line were used to show the relationship between the variables. There was a good relationship between the variables as an increase in the EAV variables had a corresponding increase in the ETL variables and all their R squared values were closer to 1. Figure and Figure shows the graphical representation of the results.

i. Weight in Kg



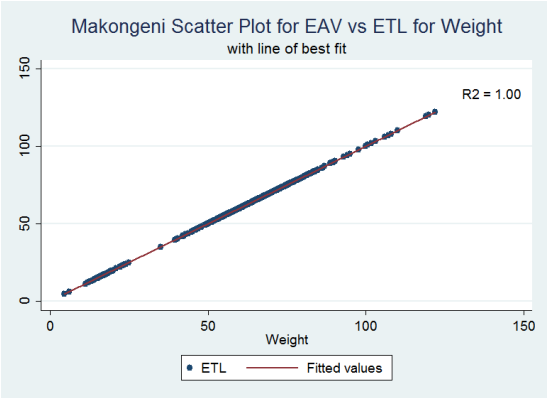
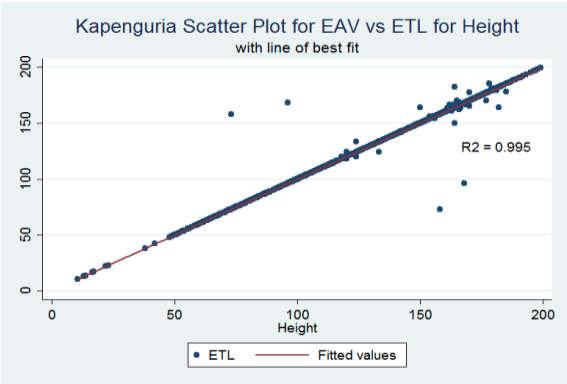


Figure 14 - R squared value for Temperature in ETL and EAV data models

ii. Height in cm



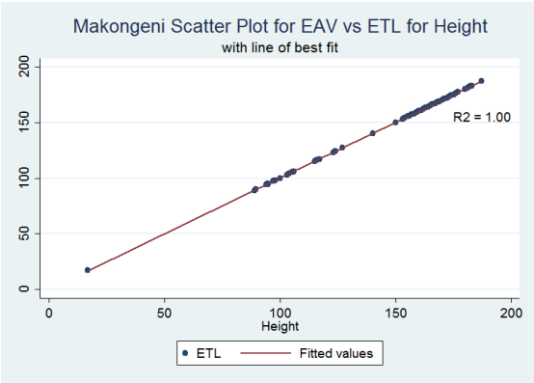


Figure 15 - R squared value for Weight in ETL and EAV data models

CHAPTER SIX: CONCLUSION

6.1 Achievements

Systems with EAV data model can take advantage of ETL methodologies to help generate flat tables for reporting. This study was setup to design an ETL methodology that addresses existing limitations of those discussed in existing research and this was achieved through the following objectives.

Objective No 1: To review existing ETL methodologies for EAV data model and identify limitations

We were able to review four ETL methodologies all of which described an ETL process to have three phases: generation of ETL specifications; ETL rules generation; and generation of SQL statements to support extract, transform, and load functions of the process. Whereas the first phase is basic and is not labor intensive and third phase fully automated, generation of ETL rules requires extensive expert knowledge and time for perfect mapping of source and target data sources. In three out of the four methodologies, generation of ETL rules is manual although with graphical interface to guide the process. It is in one methodology where rule generation is semi-automated. Other limitations include loss of data semantics in creating of flat tables.

Objective No 2: To design and develop a suitable ETL methodology that addresses the limitations of existing methodologies

We borrowed a number of concepts from the reviewed methodologies and these include use of metadata, generation of rules and use of logical groupings to preserve data semantics. In order to reduce need for experts in ETL design and development process, a suitable methodology, Auto-ETL, was designed to use form metadata for creation of rules for the ETL process. Use of form metadata also helps preserve data semantics even in the generated flat tables.

