Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Parsing AQL Queries into SQL Queries using ANTLR

by

Purani Mounagurusamy

LIU-IDA/LITH-EX-A--15/067--SE

2015-11-13

Linköpings universitet

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Linköping University
Department of Computer and Information Science

Final Thesis

Parsing AQL Queries into SQL Queries using ANTLR

by

Purani Mounagurusamy

LIU-IDA/LITH-EX-A--15/067--SE

2015-11-13

Supervisor: Fang Wei-Kleiner
Department of Computer and Information Science (IDA)

Examiner: Patrick Lambrix
Department of Computer and Information Science (IDA)

Abstract

An *Electronic Health Record* is a collection of each patient's health information which is stored electronically or in digital format. *openEHR* is an open standard specification for electronic health record data. openEHR has a method for querying a set of clinical data using the Archetype Query Language (AQL).

The EHR data is in XML format and this format is a tree like structure. Since XML databases were considerably slower, AQL needs to be translated to another query language. Researchers have already investigated translating AQL to XQuery and tested the performance. Since the performance was not satisfactory, we now investigate translating AQL to SQL.

AQL queries are translated to SQL queries using the ANTLR tool. The translation is implemented in Java language. The AQL queries which are translated into SQL queries are also tested in this thesis work. Finally, the result is to get the corresponding SQL query for any given AQL query.

Acknowledgement

I hereby acknowledge my sincere thanks to my examiner **Patrick Lambrix** and my supervisor **Fang Wei-Kleiner** in the Department of Computer and Information Science (IDA) who has given me the opportunity to work on this thesis by believing in my capabilities and keeping track of my progress every week and encouraging me to complete my thesis. I am very happy to be a part of this project which has been provided me with the opportunity to learn about a new language. I have to thank IDA administration for the technical support from the server helpdesk **Rikard Nordin** and finally I would like to thank my friends and family for their support and encouragement.

Table of contents

Chapter 1

Introduction

In this chapter, the motivation of doing this thesis work is explained with the project goals and problem statement. Finally, the outline of the thesis work is described.

1.1 MOTIVATION

An Electronic Health Record (EHR) is a collection of a patient's health information collected in an electronic or computerized format. Since medical care has become more complex, doctors felt that they were not able to access all the patient's complete health history on paper format. Then EHR has been first started in 1960s for improving the patient medical care [14, 15]. An EHR system is used to improve the quality, accuracy and efficiency of data recorded in a health record [17]. Such systems are used in various countries such as Austria, Belgium, Denmark, Czech Republic, Estonia, Finland, France, Germany, Ireland, Italy, Sweden, Switzerland, United Kingdom, etc [16, 17].

openEHR is an open standard specification that deals with EHR data. A set of specifications has been published and maintained by the openEHR foundation. It is an international and not for profit foundation company that supports research, development and implementation of openEHR EHRs. It is also an online community whose mission is to promote and facilitate progress towards EHRs in high quality, to support the needs of patients and clinicians everywhere [20]. The openEHR specifications deal with health information Reference Model (RM), a language for building Clinical Models or Archetypes which is the separation of Software and Query Language [2]. The architecture is designed to make use of external terminologies.

Most of the information systems were built using Single-Level Modelling Approach in which both the information and knowledge concepts were built into one level of object and data models. Since it is too complex to maintain and there is a constant change in the evolving concepts for the development of EHR, Multi level Modelling Approach is designed (which is the separation between domain model and reference model) in order to build future-proof systems. Here, the systems can be built more quickly from information model only and driven at runtime by domain knowledge environment [25].

The first level in the *Two-Level Modelling Approach [25]* is the stable Reference Model – the level of software object models and database schemas used to build information systems. It must be small in size and contain only non-volatile concepts (i.e classes).
The second level is the Domain Model – the level that requires its own formalisms and structures dealing with more volatile concepts of most domains. These are in the form of Archetypes and Templates.

The Reference Model (RM) or Information Model will take care of software and technical concerns dealing with information structure and data types using small set of information model classes. Example: "role", "act", "entity". The Reference Model has some types such as Composition, Section, etc.

Most formal definitions of clinical contents are in the form of two types: Archetypes and Templates. Archetypes or domain models are designed, developed and maintained by domain experts/clinical professionals. These models are used to record and capture clinical information (for example: blood pressure) of a patient. It defines the data objects through Reference Model. These data objects separate from archetypes or vice versa and so archetype has their own repository. Archetypes are included in those repository (open source) which can be viewed or used from online archetype libraries. The usage of archetypes are more specific during runtime which is used for a particular purpose (for example: data capture and validation) through templates. Archetypes are themselves represented as an instance of archetype model that defines a language to write archetypes. In order to express or to represent the openEHR archetypes in textual format, the native language or abstract language called Archetype Definition Language (ADL) is used. This language is based on Frame logic queries or F-logic with the addition of terminology. These are ISO standard and formalized by Archetype Object Model (AOM). AOM is an object model which defines the semantics of the archetypes. Each archetype includes set of constraint rules on the reference model which defines a subset of instances that are considered to conform to the subject of the archetype. Example: "Laboratory Result", "Blood Pressure", "Lab Test".

Archetypes are grouped into Templates which is an another openEHR specific concept. Data sets are defined by openEHR template. openEHR templates are more detailed

specifications that represent implementable data sets such as documents, forms, clinical notes, messages and screens that are required within and between EHRs. It is a special kind of archetypes that combine a tree of one or more archetypes, each constraining instances of various reference model types such as Composition, Section, etc. It results to form a set of data specific to a particular use. Eg: Screen form. In a simple way, we can say that templates combine one or more archetypes and add further constraints that are required for the use of those archetypes to a particular setting [24].

EHRs must have a query interface that provides rich overview of data and query mechanism. A query interface is needed that will support users at varying levels of query skills. Queries are based on the information model and the content models. Querying requires a query language which is essential for health information systems and there is a query language in openEHR. AQL is a query language that has been developed to query upon EHRs. It was first called EQL (EHR Query Language). AQL is difficult for semi skilled users. It requires knowledge of archetypes and knowledge of languages such as XML and SQL. The AQL syntax is similar to the SQL syntax that has SELECT, FROM and WHERE clauses but instead of using fields in a table as in SQL, AQL makes use of archetype paths [3, 21, 22, 29].

AQL is used to retrieve the EHR data. For optimization purpose, we first need to translate this clinically targeted AQL queries into other query languages such as SQL or XQuery [18]. Considering this, there were two works that has been processed related to this issue previously. The two previous solutions are as follows:

- The EHR data is in XML format. Since XML databases were considerably slower and require more space, they first translated AQL queries to another query language i.e. XQuery and then run the query. The translation was implemented without an index structure. Its performance was not satisfactory [1].

- Then an index is formed based on the paths and dewey coding. So the data parsing has been done and now we have dewey coding and index structures based on XML tree. Using Java programming in Apache Hadoop, the same task was done and implemented to write the query instead of SQL queries. Apache Hadoop is used to deal with large datasets. It is a distributed processing of large datasets across

clusters of computers [38]. It is an open source implementation of MapReduce framework proposed by Google [39]. So the implementation was based on programming Hadoop data instead of tables. But the performance was not satisfactory in this case either.

1.2 PROBLEM STATEMENT

If the data size or the number of records is small (say 40,000) which is passed as input to the previous work, then time taken is reasonable. But if we use larger datasets (for example: about 4 million records), then time taken is very long. Basically the queries are in the form similar to XPath so, if we implement based on XPath but using Java, then time taken to execute the query is quite long. As this work does not result in a better performance, these AQL queries are transformed into SQL queries using automatic parser generator.

1.3 PROJECT ANALYSIS AND THESIS GOAL

The larger project is based on two steps:

1) There are many XML files and each record in the XML file is in the form of parent-child tree. For the given tree structure we have to build an index based on the paths and dewey coding. This has been done already in a previous project. Now, we have to transform this tree into dynamic cluster tables. The task is to implement SQL in sort-merge join to store the data or tables in a cluster. Sort-merge join approach or algorithm is used to join the EHR data. The algorithm will sort the datasets and make joins according to the hadoop process. Since the epidemiological data is large say about 14 Gigabytes, we are implementing this to have a better performance. So the aim is to store the index data into a cluster.

2) The next step is to transform this AQL query into SQL query. So the query of AQL over the epidemiological data will be transformed into SQL query.

For a complete solution, both the steps have to me merged to get the final result. In this thesis we only focus on the second step in which AQL queries are transformed into SQL queries using ANTLR parser generator. The first step is being developed in parallel with this thesis work.

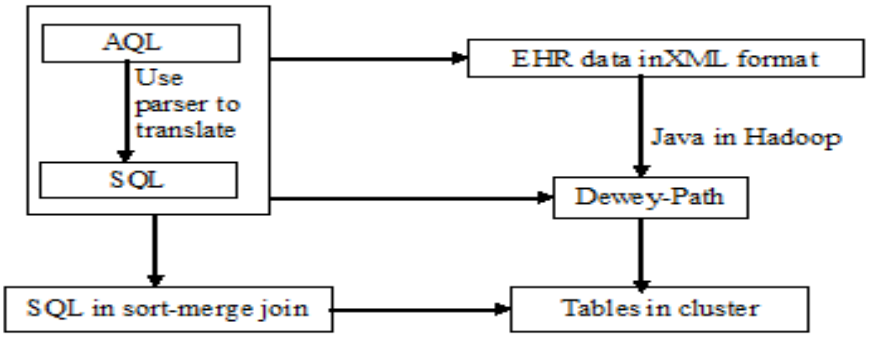


Fig: 3.1. A complete system

1.4 METHODS

There are many types of methodologies in software development process. Some of them are Waterfall, Spiral, XP or Extreme Programming, Agile, Scrum, Rapid Application Development, etc. This thesis work is the continuous development and programming based on continuous analysis and design in which we can say an iterative set of processes. In this iterative approach, the development involves several incremental steps or cycles each including analysis, design and programming. These steps are repeated for several times. This thesis work or analysis is based on the continuous approach from the previous results. The basic activities or steps involved in software development process are requirements, analysis, design, implementation, testing and maintenance. Therefore, we can say that it might be related to waterfall model, v-model or Agile model. Waterfall model is the sequential way of processing those activities step by step. Agile model is the way of user involvement by delivering the improvement or progress of the work weekly once. V-model is similar to waterfall model but in addition it includes testing during each phase and also it might include continuous improvement process. However, the analysis, design, implementation and testing has been done during the work.

Requirement Specification

The first step to start this thesis is based on the study of EHR and openEHR, AQL queries and ANTLR (Java parser generator) tool. The most important knowledge here is SQL queries. This is really important part of the thesis as it gives the clear idea how it can be achieved using automatic parser generator. The XML paths and values are retrieved using indexing and stored in separate files. These are used when querying the epidemiological data.

Analysis

The method of this thesis work was based on the analysis of National Cervical Cancer Information System in order to identify the records that belong to the same patient. Due to this, a unique id, i.e. uid or dewey id was given for each patient [1]. EQL was also developed based on the analysis of set of clinical query scenarios including the study of currently available query languages like XQuery, SQL and the study of archetypes, openEHR RM and openEHR path mechanisms [21].

Design

The whole designing of the system is based on the analysis of two steps:

1. Algorithm to join EHR data i.e. Use sort-merge join to join EHR data using hadoop
2. Translating AQL queries to SQL queries

However, the second part of the work, which is the focus of this thesis, is divided as follows:

- Get the AQL grammar and parse it to ANTLR
- Translate AQL queries to SQL queries

Implementation

For all the AQL queries, the AQL paths and values have been found from the data and keeping those path or pathvalue as a table, AQL queries are translated into SQL queries manually. Once it has been done manually, implement those using AST and translate it to SQL queries using Java language.

Testing

The testing that has been followed here is unit test. It has been tested for each section of the code step by step. Once the first section is implemented and tested, the next section is carried out.

1.5 INTENDED READERS

The intended readers of this thesis work are people with a basic knowledge in Databases and Compilers.

1.6 THESIS OUTLINE

The upcoming chapters are organized as follows:

Chapter 2 → Background. EHR, openEHR, AQL and the ANTLR tool are explained.

Chapter 3 → A detailed approach for converting SQL language from AQL language is described and some examples are given.

Chapter 4 → Implementation and testing are shown.

Chapter 5 → Concludes the thesis work and possible future work is explained.

Chapter 2

Background Study

In this chapter, a detailed study of EHR, openEHR, AQL and ANTLR are described.

2.1 EHR

An Electronic Health Record (EHR) is a collection of a patient's health information collected in a record. It is very simple and digital or computerized versions of patients paper charts (or Computerized Patient Record – CPR). It gives a detailed information about an individual patient. So the information is available instantly, whenever and wherever it is needed and they bring together in one place everything about a patient's health. The information in EHR is typically entered and accessed by health care providers. The main purpose of patient's record is to help each individual patient and improve their health care but today it is used for many purposes such as for research, etc. Especially without computers it is very hard to get all the information about a patient.

The first EHRs began to appear in the 1960s. In 1965, atleast 73 hospitals, clinical information projects, 28 projects for storage and retrieval of medical documents and other clinically relevant information were undergoing. The idea of computerized medical records has been around as one of the key research in medical informatics for more than 20 years [7]. The uses of EHRs are to support healthcare, clinical epidemiological studies, decision support systems and healthcare services management [1].

2.2 openEHR and its Approach

openEHR specifications have been developed in order to standardize the representation of EHR. It is an open standard specification. This specification deals with how EHR data are managed, stored and retrieved [7]. Also, it helps in the development towards a computerized medical record that follows a patient in his/her lifetime.

2.2.1 openEHR specification project

This project deals with many kinds of specifications. openEHR architectural specifications are composed of Reference Model (RM), Archetype Model (AM) and Service Model (SM) [8]. The Reference Model and Service Model correspond to ISO Refrence Model for Open

Distributed Processing (ISO RM/ODP) information and computational viewports repectively [26]. This RM/ODP introduces the concept of viewport to describe a system from particular set of concerns and deals with the complexity of distributed systems. These specifications are defined as a set of abstract models using UML notation and formal textual class specifications [28]. The Reference Model represents the semantics to store and process the information in systems. It contains a set of generic data structures to model the most. These data structures are decomposed into compounds or elements and these compounds can be further decomposed into compounds or elements. Archetype Model (AM) contains the knowledge enabling environment by defining domain level structure and constraints on the generic data structures which is in RM. The medium in which these constraints are delivered is said to be Archetypes [27].

2.2.2 Multi-Level Modelling Approach

In single-level modelling approach, both the information and knowledge are built into one level of object and data models. The information systems are developed in such a way that the domain concepts of the system has to be fully hard coded directly into software and database models. Systems based on such models are very expensive to maintain, exists contant changes and it has to be replaced. To avoid all these problems, multi-level modelling approach is needed which is intended to improve the semantic interoperability and reuse [25].

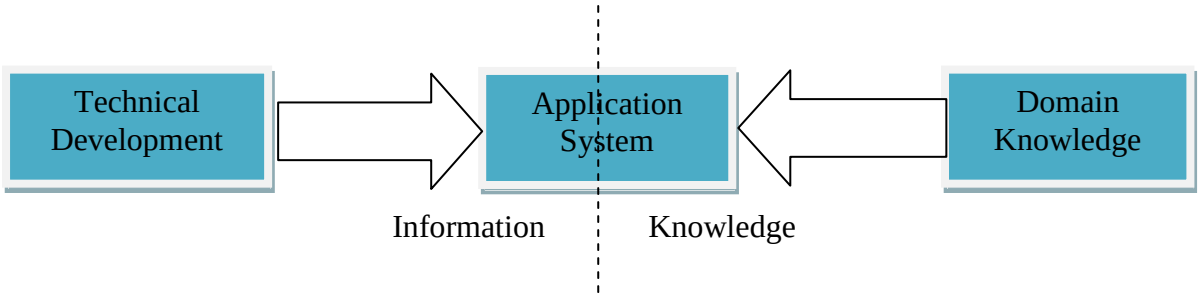


Fig: 2.2.2. [13] Multi-Level Modelling Approach

The multi-level modelling approach of clinical information deals with the separation of knowledge/domain model and information model/Reference Model in order to overcome the problems caused by the ever-changing nature of clinical knowledge. It is designed in order to build future-proof systems and it uses a stable Reference Model that can be

implemented in software, and flexible domain model expressed in Archetypes and Templates.

openEHR work contains two activity areas and they are technical and clinical ones. The technical area is where engineering work is performed such as specification development, implementation, testing, etc. The clinical area is where organizations and individuals that compose the health sector provide their knowledge by developing ontologies, archetypes, templates as well as by enabling for clinical training. These two activities are namely the two levels indicated in the two level modelling approach.

- **Information model**

  It is built as a stable Reference Model (RM) which allows for future proof information systems. This can be implemented in software and a flexible domain model expressed in archetypes and templates. These archetypes and templates are used for data validation and sharing. The classes of RM model will be persisted and tends to be stable. This means that its classes are intended not to change frequently [1].

- **Clinical model**

  The conceptual clinical information is represented via restricted formal structures called archetypes. Clinical contents can be specified in terms of two types: Archetypes and Templates.

2.2.3 Archetypes

openEHR archetypes are based on openEHR Reference (Information) Model. It gives the semantic meaning to the objects that are persisted via RM. The proposal of openEHR is that the structural changes and business rules are reflected in archetypes compared to RM. So there is no need to make changes in the persistence mechanism. The archetype is a domain content model in the form of structured constraint statements based on RM. These are formal specifications which is used for creating data structures and validate real data input. The creation and edition of archetypes are done primarily by domain experts. In general, they are defined for wide reuse and structurecd into hierarchies. They accommodate any number of natural languages and terminologies [7].

For example (Fig: 2.2.2), an archetype for “systemic arterial blood pressure measurement” is a model of what information should be captured for this kind of measurement; usually systolic and diastolic pressure, patient state (position, exertion level) and instrument or other protocol information [19].

To understand the language and the structure, Archetypes have been developed by openEHR foundation. This could be understood by computers and makes it possible to query the data. openEHR defines a method of querying a set of clinical data called Archetype Query Language (AQL), described in section 2.2.

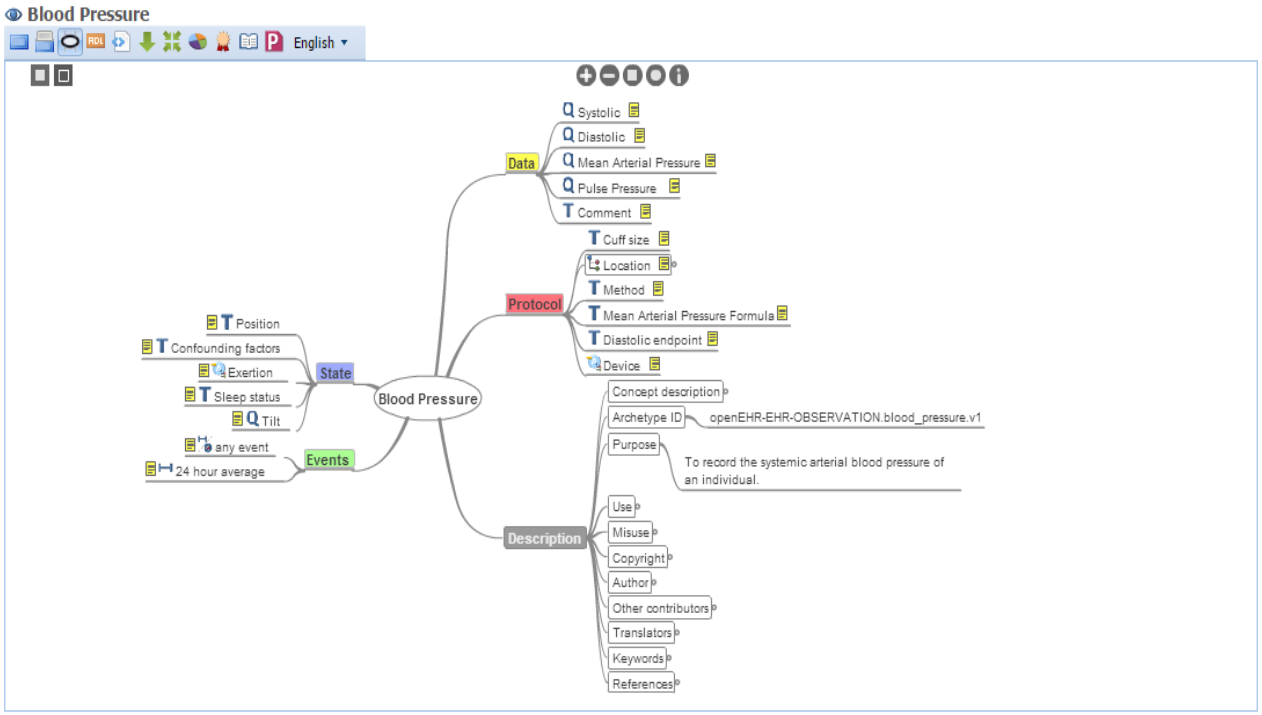


Fig: 2.2.3. [23] Systemic Arterial Blood Pressure Measurement

2.2.4 Templates

Templates are more detailed specifications which are used for local usable restrictions/constraints. It composes archetypes into larger structures that often correspond to documents, reports or messages. It defines which archetypes to chain together, establishes values for optional fields in archetypes, specificied languages and terminologies to be used and may add further local constraints on the archetype [7, 8].

Example: Archetypes of blood pressure, weight and blood sugar may be required when recording an annual review of diabetic person or in an antenental visit by a pregnant woman. So templates will be created that are specific to “diabetic review” and “antenatal visit” [31].

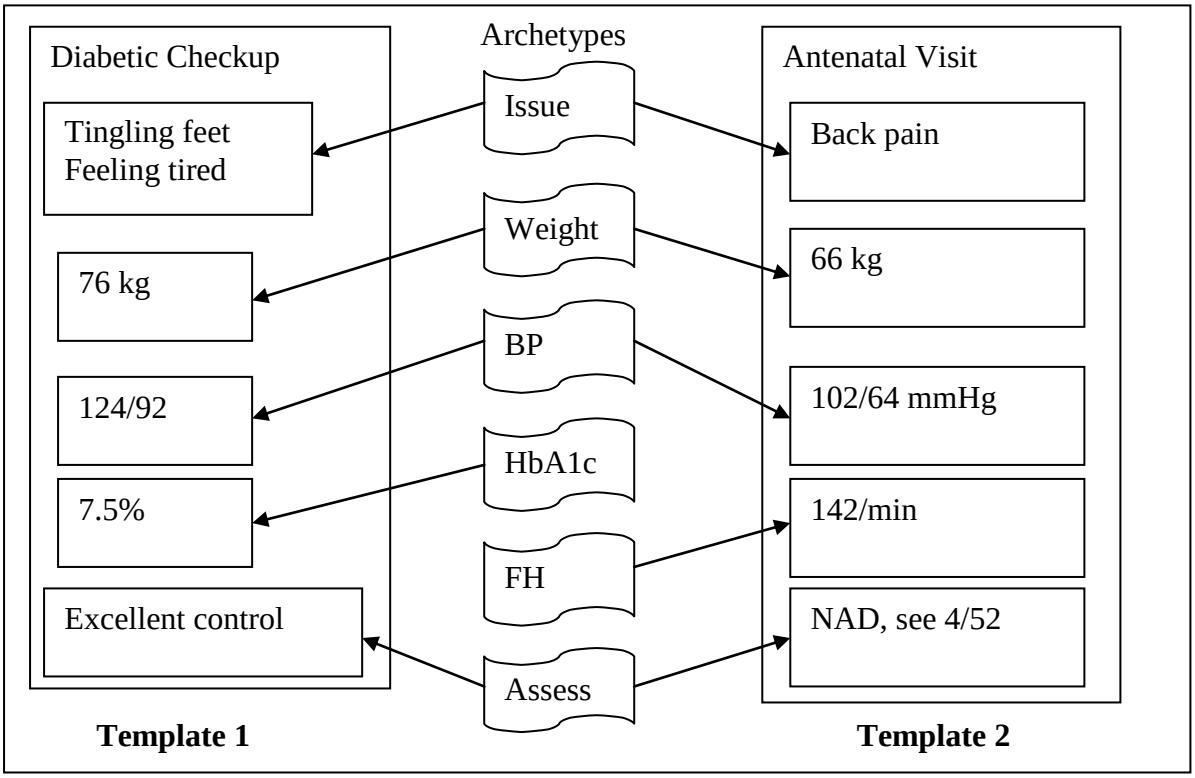


Fig: 2.2.4. [12, 30] Templates

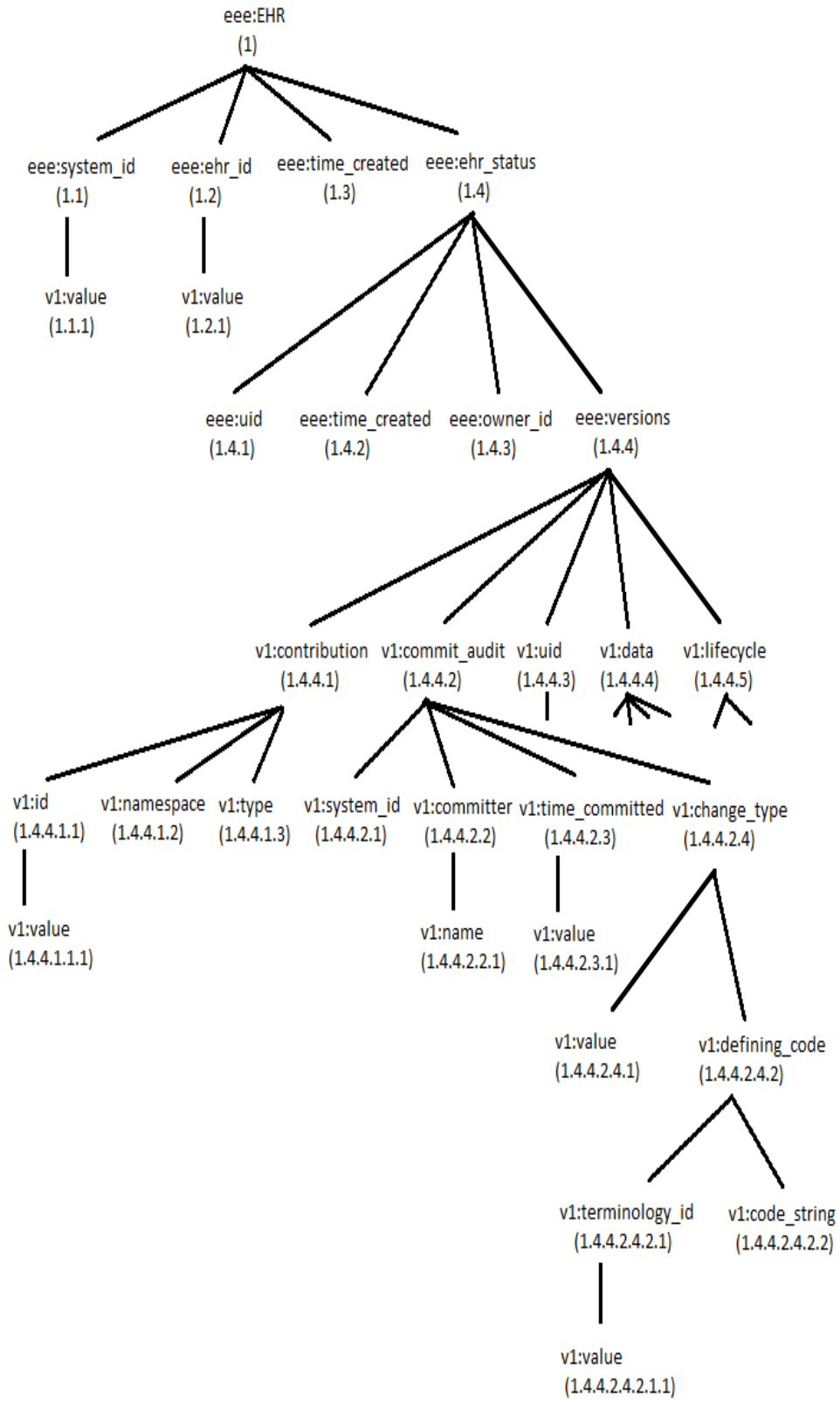
2.3 DEWEY CODING

To fetch the data from XML, indexing were done by giving dewey id’s from the root till the leaf. For example: for the following XML data, the dewey id’s are given to each element from the root till the leaf ends. So the output will be like Figure 1 below.

```
<eee:EHR xmlns:v1=“http://schemas.openEHR.org/v1”
xmlns:eee=“http://www.imt.liu.se/mi/ehr/2010/EEE-v1.xsd”>
        <eee:system_id>
                <v1:value>test2.eee.mi.imt.liu.se</v1:value>
        </eee:system_id>
        <eee:ehr_id>
                <v1:value>00000000-0000-0000-0000-000000000076</v1:value>
        </eee:her_id>
        <eee:time_created>
```

```
        <v1:value>2006-05-29T04:29:43, 000+01:00</v1:value>
</eee:time_created>
<eee:ehr_status xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                xsi:type="eee:VERSIONED_EHR_STATUS">
        <eee:uid>
                <v1:value>45941565-da9a-4a12-8e55-d4e363e50666</v1:value>
        </eee:uid>
        <eee:time_created>2006-05-29T04:29:43, 000+01:00</eee:time_created>
        <eee:owner_id>00000000-0000-0000-0000-000000000076</owner_id>
        <eee:versions>
                <v1:contribution>
                        <v1:id>
                                <v1:value>a6183d90-d438-483a-b597-9c87c8e04774
                                </v1:value>
                        </v1:id>
                        <v1:namespace>test2.eee.mi.imt.liu.se</v1:namespace>
                        <v1:type>CONTRIBUTION</v1:type>
                </v1:contribution>
                <v1:commit_audit>
                        <v1:system_id> test2.eee.mi.imt.liu.se</v1:system_id>
                        <v1:committer xsi:type="v1:PARTY_IDENTIFIED">
                                <v1:name>EEE Testcase Import Service</v1:name>
                        </v1:committer>
                        <v1:time_committed>
                                <v1:value>2006-05-29T04:29:43, 000+01:00
                                </v1:value>
                        </v1:time_committed>
                        <v1:change_type>
                                <v1:value>creation</v1:value>
                                <v1:defining_code>
                                        <v1:terminology_id>
                                                <v1:value>openEHR</v1:value>
                                        </v1:terminology_id>
                                        <v1:code_string>249</v1:code_string>
                                </v1:defining_code>
                        </v1:change_type>
                </v1:commit_audit>
                <v1:uid>
                        <v1:value>b95702fe-bb4b-4289-bc16-9c8ea326e1f2::
                                        test2.eee.mi.imt.liu.se::1
```

