



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря
Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та
спеціалізованих комп'ютерних систем**

Лабораторна робота №3

з дисципліни
«Бази даних і засоби управління»

Тема: «Засоби оптимізації роботи СУБД
PostgreSQL»

Виконав: студент III курсу

ФПМ групи КВ-84

Бугайов Д.С.

Перевірів:

Київ – 2020

Загальне завдання роботи полягає в такому:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

Варіант 6:

6	<i>BTree, BRIN</i>	<i>after update, insert</i>
---	--------------------	-----------------------------

Використана мова: C++

ORM бібліотека: ODB

Вимоги до пункту завдання №1

Для перетворення функцій, що реалізують запити до об’єктної бази даних, необхідно встановити бібліотеку sqlalchemy, налаштувати програму на роботу з ORM, розробити класи-сутності для об’єктів-сутностей, представлених відповідними таблицями БД та пов’язаних зв’язками 1:М, М:М та 1:1 виконати опис схеми бази даних. Особливу увагу приділити контролю зовнішніх зв’язків між таблицями засобами ORM.

Замінити виклики запитів мовою SQL на відповідні запити засобами SQLAlchemy по роботі з об’єктами. Обов’язковим є реалізація вставки, вилучення та редагування екземплярів класів-сутностей. Розробка запитів на генерацію даних та пошук екземплярів класів-сутностей вітається, але не є обов’язковою. Інтерфейси функцій (вхідні та вихідні аргументи функцій модуля “Модель”) мають залишитись без змін.

Вимоги до пункту завдання №2

Відповідно до варіанту індексування продемонструвати на прикладах запитів SQL SELECT підвищення швидкодії їх виконання з використанням індексів, а також пояснити чому для деяких випадків індексування використовувати недоцільно. При цьому для наочного представлення слід використати

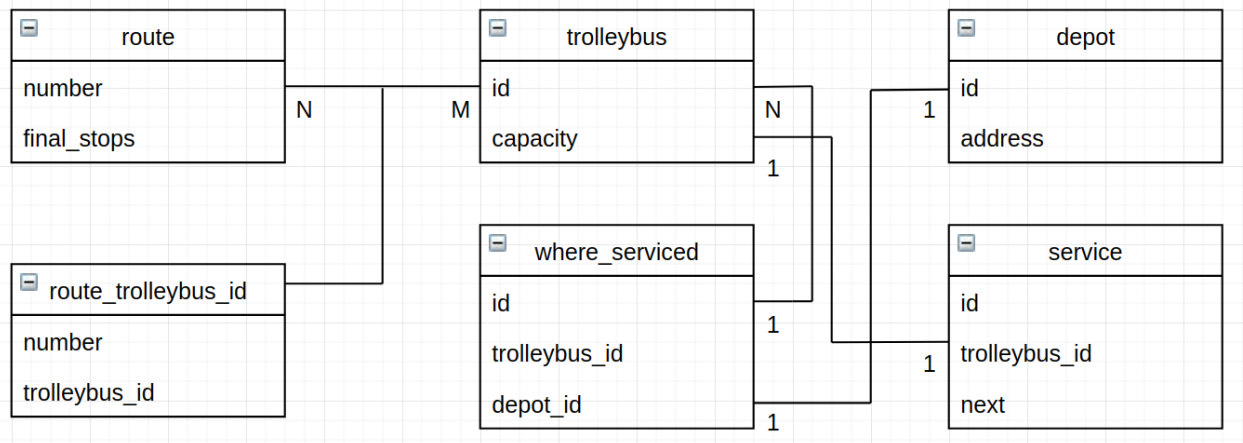
функцію генерування рандомізованих даних з лабораторної роботи №2, створивши необхідну кількість тестових даних. Навести 4-5 прикладів запитів SELECT (із виведенням результуючих даних), що містять фільтрацію, агрегатні функції, групування та сортування (у необхідних комбінаціях).

Вимоги до пункту завдання №3

Створити тригер бази даних PostgreSQL відповідно до варіанта. Тригерна функція має включати обробку запису, що модифікується (вставляється або вилучається), умовні оператори, курсорні цикли та обробку виключних ситуацій. Виконати відлагодження тригера при різних вхідних даних, навівши 2-3 приклади його використання.

