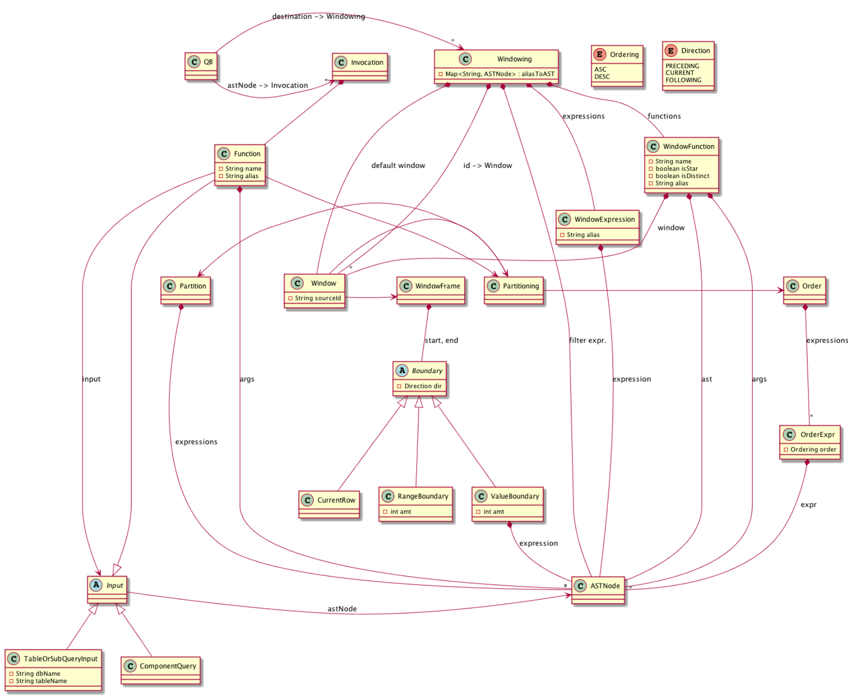
PTF Data Structures

Harish Butani, Parjakta Kalmegh

January 23, 2013

1 PTF Specification Classes



- The class names in the picture have the prefix ‘PTF’ and suffix ‘Spec’ removed. So for e.g. class *Invocation* in code is *PTFInvocationSpec*.

- Consider the following Query. In the rest of this section we document details about the Specification classes; where possible we relate the information to this e.g.

```
select p_mfgr, p_name, p_size,
rank() as r,
denserank() as dr,
sum(p_retailprice) as s1 over (rows between unbounded preceding and current row),
sum(p_retailprice) over (w1)
from noop(part
distribute by p_mfgr
sort by p_name)
having r < 4
window w1 as rows between 2 preceding and 2 following
```

This query operates on the output of a PTF(noop); the input to the PTF is the part table; it is divided into Partitions by the Manufacturer and each Partition is sorted by Part Name. The output of the PTF is processed by a set of Aggregation expressions; some of which have a Window specified.

1.1 QueryBlock

- The QueryBlock data structure is extended to hold onto information about the PTF Invocations in the *from* clause and also any 'Windowing' expressions in the *Select List*.
- For each PTF Invocation in the from clause a *Invocation* object is created. This is held in a Map in the QB that maps the root of the ASTNode for this Invocation to the corresponding Invocation object.
- A Query Block may have multiple *destinations*. From the Select List of each destination, we extract the Window Function and Windowing Expression invocations and hold them in a *Windowing* object. This held in the QB as a map from the destination to the Windowing object.
- In our E.g. an Invocation object is created for the Noop invocation in the from clause; and a Windowing object holds all the Select Expressions that have UDAF invocations. It also holds the *Having* expression(more on this later).

1.2 Invocation

An Invocation object captures the calling of a PTF in the Query. A PTF invocation can appear anywhere a table/subquery can. So a QB may have multiple Invocations. PTFs can be chained; so the input to a PTF can be

another PTF call; so an Invocation represents a PTF chain. The Invocation points to the first (top most) Function in the chain.

