Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп'ютерних систем

ЛАБОРАТОРНА РОБОТА №2

з дисципліни

"Бази даних і засоби управління"

ТЕМА: "Засоби оптимізації роботи СУБД PostgreSQL"

Група: КВ-13

Виконав: Яцков М. Ю.

Оцінка:

Київ — 2023

Метою роботи є здобуття практичних навичок використання засобів оптимізації СУБД PostgreSQL.

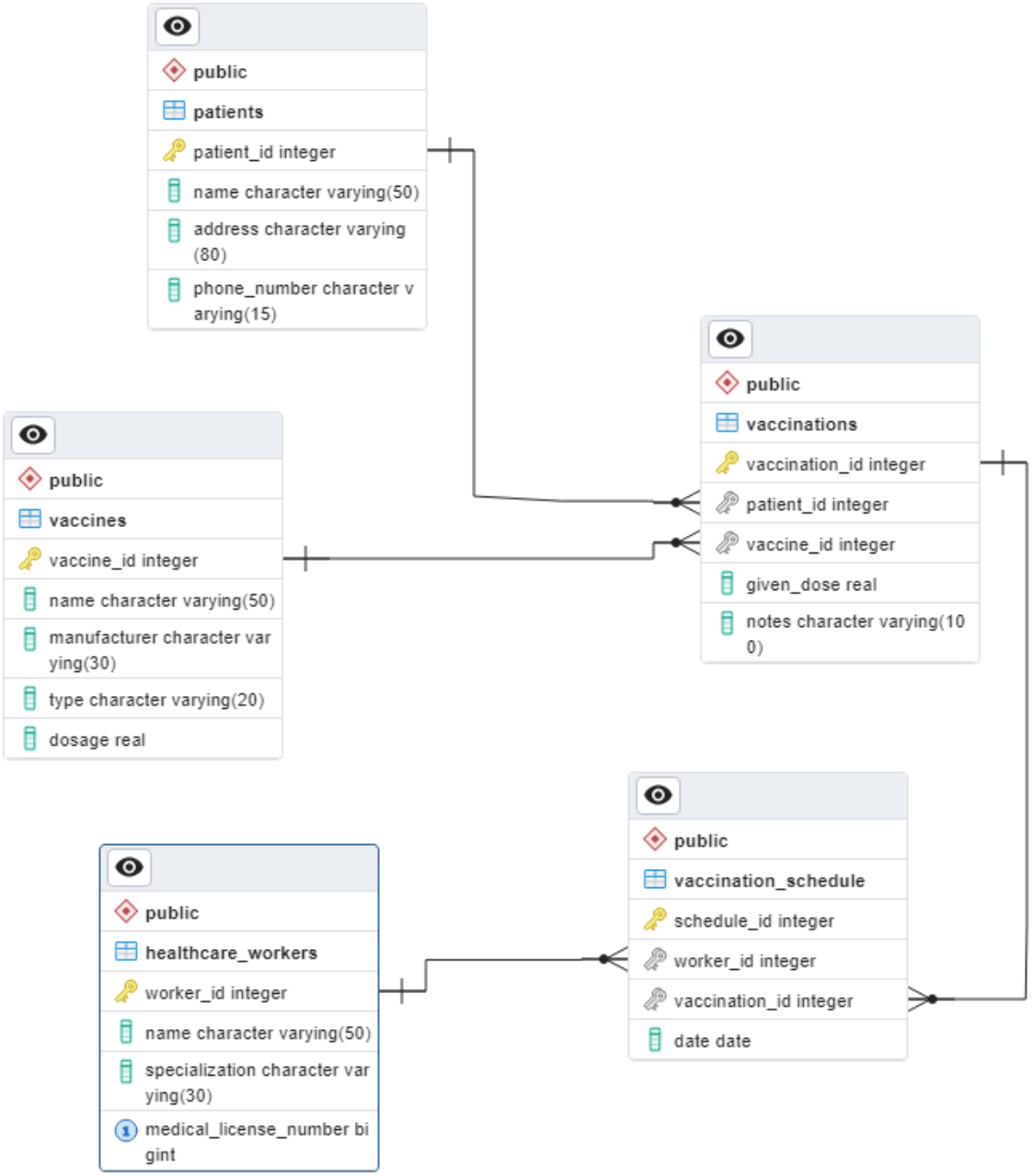
Завдання роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC РГР у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.
4. Навести приклади та проаналізувати рівні ізоляції транзакцій у PostgreSQL.

Варіанти

№ варіанта	*Види індексів*	*Умови для тригера*
25	*BTree, GIN*	*after delete, insert*

Посилання на репозиторій - https://github.com/Coltenus/BD-Lab-2
Посилання на телеграм - https://t.me/Coltenus


public
patients
patient_id integer
name character varying(50)
address character varying (80)
phone_number character v arying(15)
public
vaccinations
vaccination_id integer
patient_id integer
vaccine_id integer
given_dose real
notes character varying(10 0)
public
vaccines
vaccine_id integer
name character varying(50)
manufacturer character var ying(30)
type character varying(20)
dosage real
public
vaccination_schedule
schedule_id integer
worker_id integer
vaccination_id integer
date date
public
healthcare_workers
worker_id integer
name character varying(50)
specialization character var ying(30)
medical_license_number bi gint

C# PatientsModel.cs

```
5   namespace bd_rgr
6   {
        8 usages
7       public class PatientsContext : BaseContext
8       {
9           [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
10          [Column(name: "patient_id")]
            11 usages
11          public int PatientId { get; set; }
            7 usages
12          public string Name { get; set; }
            7 usages
13          public string Address { get; set; }
            7 usages
14          public string PhoneNumber { get; set; }
15
16      }
```

C# PatientsModel.cs

```
25      }
        13 usages
26      public virtual DbSet<PatientsContext> PContexts { get; set; }
27
28      protected override void OnModelCreating(ModelBuilder modelBuilder)
29      {
30          modelBuilder.Entity<PatientsContext>(entity =>
31          {
32              entity.ToTable(TableName);
33              entity.HasKey(p :PatientsContext => p.PatientId);
34              entity.Property(e :PatientsContext => e.Name)
35                  .HasColumnName(TableFields[1])
36                  .IsRequired();
37              entity.Property(e :PatientsContext => e.Address)
38                  .HasColumnName(TableFields[2])
39                  .IsRequired();
40              entity.Property(e :PatientsContext => e.PhoneNumber)
41                  .HasColumnName(TableFields[3])
42                  .IsRequired();
43          });
44      }
```

C# VaccinesModel.cs

```
4
5   namespace bd_rgr
6   {
        8 usages
7       public class VaccinesContext : BaseContext
8       {
9           [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
10          [Column(name:"vaccine_id")]
            11 usages
11          public int VaccineId { get; set; }
            7 usages
12          public string Name { get; set; }
            7 usages
13          public string Manufacturer { get; set; }
            7 usages
14          public string Type { get; set; }
            9 usages
15          public float Dosage { get; set; }
16      }
```

C# VaccinesModel.cs

```
26      }
        14 usages
27      public virtual DbSet<VaccinesContext> VContexts { get; set; }
28
29      protected override void OnModelCreating(ModelBuilder modelBuilder)
30      {
31          modelBuilder.Entity<VaccinesContext>(entity =>
32          {
33              entity.ToTable(TableName);
34              entity.HasKey(v :VaccinesContext => v.VaccineId);
35              entity.Property(e :VaccinesContext => e.Name)
36                  .HasColumnName(TableFields[1])
37                  .IsRequired();
38              entity.Property(e :VaccinesContext => e.Manufacturer)
39                  .HasColumnName(TableFields[2])
40                  .IsRequired();
41              entity.Property(e :VaccinesContext => e.Type)
42                  .HasColumnName(TableFields[3])
43                  .IsRequired();
44              entity.Property(e :VaccinesContext => e.Dosage)
45                  .HasColumnName(TableFields[4])
46                  .IsRequired();
47          });
48      }
```

C# VaccinationsModel.cs

```
4
5       namespace bd_rgr
6       {
            8 usages
7           public class VaccinationsContext : BaseContext
8           {
9               [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
10              [Column(name: "vaccination_id")]
                11 usages
11              public int VaccinationId { get; set; }
                10 usages
12              public int PatientID { get; set; }
                10 usages
13              public int VaccineID { get; set; }
                9 usages
14              public float GivenDose { get; set; }
                7 usages
15              public string? Notes { get; set; }
16          }
```

C# VaccinationsModel.cs

```
28
29      protected override void OnModelCreating(ModelBuilder modelBuilder)
30      {
31          modelBuilder.Entity<VaccinationsContext>(entity =>
32          {
33              entity.ToTable(TableName);
34              entity.HasKey(v :VaccinationsContext => v.VaccinationId);
35              entity.HasOne<PatientsContext>() // ReferenceNavigationBuilder<VaccinationsContext,...>
36                  .WithMany().HasForeignKey(v :VaccinationsContext => v.PatientID);
37              entity.Property(v :VaccinationsContext => v.PatientID)
38                  .HasColumnName(TableFields[1]);
39              entity.HasOne<VaccinesContext>() // ReferenceNavigationBuilder<VaccinationsContext,...>
40                  .WithMany().HasForeignKey(v :VaccinationsContext => v.VaccineID);
41              entity.Property(v :VaccinationsContext => v.VaccineID)
42                  .HasColumnName(TableFields[2]);
43              entity.Property(e :VaccinationsContext => e.GivenDose)
44                  .HasColumnName(TableFields[3])
45                  .IsRequired();
46              entity.Property(e :VaccinationsContext => e.Notes)
47                  .HasColumnName(TableFields[4]);
48          });
49      }
```

```
C# HealthcareWorkersModel.cs ×

5   namespace bd_rgr
6   {
        8 usages
7       public class HealthCareWorkersContext : BaseContext
8       {
9           [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
10          [Column(name: "worker_id")]
            11 usages
11          public int WorkerId { get; set; }
            7 usages
12          public string Name { get; set; }
            7 usages
13          public string Specialization { get; set; }
            8 usages
14          public long MedicalLicenseNumber { get; set; }
15
16      }
```

```
C# HealthcareWorkersModel.cs ×

26      public virtual DbSet<HealthCareWorkersContext> HwContexts { get; set; }
27
28      protected override void OnModelCreating(ModelBuilder modelBuilder)
29      {
30          modelBuilder.Entity<HealthCareWorkersContext>(entity =>
31          {
32              entity.ToTable(TableName);
33              entity.HasKey(h :HealthCareWorkersContext => h.WorkerId);
34              entity.Property(e :HealthCareWorkersContext => e.Name)
35                  .HasColumnName(TableFields[1])
36                  .IsRequired();
37              entity.Property(e :HealthCareWorkersContext => e.Specialization)
38                  .HasColumnName(TableFields[2])
39                  .IsRequired();
40              entity.Property(e :HealthCareWorkersContext => e.MedicalLicenseNumber)
41                  .HasColumnName(TableFields[3])
42                  .IsRequired();
43          });
44      }
```

