[Some of] New Query Optimizer features in MariaDB 10.3

Sergei Petrunia <sergey@mariadb.com>

MariaDB Shenzhen Meetup

November 2017

Plan

- MariaDB 10.2: Condition pushdown
- MariaDB 10.3: Condition pushdown through window functions
- MariaDB 10.3: GROUP BY splitting

Condition pushdown

- Just condition pushdown in 10.2
- Pushdown through window functions in 10.3

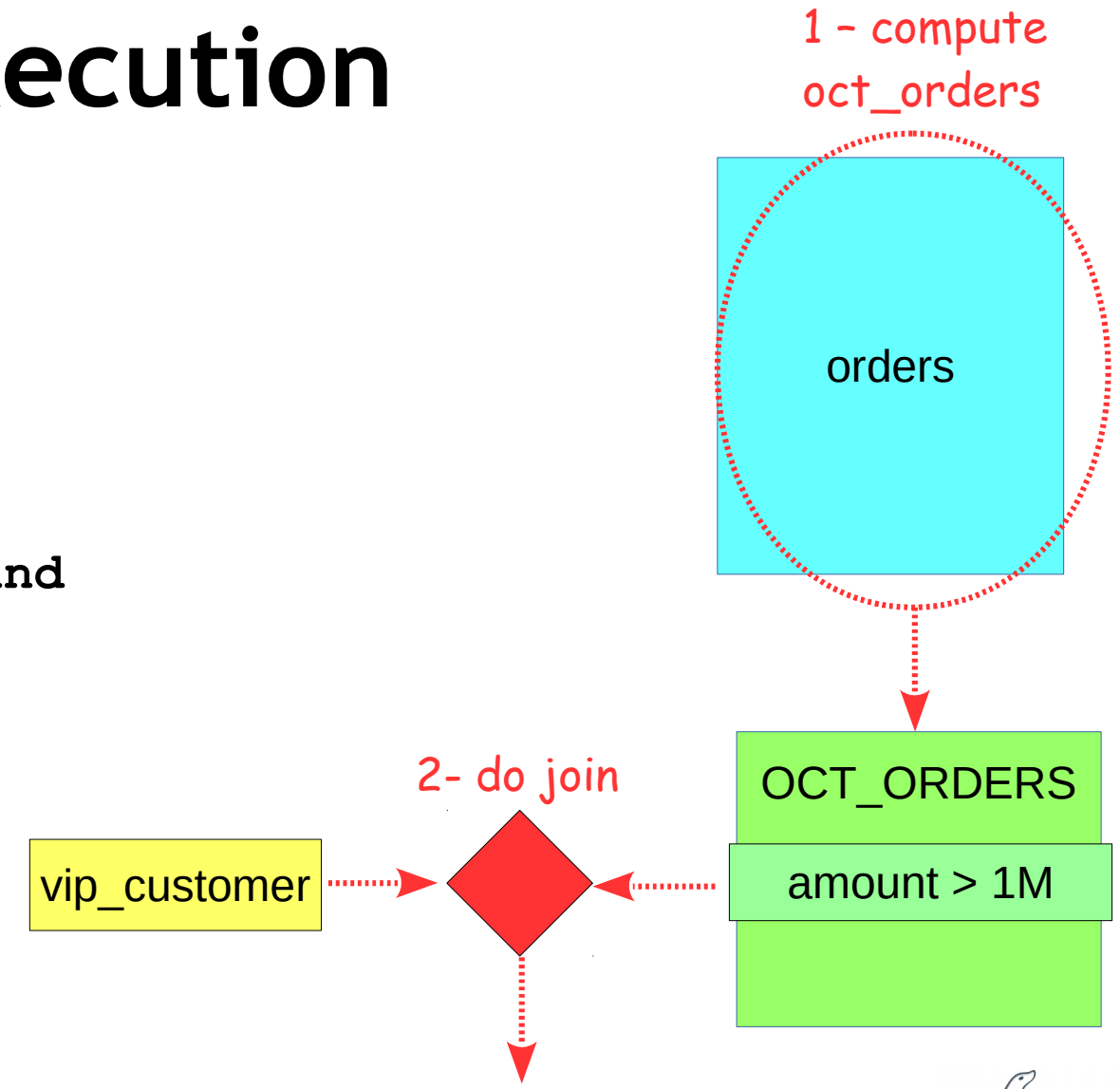
Background - derived table merge

- “VIP customers and their big orders from October”

```
select *
from
  vip_customer,
   (select *
    from orders
    where order_date BETWEEN '2017-10-01' and '2017-10-31'
    ) as OCT_ORDERS
where
  OCT_ORDERS.amount > 1M and
  OCT_ORDERS.customer_id = customer.customer_id
```

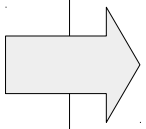
Naive execution

```
select *
from
  vip_customer,
   (select *
    from orders
    where
      order_date BETWEEN '2017-10-01' and