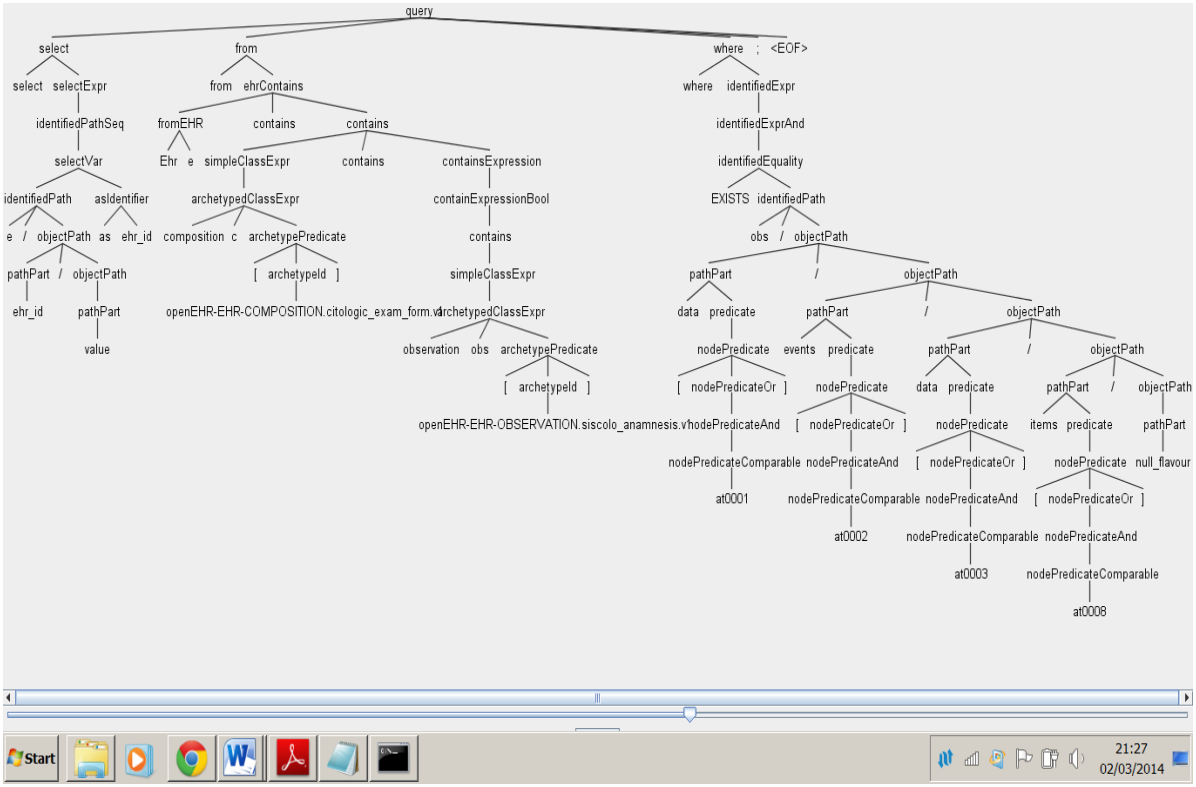
```
                    </v1:value>
                </v1:uid>
                <v1:data archetype_node_id="openEHR-EHR-
                    ITEM_TREE.ehrstatus.v1" xsi:type= "v1:EHR_STATUS">
                    <v1:name>
                        <v1:value>EHR Status</v1:value>
                    </v1:name>
                    <v1:is_queryable>true</v1:is_queryable>
                    <v1:is_modifiable>true</v1:is_modifiable>
                    <v1:subject xsi:type="v1:PARTY_SELF">
                        <v1:external_ref>
                            <v1:id xsi:type="v1:HIER_OBJECT_ID">
                                <v1:value>00000000-0000-0000-
                                0000-000000000076</v1:value>
                            </v1:id>
                            <v1:type>PARTY</v1:type>
                        </v1:external_ref>
                    </v1:subject>
                </v1:data>
                <v1:lifecycle_state>
                    <v1:value>complete</v1:value>
                    <v1:defining_code>
                        <v1:terminology_id>
                            <v1:value>openEHR</v1:value>
                        </v1:terminology_id>
                        <v1:code_string>532</v1:code_string>
                    </v1:defining_code>
                </v1:lifecycle_state>
            </v1:versions>
        </eee:ehr_status>
</eee:EHR>
```


eee:EHR
(1)
eee:system_id
(1.1)
eee:ehr_id
(1.2)
eee:time_created
(1.3)
eee:ehr_status
(1.4)
v1:value
(1.1.1)
v1:value
(1.2.1)
eee:uid
(1.4.1)
eee:time_created
(1.4.2)
eee:owner_id
(1.4.3)
eee:versions
(1.4.4)
v1:contribution
(1.4.4.1)
v1:commit_audit
(1.4.4.2)
v1:uid
(1.4.4.3)
v1:data
(1.4.4.4)
v1:lifecycle
(1.4.4.5)
v1:id
(1.4.4.1.1)
v1:namespace
(1.4.4.1.2)
v1:type
(1.4.4.1.3)
v1:system_id
(1.4.4.2.1)
v1:committer
(1.4.4.2.2)
v1:time_committed
(1.4.4.2.3)
v1:change_type
(1.4.4.2.4)
v1:value
(1.4.4.1.1.1)
v1:name
(1.4.4.2.2.1)
v1:value
(1.4.4.2.3.1)
v1:value
(1.4.4.2.4.1)
v1:defining_code
(1.4.4.2.4.2)
v1:terminology_id
(1.4.4.2.4.2.1)
v1:code_string
(1.4.4.2.4.2.2)
v1:value
(1.4.4.2.4.2.1.1)

Dewey code

2.4 ABSTRACT SYNTAX TREE (AST)

AST is an output from parser phase of the compiler. AST is a finite, directed, labeled tree where its structure depends on input sentences and parser specifications. Also its construction is based on parent/child tree relationship by providing the grammar annotations that indicate which node or what tokens are to be treated as parent or subtree root and which are to be taken or treated as leaves and which are to be ignored with respect to tree construction. This AST is given as input to the next phase of the compiler. Below is a sample AST:



2.5 AQL

More information related to this section can be found at [3].

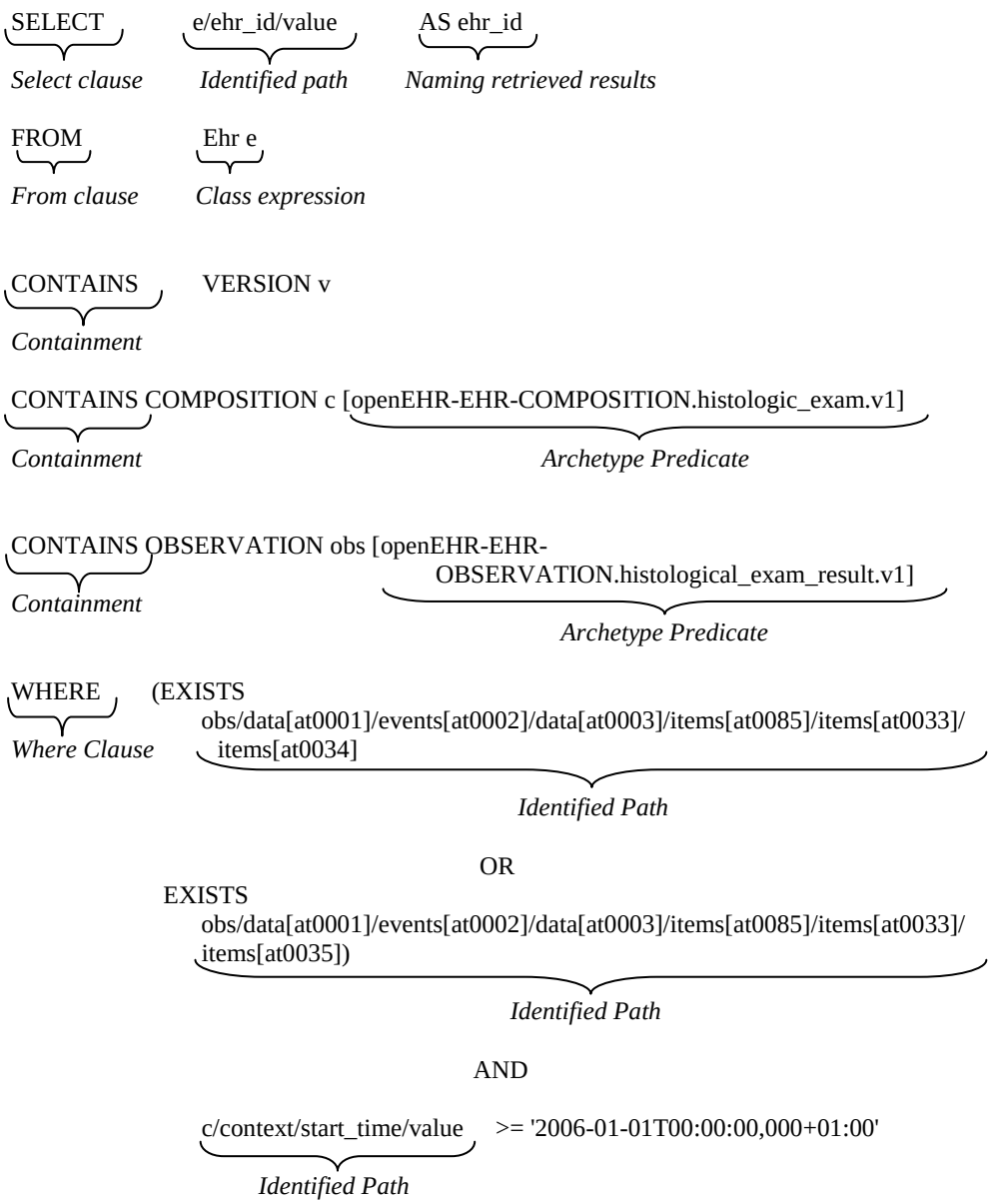
2.5.1 Introduction and Features

Archetype Query Language (AQL) is a query language which is been developed for querying EHR data. It is a declarative query language and by using this language we can

search and retrieve the clinical data found in archetype based EHRs. Unlike the other languages AQL syntax has its own formalisms which is quite different and independent of applications, programming languages, etc. AQL queries are defined at the archetype level (semantic level). It was first called as EQL (EHR Query Language) which is enhanced by two innovations [21]. AQL also has some unique features and other features which can be found from other query languages [3].

2.5.2 AQL Structure and Syntax Description

Similar to SQL clauses, AQL also has clauses like SELECT, FROM, WHERE, ORDER BY and TIMEWINDOW.



AND

c/context/start_time/value < '2006-05-01T00:00:00,000+01:00'

Identified Path

Consider the above AQL query example. This example query is to return all the record ids that have a histologic exam result between '2006-01-01T00:00:00,000+01:00' and '2006-05-01T00:00:00,000+01:00'.

AQL structure has the following clauses and it must be listed in the same order as listed below:

SELECT clause: which is mandatory

FROM clause: which is mandatory

WHERE clause: which is mandatory

ORDER BY: which is optional

TIMEWINDOW: which is optional

We can see each one by one in detail below:

2.5.2.1 AQL SELECT CLAUSE

A SELECT clause is followed by a set of *Identified Path* with the optional *Naming Retrieved Results*. A set of *Identified Paths* are separated using a ‘,’(comma). Its function is similar like SELECT clause of SQL expression.

2.5.2.1.1 Identified Paths

Identified Paths are used to find the data items to be returned in the SELECT clause on which the query criteria are applied in WHERE clause. This identified path starts with a variable followed by a slash and an openEHR path.

Example: obs/data[at0001]/events[at0002]/data[at0003]/items[at0010]/null_flavour

The identified path has three forms. It starts with AQL variable followed by:

- *an openEHR path*

  Example: obs/data[at0001]/events[at0002]/data[at0003]/items[at0010]/null_flavour
- *a Predicate*

  Example: obs[name/value=$nameValue]
- *a Predicate and an openEHR path*

Example: obs[name/value=$nameValue]/data[at0001]/events[at0002]/data[at0003]/
items[at0010]/null_flavour

2.5.2.1.2 Naming Retrieved Results

This is optional to use which comes after the "SELECT *identifiedpath*" as similar to SQL naming retrieval results to display the retrived results.

Example: SELECT e/ehr_id/value AS ehr_id….

2.5.2.2 AQL FROM CLAUSE

This clause is followed by *class expression* (RM class name. Example: EHR), *variable* (example: e) *and containment expression*.

2.5.2.2.1 Class Expression

The class expression is the RM class name (EHR) followed by a variable and/or Predicate. Here, both the variable and the Predicate are optional.

Example: FROM EHR / FROM EHR e / FROM EHR[ehr_id/value=$ehrUid]

2.5.2.2.2 Containment Expression

It is used to identify the hierarchical relationship among the found Archetypes. Consider the below example:

```
From EHR e
contains version v
contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
contains observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
```

Here, the Composition Archetype is the parent of Observation Archetype and the Version Archetype is the parent of Composition Archetype.

2.5.2.3 AQL WHERE CLAUSE

It is followed by Identified expression. These Identified expression has two operands (left and right) and an operator: The left operand is an Identified Path. The right operand is the data value such as String or Integer or parameter or Identified Path. A parameter name starts with '$' symbol. Operator is the basic operators such as (<, >, <=, >=, =, !=). The *Identified Path* is already explained under the topic 2.5.2.1.1.

2.5.2.4 AQL ORDERBY CLAUSE

ORDER BY clause is optional to use which is followed by *Identified Path* and/or the keyword containing ASC, ASCENDING, DESC or DESCENDING in order to display the result in a sorted order.

Example: *ORDER BY* c/name/value.

2.5.2.5 AQL TIMEWINDOW CLAUSE

TIMEWINDOW clause is followed by the time interval [21].

2.5.3 Other Syntax Descriptions

Similar to other languages, AQL also has the usage of Variables, Parameters, Operators and other terminologies which can be read more on [3].

2.5.3.1 openEHR Path Syntax

The openEHR path syntax has two patterns. The first pattern is the path to the attribute of RM classes and the second pattern is the path to the attribute of archetype node (indicated by square brackets '[]').

Examples: /context/start_time.

/data[at0001]/events[at0002]/…../value/value

/data[at0001]/events[at0002]/…../items[at0034]

2.6 Translators

The language translators are used to translate from one programming language written in source code to another equivalent program in object code without changing its meaning or structure of the original code or program. The source program is called high-level language and the object program is called machine code [33]. The compiler, interpreter are examples of translators.

2.6.1 Compiler Introduction

The compiler is a translator that translates the code or program written in one language (source language) to another language (target language) without changing its meaning. The source language is the High-level language which is human understandable such as C and the target language is the machine language or assembly language which is machine

understandable such as binary language or machine oriented language. An important part of compiler is translation, error detection and recovery.

There are two stages of compiler process:

- Analysis stage
- Synthesis stage

1) Analysis stage is the frontend of the compiler where the input is the source code. The compiler reads the source code and outputs the intermediate code generation. It is also responsible for error checking, lexer, grammar and parser. This stage includes first three phases of compiler.
2) Synthesis stage is the backend of the compiler where it takes the intermediate code generation as input and generates the final target code. This stage includes last three phases of compiler.

2.6.1.1 Phases of Compiler

There are six phases of compiler in which each interacts with symbol table and error handler. They are

- Lexical analyzer or scanner
- Syntax analyzer or parser
- Semantic analyzer
- Intermediate code generation
- Code optimization
- Code generation

As mentioned above, the first three phases are collectively called frontend and the last three phases are collectively called backend of the compiler process.

2.6.1.1.1 Lexical analyzer or scanner

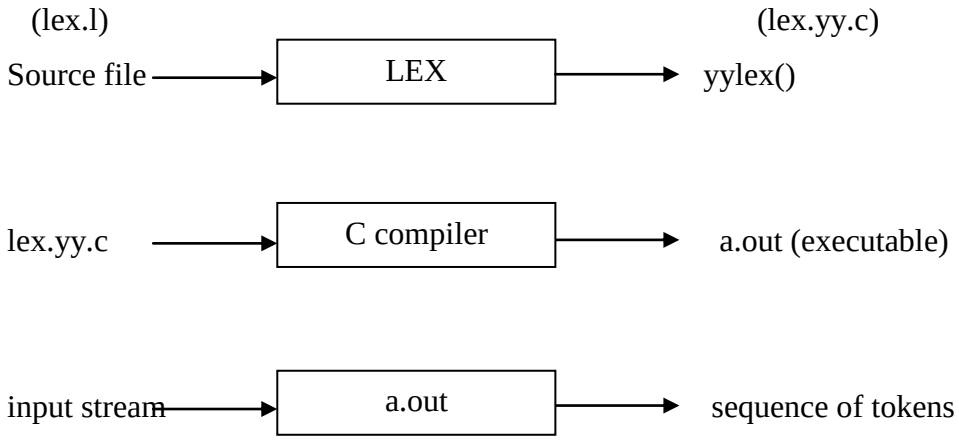
The lexical analyzer or lexical analysis can be said as lexer in short. It is otherwise called scanner in which it scans the source code, reads line by line as a stream of characters and converts into lexemes. This lexeme is represented in the form of tokens. Hence the output will be full of tokens.

Example: x = y + 1

The tokens for the above are produced as <id1> <assign> <id2> <addop> <const>
Here, id1 and id2 are identifiers or names, assign is an assignment operator, addop is an addition operand and const is a constant value.

Lexical analysis is difficult to hard code by hand. They automatically generate from regular expressions. Regular expressions are sequence of string for describing pattern matching or string matching. A set of integers or a set of variables are denoted to be said as sequence of strings, in which each letters are taken from particular alphabets. The set of strings are called language. For integer constants, the alphabets contain digits from 0-9 and for names of the variable, the alphabets contain both digits 0-9 and letters a-z.
For example: [b] is a regular expression that matches the occurrence 'b' in a given string.
[b]+ is a regular expression that matches the occurrence 'b' and '+' means 'occurring one or more times'
[b]* is a regular expression consists of strings that can be obtained by concatenating zero or more times.
For the above example regular expressions, the languages are denoted as L{b}, L{b+} and L{b*} respectively.

Example tool for Lexical analysis: Lex and Flex are the common compiler writing tools.
LEX



Overview of Lex tool

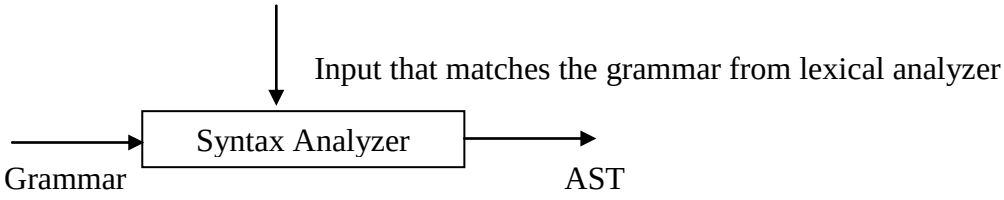
Lex is a lexical analyzer generator tool i.e. programs which recognize lexical patterns in text. These patterns are described by regular expressions. These regular expressions are defined by the user as a source file to the lex. This source file contains a table of regular expressions and program fragments [35]. So the input is given or defined in the form of regular expressions called rules. The lex written code recognizes these regular expressions in input stream and breaks the input stream into token by matching the pattern of regular expressions. Once the match or expression is found, then the corresponding actions are executed. It generates a C source code or file lex.yy.c as an output which contains or defines a scanning routine yylex(). This lex.yy.c file is then compiled and linked with the library called –lfl to produce an executable called scanner. When this executable is run, the input is analyzed to match regular expression [33]. It is not a complete language tool [34]. So it was designed to produce lexical analyzers that could be used with YACC tool [35]. This can be work on every UNIX systems. Some versions of lex are open source. Example: Flex.

Flex

Flex – Fast LEXical analyzer is a counter part of lex. It is same as lex tool and it is faster version of lex. Lex/Flex is a companion to the Yacc/Bison parser generator [33].

2.6.1.1.2 Syntax analyzer or parser

The syntax analyzer is also known as parser in which it takes the token from the lexical analyzers output and produces the syntax tree or parse tree. This is responsible for looking or checking the syntax rules of the language i.e. context-free grammar and produces intermediate representation of the language i.e. Abstract Syntax Tree.



Here, the grammar is given first to the syntax analyzer in which the input is checked according to the grammar.

Example of context-free grammar:

S → E c

E → a F

F → b

Here, ‘S’, ‘E’ and ‘F’ are terminals or terminal symbols and ‘a’, ‘b’ and ‘c’ are non-terminals or non-terminal symbols where terminal symbols can again be parsed i.e. from ‘S’, ‘E’ can be substitute as ‘a’ ‘F’ and again ‘F’ can be substitute as ‘b’ etc.

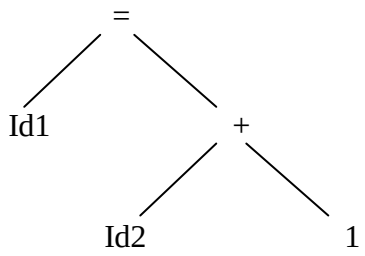
S → E c

→ a F c

→ a b c

Here, ‘S’ is the start symbol and the ‘→’ symbol means S *can have the form* E c. This rule is called production. Syntax analysis can be generated automatically from context-free grammar. LL(1), LALR(1) are the most popular ones.

Example output of Syntax analyzer:



2.6.1.1.2.1 Types of parsers:

If we can construct more than one AST for a given grammar, then those grammars are said to be ambiguous grammars. To avoid that the parsers have 2 types:

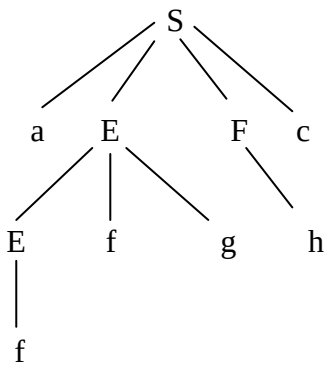
- **Top Down:** Top down parsers build the tree or parse the token from top (root) to bottom (leaves).

  Example:

  S → a E F c

  E → E f g | f

  F → h



- General form of top down parsers is called *recursive descent parsers* which may require backtracking i.e. it needs repeated scans over the input. However, it is rarely needed to parse programming language construct. So backtracking parsers are not seen often.
- Recursive descent parsers with no backtracking are called *predictive parsers*. This can be constructed for a class of grammars called LL(1).
- *Example: For the above production, it is solved as:*

  S → a E F c

  → a E f g h c

  → a f f g h c
- In LL(1), the first L means scanning from left most symbols to right most symbols is called left-to-right, the second L is the process of deriving the tree, i.e. leftmost derivation is used which is given as an example above and 1 is the lookahead which means how many symbols we are going to see while parsing. Since we are going to see one by one it is LL (1). In general, the class of grammars in which we construct predictive parsers by looking for 'k' symbols in the input are called LL(k).

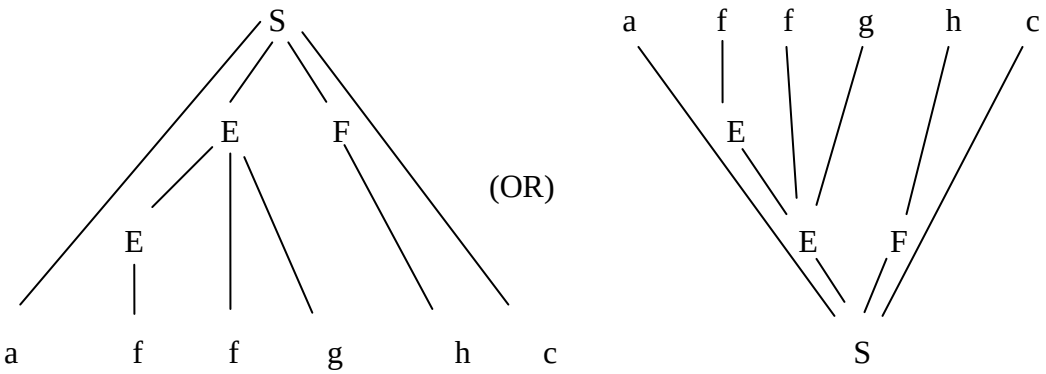
- **Bottom up:** Bottom up parsers build the tree or parse the token from bottom (leaves) to top (root). The general style of this method is called SR (Shift-Reduce) parsers. The largest class of grammars for which shift reduce parsers are built is called LR grammar. In LR(1), the first L means scanning from left most symbols to right most symbols is called left-to-right, the second R is called right most derivation and 1 is the lookahead which means how many symbols we are going to see while parsing. Since we are going to see one by one it is LR (1). In general, 'k' is the number of symbols in the input LL(k). Here, k is either 0 or 1 and when k is omitted, then k is assumed to be 1.
- The easiest method of shift reduce parsers are called SLR – Simple LR grammar. The other bottom up parsers are: LR(0) or canonical LR and LALR – Look Ahead LR grammar methods.

  Example of bottom up parser:

  S → a E F c

  E → E f g | f

  F → h



Example of LL(1) is ANTLR and example of LALR(1) is YACC and Bison.

YACC

As mentioned earlier, Yacc is commonly used with lex and it is abbreviated as Yet Another Compiler Compiler tool. It is a parser generator tool to recognize the grammatical structure of programs available as a command on most UNIX system. Given a context free grammar, YACC will generate a C program in which it will parse the input according to the grammar rules [33, 35].

The context free grammar is given as input to the yacc tool and generates a C program which will parse input to the grammar rules. The input file consists of three sections: The first section contains a list of tokens that are expected by the parser and the specifications of the start symbol of the grammar. The second section contains context free grammar for the language and the third section contains C code. The C code has a main routine called main() which calls yyparse() function. This function is the driver routine for the parser. There is also yyerror() function used to report on errors during parsing [33].

When we combine lex with yacc tool, the generated yy.lex.c by lex tool is given to the yacc for analyzing or to find the next input token. This will return the type of next token. Normally, this routine is called by the main program of lex tool but if yacc is loaded and the main program is used, yacc will call this routine [34].

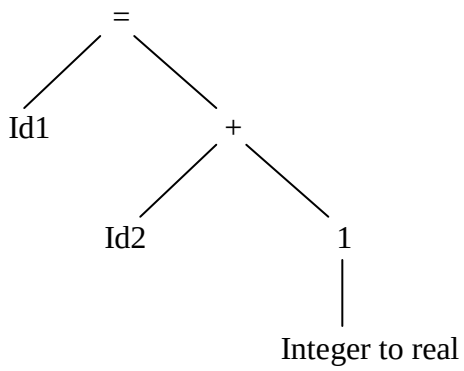
Bison

Bison is a faster version of YACC and it's a product of free software foundation [33]. It is often used with Flex.

2.6.1.1.3 Semantic analyzer

Semantic analyzer checks whether the generated tree follows the semantic rule of the language. It is responsible for type casting whether it follows or converts integer as string, float to real, string to integer, etc. In other words, it keeps track of identifiers, expressions and their types. The result will be an annotated syntax tree which results the end of front-end of the compiler.

Example:



2.6.1.1.4 Intermediate code generation

The next phase is an intermediate code generation. This takes the syntax tree and produces the intermediate code or intermediate language representation of the source code for target machine. This is in between stage of source code or High-level language and target code or machine language. Also the end of this intermediate code generation phase ends up the front-end of the compiler.

Example:

temp1 = integer to real (1)

temp2 = id2 + temp1

id1 = temp2

2.6.1.1.5 Code optimization

The input to the backend is the intermediate code representation. This is given as input to the code generation and optimizes the code to improve the performance for better output.
Example: Id1 = id2 + 1.0

2.6.1.1.6 Code generation

The next phase is the code generation in which it takes the optimized code representation and produces the target code. The code generation translates the intermediate code into sequence of re-locatable target or machine code.
Example: MOVF id2, r1

ADDF 1.0, r1

MOVF r1, id1

2.6.1.1.7 Symbol table management

The symbol table maintains and stores all the possible identifiers, variables, names, their types throughout the whole process of the compiler phases. This is a kind of data structure which makes it easier to search and fetch anything from the record.

2.6.2 Interpreter

Interpreter is another way of translating the language. Lexer, parser and semantic type checking are also in an interpreter which is similar to compiler.
The compiler and interpreter may be combined to implement a programming language. This means that, the compiler may produce intermediate code representation and gives it to the interpreter where it is interpreted instead of producing machine code [33].

2.7 Some examples of automatic Parser Generator Tools

2.7.1 ANTLR

ANTLR is a top down parser generator with many features when compared to javacc tool. It has LL(1) parser algorithm with EBNF notation. It produces lexer, parser and AST which everything is built in the tool. The interpreter can also be used in ANTLR tool to translate the languages. The documentation was good and the working platform is windows and .net. It supports C, C++, java and C#. Also, the required AQL grammar was already built using this tool.

2.7.2 Javacc

Javacc is abbreviated as JavaCompilerCompiler which is a top down parser generator tool. It is used in many applications and much like ANTLR but with having few features. The AST tree has a separate tool called JJtree which is combined with Javacc tool. However, it is only a java generator tool and also the documentation is poor when compared to ANTLR.

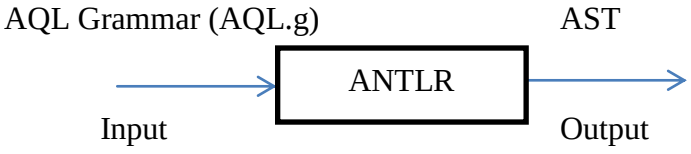
2.7.3 SableCC

SableCC is a bottom up parser generator in which it takes object oriented methodology for constructing parsers. It maintains easy code for generated parser as a result. However, some performance issues are exists. It generates or supports C++ and java.
Other example tools for bottom up parser generators are Lemon, Gold, etc.

2.8 ANTLR BASIC INTRODUCTION

ANTLR references can be seen in [4, 5, 6, 10]. ANTLR is an abbreviation for ANother Tool for Language Recognition. It is a powerful parser generator that uses LL(*) parsing for reading, processing, executing, or translating structured text or binary files. It is the successor to the Purdue Compiler Construction Tool Set (PCCTS) which is first developed in 1989. ANTLR is maintained by Terence Parr who is professor at the University of San Francisco.

ANTLR takes a grammar (here, AQL grammar) that specifies a language description as input and generates source code/parser that automatically builds AST/parse trees (Abstract Syntax Tree, data structures representing how a grammar matches the input) for a recognizer for that language as output. In addition, it also generates tree walkers to visit each node of the tree (AST) in order to execute the application specific code. ANTLR is written in Java and supports generating code in the programming languages such as C, C#, Python, Perl, Java, Javascript, etc., but at present its main focus is on Java and C#. A language is specified using context free grammar (CFG) which is expressed using EBNF. EBNF is an Extended Backups-Naur Form which is a grammar language for describing parser grammars.



ANTLR allows generating lexers and parsers automatically. Lexer translates a stream of characters to a stream of tokens and Parser translates a stream of tokens to a stream of tree nodes. Parser also generates an AST in order to process with tree parsers.

2.6.1 GRAMMAR WRITING

A grammar begins with **grammar Grammarname; [40].**

In this, the Grammarname is the name of the filename. Here, the filename is given as Aql. So we add as **grammar Aql;**

Except this **"grammar Aql";** all the rest of the grammar defined below this line such as "Grammar rule definitions, patterns, etc (See Appendix:B for defining the grammar)" are fully functional.

2.6.1.1 EBNFs

As mentioned before, EBNF is an Extended Backups-Naur Form – a language that describes about parser grammars. It is a set of rules that are structured and defined recursively. EBNF starts with a start symbol, a set of terminals, a set of non-terminals and a set of productions or rewrite-rules. The format of EBNF rule is as follows:

b : c;

Here, the symbol 'c' on the right hand side can be placed instead of symbol 'b'. This replacement process is repeated until there is no left hand symbol referencing a rule exists in the grammar. All the symbols are represented as 'tokens'. The symbols on the right hand side may be unlimited. For example:

a : b | c;

Here '|' symbol represents "or". Either the symbol "b" is chosen to be replaced or the symbol "c" is chosen to be replaced in the place of symbol "a.

For more about the grammar rules, see the reference [40].

2.6.1.2 Start Rule and EOF

ANTLR tool generates recursive-descent parsers from grammar rules (start symbol). In our case 'query' is the start symbol. The term 'descent' refers to the fact that parsing begins at the start of the root of parse tree and proceeds towards the leaves (tokens). The rule which we invoke first (query) indicates the start symbol and becomes the root of the parse tree. In

general, this type of parsing is called ‘top-down parsing’. Hence, recursive-descent parsers are top-down parsing implementation.