C# VaccinationScheduleModel.cs

```
4
5   namespace bd_rgr
6   {
        7 usages
7       public class VaccinationScheduleContext : BaseContext
8       {
9           [Key, DatabaseGenerated(DatabaseGeneratedOption.Identity)]
10          [Column(name: "schedule_id")]
            11 usages
11          public int ScheduleId { get; set; }
            10 usages
12          public int WorkerID { get; set; }
            10 usages
13          public int VaccinationID { get; set; }
            9 usages
14          public DateTime Date { get; set; }
15
16      }
```

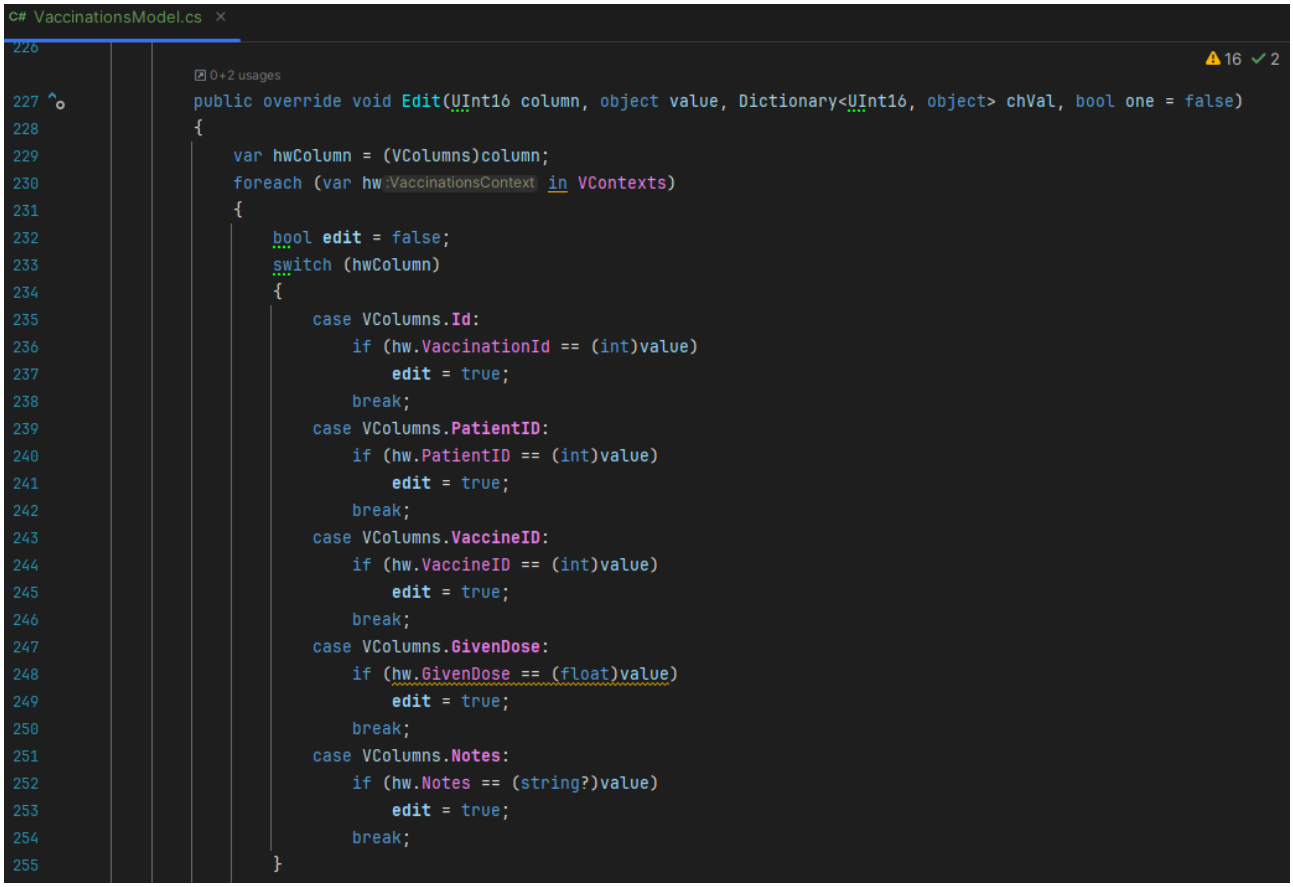
C# VaccinationScheduleModel.cs

```
26      public virtual DbSet<VaccinationScheduleContext> VSContexts { get; set; }
27
28      protected override void OnModelCreating(ModelBuilder modelBuilder)
29      {
30          modelBuilder.Entity<VaccinationScheduleContext>(entity =>
31          {
32              entity.ToTable(TableName);
33              entity.HasKey(v :VaccinationScheduleContext => v.ScheduleId);
34              entity.HasOne<HealthCareWorkersContext>() // ReferenceNavigationBuilder<VaccinationSch
35                  .WithMany().HasForeignKey(v :VaccinationScheduleContext => v.WorkerID);
36              entity.Property(v :VaccinationScheduleContext => v.WorkerID)
37                  .HasColumnName(TableFields[1]);
38              entity.HasOne<VaccinationsContext>() // ReferenceNavigationBuilder<VaccinationScheduleC
39                  .WithMany().HasForeignKey(v :VaccinationScheduleContext => v.VaccinationID);
40              entity.Property(v :VaccinationScheduleContext => v.VaccinationID)
41                  .HasColumnName(TableFields[2]);
42              entity.Property(e :VaccinationScheduleContext => e.Date)
43                  .HasColumnName(TableFields[3])
44                  .IsRequired();
45          });
46      }
```


PatientsModel.cs
0+1 usages
11
public override void Add(List<Dictionary<string, object>> values)
{
try
{
foreach(var value :Dictionary<string,object> in values)
{
int id = 1;
foreach (var hw :PatientsContext in PContexts.OrderBy(d :PatientsContext =>d.PatientId))
{
if (hw.PatientId == id) id++;
else break;
}
PContexts.Add(entity: new PatientsContext()
{ PatientId = id, Name = (string)value["name"], Address = (string)value["address"],
PhoneNumber = (string)value["phone_number"] });
SaveChanges();
}
}
catch (DbUpdateException ex)
{
Console.WriteLine(ex.Message);
Console.WriteLine(ex.InnerException?.Message);
}
}

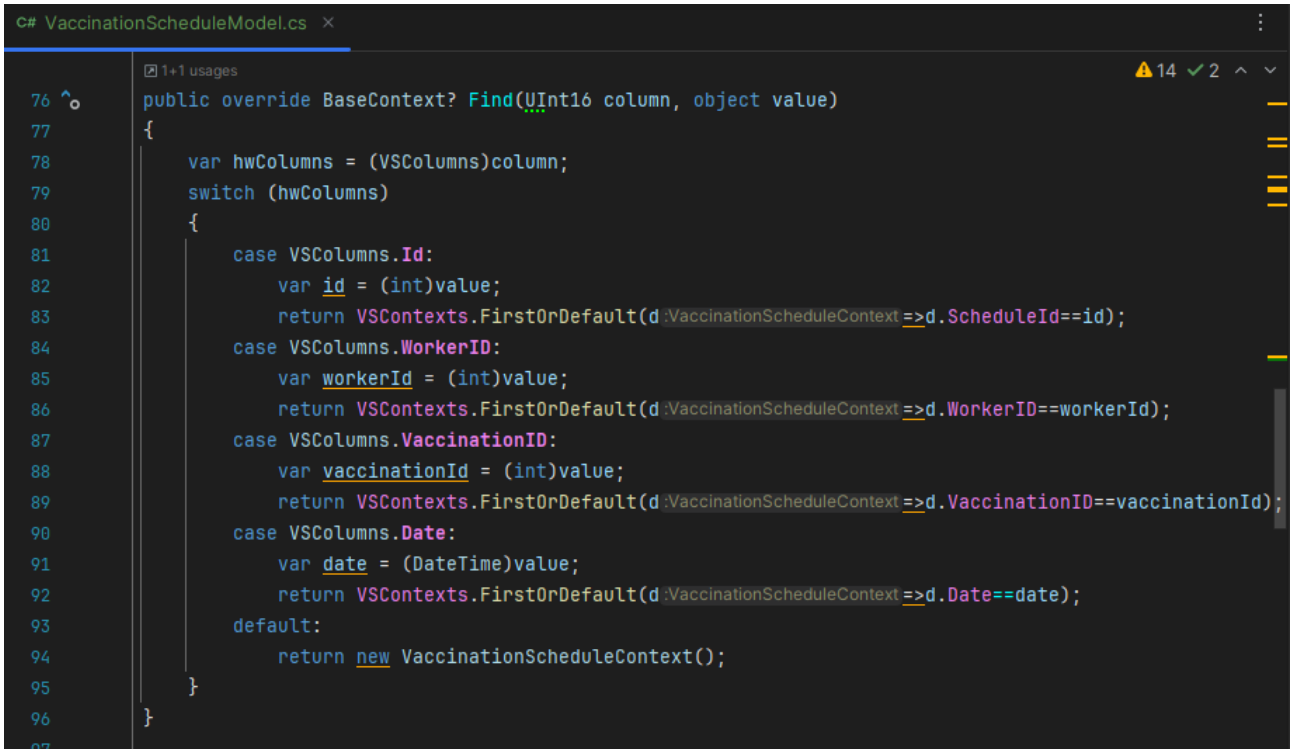
C# HealthcareWorkersModel.cs

```
0+1 usages
public override void Remove(UInt16 column, object value, bool greater)
{
    var hwColumn = (HWColumns)column;
    foreach (var hw :HealthCareWorkersContext in HwContexts)
    {
        bool remove = false;
        switch (hwColumn)
        {
            case HWColumns.Id:
                if (greater && hw.WorkerId > (int)value || !greater && hw.WorkerId < (int)value)
                    remove = true;
                break;
            case HWColumns.Name:
                break;
            case HWColumns.Specialization:
                break;
            case HWColumns.MedicalLicenseNumber:
                if (greater && hw.MedicalLicenseNumber > (long)value
                    || !greater && hw.MedicalLicenseNumber < (long)value)
                    remove = true;
                break;
        }
        if(remove) HwContexts.Remove(hw);
    }

    try
    {
        SaveChanges();
    }
    catch (DbUpdateException ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine(ex.InnerException?.Message);
    }
}
```