                         '2017-10-31'
    ) as OCT_ORDERS
where
  OCT_ORDERS.amount > 1M and
  OCT_ORDERS.customer_id =
  vip_customer.customer_id
```



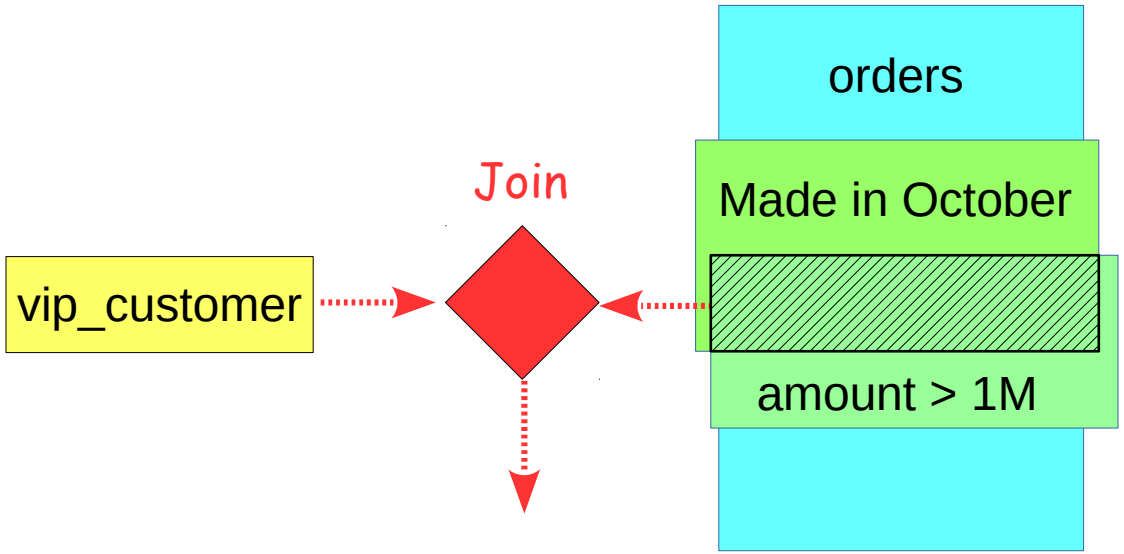
Derived table merge

```
select *
from
  vip_customer,
   (select *
    from orders
    where
      order_date BETWEEN '2017-10-01' and
                         '2017-10-31'
   ) as OCT_ORDERS
where
  OCT_ORDERS.amount > 1M and
  OCT_ORDERS.customer_id =
  vip_customer.customer_id
```



```
select *
from
   vip_customer,
   orders
where
  order_date BETWEEN '2017-10-01' and
                     '2017-10-31'
  and
  orders.amount > 1M and
  orders.customer_id =
  vip_customer.customer_id
```

Execution after merge




```
select *
from
   vip_customer,
   orders
where
   order_date BETWEEN '2017-10-01' and
                      '2017-10-31'
   and
   orders.amount > 1M and
   orders.customer_id =
   vip_customer.customer_id
```

