

-----1.1 Был выбран движок VersionedCollapsingMergeTree, так как в таблице есть Sign и Version, при вставке старой строки с sign=-1 и новой с sign=1, version=2, при запросе FINAL остаётся только актуальная версия

CREATE TABLE if not EXISTS tbl1

```
(  
    UserID UInt64,  
    PageViews UInt8,  
    Duration UInt8,  
    Sign Int8,  
    Version UInt8  
)
```

ENGINE = VersionedCollapsingMergeTree(Sign, Version)

ORDER BY UserID;

INSERT INTO tbl1 VALUES

(4324182021466249494, 5, 146, -1, 1);

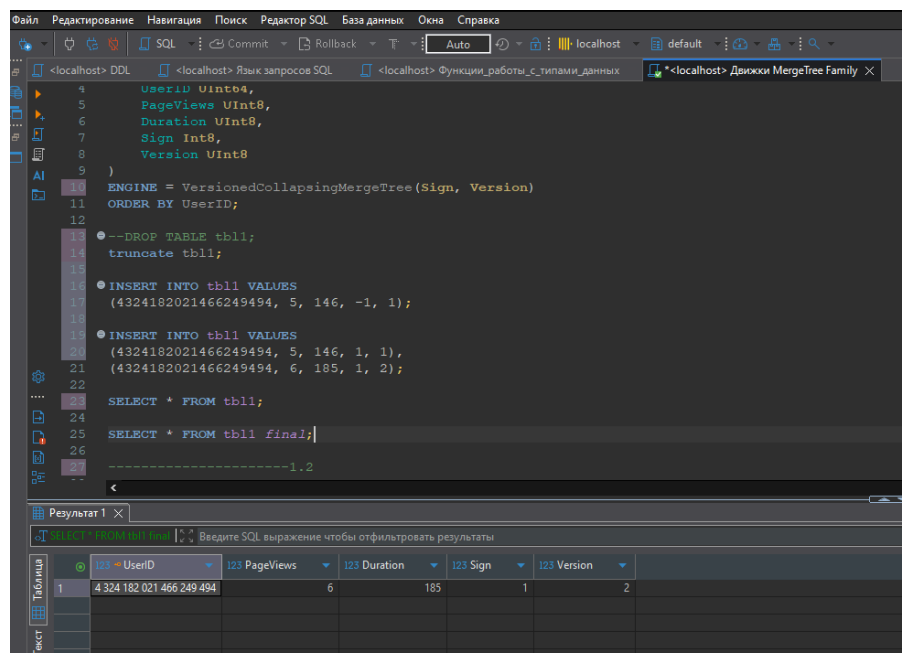
INSERT INTO tbl1 VALUES

(4324182021466249494, 5, 146, 1, 1),

(4324182021466249494, 6, 185, 1, 2);

SELECT * FROM tbl1;

SELECT * FROM tbl1 final;



-----1.2

CREATE TABLE tbl2

Есть пара key, value

Для одинаковых ключей ((1,1), (1,2)) движок должен агрегировать числовые поля

(

key **UInt32**,

value **UInt32**

)

ENGINE = SummingMergeTree

ORDER BY key;

INSERT INTO tbl2 **Values**(1,1),(1,2),(2,1);

select * from tbl2;

The screenshot shows the DBeaver 25.2.1 interface. The main editor displays the following SQL queries:

```
24  
25 SELECT * FROM tbl1 final;  
26  
27 -----1.2  
28  
29 ● CREATE TABLE if not EXISTS tbl2  
30 (  
31     key UInt32,  
32     value UInt32  
33 )  
34 ENGINE = SummingMergeTree  
35 ORDER BY key;  
36  
37 INSERT INTO tbl2 Values(1,1), (1,2), (2,1);  
38  
39 select * from tbl2;  
40  
41 -----1.3  
42 CREATE TABLE if not EXISTS tbl3  
43 (  
44     `id` Int32,  
45     `status` String,  
46     `price` String,  
47     `comment` String
```

The results pane at the bottom shows the output of the first query, "Результат 1". It displays a table with two columns: "key" and "value". The data is as follows:

key	value
1	3
2	1

-----1.3

Есть дубликаты по id, но без sign.