VaccinationsModel.cs
16 2
0+2 usages
public override void Edit(UInt16 column, object value, Dictionary<UInt16, object> chVal, bool one = false)
{
var hwColumn = (VColumns)column;
foreach (var hw:VaccinationsContext in VContexts)
{
bool edit = false;
switch (hwColumn)
{
case VColumns.Id:
if (hw.VaccinationId == (int)value)
edit = true;
break;
case VColumns.PatientID:
if (hw.PatientID == (int)value)
edit = true;
break;
case VColumns.VaccineID:
if (hw.VaccineID == (int)value)
edit = true;
break;
case VColumns.GivenDose:
if (hw.GivenDose == (float)value)
edit = true;
break;
case VColumns.Notes:
if (hw.Notes == (string?)value)
edit = true;
break;
}

C# VaccinationsModel.cs

```
257                 if (edit)
258                 {
259                     bool needBreak = false;
260                     for(var col = VColumns.Id; col<=VColumns.GivenDose; col++)
261                     {
262                         UInt16 key = (UInt16)col;
263                         if(!chVal.Keys.Contains(key)) continue;
264                         switch (col)
265                         {
266                             case VColumns.Id:
267                                 Console.WriteLine("This column can not be modified.");
268                                 needBreak = true;
269                                 break;
270                             case VColumns.PatientID:
271                                 hw.PatientID = (int)chVal[key];
272                                 break;
273                             case VColumns.VaccineID:
274                                 hw.VaccineID = (int)chVal[key];
275                                 break;
276                             case VColumns.GivenDose:
277                                 hw.GivenDose = (float)chVal[key];
278                                 break;
279                             case VColumns.Notes:
280                                 hw.Notes = (string?)chVal[key];
281                                 break;
282                         }
283                     }
284                     if(one || needBreak) break;
285                 }
286             }
287 
288             try
289             {
290                 SaveChanges();
291             }
292             catch (DbUpdateException ex)
293             {
294                 Console.WriteLine(ex.Message);
295                 Console.WriteLine(ex.InnerException?.Message);
296             }
```


C# VaccinationScheduleModel.cs
1+1 usages
14 2
public override BaseContext? Find(UInt16 column, object value)
{
var hwColumns = (VSColumns)column;
switch (hwColumns)
{
case VSColumns.Id:
var id = (int)value;
return VSContexts.FirstOrDefault(d :VaccinationScheduleContext =>d.ScheduleId==id);
case VSColumns.WorkerID:
var workerId = (int)value;
return VSContexts.FirstOrDefault(d :VaccinationScheduleContext =>d.WorkerID==workerId);
case VSColumns.VaccinationID:
var vaccinationId = (int)value;
return VSContexts.FirstOrDefault(d :VaccinationScheduleContext =>d.VaccinationID==vaccinationId);
case VSColumns.Date:
var date = (DateTime)value;
return VSContexts.FirstOrDefault(d :VaccinationScheduleContext =>d.Date==date);
default:
return new VaccinationScheduleContext();
}
}

```
CREATE INDEX btreeexample ON vaccination_schedule USING BTREE (schedule_id);
```

Data Output | Messages | Explain | Noti

```
CREATE INDEX

Query returned successfully in 34 msec.
```

```
SELECT * FROM vaccination_schedule WHERE schedule_id >= 3333 AND schedule_id <=
6666 OR schedule_id < 1200 ORDER BY date, worker_id, schedule_id;
```

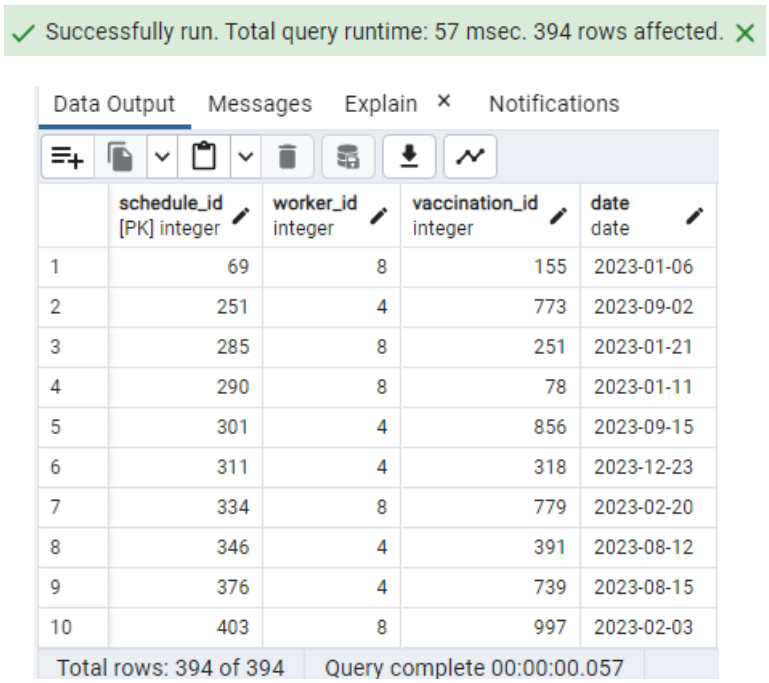
Successfully run. Total query runtime: 62 msec. 4533 rows affected.

Data Output | Messages | Explain | Notifications

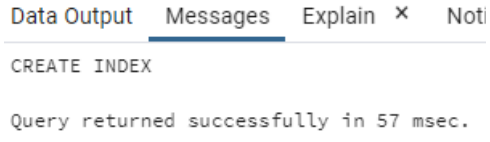
	schedule_id [PK] integer	worker_id integer	vaccination_id integer	date date
1	4402	1	900	2023-01-01
2	5829	3	575	2023-01-01
3	702	4	324	2023-01-01
4	6572	6	259	2023-01-01
5	5117	7	544	2023-01-01
6	5745	10	22	2023-01-01
7	844	11	431	2023-01-01
8	6530	11	229	2023-01-01
9	6564	14	884	2023-01-01
10	5122	18	961	2023-01-01

Total rows: 1000 of 4533 | Query complete 00:00:00.062

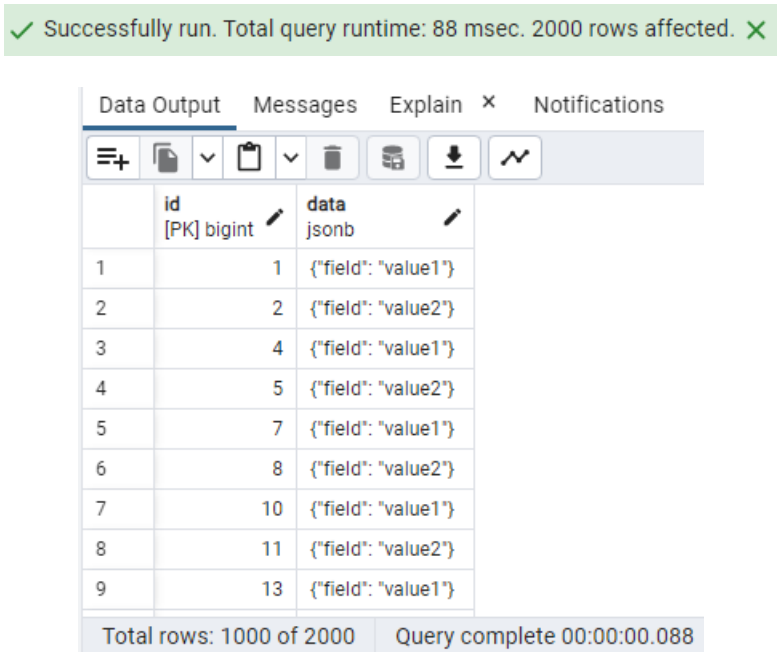
```
SELECT * FROM vaccination_schedule WHERE worker_id = 4 AND date > '2023-08-
01'::date OR worker_id = 8 AND date < '2023-04-01'::date GROUP BY worker_id,
schedule_id;
```



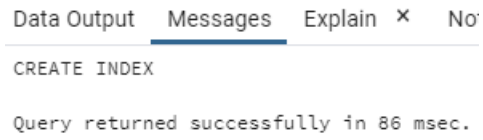

```
CREATE TABLE gintable (
  id bigserial PRIMARY KEY,
  data jsonb
);
DO
$do$
DECLARE
     i int;
BEGIN
FOR i in 1..1000
LOOP
INSERT INTO gintable(data) VALUES ('{"field": "value1"}');
INSERT INTO gintable(data) VALUES ('{"field": "value2"}');
INSERT INTO gintable(data) VALUES ('{"other_field": "value42"}');
END LOOP;
END;
$do$;
CREATE INDEX ginexample ON gintable USING gin(data);
```



