Databend

Partition Elimination in Databend

zhyass@datafuselabs

目录
CONTENT

1 TableSnapshot structure
2 Optimize before Partition Elimination
3 Build Verifiable Expression
4 WIP & TODO

Databend

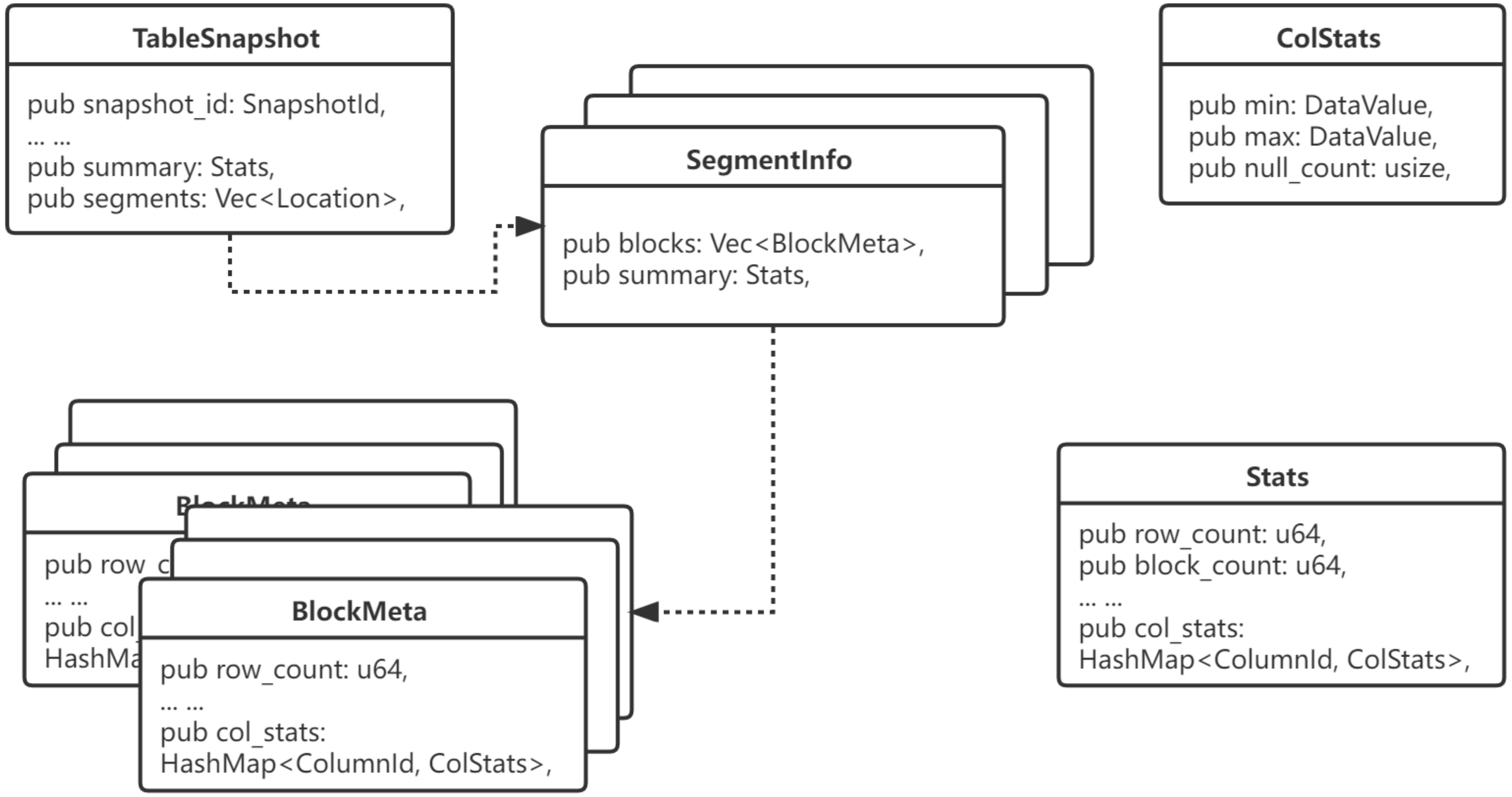
Databend

1 TableSnapshot Structure

Partition Elimination in Databend

TableSnapshot Structure

WWW.DATABEND.COM



https://github.com/datafuselabs/databend/blob/main/query/src/datasources/table/fuse/meta/table_snapshot.rs

