

Проекции и материализованные представления

-- Создаем исходную таблицу

CREATE TABLE sales

(

id **UInt32**,

product_id **UInt32**,

quantity **UInt32**,

price **Float32**,

sale_date **DateTime**

)

ENGINE = MergeTree

ORDER BY (product_id, sale_date);

-- Наполняем тестовыми данными

INSERT INTO sales

SELECT

number **AS** id,

(rand32() % 1000) + 1 **AS** product_id,

(rand32() % 10) + 1 **AS** quantity,

round((rand32() % 10000) / 100, 2) **AS** price,

now() - (rand32() % 86400) **AS** sale_date

FROM numbers(1_000_000);

-- Создаём проекцию

ALTER TABLE sales

ADD PROJECTION sales_projection

(

SELECT

product_id,

sum(quantity) **AS** total_quantity,

sum(quantity * price) **AS** total_sales

GROUP BY product_id

);

-- Материализуем проекцию для уже вставленных данных

ALTER TABLE sales **MATERIALIZE PROJECTION** sales_projection;

-- Проверим что проекция создалась

SELECT

table,

name,

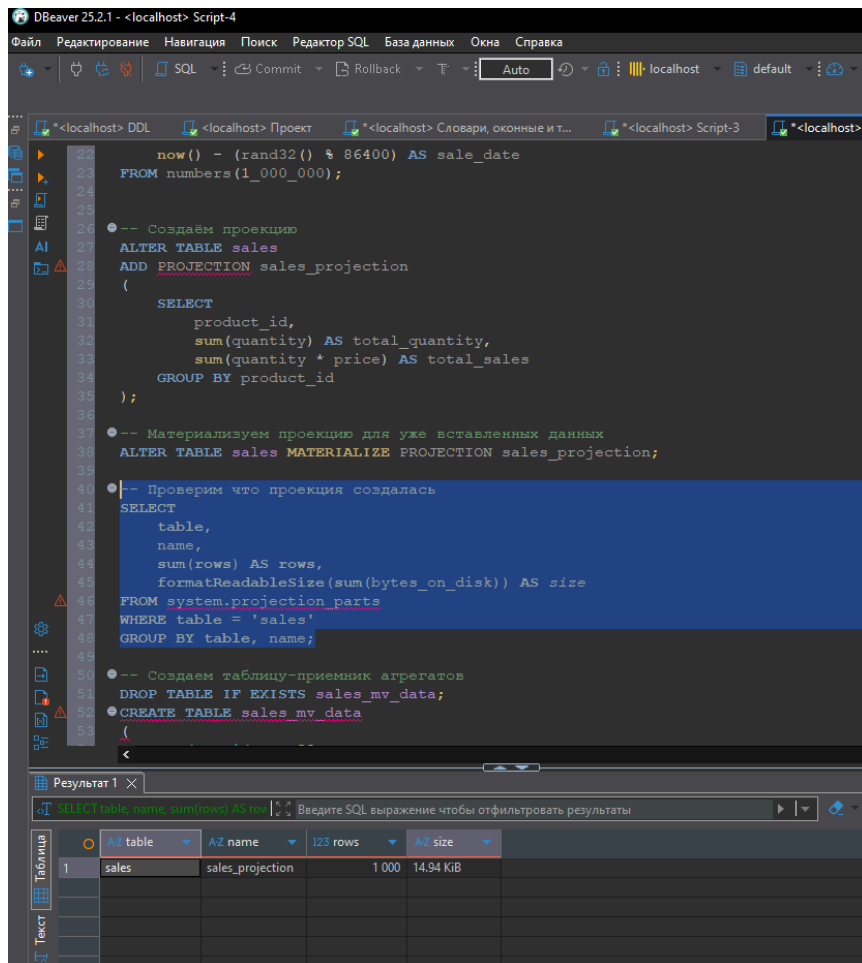
sum(rows) AS rows,

formatReadableSize(sum(bytes_on_disk)) AS size

FROM system.projection_parts

WHERE table = 'sales'

GROUP BY table, name;