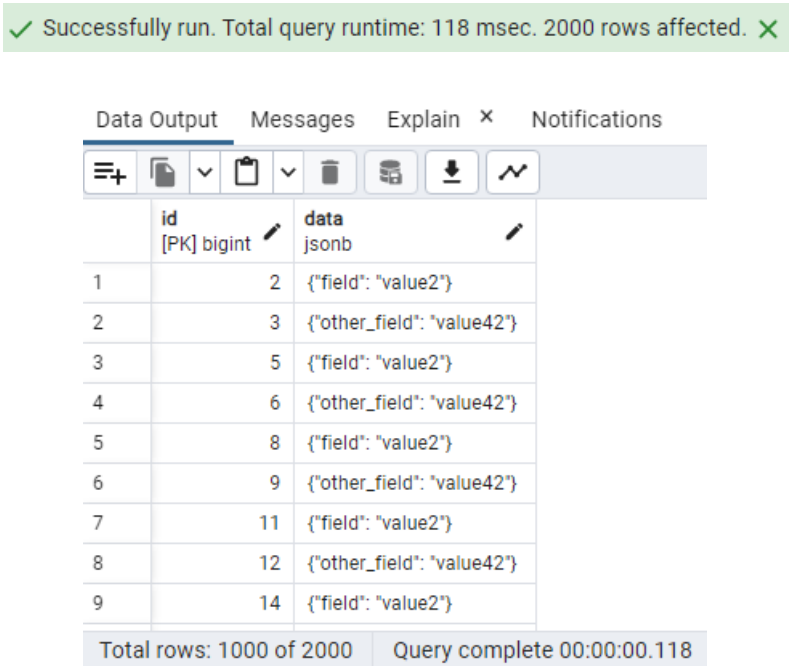
```
SELECT * FROM gintable WHERE data ? 'field';
```




```
CREATE INDEX ginexample2 ON gintable USING btree ((data ->> 'field'));
```




```
SELECT * FROM gintable WHERE data->>'field' = 'value2' OR data @>
'{ "other_field": "value42" }';
```



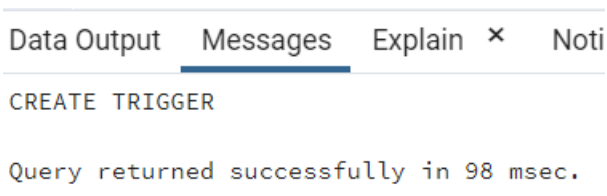
B-Tree

Цей індекс пришвидшив роботу записів, тому що таблиця має велику кількість даних, а індекс добре працює з великими обсягами даних. Також він допомагає в оптимізації при сортуванні. Мінусом є більше зайнятого місця у порівнянні з іншими індексами. Також ефективність зменшується при великій кількості записів та збільшенні кількості ключів.

GIN

Цей індекс зазвичай працює з типом даних jsonb, тому було створено додаткову таблицю. GIN ефективний для роботи з jsonb та масивами. Недоліками є ймовірність, що буде зайнято багато місця, вставка та оновлення може вимагати багато ресурсів. Також до них можна віднести те, що цей індекс працює лише зі складними структурами даних.

```
CREATE OR REPLACE FUNCTION hcw_insert()
      RETURNS trigger AS
$$
BEGIN

      INSERT INTO vaccination_schedule(schedule_id, worker_id, vaccination_id,
date)
      VALUES ((SELECT MAX(schedule_id) FROM vaccination_schedule)+1,
                  (SELECT MAX(worker_id) FROM healthcare_workers), 1, '2023-01-
01'::date);

RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER hcw_insert_trigger
      AFTER INSERT
      ON healthcare_workers
      EXECUTE PROCEDURE hcw_insert();
```



```
INSERT INTO healthcare_workers(worker_id, name, specialization,
medical_license_number)
VALUES ((SELECT MAX(worker_id) FROM healthcare_workers)+1, 'fdkjf',
'dsjndksjcn', 3894333298);
```

Data Output | Messages | Explain | Notifications

```
INSERT 0 1

Query returned successfully in 38 msec.
```

Data Output | Messages | Notifications

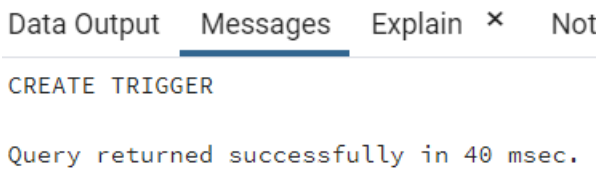
	worker_id [PK] integer	name character varying (50)	specialization character varying (30)	medical_license_number bigint
16	16	dsbsdjbc	Ctjaxmqioc	38213971184000
17	17	dsbsdjbc	Jcwniahvwq	97619922303042
18	18	dsbsdjbc	Vdypbkvykm	75920026139020
19	19	fkjvkddv	sdkjcnksd	1237
20	20	abcd	dsjndksjcn	38923298
21	21	fdkjf	dsjndksjcn	3894333298

Data Output | Messages | Notifications

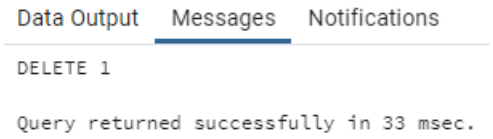
	schedule_id [PK] integer	worker_id integer	vaccination_id integer	date date
1	10002	21	1	2023-01-01
2	10001	20	24	2023-01-12
3	10000	7	665	2023-04-02
4	9999	6	56	2023-05-04
5	9998	17	926	2023-10-19
6	9997	2	593	2023-10-21

```
CREATE OR REPLACE FUNCTION vs_delete()
      RETURNS trigger AS
$$
BEGIN

      UPDATE healthcare_workers
      SET name = 'qwerty'
      WHERE worker_id = 21;
```

```
RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE TRIGGER vs_delete_trigger
      AFTER INSERT
      ON vaccination_schedule
      EXECUTE PROCEDURE vs_delete();
```




```
DELETE FROM vaccination_schedule WHERE schedule_id = 10001;
```

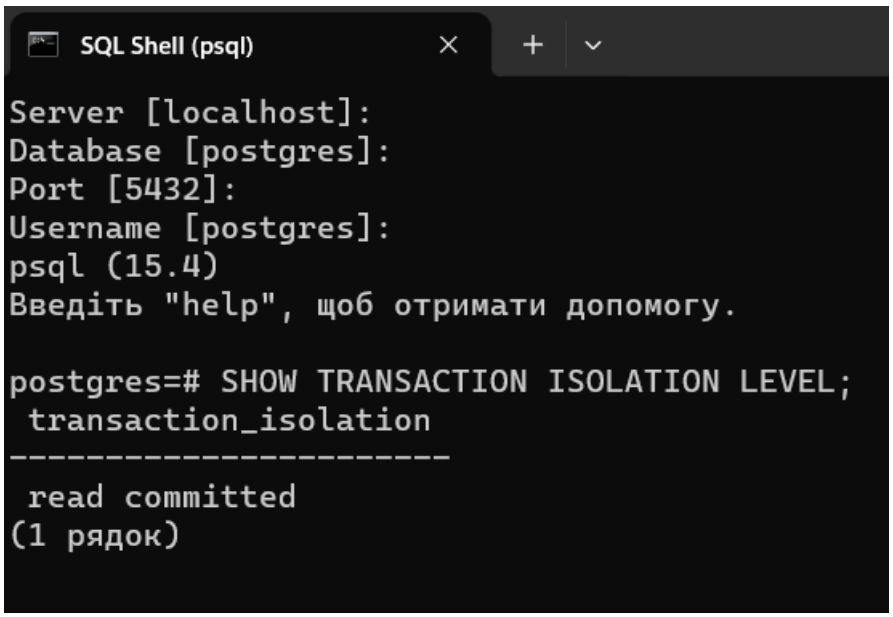


Data Output Messages Notifications

	schedule_id [PK] integer	worker_id integer	vaccination_id integer	date date
1	10000	7	665	2023-04-02
2	9999	6	56	2023-05-04
3	9998	17	926	2023-10-19
4	9997	2	593	2023-10-21
5	9996	3	782	2023-05-26
6	9995	2	801	2023-08-28