The start symbol ‘query’ is defined as a ‘rule definition’ which means the start symbol begins with the root of parse tree ‘query’.

// Rule Definition

query : select from where? orderBy? ';'? EOF;

Here, the symbol ‘?’ means that the symbol (or group of symbols in parenthesis) to the left of the operator is optional (it can appear zero or one times). To denote the end of file token inside ANTLR rules, we simply use EOF which means it ends once it matches all the input.

Some example input for matching the grammar:

query : select from where? orderBy? ';'? EOF;

select : …….;

from : ……..;

where : ……..;

orderby : ……..;

is,

SELECT emp_id FROM employee;

SELECT emp_id FROM employee WHERE emp_id > 20;

SELECT emp_id FROM employee ORDERBY emp_name;

Here, “where”, “orderby” and “;” appears optionally.

Some of the rules with notations and their meaning are as follows:

( ) → Parenthesis can be used to group several elements and can be treated as one single token.

AB → Matches A followed by B

A | B → Matches A or B

A? → Matches zero or one occurences of A

A* → Matches zero or more occurences of A

A+ → Matches one or more occurences of A

Let us take one example grammar from the appendix.

AQL:

```
select e/ehr_id/value as ehr_id
from Ehr e
contains version v
contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
contains observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
where (EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/
      items[at0034] OR
      EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/
      items[at0035]) AND
      c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00' AND
      c/context/start_time/value < '2006-05-01T00:00:00,000+01:00';
```

This AQL query example is to return all the record ids that had a histologic exam result indicating neoplastic lesions between 2006-01-01 and 2006-05-01 [1].

Substitution

- query : select
- select : SELECT (selectExpr)[1]
- **"select"** (selectExpr)[1]
- (selectExpr)[1] : identifiedPathSeq
- identifiedPathSeq : selectVar
- selectVar : (identifiedPath)[2] (asIdentifier)[3]
- (identifiedPath)[2] : (IDENTIFIER)[4] '/' (objectPath)[5]
- (IDENTIFIER)[4] : '\"? LETTERMINUSA IDCHAR* '\"?
- "e" → **"e" "/"**
- (objectPath)[5] : (pathPart)[6] '/' (objectPath)[7]
- (pathPart)[6] : IDENTIFIER
- IDENTIFIER : '\"? LETTERMINUSA IDCHAR* '\"?
- "ehr_id" →**"e" "/" "ehr_id" "/"**
- (objectPath)[7] : pathPart
- pathPart : IDENTIFIER

- IDENTIFIER : '\"? LETTERMINUSA IDCHAR* '\"?
- “value” → **“e” “/” “ehr_id” “/” “value”**
- (asIdentifier)[3] : AS (IDENTIFIER)[8]
- **“as”** (IDENTIFIER)[8]
- (IDENTIFIER)[8] : '\"? LETTERMINUSA IDCHAR* '\"?
- **“ehr_id”**
- from : FROM (ehrContains)[9]
- **“from”** (ehrContains)[9]
- (ehrContains)[9] : fromEHR (CONTAINS)[10] (contains)[11]
- fromEHR : EHR (IDENTIFIER)[12]
- **“ehr”** (IDENTIFIER)[12]
- (IDENTIFIER)[12] : '\"? LETTERMINUSA IDCHAR* '\"?
- **“e”**
- (CONTAINS)[10] : **“contains”**
- (contains)[11] : simpleClassExpr (CONTAINS)[13] (containsExpression)[14]
- simpleClassExpr : IDENTIFIER (IDENTIFIER?)[15]
- IDENTIFIER : '\"? LETTERMINUSA IDCHAR* '\"?
- **“version”**
- (IDENTIFIER)[15] : '\"? LETTERMINUSA IDCHAR* '\"?
- **“v”**
- (CONTAINS)[13] : **“contains”**
- (containsExpression)[14] : containExpressionBool
- containExpressionBool : contains
- contains : simpleClassExpr (CONTAINS)[16] (containsExpression)[17]
- simpleClassExpr : archetypedClassExpr
- archetypedClassExpr : IDENTIFIER (IDENTIFIER)[18] (archetypePredicate)[19]
- IDENTIFIER : '\"? LETTERMINUSA IDCHAR* '\"?
- **“composition”**
- (IDENTIFIER)[18] : '\"? LETTERMINUSA IDCHAR* '\"?
- **“c”**
- (archetypePredicate)[19] : OPENBRACKET (archetypeId)[20] CLOSEBRACKET
- **“[“**archetypeId **“]”**
- (archetypeId)[20] : ARCHETYPEID

- ARCHETYPEID : LETTER+ '-' LETTER+ '-' (LETTER|'_')+ '.' (IDCHAR|'-')+ '.v'
- DIGIT+ ('.' DIGIT+)?
- **“openEHR-EHR-COMPOSITION.histologic_exam.v1”**
- (CONTAINS)[16] : **“contains”**
- (containsExpression)[17] : containExpressionBool
- containExpressionBool : contains
- contains : simpleClassExpr
- simpleClassExpr : archetypedClassExpr
- archetypedClassExpr : IDENTIFIER (IDENTIFIER)[21] (archetypePredicate)[22]
- IDENTIFIER : '\"? LETTERMINUSA IDCHAR* '\"?
- **“observation”**
- (IDENTIFIER)[21] : '\"? LETTERMINUSA IDCHAR* '\"?
- **“obs”**
- (archetypePredicate)[22] : OPENBRACKET (archetypeId)[23] CLOSEBRACKET
- **“[”** (archetypeId)[23] **“]”**
- (archetypeId)[23] : ARCHETYPEID
- ARCHETYPEID : LETTER+ '-' LETTER+ '-' (LETTER|'_')+ '.' (IDCHAR|'-')+ '.v'
- DIGIT+ ('.' DIGIT+)?
- **“openEHR-EHR-OBSERVATION.histological_exam_result.v1”**
- where : WHERE (identifiedExpr)[24]
- **“where”** (identifiedExpr)[24]
- (identifiedExpr)[24] : identifiedExprAnd
- identifiedExprAnd : identifiedEquality (identifiedAndOperator)[25] (identifiedEquality)[26] (identifiedAndOperator)[27] (identifiedEquality)[28]
- identifiedEquality : **'('** (identifiedExpr)[29] **')'**
- (identifiedExpr)[29] : identifiedExprAnd (identifiedOrOperator)[30] (identifiedExprAnd)[31]
- identifiedExprAnd : identifiedEquality
- identifiedEquality : EXISTS (identifiedPath)[32]
- **“exists”** (identifiedPath)[32]

- (identifiedPath)[32] : IDENTIFIER '/' (objectPath)[33]
- **“obs” “/”**
- (objectPath)[33] : pathPart '/' (objectPath)[34]
- pathPart : IDENTIFIER (predicate)[35]
- **“data”** (predicate)[35]
- (predicate)[35] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[36] CLOSEBRACKET
- **“[”**(nodePredicateOr)[36] **“]”**
- (nodePredicateOr)[36] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0001” “]” “/”**
- (objectPath)[34] : pathPart '/' (objectPath)[37]
- pathPart : IDENTIFIER (predicate)[38]
- **“events”** (predicate)[38]
- (predicate)[38] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[39] CLOSEBRACKET
- **“[”**(nodePredicateOr)[39] **“]”**
- (nodePredicateOr)[39] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0002” “]” “/”**
- (objectPath)[37] : pathPart '/' (objectPath)[40]
- pathPart : IDENTIFIER (predicate)[41]
- **“data”** (predicate)[41]
- (predicate)[41] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[42] CLOSEBRACKET
- **“[”**(nodePredicateOr)[42] **“]”**

- (nodePredicateOr)[42] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0003” “]” “/”**
- (objectPath)[40] : pathPart '/' (objectPath)[43]
- pathPart : IDENTIFIER (predicate)[44]
- **“items”** (predicate)[44]
- (predicate)[44] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[45] CLOSEBRACKET
- **“[”**(nodePredicateOr)[45] **“]”**
- (nodePredicateOr)[45] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0085” “]” “/”**
- (objectPath)[43] : pathPart '/' (objectPath)[46]
- pathPart : IDENTIFIER (predicate)[47]
- **“events”** (predicate)[47]
- (predicate)[47] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[48] CLOSEBRACKET
- **“[”**(nodePredicateOr)[48] **“]”**
- (nodePredicateOr)[48] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0033” “]” “/”**
- (objectPath)[46] : pathPart
- pathPart : IDENTIFIER (predicate)[49]
- **“data”** (predicate)[49]
- (predicate)[49] : nodePredicate

- nodePredicate : OPENBRACKET (nodePredicateOr)[50] CLOSEBRACKET
- **“[”**(nodePredicateOr)[50] **“]”**
- (nodePredicateOr)[50] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0034” “]”**
- (identifiedOrOperator)[30]: **“OR”**
- (identifiedExprAnd)[31] : identifiedEquality
- identifiedEquality : EXISTS (identifiedPath)[51]
- **“exists”** (identifiedPath)[51]
- (identifiedPath)[51] : IDENTIFIER '/' (objectPath)[52]
- **“obs” “/”**
- (objectPath)[52] : pathPart '/' (objectPath)[53]
- pathPart : IDENTIFIER (predicate)[54]
- **“data”** (predicate)[54]
- (predicate)[54] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[55] CLOSEBRACKET
- **“[”**(nodePredicateOr)[55] **“]”**
- (nodePredicateOr)[55] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0001” “]” “/”**
- (objectPath)[53] : pathPart '/' (objectPath)[56]
- pathPart : IDENTIFIER (predicate)[57]
- **“events”** (predicate)[57]
- (predicate)[57] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[58] CLOSEBRACKET
- **“[”**(nodePredicateOr)[58] **“]”**

- (nodePredicateOr)[58] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0002” “]” “/”**
- (objectPath)[56] : pathPart '/' (objectPath)[59]
- pathPart : IDENTIFIER (predicate)[60]
- **“data”** (predicate)[60]
- (predicate)[60] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[61] CLOSEBRACKET
- **“[”**(nodePredicateOr)[61] **“]”**
- (nodePredicateOr)[61] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0003” “]” “/”**
- (objectPath)[59] : pathPart '/' (objectPath)[62]
- pathPart : IDENTIFIER (predicate)[63]
- **“items”** (predicate)[63]
- (predicate)[63] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[64] CLOSEBRACKET
- **“[”**(nodePredicateOr)[64] **“]”**
- (nodePredicateOr)[64] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0085” “]” “/”**
- (objectPath)[62] : pathPart '/' (objectPath)[65]
- pathPart : IDENTIFIER (predicate)[66]
- **“events”** (predicate)[66]
- (predicate)[66] : nodePredicate

- nodePredicate : OPENBRACKET (nodePredicateOr)[67] CLOSEBRACKET
- **“[”**(nodePredicateOr)[67] **“]”**
- (nodePredicateOr)[67] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0033” “]” “/”**
- (objectPath)[65] : pathPart
- pathPart : IDENTIFIER (predicate)[68]
- **“data”** (predicate)[68]
- (predicate)[68] : nodePredicate
- nodePredicate : OPENBRACKET (nodePredicateOr)[69] CLOSEBRACKET
- **“[”**(nodePredicateOr)[69] **“]”**
- (nodePredicateOr)[69] : nodePredicateAnd
- nodePredicateAnd : nodePredicateComparable
- nodePredicateComparable: NODEID
- NODEID : 'at' DIGIT+
- **“at0035” “]” “)”**
- (identifiedAndOperator)[25]: **“AND”**
- (identifiedEquality)[26] : identifiedOperand (COMPARABLEOPERATOR)[70] (identifiedOperand) [71]
- identifiedOperand : identifiedPath
- identifiedPath : IDENTIFIER '/' (objectPath)[72]
- **“c” “/”**
- (objectPath)[72] : pathPart '/' (objectPath)[73]
- pathPart : IDENTIFIER
- **“context” “/”**
- (objectPath)[73] : pathPart '/' (objectPath)[74]
- pathPart : IDENTIFIER
- **“start_time” “/”**
- (objectPath)[74] : pathPart

- pathPart : IDENTIFIER
- **“value”**
- (COMPARABLEOPERATOR)[70] : **“>=”**
- (identifiedOperand)[71] : operand
- operand : operandDate
- **“2006-01-01T00:00:00,000+01:00”**
- (identifiedAndOperator)[27] : **“AND”**
- (identifiedEquality)[28] : identifiedOperand (COMPARABLEOPERATOR)[75] (identifiedOperand)[76]
- identifiedOperand : identifiedPath
- identifiedPath : IDENTIFIER '/' (objectPath)[77]
- **“c” “/”**
- (objectPath)[77] : pathPart '/' (objectPath)[78]
- pathPart : IDENTIFIER
- **“context” “/”**
- (objectPath)[78] : pathPart '/' (objectPath)[79]
- pathPart : IDENTIFIER
- **“start_time” “/”**
- (objectPath)[79] : pathPart
- pathPart : IDENTIFIER
- **“value”**
- (COMPARABLEOPERATOR)[75] **: “<”**
- (identifiedOperand)[76] : operand
- operand : operandDate
- **“2006-05-01T00:00:00,000+01:00”**
-
-
-
-
- **query : select e/ehr_id/value as ehr_id from Ehr e contains version v contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1] contains observation**

obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1] where (EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034] OR EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0035]) AND c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00' AND c/context/start_time/value < '2006-05-01T00:00:00,000+01:00';

Grammar

```
query               :   select from where? orderBy? ';'? EOF;
select              :   SELECT top? selectExpr;
selectExpr          :   identifiedPathSeq;
identifiedPathSeq   :   selectVar;
selectVar           :   identifiedPath asIdentifier;
identifiedPath      :   IDENTIFIER '/' objectPath;
objectPath          :   pathPart
                    |   pathPart '/' objectPath;
pathPart            :   IDENTIFIER;
                    |   IDENTIFIER predicate
asIdentifier        :   AS IDENTIFIER;
from                :   FROM ehrContains;
ehrContains         :   fromEHR CONTAINS contains;
fromEHR             :   EHR IDENTIFIER;
contains            :   simpleClassExpr
                    |   simpleClassExpr CONTAINS containsExpression;
simpleClassExpr     :   IDENTIFIER IDENTIFIER?
                    |   archetypedClassExpr
containsExpression:     containExpressionBool
containExpressionBool:  contains
archetypedClassExpr:    IDENTIFIER IDENTIFIER archetypePredicate;
archetypePredicate :    OPENBRACKET archetypeId CLOSEBRACKET
```

```
archetypeId          :    ARCHETYPEID;
where                :    WHERE identifiedExpr;
identifiedExpr       :    identifiedExprAnd
                     |    identifiedExprAnd identifiedOrOperator identifiedExprAnd;
identifiedExprAnd    :    identifiedEquality
                     |    identifiedEquality (identifiedAndOperator
                          identifiedEquality)*;
identifiedEquality   :    '(' identifiedExpr ')'
                     |    EXISTS identifiedPath
                     |    identifiedOperand COMPARABLEOPERATOR
                          identifiedOperand
predicate            :    nodePredicate;
nodePredicate        :    OPENBRACKET nodePredicateOr CLOSEBRACKET;
nodePredicateOr      :    nodePredicateAnd
nodePredicateAnd     :    nodePredicateComparable
nodePredicateComparable: NODEID
identifiedOrOperator:     OR;
identifiedAndOperator:    AND;
identifiedOperand    :    operand
                     |    identifiedPath;
operand              :    operandDate;
operandDate          :    DATE;
SELECT               :    ('S' | 's') ('E' | 'e') ('L' | 'l') ('E' | 'e') ('C' | 'c') ('T' | 't');
AS                   :    ('A' | 'a') ('S' | 's');
FROM                 :    ('F' | 'f') ('R' | 'r') ('O' | 'o') ('M' | 'm');
CONTAINS             :    ('C' | 'c') ('O' | 'o') ('N' | 'n') ('T' | 't') ('A' | 'a') ('I' | 'i') ('N' | 'n')
                          ('S' | 's') ;
WHERE                :    ('W' | 'w') ('H' | 'h') ('E' | 'e') ('R' | 'r') ('E' | 'e');
EHR                  :    ('E' | 'e') ( 'H' | 'h') ( 'R' | 'r');
AND                  :    ('A' | 'a') ('N' | 'n') ('D' | 'd');
OR                   :    ('O' | 'o') ('R' | 'r');
EXISTS               :    ('E' | 'e') ('X' | 'x') ('I' | 'i') ('S' | 's') ('T' | 't') ('S' | 's') ;
WS                   :    [ \t\r\n]+ -> skip;
```

```
IDENTIFIER      :       '\"? ('a'|'A') (ALPHANUM|'_')* '\"?
                |       '\"? LETTERMINUSA IDCHAR* '\"?
                ;
NODEID          :       'at' DIGIT+;
ARCHETYPEID     :       LETTER+ '-' LETTER+ '-' (LETTER|'_')+ '.' (IDCHAR|'-')+
                        '.v'  DIGIT+ ('.' DIGIT+)? ;
COMPARABLEOPERATOR:     '=' | '!=' | '>' | '>=' | '<' | '<=' ;
OPENBRACKET     :       '[';
CLOSEBRACKET:           ']';
fragment
ALPHANUM        :        LETTER|DIGIT;
fragment
LETTER          :       'a'..'z'|'A'..'Z';
fragment
DIGIT           :       '0'..'9';
fragment
LETTERMINUSA    :       'b'..'z'|'B'..'Z';
fragment
IDCHAR          :       ALPHANUM|'_';
```

Chapter 3

Approach

In this chapter, we are going to see how we are going to translate this AQL query into SQL query.

3.1 Approach

The AQL language is developed to query upon EHR. AQL is not difficult for those who know SQL. To increase the performance and for optimization purpose, we have to translate to high level languages such as XML, XQuery, SQL, etc. As we already discussed about the previous work, EHR data are stored in XPath format which is indexed and parsed as different files with dewey coding. The approach is to translate AQL queries to other query languages such as SQL or XQuery and then run the query natively.

EHR and openEHR references can be read on [1, 7, 8]. The EHR data which is generated by the RM can be serialized into different formats such as JSON, XML, etc. Since there were several documents available, these can be used to store openEHR compliant XML documents. In the paper, "Performance of XML Databases for Epidemiological Queries in Archetype-Based EHRs" [1], the authors had clearly mentioned about the performance of XML database that stores openEHR compliant documents in terms of size and response times to the epidemiological queries. Since the XML databases need more space compared to relational databases, we first translate all the epidemiological data into relational tables where then can be retrieved from the table using hadoop. Hence this part focuses mainly on translating AQL queries into SQL queries which can be used to retrieve the data using hadoop.

As mentioned in the first chapter, the whole system consists of two parts:

- Translating AQL queries into SQL queries and
- Implementing those SQL queries in sort-merge join using hadoop

The main focus of this thesis work is based on first part i.e. to parse the AQL queries into SQL queries. The data which is in the form of XML file will be transformed into the format of dewey coding and this will be stored in different files. These files are considered as

tables and queried as SQL. The AQL query which is transformed into SQL will be implemented over Sort-Merge join instead of normal relational database.

Here is an example how the input AQL query and the corresponding output SQL query looks like:

AQL QUERY

```
➔ select    e/ehr_id/value as ehr_id
  from      Ehr e
  contains  composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
  contains  observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
  where     EXISTS  obs/data[at0001]/events[at0002]/data[at0003]/items[at0010]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2)
in (select substring_index(P699.dewey_id, '.', 2) from P699)
```

AQL queries are defined in the form of XPath whereas in SQL queries, the data are retrieved from the tables consisting of rows and columns. So, we need to find all the paths and its values, and consider a single path containing a value to be a single table. For example, if we take the path from select expression *'e/ehr_id/value'* it is taken from the path number P5. Likewise, there are 1819 paths starting from P0 to P1819. 'P' is short for Path. Also, there are 806 paths containing with its values starting from PV0 to PV806. 'PV' is short for PathValue. All the paths are identified through dewey_id which is indexed already. In short or to make it easy to understand, the collective Paths are created in *0index.txt* and PathValues are created in *0indexValue.txt*. Some elements contains a version type like Composition and Observation. For this kind of versions, *0type.txt* is created to see in which type number the Composition is presented. So there are 23 types of version starting from T0 to T23.

So here if AQL query has a path, we look up into *0index.txt* to identify the path number. For example, if we find the path number say P5, then to consider that path as a table, we look up into P5 file. Note that, the paths contain all the dewey_id's of that corresponding tree structure. Similarly, if AQL query has a path and its value, then we look up into *0indexValue.txt* to identify that corresponding pathValue number. If we find the pathValue

in PV101, then we consider that path (dewey_id) and its corresponding value as PV101 table.

Now we see how the queries are translated from AQL to SQL language.

3.2 AQL to SQL QUERY EXAMPLE 1

Consider the example below:

➔ select e/ehr_id/value as ehr_id
from Ehr e
contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0010]/
null_flavour

For the following AQL query, the corresponding AST will look like below:

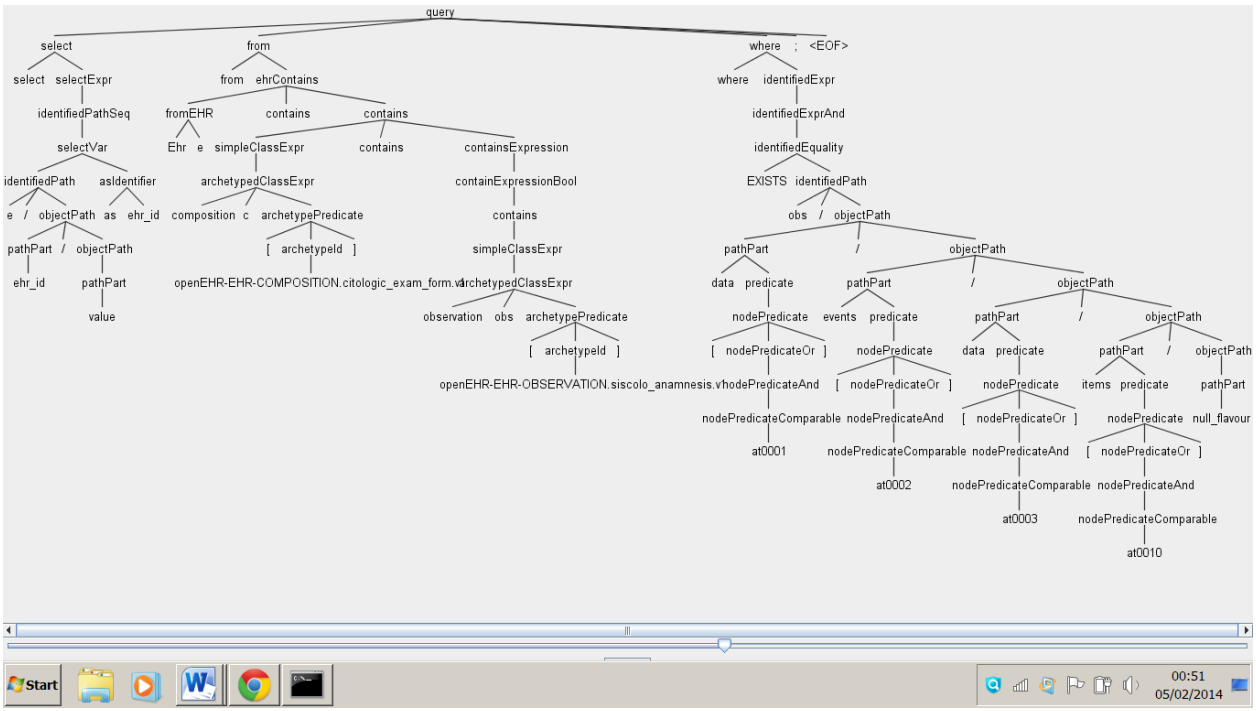


Figure 3.2

All the SELECT, FROM, WHERE, AND, OR clauses are same as SQL query. So wherever these clauses appear in AQL query will be replaced as such. So for the above AQL query we replace these clauses as such for SQL query.

SELECT… FROM… WHERE…

Next comes the *Identified path* (e/ehr_id/value) which is in between SELECT and WHERE. As we already discussed, basically the Paths in openEHR are defined in XPath compatible syntax. Hence the AQL queries are in the form similar like XPath. So we need to find this path which is indexed already and stored in separate files. The location of these files have to be found using these paths and assume like SQL table. So the table name of this path (e/ehr_id/value) is P5 and the table looks like example below:

dewey_id
0.0.1.0
0.1.1.0
0.2.1.0
0.3.1.0
0.4.1.0
0.5.1.0
0.6.1.0
0.1041.1.0
0.1777.1.0
0.1817.1.0
0.2024.1.0
0.2128.1.0
0.2459.1.0
0.2716.1.0
0.3059.1.0

Table P5

Consider the above dewey_code attribute as 'Strings'. Since we need to neglect the dewey_code column from 2nd delimiter (.) i.e we need only the first 1st delimiter (.) followed by a digit, so we use substring_index() function. Now we can write the above path (e/ehr_id/value) as

➔ Subtring_index(P5.dewey_id, '.', 2)

Next comes *naming the retrieved results* (AS ehr_id). This is same like SQL where AQL allows users to rename the returned items specified in SELECT clause. AS is a keyword which is followed by the specified name ehr_id. Since this is optional, we may or may not use this.

So we proceed with the next step which is the *class expression* which is in between FROM clause and Containment (Ehr e). Class expression consists of RM class name, optional AQL variable and optional Predicate. So the corresponding table for Ehr is P5 and e is a Variable which is optional.

So the query becomes,

➔ SELECT Subtring_index(P5.dewey_id, '.', 2) FROM P5

Next comes the *Containment expression* with the keyword CONTAINS. Here, the containment COMPOSITION is the parent of containment OBSERVATION. So we need to find the WHERE condition path along with the containment expression. The EXISTS path obs/data[at0001]/events[at0002]/data[at0003]/items[at0010]/null_flavour along with containment expression comes under P699 table. Table P699 looks like below:

dewey_id
0.0.7.3.3.7.5.2.2.5.1
0.0.8.3.3.7.5.2.2.5.1
0.0.9.3.3.7.5.2.2.5.1
0.0.10.3.3.7.5.2.2.5.1
0.0.12.3.3.7.5.2.2.5.1
0.0.13.3.3.7.5.2.2.5.1
0.0.14.3.3.7.5.2.2.5.1
0.1.8.3.3.7.5.2.2.5.1

Table P699

Now the condition is whether this Path exists along with the containment expression. If that is true, then retrieve all the common items from the table P5 in which the dewey_code columns are selected from the table P699.

So now the query becomes,

➔ select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P699.dewey_id, '.', 2) from P699)

This is the corresponding output query to the given AQL query and the corresponding output table is shown below:

substring_index(P5.dewey_id, '.', 2)
0.0
0.1

Output table

3.3 AQL to SQL QUERY EXAMPLE 2

Consider the next example below,

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value = 'true'
```

For the above AQL query, the corresponding AST will look like figure 3.2 below:

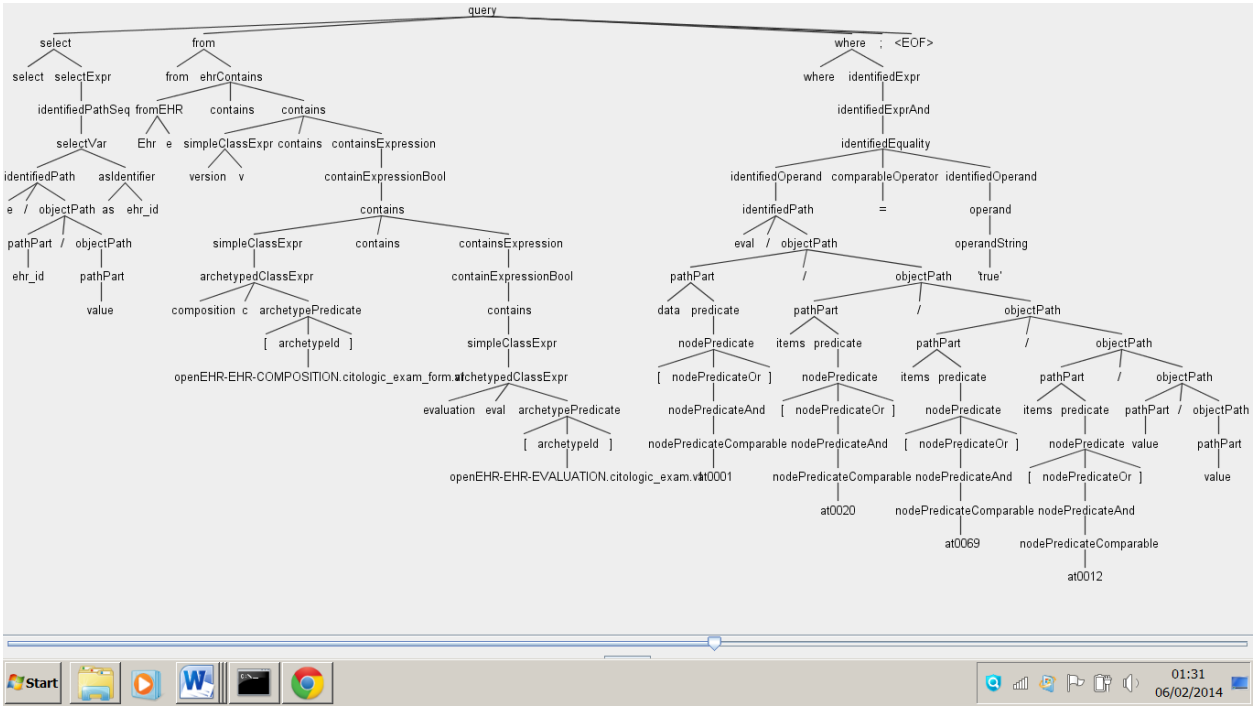


Figure 3.3

Now, replace the SELECT…FROM…WHERE as such.

Next comes with the *Identified path* (e/ehr_id/value) which is in between SELECT and WHERE. The table name of this path (e/ehr_id/value) is P5.

Now we can write the above path (e/ehr_id/value) as