Objective No 3: To evaluate and validate the new methodology for accuracy and efficiency

After a successful ETL process through Auto-ETL, data was abstracted from EAV and ETL data datasets for analysis. The results recorded same median across the two data sources, p-value of <0.001 for R squared and CI value of 0.997 all of which indicate a strong

concordance and zero data loss between the two data sources. Implementation of Auto-ETL in our office took less than a minute to generate ETL SQL statements for six forms. The same task took 4 days to be accomplished by two SQL programmers working for 8 hours a day. All SQL statements generated by Auto-ETL were confirmed to be accurate except for multi-select variables which require enhancements in the methodology for proper handling.

Through this work, we can therefore conclude that form metadata can be used to generate SQL statements to support ETL process thus automating the manual phase in the process that is very slow and costly.

6.2 Recommendations

1. Enhance the prototype for use by OpenMRS community.
2. Apply Auto-ETL methodology to other systems using EAV data model i.e. DHIS2 for automated flat table generation.

6.3 Future Work

Auto-ETL methodology has worked well for forms with simple metadata. Certain areas, however, still needs to be addressed and they include the following:

1. Enhance Auto-ETL to process forms in a work-flow. The current methodology will generate as many tables as the forms although a work-flow should have just a single flat table with data for all forms.
2. Provide processors for other common metadata formats i.e. XML and JSON.
3. Enhance the methodology to process nested tags i.e. a parent tag with multiple nested child nodes.
4. Integrate use of form metadata and descriptive characteristics of data as described in (Lin & Haug, 2006) . This will help address systems with no form metadata.

REFERENCES

Akanbi, M. O., Ocheke, A. N., Agaba, P. A., Daniyam, C. A., Agaba, E. I., Okeke, E. N., & Ukoli, C. O. (2014). Use of Electronic Health Records in sub-Saharan Africa: Progress and challenges. *Journal of Medicine in the Tropics*, *14*(1), 1–6. Retrieved from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4167769/

Anhoj, J. (2003). Generic design of Web-based clinical databases. *Journal of Medical Internet Research*, *5*(4), e27. https://doi.org/10.2196/jmir.5.4.e27

Chen, R. S., Nadkarni, P. M., Marenco, L., Levin, F., Erdos, J., & Miller, P. L. (2000). Exploring performance issues for a clinical database organized using an entity-attribute-value representation. *Journal of the American Medical Informatics Association*, *7*(5), 475–487. https://doi.org/10.1136/jamia.2000.0070475

Corwin, J., Silberschatz, A., Miller, P. L., & Marenco, L. (2007). Dynamic Tables: An Architecture for Managing Evolving, Heterogeneous Biomedical Data in Relational Database Management Systems. *Journal of the American Medical Informatics Association : JAMIA*, *14*(1), 86–93. https://doi.org/10.1197/jamia.M2189

Denney, M. J., Long, D. M., Armistead, M. G., Anderson, J. L., & Conway, B. N. (2016). Validating the extract, transform, load process used to populate a large clinical research database. *International Journal of Medical Informatics*. https://doi.org/10.1016/j.ijmedinf.2016.07.009

Dinu, V., & Nadkarni, P. (2007). Guidelines for the effective use of entity-attribute-value modeling for biomedical databases. *International Journal of Medical Informatics*, *76*(11–12), 769–779. https://doi.org/10.1016/j.ijmedinf.2006.09.023

Dinu, V., Nadkarni, P., & Brandt, C. (2006). Pivoting approaches for bulk extraction of Entity–Attribute–Value data. *Computer Methods and Programs in Biomedicine*, *82*(1), 38–43. https://doi.org/http://dx.doi.org/10.1016/j.cmpb.2006.02.001

Eessaar, E., & Soobik, M. (2008). On Universal Database Design. In *Proceedings of the Eighth International Baltic Conference onDatabases and Information Systems* (pp. 349–360). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.452.5478&rep=rep1&type=pdf