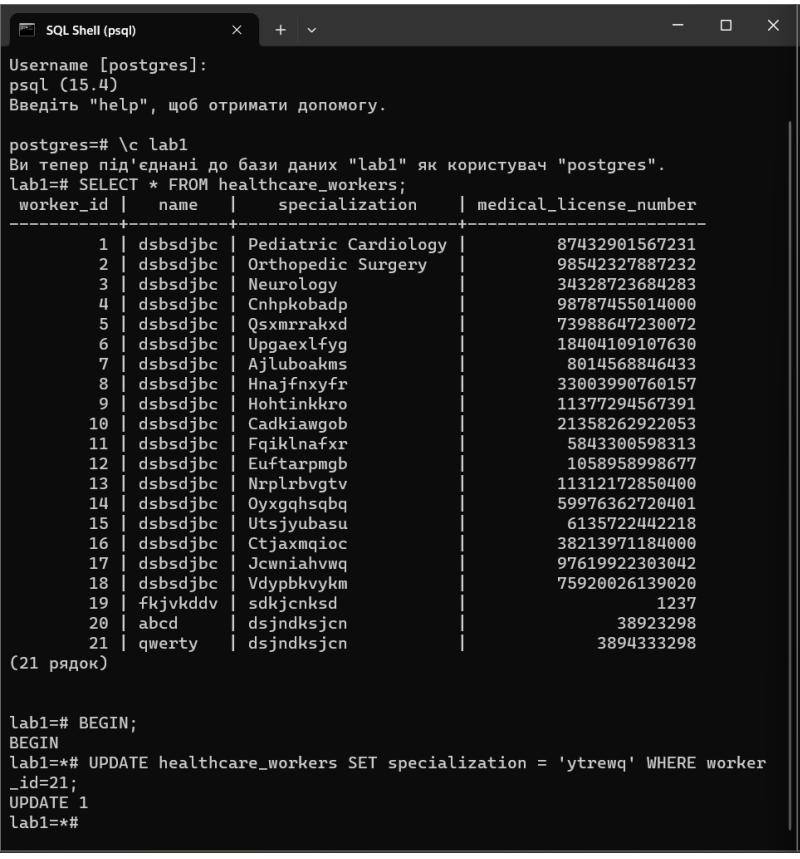
Data Output Messages Notifications

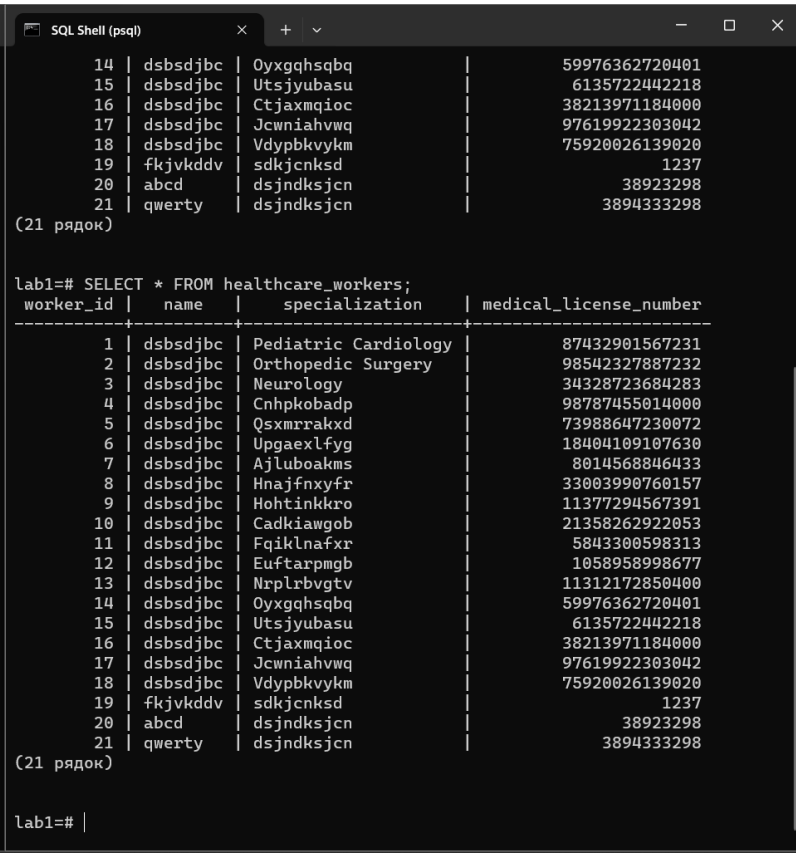
	worker_id [PK] integer	name character varying (50)	specialization character varying (30)	medical_license_number bigint
15	15	dsbsdjbc	Otsjyubasd	61357224422 18
16	16	dsbsdjbc	Ctjaxmqioc	38213971184000
17	17	dsbsdjbc	Jcwniahvwq	97619922303042
18	18	dsbsdjbc	Vdypbkvykm	75920026139020
19	19	fkjvkddv	sdkjcnksd	1237
20	20	abcd	dsjndksjcn	38923298
21	21	qwerty	dsjndksjcn	3894333298

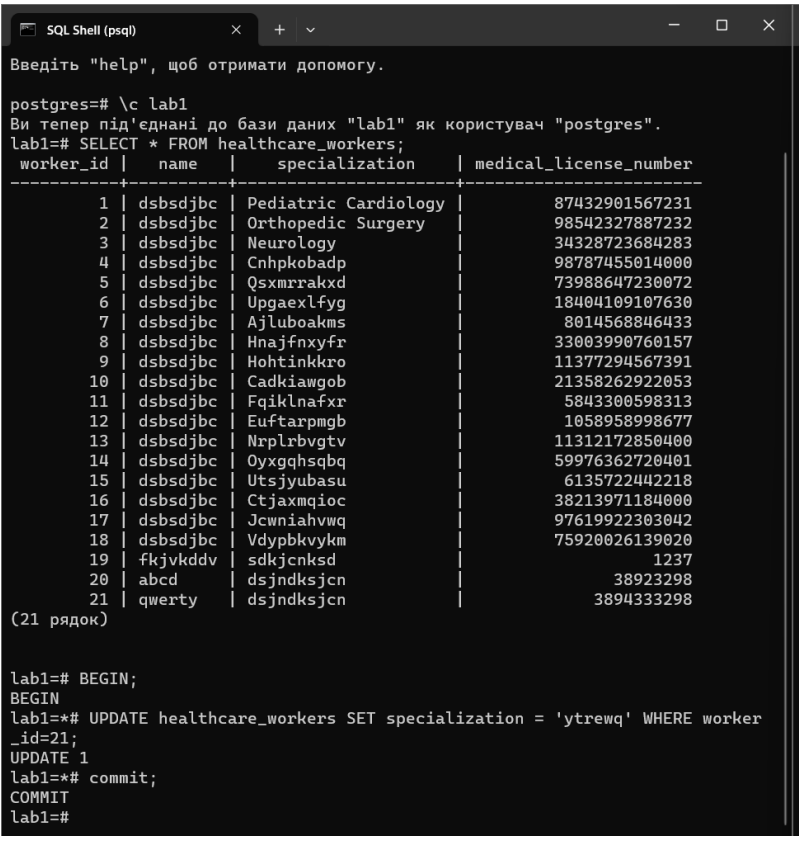


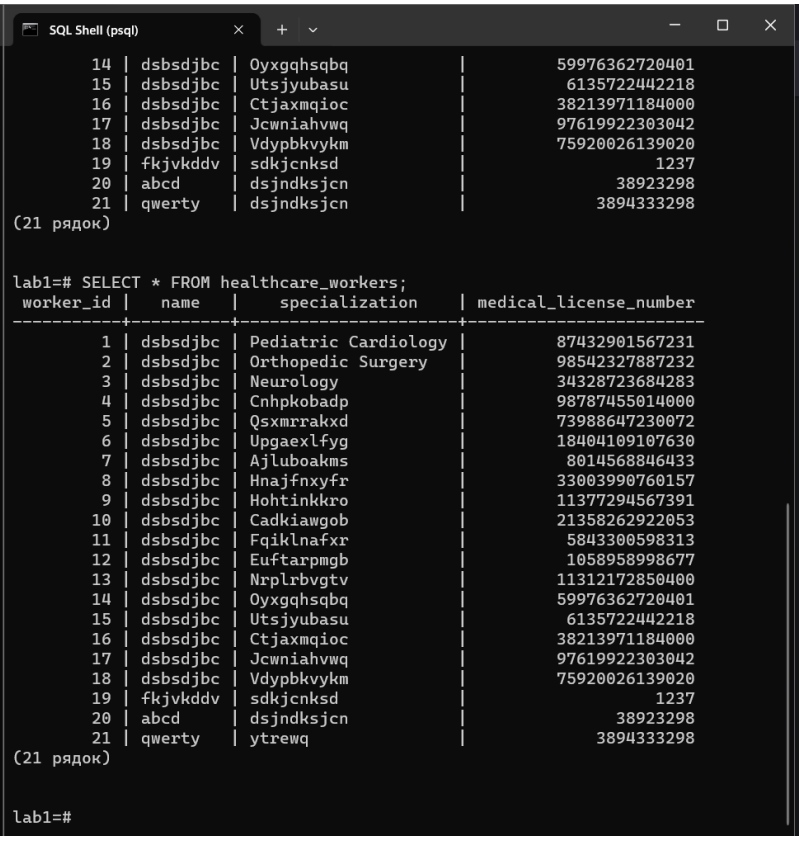
1. Dirty read

Цей феномен відбувається при доступі до даних під час редагування даних в режимі READ COMMITED. У цьому прикладі при виконанні SELECT під час операції ми отримуємо дані до початку виконання. При перевірці після коміту отримуємо змінені дані.