Звіт до пункту завдання №1

Схема бази даних у вигляді таблиць і зв'язки між ними



Класи ORM

```
class trolleybus;
class depot;
class route;
class where_served;
class service;

#pragma db object session
class trolleybus {...};

#pragma db object session
class depot {...};

#pragma db object session pointer(std::shared_ptr)
class route {...};

#pragma db object session pointer(std::shared_ptr)
class where_served {...};

#pragma db object session pointer(std::shared_ptr)
class service {...};
```

*назви класів, дані класів і зв'язки між ними збігаються зі схемою бази даних, окрім таблиці “route_trolleybus_id”. Вона оновлюється автоматично під час маніпуляцій з таблицею “route” завдяки зв'язку M:N, тобто “route_trolleybus_id” не потребує окремого ORM класу. В цьому можна переконатись у файлі orm_classes.h

Типовий запит на запис рядка у вигляді ORM

```
if(tab_name == "trolleybus"){

    std::cout << "\nInput trolleybus id >> ";
    int tr_id = safe_uint_input();
    std::cout << "\nInput trolleybus capacity >> ";
    int tr_capacity = safe_uint_input();

    ctx.insert_row(std::make_shared<trolleybus>(tr_id, tr_capacity));
}
```

```
<OdbT>
void insert_row( const std::shared_ptr< OdbT >& obj )
{
    tr_.database().persist( *obj );
}
```

Типовий запит на видалення рядка у вигляді ORM

```
if(tab_name == "trolleybus"){
    std::cout << "\nInput trolleybus id >> ";
    unsigned long tr_id = safe_uint_input();

    ctx.delete_row<trolleybus>(tr_id);
}
```

```
template< typename OdbT >
void delete_row( const unsigned long& id)
{
    tr_.database().erase<OdbT>( id );
}
```

Типовий запит на редагування рядка у вигляді ORM

```
if(tab_name == "trolleybus"){

    std::cout << "\nInput trolleybus id to update>> ";
    int tr_id = safe_uint_input();
    std::cout << "\nInput new trolleybus capacity >> ";
    int tr_capacity = safe_uint_input();

    ctx.update_row(std::make_shared<trolleybus>(tr_id, tr_capacity));
}
```

```
template< typename OdbT >
void update_row( const std::shared_ptr< OdbT >& obj )
{
    tr_.database().update( *obj );
}
```

Звіт до пункту завдання №2

Команда створення індекса

Btree

postgres/postgres@Trolleybus

Query Editor Query History

1

2

3

4

CREATE INDEX index_sometext ON testindex USING btree (
sometext
);

Data Output Explain Messages Notifications

CREATE INDEX

Query returned successfully in 1 min 52 secs.

BRIN

postgres/postgres@Trolleybus

Query Editor Query History

1

2

3

4

CREATE INDEX index_sometext_brin ON testindex USING brin (
sometext
);

Data Output Explain Messages Notifications

CREATE INDEX

Query returned successfully in 8 secs 12 msec.

Приклад фільтрації даних

Без індексації

postgres/postgres@Trolleybus

Query Editor Query History

1

explain analyze select * from testindex where sometext = '생약鋒'

Data Output Explain Messages Notifications

QUERY PLAN

text

1

2

3

4

5

6

7

8

Gather (cost=1000.00..11614.43 rows=1 width=13) (actual time=39.513..82.947 rows=1 loops=1)

Workers Planned: 2

Workers Launched: 2

-> Parallel Seq Scan on testindex (cost=0.00..10614.33 rows=1 width=13) (actual time=60.802..72.838 rows=0 loops=3)

Filter: ((sometext)::text = '생약鋒':text)

Rows Removed by Filter: 333333

Planning Time: 0.068 ms

Execution Time: 82.987 ms

Btree

postgres/postgres@Trolleybus

Query Editor Query History

1 explain analyze select * from testindex where sometext = '생苟鋒'

Data Output Explain Messages Notifications

QUERY PLAN
text

1	Index Scan using index_sometext on testindex (cost=0.42..8.44 rows=1 width=13) (actual time=0.047..0.048 rows=1 loops=1)
2	Index Cond: ((sometext)::text = '생苟鋒'::text)
3	Planning Time: 0.071 ms
4	Execution Time: 0.070 ms