```
Subtring_index(P5.dewey_id, '.', 2)
```

Next comes *naming the retrieved results* (AS ehr_id). So we proceed with the next step which is the *class expression* which is in between FROM clause and Containment (Ehr e). The corresponding table for Ehr is P5. So the query becomes,

➔ SELECT Subtring_index(P5.dewey_id, '.', 2) FROM P5

Next comes the *Containment expression*. Version v is the Parent of Composition c and Composition c is the Parent of Evaluation eval. Find the WHERE condition path along with the containment expression. Here, the given Path is different from the previous EXISTS path. There is a left operand, operator and right operand. Left operand is the Path i.e., eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value, Operator is '=' and the Right operand is Value which is 'true'. Now find this path eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value along with containment expression and this comes under PV500 table. Since the query contains the path with the value (Right operand which is 'true' in this query), we consider Path and Value table combined. Now the condition is whether this Path contains with the value 'true'. If that is true, then retrieve all the attributes of P5 and PV500. For this, we join these two tables and retrieve the common items from both these tables with the condition PV500.status='true'.

Example of both these tables are shown below:

dewey_id	**status**
0.74.7.3.3.8.5.2.2.1.1.0	false
0.238.7.3.3.8.5.2.2.1.1.0	true
0.390.7.3.3.8.5.2.2.1.1.0	false
0.421.6.3.3.8.5.2.2.1.1.0	false
0.461.7.3.3.8.5.2.2.1.1.0	false
0.533.9.3.3.8.5.2.2.1.1.0	false
0.751.7.3.3.8.5.2.2.1.1.0	false
0.798.6.3.3.8.5.2.2.1.1.0	false
0.907.6.3.3.8.5.2.2.1.1.0	false
0.1114.7.3.3.8.5.2.2.1.1.0	true
0.1201.7.3.3.8.5.2.2.1.1.0	false
0.1382.9.3.3.8.5.2.2.1.1.0	false

Table PV500

dewey_id
0.0.1.0
0.6.1.0
0.7.1.0
0.18.1.0
0.24.1.0
0.36.1.0
0.53.1.0
0.61.1.0
0.1777.1.0
0.1817.1.0
0.3059.1.0
0.238.1.0

Table P5

So now the query becomes,

➔ select substring_index(P5.dewey_id, '.', 2), substring_index(PV500.dewey_id, '.', 2), PV500.status from P5 join PV500 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV500.dewey_id, '.', 2) where PV500.status = ‘true’;

This is the corresponding output query to the given AQL query and the corresponding output table is shown below:

substring_index (P5.dewey_id, ‘.’, 2)	**substring_index (P5.dewey_id, ‘.’, 2)**	**PV500.status**
0.238	0.238	True

Output table

Chapter 4

Implementation And Testing

In this chapter, the implementation and testing with sample queries are described.

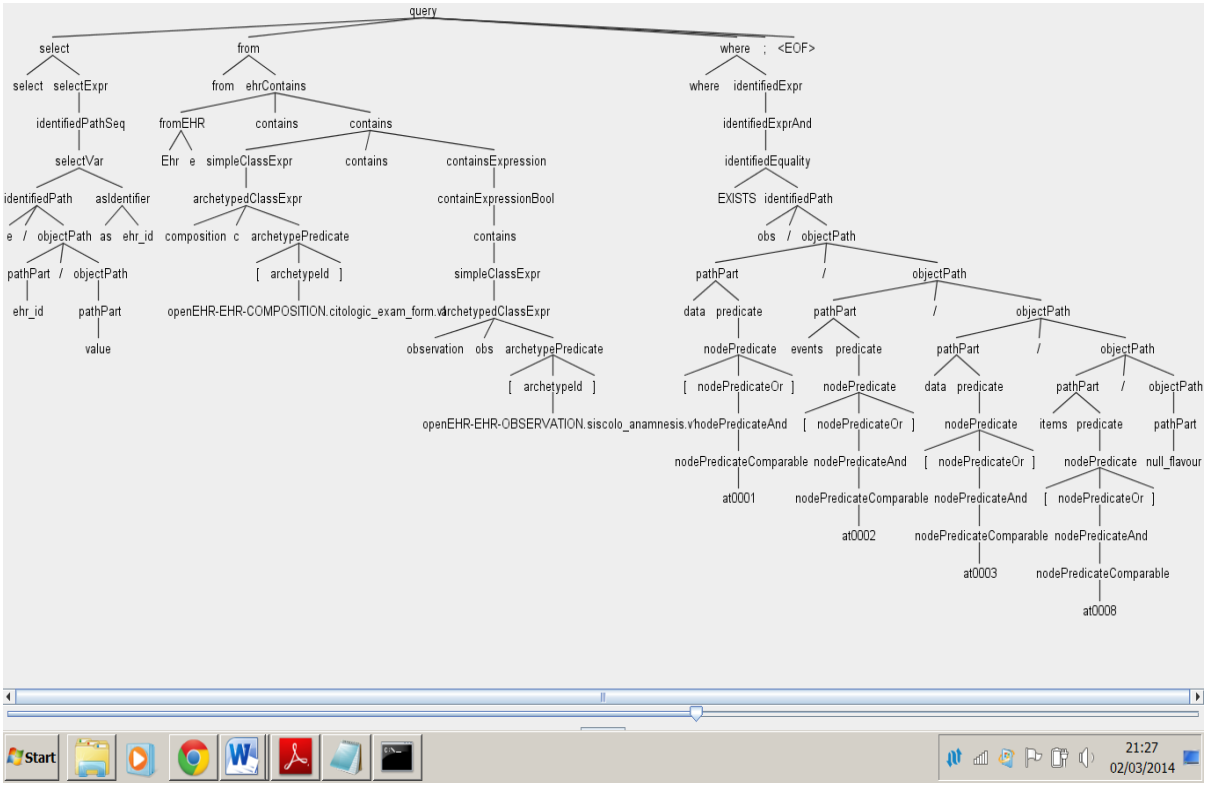
4.1 IMPLEMENTATION

Generating AST

After constructing the grammar, run the command in command prompt window,

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Purani>cd..

C:\Users>cd..

C:\>cd javalib

C:\Javalib>cd tmp1

C:\Javalib\tmp1>antlr4 Aql.g4

C:\Javalib\tmp1>java org.antlr.v4.Tool Aql.g4

C:\Javalib\tmp1>javac Aql*.java

C:\Javalib\tmp1>grun Aql query -gui

C:\Javalib\tmp1>java org.antlr.v4.runtime.misc.TestRig Aql query -gui
select                e/ehr_id/value as ehr_id
   from               Ehr e
   contains           composition c[openEHR-EHR-COMPOSITION.citologic_exam_for
m.v1]
   contains           observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesi
s.v1]
   where                    EXISTS    obs/data[at0001]/events[at0002]/data[a
t0003]/items[at0008]/null_flavour
;
^Z
```

‘^Z’ is entered after the line semicolon ‘;’ and then the enter key is pressed. Now the Abstract Syntax Tree (AST) tree or parse tree dialog box that pops up like shown below:



ANTLR generated files

Here, we are going to manipulate the trees using tree grammar actions. From the grammar Aql.g4, ANTLR generates lots of files with an extension .java, .tokens, .class such as AqlParser.java, AqlLexer.java, AqlBaseListener.java, etc., and the explanations are as follows:

- *AqlParser.java:* This file contains the parser class definition which is specific to Aql grammar that recognizes the language syntax. Also it contains a method for each rule in the grammar.

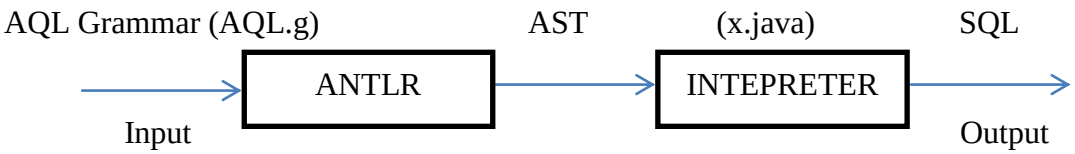
- *AqlLexer.java:* This file contains the lexer class definition by which ANTLR generated this by analyzing each and every lexical rules and literals in the grammar. The lexer tokenizes the input breaking up into vocabulary symbols.

- *Aql.tokens:* This file contains token type number to each token that we defined. These values are needed when we split larger grammars to multiple small grammars so that ANTLR can synchronize all the token type numbers.

- *AqlListener.java and AqlBaseListener.java:* By default, AST is automatically built up by ANTLR parsers. ANTLR has a ParseTreeWalker that knows how to walk these parse trees and triggers events or callbacks in listener implementation object that we create. AqlListener is the interface where we have enter and exit method for each rule in the parse grammar. AqlBaseListener is a set of empty implementations of all listener interface methods. We build listener by subclassing this base and can override the methods by calling '@Override'.

Translation part

Our next step is to translate the given query (AQL) to SQL query. So, for language translation, Abstract Syntax Tree is given as input to the interpreter that we implement as Java programming language and we are going to call some of the methods in a subclass of AqlBaseListener.



Translate.java

Translate.java is a file (added on Appendix) that we are going to create for integrating a generated parser into java program. The main() method in this file invokes initializer parser and prints out the parse tree. It also represents the overall data flow of the recognizer. The each and every line code with its explanation is as follows:

ANTLRInputStream input = new ANTLRInputStream(System.in);

- The above line creates an InputStream of characters for the lexer that is used to read from the standard input.

AqlLexer lexer = new AqlLexer(input);

CommonTokenStream tokens = new CommonTokenStream(lexer);
AqlParser parser = new AqlParser(tokens);

- Next we create the lexer and parser objects derived from the Aql grammar file and a TokenStream Pipes between them.

ParseTree tree = parser.query();

- This line launches the parser which parses beginning at query rule.

ParseTreeWalker walker = new ParseTreeWalker();

- This line creates the ParseTreeWalker that can trigger events or callbacks.

walker.walk(new AqlToSql(), tree);

- This line knows how to walk these parse trees during the parse and triggers AqlToSql() events or callbacks in listener implementation objects. Also AqlToSql method inherits or extends from AqlBaseListener, override the methods inherited from AqlBaseListener class in order to do parsing

System.out.println();

- This line is used to print the statement after any modification (Translation) in AqlToSql file.

Finally, we are going to invoke the listener implementation method (AqlBaseListener.java which has been added on Appendix) by creating AqlToSql.java file. This is the most important step where we are going to translate the fetched token to corresponding SQL query.

AqlToSql.java

We are going to invoke a few methods in a subclass of AqlBaseListener to write a program that reacts according to the input. Each listener method prints out the corresponding output for a given piece of input when called do so by TreeWalker. Now we have to figure it out how the output should be for the corresponding input and which methods we have to invoke. For example,

Input_1 (AQL Query)

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0008]/null_flavour
```

Output_1 (SQL Query)

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in
(select substring_index(P681.dewey_id, '.', 2) from P681)
```

Here, we need select() method, from() method, where() method, contains() method, etc. To write a program, AST is an important one where it helps us to identify which is the root, which is the parent and which are all the child of that parent, etc.

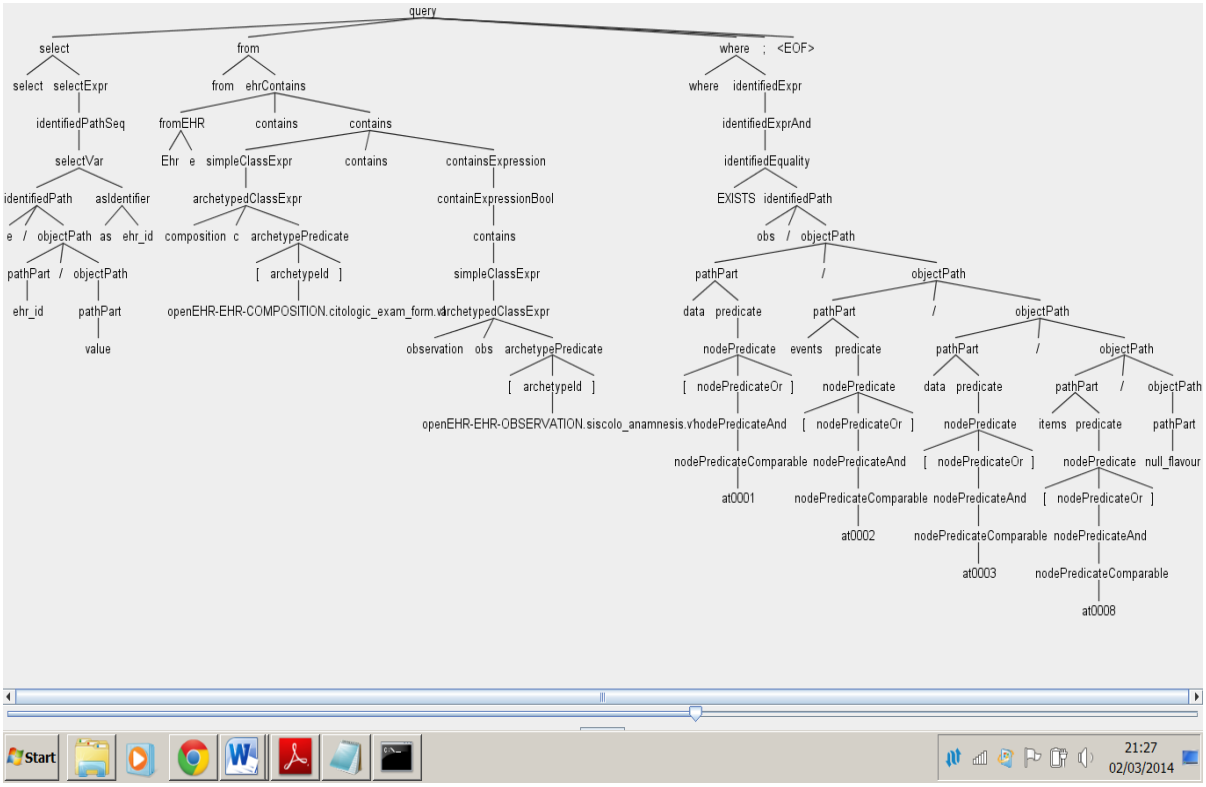


Figure: Input_AST_1

Now we start from the root query the first child is taken to be select node. Select node has two children; one is select clause and the other one is selectExpr. So at the start of select node, we have to print “select” clause as such and hence we choose enterSelect() method in AqlBaseListener.java file. Create a string selectStatement and just call to get the text of select.

```
@Override
public void enterSelect(AqlParser.SelectContext ctx)
{
	selectStatement += (ctx.SELECT().getText()+"\t");
}
```

By default, all the nodes in a parse tree are visited by a listener or simply we can say that the entire parse tree is walked. To create a listener, we simply create a java class that extends AqlBaseListener class. To specify the code that should run upon visiting a rule in the grammar, override the appropriate method found in the following class AqlBaseListener.java. In my AqlToSql class, I have overridden several of the base methods. To capture the first parse tree node in a method declaration within Aql program (this may visit the node again if that method is called during interpretation), I overrode the enterSelect method found in my BaseListener.

Listener allows us to translate two methods per rule in our grammar ie. enter<RuleId> and exit<RuleId> where their puposes are self-explantory. When ANTLR reaches the first token method matched to a rule, the “enter” method is called for that rule and when it reaches the last token method matched to the rule, then the corresponding “exit” method is called for that rule.

Here, we simply inherit from the base class and override the enterSelect(AqlParser.SelectContext ctx) method. This allows us to capture every time the parser visits this ‘select’ rule. ctx.SELECT() just asks the context object for the SELECT token and prints out the text by getText(). Context objects just records everything that happens during the recognition of a rule.

For example:

```
public class AqlBaseListener implements AqlListener {
	public void enterSelect() {
	}
}
public class AqlToSql extends AqlBaseListener {
	@override
	public void enterSelect() {
	}
}
```

Class AqlBaseListener represents the superclass and implements a method call enterSelect(). The subclass called AqlToSql inherits all the methods that could be in the AqlBaseListener class. However, the class AqlToSql overrides the method enterSelect(), replacing its functionality from AqlBaseListener.

Next comes the statement “e/ehr_id/value” as a children of identifiedpath node and “as ehr_id” as a children of asidentifier node. We have to select everything together as “e/ehr_id/value as ehr_id” which is a children of selectvar node. So choose enterSelectVar() method from listener file.

```
@Override
public void enterSelectVar(AqlParser.SelectVarContext ctx)
{
	if(ctx.identifiedPath() != null)	{
		String selectPath = ctx.identifiedPath().getText();
		if (selectPath.equals("e/ehr_id/value"))	{
			selectStatement += ("substring(P5.dewey_id, '.', 2)");
		}
	}
}
```

Here, we have to check the condition whether identifiedpath contains this path “e/ehr_id/value” then we need to substitute the corresponding output “substring_index(P5.dewey_id, '.', 2)”. Hence, we add this statement next to the “select” in the string selectStatement. Since naming the retrieved results (AS ehr_id) is optional at present we can print empty string to it. So add this statement above the method enterSelectVar():

```
@Override
public void enterSelectVar(AqlParser.SelectVarContext ctx)
{
	if(ctx.identifiedPath() != null)	{
		String selectPath = ctx.identifiedPath().getText();
		if (selectPath.equals("e/ehr_id/value"))	{
			selectStatement += ("substring(P5.dewey_id, '.', 2)");
		}
	}
	if(ctx.asIdentifier() != null) {
		selectStatement += ("");
	}
}
```

Next begins with from node and its childrens are from and ehrcontains. We need to just print “from” as such like select clause and hence we choose enterFrom() method. So create a string fromStatement and just call to get the text of “from” with getText() method.

```
@Override
```

```
public void enterFrom(AqlParser.FromContext ctx)
{
	fromStatement += ("\n" + ctx.FROM().getText()+"\t");
}
```

ehrcontains has three childrens; first one is fromEHR, second one is contains and the third one is contains. fromEHR has childrens “ehr e” and if that contains in the tree, just print “P5” which is the path of it. Since it comes under the root of from node it is just added to the fromStatement after the from clause.

```
@Override
public void enterEhrContains(AqlParser.EhrContainsContext ctx)
{
	if(ctx.fromEHR() != null)   {
		String fromehr = ctx.fromEHR().getText();
		if (fromehr.equals("Ehre") || fromehr.equals("EHRe") || fromehr.equals("ehre"))    {
			fromStatement += ("P5\t");
		}
	}
}
```

Next comes the *Containment expression* with the keyword CONTAINS in AQL statement. Here, the containment COMPOSITION is the parent of containment OBSERVATION. So we need to find the WHERE condition path along with the containment expression. The EXISTS path obs/data[at0001]/events[at0002]/data[at0003]/items[at0008]/null_flavour along with containment expression comes under P681 table. Just print the where clause which comes under the node where and choose enterWhere() method to print it in the string whereStatement.

```
@Override
public void enterWhere(AqlParser.WhereContext ctx)
{
	whereStatement += ("\n"+ ctx.WHERE().getText()+"\t");
}
```

Now the condition is whether this Path “obs/data[at0001]/events[at0002]/data[at0003]/ items[at0008]/null_flavour” exists along with the containment expression. If that is true, then retrieve all the common items from the table P5 in which the dewey_code columns are selected from the table P681.

So now the query becomes,

- ➔ select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P681.dewey_id, '.', 2) from P681)

Now we know that for this corresponding path P681 is the table and to retrieve all the common items from the table P5, we need to change the input query as "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P681.dewey_id, '.', 2) from P681)" and add this statement at the end of whereStatement.

For this kind of paths to translate to the corresponding output, we create a HashMap which means that if we have a key, we can lookup a value using that key. It uses a hash of the key for lookups.

Example: Map<String, String> map = new HashMap<String, String>();

```
map.put("keyone", "valueone");
map.put("keytwo", "valuetwo");
System.out.println(map.get("keytwo"));
```

Then the output will be 'valuetwo'.

The HashMap is a data structure which is based on hashing. This allows us to store the object as key,value pair. If we know the key, we can retrieve the object on constant time. It implements map interface which maps a unique key to the corresponding value. Given a key and a value, we can store the value in Map object. Once the value is stored we can retieve it by Map method. So we create a method called AqlToSql() and add a value corresponding to that path (key).

```
public AqlToSql()
{
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0008]/null_flavour","substring_ind
ex(P5.dewey_id, '.', 2) in (select substring_index(P681.dewey_id, '.', 2) from P681)");
}
```

So when we see the path 'obs/data[at0001]/events[at0002]/data[at0003]/items[at0008]/ null_flavour' in AST, it comes under the parent node of identifiedPath. So we choose enterIdentifiedEquality() method to check whether identifiedPath()!=null. If that's true, create a string path and get the whole text/path which maches in the map interface. Retrieve its value and add the string of a statement in whereStatement.

```
@Override
public void enterIdentifiedEquality(AqlParser.IdentifiedEqualityContext ctx)
{
        if(ctx.identifiedPath() != null) {
                String path = (map.get(ctx.identifiedPath().getText())+"\t");
```

```
                whereStatement += path;
        }
}
```

Now if we run this program then the output will be as follows:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
C:\Users\Purani>cd..
C:\Users>cd..
C:\>cd javalib
C:\Javalib>cd tmp1
C:\Javalib\tmp1>javac Aql*.java Translate.java
C:\Javalib\tmp1>java Translate
select                e/ehr_id/value as ehr_id
   from               Ehr e
   contains           composition c[openEHR-EHR-
COMPOSITION.citologic_exam_for
m.v1]
   contains           observation obs[openEHR-EHR-
OBSERVATION.siscolo_anamnesi
s.v1]
   where                    EXISTS    obs/data[at0001]/events[at0002]/data[a
t0003]/items[at0008]/null_flavour
;
^Z
select  substring(P5.dewey_id, '.', 2)
from    P5
where   substring_index(P5.dewey_id, '.', 2) in (select substring_index(P681.dew
ey_id, '.', 2) from P681)

C:\Javalib\tmp1>
```

Figure: Output_1

Input_2 (AQL Query)

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value = 'true'
```

Output_2 (SQL Query)

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV500.dewey_id, '.', 2),
PV500.status from P5 join PV500 on substring_index(P5.dewey_id, '.', 2) =
substring_index(PV500.dewey_id, '.', 2) where PV500.status = 'true';
```

Input_2 looks similar to the input_1 which we executed before but the only last statement differs. The change is that, in Input_1, the last statement exists with the keyword EXISTS followed by the Identified Paths. But in Input_2, it exists with standard predicate. Since here exists with a value in right operand, we need to find the path and value table in the record. This is mentioned simply as "PVtablename" where P represents the Path and V represents the Value.

So, we should join P5 table with this path "eval/data[at0001]/items[at0020]/items[at0069]/ items[at0012]/value/value = 'true'" to retrieve the whole table (dewey_id as column1 and status as column2). Hence the corresponding output will be output_2.

To write a program, just we look up into AST as below:

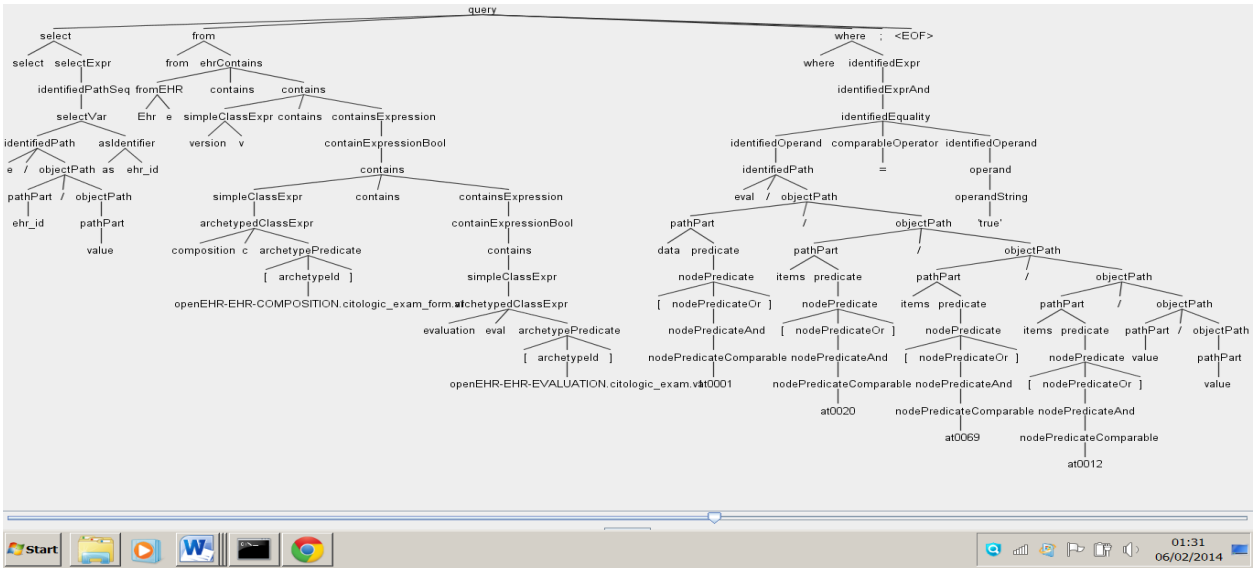


Figure: Input_AST_2

Now we start from the root query. The first child is taken to be select node. If we compare the select statement with the first query, the structure is same. Hence we write the code as follows:

```
@Override
public void enterSelect(AqlParser.SelectContext ctx)
{
      selectStatement += (ctx.SELECT().getText()+"\t");
}

@Override
public void enterSelectVar(AqlParser.SelectVarContext ctx)
{
```

```
        if(ctx.identifiedPath() != null)    {
                String selectPath = ctx.identifiedPath().getText();
                if (selectPath.equals("e/ehr_id/value"))   {
                        selectStatement += ("substring(P5.dewey_id, '.', 2)");
                }
        }
        if(ctx.asIdentifier() != null) {
                selectStatement += ("");
        }
    }
```

Next begins with from node and enterFrom() method is written as same as first query.

```
@Override
public void enterFrom(AqlParser.FromContext ctx)
{
        fromStatement += ("\n" + ctx.FROM().getText()+"\t");
}
```

enterEhrContains() method will also be same:

```
@Override
public void enterEhrContains(AqlParser.EhrContainsContext ctx)
{
        stack.push("ehrContains");
        if(ctx.fromEHR() != null)   {
                String fromehr = ctx.fromEHR().getText();
                if (fromehr.equals("Ehre") || fromehr.equals("EHRe") || fromehr.equals("ehre"))   {
                        fromStatement += ("P5\t");
                }
        }
}
```

Next comes the *Containment expression*. So, Find the WHERE condition path along with the containment expression. Here, the given Path is different from the previous EXISTS path. There is a left operand, operator and right operand. Left operand is the Path i.e., eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value, Operator is '=' and the Right operand is Value which is 'true'. Now find this path eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value along with containment expression and this comes under PV500 table. Since the query contains the path with the value (Right operand which is 'true' in this query), we consider Path and Value table combined. Now the condition is whether this Path contains with the value 'true'. If that is true, then retrieve all the attributes of P5 and PV500. For this, we join these two tables and retrieve the common items from both these tables with the condition PV500.status='true'.

In enterWhere() method where clause will be added to whereStatement.

```
@Override
public void enterWhere(AqlParser.WhereContext ctx)
{
        whereStatement += ("\n"+ ctx.WHERE().getText()+"\t");
}
```

So now the query becomes,

➔ select substring_index(P5.dewey_id, '.', 2) from P5
where

Now we know what is the condition to check in where statement. The corresponding output table for the given path is PV500 and we need to check and retrieve the column2 which has the status as ‘true’. So we use HashMap to retrieve the value by storing its key.

In AqlToSql() method, simply add a value corresponding to that path (key).

```
public AqlToSql()
{
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0008]/null_flavour","substring_ind
ex(P5.dewey_id, '.', 2) in (select substring_index(P681.dewey_id, '.', 2) from P681)");
map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value","PV500.status"
);
}
```

So when we see the path
‘eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/
value’ in AST, it comes under the parent node of identifiedPath. So we choose enterIdentifiedOperand() method to check whether identifiedPath()!=null. If that’s true, create a string path and retrieve its value based on whole text/path as key which maches in the map interface.

```
@Override
public void enterIdentifiedOperand(AqlParser.IdentifiedOperandContext ctx)
{
        if(ctx.identifiedPath() != null) {
                String path = (map.get(ctx.identifiedPath().getText())+"\t");
        }
}
```

We retrieved the value PV500.status and stored in String path. Now to get only the table name PV500 from the path's string, we add some two lines to the enterIdentifiedOperand() method which looks like below:

```
@Override
public void enterIdentifiedOperand(AqlParser.IdentifiedOperandContext ctx)
{
	if(ctx.identifiedPath() != null) {
		String path = (map.get(ctx.identifiedPath().getText())+"\t");
	}
	whereStatement += path + "\t";
	path = path.substring(0,path.indexOf('.'));
}
```

Now just add all the Strings in whereStatement and selectSattement as below:

```
@Override
public void enterIdentifiedOperand(AqlParser.IdentifiedOperandContext ctx)
{
	if(ctx.identifiedPath() != null) {
		String path = (map.get(ctx.identifiedPath().getText())+"\t");
	}
	whereStatement += path + "\t";
	path = path.substring(0,path.indexOf('.'));

	fromStatement +=  (" join " + path + " on substring_index(P5.dewey_id, \'.\', 2) =
				substring_index(" + path + ".dewey_id, '.', 2)\t");

	selectStatement +=  (", substring (" + path + ".dewey_id, '.', 2), " + spath);
}
```

So now the query becomes,

➔ select substring_index(P5.dewey_id, '.', 2), substring_index(PV500.dewey_id, '.', 2), PV500.status from P5 join PV500 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV500.dewey_id, '.', 2) where PV500.status

To print "=" and "true" to the query, find its parent node in AST and just add it to the whereStatement.

```
@Override
public void enterComparableOperator(AqlParser.ComparableOperatorContext ctx)  {
	whereStatement += (ctx.getText()+"\t");   }
@Override
public void enterOperand(AqlParser.OperandContext ctx)
{
	if(ctx.operandString() != null)
	{
		whereStatement += (ctx.operandString().getText());
	}
```

}
Now if we run this program then the output will be as like “figure:Output_2” which is shown below.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Purani>cd..

C:\Users>cd..

C:\>cd javalib

C:\Javalib>cd tmp1

C:\Javalib\tmp1>javac Aql*.java Translate.java

C:\Javalib\tmp1>java Translate
select                e/ehr_id/value as ehr_id
   from               Ehr e
   contains           composition c[openEHR-EHR-
COMPOSITION.citologic_exam_for
m.v1]
   contains           observation obs[openEHR-EHR-
OBSERVATION.siscolo_anamnesi
s.v1]
   where                    EXISTS    obs/data[at0001]/events[at0002]/data[a
t0003]/items[at0008]/null_flavour
;
^Z
select  substring(P5.dewey_id, '.', 2)
from    P5
where   substring_index(P5.dewey_id, '.', 2) in (select substring_index(P681.dew
ey_id, '.', 2) from P681)

C:\Javalib\tmp1>
```

Figure: Output_2

Input_3 (AQL Query)

```
Select      e/ehr_id/value as ehr_id
From        Ehr e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains  observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where    (EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/
            items[at0034] OR
            EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/
```

items[at0035]) AND
c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00' AND
c/context/start_time/value < '2006-05-01T00:00:00,000+01:00'

Output_3 (SQL Query)

```
select  substring_index(P5.dewey_id, '.', 2), substring_index(PV101.dewey_id, '.', 2),
PV101.date_time_zone
from    P5 join PV101 on substring_index(P5.dewey_id, '.', 2)=substring_index(PV101.dewey_id,
'.', 2)
where   (substring_index(P5.dewey_id, '.', 2) in (select substring_index(P343.dewey_id, '.', 2) from
        P343)       OR
        substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1271.dewey_id, '.', 2) from
        P1271))    AND   (PV101.date_time_zone between '2006-01-01T00:00:00,000+01:00'
        AND '2006-05-01T00:00:00,000+01:00');
```

The AST of Input_3 will be as “figure: Input_AST_3” which is shown below.

