Join(smartIterator<T>, smartIterator<T>, PredicateTree, #block, JoinType)
enum JoinType {inner, left, right, full}

Outer joins can be done either by:
1, first finishing the inner join, then add further tuples to the join result;
2, modifying the join algorithms. For nested loop join, consider each tuple in the target iterator
during each comparison. For merge join, we can output each of the further tuples during the
merge step.

In our DBMS, the JoinType will determine the methods inside the Join. The Join method returns
a smartIterator<T>.
Smarterator is the combined idea of iterator and temporary relation. It is designed to efficiently hold or store the relation in order to support lazy fetching for query processor and pipelining in our DBMS. Here is the API for smarterator:

\[\text{SI}<T> \{ \]
\[\text{SI}<T>(\text{relationID}) \{ \]
\[\text{SI}<T>(\text{stepFunction}) \{ \]

\[\text{List<attrID>} \text{ attrList};\]
\[\text{isOnDisk()} \{ \]
\[\text{isSorted()} \{ \]
\[\text{setSorted()} \{ \]
\[\text{hasNext()} \{ \]
\[\text{next()} \{ \text{stepFunction()} \}
\]

\[\text{SI}<T>(\text{relationID})\]
Constructor with relationID for this smarterator.

\[\text{SI}<T>(\text{stepFunction})\]
Constructor with the step function. We use step functions in order to support pipelining. Each step function contains a tuple associated with one specific operation type, and different types of operations (eg, projection, set operation) contain different settings for that operation. Check Operator APIs for details.

\[\text{List<attrID>} \text{ attrList};\]
A list of attribute, which is used for isSorted() and setSorted() to indicate the smaterator has been sorted on those attributes.

\[\text{isSorted()} \{ \]
\[\text{setSorted()} \{ \]
Methods on the attribute list when checking the sort condition and setting the sorted attributes.

\[\text{isOnDisk()} \{ \]
Return boolean indicating whether the current smaterator is on disk or not.

\[\text{hasNext()} \{ \]
Return boolean indicating whether we reach the end of the current relation.

\[\text{next()} \{ \text{stepFunction()} \}
This will move forward and return the next tuple with the form of a step function.
In order to support the query optimization (Query Optimizer), we should add the following fields for smaterator:

- Int Priority;
- PolicyType Policy;
- Boolean SaveOnNext;

These will be set and get by operations during the optimization. The Priority and Policy will also be passed to buffer manager as the data information inside the stepFunction.