BRIN

Query Editor Query History

1 explain analyze select * from testindex where sometext = '생苟鋒'

Data Output Explain Messages Notifications

QUERY PLAN
text

1	Gather (cost=1000.00..11614.43 rows=1 width=13) (actual time=40.763..81.822 rows=1 loops=1)
2	Workers Planned: 2
3	Workers Launched: 2
4	-> Parallel Seq Scan on testindex (cost=0.00..10614.33 rows=1 width=13) (actual time=58.858..70.617 rows=0 loops=3)
5	Filter: ((sometext)::text = '생苟鋒'::text)
6	Rows Removed by Filter: 333333
7	Planning Time: 0.109 ms
8	Execution Time: 81.850 ms

Вихідні дані

Query Editor Query History

1 select * from testindex where sometext = '생苟鋒'

Data Output Explain Messages Notifications

	Id	sometext
	integer	character varying (64)
1	500000	생苟鋒

Висновок

Індексація Btree значно прискорила пошук, індексація BRIN не вплинула на швидкість пошуку у порівнянні з експериментом без індексації. Неефективність BRIN пов'язана з тим що у даному випадку проіндексовані дані(текстові рядки) не мають ознаки групування, тобто групою для пошуку є уся множина рядків. Ефективність індексації Btree може бути зумовлена відсортованістю даних за алфавітом.

Приклад з агрегатними функціями

Без індексації

Query Editor		Query History
1	explain analyze select count(sometext) from testindex	
Data Output		Explain Messages Notifications
QUERY PLAN		
text		
1	Finalize Aggregate (cost=11614.55..11614.56 rows=1 width=8) (actual time=116.985..121.247 rows=1 loops=1)	
2	-> Gather (cost=11614.33..11614.54 rows=2 width=8) (actual time=116.539..121.225 rows=3 loops=1)	
3	Workers Planned: 2	
4	Workers Launched: 2	
5	-> Partial Aggregate (cost=10614.33..10614.34 rows=1 width=8) (actual time=113.404..113.408 rows=1 loops=3)	
6	-> Parallel Seq Scan on testindex (cost=0.00..9572.67 rows=416667 width=9) (actual time=0.013..60.408 rows=333333 loops=3)	
7	Planning Time: 0.053 ms	
8	Execution Time: 121.282 ms	

Btree

postgres/postgres@Trolleybus

Query Editor

Query History

1

explain analyze select count(sometext) from testindex

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1

Finalize Aggregate (cost=11614.55..11614.56 rows=1 width=8) (actual time=125.267..136.950 rows=1 loops=1)

2

-> Gather (cost=11614.33..11614.54 rows=2 width=8) (actual time=124.219..136.912 rows=3 loops=1)

3

Workers Planned: 2

4

Workers Launched: 2

5

-> Partial Aggregate (cost=10614.33..10614.34 rows=1 width=8) (actual time=113.638..113.641 rows=1 loops=3)

6

-> Parallel Seq Scan on testindex (cost=0.00..9572.67 rows=416667 width=9) (actual time=0.023..61.912 rows=333333 loops=3)

7

Planning Time: 0.200 ms

8

Execution Time: 137.031 ms

BRIN

Query Editor		Query History
1	<code>explain analyze select count(sometext) from testindex</code>	
<div>Data OutputExplainMessagesNotifications</div>		
<div><div>QUERY PLAN</div><div>text</div><div></div></div>		
1	Finalize Aggregate (cost=11614.55..11614.56 rows=1 width=8) (actual time=112.371..118.282 rows=1 loops=1)	
2	-> Gather (cost=11614.33..11614.54 rows=2 width=8) (actual time=112.268..118.273 rows=3 loops=1)	
3	Workers Planned: 2	
4	Workers Launched: 2	
5	-> Partial Aggregate (cost=10614.33..10614.34 rows=1 width=8) (actual time=105.129..105.130 rows=1 loops=3)	
6	-> Parallel Seq Scan on testindex (cost=0.00..9572.67 rows=416667 width=9) (actual time=0.013..56.469 rows=333333 loops=3)	
7	Planning Time: 0.107 ms	
8	Execution Time: 118.343 ms	

Вихідні дані

Query Editor

Query History

1

select count(sometext) from testindex

Data Output

Explain

Messages

Notifications

count
bigint

1

1000000

Висновок



Жоден з методів індексації не вплинув на швидкість обробки запиту, оскільки агрегатна функція “count()” не використовує безпосередній вміст рядка.

