**Sales Precomputation**

**1.** Precomputation Table:

create table sales.pre\_cat\_sale as

SELECT category\_id, sum(price) dollar\_value

FROM sales.category

NATURAL INNER JOIN sales.product

NATURAL INNER JOIN sales.sale

GROUP BY category\_id;

create table sales.pre\_cus\_sale as

SELECT customer\_id, sum(price) dollar\_value

FROM sales.sale

GROUP BY customer\_id;

2. Trigger

CREATE FUNCTION sales.Pre\_cat\_sale\_func() RETURNS trigger $Pre\_cat\_sale\_func$

BEGIN

UPDATE sales. sales.pre\_cat\_sale pre

SET dollar\_value = pre.dollar\_value + NEW.price

FROM sales.product p

WHERE pre.category\_id = p.category\_Id

AND p.product\_id = NEW.product\_id;

IF NOT FOUND THEN

INSERT INTO sales.pre\_cat\_sale Sale (category\_id, dollar\_value)

SELECT p.category\_id, NEW.price

FROM sales.product p

WHERE p.product\_id = NEW.product\_id;

END IF;

RETURN NEW;

END;

$Pre\_cat\_sale\_func$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_Sale\_Category

BEFORE INSERT ON sales.sale

FOR EACH ROW EXECUTE PROCEDURE sales.Pre\_cat\_sale\_func();

CREATE sales.Pre\_cus\_sale\_func() RETURNS trigger $Pre\_cus\_sale\_func$

BEGIN

UPDATE sales.pre\_cat\_sale

SET dollar\_value = dollar\_value + NEW.price

WHERE customer\_id = NEW.customer\_id;

IF NOT FOUND THEN

INSERT INTO sales.pre\_cat\_sale (customer\_id, dollar\_value)

SELECT NEW.customer\_id, NEW.price;

END IF;

RETURN NEW;

END;

$Pre\_cus\_sale\_func$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS trigger\_Sale\_Customer ON sales.sale;

CREATE TRIGGER trigger\_Sale\_Customer

BEFORE INSERT ON sales.sale

FOR EACH ROW EXECUTE PROCEDURE sales.Pre\_cus\_sale\_func();

**4. Original Query Cost:**

SELECT cate.category\_id

,cust.customer\_id

,sum(quantity)

,sum(price)

FROM

(

SELECT category\_id

,sum(price) AS dollar\_value

FROM sales.category NATURAL

INNER JOIN sales.product NATURAL

INNER JOIN sales.sale

GROUP BY category\_id

ORDER BY dollar\_value DESC limit 10

)

AS cate,

(

SELECT customer\_id, sum(price) dollar\_value

FROM sales.sale

GROUP BY customer\_id

ORDER BY dollar\_value DESC limit 10

)

AS cust, sales.sale s, sales.product p

WHERE p.category\_id = cate.category\_id AND s.customer\_id = cust.customer\_id AND s.product\_id = p.product\_id

GROUP BY cate.category\_id, cust.customer\_id

ORDER BY cate.category\_id

**Cost (cost=170725.08..170725.28 rows=80 width=18)**

**5. New Query Cost:**

WITH cate as

(

SELECT category\_id

FROM sales.PRE\_Category\_Sale

ORDER BY dollar\_value DESC limit 10

) ,

cust as

(

SELECT customer\_id

FROM sales.PRE\_Customer\_Sale

ORDER BY dollar\_value DESC limit 10

)

SELECT cate.category\_id, cust.customer\_id, sum(quantity), sum(price)

FROM cate, cust, sales.sale s, sales.product p

WHERE p.category\_id = cate.category\_id AND s.customer\_id = cust.customer\_id AND s.product\_id = p.product\_id

GROUP BY cate.category\_id, cust.customer\_id

ORDER BY cate.category\_id

**Query Cost: 26452.2**

### **6. Indices**

CREATE INDEX pre\_cat\_sale\_dol\_cat

ON sales.pre\_cat\_sale

USING btree

(dollar\_value DESC, category\_id);

CREATE INDEX product\_cid\_pid

ON sales.product

USING btree

(category\_id, product\_id);

### 

### CREATE INDEX sale\_pid\_num\_price

ON sales.sale

USING btree

(product\_id, quantity, price);