LDX User Technical Guide

LDX (Language for Data eXploration) is a specification language that extends Tregex, a query language for tree-structured data. It allows you to partially specify structural properties of a tree, as well as the nodes' labels. The language is especially useful for specifying the order of notebook's query operations and their type and parameters.

1 Hello World LDX Example

The following LDX query describe a simple exploratory session with two analytical operations: (1) a group-by and (2) a filter. We further specify that the filter is to be performed on the same attribute as the group-by. The rest of the parameters are *unspecified*, and will be completed using the ANON-SYS CDRL engine. A tree illustration of the query is depicted in Figure 1.

```
ROOT CHILDREN <A,B>
    A LIKE [G,(?<X>.*),.*]
    B LIKE [F,(?<X>.*),.*]
```

In the query, the `ROOT` node represent the raw dataset, and its two children `A` and `B` – the analytical filter and group-by operations. A filter operation is generally specified

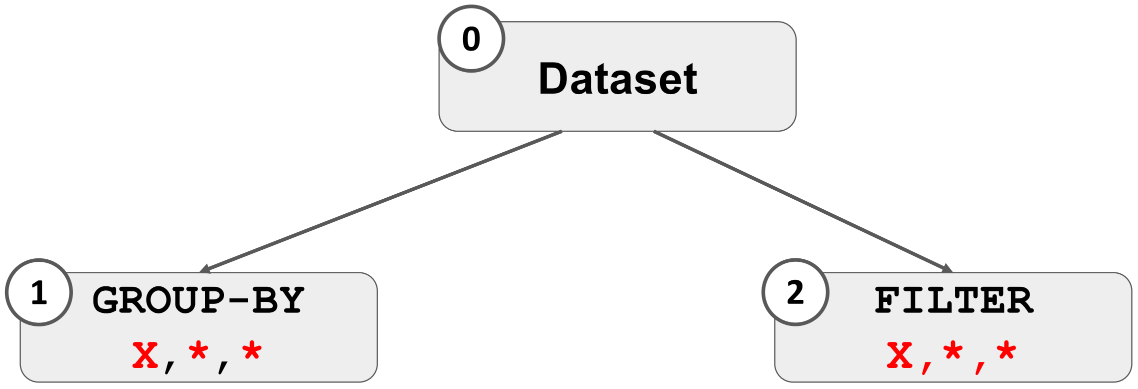

Figure 1: "Hello World" Query – Tree Representation

by `[F,attr,op,term]` and a group-by via `[G,g_attr,agg_func,agg_attr]`. In most LDX queries, the operations are unspefied or partially specified. In this case:

`A` is a group-by with unspecified parameters (using RegEx .* syntax), and `B` is a filter operation, also with unspecified parameters (to be instantiated by the ANON-SYS CDRL engine). Using a continuity variable `X` we further specify that the group-by attribute should be the same as the one used in the filter.

We next explain the syntax of LDX in more detail, focusing on: *structural specifications*, *operational specifications*, and *continuity variables*

2 Structural Specifications

Structural specifications connects the named node to other nodes in the tree. This is done by combining regular expressions together with tree-structure primitives such as `CHILDREN`, `DESCENDANTS`, and `SIBLINGS`. For instance, `'A CHILDREN <B,+>'` states that the named-node `A` has a (named) child `B` and at least one more unnamed children.

Recall that the fact that `B` is a child of `A` not only means that Operation `B` was executed after Operation `A`, but also that `B` is employed on the results of Operation `A` (i.e., rather than on the original dataset). Last, since `B` is a named-node, it can take its own set of structural/operational specifications, and be connected to other named-node via the continuity variables, as described next.

Here are some examples for Structural relationships between nodes:

Children Relationship:

`Expression: A CHILDREN <B,C>`

Description: B and C are the only children of A in the specified order.

Siblings Relationship:

`Expression: A SIBLINGS <B,C>`

Description: B and C are siblings of A.

Descendants Relationship:

`Expression: A DESCENDANTS <B,C>`

Description: B and C are descendants of A.

Unordered Relationship:

`Expression: A DESCENDANTS {B,C}`

Description: B and C are descendants of A, not necessarily in any specific order.