Для id=23 вставлены разные строки (разные price, comment)

CREATE TABLE tbl3

```
(  
    `id` Int32,  
    `status` String,  
    `price` String,  
    `comment` String  
)
```

ENGINE = ReplacingMergeTree()

PRIMARY KEY (id)

ORDER BY (id, status);

INSERT INTO tbl3 **VALUES** (23, 'success', '1000', 'Confirmed');

INSERT INTO tbl3 **VALUES** (23, 'success', '2000', 'Cancelled');

SELECT * **from** tbl3 **WHERE** id=23;

SELECT * **from** tbl3 *final* **WHERE** id=23;

The screenshot shows a SQL IDE with a dark theme. The editor contains the following SQL code:

```
41 -----1.3  
42 CREATE TABLE if not EXISTS tbl3  
43 (  
44     `id` Int32,  
45     `status` String,  
46     `price` String,  
47     `comment` String  
48 )  
49 ENGINE = ReplacingMergeTree()  
50 PRIMARY KEY (id)  
51 ORDER BY (id, status);  
52  
53 INSERT INTO tbl3 VALUES (23, 'success', '1000', 'Confirmed');  
54 INSERT INTO tbl3 VALUES (23, 'success', '2000', 'Cancelled');  
55  
56 SELECT * from tbl3 WHERE id=23;  
57  
58 SELECT * from tbl3 final WHERE id=23;  
59
```

Below the editor, the results of the last query are displayed. The title bar says "Результат 1". The SQL expression is "SELECT * from tbl3 final WHERE id=23". The results are shown in a table with 5 columns: id, status, price, comment, and an empty column. The first row shows the data for id=23.

	id	status	price	comment	
1	23	success	2000	Cancelled	

-----1.4-1.5

Есть партиционирование по месяцу (PARTITION BY toYYYYMM(StartDate)), сортировка по ключу (ORDER BY)

В структуре таблицы есть поле UserID AggregateFunction(uniq, UInt64).

Вставка выполняется через uniqState(UserID), потом данные агрегируются функцией uniqMerge().

Это уникальная особенность AggregatingMergeTree: хранить состояния агрегатных функций, а не сами данные.

Позволяет эффективно строить отчёты без пересчёта «с нуля»

```
CREATE TABLE tbl4
```

```
( CounterID UInt8,
```

```
  StartDate Date,
```

```
  UserID UInt64
```

```
) ENGINE = MergeTree
```

```
PARTITION BY toYYYYMM(StartDate)
```

```
ORDER BY (CounterID, StartDate);
```

```
INSERT INTO tbl4 VALUES(0, '2019-11-11', 1);
```

```
INSERT INTO tbl4 VALUES(1, '2019-11-12', 1);
```

```
CREATE TABLE tbl5
```

```
( CounterID UInt8,
```

```
  StartDate Date,
```

```
  UserID AggregateFunction(uniq, UInt64)
```

```
) ENGINE = AggregatingMergeTree
```

```
PARTITION BY toYYYYMM(StartDate)
```

```
ORDER BY (CounterID, StartDate);
```

```
INSERT INTO tbl5
```

```
select CounterID, StartDate, uniqState(UserID)
```

```
from tbl4
```

```
group by CounterID, StartDate;
```