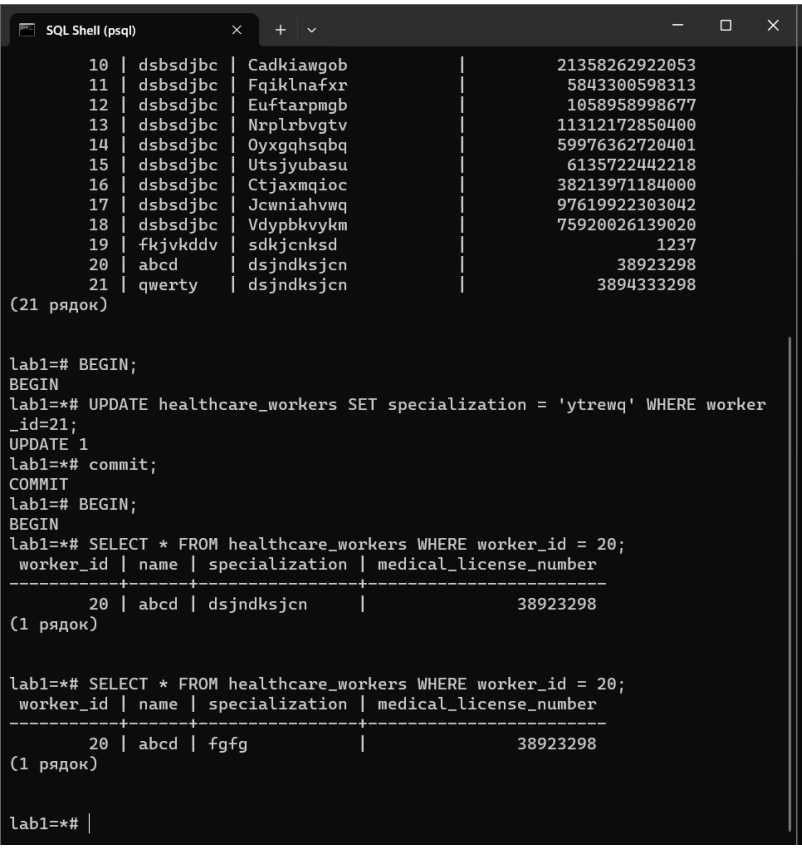


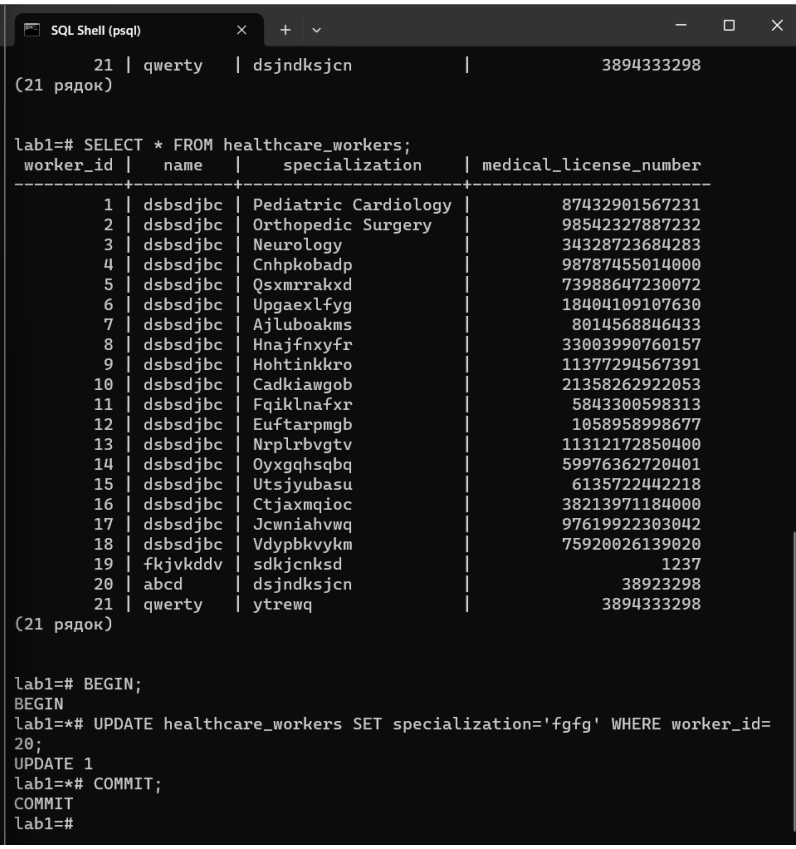

SQL Shell (psql)
Введіть "help", щоб отримати допомогу.
postgres=# \c lab1
Ви тепер під'єднані до бази даних "lab1" як користувач "postgres".
lab1=# SELECT * FROM healthcare_workers;
worker_id | name | specialization | medical_license_number
1 | dsbsdjbc | Pediatric Cardiology | 87432901567231
2 | dsbsdjbc | Orthopedic Surgery | 98542327887232
3 | dsbsdjbc | Neurology | 34328723684283
4 | dsbsdjbc | Cnhpkobadp | 98787455014000
5 | dsbsdjbc | Qsxmrrakxd | 73988647230072
6 | dsbsdjbc | Upgaexlfyg | 18404109107630
7 | dsbsdjbc | Ajluboakms | 8014568846433
8 | dsbsdjbc | Hnajfnxyfr | 33003990760157
9 | dsbsdjbc | Hohtinkkro | 11377294567391
10 | dsbsdjbc | Cadkiawgob | 21358262922053
11 | dsbsdjbc | Fqiklnafxr | 5843300598313
12 | dsbsdjbc | Euftarpmgb | 1058958998677
13 | dsbsdjbc | Nrplrbvgtv | 11312172850400
14 | dsbsdjbc | Oyxgqhsqbq | 59976362720401
15 | dsbsdjbc | Utsjyubasu | 6135722442218
16 | dsbsdjbc | Ctjaxmqioc | 38213971184000
17 | dsbsdjbc | Jcwniahvwq | 97619922303042
18 | dsbsdjbc | Vdypbkvykm | 75920026139020
19 | fkjvkddv | sdkjcnksd | 1237
20 | abcd | dsjndksjcn | 38923298
21 | qwerty | dsjndksjcn | 3894333298
(21 рядок)
lab1=# BEGIN;
BEGIN
lab1=*# UPDATE healthcare_workers SET specialization = 'ytrewq' WHERE worker_id=21;
UPDATE 1
lab1=*# commit;
COMMIT
lab1=#

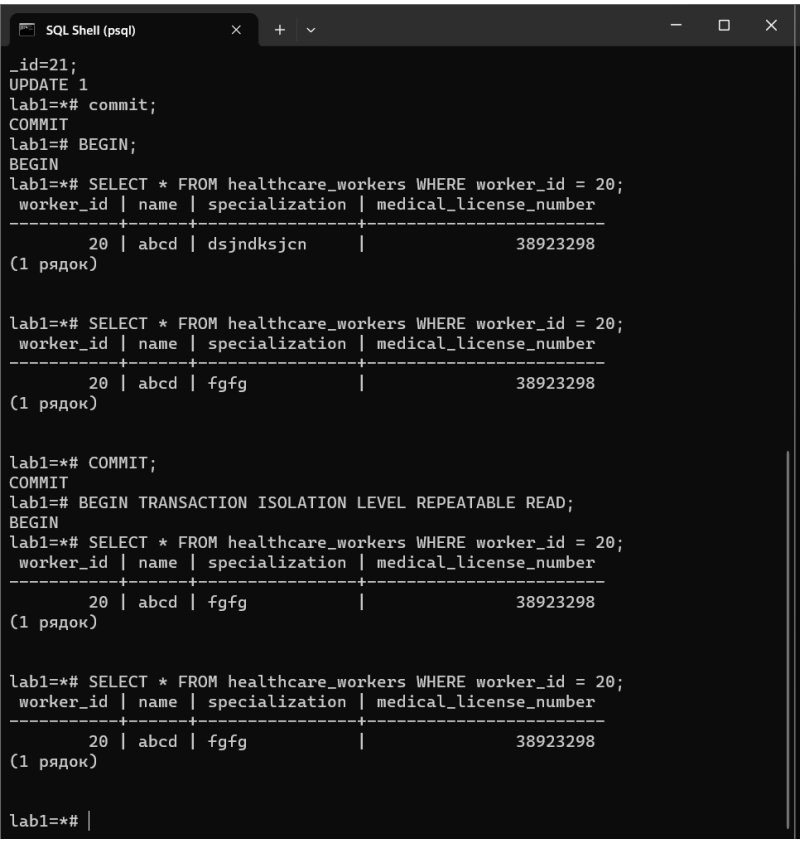

SQL Shell (psql)
14 | dsbsdjbc | Oyxgqhsqbq | 59976362720401
15 | dsbsdjbc | Utsjyubasu | 6135722442218
16 | dsbsdjbc | Ctjaxmqioc | 38213971184000
17 | dsbsdjbc | Jcwniahvwq | 97619922303042
18 | dsbsdjbc | Vdypbkvykm | 75920026139020
19 | fkjvkddv | sdkjcnksd | 1237
20 | abcd | dsjndksjcn | 38923298
21 | qwerty | dsjndksjcn | 3894333298
(21 рядок)
lab1=# SELECT * FROM healthcare_workers;
worker_id | name | specialization | medical_license_number
1 | dsbsdjbc | Pediatric Cardiology | 87432901567231
2 | dsbsdjbc | Orthopedic Surgery | 98542327887232
3 | dsbsdjbc | Neurology | 34328723684283
4 | dsbsdjbc | Cnhpkobadp | 98787455014000
5 | dsbsdjbc | Qsxmrrakxd | 73988647230072
6 | dsbsdjbc | Upgaexlfyg | 18404109107630
7 | dsbsdjbc | Ajluboakms | 8014568846433
8 | dsbsdjbc | Hnajfnxyfr | 33003990760157
9 | dsbsdjbc | Hohtinkkro | 11377294567391
10 | dsbsdjbc | Cadkiawgob | 21358262922053
11 | dsbsdjbc | Fqiklnafxr | 5843300598313
12 | dsbsdjbc | Euftarpmgb | 1058958998677
13 | dsbsdjbc | Nrplrbvgtv | 11312172850400
14 | dsbsdjbc | Oyxgqhsqbq | 59976362720401
15 | dsbsdjbc | Utsjyubasu | 6135722442218
16 | dsbsdjbc | Ctjaxmqioc | 38213971184000
17 | dsbsdjbc | Jcwniahvwq | 97619922303042
18 | dsbsdjbc | Vdypbkvykm | 75920026139020
19 | fkjvkddv | sdkjcnksd | 1237
20 | abcd | dsjndksjcn | 38923298
21 | qwerty | ytrewq | 3894333298
(21 рядок)
lab1=#

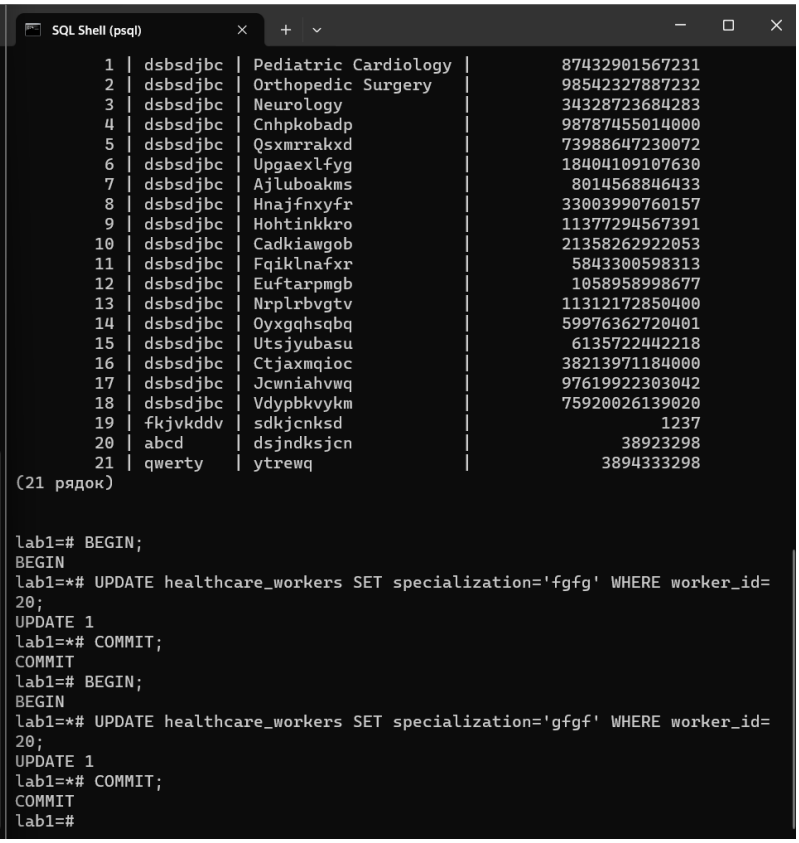
2. Nonrepeatable read

Цей феномен відбувається при доступі до даних в одній операції в режимі READ COMMITED, коли відбувається зміна даних в іншій операції. Рішенням є режим ізоляції REPEATABLE READ в операції читання. У цьому прикладі відбулось отримання даних до зміни їх у іншій операції та після. Отримані дані були змінені. При режимі REPEATABLE READ дані залишаються не зміненими до кінця операції.