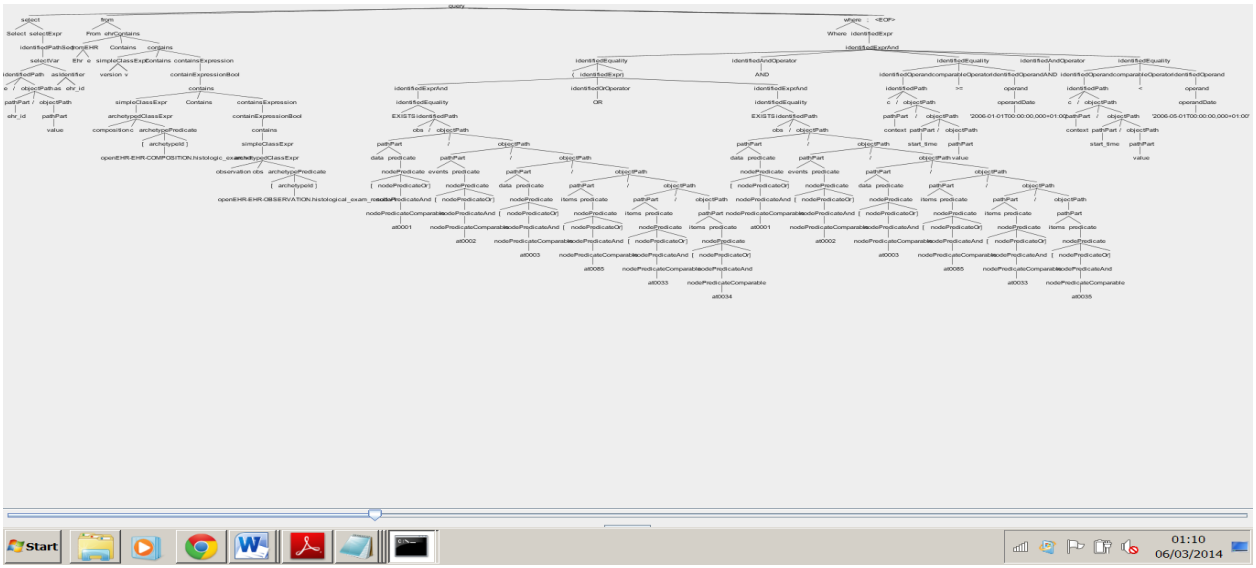


Figure: Input_AST_3

Just compare and add all the code we know from previous examples. In addition, the codes we need to add for this example is AND and OR operators, Date in place of String in previous example. The corresponding output is shown in “figure: Output_3”.

```
C:\Javalib\tmp1>javac Aql*.java Translate.java

C:\Javalib\tmp1>java Translate
Select      e/ehr_id/value as ehr_id
From        Ehr e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains  observation obs[openEHR-EHR-
OBSERVATION.histological_exam_result.v1]
Where     (EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/ite
ms[at0033]/
    items[at0034] OR
    EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at00
33]/
    items[at0035]) AND
    c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00'  AND
    c/context/start_time/value < '2006-05-01T00:00:00,000+01:00'
;
^Z
Select  substring(P5.dewey_id, '.', 2), substring (PV101.dewey_id, '.', 2), PV10
1.date_time_zone
From    P5        join PV101 on substring_index(P5.dewey_id, '.', 2) = substring
_index(PV101.dewey_id, '.', 2)
Where   (substring_index(P5.dewey_id, '.', 2) in (select substring_index(P343.de
wey_id, '.', 2) from P343)       OR
substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1271.dewey_id,
'.', 2) from P1271)      )      AND
PV101.date_time_zone    >=      '2006-01-01T00:00:00,000+01:00' AND
PV101.date_time_zone    <       '2006-05-01T00:00:00,000+01:00'
```

Figure: Output_3

Input_4 (AQL Query)

```
Select      e/ehr_id/value as ehr_id
From        Ehr e
Contains            version v
Contains            composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains            observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
          ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/
value/defining_code/code_string matches {‘at0049’, ‘at0050’, ‘at0051’, ‘at0052’, ‘at0053’} AND
c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00')  AND c/context/start_time/value
< '2006-05-01T00:00:00,000+01:00')
```

Output_4 (SQL Query)

```
Select  substring(P5.dewey_id, '.', 2), substring (PV160.dewey_id, '.', 2), PV160.atid, substring
(PV101.dewey_id, '.', 2), PV101.date_time_zone
From    P5        join PV160 on substring_index(P5.dewey_id, '.', 2) =
substring_index(PV160.dewey_id, '.', 2)    join PV101 on substring_index(P5.dewey_id, '.',  2) =
substring_index(PV101.dewey_id, '.', 2)
```

Where ((PV160.atid IN ('at0049', 'at0050', 'at0051', 'at0052', 'at0053') AND PV101.date_time_zone >= '2006-01-01T00:00:00,000+01:00') AND PV101.date_time_zone < '2006-05-01T00:00:00,000+01:00')

AST will look like below:

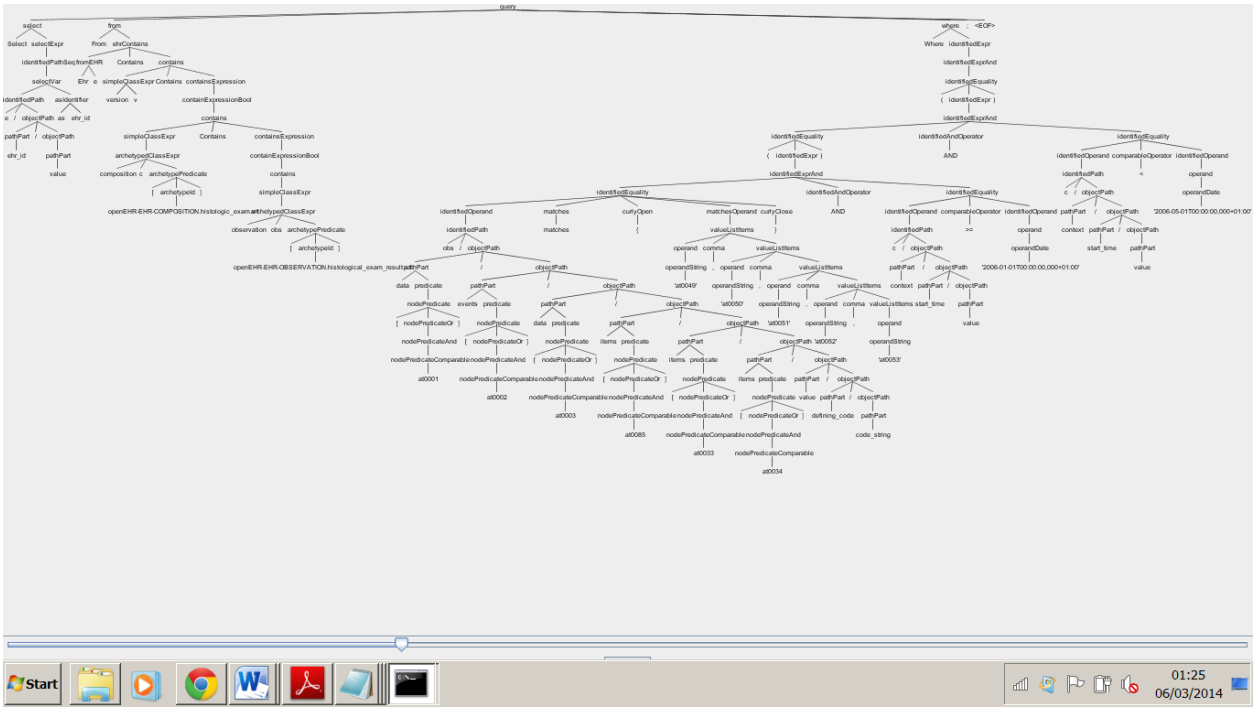


Figure: Input_AST_4

compare and add all the code we know from previous examples. In addition, the only code we need to add for this example is matches{……}. The corresponding output is shown in "figure: Output_4".

```
C:\Javalib\tmp1>java Translate
Select    e/ehr_id/value as ehr_id
From      Ehr e
Contains          version v
Contains          composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains          observation obs[openEHR-EHR-
OBSERVATION.histological_exam_resu
lt.v1]
Where   ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at003
3]/items[at0034]/value/defining_code/code_string matches {'at0049', 'at0050', 'a
t0051', 'at0052', 'at0053'} AND c/context/start_time/value >= '2006-01-01T00:00:
00,000+01:00')  AND c/context/start_time/value < '2006-05-
01T00:00:00,000+01:00'
)
;
^Z
Select  substring(P5.dewey_id, '.', 2), substring (PV160.dewey_id, '.', 2), PV16
0.atid, substring (PV101.dewey_id, '.', 2), PV101.date_time_zone
From    P5        join PV160 on substring_index(P5.dewey_id, '.', 2) = substring
_index(PV160.dewey_id, '.', 2)    join PV101 on substring_index(P5.dewey_id, '.'
, 2) = substring_index(PV101.dewey_id, '.', 2)
Where   ((PV160.atid    IN      ('at0049',      'at0050',       'at0051',
'at0052',       'at0053')AND
PV101.date_time_zone    >=      '2006-01-01T00:00:00,000+01:00' )       AND
PV101.date_time_zone    <       '2006-05-01T00:00:00,000+01:00' )


C:\Javalib\tmp1>
```

Figure: Output_4

4.2 TESTING

AQL queries are similar to SQL queries. So we used some examples to test this conversion to check whether the AQL queries are properly converting it into SQL queries. Some examples of AQL queries that we have tested during the implementation process are as follows:

EXAMPLE 1

AQL QUERY *(A query example is to return all record ids that had a histologic exam result indicating neoplastic lesions between 2006-01-01 and 2006-05-01).*

```
Select      e/ehr_id/value as ehr_id
From        Ehr e
Contains    version v
Contains    composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains    observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
```

```
Where    (EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]
OR       EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0035])
AND      c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00'  AND
         c/context/start_time/value < '2006-05-01T00:00:00,000+01:00'
```

AQL PATH

```
Select   e/ehr_id/value          Table P5  as ehr_id
From     Ehr          Table P1  e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where    (EXISTS
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]  P343  OR
         EXISTS
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0035])  P1271  AND
         c/context/start_time/value  PV101  >= '2006-01-01T00:00:00,000+01:00'  AND
         c/context/start_time/value  PV101  < '2006-05-01T00:00:00,000+01:00'
```

SQL QUERY

```
select  substring_index(P5.dewey_id, '.', 2), substring_index(PV101.dewey_id, '.', 2), PV101.date_time_zone

from   P5 join PV101 on substring_index(P5.dewey_id, '.', 2)=substring_index(PV101.dewey_id, '.', 2)

where (substring_index(P5.dewey_id, '.', 2) in (select substring_index(P343.dewey_id, '.', 2) from P343)  OR

     substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1271.dewey_id, '.', 2) from P1271))  AND

 (PV101.date_time_zone between '2006-01-01T00:00:00,000+01:00' AND '2006-05-1T00:00:00,000+01:00');
```

EXAMPLE 2

AQL MAIN QUERY

```
select          e/ehr_id/value as ehr_id
from            Ehr e
Contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where           EXISTS   obs/data[at0001]/events[at0002]/data[at0003]/items[$node]/null_flavour
node in (at0004, at0008, at0009, at0010, at0011, at0012, at0013, at0014, at0015, at0016, at0021)
```

Here in this example, we substitute each nodein values in the place of [$node]. So hence we get 11 queries with solutions.

AQL PATH

```
Select   e/ehr_id/value          Table P5  as ehr_id
From     Ehr          Table P1  e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
Where    EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[$node]/null_flavour
```

node in (at0004 **P1095**, at0008 **P681**, at0009 **P690**, at0010 **P699**, at0011 **P708**, at0012 **P717**, at0013 **P726**, at0014 **P735**, at0015 **P744**, at0016 **P753**, at0021 **P762**)

AQL QUERY 1

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0004]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1095.dewey_id, '.', 2) from P1095)
```

AQL QUERY 2

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0008]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P681.dewey_id, '.', 2) from P681)
```

AQL QUERY 3

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0009]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P690.dewey_id, '.', 2) from P690)
```

AQL QUERY 4

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0010]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P699.dewey_id, '.', 2) from P699)
```

AQL QUERY 5

```
select      e/ehr_id/value as ehr_id
from        Ehr e
```

```
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where           EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0011]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P708.dewey_id, '.', 2) from P708)
```

AQL QUERY 6

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where           EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0012]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P717.dewey_id, '.', 2) from P717)
```

AQL QUERY 7

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where           EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0013]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P726.dewey_id, '.', 2) from P726)
```

AQL QUERY 8

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where           EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0014]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P735.dewey_id, '.', 2) from P735)
```

AQL QUERY 9

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where           EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0015]/null_flavour
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P744.dewey_id, '.', 2) from P744)
```

AQL QUERY 10

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS  obs/data[at0001]/events[at0002]/data[at0003]/items[at0016]/null_flavour
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P753.dewey_id, '.', 2) from P753)

AQL QUERY 11

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS  obs/data[at0001]/events[at0002]/data[at0003]/items[at0021]/null_flavour
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P762.dewey_id, '.', 2) from P762)

EXAMPLE 3

AQL QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
where       EXISTS  obs/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value AND
            EXISTS  obs/data[at0001]/events[at0002]/data[at0003]/items[at0022]/null_flavour
```

AQL PATH

Table P5

```
Select    e/ehr_id/value as ehr_id
```

Table P1

```
From      Ehr e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains  observation obs[openEHR-EHR-OBSERVATION.siscolo_anamnesis.v1]
```

P667

```
Where     EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value AND
```

P1110

```
          EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0022]/null_flavour
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P667.dewey_id, '.', 2) from P667) AND substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1110.dewey_id, '.', 2) from P1110)

EXAMPLE 4

AQL MAIN QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0011]/value/value = $status
```

$status = (true, false)

This query is same as in example 2 but instead of substituting the node values, we substitute the status value which is true and false. So hence we get 2 queries with solutions.

AQL PATH

Table P5

```
Select    e/ehr_id/value as ehr_id
```

Table P1

```
From      Ehr e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains  evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
```

PV360

```
Where     eval/data[at0001]/items[at0020]/items[at0011]/value/value = $status
```

$status = (true, false)

PV360 & root[]/eee:EHR[]/eee:compositions[]/eee:versions[]/v1:data[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]/v1:content[openEHR-EHR-EVALUATION.citologic_exam.v1]/v1:data[at0001]/v1:items[at0020]/v1:items[at0011]/v1:value[]/v1:value[]

AQL QUERY 1

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0011]/value/value = ‘true’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV360.dewey_id, '.', 2), PV360.status from P5
join PV360 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV360.dewey_id, '.', 2) where
PV360.status = ‘true’
```

AQL QUERY 2

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0011]/value/value = ‘false’;
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2), substring_index(PV360.dewey_id, '.', 2), PV360.status from P5 join PV360 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV360.dewey_id, '.', 2) where PV360.status = ‘false’

EXAMPLE 5

AQL MAIN QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0069]/items[$node]/value/value = $status
```

$node = (at0012, at0013, at0014, at0015, at0016, at0017, at0018)
$status = (true, false)

In this example, we substitute both the node values as well as the status values. So hence we get 14 queries with the solutions.

AQL PATH

Select e/ehr_id/value **Table P5** as ehr_id
From Ehr **Table P1** e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
Where eval/data[at0001]/items[at0020]/items[at0069]/items[$node]/value/value = $status

$node = (at0012 **PV500**, at0013 **PV502**, at0014 **PV504**, at0015 **PV506**, at0016 **PV508**, at0017 **PV510**, at0018 **PV512**)
$status = (true, false)

AQL QUERY 1a

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value = ‘true’
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2), substring_index(PV500.dewey_id, '.', 2), PV500.status from P5 join PV500 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV500.dewey_id, '.', 2) where PV500.status = ‘true’;

AQL QUERY 1b

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
```

```
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value = ‘false’
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2), substring_index(PV500.dewey_id, '.', 2), PV500.status from P5 join PV500 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV500.dewey_id, '.', 2) where PV500.status = ‘false’

AQL QUERY 2a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0013]/value/value = ‘true’
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2), substring_index(PV502.dewey_id, '.', 2), PV502.status from P5 join PV502 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV502.dewey_id, '.', 2) where PV502.status = ‘true’

AQL QUERY 2b

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0013]/value/value = ‘false’
```

SQL QUERY

2b) select substring_index(P5.dewey_id, '.', 2), substring_index(PV502.dewey_id, '.', 2), PV502.status from P5 join PV502 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV502.dewey_id, '.', 2) where PV502.status = ‘false’

AQL QUERY 3a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0014]/value/value = ‘true’
```

SQL QUERY

select substring_index(P5.dewey_id, '.', 2), substring_index(PV504.dewey_id, '.', 2), PV504.status from P5 join PV504 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV504.dewey_id, '.', 2) where PV504.status = ‘true’

AQL QUERY 3b

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
```

```
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0014]/value/value = ‘false’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV504.dewey_id, '.', 2), PV504.status from P5 join PV504 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV504.dewey_id, '.', 2) where PV504.status = ‘false’
```

AQL QUERY 4a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0015]/value/value = ‘true’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV506.dewey_id, '.', 2), PV506.status from P5 join PV506 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV506.dewey_id, '.', 2) where PV506.status = ‘true’
```

AQL QUERY 4b

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0015]/value/value = ‘false’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV506.dewey_id, '.', 2), PV506.status from P5 join PV506 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV506.dewey_id, '.', 2) where PV506.status = ‘false’
```

AQL QUERY 5a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0016]/value/value = ‘true’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV508.dewey_id, '.', 2), PV508.status from P5 join PV508 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV508.dewey_id, '.', 2) where PV508.status = ‘true’
```

AQL QUERY 5b

```
select          e/ehr_id/value as ehr_id
from            Ehr e
```

```
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0016]/value/value = ‘false’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV508.dewey_id, '.', 2), PV508.status from P5
join PV508 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV508.dewey_id, '.', 2) where
PV508.status = ‘false’
```

AQL QUERY 6a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0017]/value/value = ‘true’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV510.dewey_id, '.', 2), PV510.status from P5
join PV510 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV510.dewey_id, '.', 2) where
PV510.status = ‘true’;
```

AQL QUERY 6b

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0017]/value/value = ‘false’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV510.dewey_id, '.', 2), PV510.status from P5
join PV510 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV510.dewey_id, '.', 2) where
PV510.status = ‘false’;
```

AQL QUERY 7a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           eval/data[at0001]/items[at0020]/items[at0069]/items[at0018]/value/value = ‘true’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV512.dewey_id, '.', 2), PV512.status from P5
join PV512 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV512.dewey_id, '.', 2) where
PV512.status = ‘true’;
```

AQL QUERY 7b

```
select          e/ehr_id/value as ehr_id
```

```
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       eval/data[at0001]/items[at0020]/items[at0069]/items[at0018]/value/value = ‘false’
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV512.dewey_id, '.', 2), PV512.status from P5
join PV512 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV512.dewey_id, '.', 2) where
PV512.status = ‘false’;
```

EXAMPLE 6

AQL QUERY

```
Select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
contains    observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
where       EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0069]/null_flavour
```

AQL PATH

Table P5

```
Select      e/ehr_id/value as ehr_id
```

Table P1

```
From        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
contains    observation obs[openEHR-EHR-OBSERVATION. histological_exam_result.v1]
```

P1295

```
where       EXISTS    obs/data[at0001]/events[at0002]/data[at0003]/items[at0069]/null_flavour
```

P1295 & root[]/eee:EHR[]/eee:compositions[]/eee:versions[]/v1:data[openEHR-EHR-COMPOSITION.histologic_exam.v1]/v1:content[openEHR-EHR-OBSERVATION.histological_exam_result.v1]/v1:data[at0001]/v1:events[at0002]/v1:data[at0003]/v1:items[at0069]/null_flavour

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2) from P5 where substring_index(P5.dewey_id, '.', 2) in (select
substring_index(P1295.dewey_id, '.', 2) from P1295);
```

EXAMPLE 7

AQL QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
where       (c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00' AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

AQL PATH

Table P5

```
Select      e/ehr_id/value as ehr_id
```

Table P1

```
From        Ehr e
```

Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Where (c/context/start_time/value **PV275** >= '2006-01-01T00:00:00,000+01:00' AND
c/context/start_time/value **PV275** < '2006-05-01T00:00:00,000+01:00')

SQL QUERY

select substring_index(P5.dewey_id, '.', 2), substring_index(PV275.dewey_id, '.', 2), PV275.date_time_zone from P5 join PV275 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV275.dewey_id, '.', 2) where PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00';

EXAMPLE 8

AQL QUERY

select e/ehr_id/value as ehr_id
from Ehr e
contains version v
contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where (((((eval/data[at0001]/items[at0019]/items[at0002]/value/value = ‘true’ OR
eval/data[at0001]/items[at0019]/items[at0003]/value/value = ‘true’) OR
eval/data[at0001]/items[at0019]/items[at0004]/value/value = ‘true’) OR
eval/data[at0001]/items[at0019]/items[at0006]/value/value = ‘true’) AND
c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

AQL PATH

Select e/ehr_id/value **Table P5** as ehr_id
From Ehr **Table P1** e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
Where (((((eval/data[at0001]/items[at0019]/items[at0002]/value/value **PV351** = ‘true’ OR
eval/data[at0001]/items[at0019]/items[at0003]/value/value **PV353** = ‘true’) OR
eval/data[at0001]/items[at0019]/items[at0004]/value/value **PV355** = ‘true’) OR
eval/data[at0001]/items[at0019]/items[at0006]/value/value **PV357** = ‘true’) AND
c/context/start_time/value **PV275** >= '2006-01-01T00:00:00,000+01:00') AND
c/context/start_time/value **PV275** < '2006-05-01T00:00:00,000+01:00')

SQL QUERY

Select substring (P5.dewey_id, ‘.’, 2), substring (PV351.dewey_id, ‘.’, 2), PV351.status, substring (PV353.dewey_id, ‘.’, 2), PV353.status, substring (PV355.dewey_id, ‘.’, 2), PV355.status, substring (PV357.dewey_id, ‘.’, 2), PV357.status, substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV351 on substring (P5.dewey_id, ‘.’, 2) = substring (PV351.dewey_id, ‘.’, 2)
join PV353 on substring (P5.dewey_id, ‘.’, 2) = substring (PV353.dewey_id, ‘.’, 2)
join PV355 on substring (P5.dewey_id, ‘.’, 2) = substring (PV355.dewey_id, ‘.’, 2)
join PV357 on substring (P5.dewey_id, ‘.’, 2) = substring (PV357.dewey_id, ‘.’, 2)
join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)

where (PV351.satus = ‘true’ OR PV353.status = ‘true’ OR PV355.status = ‘true’ OR PV357.status = ‘true’)
AND
(PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00')

EXAMPLE 9

AQL MAIN QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       (((eval/data[at0001]/items[at0019]/items[$node]/value/value = ‘true’) AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

$node = (at0002, at0003, at0004, at0006)

AQL PATH

Select e/ehr_id/value **Table P5** as ehr_id
From Ehr **Table P1** e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
Where (((eval/data[at0001]/items[at0019]/items[$node]/value/value = ‘true’) AND
c/context/start_time/value **PV275** >= '2006-01-01T00:00:00,000+01:00') AND
c/context/start_time/value **PV275** < '2006-05-01T00:00:00,000+01:00')

$node = (at0002 **PV351**, at0003 **PV353**, at0004 **PV355**, at0006 **PV357**)

AQL QUERY 1

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       (((eval/data[at0001]/items[at0019]/items[at0002]/value/value = ‘true’) AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV351.dewey_id, ‘.’, 2), PV351.status,
       substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone from P5
        join PV351 on substring (P5.dewey_id, ‘.’, 2) = substring (PV351.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV351.status = ‘true’ AND
       PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
       PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       (((eval/data[at0001]/items[at0019]/items[at0003]/value/value = ‘true’) AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV353.dewey_id, ‘.’, 2), PV353.status,
        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone from P5
         join PV353 on substring (P5.dewey_id, ‘.’, 2) = substring (PV353.dewey_id, ‘.’, 2)
         join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV353.status = ‘true’ AND
        PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
        PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 3

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       (((eval/data[at0001]/items[at0019]/items[at0004]/value/value = ‘true’) AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV355.dewey_id, ‘.’, 2), PV355.status,
        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone from P5
         join PV355 on substring (P5.dewey_id, ‘.’, 2) = substring (PV355.dewey_id, ‘.’, 2)
         join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
 where PV355.status = ‘true’ AND
         PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
         PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 4

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       (((eval/data[at0001]/items[at0019]/items[at0006]/value/value = ‘true’) AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV357.dewey_id, ‘.’, 2), PV357.status,
        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone from P5
```

```
        join PV357 on substring (P5.dewey_id, '.', 2) = substring (PV357.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV357.status = 'true' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 10

AQL QUERY

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           ((eval/data[at0001]/items[at0020]/items[at0069]/items[$node]/value/value = 'true' AND
                c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

$node = (at0012, at0013, at0014, at0015, at0016, at0017, at0018)

AQL PATH

Select e/ehr_id/value **Table P5** as ehr_id
From Ehr **Table P1** e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
Where ((eval/data[at0001]/items[at0020]/items[at0069]/items[$node]/value/value = 'true' AND
c/context/start_time/value **PV275** >= '2006-01-01T00:00:00,000+01:00') AND
c/context/start_time/value **PV275** <= '2006-05-01T00:00:00,000+01:00')

$node = (at0012 **PV500**, at0013 **PV502**, at0014 **PV504**, at0015 **PV506**, at0016 **PV508**, at0017 **PV510**, at0018 **PV512**)

AQL QUERY 1

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where           ((eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value = 'true' AND
                c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV500.dewey_id, '.', 2), PV500.status,
       substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV500 on substring (P5.dewey_id, '.', 2) = substring (PV500.dewey_id, '.', 2)
          join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV500.status = 'true' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       ((eval/data[at0001]/items[at0020]/items[at0069]/items[at0013]/value/value = ‘true’ AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV502.dewey_id, ‘.’, 2), PV502.status,
      substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV502 on substring (P5.dewey_id, ‘.’, 2) = substring (PV502.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV502.status = ‘true’ AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 3

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       ((eval/data[at0001]/items[at0020]/items[at0069]/items[at0014]/value/value = ‘true’ AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV504.dewey_id, ‘.’, 2), PV504.status,
      substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV504 on substring (P5.dewey_id, ‘.’, 2) = substring (PV504.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
 where PV504.status = ‘true’ AND
       PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
       PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 4

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       ((eval/data[at0001]/items[at0020]/items[at0069]/items[at0015]/value/value = ‘true’ AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV506.dewey_id, ‘.’, 2), PV506.status,
      substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV506 on substring (P5.dewey_id, ‘.’, 2) = substring (PV506.dewey_id, ‘.’, 2)
```

```
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV506.status = 'true' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 5

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       ((eval/data[at0001]/items[at0020]/items[at0069]/items[at0016]/value/value = 'true' AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV508.dewey_id, '.', 2), PV508.status,
      substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV508 on substring (P5.dewey_id, '.', 2) = substring (PV508.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV508.status = 'true' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 6

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       ((eval/data[at0001]/items[at0020]/items[at0069]/items[at0017]/value/value = 'true' AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV510.dewey_id, '.', 2), PV510.status,
      substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV510 on substring (P5.dewey_id, '.', 2) = substring (PV510.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
 where PV510.status = 'true' AND
       PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
       PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 7

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       ((eval/data[at0001]/items[at0020]/items[at0069]/items[at0018]/value/value = 'true' AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value <= '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV512.dewey_id, '.', 2), PV512.status,
       substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV512 on substring (P5.dewey_id, '.', 2) = substring (PV512.dewey_id, '.', 2)
         join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV512.status = 'true' AND
       PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
       PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 11

AQL QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
where       ((eval/data[at0001]/items[at0022]/items[at0023]/value/value = 'true' AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value <  '2006-05-01T00:00:00,000+01:00')
```

AQL PATH

```
                    Table P5
Select    e/ehr_id/value          as ehr_id
               Table P1
From      Ehr             e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains  evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
                                                          PV370
Where     ((eval/data[at0001]/items[at0022]/items[at0023]/value/value      = 'true' AND
                    PV275
          c/context/start_time/value      >= '2006-01-01T00:00:00,000+01:00') AND
                    PV275
          c/context/start_time/value      <  '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV370.dewey_id, '.', 2), PV370.status,
       substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV370 on substring (P5.dewey_id, '.', 2) = substring (PV370.dewey_id, '.', 2)
         join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV370.status = 'true' AND
       PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
       PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 12

AQL QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
```

```
where       (((((((eval/data[at0001]/items[at0022]/items[at0070]/items[at0024]/value/value = 'true' OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0025]/value/value = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0026]/value/value = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0027]/value/value = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0028]/value/value = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0029]/value/value = 'true') AND
            c/context/start_time/value >='2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

AQL PATH

```
Select      e/ehr_id/value [Table P5] as ehr_id
From        Ehr [Table P1] e
Contains    version v
Contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains    evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
Where       (((((((eval/data[at0001]/items[at0022]/items[at0070]/items[at0024]/value/value [PV373] = 'true' OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0025]/value/value [PV375] = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0026]/value/value [PV377] = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0027]/value/value [PV379] = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0028]/value/value [PV381] = 'true') OR
            eval/data[at0001]/items[at0022]/items[at0070]/items[at0029]/value/value [PV383] = 'true') AND
            c/context/start_time/value [PV275] >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value [PV275] < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV373.dewey_id, '.', 2), PV373.status,
                                        substring (PV375.dewey_id, '.', 2), PV375.status,
                                        substring (PV377.dewey_id, '.', 2), PV377.status,
                                        substring (PV379.dewey_id, '.', 2), PV379.status,
                                        substring (PV381.dewey_id, '.', 2), PV381.status,
                                        substring (PV383.dewey_id, '.', 2), PV383.status,
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from   P5 join PV373 on substring (P5.dewey_id, '.', 2) = substring (PV373.dewey_id, '.', 2)
          join PV375 on substring (P5.dewey_id, '.', 2) = substring (PV375.dewey_id, '.', 2)
          join PV377 on substring (P5.dewey_id, '.', 2) = substring (PV377.dewey_id, '.', 2)
          join PV379 on substring (P5.dewey_id, '.', 2) = substring (PV379.dewey_id, '.', 2)
          join PV381 on substring (P5.dewey_id, '.', 2) = substring (PV381.dewey_id, '.', 2)
          join PV383 on substring (P5.dewey_id, '.', 2) = substring (PV383.dewey_id, '.', 2)
          join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where (PV373.satus = 'true' OR PV375.status = 'true' OR PV377.status = 'true' OR PV379.status = 'true'
       OR PV381.status = 'true' OR PV383.status = 'true') AND
       (PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
        PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00')
```

EXAMPLE 13

AQL MAIN QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
```

```
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[$indNode]/value/defining_code/code_string = $code AND
                c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

indNode in (at0049, at0050, at0051)
code in (at0061, at0066), (at0062, at0065), (at0063, at0064)
```

AQL PATH

Table P5

```
Select    e/ehr_id/value          as ehr_id
```

Table P1

```
From      Ehr             e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains  evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
contains  cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
Where     ((c1/items[at0041]/items[$indNode]/value/defining_code/code_string = $code AND
```

PV275

```
          c/context/start_time/value      >= '2006-01-01T00:00:00,000+01:00') AND
```

PV275

```
          c/context/start_time/value      < '2006-05-01T00:00:00,000+01:00')
```

PV529 **PV497** **PV568**

```
indNode in (at0049      , at0050        , at0051        )
code in (at0061, at0066), (at0062, at0065), (at0063, at0064)
```

AQL QUERY 1a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[at0049]/value/defining_code/code_string = ‘at0061’ AND
                  c1/items[at0041]/items[at0049]/value/defining_code/code_string = ‘at0066’ AND
                  c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                  c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, ‘.’, 2), substring (PV529.dewey_id, ‘.’, 2), PV529.atid,
                                        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV529 on substring (P5.dewey_id, ‘.’, 2) = substring (PV529.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV529.atid = ‘at0061’ AND
      PV529.atid = ‘at0066’ AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 1b

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
```

```
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[at0049]/value/defining_code/code_string = ‘at0062’ AND
                  c1/items[at0041]/items[at0049]/value/defining_code/code_string = ‘at0065’ AND
                  c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                  c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, ‘.’, 2), substring (PV529.dewey_id, ‘.’, 2), PV529.atid,
                                        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV529 on substring (P5.dewey_id, ‘.’, 2) = substring (PV529.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV529.atid = ‘at0062’ AND
      PV529.atid = ‘at0065’ AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 1c