Databend

2 Optimize before Partition Elimination

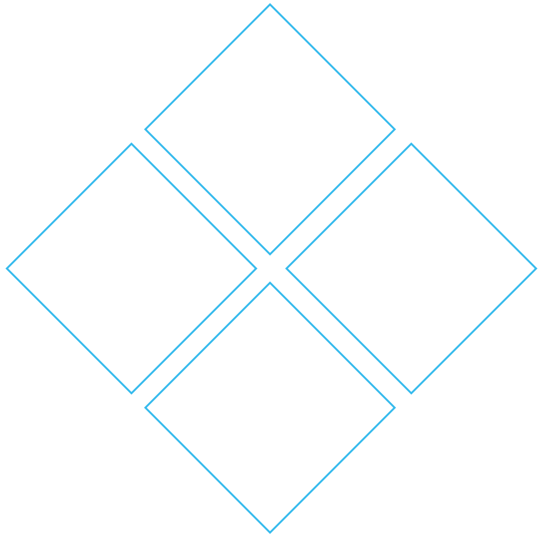
Partition Elimination in Databend

Optimize before Partition Elimination

1 Constant Folding

Computes value of constant expression

e.g. a>1+3 => a>4



2 Remove Constant Condition

e.g.

true and something => something

false and something => false

true or something => true

false or something => something

3 Transform to Boolean Expression

Ensure that all expressions involved in conditions are boolean expressions.

e.g.

<non-bool-expr> => (<non-bool-expr> != 0)

4 Inverse Not Expression

e.g.

NOT(a AND b AND ...) => NOT(a) OR NOT(b) OR ...

NOT(a OR b OR ...) => NOT(a) AND NOT(b) AND ...

NOT(a = b) => a != b

NOT(a !=b) => a = b

NOT(NOT(a)) => a

... ...

Databend

3 Build Verifiable Expression

Partition Elimination in Databend

Build Verifiable Expression

A verifiable expression which is a boolean expression derived from the query filters.
For a given block([min, max]), if the verifiable expression evaluates to true, that block need to be scanned by the original query.

Criterias
➢Complexity of expression
➢Tightness

Section 5.2 of http://vldb.org/pvldb/vol14/p3083-edara.pdf
Code: databend/range_filter.rs at main · datafuselabs/databend (github.com)

Databend

Algorithm for Building Verifiable Expression

WWW.DATABEND.COM

Functions	Origin Expression	Verifiable Expression
=	x = 1	max_x >= 1 and min_x <= 1
!=	x != 1	max_x != 1 or min_x != 1
isNull	isNull(x)	null_cnt_x > 0
isNotNull	isNotNull(x)	isnotnull(min_x)
<	x < 1	min_x < 1
<=	x <= 1	min_x <= 1
>	x > 1	max_x > 1
>=	x >= 1	max_x >= 1
like	x like ‘a%’ => x >= ‘a’ and x < ‘b’	max_x >= ‘a’ and min_x < ‘b’
not like	x not like ‘ab’	max_x != ‘ab’ or min_x != ‘ab’
	x not like ‘a%’	max_x >= ‘b’ or min_x < ‘a’
trival	f(x)	true