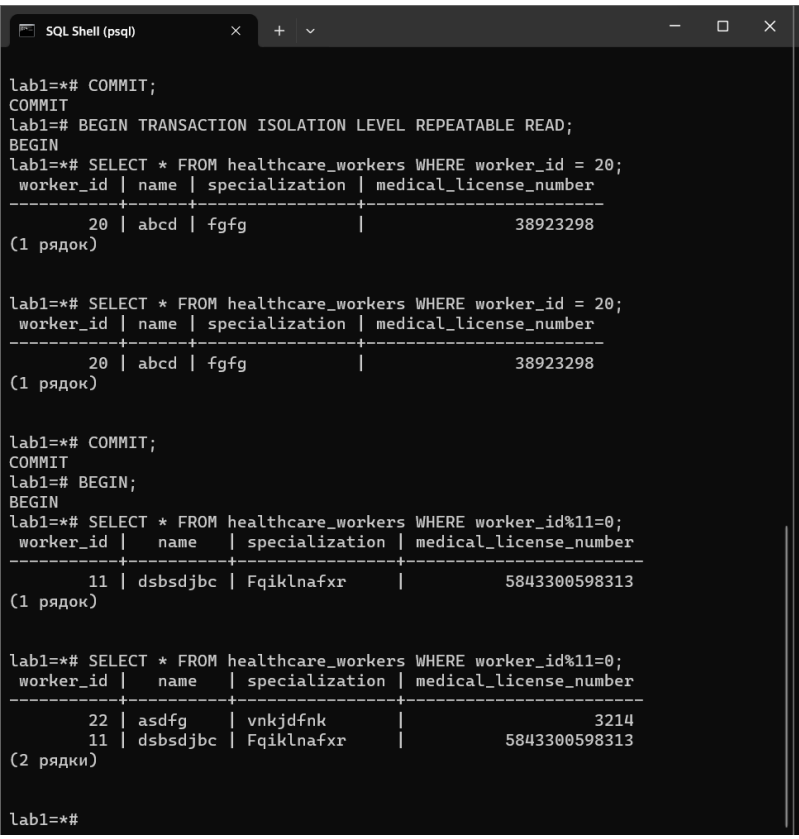


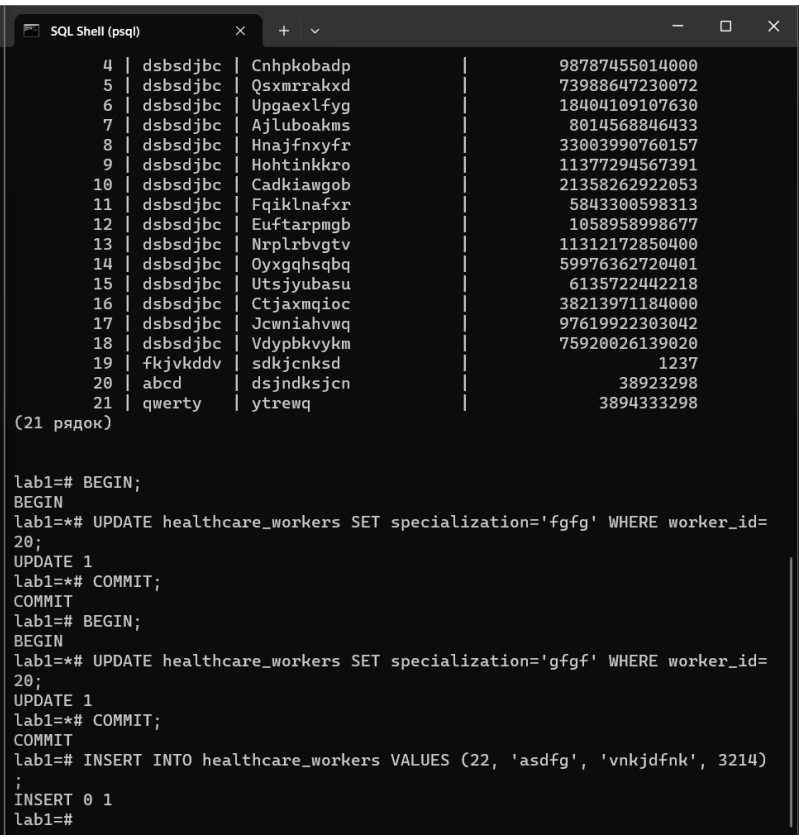

SQL Shell (psql)
_id=21;
UPDATE 1
lab1=*# commit;
COMMIT
lab1=# BEGIN;
BEGIN
lab1=*# SELECT * FROM healthcare_workers WHERE worker_id = 20;
worker_id | name | specialization | medical_license_number
20 | abcd | dsjndksjcn | 38923298
(1 рядок)
lab1=*# SELECT * FROM healthcare_workers WHERE worker_id = 20;
worker_id | name | specialization | medical_license_number
20 | abcd | fgfg | 38923298
(1 рядок)
lab1=*# COMMIT;
COMMIT
lab1=# BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN
lab1=*# SELECT * FROM healthcare_workers WHERE worker_id = 20;
worker_id | name | specialization | medical_license_number
20 | abcd | fgfg | 38923298
(1 рядок)
lab1=*# SELECT * FROM healthcare_workers WHERE worker_id = 20;
worker_id | name | specialization | medical_license_number
20 | abcd | fgfg | 38923298
(1 рядок)
lab1=*#


SQL Shell (psql)
1 | dsbsdjbc | Pediatric Cardiology | 87432901567231
2 | dsbsdjbc | Orthopedic Surgery | 98542327887232
3 | dsbsdjbc | Neurology | 34328723684283
4 | dsbsdjbc | Cnhpkobadp | 98787455014000
5 | dsbsdjbc | Qsxmrrakxd | 73988647230072
6 | dsbsdjbc | Upgaexlfyg | 18404109107630
7 | dsbsdjbc | Ajluboakms | 8014568846433
8 | dsbsdjbc | Hnajfnxyfr | 33003990760157
9 | dsbsdjbc | Hohtinkkro | 11377294567391
10 | dsbsdjbc | Cadkiawgob | 21358262922053
11 | dsbsdjbc | Fqiklnafxr | 5843300598313
12 | dsbsdjbc | Euftarpmgb | 1058958998677
13 | dsbsdjbc | Nrplrbvgtv | 11312172850400
14 | dsbsdjbc | Oyxgqhsqbq | 59976362720401
15 | dsbsdjbc | Utsjyubasu | 6135722442218
16 | dsbsdjbc | Ctjaxmqioc | 38213971184000
17 | dsbsdjbc | Jcwniahvwq | 97619922303042
18 | dsbsdjbc | Vdypbkvykm | 75920026139020
19 | fkjvkddv | sdkjcnksd | 1237
20 | abcd | dsjndksjcn | 38923298
21 | qwerty | ytrewq | 3894333298
(21 рядок)
lab1=# BEGIN;
BEGIN
lab1=*# UPDATE healthcare_workers SET specialization='fgfg' WHERE worker_id=
20;
UPDATE 1
lab1=*# COMMIT;
COMMIT
lab1=# BEGIN;
BEGIN
lab1=*# UPDATE healthcare_workers SET specialization='gfgf' WHERE worker_id=
20;
UPDATE 1
lab1=*# COMMIT;
COMMIT
lab1=#

3. Phantom reads

Цей феномен відбувається при читанні даних за критерієм після додавання нових рядків, які виконують ці критерії. У прикладі показано читання до та після додавання. При появі нового рядка результати змінились. При використанні режиму REPEATABLE READ результат залишиться незмінним до кінця операції.





SQL Shell (psql)

```
BEGIN
lab1=*# SELECT * FROM healthcare_workers WHERE worker_id%11=0;
 worker_id |   name   | specialization | medical_license_number
-----------+----------+----------------+------------------------
        11 | dsbsdjbc | Fqiklnafxr     |          5843300598313
(1 рядок)


lab1=*# SELECT * FROM healthcare_workers WHERE worker_id%11=0;
 worker_id |   name   | specialization | medical_license_number
-----------+----------+----------------+------------------------
        22 | asdfg    | vnkjdfnk       |                   3214
        11 | dsbsdjbc | Fqiklnafxr     |          5843300598313
(2 рядки)


lab1=*# COMMIT;
COMMIT
lab1=# BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN
lab1=*# SELECT * FROM healthcare_workers WHERE worker_id%11=0;
 worker_id |   name   | specialization | medical_license_number
-----------+----------+----------------+------------------------
        22 | asdfg    | vnkjdfnk       |                   3214
        11 | dsbsdjbc | Fqiklnafxr     |          5843300598313
(2 рядки)


lab1=*# SELECT * FROM healthcare_workers WHERE worker_id%11=0;
 worker_id |   name   | specialization | medical_license_number
-----------+----------+----------------+------------------------
        22 | asdfg    | vnkjdfnk       |                   3214
        11 | dsbsdjbc | Fqiklnafxr     |          5843300598313
(2 рядки)


lab1=*# COMMIT;
COMMIT
lab1=#
```