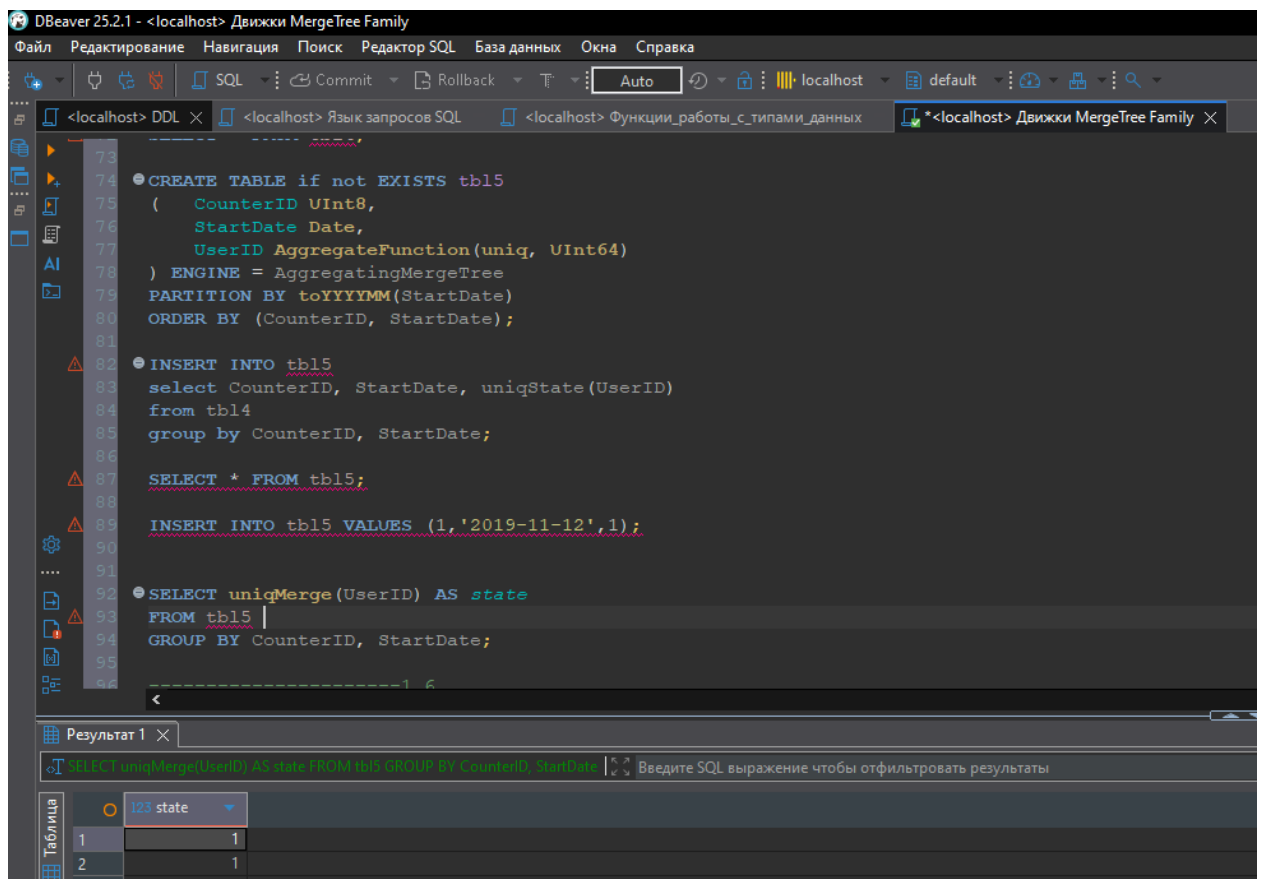
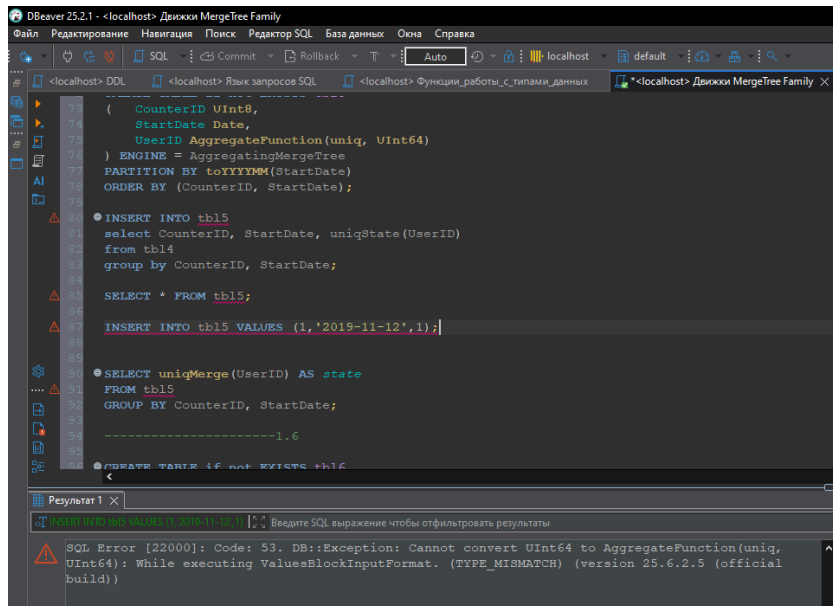
```
SELECT * FROM tbl5;
```

INSERT INTO tbl5 VALUES (1,'2019-11-12',1);

SELECT uniqMerge(UserID) AS state

FROM tbl5

GROUP BY CounterID, StartDate;



-----1.6

В таблице есть колонка sign.

Вставляются строки с sign=+1 и sign=-1.

При мердже они исчезнут, оставив только актуальные

CREATE TABLE tbl6

(

 `id` Int32,

 `status` String,

 `price` String,

 `comment` String,

 `sign` Int8

)

ENGINE = CollapsingMergeTree(**sign**)

PRIMARY KEY (id)

ORDER BY (id, status);

INSERT INTO tbl6 **VALUES** (23, 'success', '1000', 'Confirmed', 1);

INSERT INTO tbl6 **VALUES** (23, 'success', '1000', 'Confirmed', -1),

(23, 'success', '2000', 'Cancelled', 1);

SELECT * FROM tbl6;

SELECT * FROM tbl6 **FINAL**;

```
-----1.6
CREATE TABLE if not EXISTS tbl6
(
  `id` Int32,
  `status` String,
  `price` String,
  `comment` String,
  `sign` Int8
)
ENGINE = CollapsingMergeTree(sign)
PRIMARY KEY (id)
ORDER BY (id, status);

INSERT INTO tbl6 VALUES (23, 'success', '1000', 'Confirmed', 1);
INSERT INTO tbl6 VALUES (23, 'success', '1000', 'Confirmed', -1),
(23, 'success', '2000', 'Cancelled', 1);

SELECT * FROM tbl6;

SELECT * FROM tbl6 FINAL;
```

1 X

Введите SQL выражение чтобы отфильтровать результаты

id	status	price	comment	sign
23	success	2000	Cancelled	1