1.3 Input

A PTF Input represents the input to a PTF Function. An Input can be a Hive SubQuery or Table or another PTF Function. An Input instance captures the ASTNode that this instance was created from.

1.3.1 TableOrSubQuery

A PTF input that represents a table reference or SubQuery in the Query. In our e.g. the refernce to 'part' table in the Noop PTF invocation will be held in an instance of TableOrSubQuery. For a table the dbName and tableName are extracted from the AST tree. For a SubQuery the alias is extracted from the alias AST.

1.3.2 Component Query

This object is created during Query Componentization. If a PTF chain requires execution by multiple PTF Operators; then the original Invocation object is decomposed into a set of Component Invocations. Every component Invocation but the first one ends in a ComponentQuery instance. During the construction of the Operator plan a ComponentQuery object in the chain implies connect the PTF Operator to the 'input' i.e. has been generated so far.

1.4 Function

Represents a PTF Invocation. Captures:

- function name and alias
- the *Partitioning* details about its input
- its arguments. The ASTNodes representing the arguments are captured here.
- a reference to its Input.

1.5 Partitioning

Captures how the Input to a PTF Function should be partitioned and ordered. Refers to a *Partition* and *Order* instance.

1.5.1 Partition

Captures how an Input should be Partitioned. This is captured as a list of ASTNodes that are the expressions in the Distribute/Cluster by clause specifying the partitioning applied for a PTF invocation.

1.5.2 Order

Captures how the Input should be Ordered. This is captured as a list of ASTNodes that are the expressions in the Sort By clause in a PTF invocation.

1.6 Window

It represents a WindowFrame applied to a Partitioning. A Window can refer to a *source* Window by name. The source Window provides the basis for this Window definition. This Window specification extends/overrides the *source* Window definition. In our e.g. the Select Expression $sum(p_retailprice)over(w1)$ is translated into a WindowFunction instance that has a Window specification that refers to the global Window Specification 'w1'. The Function's specification has no content, but inheirts all its attributes from 'w1' during subsequent phases of translation.

1.6.1 WindowFrame

A WindowFrame specifies the Range on which a Window Function should be applied for the 'current' row. Its is specified by a *start* and *end* Boundary.

1.6.2 Boundary

A Boundary specifies how many rows back/forward a WindowFrame extends from the current row. A Boundary is specified as:

Range Boundary as the number of rows to go forward or back from the Current Row.

Current Row which implies the Boundary is at the current row.

Value Boundary which is specified as the amount the value of an Expression must decrease/increase

1.7 Windowing

Captures the Window processing specified in a Query. A Query may contain:

- UDAF invocations on a Window.
- Lead/Lag function invocations that can only be evaluated in a Partition.
- For Queries that don't have a Group By all UDAF invocations are treated as Window Function invocations.
- For Queries that don't have a Group By, the Having condition is handled as a post processing on the rows output by Windowing processing.

Windowing is a container of all the Select Expressions that are to be handled by Windowing. These are held in 2 lists: the functions list holds WindowFunction invocations; the expressions list holds Select Expressions having Lead/Lag function calls. It may also contain an ASTNode representing the post filter to apply on the output of Window Functions. Windowing also contains all the Windows defined in the Query. One of the Windows is designated as the 'default' Window. If the Query has a Distribute By/Cluster By clause; then the information in these clauses is captured as a Partitioning and used as the default Window for the Query. Otherwise the first Window specified is treated as the default. Finally Windowing maintains a Map from an 'alias' to the ASTNode that represents the Select Expression that was translated to a Window Function invocation or a Window Expression. This is used when building RowResolvers.

1.7.1 Window Function

Represents a UDAF invocation in the context of a Window Frame. As explained above sometimes UDAFs will be handled as Window Functions even w/o an explicit Window specification. This is to support Queries that have no Group By clause. A Window Function invocation captures:

- the ASTNode that represents this invocation
- its name
- whether it is star/distinct invocation.
- its alias
- and an optional Window specification

1.7.2 Window Expression

Represents a Select Expression in the context of Windowing. These can refer to the output of Windowing Functions and can navigate the Partition using Lead/Lag functions.