- Allows the optimizer to do customer->orders or orders → customer
- Good for optimization

MariaDB

Another use case - grouping

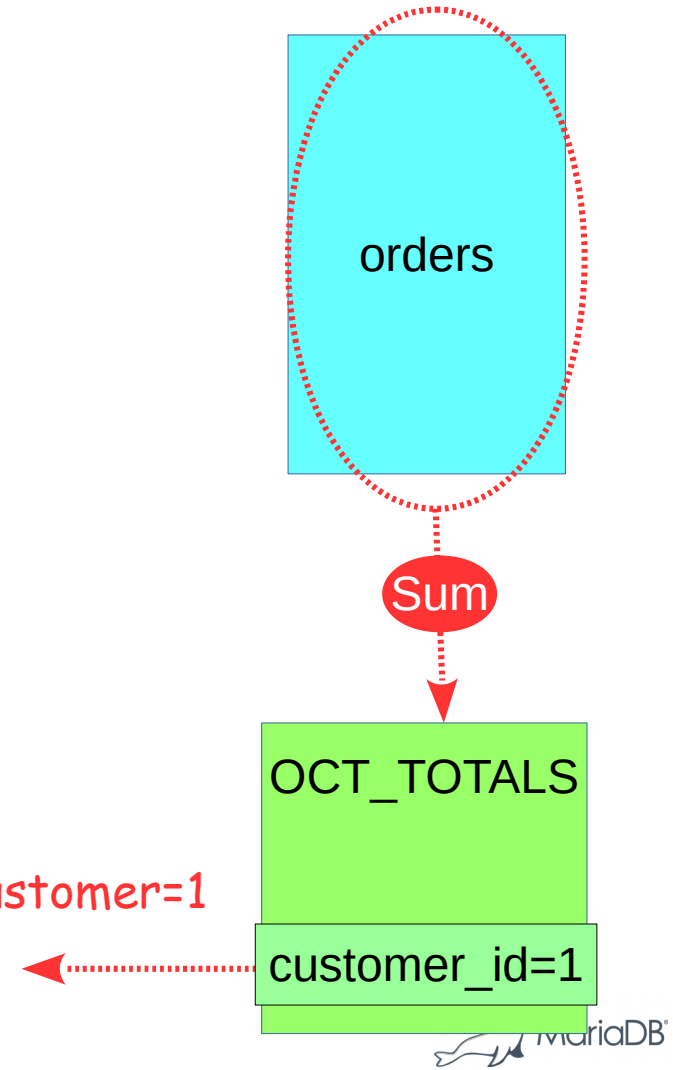
```
create view OCT_TOTALS as
select
   customer_id,
   SUM(amount) as TOTAL_AMT
from orders
where
   order_date BETWEEN '2017-10-01' and '2017-10-31'
group by
   customer_id


select * from OCT_TOTALS where customer_id=1
```

- Can’t merge due to GROUP BY in the child.

Execution is inefficient

```
create view OCT_TOTALS as
select
    customer_id,
    SUM(amount) as TOTAL_AMT
from orders
where
    order_date BETWEEN '2017-10-01' and '2017-10-31'
group by
    customer_id
```

```
select * from OCT_TOTALS where customer_id=1
```



Condition pushdown

```
create view OCT_TOTALS as
select
   customer_id,
   SUM(amount) as TOTAL_AMT
from orders
where
  order_date BETWEEN '2017-10-01' and '2017-10-31'
group by
  customer_id
```

```
select *
from OCT_TOTALS
where customer_id=1
```

