ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №4

Обмеження цілісності даних та індекси в SQL

Виконав:

Ст *Кравець Ольга*

ПМО-21

Оцінка

Перевірила:

доц. Малець Р.Б.

2022

Тема: Обмеження цілісності даних та індекси в SQL.

Мета роботи: Ознайомлення з поняттями обмеження цілісності даних та індексами в SQL, їх створенням і використанням

Хід роботи

1. Опрацювала теоретичний матеріал.

2. Проаналізувала наявні обмеження цілісності даних в створених таблицях та додала відсутні, а саме unique та not null constraint для поля id у всіх таблицях.

```
CREATE TABLE IF NOT EXISTS public.client
(
    client_id integer NOT NULL,
    cl_name character(64) COLLATE pg_catalog."default" NOT NULL,
    cl_address text COLLATE pg_catalog."default" NOT NULL,
    registered registered_check NOT NULL,
    bank_account bigint NOT NULL,
    responsible_persons character(100) COLLATE pg_catalog."default" NOT NULL,
    cl_telephone telephone_check COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT client_pkey PRIMARY KEY (client_id),
    CONSTRAINT unique_client_bank_account UNIQUE (bank_account),
    CONSTRAINT unique_client_id UNIQUE (client_id)
)

CREATE TABLE IF NOT EXISTS public.cl_order
(
    id_goods integer NOT NULL,
    or_number integer NOT NULL,
    delivery_method delivery_method_check COLLATE pg_catalog."default" NOT NULL,
    amount smallint,
    client_id integer NOT NULL,
    price price_check NOT NULL,
    total_price integer,
    provider_id integer NOT NULL,
    CONSTRAINT cl_order_pkey PRIMARY KEY (or_number),
    CONSTRAINT cl_order_id_goods_key UNIQUE (id_goods)
)
```

```
CREATE TABLE IF NOT EXISTS public.goods
(
    id_goods integer NOT NULL,
    price weight_check NOT NULL,
    weight weight_check NOT NULL,
    go_name character(64) COLLATE pg_catalog."default" NOT NULL,
    color character(32) COLLATE pg_catalog."default" NOT NULL,
    material character(64) COLLATE pg_catalog."default" NOT NULL,
    provider_id integer NOT NULL,
    quantity integer NOT NULL,
    CONSTRAINT goods_pkey PRIMARY KEY (id_goods),
    CONSTRAINT unique_id_goods UNIQUE (id_goods)
        INCLUDE(id_goods),
    CONSTRAINT goods_id_goods_fkey FOREIGN KEY (id_goods)
        REFERENCES public.cl_order (id_goods) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT provider_idkey FOREIGN KEY (provider_id)
        REFERENCES public.provider (provider_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

```
CREATE TABLE IF NOT EXISTS public.provider
(
    provider_id integer NOT NULL,
    id_number id_number_check NOT NULL,
    pr_name character(64) COLLATE pg_catalog."default" NOT NULL,
    pr_address text COLLATE pg_catalog."default" NOT NULL,
    pr_telephone telephone_check COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT provider_pkey PRIMARY KEY (provider_id),
    CONSTRAINT unique_id_number UNIQUE (id_number),
    CONSTRAINT unique_provider_id UNIQUE (provider_id)
)
```

3. Створила індекс pr_name_indx до таблиці provider, попередньо додавши до неї 200 елементів.

Результати пошуку до створення індексу pr_name_indx:

Query Editor

```
1  select count(*)
2  from provider
3  where pr_name = 'Letro'
```

Data Output | Explain | Messages | Query History | Notifications

```
Successfully run. Total query runtime: 106 msec.
1 rows affected.
```

Результати пошуку після створення індексу pr_name_indx:

Query Editor

```
1  EXPLAIN SELECT *
2  FROM public.provider
3  WHERE pr_name = 'Letro'
```

Data Output | Explain | Messages | Query History | Notifications

	QUERY PLAN text
1	Seq Scan on provider (cost=0.00..7.50 rows=1 width=112)
2	[...] Filter: (pr_name = 'Letro'::bpchar)

Query Editor

```
1  set enable_seqscan = off;
2  EXPLAIN SELECT *
3  FROM public.provider
4  WHERE pr_name = 'Letro'
```

Data Output | Explain | Messages | Query History | Notifications

	QUERY PLAN text
1	Index Scan using pr_name_indx on provider (cost=0.27..8.29 rows=1 width=112)
2	[...] Index Cond: (pr_name = 'Letro'::bpchar)

Створила індекс pr_address_indx до таблиці provider, попередньо додавши до неї 200 елементів.

Результати пошуку до створення індексу pr_address_indx:

Query Editor

```
1  select count(*)
2  from provider
3  where pr_address = 'Львів'
```

Data Output | Explain | Messages | Query History | Notifications

```
Successfully run. Total query runtime: 105 msec.
1 rows affected.
```

Результати пошуку після створення індексу pr_address_indx:

Query Editor

```
1  EXPLAIN SELECT *
2  FROM public.provider
3  WHERE pr_address = 'Львів'
```

Data Output | Explain | Messages | Query History | Notifications

	QUERY PLAN text
1	Seq Scan on provider (cost=0.00..7.50 rows=21 width=112)
2	[...] Filter: (pr_address = 'Львів'::text)

Query Editor

```
1  set enable_seqscan = off;
2  EXPLAIN SELECT *
3  FROM public.provider
4  WHERE pr_address = 'Львів'
```

Data Output | Explain | Messages | Query History | Notifications

	QUERY PLAN text
1	Bitmap Heap Scan on provider (cost=4.31..9.57 rows=21 width=112)
2	[...] Recheck Cond: (pr_address = 'Львів'::text)
3	[...] -> Bitmap Index Scan on pr_address_indx (cost=0.00..4.30 rows=21 width=0)
4	[...] Index Cond: (pr_address = 'Львів'::text)

Висновок: під час виконання лабораторно роботи я ознайомилась з поняттями обмеження цілісності даних та індексами в SQL, їх створенням і використанням.