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 LINX 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
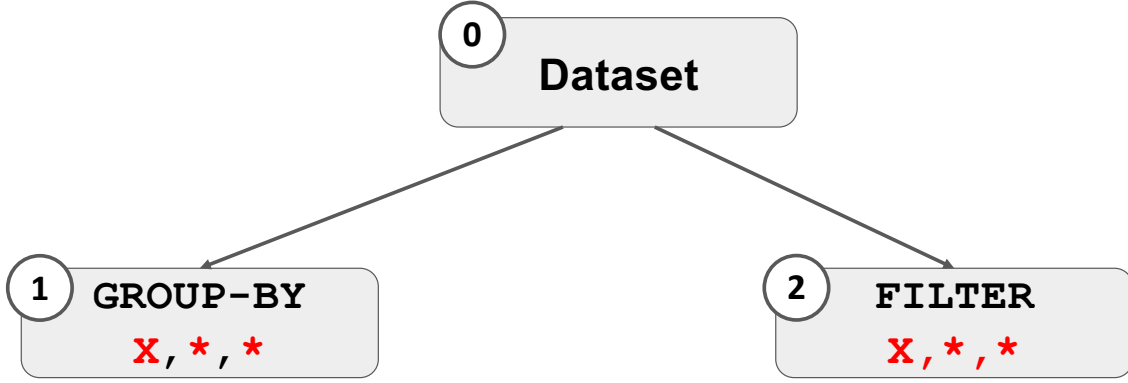
Figure 1: "Hello World" Query – Tree Representation

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 LINX 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.

5   Full LDX queries examples

We can now describe the LDX queries, composed to fit the exploration needs of "atypical country" and "successful TV shows", as described in our running example.

Q_1: **"Atypical country"**. The goal of the "atypical country" notebook is to point out a specific country that shows different trends or patterns compared to the rest of the world. The corresponding LDX query Q_1 is depicted in Figure 2. Q_1 is composed of two sub-trees, `A1,B1,B2` (Lines 2-4) and `A2,C1,C2` (Lines 5-7) with identical specifications. Each sub-tree forms a "comparison" of a country to the rest of the world, w.r.t. a dataset attribute. The first sub-tree is specified as follows: `A1` is a group-by operation (with unspecified parameters) with two immediate children `B1` and `B2` (Line 2). `B1` and `B2` specify two subsequent filter operations on the attribute *'country'*, where the unspecified filter term is captured by the continuity variable `CNTRY` (Lines 3-4). This enforces that both `B1` and `B2` operate on the same term, while `B1` filters *in* and `B2` filters *out*. The rest of the query (Lines 5-7) specifies the sub-tree `A2,C1,C2` which have the same specifications as `A1,B1,B2`.

Q_2: **"Successful TV Shows"**. Recall that the requirement from the second exploratory session is to explore interesting properties of TV shows with more than one season. This reflects in the corresponding LDX query Q_2 (Figure 3), which first filters the Netflix data on TV shows with at least two seasons (Lines 2-3), then specifies two pairs of "sibling" group-by and filter operations (`C1,C2`, and `D1, D2`). Each such pair is contextually connected using a continuity variable: `COL1` enforces that the group-by attribute in `C1` is the same as in the subsequent filter `C2` (`COL2` enforces the same for `D1` and `D2`).

```
1    ROOT CHILDREN {A1, A2}

2        A1 LIKE [G,.*] and CHILDREN <B1, B2>

3            B1 LIKE [F,'country',eq,(?<X>.*)]

4            B2 LIKE [F,'country',ne,(?<X>.*)]

5        A2 LIKE [G,.*] and CHILDREN <C1, C2>

6            C1 LIKE [F,'country',eq,(?<X>.*)]

7            C2 LIKE [F,'country',ne,(?<X>.*)]
```

Figure 2: Q_1: "Atypical Country" LDX Specifications

```
1    ROOT CHILDREN <A1>

2        A1 LIKE [F,'type',eq,"TV Show"] and CHILDREN <B1>

3            B1 LIKE [F,'duration',ge,2] and CHILDREN <C1, C2>

4                C1 LIKE [G,(?<col1>.*),.*]

5                C2 LIKE [F,(?<col1>.*),.*] and CHILDREN <D1, D2>

6                    D1 LIKE [G,(?<col2>.*),.*]

7                    D2 LIKE [F,(?<col2>.*),.*] and CHILDREN {*}
```

Figure 3: Q_2: "Successful TV Shows" LDX Specifications