Приклад з агрегатними функціями і групуванням

Без індексації

Query Editor Query History	
1	<code>explain analyze select count(sometext) from testindex group by sometext</code>
Data Output Explain Messages Notifications	
	QUERY PLAN text
1	GroupAggregate (cost=132154.34..149654.34 rows=1000000 width=17) (actual time=118059.287..181499.564 rows=1000000 loops=1)
2	Group Key: sometext
3	-> Sort (cost=132154.34..134654.34 rows=1000000 width=9) (actual time=118059.122..180174.523 rows=1000000 loops=1)
4	Sort Key: sometext
5	Sort Method: external merge Disk: 19464kB
6	-> Seq Scan on testindex (cost=0.00..15406.00 rows=1000000 width=9) (actual time=63.911..241.832 rows=1000000 loops=1)
7	Planning Time: 0.060 ms
8	JIT:
9	Functions: 7
10	Options: Inlining false, Optimization false, Expressions true, Deforming true
11	Timing: Generation 1.673 ms, Inlining 0.000 ms, Optimization 14.297 ms, Emission 48.003 ms, Total 63.973 ms
12	Execution Time: 181790.136 ms

Btree

 postgres/postgres@Trolleybus ▾	
Query Editor Query History	
1	<code>explain analyze select count(sometext) from testindex group by sometext</code>
Data Output Explain Messages Notifications	
	QUERY PLAN text 
1	GroupAggregate (cost=0.42..67000.33 rows=1000000 width=17) (actual time=0.488..3098.245 rows=1000000 loops=1)
2	Group Key: sometext
3	-> Index Only Scan using index_sometext on testindex (cost=0.42..52000.33 rows=1000000 width=9) (actual time=0.474..2398.588 rows=1000000 loops=1)
4	Heap Fetches: 1000000
5	Planning Time: 0.117 ms
6	Execution Time: 3180.527 ms

BRIN

Query Editor		Query History
1	explain analyze select count(sometext) from testindex group by sometext	
Data Output		
Explain		
Messages		
Notifications		
QUERY PLAN		
text		
1	GroupAggregate (cost=132154.34..149654.34 rows=1000000 width=17) (actual time=119982.522..184491.985 rows=1000000 loops=1)	
2	Group Key: sometext	
3	-> Sort (cost=132154.34..134654.34 rows=1000000 width=9) (actual time=119982.411..183102.298 rows=1000000 loops=1)	
4	Sort Key: sometext	
5	Sort Method: external merge Disk: 19464kB	
6	-> Seq Scan on testindex (cost=0.00..15406.00 rows=1000000 width=9) (actual time=15.961..192.911 rows=1000000 loops=1)	
7	Planning Time: 0.075 ms	
8	JIT:	
9	Functions: 7	
10	Options: Inlining false, Optimization false, Expressions true, Deforming true	
11	Timing: Generation 0.675 ms, Inlining 0.000 ms, Optimization 0.330 ms, Emission 15.429 ms, Total 16.433 ms	
12	Execution Time: 184697.978 ms	

Вихідні дані

Query Editor

Query History

1 select count(sometext) from testindex group by sometext

Data Output

Explain

Messages

Notifications

	count bigint	🔒
1	1	
2	1	
3	1	
4	1	
5	1	
6	1	
7	1	
8	1	
9	1	
10	1	
11	1	
12	1	
13	1	
14	1	
15	1	
16	1	

Висновок

Індексація Btree значно прискорила роботу, індексація BRIN не вплинула на час запиту у порівнянні з експериментом без індексації. Неефективність BRIN пов'язана з тим що у даному випадку серед проіндексованих даних(текстові рядки) не можливо знайти мінімальний або максимальний елемент, що необхідно для ефективної роботи даного методу індексації. Пояснень ефективності методу Btree в даному експерименті не знайдено.