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[at0049]/value/defining_code/code_string = ‘at0063’ AND
                  c1/items[at0041]/items[at0049]/value/defining_code/code_string = ‘at0064’ AND
                  c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                  c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, ‘.’, 2), substring (PV529.dewey_id, ‘.’, 2), PV529.atid,
                                        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV529 on substring (P5.dewey_id, ‘.’, 2) = substring (PV529.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV529.atid = ‘at0063’ AND
      PV529.atid = ‘at0064’ AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2a

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[at0050]/value/defining_code/code_string = ‘at0061’ AND
                  c1/items[at0041]/items[at0050]/value/defining_code/code_string = ‘at0066’ AND
                  c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                  c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, ‘.’, 2), substring (PV497.dewey_id, ‘.’, 2), PV497.atid,
                                        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV497 on substring (P5.dewey_id, ‘.’, 2) = substring (PV497.dewey_id, ‘.’, 2)
```

```
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV497.atid = 'at0061' AND
      PV497.atid = 'at0066' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2b

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/items[at0041]/items[at0050]/value/defining_code/code_string = 'at0062' AND
              c1/items[at0041]/items[at0050]/value/defining_code/code_string = 'at0065' AND
              c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
              c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, '.', 2), substring (PV497.dewey_id, '.', 2), PV497.atid,
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV497 on substring (P5.dewey_id, '.', 2) = substring (PV497.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV497.atid = 'at0062' AND
      PV497.atid = 'at0065' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2c

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/items[at0041]/items[at0050]/value/defining_code/code_string = 'at0063' AND
              c1/items[at0041]/items[at0050]/value/defining_code/code_string = 'at0064' AND
              c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
              c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, '.', 2), substring (PV497.dewey_id, '.', 2), PV497.atid,
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV497 on substring (P5.dewey_id, '.', 2) = substring (PV497.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV497.atid = 'at0063' AND
      PV497.atid = 'at0064' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 3a

```
Select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
```

```
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[at0051]/value/defining_code/code_string = ‘at0061’ AND
                 c1/items[at0041]/items[at0051]/value/defining_code/code_string = ‘at0066’ AND
                 c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                 c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, ‘.’, 2), substring (PV568.dewey_id, ‘.’, 2), PV568.atid,
                                        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV568 on substring (P5.dewey_id, ‘.’, 2) = substring (PV568.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV568.atid = ‘at0061’ AND
      PV568.atid = ‘at0066’ AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 3b

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[at0051]/value/defining_code/code_string = ‘at0062’ AND
                  c1/items[at0041]/items[at0051]/value/defining_code/code_string = ‘at0065’ AND
                  c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                  c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, ‘.’, 2), substring (PV568.dewey_id, ‘.’, 2), PV568.atid,
                                        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV568 on substring (P5.dewey_id, ‘.’, 2) = substring (PV568.dewey_id, ‘.’, 2)
        join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV568.atid = ‘at0062’ AND
      PV568.atid = ‘at0065’ AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 3c

```
select          e/ehr_id/value as ehr_id
from            Ehr e
contains        version v
contains        composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains        evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains        cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where           ((c1/items[at0041]/items[at0051]/value/defining_code/code_string = ‘at0063’ AND
                  c1/items[at0041]/items[at0051]/value/defining_code/code_string = ‘at0064’ AND
                  c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                  c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring (P5.dewey_id, ‘.’, 2), substring (PV568.dewey_id, ‘.’, 2), PV568.atid,
```

```
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV568 on substring (P5.dewey_id, '.', 2) = substring (PV568.dewey_id, '.', 2)
          join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV568.atid = 'at0063' AND
       PV568.atid = 'at0064' AND
       PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
       PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 14

AQL QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/ items[$node]/value/defining_code/code_string = $code AND
            c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
            c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

node in (at0043, at0044)
code in (at0053, at0054, at0055, at0056), (at0057, at0058, at0059, at0060)
```

AQL PATH

```
                    Table P5
Select    e/ehr_id/value        as ehr_id
               Table P1
From      Ehr          e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
Contains  evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
contains  cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
Where     ((c1/ items[$node]/value/defining_code/code_string = $code AND
                    PV275
          c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
                    PV275
          c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

              PV487      PV533
node in (at0043     , at0044     )
code in (at0053, at0054, at0055, at0056), (at0057, at0058, at0059, at0060)
```

AQL QUERY 1a

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/items[at0043]/value/defining_code/code_string = 'at0053' AND
              c1/items[at0043]/value/defining_code/code_string = 'at0054' AND
              c1/items[at0043]/value/defining_code/code_string = 'at0055' AND
              c1/items[at0043]/value/defining_code/code_string = 'at0056' AND
              c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
              c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV487.dewey_id, '.', 2), PV487.atid,
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV487 on substring (P5.dewey_id, '.', 2) = substring (PV487.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV487.atid = 'at0053' AND
      PV487.atid = 'at0054' AND
      PV487.atid = 'at0055' AND
      PV487.atid = 'at0056' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 1b

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/items[at0043]/value/defining_code/code_string = 'at0057' AND
              c1/items[at0043]/value/defining_code/code_string = 'at0058' AND
              c1/items[at0043]/value/defining_code/code_string = 'at0059' AND
              c1/items[at0043]/value/defining_code/code_string = 'at0060' AND
              c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
              c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV487.dewey_id, '.', 2), PV487.atid,
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV487 on substring (P5.dewey_id, '.', 2) = substring (PV487.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV487.atid = 'at0057' AND
      PV487.atid = 'at0058' AND
      PV487.atid = 'at0059' AND
      PV487.atid = 'at0060' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2a

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/items[at0044]/value/defining_code/code_string = 'at0053' AND
              c1/items[at0044]/value/defining_code/code_string = 'at0054' AND
              c1/items[at0044]/value/defining_code/code_string = 'at0055' AND
              c1/items[at0044]/value/defining_code/code_string = 'at0056' AND
              c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
              c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV533.dewey_id, '.', 2), PV533.atid,
```

```
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV533 on substring (P5.dewey_id, '.', 2) = substring (PV533.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV533.atid = 'at0053' AND
      PV533.atid = 'at0054' AND
      PV533.atid = 'at0055' AND
      PV533.atid = 'at0056' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2b

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/items[at0044]/value/defining_code/code_string = 'at0057' AND
              c1/items[at0044]/value/defining_code/code_string = 'at0058' AND
              c1/items[at0044]/value/defining_code/code_string = 'at0059' AND
              c1/items[at0044]/value/defining_code/code_string = 'at0060' AND
              c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
              c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV533.dewey_id, '.', 2), PV533.atid,
                                        substring (PV275.dewey_id, '.', 2), PV275.date_time_zone
from P5 join PV533 on substring (P5.dewey_id, '.', 2) = substring (PV533.dewey_id, '.', 2)
        join PV275 on substring (P5.dewey_id, '.', 2) = substring (PV275.dewey_id, '.', 2)
where PV533.atid = 'at0057' AND
      PV533.atid = 'at0058' AND
      PV533.atid = 'at0059' AND
      PV533.atid = 'at0060' AND
      PV275.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV275.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 15

AQL QUERY

```
select      e/ehr_id/value as ehr_id
from        Ehr e
contains    version v
contains    composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
contains    evaluation eval[openEHR-EHR-EVALUATION.citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
where       ((c1/ items[at0045]/value/value = 'true' AND
              c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
              c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

AQL PATH

```
                    Table P5
Select    e/ehr_id/value          as ehr_id
              Table P1
From      Ehr             e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.citologic_exam_form.v1]
```

```
Contains  evaluation eval[openEHR-EHR-EVALUATION. citologic_exam.v1]
contains    cluster c1[openEHR-EHR-CLUSTER.follow_up_citologic_exam_result.v1]
                                                    PV407
Where     ((c1/ items[at0045]/value/value          = ‘true’ AND
                                          PV275
            c/context/start_time/value          >= '2006-01-01T00:00:00,000+01:00') AND
                                          PV275
            c/context/start_time/value          < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV407.dewey_id, ‘.’, 2), PV407.status,
                                        substring (PV275.dewey_id, ‘.’, 2), PV275.date_time_zone
from P5 join PV407 on substring (P5.dewey_id, ‘.’, 2) = substring (PV407.dewey_id, ‘.’, 2)
          join PV275 on substring (P5.dewey_id, ‘.’, 2) = substring (PV275.dewey_id, ‘.’, 2)
where PV407.status = ‘true’ AND
        PV275.date_time_zone >=  '2006-01-01T00:00:00,000+01:00' AND
        PV275.date_time_zone <  '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 16

AQL QUERY

```
Select    e/ehr_id/value as ehr_id
From      Ehr e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Where     (c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00' AND
           c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

AQL PATH

```
                          Table P5
Select    e/ehr_id/value              as ehr_id
                Table P1
From      Ehr               e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
                                          PV101
Where     (c/context/start_time/value          >= '2006-01-01T00:00:00,000+01:00' AND
                                          PV101
           c/context/start_time/value          < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select substring_index(P5.dewey_id, '.', 2), substring_index(PV101.dewey_id, '.', 2), PV101.date_time_zone
from P5 join PV101 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV101.dewey_id, '.', 2)
where PV101.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
        PV101.date_time_zone < '2006-05-01T00:00:00,000+01:00';
```

EXAMPLE 17

AQL QUERY

```
Select    e/ehr_id/value as ehr_id
From      Ehr e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains  observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
          (((((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0055]/value/
          value = ‘true’ OR
```

obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0056]/value/valu
e = ‘true’) OR
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0057]/value/valu
e = ‘true’) OR
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0058]/value/valu
e = ‘true’)
AND c/context/start_time/value >= $beginTime) AND c/context/start_time/value < $endTime)

AQL PATH

Select e/ehr_id/value **Table P5** as ehr_id
From Ehr **Table P1** e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where

(((((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0055]/value/
value **PV149** = ‘true’ OR
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0056]/value/valu
e **PV151** = ‘true’) OR
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0057]/value/valu
e **PV153** = ‘true’) OR
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/items[at0058]/value/valu
e **PV155** = ‘true’) AND
c/context/start_time/value **PV101** >= '2006-01-01T00:00:00,000+01:00') AND
c/context/start_time/value **PV101** < '2006-05-01T00:00:00,000+01:00')

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV149.dewey_id, ‘.’, 2), PV149.status,
                                        substring (PV151.dewey_id, ‘.’, 2), PV151.status,
                                        substring (PV153.dewey_id, ‘.’, 2), PV153.status,
                                        substring (PV155.dewey_id, ‘.’, 2), PV155.status,
                                        substring (PV101.dewey_id, ‘.’, 2), PV101.date_time_zone
from P5 join PV149 on substring (P5.dewey_id, ‘.’, 2) = substring (PV149.dewey_id, ‘.’, 2)
        join PV151 on substring (P5.dewey_id, ‘.’, 2) = substring (PV151.dewey_id, ‘.’, 2)
        join PV153 on substring (P5.dewey_id, ‘.’, 2) = substring (PV153.dewey_id, ‘.’, 2)
        join PV155 on substring (P5.dewey_id, ‘.’, 2) = substring (PV155.dewey_id, ‘.’, 2)
        join PV101 on substring (P5.dewey_id, ‘.’, 2) = substring (PV101.dewey_id, ‘.’, 2)
where PV149.status = ‘true’ OR
      PV151.status = ‘true’ OR
      PV153.status = ‘true’ OR
      PV155.status = ‘true’ AND
      PV101.date_time_zone = '2006-01-01T00:00:00,000+01:00' AND
      PV101.date_time_zone = '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 18

AQL QUERY *(A query example is to return all record ids that had a histologic exam result indicating neoplastic lesions between 2006-01-01 and 2006-05-01).*

```
Select    e/ehr_id/value as ehr_id
From      Ehr e
Contains  version v
```

```
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains  observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where     (((EXISTS
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034] OR
            EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0035])
AND
            c/context/start_time/value >= $beginTime)  AND
            c/context/start_time/value < $endTime)

$beginTime = '2006-01-01T00:00:00,000+01:00'
$endTime = '2006-05-01T00:00:00,000+01:00'
```

This query is same as 1st example query but instead of giving the beginTime and endTime separately, there its given directly.

AQL PATH

```
Select      e/ehr_id/value                Table P5   as ehr_id
From        Ehr                 Table P1  e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains  observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where     (((EXISTS
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]  P343  OR
            EXISTS
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0035])  P1271  AND
            c/context/start_time/value  PV101  >= '2006-01-01T00:00:00,000+01:00')  AND
            c/context/start_time/value  PV101  < '2006-05-01T00:00:00,000+01:00')
```

SQL QUERY

```
select  substring_index(P5.dewey_id, '.', 2), substring_index(PV101.dewey_id, '.', 2), PV101.date_time_zone

from   P5 join PV101 on substring_index(P5.dewey_id, '.', 2)=substring_index(PV101.dewey_id, '.', 2)

where (substring_index(P5.dewey_id, '.', 2) in (select substring_index(P343.dewey_id, '.', 2) from P343)  OR

     substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1271.dewey_id, '.', 2) from P1271)) AND

 (PV101.date_time_zone between '2006-01-01T00:00:00,000+01:00' AND '2006-05-1T00:00:00,000+01:00');
```

EXAMPLE 19

AQL MAIN QUERY

```
Select      e/ehr_id/value as ehr_id
From        Ehr e
Contains  version v
Contains  composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains  observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
          ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/value/def
ining_code/code_string = $nic AND c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')
```

nic in (at0046, at0047, at0048)

AQL PATH

Table P5

Select e/ehr_id/value as ehr_id

Table P1

From Ehr e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
(obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/value/defi
PV160
ning_code/code_string = $nic AND
PV101
c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
PV101
c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

nic in (at0046, at0047, at0048)

AQL QUERY 1

```
Select          e/ehr_id/value as ehr_id
From            Ehr e
Contains        version v
Contains        composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains        observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
                ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/
                value/defining_code/code_string = ‘at0046’ AND c/context/start_time/value >= '2006-01-
                01T00:00:00,000+01:00') AND c/context/start_time/value < '2006-05-
                01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, ‘.’, 2), substring (PV160.dewey_id, ‘.’, 2), PV160.atid,
                                        substring (PV101.dewey_id, ‘.’, 2), PV101.date_time_zone
from P5 join PV160 on substring (P5.dewey_id, ‘.’, 2) = substring (PV160.dewey_id, ‘.’, 2)
          join PV101 on substring (P5.dewey_id, ‘.’, 2) = substring (PV101.dewey_id, ‘.’, 2)
where PV160.atid = ‘at0046’AND
        PV101.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
        PV101.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 2

```
Select          e/ehr_id/value as ehr_id
From            Ehr e
Contains        version v
Contains        composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains        observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
                ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/
                value/defining_code/code_string = ‘at0047’ AND c/context/start_time/value >= '2006-01-
                01T00:00:00,000+01:00') AND c/context/start_time/value < '2006-05-
                01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV160.dewey_id, '.', 2), PV160.atid,
                                        substring (PV101.dewey_id, '.', 2), PV101.date_time_zone
from P5 join PV160 on substring (P5.dewey_id, '.', 2) = substring (PV160.dewey_id, '.', 2)
        join PV101 on substring (P5.dewey_id, '.', 2) = substring (PV101.dewey_id, '.', 2)
where PV160.atid = 'at0047'AND
      PV101.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV101.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

AQL QUERY 3

```
Select      e/ehr_id/value as ehr_id
From        Ehr e
Contains    version v
Contains    composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains    observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
            ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/
            value/defining_code/code_string = 'at0048' AND c/context/start_time/value >=  '2006-01-
            01T00:00:00,000+01:00') AND c/context/start_time/value < '2006-05-
            01T00:00:00,000+01:00')
```

SQL QUERY

```
Select substring (P5.dewey_id, '.', 2), substring (PV160.dewey_id, '.', 2), PV160.atid,
                                        substring (PV101.dewey_id, '.', 2), PV101.date_time_zone
from P5 join PV160 on substring (P5.dewey_id, '.', 2) = substring (PV160.dewey_id, '.', 2)
        join PV101 on substring (P5.dewey_id, '.', 2) = substring (PV101.dewey_id, '.', 2)
where PV160.atid = 'at0048'AND
      PV101.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
      PV101.date_time_zone < '2006-05-01T00:00:00,000+01:00'
```

EXAMPLE 20

AQL QUERY

```
Select      e/ehr_id/value as ehr_id
From        Ehr e
Contains    version v
Contains    composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains    observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
            ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/value/def
ining_code/code_string matches {'at0049', 'at0050', 'at0051', 'at0052', 'at0053'} AND
c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00')  AND c/context/start_time/value < '2006-
05-01T00:00:00,000+01:00')
```

AQL PATH

Table P5

```
Select      e/ehr_id/value              as ehr_id
```

Table P1

```
From        Ehr             e
Contains    version v
Contains    composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains    observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where
            ((obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0034]/value/def
```

PV160
ining_code/code_string matches {‘at0049’, ‘at0050’, ‘at0051’, ‘at0052’, ‘at0053’} AND
PV101
c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
PV101
c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

SQL QUERY

Select substring(P5.dewey_id, '.', 2), substring (PV160.dewey_id, '.', 2), PV160.atid, substring (PV101.dewey_id, '.', 2), PV101.date_time_zone
From P5 join PV160 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV160.dewey_id, '.', 2) join PV101 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV101.dewey_id, '.', 2)
Where ((PV160.atid IN ('at0049', 'at0050', 'at0051', 'at0052', 'at0053') AND PV101.date_time_zone >= '2006-01-01T00:00:00,000+01:00’) AND PV101.date_time_zone < '2006-05-01T00:00:00,000+01:00’)

EXAMPLE 21

AQL QUERY

Select e/ehr_id/value as ehr_id
From Ehr e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where ((EXISTS obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0035] AND
c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

AQL PATH

Table P5
Select e/ehr_id/value as ehr_id
Table P1
From Ehr e
Contains version v
Contains composition c[openEHR-EHR-COMPOSITION.histologic_exam.v1]
Contains observation obs[openEHR-EHR-OBSERVATION.histological_exam_result.v1]
Where ((EXISTS
P1271
obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/items[at0035]
PV101
AND c/context/start_time/value >= '2006-01-01T00:00:00,000+01:00') AND
PV101
c/context/start_time/value < '2006-05-01T00:00:00,000+01:00')

SQL QUERY

select substring_index(P5.dewey_id, '.', 2), substring_index(PV101.dewey_id, '.', 2), PV101.date_time_zone
from P5 join PV101 on substring_index(P5.dewey_id, '.', 2) = substring_index(PV101.dewey_id, '.', 2)
where substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1271.dewey_id, '.', 2) from P1271)
AND
PV101.date_time_zone >= '2006-01-01T00:00:00,000+01:00' AND
PV101.date_time_zone < '2006-05-01T00:00:00,000+01:00';

These are some of the sample queries that have been tested during the implementation process. Also, these tests were performed in windows command prompt with installation of ANTLR4.2-complete.jar file and JAVA1.6.

Chapter 5

Conclusion and Future Work

This chapter concludes the work that has been done in this thesis work and possibilities of future work.

5.1 Conclusion

The aim of this thesis work is to improve and optimize the performance of retrieving the Archetype Query Language (AQL) queries which are stored in the form of XML database. XML databases are tree like structures. Earlier approaches for improving the query performance used XML databases with XQuery or an index and Hadoop. As the performance were not satisfactory, in this thesis we convert AQL queries into SQL queries. Since the archetypes are based on clinical data, the study is based on Electronic Health Record (EHR) and openEHR.

These translation part is done manually in the starting stage. While translating, the paths and values are considered as tables and after the translation, the AQL grammar is debugged in this report and the translation is implemented in Java language. The AQL queries which are translated into SQL queries are also tested in this thesis work. Finally, the result is to get the corresponding SQL query for any given AQL query.

The ANTLR parser generator is a powerful tool, in which, the input which we give is a grammar of AQL language and the output which we get is a source code or source program for a recognizer (i.e., a parser and a lexer) that recognizes a language described by the grammar. A parser will automatically generate a parse tree called an Abstract Syntax Tree (AST) according to the grammar that matches with the given input.

In order to translate, we implement language applications based on those grammars by walking the automatically generated parse trees. Also, some actions are attached to the grammar elements in order to parse the input. These actions are written in programming languages like java, C#, etc. Here, the actions are arbitrary chunks of code written in target language (i.e., the language in which the ANTLR generates or translates the code) enclosed in {……}. Generally, these actions operate on attributes of tokens and rule references. For

example, we can ask for text of tokens or a text matched by entire rule invocations. Based on data derived from the token and rule references, we can print those texts and perform some computations.

5.2 Future Work

The whole system process are focused on two parts of which we did the second part:

- Implementing the SQL queries in sort-merge join using hadoop
- Translating AQL queries to SQL queries

These parts are done separately and should be merged to get the whole system. Further, AQL query translation is done using ANTLR parser tool. This can be compared to other best tools for translation. Since only queries are translated in this work, the database may also be translated to do the same. Once the database is translated, it can be tested and improved to store the data in cluster (for big data using hadoop).

References

[1] Sergio Miranda Freire, Erik Sundvall, Daniel Karlsson, Patrick Lambrix – *Performance of XML Databases for Epidemiological Queries in Archetype-Based EHRs*. ISBN: 978-91-7519-758-6. Scandinavian Conference on Health Informatics, Pages: 51-57, Linköping, Sverige, 2012.

[2] openEHR. The openEHR Foundation [Internet]. Available at: http://www.openEHR.org. Accessed: 2015-08-10.

[3] openEHR Foundation. Archetype Query Language [Internet]. Available at: http://www.openEHR.org/wiki/display/spec/Archetype+Query+Language+Description#ArchetypeQueryLanguageDescription-AQLidentifiedpaths. Accessed: 2015-08-30.

[4] ANTLR. The ANTLR Foundation [Internet]. Available at: http://www.antlr.org. Accessed: 2015-08-30.

[5] Terence Parr – The Definitive ANTLR Reference, 2012

[6] Terence Parr – The Definitive ANTLR Reference – Building Domain-Specific Languages, 2007

[7] Hajar Kashfi – *The Intersection of Clinical Decision Support and Electronic Health Record: A Literature Review*. E-ISBN: 978-83-60810-35-4, IEEE Proceeding of the Federated Conference on Computer Science and Information System, Pages: 347-353, Chalmers University of Technology, 2011.

[8] Filgueira R, Odriazola A, Simini F – *Using openEHR in SICTI an electronic Health record System for critical medicine*. Journal of Physics, Pages: 012001, 2007.

[9] Shelly Sachdeva, Daigo Yaginuma, Wanming Chu, and Subhash Bhalla – *Dynamic Generation of Archetype-Based User Interfaces for Queries on Electronic Health Record Databases*. Online ISBN: 978-3-642-25731-5. Proceedings of the Seventh International Conference on Databases in Networked Information Systems, Pages: 109-125, Aizu-Wakamatsu, Japan, 2011. © Springer-Verlag.

[10] ANTLR: Getting started. The ANTLR foundation [Internet]. Available at: https://theantlrguy.atlassian.net/wiki/display/ANTLR4/Getting+Started+with+ANTLR+v4. Accessed: 2015-08-30.

[11] Introducing openEHR. Rev 1.1, © 2003-2007. The openEHR foundation [Internet]. Available at: http://www.openehr.org/releases/1.0.1/openEHR/introducing_openEHR.pdf. Accessed: 2015-08-30.

[12] Dr. Ian McNicoll, Dr. Healther Leslie – *Introduction to openEHR Archetypes and Templates*. © Ocean Informatics 2010. The openEHR foundation.
[13] Dr. Sebastian Garde. *Electronic Health Records – An Introduction to openEHR and archetypes*. CCR Workshop Munich, 2008. © Ocean Informatics 2008. The openEHR foundation [Internet]. Available at: http://www.helmholtz-muenchen.de/fileadmin/BYMEDCONNECT/PDF/2008_04_CCR-DE_Garde_-_An_introduction_to_openEHR_and_archetypes.pdf
[14] Byron R. Hamilton. Electronic Health Records, 2/e ISBN: 0077477553, 2011 by The McGraw-Hill Companies, Inc.
[15] Byron R. Hamilton. Electronic Health Records, 3/e ISBN: 0073402141, 2013 by The McGraw-Hill Companies, Inc.
[16] Report of the eHealth Stakeholder Group. *Patient access to Electronic Health Records*. Issue Leader: Illaria Passarani, BEUC. Version: June 2013.
[17] Electronic Health Records, A Manual for Developing Countries. World Health Organization.
[18] Sundvall E, Nyström M, Sandström M, Eneling M, Örman H, Karlsson D – *REST Based Services and Storage Interfaces for openEHR Implementations.* Department of Biomedical Engineering, Linköpings universitet, Sweden.
[19] openEHR. The openEHR Foundation [Internet]. Available at: https://openehr.atlassian.net/wiki/display/resources/Archetype+FAQs. Accessed: 2015-08-30.
[20] Heather Leslie. *International developments in openEHR archetypes and templates*. Health Information Management Journal Vol 37 No 1 2008 ISSN 1833-3583 (PRINT) ISSN 1833-3575 (ONLINE)
[21] Chunlan Ma, Heath Frankel, Thomas Beale, Sam Heard – *EHR Query Language (EQL) – A Query Language for Archetype-Based Health Records*. Ocean Informatics Pty. Ltd, Australia.
[22] Shelly Sachdeva, Subhash Bhalla – *Implementing High-Level Query Language Interfaces for Archetype-Based Electronic Health Records Database*. 15th International Conference on Management of Data. COMAD 2009, Mysore, India, December 9-12, 2009, © Computer Society of India.
[23] openEHR Clinical Knowledge Manager. The openEHR Foundation [Internet]. Available at: http://www.openehr.org/ckm/. Accessed: 2015-08-30.

[24] openEHR. The openEHR Foundation [Internet]. Available at: https://openehr.atlassian.net/wiki/display/healthmod/Clinical+Content+Models. Accessed: 2015-08-30.
[25] Thomas Beale. Archetypes – Constraint-based Domain Models for Future-proof Information Systems. *11th OOPSLA Workshop on Behaviroral Semantics*. Available at: http://www.openehr.org/files/resources/publications/archetypes/archetypes_beale_oopsla_2002.pdf.
[26] Beale T, Heard S, Kalra D, Lloyd D – openEHR Architecture Overview. Available at: http://www.openehr.org/releases/1.0/architecture/overview.pdf
[27] Jon Patrick, Richard Ly, Donna Truran – *Evaluation of a Persistent Store for openEHR*. Publisher: Health Informatics Society of Australia Ltd (HISA). ISBN: 0 9751013 7 4
[28] Kazi Farooqui, Luigi Logrippo, Jan de Meer – *The ISO Reference Model for Open Distributed Processing- An Introduction*. Available at: http://www.enterprise-architecture.info/Images/Documents/RM-ODP2.pdf
[29] openEHR Foundation. Archetype Query Language [Internet]. Available at: https://openehr.atlassian.net/wiki/display/spec/Archetype+Query+Language+Description. Accessed: 2015-08-30.
[30] Gornik T, Fabjan B – Changing the Rules: the openEHR and IHE ecosystem [Internet]. Available at: http://www.interconnected-health.org/presentations/02-T2-01_Gornik-Fabjan.pdf
[31] openEHR Foundation. Archetype Query Language [Internet]. Available at: http://www.openehr.org/programs/clinicalmodels/. Accessed: 2015-08-30.
[32] AQL Grammar. The ANTLR foundation [Internet]. Available at: https://openehr.atlassian.net/wiki/display/spec/ANTLR+AQL+grammar. Accessed: 2015-08-30.
[33] Anthony A. Aaby. "Compiler Construction using Flex and Bison". Walla Walla College. Version of February 25, 2004. Copyright 2003.
[34] M.E. Lesk, E. Schmidt. *Lex – A Lexical Analyzer Generator*. Computing Science Technical Report No. 39, Bell Laboratories, Murray Hills, New Jersey, 1975. Pages: 375-387. ISBN: 0-03-047529-5.

[35] Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. *Compilers Principles, Techniques and Tools*, Second Edition. Addison-Wesley, 2006. ISBN-13: 978-0321486813.
[36] Torben Ægidius Mogensen. *Basics of Compiler Design*. Department of Computer Science, University of Copenhagen, 2010. eBook: PDF, 283 pages. ISBN-13: 978-87-993154-0-6.
[37] Supriya V. Deshmukh, Prof. Dr. Emmanuel M. *Survey on Aggregation Operators and Functions of Relational Database Management System*, International Journal of Advanced and Innovative Research, pp.292-298, 2012. ISSN: 2278-7844.
[38] Hadoop. The Hadoop Foundation [Internet]. Available at: https://hadoop.apache.org/. Accessed: 2015-08-30.
[39] Jens Dittrich, Jorge-Arnulfo Quiane-Ruiz. *Efficient Big Data Processing in Hadoop MapReduce*. Proceedings of the VLDB Endowment,Volume 5 Issue 12, 2012. Pages: 2014-2015, ISSN: 2150-8097 doi>10.14778/2367502.2367562.
[40] ANTLR Grammar. The ANTLR foundation [Internet]. Available at: https://theantlrguy.atlassian.net/wiki/display/ANTLR3/Quick+Starter+on+Parser+Grammars+-+No+Past+Experience+Required. Accessed: 2015-12-24.

Appendix A

This appendix consist the original grammar of AQL language Aql.g4 with error which is downloaded from [32].