Fraser, H. S. F., Blaya, J., Choi, S. S., Bonilla, C., & Jazayeri, D. (2006). Evaluating the impact and costs of deploying an electronic medical record system to support TB treatment in Peru. *AMIA ... Annual Symposium Proceedings / AMIA Symposium. AMIA Symposium*, *2006*, 264–268. https://doi.org/PMC1839453

Jawhari, B., Ludwick, D., Keenan, L., Zakus, D., & Hayward, R. (2016). Benefits and challenges of EMR implementations in low resource settings: a state-of-the-art review. *BMC Medical Informatics and Decision Making*, *16*(1), 116. https://doi.org/10.1186/s12911-016-0354-8

Lin, J.-H., & Haug, P. J. (2006). Data preparation framework for preprocessing clinical data in data mining. *AMIA ... Annual Symposium Proceedings. AMIA Symposium*, *2006*, 489–493. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/17238389%0Ahttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1839316

Löper, D., Klettke, M., Bruder, I., & Heuer, A. (2013). Enabling flexible integration of healthcare information using the entity-attribute-value storage model. *Health Information Science and Systems*, *1*, 9. https://doi.org/10.1186/2047-2501-1-9

Luo, G., & Frey, L. J. (2016). Efficient Execution Methods of Pivoting for Bulk Extraction of Entity-Attribute-Value-Modeled Data. *IEEE Journal of Biomedical and Health Informatics*. https://doi.org/10.1109/JBHI.2015.2392553

Mamlin, B. W., Biondich, P. G., Wolfe, B. A., & Fraser, H. S. F. (2006). Cooking up an open source EMR for developing countries: OpenMRS – A recipe for successful collaboration BT - Proc AMIA Symp.

Mate, S., Köpcke, F., Toddenroth, D., Martin, M., Prokosch, H.-U., Bürkle, T., & Ganslandt, T. (2015). Ontology-Based Data Integration between Clinical and Research Systems. *PLOS ONE*, *10*(1), e0116656. Retrieved from http://dx.doi.org/10.1371%2Fjournal.pone.0116656

Ministry of Health. (2013). *MINISTERIAL STRATEGIC & INVESTMENT PLAN JULY 2014– JUNE 2018*. Retrieved from http://www.health.go.ke/wp-content/uploads/2016/03/MINISTERIAL-STRATEGIC-INVESTMENT-PLAN.pdf

Moturi, C., Mburu, R., & Ngaruiya, N. (2016). A Case for Judicial Data Warehousing and Data Mining in Kenya, *4*(1), 7–14. https://doi.org/10.12691/ajcrr-4-1-2

Nadkarni, P. M., & Marenco, L. (2001). Easing the transition between attribute-value databases and conventional databases for scientific data. *Proceedings / AMIA ... Annual Symposium. AMIA Symposium*.

Ong, T. C., Kahn, M. G., Kwan, B. M., Yamashita, T., Brandt, E., Hosokawa, P., … Schilling, L. M. (2017). Dynamic-ETL: A hybrid approach for health data extraction, transformation and loading. *BMC Medical Informatics and Decision Making*, *17*(1). https://doi.org/10.1186/s12911-017-0532-3

Paul, R., & Hoque, A. S. M. L. (2011). Optimized Entity Attribute Value Model: A Search Efficient Representation of High Dimensional and Sparse Data. *Interdisciplinary Bio Central*, *3*(3), 1–5. https://doi.org/10.4051/ibc.2011.3.3.0009

Van Velthoven, M. H., Mastellos, N., Majeed, A., O'Donoghue, J., & Car, J. (2016). Feasibility of extracting data from electronic medical records for research: An international comparative study 90. *BMC Medical Informatics and Decision Making*, *16*(1), 1–10. https://doi.org/10.1186/s12911-016-0332-1

World Health Organization. (2007). *TOWARDS UNIVERSAL ACCESS Scaling up priority HIV/AIDS interventions in the health sector*. Retrieved from www.who.int/hivISBN9789241595391

World Health Organization (WHO). (2007). *Everybody business: Strengthenig Health Systems to Improve Health Outcomes*. https://doi.org/10 July 2012