Приклад з фільтрацією і сортуванням

Без індексації

Query Editor		Query History			
1	explain analyze select * from testindex where sometext like '생%' order by sometext asc				
Data Output			Explain	Messages	Notifications
QUERY PLAN					
text					
1	Gather Merge (cost=11615.49..11625.29 rows=84 width=13) (actual time=102.400..109.938 rows=24 loops=1)				
2	Workers Planned: 2				
3	Workers Launched: 2				
4	-> Sort (cost=10615.47..10615.57 rows=42 width=13) (actual time=80.054..80.060 rows=8 loops=3)				
5	Sort Key: sometext				
6	Sort Method: quicksort Memory: 25kB				
7	Worker 0: Sort Method: quicksort Memory: 25kB				
8	Worker 1: Sort Method: quicksort Memory: 25kB				
9	-> Parallel Seq Scan on testindex (cost=0.00..10614.33 rows=42 width=13) (actual time=6.019..79.645 rows=8 loops=3)				
10	Filter: (((sometext)::text ~~ '생% '::text)				
11	Rows Removed by Filter: 333325				
12	Planning Time: 0.083 ms				
13	Execution Time: 109.976 ms				

Btree

postgres/postgres@Trolleybus

Query Editor

Query History

1

explain analyze select * from testindex where sometext like '뽕%' order by sometext asc

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1

Gather Merge (cost=11615.49..11625.29 rows=84 width=13) (actual time=96.402..104.223 rows=24 loops=1)

2

Workers Planned: 2

3

Workers Launched: 2

4

-> Sort (cost=10615.47..10615.57 rows=42 width=13) (actual time=84.986..84.990 rows=8 loops=3)

5

Sort Key: sometext

6

Sort Method: quicksort Memory: 25kB

7

Worker 0: Sort Method: quicksort Memory: 25kB

8

Worker 1: Sort Method: quicksort Memory: 25kB

9

-> Parallel Seq Scan on testindex (cost=0.00..10614.33 rows=42 width=13) (actual time=5.919..84.574 rows=8 loops=3)

10

Filter: ((sometext)::text ~~ '뽕%':text)

11

Rows Removed by Filter: 333325

12

Planning Time: 0.163 ms

13

Execution Time: 104.261 ms

BRIN

postgres/postgres@Trolleybus

Query Editor

Query History

1

explain analyze select * from testindex where sometext like '뽕%' order by sometext asc

Data Output

Explain

Messages

Notifications

QUERY PLAN

text

1

Gather Merge (cost=11615.49..11625.29 rows=84 width=13) (actual time=97.648..108.642 rows=24 loops=1)

2

Workers Planned: 2

3

Workers Launched: 2

4

-> Sort (cost=10615.47..10615.57 rows=42 width=13) (actual time=77.687..77.690 rows=8 loops=3)

5

Sort Key: sometext

6

Sort Method: quicksort Memory: 25kB

7

Worker 0: Sort Method: quicksort Memory: 25kB

8

Worker 1: Sort Method: quicksort Memory: 25kB

9

-> Parallel Seq Scan on testindex (cost=0.00..10614.33 rows=42 width=13) (actual time=3.793..77.328 rows=8 loops=3)

10

Filter: ((sometext)::text ~~ '뽕%':text)

11

Rows Removed by Filter: 333325

12

Planning Time: 0.136 ms

13

Execution Time: 108.672 ms

Вихідні дані

Query Editor

Query History

1select * from testindex where sometext like '생%' order by sometext asc

Data Output

Explain

Messages

Notifications

	<div>id</div> <div>integer</div>	<div>sometext</div> <div>character varying (64)</div>	
1	766464	생애돌	
2	62432	생애0	
3	291339	생애차	
4	227141	생애차	
5	12269	생애차	
6	707774	생애차	
7	25326	생애차	
8	401978	생애차	
9	159182	생애차	
10	141270	생애차	
11	254259	생애차	
12	190971	생애차	
13	57325	생애차	
14	757065	생애차	
15	202240	생애차	
16	37455	생애차	
17	651497	생애차	
18	940632	생애차	
19	618516	생애차	
20	90910	생애차	

Висновок

Жоднен метод індексації не вплинув на швидкість сортування, оскільки в розглянутому випадку проіндексовані данні(рядки) неможливо відсортувати за ознакою більше/менше.