Aql.g4

```
// Author: Bostjan Lah
// (c) Copyright, Marand, http://www.marand.com
// Licensed under LGPL: http://www.gnu.org/copyleft/lesser.html
// Based on AQL grammar by Ocean Informatics:
http://www.openehr.org/wiki/download/attachments/2949295/EQL_v0.6.grm?version=1&modificationDate=1259650833000
grammar Aql;

options {
	output=AST;
	ASTLabelType=CommonTree;
	backtrack=true;
	memoize=true;
}

tokens {
	ORDERDESC;
	ORDERASC;
}

@header {
package com.marand.thinkehr.aql.antlr;
import com.marand.thinkehr.aql.AqlRecognitionException;
}

@lexer::header{
package com.marand.thinkehr.aql.antlr;
import com.marand.thinkehr.aql.AqlRecognitionException;
}
```

```
@members{
public void displayRecognitionError(String[] tokenNames, RecognitionException e)
{
  throw new AqlRecognitionException(e);
}
}

// Rule Definitions
//<Query> ::= <Select> <From>
//              | <Select> <From> <Where>
//              | <Select> <From> <OrderBy>   ! is this allowed?
//              | <Select> <From> <Where> <OrderBy>
query        : select from where? orderBy? ';'!? EOF!;

//<Select> ::= 'SELECT' <SelectExpr>
//             | 'SELECT' <TOP> <SelectExpr>
select       : SELECT top? selectExpr -> ^(SELECT top? selectExpr);

//<Top> ::= 'TOP' Integer
//          | 'TOP' Integer 'BACKWARD'
//          | 'TOP' Integer 'FORWARD'
top        : TOP INTEGER FORWARD? -> ^(TOP INTEGER)
           | TOP INTEGER BACKWARD -> ^(TOP INTEGER BACKWARD);

//<Where> ::= 'WHERE' <IdentifiedExpr>
where        :     WHERE identifiedExpr -> ^(WHERE identifiedExpr);

//<OrderBy> ::= 'ORDER BY' <OrderBySeq>
orderBy        : ORDERBY orderBySeq -> ^(ORDERBY orderBySeq);

//<OrderBySeq>  ::= <OrderByExpr>
//                  | <OrderByExpr> ',' <OrderBySeq>
orderBySeq         : orderByExpr (','! orderByExpr)*;

//<OrderByExpr> ::= <IdentifiedPath>
//                   | <IdentifiedPath> 'DESCENDING'
```

```
//                  | <IdentifiedPath> 'ASCENDING'
//                  | <IdentifiedPath> 'DESC'
//                  | <IdentifiedPath> 'ASC'
orderByExpr       : identifiedPath (DESCENDING|DESC) -> ^(ORDERDESC identifiedPath)
                  | identifiedPath (ASCENDING|ASC)? -> ^(ORDERASC identifiedPath);

//<SelectExpr> ::= <IdentifiedPathSeq>
selectExpr        : identifiedPathSeq;

//! When multiple paths provided, each IdentifiedPath must represent an object of type DataValue
//<IdentifiedPathSeq> ::= <IdentifiedPath//>
//                        | <IdentifiedPath> 'as' Identifier
//                        | <IdentifiedPath> ',' <IdentifiedPathSeq>
//                        | <IdentifiedPath> 'as' Identifier ',' <IdentifiedPathSeq>
identifiedPathSeq       : selectVar (','! selectVar)*;

selectVar               : identifiedPath^ asIdentifier?;

asIdentifier            : AS IDENTIFIER;

//<From> ::=   'FROM' <FromExpr>              ! stop or/and without root class
//           | 'FROM' <FromEHR> <ContainsExpr>
//           | 'FROM' <FromEHR>
//!          'FROM' <ContainsOr>
from       : FROM fromExpr -> ^(FROM fromExpr)
           | FROM ehrContains -> ^(FROM ehrContains);

//<FromEHR> ::= 'EHR' <StandardPredicate>
//              | 'EHR' Identifier <StandardPredicate>
//              | 'EHR' Identifier
fromEHR       : EHR IDENTIFIER -> ^(EHR IDENTIFIER)
              | EHR IDENTIFIER standardPredicate -> ^(EHR IDENTIFIER standardPredicate)
              | EHR standardPredicate -> ^(EHR standardPredicate);

ehrContains   : fromEHR (CONTAINS^ contains)?;
```

```
//<IdentifiedExpr> ::= <IdentifiedEquality>
//                  | <IdentifiedExprBoolean>
//                  |'(' <IdentifiedExprBoolean> ')'
//identifiedExpr
//                  : identifiedExprBoolean;
//                  | '('! identifiedExprBoolean ')'!;

//<IdentifiedExprBoolean> ::= <IdentifiedExpr> 'OR' <IdentifiedExpr>
//                          | <IdentifiedExpr> 'AND' <IdentifiedExpr>
//                          | <IdentifiedExpr> 'XOR' <IdentifiedExpr>
//                          | 'NOT"(' <IdentifiedExprBoolean> ')'
//                          | 'NOT' <IdentifiedEquality>
identifiedExpr            : identifiedExprAnd ((OR|XOR)^ identifiedExprAnd)*;

identifiedExprAnd         : identifiedEquality (AND^ identifiedEquality)*;

//<IdentifiedEquality> ::= <IdentifiedOperand> ComparableOperator <IdentifiedOperand>
//                       | <IdentifiedOperand> 'matches' '{' <MatchesOperand> '}'
//                       | <IdentifiedOperand> 'matches' RegExPattern
//                       | 'EXISTS' <IdentifiedPath>
identifiedEquality      : identifiedOperand ((MATCHES^ '{'! matchesOperand
'}'!)|(COMPARABLEOPERATOR^ identifiedOperand))
                        | EXISTS identifiedPath -> ^(EXISTS identifiedPath)
                        | '('! identifiedExpr ')'!
                        | NOT^ identifiedEquality;
//                      | identifiedOperand 'matches' '{' matchesOperand '}'
//                      | identifiedOperand 'matches' REGEXPATTERN
//                      | 'EXISTS' identifiedPath;

//<IdentifiedOperand> ::= <Operand> | <IdentifiedPath>
identifiedOperand       : operand | identifiedPath;
//!<IdentifiedOperand> ::= <Operand> | <RelativePath>

//<IdentifiedPath>::= Identifier
//                  | Identifier <Predicate>
//                  | Identifier '/' <ObjectPath>
```

```
//                  | Identifier <Predicate> '/' <ObjectPath>
identifiedPath      : IDENTIFIER predicate? ('/' objectPath)? -> ^(IDENTIFIER predicate?
objectPath?);
//!                 | Identifer <AbsolutePath>
//!                 | Identifer <Predicate> <AbsolutePath>

//<Predicate> ::= <NodePredicate>
Predicate       : nodePredicate;

//<NodePredicate> ::= '['<NodePredicateOr>']'
nodePredicate       : OPENBRACKET nodePredicateOr CLOSEBRACKET;

//<NodePredicateOr> ::= <NodePredicateAnd>
//                    | <NodePredicateOr> 'or' <NodePredicateAnd>
nodePredicateOr     : nodePredicateAnd (OR^ nodePredicateAnd)*;

//<NodePredicateAnd> ::= <NodePredicateComparable>
//                     | <NodePredicateAnd> 'and' <NodePredicateComparable>
nodePredicateAnd    : nodePredicateComparable (AND^ nodePredicateComparable)*;

//<NodePredicateComparable> ::= <PredicateOperand> ComparableOperator <PredicateOperand>
//                              | NodeId
//                              | NodeId ',' String           ! <NodeId> and name/value =
<String> shortcut
//                              | NodeId ',' parameter        ! <NodeId> and name/value =
<Parameter> shortcut
//                              | <NodePredicateRegEx>    ! /items[{/at0001.*/}], /items[at0001
and name/value matches {//}]
//                              | ArchetypeId
//                              | ArchetypeId ',' String      ! <NodeId> and name/value =
<String> shortcut
//                              | ArchetypeId ',' parameter  ! <NodeId> and name/value =
<Parameter> shortcut
nodePredicateComparable     : NODEID (COMMA^ (STRING|PARAMETER))?
                              | ARCHETYPEID (COMMA^ (STRING|PARAMETER))?
```

```
                                    | predicateOperand ((COMPARABLEOPERATOR^
predicateOperand)|(MATCHES^ REGEXPATTERN))
                                    | REGEXPATTERN     //! /items[{/at0001.*/}], /items[at0001 and
name/value matches {//}];

//<NodePredicateRegEx> ::= RegExPattern
//                          | <PredicateOperand> 'matches' RegExPattern
nodePredicateRegEx        : REGEXPATTERN
                          | predicateOperand MATCHES^ REGEXPATTERN;

//<MatchesOperand> ::= <ValueListItems>
//                       | UriValue
matchesOperand        : valueListItems | URIVALUE;

//! <ValueList> ::= '{'<ValueListItems>'}'
//<ValueListItems> ::= <Operand>
//                     | <Operand> ',' <ValueListItems>
valueListItems        : operand (','! operand)*;

//<URI> ::= '{' UriValue '}'
uri         : '{' URIVALUE '}';

//<ArchetypePredicate> ::= '[' ArchetypeId ']'
//                         | '[' Parameter ']'
//                         | '[' RegExPattern ']'
archetypePredicate        : OPENBRACKET (archetypeId|PARAMETER|REGEXPATTERN)
CLOSEBRACKET;

archetypeId               : ARCHETYPEID;

//<VersionPredicate> ::= '[' <VersionPredicateOptions> ']'
versionPredicate        : OPENBRACKET versionPredicateOptions CLOSEBRACKET;

//<VersionPredicateOptions> ::= 'latest_version' | 'all_versions'
versionPredicateOptions        : 'latest_version' | ALL_VERSIONS;
```

```
//<StandardPredicate> ::= '[' <PredicateExpr> ']'
standardPredicate        : '['! predicateExpr ']'!;

//<PredicateExpr> ::= <PredicateOr>
predicateExpr        : predicateOr;

//<PredicateOr> ::= <PredicateAnd>
//                  | <PredicateOr> 'or' <PredicateAnd>
predicateOr       : predicateAnd (OR^ predicateAnd)*;
//                  : (predicateOr 'or')? predicateAnd; !!!

//<PredicateAnd> ::= <PredicateEquality>
//                   | <PredicateAnd> 'and' <PredicateEquality>
predicateAnd       : predicateEquality (AND^ predicateEquality)*;
//                   : (predicateAnd 'and')? predicateEquality; !!!

//<PredicateEquality> ::= <PredicateOperand> ComparableOperator <PredicateOperand>
predicateEquality       : predicateOperand COMPARABLEOPERATOR^ predicateOperand;

//<PredicateOperand> ::= !Identifier
//                       !| Identifier PathItem
//                        | <ObjectPath>
//                        | <Operand>
predicateOperand       : objectPath | operand;

//<Operand> ::= String | Integer | Float | Date | Parameter | Boolean
Operand       : STRING | INTEGER | FLOAT | DATE | PARAMETER | BOOLEAN;

//<ObjectPath> ::=  <PathPart>
//                | <PathPart> '/' <ObjectPath>
objectPath        : pathPart ('/' pathPart)*;
//<PathPart> ::= Identifier
//             | Identifier <Predicate>
pathPart       : IDENTIFIER predicate?;

//<FromExpr> ::=  <SimpleClassExpr>
```

```
//                  | <SimpleClassExpr> <ContainsExpr>
fromExpr        : containsExpression;

contains        : simpleClassExpr (CONTAINS^ containsExpression)?;

//! Check thislass
//<ContainsExpr>::= 'CONTAINS' <ContainsExpression>
//                      !'CONTAINS' <ContainsOr>
//<ContainsExpression> ::= <ClassExpr>
//                              | <ContainExpressionBoolean>
//                              |'(' <ContainExpressionBoolean> ')'
containsExpression      : containExpressionBool (boolOp containsExpression)?
//                              | '(' containExpressionBool ')' -> ^(OPEN containExpressionBool);

//<ContainExpressionBoolean> ::= <ContainsExpression> 'OR' <ContainsExpression>
//                                      | <ContainsExpression> 'AND' <ContainsExpression>
//                                      | <ContainsExpression> 'XOR' <ContainsExpression>
containExpressionBool           : contains
                                      | '(' containsExpression ')' -> ^(OPEN containsExpression CLOSE);

boolOp                                : OR|XOR|AND;

//<ClassExpr>::=   <SimpleClassExpr>
//                | '(' <SimpleClassExpr> <ContainsExpr> ')'
//                | <SimpleClassExpr> <ContainsExpr>
//classExpr
//                : '(' simpleClassExpr ')'
//                | simpleClassExpr
//                ;

//<SimpleClassExpr>::= Identifier                               ! RM_TYPE_NAME
//                        | Identifier Identifier               ! RM_TYPE_NAME variable
//                        | <ArchetypedClassExpr>
//                        | <VersionedClassExpr>
//                        | <VersionClassExpr>
```

```
//                    ! | <IdentifiedObjectExpr>                    ! need to be used once
VersionedClassExpr is removed
simpleClassExpr      : IDENTIFIER IDENTIFIER?
//! RM_TYPE_NAME .. RM_TYPE_NAME variable
                     | archetypedClassExpr
                     | versionedClassExpr
                     | versionClassExpr;
//                   ! | <IdentifiedObjectExpr>                    ! need to be used once
VersionedClassExpr is removed

//<ArchetypedClassExpr>::= Identifier <ArchetypePredicate>          ! RM_TYPE_NAME
[archetype_id]
//                       | Identifier Identifier <ArchetypePredicate>   ! RM_TYPE_NAME
variable [archetype_id]
archetypedClassExpr      : IDENTIFIER^ IDENTIFIER? archetypePredicate;
//! RM_TYPE_NAME variable? [archetype_id]

//! need to be used once VersionedClassExpr is removed
//!<IdentifiedObjectExpr>::= Identifier <StandardPredicate>          ! RM_TYPE_NAME
[path operator operand]
//!                       | Identifier Identifier <StandardPredicate>   ! RM_TYPE_NAME
variable [path operator operand]
//<VersionedClassExpr>::= 'VERSIONED_OBJECT'
//                       | 'VERSIONED_OBJECT' Identifier
//                       | 'VERSIONED_OBJECT' <StandardPredicate>
//                       | 'VERSIONED_OBJECT' Identifier <StandardPredicate>
versionedClassExpr       : VERSIONED_OBJECT^ IDENTIFIER? standardPredicate?;

//<VersionClassExpr>::= 'VERSION'
//                     | 'VERSION' Identifier
//                     | 'VERSION' <StandardPredicate>
//                     | 'VERSION' Identifier <StandardPredicate>
//                     | 'VERSION' <VersionPredicate>
//                     | 'VERSION' Identifier <VersionPredicate>
versionClassExpr     : VERSION^ IDENTIFIER? (standardPredicate|versionPredicate)?;
```

```
//
// LEXER PATTERNS
//

WS  :   ( ' '
        | '\t'
        | '\r'
        | '\n'
        ) {$channel=HIDDEN;};

SELECT : ('S'|'s')('E'|'e')('L'|'l')('E'|'e')('C'|'c')('T'|'t') ;
TOP : ('T'|'t')('O'|'o')('P'|'p') ;
FORWARD : ('F'|'f')('O'|'o')('R'|'r')('W'|'w')('A'|'a')('R'|'r')('D'|'d') ;
BACKWARD : ('B'|'b')('A'|'a')('C'|'c')('K'|'k')('W'|'w')('A'|'a')('R'|'r')('D'|'d') ;
AS : ('A'|'a')('S'|'s') ;
CONTAINS : ('C'|'c')('O'|'o')('N'|'n')('T'|'t')('A'|'a')('I'|'i')('N'|'n')('S'|'s') ;
WHERE : ('W'|'w')('H'|'h')('E'|'e')('R'|'r')('E'|'e') ;
ORDERBY : ('O'|'o')('R'|'r')('D'|'d')('E'|'e')('R'|'r')(' ')('B'|'b')('Y'|'y') ;
FROM : ('F'|'f')('R'|'r')('O'|'o')('M'|'m') ;
DESCENDING : ('D'|'d')('E'|'e')('S'|'s')('C'|'c')('E'|'e')('N'|'n')('D'|'d')('I'|'i')('N'|'n')('G'|'g') ;
ASCENDING : ('A'|'a')('S'|'s')('C'|'c')('E'|'e')('N'|'n')('D'|'d')('I'|'i')('N'|'n')('G'|'g') ;
DESC : ('D'|'d')('E'|'e')('S'|'s')('C'|'c') ;
ASC : ('A'|'a')('S'|'s')('C'|'c') ;
EHR : 'EHR';
AND : ('A'|'a')('N'|'n')('D'|'d') ;
OR : ('O'|'o')('R'|'r') ;
XOR : ('X'|'x')('O'|'o')('R'|'r') ;
NOT : ('N'|'n')('O'|'o')('T'|'t') ;
MATCHES : ('M'|'m')('A'|'a')('T'|'t')('C'|'c')('H'|'h')('E'|'e')('S'|'s') ;
EXISTS: ('E'|'e')('X'|'x')('I'|'i')('S'|'s')('T'|'t')('S'|'s') ;
VERSION : 'VERSION';
VERSIONED_OBJECT : 'VERSIONED_OBJECT';
ALL_VERSIONS : 'all_versions';

fragment
ESC_SEQ
```

```
    :   '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')
    |   UNICODE_ESC
    |   OCTAL_ESC;

fragment
OCTAL_ESC
    :   '\\' ('0'..'3') ('0'..'7') ('0'..'7')
    |   '\\' ('0'..'7') ('0'..'7')
    |   '\\' ('0'..'7');

fragment
UNICODE_ESC :  '\\' 'u' HEX_DIGIT HEX_DIGIT HEX_DIGIT HEX_DIGIT;

fragment
HEX_DIGIT : ('0'..'9'|'a'..'f'|'A'..'F');

QUOTE : '\'';

fragment
DIGIT : '0'..'9';

fragment
HEXCHAR : DIGIT|'a'|'A'|'b'|'B'|'c'|'C'|'d'|'D'|'e'|'E'|'f'|'F';

fragment
LETTER : 'a'..'z'|'A'..'Z';

fragment
ALPHANUM : LETTER|DIGIT;

fragment
LETTERMINUSA : 'b'..'z'|'B'..'Z';
fragment
LETTERMINUST : 'a'..'s'|'A'..'S'|'u'..'z'|'U'..'Z';
```

```
fragment
IDCHAR : ALPHANUM|'_';

fragment
IDCHARMINUST : LETTERMINUST|DIGIT|'_';

fragment
URISTRING : ALPHANUM|'_'|'-'|'/'|':'|'.'|'?'|'&'|'%'|'$'|'#'|'@'|'!'|'+'|'='|'*';

fragment
REGEXCHAR : URISTRING|'('|')'|'\\'|'^'|'{'|'}'|']'|'[';

// Terminal Definitions

//Boolean    = 'true' | 'false'
BOOLEAN : 'true' | 'false' | 'TRUE' | 'FALSE' ;

//!NodeId     = 'a"t'{Digit}{Digit}{Digit}{Digit}
//! conflict with Identifier
//!NodeId     =
'at'({Digit}{Digit}{Digit}{Digit}('.0'*('.'{NonZeroDigit}{Digit}*)+|('.'{NonZeroDigit}{Digit}*)*)|
'0''.0'*('.'{NonZeroDigit}{Digit}*)+|('.'{NonZeroDigit}{Digit}*)+)
//NodeId      = 'at'({Digit}+('.'{Digit}+)*)
//NODEID : 'at' DIGIT+ ('.' DIGIT+)*;
NODEID   : 'at' DIGIT+ ('.' DIGIT+)*; // DIGIT DIGIT DIGIT DIGIT;

//!Identifier = {Letter}({Alphanumeric}|'_')*   ! Conflicts with UID
//!Identifier = {Letter}{IdChar}*   ! Conflicts with extended NodeId
//! restricted to allow only letters after the 4th character due to conflict with extended NodeId
//!Identifier = {Letter}{IdChar}?{IdChar}?{IdChar}?({Letter}|'_')*  !Conficts with NodeId which
may have any length of digit, such as at0.9
//Identifier  = {LetterMinusA}{IdCharMinusT}?{IdChar}* |
'a"t'?(({letter}|'_')*|{LetterMinusT}{Alphanumeric}*)
// ???
IDENTIFIER : ('a'|'A') (ALPHANUM|'_')*
                    | LETTERMINUSA IDCHAR*;
```

```
//!PathItem = '/'{Letter}({Alphanumeric}|'_')*

//Integer    = {Digit}+
INTEGER : '-'? DIGIT+;

//Float    = {Digit}+'.'{Digit}+
FLOAT  : '-'? DIGIT+ '.' DIGIT+;

//Date = "{Digit}{Digit}{Digit}{Digit}'-'{Digit}{Digit}'-'{Digit}{Digit}"
DATE : '\" DIGIT DIGIT DIGIT DIGIT DIGIT DIGIT DIGIT DIGIT 'T' DIGIT DIGIT DIGIT
DIGIT DIGIT DIGIT '.' DIGIT DIGIT DIGIT '+' DIGIT DIGIT DIGIT DIGIT '\";

//!Parameter     = '$'{letter}({Alphanumeric}|'_')*
//Parameter      = '$'{letter}{IdChar}*
PARAMETER : '$' LETTER IDCHAR*;

//! could constrain UID further
//UniqueId   = {digit}+('.'{digit}+)+'.'{digit}+  ! OID
//                | {Hex Char}+('-'{Hex Char}+)+            ! UUID
UNIQUEID : DIGIT+ ('.' DIGIT+)+ '.' DIGIT+  // OID
                | HEXCHAR+ ('-' HEXCHAR+)+  // UUID;

//! could constrain ArchetypeId further
//!ArchetypeId = {Letter}+'-'{Letter}+'-'({Letter}|'_')+'.'({Letter}|'_'|'-')+'.v'{Digit}+('.'{Digit}+)?  !
not allow a number in archetype id concept, such as openEHR-EHR-OBSERVATION.laboratory-
hba1c.v1
//ArchetypeId = {Letter}+'-'{Letter}+'-'({Letter}|'_')+'.'({IdChar}|'-')+'.v'{Digit}+('.'{Digit}+)?
ARCHETYPEID : LETTER+ '-' LETTER+ '-' (LETTER|'_')+ '.' (IDCHAR|'-')+ '.v' DIGIT+ ('.'
DIGIT+)?;

//ComparableOperator              = '=' | '!=' | '>' | '>=' | '<' | '<='
COMPARABLEOPERATOR :  '=' | '!=' | '>' | '>=' | '<' | '<=';

//UriValue   = {Letter}+'://'({UriString}|'['|']'|'', '"|'')*
//                |{Letter}+':'({UriString}|'['|']'|'')*
URIVALUE: LETTER+ '://' (URISTRING|'['|']'|', \"|\'')*
```

```
//                | LETTER+ ':' (URISTRING|'['|']'|'\')*;

//RegExPattern = '{/'{RegExChar}+'/}'
REGEXPATTERN : '{/' REGEXCHAR+ '/}';

//String     = '"'{String Char}*'"'
//             | "{String Char}*"
STRING  :  '\" ( ESC_SEQ | ~('\\'|'\") )* '\"
             |  '"' ( ESC_SEQ | ~('\\'|'"') )* '";

SLASH : '/';

COMMA : ',';

OPENBRACKET : '[';

CLOSEBRACKET : ']';

OPEN : '(';
CLOSE : ')';
```

Errors:

When we run the above grammar Aql.g4, the following errors are displayed:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Purani>cd..

C:\Users>cd..

C:\>cd javalib

C:\Javalib>cd tmp

C:\Javalib\tmp>antlr4 Aql.g4

C:\Javalib\tmp>java org.antlr.v4.Tool Aql.g4
warning(202): C:\Javalib\tmp\Aql.g4:14:0: 'tokens {A; B;}' syntax is now 'tokens
 {A, B}' in ANTLR 4
error(50): C:\Javalib\tmp\Aql.g4:41:39: syntax error: '!' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:41:45: syntax error: '!' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:45:32: syntax error: '->' came as a complete su
rprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:45:35: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:50:30: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:50:27: syntax error: mismatched input '->' expe
cting SEMI while matching a rule
error(50): C:\Javalib\tmp\Aql.g4:50:36: syntax error: missing COLON at 'INTEGER'
 while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:50:43: syntax error: mismatched input ')' expec
ting SEMI while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:51:7: syntax error: missing COLON at 'INTEGER'
while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:51:27: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:51:28: syntax error: '(' came as a complete sur
prise to me while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:51:33: syntax error: missing COLON at 'INTEGER'
 while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:51:49: syntax error: extraneous input ')' expec
ting SEMI while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:55:29: syntax error: '->' came as a complete su
rprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:55:32: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:58:29: syntax error: '->' came as a complete su
rprise to me while looking for rule element
```

```
error(50): C:\Javalib\tmp\Aql.g4:58:32: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:63:19: syntax error: '!' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:71:36: syntax error: '->' came as a complete su
rprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:71:39: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:72:38: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:72:35: syntax error: mismatched input '->' expe
cting SEMI while matching a rule
error(50): C:\Javalib\tmp\Aql.g4:72:49: syntax error: missing COLON at 'identifi
edPath' while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:72:63: syntax error: extraneous input ')' expec
ting SEMI while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:84:17: syntax error: '!' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:87:17: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:96:21: syntax error: '->' came as a complete su
rprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:96:24: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:97:20: syntax error: '->' came as a complete su
rprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:97:23: syntax error: '^' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:102:25: syntax error: '->' came as a complete s
urprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:102:28: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:103:36: syntax error: '->' came as a complete s
urprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:103:39: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:104:25: syntax error: '->' came as a complete s
urprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:104:28: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:107:20: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:122:31: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:125:26: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:133:31: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:133:36: syntax error: '!' came as a complete su
rprise to me
```

```
error(50): C:\Javalib\tmp\Aql.g4:133:56: syntax error: '!' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:133:78: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:134:32: syntax error: '->' came as a complete s
urprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:134:35: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:135:13: syntax error: '!' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:135:33: syntax error: '!' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:136:13: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:152:47: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:152:44: syntax error: mismatched input '->' exp
ecting SEMI while matching a rule
error(50): C:\Javalib\tmp\Aql.g4:152:60: syntax error: missing COLON at 'predica
te' while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:152:82: syntax error: extraneous input ')' expe
cting SEMI while matching a lexer rule
error(50): C:\Javalib\tmp\Aql.g4:168:24: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:173:32: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:184:17: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:185:22: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:186:41: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:186:69: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:194:28: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:205:16: syntax error: '!' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:230:7: syntax error: '!' came as a complete sur
prise to me
error(50): C:\Javalib\tmp\Aql.g4:230:26: syntax error: '!' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:239:20: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:245:26: syntax error: '^' came as a complete su
rprise to me
```

```
error(50): C:\Javalib\tmp\Aql.g4:250:39: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:278:35: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:296:31: syntax error: '->' came as a complete s
urprise to me while looking for rule element
error(50): C:\Javalib\tmp\Aql.g4:296:34: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:324:14: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:334:20: syntax error: '^' came as a complete su
rprise to me
error(50): C:\Javalib\tmp\Aql.g4:343:11: syntax error: '^' came as a complete su
rprise to me

C:\Javalib\tmp>
```

Appendix B

This appendix contains the edited grammar of AQL language AQL.g4, BaseListener, and Translator AqlToSql.

AQL.g4

```
grammar Aql;
// Rule Definitions
Query                       :   select from where? (IDENTIFIER in OPEN
                                NODEID(COMMA NODEID)* CLOSE)?
                                orderBy? ';'? EOF;
//SELECT
select                      :   SELECT top? selectExpr;
selectExpr                  :   identifiedPathSeq;
identifiedPathSeq           :   selectVar
                            |   selectVar (COMMA selectVar)*;
selectVar                   :   identifiedPath
                            |   identifiedPath asIdentifier
                            |   identifiedPath COMMA identifiedPathSeq
                            |   identifiedPath asIdentifier COMMA
identifiedPathSeq;
identifiedPath              :   IDENTIFIER
                            |   IDENTIFIER predicate
                            |   IDENTIFIER SLASH objectPath
                            |   IDENTIFIER predicate SLASH objectPath;
predicate                   :   nodePredicate;
nodePredicate               :   OPENBRACKET nodePredicateOr
                                CLOSEBRACKET;
nodePredicateOr             :   nodePredicateAnd
                            |   nodePredicateOr OR nodePredicateAnd;
nodePredicateAnd            :   nodePredicateComparable
                            |   nodePredicateAnd AND
                                nodePredicateComparable;
nodePredicateComparable     :   NODEID
                            |   predicateOperand
                            |   NODEID COMMA STRING
```

```
                                |   NODEID COMMA PARAMETER
                                |   ARCHETYPEID
                                |   ARCHETYPEID COMMA STRING
                                |   ARCHETYPEID COMMA PARAMETER
                                |   predicateOperand comparableOperator
predicateOperand
                                |   nodePredicateRegEx;
nodePredicateRegEx              :   predicateOperand MATCHES
                                    REGEXPATTERN
                                |   REGEXPATTERN
 //! /items[{/at0001.*/}], /items[at0001 and name/value matches {//}];
predicateOperand                :   objectPath | operand;
operand                         :   operandString | operandInteger | operandFloat |
                                    operandDate | operandParameter | BOOLEAN;
operandFloat                    :   FLOAT;
operandString                   :   STRING;
operandInteger                  :   INTEGER;
operandDate                     :   DATE;
operandParameter                :   PARAMETER;
objectPath                      :   pathPart
                                |   pathPart SLASH objectPath;
pathPart                        :   IDENTIFIER
                                |   IDENTIFIER predicate;
asIdentifier                    :   AS IDENTIFIER;

//TOP
top                             :   TOP INTEGER
                                |   TOP INTEGER FORWARD
                                |   TOP INTEGER BACKWARD;

//FROM
from                            :   FROM ehrContains
                                |   FROM fromExpr;
fromExpr                        :   containsExpression;
containsExpression              :   containExpressionBool
                                |   containExpressionBool OPEN AND
```

```
                                containsExpression CLOSE
                        |       containExpressionBool OPEN OR
                                containsExpression CLOSE
                        |       containExpressionBool OPEN XOR
                                containsExpression CLOSE
                        |       containExpressionBool AND
                                containsExpression
                        |       containExpressionBool OR containsExpression
                        |       containExpressionBool XOR
                                containsExpression;
containExpressionBool   :       contains
                        |       OPEN containsExpression CLOSE;
contains                :       simpleClassExpr
                        |       CONTAINS
                        |       simpleClassExpr CONTAINS
                        |       simpleClassExpr CONTAINS
                                containsExpression;
simpleClassExpr         :       IDENTIFIER
                        |       IDENTIFIER IDENTIFIER?
                        |       archetypedClassExpr
                        |       versionedClassExpr
                        |       versionClassExpr;
archetypedClassExpr     :       IDENTIFIER archetypePredicate
                        |       IDENTIFIER IDENTIFIER archetypePredicate;
archetypePredicate      :       OPENBRACKET archetypeId
                                CLOSEBRACKET
                        |       OPENBRACKET PARAMETER
                                CLOSEBRACKET
                        |       OPENBRACKET REGEXPATTERN
                                CLOSEBRACKET;
archetypeId             :       ARCHETYPEID;
versionedClassExpr      :       VERSIONED_OBJECT
                        |       VERSIONED_OBJECT IDENTIFIER
                        |       VERSIONED_OBJECT standardPredicate
                        |       VERSIONED_OBJECT IDENTIFIER
                                standardPredicate;
```

```
standardPredicate          :    OPENBRACKET predicateExpr
                                CLOSEBRACKET;
predicateExpr              :    predicateOr;
predicateOr                :    predicateAnd
                           |    predicateOr OR predicateAnd;
predicateAnd               :    predicateEquality
                           |    predicateAnd AND predicateEquality;
predicateEquality          :    predicateOperand comparableOperator
                                predicateOperand;
comparableOperator         :    EQUALTO
                           |    NOTEQUALTO
                           |    GREATERTHAN
                           |    GREATERTHANEQUALTO
                           |    LESSTHAN
                           |    LESSTHANEQUALTO ;
versionClassExpr           :    VERSION
                           |    VERSION IDENTIFIER
                           |    VERSION standardPredicate
                           |    VERSION versionPredicate
                           |    VERSION IDENTIFIER standardPredicate
                           |    VERSION IDENTIFIER versionPredicate;
versionPredicate           :    OPENBRACKET versionPredicateOptions
                                CLOSEBRACKET;
versionPredicateOptions    :    'latest_version' | ALL_VERSIONS;
ehrContains                :    fromEHR
                           |    fromEHR CONTAINS
                           |    fromEHR CONTAINS contains
                           |    CONTAINS contains;
fromEHR                    :    EHR IDENTIFIER
                           |    EHR IDENTIFIER standardPredicate
                           |    EHR standardPredicate;

//WHERE
where                      :    WHERE identifiedExpr;
identifiedExpr             :    identifiedExprAnd
                           |    identifiedExprAnd identifiedOrOperator
```

```
                                identifiedExprAnd;
identifiedExprAnd       :       identifiedEquality
                        |       identifiedEquality (identifiedAndOperator
                                identifiedEquality)*;
identifiedOrOperator    :       OR;
identifiedAndOperator   :       AND;
identifiedEquality      :       identifiedOperand
                        |       identifiedOperand comparableOperator
identifiedOperand       |       identifiedOperand matches curlyOpen
                                matchesOperand curlyClose
                        |       identifiedOperand matches REGEXPATTERN
                        |       EXISTS identifiedPath
                        |       NOT OPEN identifiedExpr CLOSE
                        |       OPEN identifiedExpr CLOSE
                        |       NOT identifiedEquality;
matches                 :       MATCHES;
curlyOpen               :       CURLYOPEN;
curlyClose              :       CURLYCLOSE;
identifiedOperand       :       identifiedPath | operand ;
matchesOperand          :       valueListItems
                        |       URIVALUE;
valueListItems          :       operand
                        |       operand comma valueListItems;
comma                   :       COMMA;