Relationship With Additional Unnamed Nodes:

`Expression: A DESCENDANTS {B,C, *}`

Description: B and C are descendants of A, in no particular order, with potentially more unnamed descendants.

3 Operational Specifications

Operations in LDX are used to define actions performed on the nodes using the LIKE operator. Nodes operations can be categorized into two types: simple operations

Expression	Type	Description
A CHILDREN <B,C>	Structure	B and C are the only (ordered) children of A
A SIBLINGS {B,C,*}	Structure	B and C are siblings of A (unordered), and there may be more, unnamed ones
A DESCENDANTS {B,C *}	Structure	B and C are two of the (unordered) descendants of A
A LIKE [G,.*,AVERAGE,.*]	Operational	A is a group-by operation on *some* column employing *average* on *some* column
A LIKE [F,category,eq|ne,.*comedy.*]	Operational	A is an equality/inequality filter on ‘category’, where the filter term includes the string ‘comedy’
A LIKE [F,(?<col>.*),.*] B LIKE [G,(?<col>.*),.*]	Continuity	A is a filter operation on *some* column, and B is group-by on the *same* column
A LIKE [G,.*,(?<func>.*),(?<col>.*)] B LIKE [G,.*,(?<func>.*),(?<col>.*)]	Continuity	A and B are group-by operations with the same aggregation function and column
A LIKE [F,(?<col>.*delay.*),ge,(?<term>[0-9]{3,4}) B LIKE [F,(?<col>.*),le,(?<term>.*)]	Continuity	A is a filter operation on *some* column that includes the word ‘delay’ greater equal *some* value between 100 to 1000 and B is the same action but with lower equal

Table 1: Example LDX Expressions

and special operations.

Regular Filter:

> `Expression: A LIKE [F,category,ne,.*]`
>
> Description: A is a non-equality filter on 'category', where the filter term is some term.

Regular Group-By:

> `Expression: A LIKE [G,.*,AVERAGE,.*]`
>
> Description: A is a group-by operation on some column, employing average on some column.

4 Contextual specifications Using Continuity Variables

Structural and operational specifications allow to *explicitly* constrain the operations' parameters, input data and order of execution. We next introduce the continuity variables in LDX, which allows constructing more complex specifications that *semantically* connect between operations' *unspecified* parameters.

LDX allows this using named-groups syntax. Yet differently than standard regular expressions, which only allow "capturing" a specific part of the string, in LDX these variables are used to constrain the operations in subsequent nodes. For instance, the statement '`B1 LIKE [F,'country',eq,.*]`' specifies that the operation is an *equality filter on the attribute 'country', where the filter term is free.* To capture the filter term in a continuity variable we use named-groups syntax: '`B1 LIKE [F,'country',eq,(?<CNTRY>.*)]`' – in which the free filter term (`.*`) is captured into the variable `CNTRY`. Using this variable in subsequent operation specifications will restrict them to the same filter term (despite the fact that the term is not explicitly specified). For instance, a subsequent specification is ‘`B2 LIKE [F,'country',neq,(?<CNTRY>.*)]`’, indicating that the next filter should focus on all *other* countries than the one specified in the previous query operation. Refer to Table 1 for additional use cases of continuity variables.

Group-By Using Continuity Variable:

```
Expression:
    A LIKE [G,.*,(?<func>.*),(?<col>.*)]
    B LIKE [G,.*,(?<func>.*),(?<col>.*)]
```

Description: This example uses a continuity variable which is the same as ’.*’ but also stores the value. In this example, A and B are group-by operations with the same aggregation function and column.

```
Expression:
    A LIKE [F,(?<col>.*),.*]
    B LIKE [G,(?<col>.*),.*]
```

Description: A is a filter operation on some column, and B is a group-by operation on the **same** column.

Advanced Example:

```
Expression:
    A LIKE [F,(?<col>.*delay.*),ge,(?<term>[0-9]{3,4})]
    B LIKE [F,(?<col>.*),le,(?<term>.*)]
```

Description: This is a more complex example that uses regex and continuity variables. A is a filter operation on some column that includes the word ’delay’ and is greater than or equal to some value between 100 to 1000. B is the same action but with less than or equal condition.