Загальні висновки щодо індексації

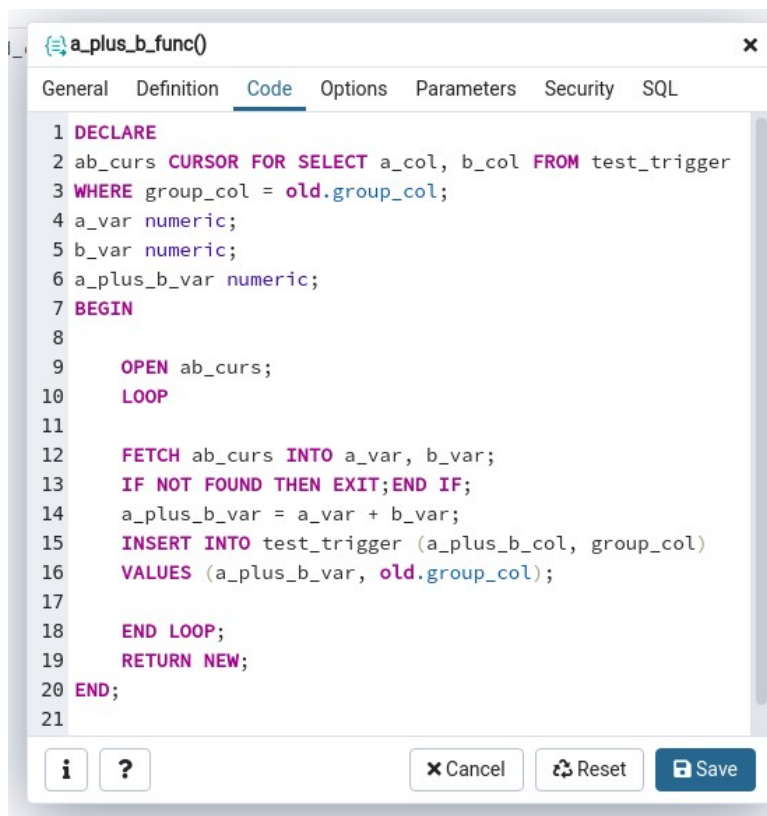
- Метод індексації BRIN не є ефективним для текстових рядків.
- Індексація не є ефективною, якщо запит не оперує безпосередньо вмістом рядків.
- Метод індексації BRIN не є ефективним у випадку неможливості групування даних за ознакою.
- Індексація є ефективною для великої кількості даних, що змінюються нечасто.

Звіт до пункту завдання №3

Код тригера

```
CREATE TRIGGER aplusb  
AFTER UPDATE  
ON test_trigger  
FOR EACH ROW  
EXECUTE PROCEDURE a_plus_b_func();
```






Код тригерної функції








Команда, що ініціює виконання тргера



Початкова таблиця з даними

Data Output		Explain	Messages	Notifications	
 a_col integer	 b_col integer	 a_plus_b_col integer	 group_col integer	 Id_col integer	
1	4	5	[null]	2	22
2	4	4	[null]	3	23
3	2	1	[null]	8	24
4	2	7	[null]	9	25
5	3	5	[null]	1	26
6	6	2	[null]	3	27
7	1	4	[null]	3	28

Таблиця після вионання команд, що ініціюють виконання тргера

Data Output		Explain	Messages	Notifications	
	a_col integer	 b_col integer	 a_plus_b_col integer	 group_col integer	 id_col integer
1	4	5	[null]	2	22
2	4	4	[null]	3	23
3	2	1	[null]	8	24
4	2	7	[null]	9	25
5	6	2	[null]	3	27
6	10	5	[null]	1	26
7	[null]	[null]	15	1	29
8	10	4	[null]	3	28
9	[null]	[null]	8	3	30
10	[null]	[null]	8	3	31
11	[null]	[null]	14	3	32

Пояснення

Після оновлення рядка тригер сумує стовпці “a_col” та “b_col” та записує результат до стовпця “a_plus_b_col”. Завдяки курсорному циклу, сума стовпців обчислюються не лише для рядка, що оновлюється, а й для рядків усієї групи, до якої входить рядок, що оновлюється. Приналежність до групи задається стовпцем “group_id”.