//IN
in                      :       IN;

//ORDERBY
orderBy                 :       ORDERBY orderBySeq;
orderBySeq              :       orderByExpr
                        |       orderByExpr COMMA orderBySeq;
orderByExpr             :       identifiedPath
                        |       identifiedPath DESCENDING
                        |       identifiedPath ASCENDING
                        |       identifiedPath DESC
```

```
                    |       identifiedPath ASC;

// LEXER PATTERNS
WS                  :       [ \t\r\n]+ -> skip ;
SELECT              :       ('S'|'s')('E'|'e')('L'|'l')('E'|'e')('C'|'c')('T'|'t') ;
TOP                 :       ('T'|'t')('O'|'o')('P'|'p') ;
FORWARD             :       ('F'|'f')('O'|'o')('R'|'r')('W'|'w')('A'|'a')('R'|'r')('D'|'d') ;
BACKWARD            :       ('B'|'b')('A'|'a')('C'|'c')('K'|'k')('W'|'w')('A'|'a')('R'|'r')('D'|'d') ;
AS                  :       ('A'|'a')('S'|'s') ;
CONTAINS            :       ('C'|'c')('O'|'o')('N'|'n')('T'|'t')('A'|'a')('I'|'i')('N'|'n')('S'|'s') ;
WHERE               :       ('W'|'w')('H'|'h')('E'|'e')('R'|'r')('E'|'e') ;
IN                  :       ('I'|'i')('N'|'n') ;
ORDERBY             :       ('O'|'o')('R'|'r')('D'|'d')('E'|'e')('R'|'r')(' ')('B'|'b')('Y'|'y') ;
FROM                :       ('F'|'f')('R'|'r')('O'|'o')('M'|'m') ;
DESCENDING          :       ('D'|'d')('E'|'e')('S'|'s')('C'|'c')('E'|'e')('N'|'n')('D'|'d')('I'|'i')
                            ('N'|'n')('G'|'g') ;
ASCENDING           :       ('A'|'a')('S'|'s')('C'|'c')('E'|'e')('N'|'n')('D'|'d')('I'|'i')('N'|'n')
('G'|'g') ;
DESC                :       ('D'|'d')('E'|'e')('S'|'s')('C'|'c') ;
ASC                 :       ('A'|'a')('S'|'s')('C'|'c') ;
EHR                 :       ('E'|'e')('H'|'h')('R'|'r');
AND                 :       ('A'|'a')('N'|'n')('D'|'d') ;
OR                  :       ('O'|'o')('R'|'r') ;
XOR                 :       ('X'|'x')('O'|'o')('R'|'r') ;
NOT                 :       ('N'|'n')('O'|'o')('T'|'t') ;
MATCHES             :       ('M'|'m')('A'|'a')('T'|'t')('C'|'c')('H'|'h')('E'|'e')('S'|'s') ;
EXISTS              :       ('E'|'e')('X'|'x')('I'|'i')('S'|'s')('T'|'t')('S'|'s') ;
VERSION             :       'VERSION';
VERSIONED_OBJECT    :       'VERSIONED_OBJECT';
ALL_VERSIONS        :       'all_versions';
fragment
ESC_SEQ             :       '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')
                    |       UNICODE_ESC
                    |       OCTAL_ESC;
fragment
OCTAL_ESC           :       '\\' ('0'..'3') ('0'..'7') ('0'..'7')
```

```
                    |   '\\' ('0'..'7') ('0'..'7')
                    |   '\\' ('0'..'7');
fragment
UNICODE_ESC         :   '\\' 'u' HEX_DIGIT HEX_DIGIT HEX_DIGIT
                        HEX_DIGIT;
fragment
HEX_DIGIT           :   ('0'..'9'|'a'..'f'|'A'..'F') ;

QUOTE               :   '\'';

fragment
DIGIT               :   '0'..'9';
fragment
HEXCHAR             :    DIGIT|'a'|'A'|'b'|'B'|'c'|'C'|'d'|'D'|'e'|'E'|'f'|'F';
fragment
LETTER              :   'a'..'z'|'A'..'Z';
fragment
ALPHANUM            :   LETTER|DIGIT;
fragment
LETTERMINUSA        :   'b'..'z'|'B'..'Z';
fragment
LETTERMINUST        :   'a'..'s'|'A'..'S'|'u'..'z'|'U'..'Z';
fragment
IDCHAR              :   ALPHANUM|'_';
fragment
IDCHARMINUST        :   LETTERMINUST|DIGIT|'_';
fragment
URISTRING           :   ALPHANUM|'_'|'-'|'/'|':'|'.'|'?'|'&'|'%'|'$'|'#'|'@'|'!'|'+'|'='|'*';
fragment
REGEXCHAR           :   URISTRING|'('|')'|'\\'|'^'|'{'|'}'|']'|'[';

// Terminal Definitions
BOOLEAN             :   'true' | 'false' | 'TRUE' | 'FALSE' ;
NODEID              :   'at' DIGIT+;
IDENTIFIER          :   ('a'|'A') (ALPHANUM|'_')*
                    |   LETTERMINUSA IDCHAR*;
```

```
INTEGER                 :   '\"? '-'? DIGIT+ '\"?;
FLOAT                   :   '\"? '-'? DIGIT+ '.' DIGIT+ '\"?;
DATE                    :   '\" DIGIT DIGIT DIGIT DIGIT '-' DIGIT
                            DIGIT '-' DIGIT DIGIT 'T' DIGIT DIGIT ':' DIGIT DIGIT
                            ':' DIGIT DIGIT ',' DIGIT DIGIT DIGIT '+' DIGIT DIGIT
                            ':' DIGIT DIGIT '\";
PARAMETER               :   '$' LETTER IDCHAR*;
UNIQUEID                :   DIGIT+ ('.' DIGIT+)+ '.' DIGIT+  // OID
                        |   HEXCHAR+ ('-' HEXCHAR+)+       // UUID;
ARCHETYPEID             :   LETTER+ '-' LETTER+ '-' (LETTER|'_')+ '.'
                            (IDCHAR|'-')+ '.v' DIGIT+ ('.' DIGIT+)?  ;
EQUALTO                 :   '=' ;
NOTEQUALTO              :   '!=' ;
GREATERTHAN             :   '>' ;
GREATERTHANEQUALTO      :   '>=' ;
LESSTHAN                :   '<' ;
LESSTHANEQUALTO         :   '<=' ;
URIVALUE                :   LETTER+ '://' (URISTRING|'['|']'|', \"|\")*  ;
REGEXPATTERN            :   '{/' REGEXCHAR+ '/}';
STRING                  :   "" ( ESC_SEQ | ~('\\'|"") )* ""
                        |   '\" ( ESC_SEQ | ~('\\'|'\") )* '\"
                        |   '\"? ('a'|'A') (ALPHANUM|'_')* '\"?
                        |   '\"? LETTERMINUSA IDCHAR* '\"?;
SLASH                   :   '/';
COMMA                   :   ',';
CURLYOPEN               :   '{';
CURLYCLOSE              :   '}';
OPENBRACKET             :   '[';
CLOSEBRACKET            :   ']';
OPEN                    :   '(';
CLOSE                   :   ')';
```

Translate.java

```
// import ANTLR's runtime libraries
import org.antlr.v4.runtime.*;
import org.antlr.v4.runtime.tree.*;
public class Translate
```

```
{
        public static void main(String[] args) throws Exception
        {
                // create a CharStream that reads from standard input
                ANTLRInputStream input = new ANTLRInputStream(System.in);

                // create a lexer that feeds off of input CharStream
                AqlLexer lexer = new AqlLexer(input);

                // create a buffer of tokens pulled from the lexer
                CommonTokenStream tokens = new CommonTokenStream(lexer);

                // create a parser that feeds off the tokens buffer
                AqlParser parser = new AqlParser(tokens);
                ParseTree tree = parser.query(); // begin parsing at query rule

                // Create a generic parse tree walker that can trigger callbacks
                ParseTreeWalker walker = new ParseTreeWalker();

                // Walk the tree created during the parse, trigger callbacks
                walker.walk(new AqlToSql(), tree);

                System.out.println(); // print a \n after translation
        }
}
```

BaseListener.java

```
// Generated from Aql.g4 by ANTLR 4.0

import org.antlr.v4.runtime.ParserRuleContext;
import org.antlr.v4.runtime.Token;
import org.antlr.v4.runtime.tree.TerminalNode;
import org.antlr.v4.runtime.tree.ErrorNode;

public class AqlBaseListener implements AqlListener {
   @Override public void enterPathPart(AqlParser.PathPartContext ctx) { }
   @Override public void exitPathPart(AqlParser.PathPartContext ctx) { }
   @Override public void enterOperandInteger(AqlParser.OperandIntegerContext
 ctx) { }
   @Override public void exitOperandInteger(AqlParser.OperandIntegerContext
                                                                    ctx) { }
   @Override public void enterOrderBy(AqlParser.OrderByContext ctx) { }
   @Override public void exitOrderBy(AqlParser.OrderByContext ctx) { }
   @Override public void nterOperandParameter(AqlParser.OperandParameterContext
                                                                    ctx) {}
   @Override public void exitOperandParameter(AqlParser.OperandParameterContext
                                                                    ctx) { }
   @Override public void enterPredicate(AqlParser.PredicateContext ctx) { }
   @Override public void exitPredicate(AqlParser.PredicateContext ctx) { }
   @Override public void enterContains(AqlParser.ContainsContext ctx) { }
   @Override public void exitContains(AqlParser.ContainsContext ctx) { }
   @Override public void enterWhere(AqlParser.WhereContext ctx) { }
```

```
    @Override public void exitWhere(AqlParser.WhereContext ctx) { }
    @Override public void enterIdentifiedOperand(AqlParser.IdentifiedOperandContext
                                                                                ctx) { }
    @Override public void exitIdentifiedOperand(AqlParser.IdentifiedOperandContext
                                                                                ctx) { }
    @Override public void enterNodePredicateRegEx(AqlParser.
                                        NodePredicateRegExContext ctx) { }
    @Override public void exitNodePredicateRegEx(AqlParser.
                                                    NodePredicateRegExContext ctx) { }
    @Override public void enterMatches(AqlParser.MatchesContext ctx) { }
    @Override public void exitMatches(AqlParser.MatchesContext ctx) { }
    @Override public void enterNodePredicateOr(AqlParser.NodePredicateOrContext
                                                                                ctx) { }
    @Override public void exitNodePredicateOr(AqlParser.NodePredicateOrContext
                                                                                ctx) { }
    @Override public void enterNodePredicateComparable(AqlParser.
                                                  NodePredicateComparableContext ctx) { }
    @Override public void exitNodePredicateComparable(AqlParser.
                                                  NodePredicateComparableContext ctx) { }

     @Override public void enterSimpleClassExpr(AqlParser.SimpleClassExprContext
                                                                                ctx) { }
     @Override public void exitSimpleClassExpr(AqlParser.SimpleClassExprContext
                                                                                ctx) { }
     @Override public void enterStandardPredicate(AqlParser.StandardPredicateContext
                                                                                ctx) { }
     @Override public void exitStandardPredicate(AqlParser.StandardPredicateContext
                                                                                ctx) { }
     @Override public void enterIdentifiedPathSeq(AqlParser.IdentifiedPathSeqContext
                                                                                ctx) { }
     @Override public void exitIdentifiedPathSeq(AqlParser.IdentifiedPathSeqContext
                                                                                ctx) { }
     @Override public void enterComma(AqlParser.CommaContext ctx) { }
     @Override public void exitComma(AqlParser.CommaContext ctx) { }
     @Override public void enterComparableOperator(AqlParser.
                                                    ComparableOperatorContext
ctx) { }
    @Override public void exitComparableOperator(AqlParser.
                                                    ComparableOperatorContext
ctx) { }
    @Override public void enterCurlyClose(AqlParser.CurlyCloseContext ctx) { }
    @Override public void exitCurlyClose(AqlParser.CurlyCloseContext ctx) { }
    @Override public void enterSelectExpr(AqlParser.SelectExprContext ctx) { }
    @Override public void exitSelectExpr(AqlParser.SelectExprContext ctx) { }
    @Override public void enterSelect(AqlParser.SelectContext ctx) { }
    @Override public void exitSelect(AqlParser.SelectContext ctx) { }
    @Override public void enterOrderBySeq(AqlParser.OrderBySeqContext ctx) { }
    @Override public void exitOrderBySeq(AqlParser.OrderBySeqContext ctx) { }
    @Override public void enterObjectPath(AqlParser.ObjectPathContext ctx) { }
    @Override public void exitObjectPath(AqlParser.ObjectPathContext ctx) { }
    @Override public void enterPredicateEquality(AqlParser.PredicateEqualityContext
                                                                                ctx) { }
    @Override public void exitPredicateEquality(AqlParser.PredicateEqualityContext
                                                                                ctx) { }
```

```
    @Override public void enterOrderByExpr(AqlParser.OrderByExprContext ctx) { }
    @Override public void exitOrderByExpr(AqlParser.OrderByExprContext ctx) { }
    @Override public void enterContainsExpression(AqlParser.
ContainsExpressionContext  ctx) { }
    @Override public void exitContainsExpression(AqlParser.
ContainsExpressionContext  ctx) { }
    @Override public void enterNodePredicate(AqlParser.NodePredicateContext ctx) { }
    @Override public void exitNodePredicate(AqlParser.NodePredicateContext ctx) { }
    @Override public void enterIdentifiedExpr(AqlParser.IdentifiedExprContext ctx) { }
    @Override public void exitIdentifiedExpr(AqlParser.IdentifiedExprContext ctx) { }
    @Override public void enterVersionedClassExpr(AqlParser.
                                                  VersionedClassExprContext ctx) { }
    @Override public void exitVersionedClassExpr(AqlParser.
                                                  VersionedClassExprContext ctx) { }
    @Override public void enterIdentifiedPath(AqlParser.IdentifiedPathContext ctx) { }
    @Override public void exitIdentifiedPath(AqlParser.IdentifiedPathContext ctx) { }
    @Override public void enterAsIdentifier(AqlParser.AsIdentifierContext ctx) { }
    @Override public void exitAsIdentifier(AqlParser.AsIdentifierContext ctx) { }
    @Override public void enterIdentifiedAndOperator(AqlParser.
                                            IdentifiedAndOperatorContext ctx) { }
    @Override public void exitIdentifiedAndOperator(AqlParser.
                                            IdentifiedAndOperatorContext ctx) { }
    @Override public void enterPredicateOr(AqlParser.PredicateOrContext ctx) { }
    @Override public void exitPredicateOr(AqlParser.PredicateOrContext ctx) { }
    @Override public void enterQuery(AqlParser.QueryContext ctx) { }
    @Override public void exitQuery(AqlParser.QueryContext ctx) { }
    @Override public void enterFrom(AqlParser.FromContext ctx) { }
    @Override public void exitFrom(AqlParser.FromContext ctx) { }
    @Override public void enterSelectVar(AqlParser.SelectVarContext ctx) { }
    @Override public void exitSelectVar(AqlParser.SelectVarContext ctx) { }
    @Override public void enterValueListItems(AqlParser.ValueListItemsContext
ctx) { }
    @Override public void exitValueListItems(AqlParser.ValueListItemsContext ctx) { }
    @Override public void enterMatchesOperand(AqlParser.MatchesOperandContext
                                                                        ctx) { }
    @Override public void exitMatchesOperand(AqlParser.MatchesOperandContext
                                                                        ctx) { }
    @Override public void enterArchetypedClassExpr(AqlParser.
                                                  ArchetypedClassExprContext ctx) { }
    @Override public void exitArchetypedClassExpr(AqlParser.
                                                  ArchetypedClassExprContext ctx) { }
    @Override public void enterArchetypeId(AqlParser.ArchetypeIdContext ctx) { }
    @Override public void exitArchetypeId(AqlParser.ArchetypeIdContext ctx) { }
    @Override public void enterPredicateAnd(AqlParser.PredicateAndContext ctx) { }
    @Override public void exitPredicateAnd(AqlParser.PredicateAndContext ctx) { }
    @Override public void enterIdentifiedExprAnd(AqlParser.IdentifiedExprAndContext
                                                                        ctx) { }
    @Override public void exitIdentifiedExprAnd(AqlParser.IdentifiedExprAndContext
                                                                        ctx) { }
    @Override public void enterContainExpressionBool(AqlParser.
                                              ContainExpressionBoolContext ctx) { }
    @Override public void exitContainExpressionBool(AqlParser.
                                              ContainExpressionBoolContext ctx) { }
    @Override public void enterNodePredicateAnd(AqlParser.NodePredicateAndContext
```

```
                                                                    ctx) { }
@Override public void exitNodePredicateAnd(AqlParser.NodePredicateAndContext
                                                                    ctx) { }
@Override public void enterOperandFloat(AqlParser.OperandFloatContext ctx) { }
@Override public void exitOperandFloat(AqlParser.OperandFloatContext ctx) { }
@Override public void enterTop(AqlParser.TopContext ctx) { }
@Override public void exitTop(AqlParser.TopContext ctx) { }
@Override public void enterOperandDate(AqlParser.OperandDateContext ctx) { }
@Override public void exitOperandDate(AqlParser.OperandDateContext ctx) { }
@Override public void enterOperandString(AqlParser.OperandStringContext ctx) { }
@Override public void exitOperandString(AqlParser.OperandStringContext ctx) { }
@Override public void enterVersionClassExpr(AqlParser.VersionClassExprContext
                                                                    ctx) { }
@Override public void exitVersionClassExpr(AqlParser.VersionClassExprContext
                                                                    ctx) { }
@Override public void enterCurlyOpen(AqlParser.CurlyOpenContext ctx) { }
@Override public void exitCurlyOpen(AqlParser.CurlyOpenContext ctx) { }
@Override public void enterVersionPredicate(AqlParser.VersionPredicateContext
                                                                    ctx) { }
@Override public void exitVersionPredicate(AqlParser.VersionPredicateContext
                                                                    ctx) { }
@Override public void enterFromExpr(AqlParser.FromExprContext ctx) { }
@Override public void exitFromExpr(AqlParser.FromExprContext ctx) { }
@Override public void enterOperand(AqlParser.OperandContext ctx) { }
@Override public void exitOperand(AqlParser.OperandContext ctx) { }
@Override public void enterIn(AqlParser.InContext ctx) { }
@Override public void exitIn(AqlParser.InContext ctx) { }
@Override public void enterIdentifiedEquality(AqlParser.IdentifiedEqualityContext
                                                                    ctx) { }
@Override public void exitIdentifiedEquality(AqlParser.IdentifiedEqualityContext
                                                                    ctx) { }
@Override public void enterPredicateOperand(AqlParser.PredicateOperandContext
                                                                    ctx) { }
@Override public void exitPredicateOperand(AqlParser.PredicateOperandContext
                                                                    ctx) { }
@Override public void enterPredicateExpr(AqlParser.PredicateExprContext ctx) { }
@Override public void exitPredicateExpr(AqlParser.PredicateExprContext ctx) { }
@Override public void enterEhrContains(AqlParser.EhrContainsContext ctx) { }
@Override public void exitEhrContains(AqlParser.EhrContainsContext ctx) { }
@Override public void enterArchetypePredicate(AqlParser.
                                                  ArchetypePredicateContext ctx) { }
@Override public void exitArchetypePredicate(AqlParser.
                                                ArchetypePredicateContext ctx) { }
@Override public void enterIdentifiedOrOperator(AqlParser.
                                                  IdentifiedOrOperatorContext ctx) { }
@Override public void exitIdentifiedOrOperator(AqlParser.
                                                  IdentifiedOrOperatorContext ctx) { }
@Override public void enterFromEHR(AqlParser.FromEHRContext ctx) { }
@Override public void exitFromEHR(AqlParser.FromEHRContext ctx) { }
@Override public void enterVersionPredicateOptions(AqlParser.
                                                  VersionPredicateOptionsContext ctx) { }
@Override public void exitVersionPredicateOptions(AqlParser.
                                                  VersionPredicateOptionsContext ctx) { }
@Override public void enterEveryRule(ParserRuleContext ctx) { }
```

```
   @Override public void exitEveryRule(ParserRuleContext ctx) { }
   @Override public void visitTerminal(TerminalNode node) { }
   @Override public void visitErrorNode(ErrorNode node) { }
}
```

AqlToSql.java

```
/** Convert Aql to Sql */

import org.stringtemplate.v4.*;
import org.antlr.v4.runtime.TokenStream;
import org.antlr.v4.runtime.misc.Interval;
import java.lang.Object;
import java.util.*;
import java.util.HashMap;
import java.util.Stack;

public class AqlToSql extends AqlBaseListener
{
     String selectStatement = "";
     String fromStatement = "";
     String whereStatement = "";

     Map<String,String> map = new HashMap<String, String>();
     Stack<String> stack = new Stack<String>();

     public AqlToSql()
     {

   map.put("eval/data[at0001]/items[at0019]/items[at0002]/value/value","PV351.status");
   map.put("eval/data[at0001]/items[at0019]/items[at0003]/value/value","PV353.status");
   map.put("eval/data[at0001]/items[at0019]/items[at0004]/value/value","PV355.status");
   map.put("eval/data[at0001]/items[at0019]/items[at0006]/value/value","PV357.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0011]/value/value","PV360.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0012]/value/value",
                                                                        "PV500.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0013]/value/value",
                                                                        "PV502.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0014]/value/value",
                                                                        "PV504.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0015]/value/value",
                                                                        "PV506.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0016]/value/value",
                                                                        "PV508.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0017]/value/value",
                                                                        "PV510.status");
   map.put("eval/data[at0001]/items[at0020]/items[at0069]/items[at0018]/value/value",
                                                                        "PV512.status");
   map.put("eval/data[at0001]/items[at0022]/items[at0023]/value/value","PV370.status");
   map.put("eval/data[at0001]/items[at0022]/items[at0070]/items[at0024]/value/value",
                                                                        "PV373.status");
   map.put("eval/data[at0001]/items[at0022]/items[at0070]/items[at0025]/value/value",
                                                                      "PV375.status");
```

```
map.put("eval/data[at0001]/items[at0022]/items[at0070]/items[at0026]/value/value",
                                                                    "PV377.status");
map.put("eval/data[at0001]/items[at0022]/items[at0070]/items[at0027]/value/value",
                                                                    "PV379.status");
map.put("eval/data[at0001]/items[at0022]/items[at0070]/items[at0028]/value/value",
                                                                    "PV381.status");
map.put("eval/data[at0001]/items[at0022]/items[at0070]/items[at0029]/value/value",
                                                                    "PV383.status");
map.put("c1/items[at0041]/items[at0049]/value/defining_code/code_string",
                                                                      "PV529.atid");
map.put("c1/items[at0041]/items[at0050]/value/defining_code/code_string",
                                                                      "PV497.atid");
map.put("c1/items[at0041]/items[at0051]/value/defining_code/code_string",
                                                                      "PV568.atid");
map.put("c1/items[at0043]/value/defining_code/code_string","PV487.atid");
map.put("c1/items[at0044]/value/defining_code/code_string","PV533.atid");
map.put("c1/ items[at0045]/value/value","PV407.status");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/
          items[at0033]/items[at0034]/value/defining_code/code_string","PV160.atid");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/
             items[at0034]","substring_index(P5.dewey_id, '.', 2) in (select
             substring_index(P343.dewey_id, '.', 2) from P343)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0033]/
                items[at0035]","substring_index(P5.dewey_id, '.', 2) in (select
                substring_index(P1271.dewey_id, '.', 2) from P1271)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/

                                                items[at0055]/value/value","PV149.status");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/
                                                items[at0056]/value/value","PV151.status");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/
                                                items[at0057]/value/value","PV153.status");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0085]/items[at0054]/
                                                items[at0058]/value/value","PV155.status");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value",
  "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P667.dewey_id, '.', 2)
                                                                      from P667)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0004]/null_flavour",
 "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1095.dewey_id, '.', 2)
                                                                    from P1095)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0008]/null_flavour",
   "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P681.dewey_id, '.', 2)
                                                                      from P681)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0009]/null_flavour",
   "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P690.dewey_id, '.', 2)
                                                                      from P690)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0010]/null_flavour",
   "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P699.dewey_id, '.', 2)
                                                                      from P699)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0011]/null_flavour",
   "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P708.dewey_id, '.', 2)
                                                                      from P708)");
map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0012]/null_flavour",
   "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P717.dewey_id, '.', 2)
```

```
                                                                    from P717)");
    map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0013]/null_flavour",
      "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P726.dewey_id, '.', 2)
                                                                    from P726)");
    map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0014]/null_flavour",
      "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P735.dewey_id, '.', 2)
                                                                    from P735)");
    map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0015]/null_flavour",
      "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P744.dewey_id, '.', 2)
                                                                    from P744)");
    map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0016]/null_flavour",
      "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P753.dewey_id, '.', 2)
                                                                    from P753)");
    map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0021]/null_flavour",
      "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P762.dewey_id, '.', 2)
                                                                    from P762)");
    map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0022]/null_flavour",
     "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1110.dewey_id, '.', 2)
                                                                   from P1110)");
    map.put("obs/data[at0001]/events[at0002]/data[at0003]/items[at0069]/null_flavour",
     "substring_index(P5.dewey_id, '.', 2) in (select substring_index(P1295.dewey_id, '.', 2)
                                                                    from P1295)");
}

  @Override
  public void enterSelect(AqlParser.SelectContext ctx)
  {
selectStatement += (ctx.SELECT().getText()+"\t");
  }

  @Override
  public void enterSelectVar(AqlParser.SelectVarContext ctx)
  {
          if(ctx.identifiedPath() != null)
          {
                  String selectPath = ctx.identifiedPath().getText();
                  if (selectPath.equals("e/ehr_id/value"))
                  {
                          selectStatement += ("substring(P5.dewey_id, '.', 2)");
                  }
          }
if(ctx.asIdentifier() != null)
          {
                  selectStatement += ("");
          }
          if(ctx.COMMA() != null)
          {
                  selectStatement += (ctx.COMMA().getText()+"\t");
          }
 }

 @Override
 public void enterFrom(AqlParser.FromContext ctx)
 {
```

```
        fromStatement += ("\n" + ctx.FROM().getText()+"\t");
 }

 @Override
 public void enterEhrContains(AqlParser.EhrContainsContext ctx)
 {
        stack.push("ehrContains");
        if(ctx.fromEHR() != null)
        {
                String fromehr = ctx.fromEHR().getText();
                if (fromehr.equals("Ehre") || fromehr.equals("EHRe") ||
                                                                fromehr.equals("ehre"))
                {
                        fromStatement += ("P5\t");
                        fromIndex = fromStatement.length();
                }
        }
        if(ctx.contains() != null)
        {
                if (ctx.contains().getText().contains("cito"))
                {
                        map.put("c/context/start_time/value","PV275.date_time_zone");
                }
                else if (ctx.contains().getText().contains("histo"))
                {
                        map.put("c/context/start_time/value","PV101.date_time_zone");
                }
        }
 }

 @Override
 public void exitEhrContains(AqlParser.EhrContainsContext ctx)
 {
        stack.pop();
 }

 @Override
 public void enterWhere(AqlParser.WhereContext ctx)
 {
        whereStatement += ("\n"+ ctx.WHERE().getText()+"\t");
 }

 boolean nullRemoved = false;              //to remove the null after from
 @Override
 public void enterIdentifiedEquality(AqlParser.IdentifiedEqualityContext ctx)
 {
        stack.push("identifiedEquality");
        if (!nullRemoved && ctx.EXISTS() != null){
                nullRemoved=true;
        }
        if (ctx.EXISTS() == null)
stack.push("notExistsIdentifiedEquality");
        if(ctx.identifiedPath() != null)
        {
```

```
                String path = (map.get(ctx.identifiedPath().getText())+"\t");
                whereStatement += path;
        }
        if(ctx.OPEN() != null)
        {
                whereStatement += (ctx.OPEN().getText());
        }
        if(ctx.EXISTS() != null)
        {
                whereStatement += (" ");
        }
}

@Override
public void exitIdentifiedEquality(AqlParser.IdentifiedEqualityContext ctx)
{
        if(ctx.CLOSE() != null)
        {
                whereStatement += (ctx.CLOSE().getText()+"\t");
        }
        if (ctx.EXISTS() == null)
stack.pop();
        stack.pop();
}

@Override
public void enterIdentifiedOrOperator(AqlParser.IdentifiedOrOperatorContext ctx)
{
        if(ctx.OR() != null)
        {
                whereStatement += (ctx.OR().getText()+"\n");
        }
}

@Override
public void enterIdentifiedAndOperator(AqlParser.IdentifiedAndOperatorContext ctx)
{
        if(ctx.AND() != null)
        {
                whereStatement += (ctx.AND().getText()+"\n");
        }
}

HashMap<String,Integer> joinMap = new HashMap<String,Integer>();
@Override
public void enterIdentifiedOperand(AqlParser.IdentifiedOperandContext ctx)
{
        if(ctx.identifiedPath() != null)
        {
                String path = (map.get(ctx.identifiedPath().getText()));
                String spath = path;
                whereStatement += path + "\t";
                path = path.substring(0,path.indexOf('.'));
                if (joinMap.get(path) == null) {
```

```
                        joinMap.put(path,0);
                        fromStatement +=
                                ("  join "
                                + path
                                + " on substring_index(P5.dewey_id, \'.\', 2) =
                                                                substring_index("
                                + path
                                + ".dewey_id, '.', 2)\t");

                        selectStatement +=
                                (", substring ("
                                + path
                                + ".dewey_id, '.', 2), "
                                + spath);
                }
        }
}

@Override
public void enterComparableOperator(AqlParser.ComparableOperatorContext ctx)
{
        if (stack.contains("notExistsIdentifiedEquality"))
        {
                whereStatement += (ctx.getText()+"\t");
        }
}

@Override
public void enterOperand(AqlParser.OperandContext ctx)
{
        if(ctx.operandDate() != null)
        {
                whereStatement += (ctx.operandDate().getText()+"\t");
        }
        if(ctx.operandInteger() != null)
        {
                whereStatement += (ctx.operandInteger().getText()+"\t");
        }
        if(ctx.operandString() != null)
        {
                if(stack.contains("notExistsIdentifiedEquality"))
                {
                        whereStatement += (ctx.operandString().getText());
                }
        }
}

@Override
public void enterNodePredicateComparable(AqlParser.
                                                NodePredicateComparableContext ctx)
{
        stack.push("nodePredicateComparable");
}
```

```
@Override
public void exitNodePredicateComparable(AqlParser.
                                                    NodePredicateComparableContext ctx)
{
        stack.pop();
}

@Override
public void enterComma(AqlParser.CommaContext ctx)
{
        whereStatement += (",\t");
}

@Override
public void enterCurlyOpen(AqlParser.CurlyOpenContext ctx)
{
        whereStatement += ("(");
}

@Override
public void enterCurlyClose(AqlParser.CurlyCloseContext ctx)
{
        whereStatement += (")");
}

@Override
public void enterMatches(AqlParser.MatchesContext ctx)
{
        whereStatement += ("IN\t");
}

@Override
public void exitQuery(AqlParser.QueryContext ctx)
{
        System.out.println(selectStatement+fromStatement+whereStatement);
}
}
```