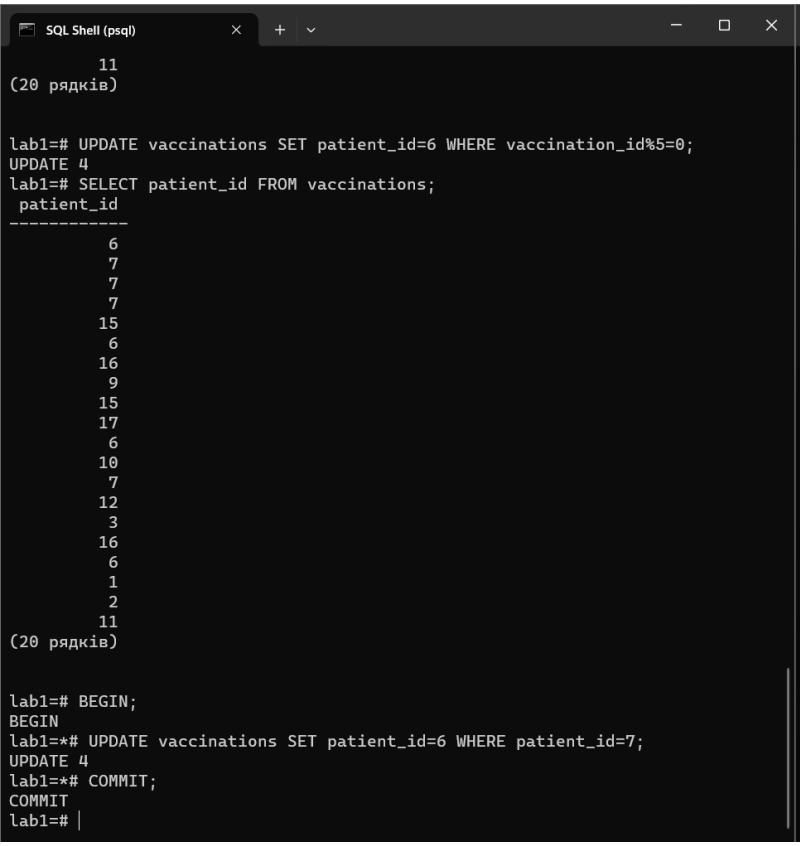
SQL Shell (psql)

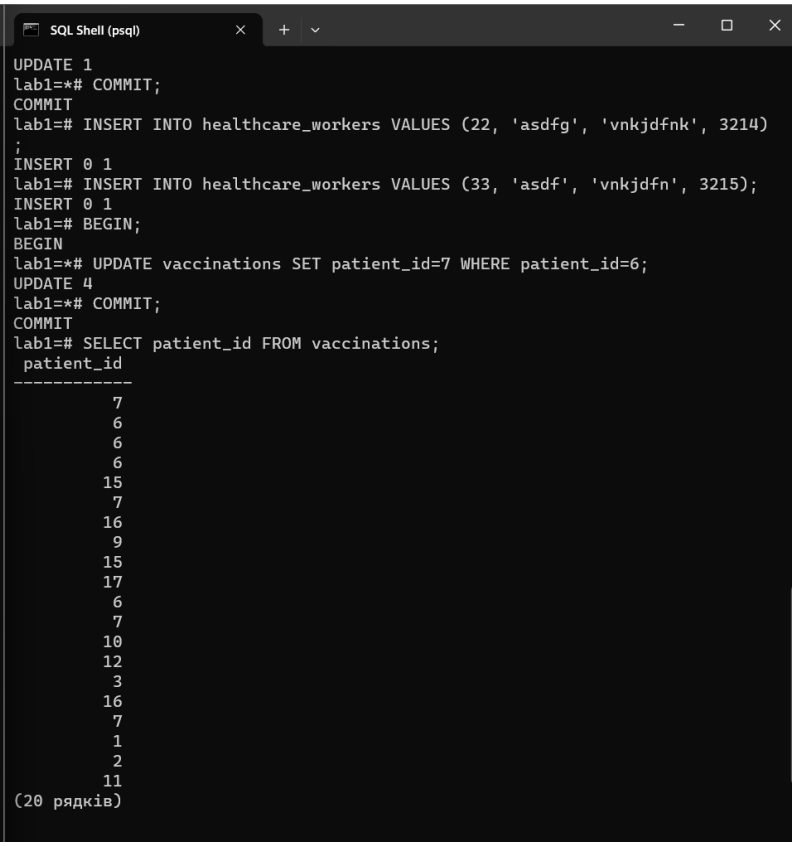
```
         6 | dsbsdjbc | Upgaexlfyg     |         184041091076 30
         7 | dsbsdjbc | Ajluboakms     |          8014568846433
         8 | dsbsdjbc | Hnajfnxyfr     |         33003990760157
         9 | dsbsdjbc | Hohtinkkro     |         11377294567391
        10 | dsbsdjbc | Cadkiawgob     |         21358262922053
        11 | dsbsdjbc | Fqiklnafxr     |          5843300598313
        12 | dsbsdjbc | Euftarpmgb     |          1058958998677
        13 | dsbsdjbc | Nrplrbvgtv     |         11312172850400
        14 | dsbsdjbc | Oyxgqhsqbq     |         59976362720401
        15 | dsbsdjbc | Utsjyubasu     |          6135722442218
        16 | dsbsdjbc | Ctjaxmqioc     |         38213971184000
        17 | dsbsdjbc | Jcwniahvwq     |         97619922303042
        18 | dsbsdjbc | Vdypbkvykm     |         75920026139020
        19 | fkjvkddv | sdkjcnksd      |                   1237
        20 | abcd     | dsjndksjcn     |               38923298
        21 | qwerty   | ytrewq         |             3894333298
(21 рядок)


lab1=# BEGIN;
BEGIN
lab1=*# UPDATE healthcare_workers SET specialization='fgfg' WHERE worker_id=
20;
UPDATE 1
lab1=*# COMMIT;
COMMIT
lab1=# BEGIN;
BEGIN
lab1=*# UPDATE healthcare_workers SET specialization='gfgf' WHERE worker_id=
20;
UPDATE 1
lab1=*# COMMIT;
COMMIT
lab1=# INSERT INTO healthcare_workers VALUES (22, 'asdfg', 'vnkjdfnk', 3214)
;
INSERT 0 1
lab1=# INSERT INTO healthcare_workers VALUES (33, 'asdf', 'vnkjdfn', 3215);
INSERT 0 1
lab1=#
```

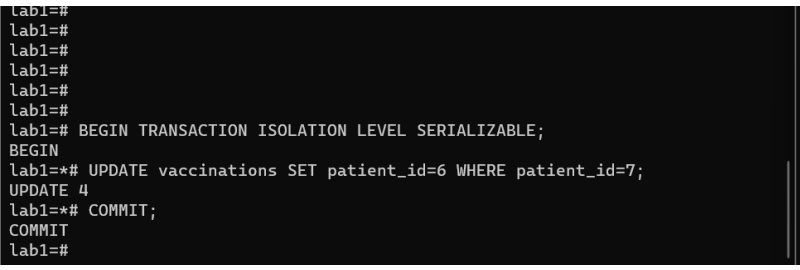
4. Serialization Anomaly

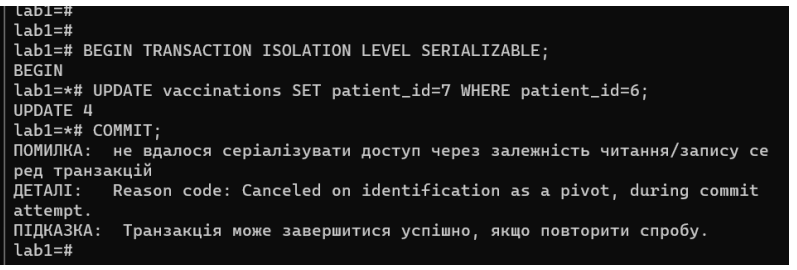
Цей феномен відбувається при одночасній зміні даних, які залежать один від одного, у різних операціях. У прикладі були змінені значення стовпця, на які були змінені інші стовпці, на значення, від яких залежать інші операції. Значення змінились на протилежні. При режимі READ COMMITED значення зміняться на одне з можливих. При режимі REPEATABLE READ значення поміняються місцями. При режимі SERIALIZABLE з’явиться попередження про залежність даних і операція відкотиться.

```
lab1=# SELECT patient_id FROM vaccinations;
 patient_id
------------
          6
          7
          7
          7
         15
          6
         16
          9
         15
         17
          6
         10
          7
         12
          3
         16
          6
          1
          2
         11
(20 рядків)
```


SQL Shell (psql)
11
(20 рядків)
lab1=# UPDATE vaccinations SET patient_id=6 WHERE vaccination_id%5=0;
UPDATE 4
lab1=# SELECT patient_id FROM vaccinations;
patient_id
6
7
7
7
15
6
16
9
15
17
6
10
7
12
3
16
6
1
2
11
(20 рядків)
lab1=# BEGIN;
BEGIN
lab1=*# UPDATE vaccinations SET patient_id=6 WHERE patient_id=7;
UPDATE 4
lab1=*# COMMIT;
COMMIT
lab1=#


SQL Shell (psql)
UPDATE 1
lab1=*# COMMIT;
COMMIT
lab1=# INSERT INTO healthcare_workers VALUES (22, 'asdfg', 'vnkjdfnk', 3214)
;
INSERT 0 1
lab1=# INSERT INTO healthcare_workers VALUES (33, 'asdf', 'vnkjdfn', 3215);
INSERT 0 1
lab1=# BEGIN;
BEGIN
lab1=*# UPDATE vaccinations SET patient_id=7 WHERE patient_id=6;
UPDATE 4
lab1=*# COMMIT;
COMMIT
lab1=# SELECT patient_id FROM vaccinations;
patient_id
7
6
6
6
15
7
16
9
15
17
6
7
10
12
3
16
7
1
2
11
(20 рядків)


lab1=#
lab1=#
lab1=#
lab1=#
lab1=#
lab1=#
lab1=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
lab1=*# UPDATE vaccinations SET patient_id=6 WHERE patient_id=7;
UPDATE 4
lab1=*# COMMIT;
COMMIT
lab1=#


lab1=#
lab1=#
lab1=# BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN
lab1=*# UPDATE vaccinations SET patient_id=7 WHERE patient_id=6;
UPDATE 4
lab1=*# COMMIT;
ПОМИЛКА: не вдалося серіалізувати доступ через залежність читання/запису се
ред транзакцій
ДЕТАЛІ: Reason code: Canceled on identification as a pivot, during commit
attempt.
ПІДКАЗКА: Транзакція може завершитися успішно, якщо повторити спробу.
lab1=#