The screenshot shows the DBeaver SQL editor with a script containing the following SQL commands:

```
22 now() - (rand32() % 86400) AS sale_date
23 FROM numbers(1_000_000);
24
25
26 -- Создаём проекцию
27 ALTER TABLE sales
28 ADD PROJECTION sales_projection
29 (
30     SELECT
31         product_id,
32         sum(quantity) AS total_quantity,
33         sum(quantity * price) AS total_sales
34     GROUP BY product_id
35 );
36
37 -- Материализуем проекцию для уже вставленных данных
38 ALTER TABLE sales MATERIALIZE PROJECTION sales_projection;
39
40 -- Проверим что проекция создалась
41 SELECT
42     table,
43     name,
44     sum(rows) AS rows,
45     formatReadableSize(sum(bytes_on_disk)) AS size
46 FROM system.projection_parts
47 WHERE table = 'sales'
48 GROUP BY table, name;
49
50 -- Создаем таблицу-приемник агрегатов
51 DROP TABLE IF EXISTS sales_mv_data;
52 CREATE TABLE sales_mv_data
53 (
```

The results of the query are displayed in a table with the following columns: table, name, rows, and size. The results show that the projection was successfully created and materialized.

table	name	rows	size
sales	sales_projection	1 000	14.94 KiB

-- Создаем таблицу-приемник агрегатов

DROP TABLE IF EXISTS sales_mv_data;

CREATE TABLE sales_mv_data

(

product_id **UInt32**,

total_quantity **UInt64**,

total_sales **Float64**

)

ENGINE = SummingMergeTree

ORDER BY product_id;

-- Создаём Materialized View, слушающее таблицу sales

DROP VIEW IF EXISTS sales_mv;

CREATE MATERIALIZED VIEW sales_mv

TO sales_mv_data

AS

SELECT

product_id,

sum(quantity) **AS** total_quantity,

sum(quantity * price) **AS** total_sales

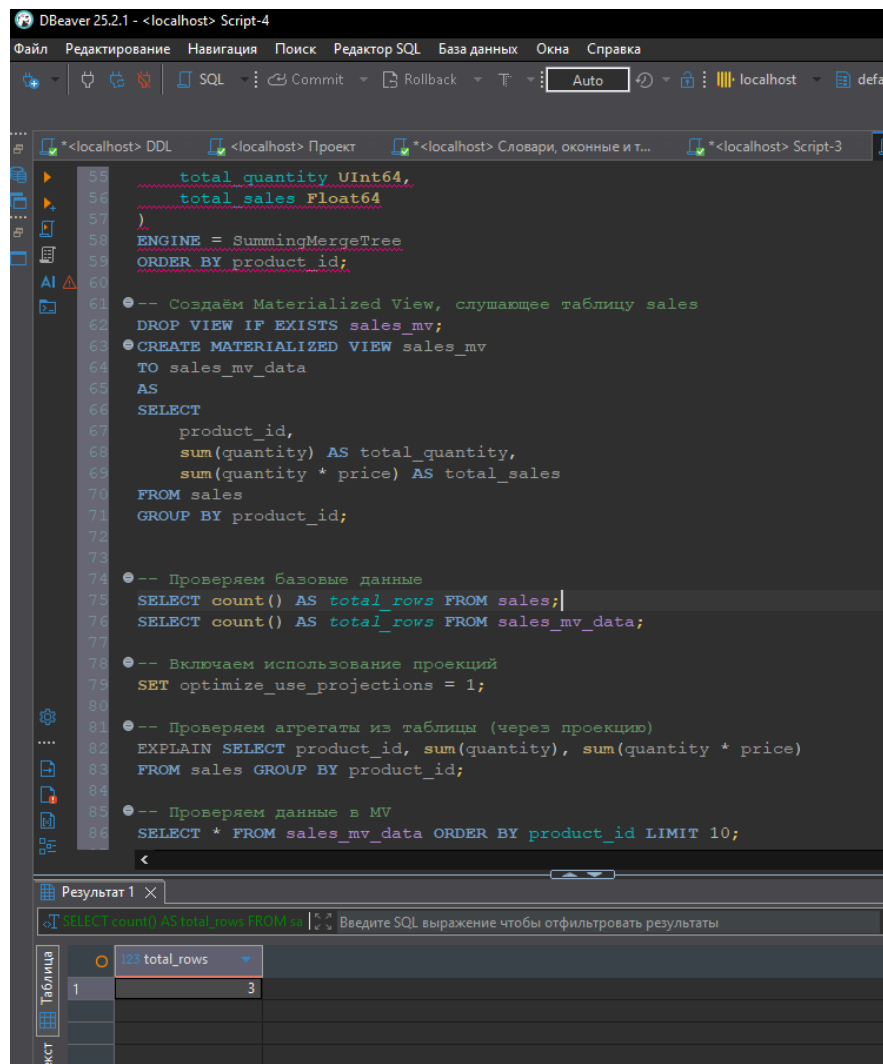
FROM sales

GROUP BY product_id;