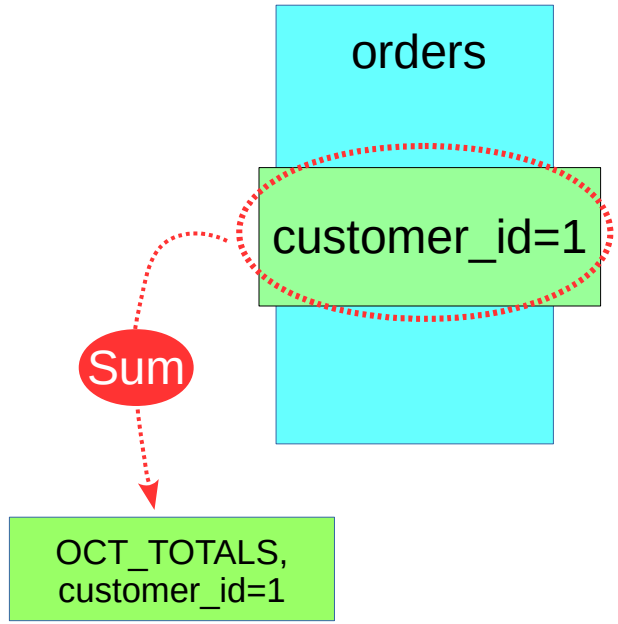
- Can push down conditions on GROUP BY columns
- … to filter out rows that go into groups we dont care about

MariaDB

Condition pushdown

```
create view OCT_TOTALS as
select
    customer_id,
    SUM(amount) as TOTAL_AMT
from orders
where
  order_date BETWEEN '2017-10-01' and '2017-10-31'
group by
  customer_id

select *
from OCT_TOTALS
where customer_id=1
```

1 - find customer_id=1



- Looking only at rows you're interested in is much more efficient

MariaDB 10.3: Pushdown through Window Functions

- "Customer's biggest orders"

```
create view top_three_orders as
select *
from
(
  select
    customer_id,
    amount,
    rank() over (partition by customer_id
                 order by amount desc
                 ) as order_rank
  from orders
) as ordered_orders
where order_rank<3
```

```
+-------------+--------+------------+
| customer_id | amount | order_rank |
+-------------+--------+------------+
|           1 |  10000 |          1 |
|           1 |   9500 |          2 |
|           1 |    400 |          3 |
|           2 |   3200 |          1 |
|           2 |   1000 |          2 |
|           2 |    400 |          3 |
...
```

```
select * from top_three_orders where customer_id=1
```

MariaDB 10.3: Pushdown through Window Functions

```
select * from top_three_orders where customer_id=1
```

MariaDB 10.2, MySQL 8.0

- Compute `top_three_orders` for all customers
- select rows with `customer_id=1`

MariaDB 10.3 (and e.g. PostgreSQL)

- Only compute `top_three_orders` for `customer_id=1`
  - This can be much faster!
  - Can make use of index(customer_id)

“Split grouping for derived”