Comparisons between variable(only one column) and constant

Databend

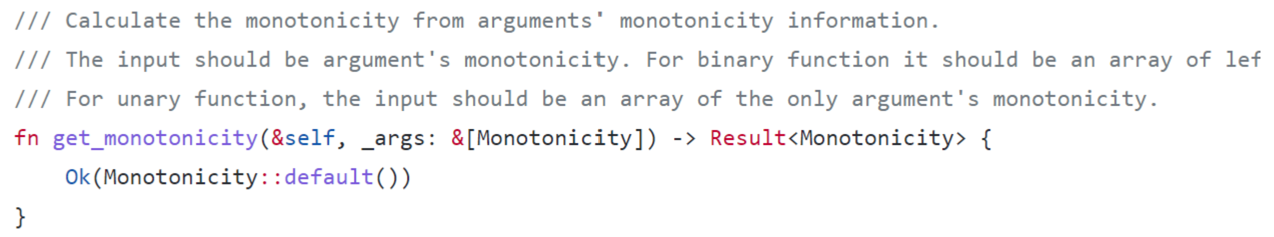
4 WIP & TODO

Partition Elimination in Databend

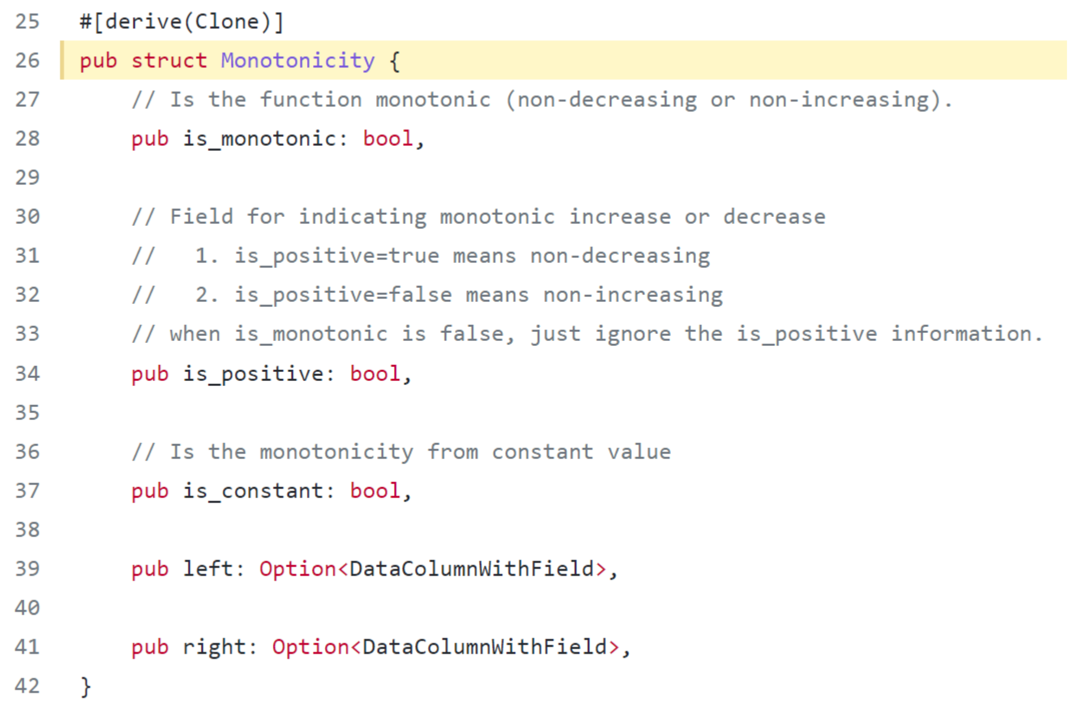
WIP & TODO

- Add Functions Monotonicity Check@junli1026. #2743 #2933
- Add maybe_monotonic feature for functions #3009

- Support More Function Monotonicity Check.
- Add Monotonicity Check in range filter.



```rust
/// Calculate the monotonicity from arguments' monotonicity information.
/// The input should be argument's monotonicity. For binary function it should be an array of lef
/// For unary function, the input should be an array of the only argument's monotonicity.
fn get_monotonicity(&self, _args: &[Monotonicity]) -> Result<Monotonicity> {
    Ok(Monotonicity::default())
}
```



```rust
#[derive(Clone)]
pub struct Monotonicity {
    // Is the function monotonic (non-decreasing or non-increasing).
    pub is_monotonic: bool,

    // Field for indicating monotonic increase or decrease
    //   1. is_positive=true means non-decreasing
    //   2. is_positive=false means non-increasing
    // when is_monotonic is false, just ignore the is_positive information.
    pub is_positive: bool,

    // Is the monotonicity from constant value
    pub is_constant: bool,

    pub left: Option<DataColumnWithField>,

    pub right: Option<DataColumnWithField>,
}
```

Databend

A Preliminary Plan

E.g. f(x) > 0

- Traverse f(x), get maybe_monotonic feature during build verifiable expression.
- The verifiable expression changes from f(max_x) > 0 to max_f(x) > 0.
- Call MonotonicityCheck in RangeFilter::eval.
- If f(x) in x[min, max] is monotonic, get a new [min, max]. Use the new [min, max] to evaluation.

Databend

感谢您的观看

THANK YOU FOR WATCHING