-- Проверяем базовые данные

```
SELECT count() AS total_rows FROM sales;
```

```
SELECT count() AS total_rows FROM sales_mv_data;
```



-- Включаем использование проекций

```
SET optimize_use_projections = 1;
```

-- Проверяем агрегаты из таблицы (через проекцию)

```
EXPLAIN SELECT product_id, sum(quantity), sum(quantity * price)
```

```
FROM sales GROUP BY product_id;
```

```

81 -- Провераем агрегаты из таблицы (через проекцию)
82 EXPLAIN SELECT product_id, sum(quantity), sum(quantity * price)
83 FROM sales GROUP BY product_id;
84
85 -- Провераем данные в MV
86 SELECT * FROM sales_mv_data ORDER BY product_id LIMIT 10;
87
88
89 -- Добавляем новую продажу
90 INSERT INTO sales VALUES (99999999, 10, 5, 99.9, now());
91
92 -- Провераем результат
93 SELECT * FROM sales_mv_data WHERE product_id = 10; -- MV обновилось
94 SELECT sum(quantity), sum(quantity * price)
95 FROM sales WHERE product_id = 10; -- Основная таблица

```

Результат 1 X

EXPLAIN SELECT product_id, sum(quantity), sum(quantity * price) FROM sales GROUP BY product_id; Введите SQL выражение чтобы отфильтровать результаты

	Expression ((Project names + Projection))
1	Expression ((Project names + Projection))
2	Aggregating
3	Expression
4	ReadFromMergeTree (sales_projection)

-- Провераем данные в MV

```
SELECT * FROM sales_mv_data ORDER BY product_id LIMIT 10;
```

-- Добавляем новую продажу

```
INSERT INTO sales VALUES (99999999, 10, 5, 99.9, now());
```

-- Провераем результат

```
SELECT * FROM sales_mv_data WHERE product_id = 10; -- MV обновилось
```

```

92 -- Провераем результат
93 SELECT * FROM sales_mv_data WHERE product_id = 10; -- MV обновилось
94
95 SELECT sum(quantity), sum(quantity * price)
96 FROM sales WHERE product_id = 10; -- Основная таблица
97 SET optimize_use_projections = 1;
98 SELECT sum(quantity), sum(quantity * price)
99 FROM sales WHERE product_id = 10; -- Проекция
100
101 -- Ожидание: и MV, и проекция показывают обновлённые данные
102
103 -- Удаляем продажи для product_id = 10
104 ALTER TABLE sales DELETE WHERE product_id = 10;

```

Результат 1 X

SELECT * FROM sales_mv_data WHERE product_id = 10; Введите SQL выражение чтобы отфильтровать результаты

	product_id	total_quantity	total_sales
1	10	10	500
2	10	5	499,5000076294

```
SELECT sum(quantity), sum(quantity * price)  
FROM sales WHERE product_id = 10;           -- Основная таблица
```

```
SET optimize_use_projections = 1;
```

```
SELECT sum(quantity), sum(quantity * price)  
FROM sales WHERE product_id = 10;           -- Проекция
```

-- Ожидание: и MV, и проекция показывают обновлённые данные

-- Удаляем продажи для product_id = 10

```
ALTER TABLE sales DELETE WHERE product_id = 10;
```

-- Проверяем, что в sales данные удалены

```
SELECT count() FROM sales WHERE product_id = 10;
```

-- Проверяем MV и проекцию

```
SELECT * FROM sales_mv_data WHERE product_id = 10;   -- MV не изменится!
```

```
SET optimize_use_projections = 1;
```

```
SELECT sum(quantity), sum(quantity * price)  
FROM sales WHERE product_id = 10;           -- Проекция тоже ещё содержит старые данные
```

-- Пересоздаём проекцию, чтобы обновить агрегаты

```
ALTER TABLE sales MATERIALIZED VIEW sales_projection;
```

-- Проверяем снова

```
SET optimize_use_projections = 1;
```

```
SELECT sum(quantity), sum(quantity * price)  
FROM sales WHERE product_id = 10;
```

-- Проверяем метаданные проекций

```
SELECT  
  
  table,  
  
  name,  
  
  rows,  
  
  modification_time  
  
FROM system.projection_parts  
  
WHERE table = 'sales'  
  
ORDER BY modification_time DESC;
```

-- Проверяем метаданные MV

```
SELECT  
  
  table,  
  
  sum(rows) AS rows,  
  
  max(modification_time) AS last_update  
  
FROM system.parts  
  
WHERE table = 'sales_mv_data'  
  
GROUP BY table;
```