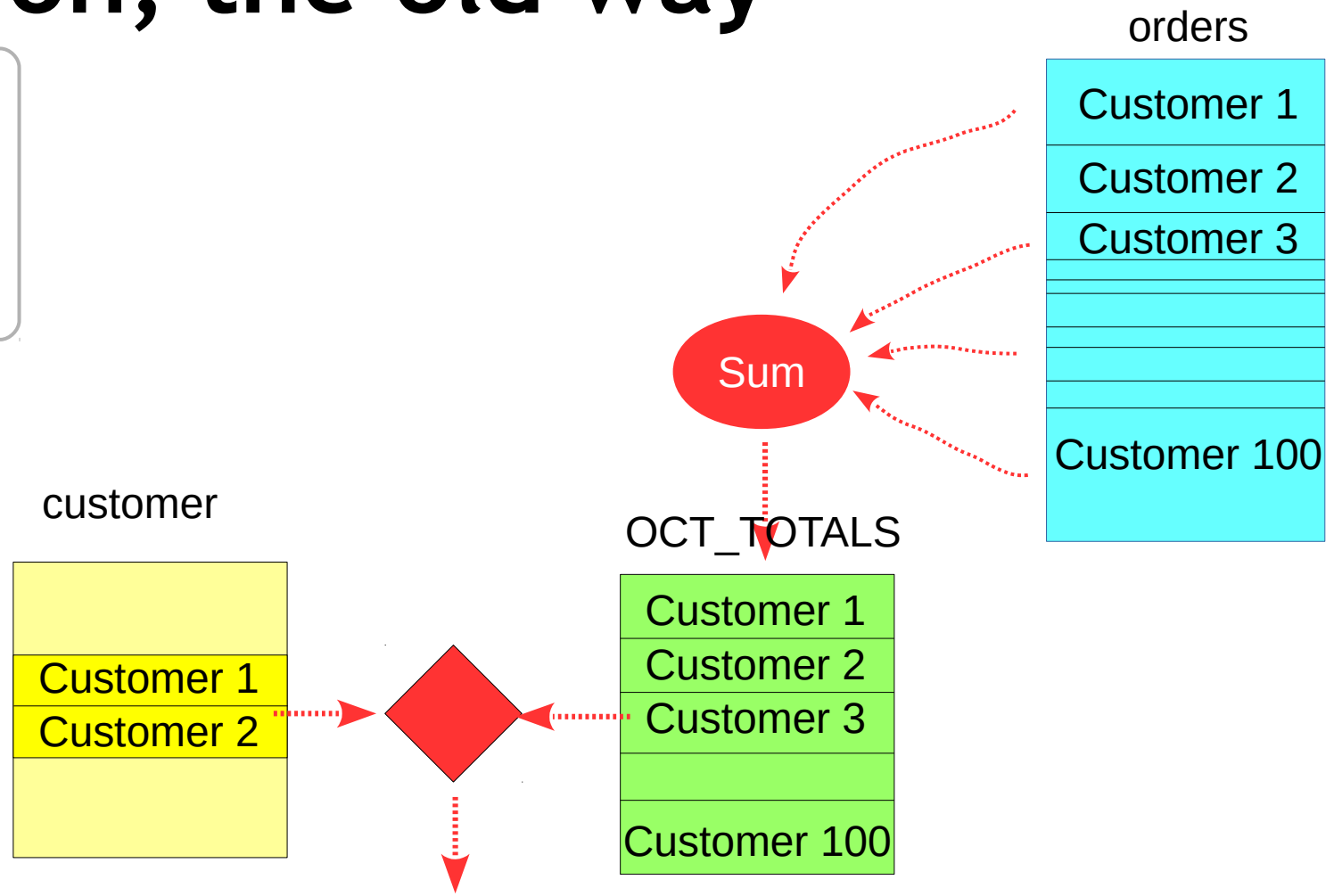
```
create view OCT_TOTALS as
select
   customer_id,
   SUM(amount) as TOTAL_AMT
from orders
where
  order_date BETWEEN '2017-10-01' and '2017-10-31'
group by
  customer_id
```

```
select *
from
  customer, OCT_TOTALS
where
  customer.customer_id=OCT_TOTALS.customer_id and
  customer.customer_name IN ('Customer 1', 'Customer 2')
```

Execution, the old way

```
create view OCT_TOTALS as
select
   customer_id,
   SUM(amount) as TOTAL_AMT
from orders
where
  order_date BETWEEN '2017-10-01' and '2017-10-31'
group by
  customer_id
```

```
select *
from
  customer, OCT_TOTALS
where
  customer.customer_id=
  OCT_TOTALS.customer_id and
  customer.customer_name IN ('Customer 1',
                             'Customer 2')
```



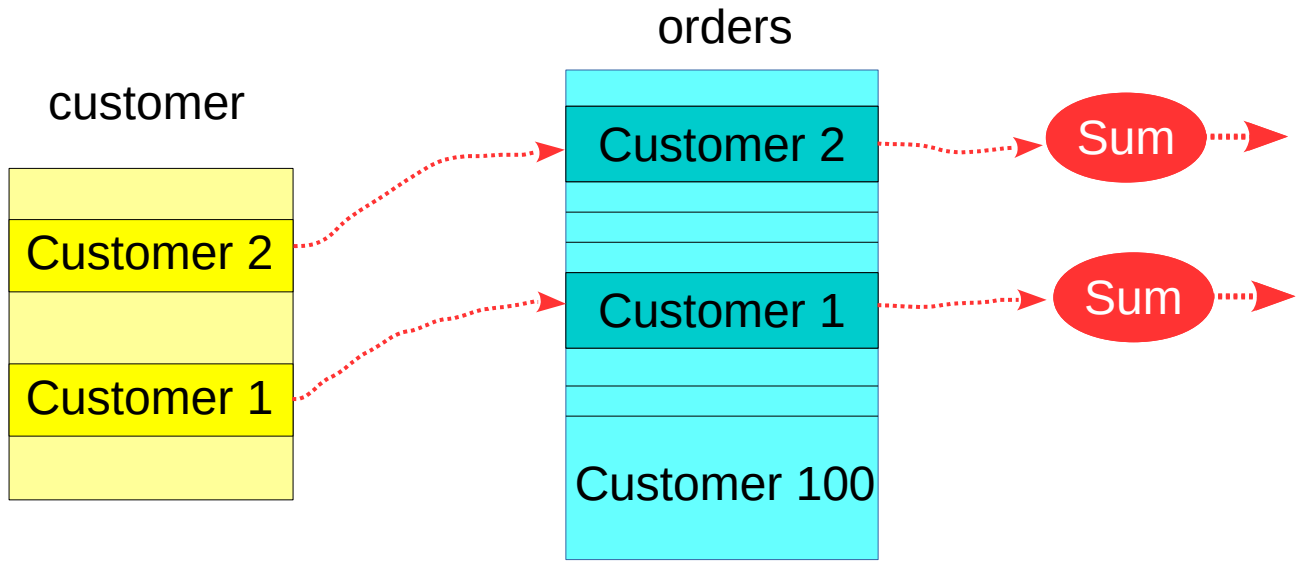
- Inefficient, OCT_TOTALS is computed for *all* customers.

Split grouping execution

- Can be used when doing join from `customer` to `orders`
- Must have equalities for GROUP BY columns:

```
OCT_TOTALS.customer_id=customer.customer_id
```

  - This allows to select one group
- The underlying table (orders) must have an index on the GROUP BY column (`customer_id`)
  - This allows to use ref access



Split grouping execution

```
create view OCT_TOTALS as
select
   customer_id,
   SUM(amount) as TOTAL_AMT
from orders
where
  order_date BETWEEN '2017-10-01' and '2017-10-31'
group by
  customer_id
```

```
select *
from
  customer, OCT_TOTALS
where
  customer.customer_id=
  OCT_TOTALS.customer_id and
  customer.customer_name IN ('Customer 1',
                             'Customer 2')
```

```
+------+-----------------+------------+------+---------------+-------------+---------+----------------------+------+-------------+
| id   | select_type     | table      | type | possible_keys | key         | key_len | ref                  | rows | Extra       |
+------+-----------------+------------+------+---------------+-------------+---------+----------------------+------+-------------+
|    1 | PRIMARY         | customer   | ALL  | PRIMARY       | NULL        | NULL    | NULL                 | 1000 |             |
|    1 | PRIMARY         | <derived2> | ref  | key0          | key0        | 4       | customer.customer_id |   36 |             |
|    2 | LATERAL DERIVED | orders     | ref  | customer_id   | customer_id | 4       | customer.customer_id |  365 | Using where |
+------+-----------------+------------+------+---------------+-------------+---------+----------------------+------+-------------+
```

- EXPLAIN shows “LATERAL DERIVED”
- @@optimizer_switch flag: split_grouping_derived (ON by default)
- Not fully cost-based choice atm (check query plan, use if possible and certainly advantageous)

MariaDB

Summary

- MariaDB 10.2: **Condition pushdown for derived tables** optimization
  - Push a condition into derived table
  - Used when derived table cannot be merged
  - Biggest effect is for subqueries with GROUP BY
- MariaDB 10.3: **Condition Pushdown through Window functions**
- MariaDB 10.3: **Lateral derived** optimization
  - When doing a join, can’t do condition pushdown
  - So, lateral derived is used. It allows to only examine GROUP BY groups that match other tables. It needs index on grouped columns
  - Work in progress (optimization process is very basic ATM)

Thanks!

Discussion