



**Національний технічний університет України
“Київський політехнічний інститут”**

**Факультет прикладної математики
Кафедра системного програмування і спеціалізованих
комп’ютерних систем**

**Лабораторна робота №3
з дисципліни “*Бази даних і засоби управління*”**

Тема: “Засоби оптимізації роботи СУБД PostgreSQL”

Виконав:
студент III курсу
групи KB-01
Таранич Артем
Перевірів: Павловський В. І.

Київ – 2022

Завдання роботи полягає у наступному:

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проєкції (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.

Варіант 19

19	<i>BTree, BRIN</i>	<i>before insert, delete</i>
----	--------------------	------------------------------

Логічна модель предметної області «Аеропорт»

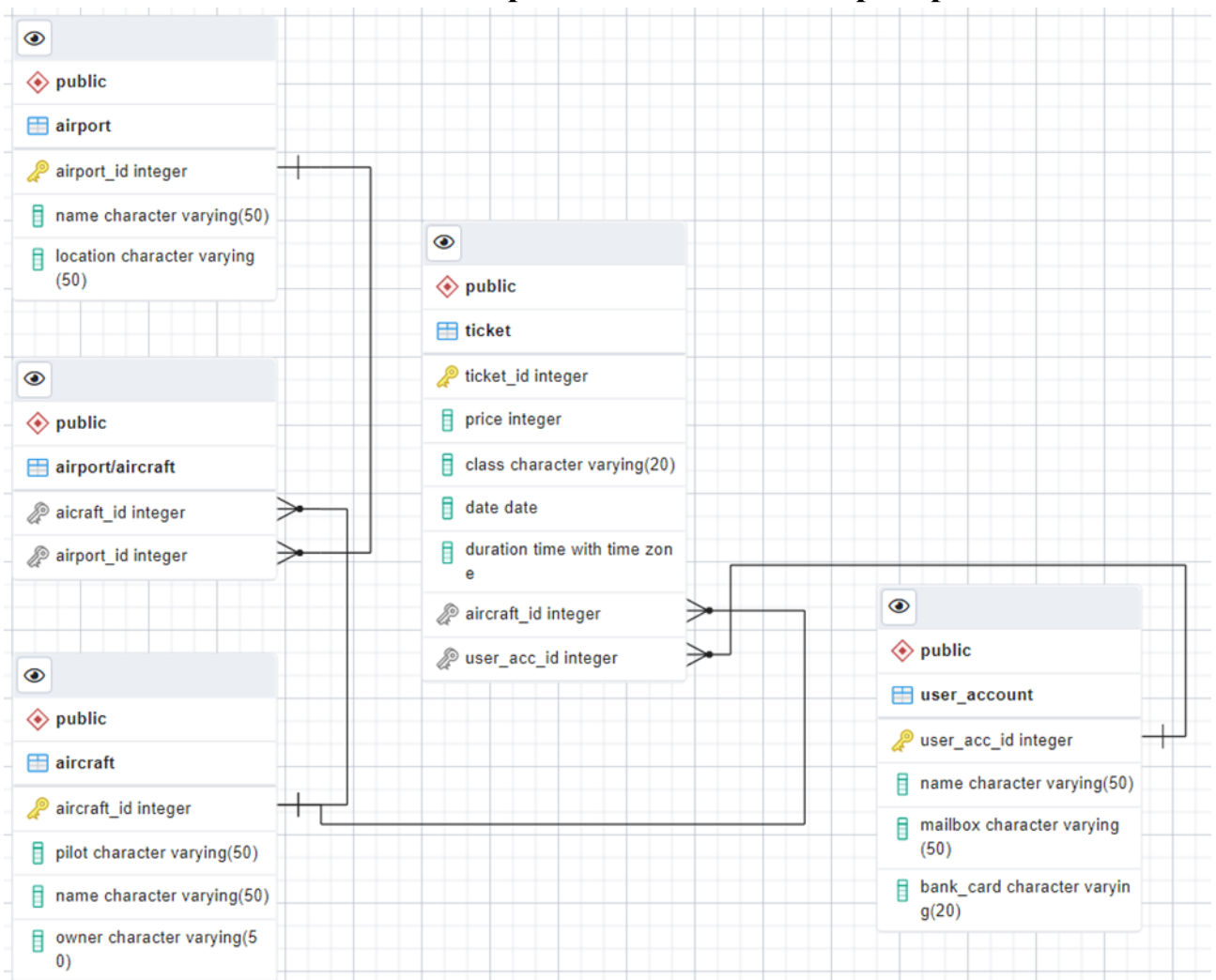


Рисунок 1. Схема бази даних, побудовано у pgAdmin 4.

Завдання 1

Класи ORM

```
class Airport(Base):
    __tablename__ = 'airport'
    columns = ['airport_id', 'name', 'location']
    airport_id = Column(Integer, primary_key=True)
    name = Column(String)
    location = Column(String)

    def __init__(self, name, location):
        self.name = name
        self.location = location
        super(Airport, self).__init__()
    def id(self):
        return self.airport_id

class Aircraft(Base):
    __tablename__ = 'aircraft'
    columns = ['aircraft_id', 'pilot', 'name', 'owner']
    addition_keys = ['airport_id']
    aircraft_id = Column(Integer, primary_key=True)
    pilot = Column(String)
    name = Column(String)
    owner = Column(String)

    def __init__(self, pilot, name, owner):
        self.pilot = pilot
        self.name = name
        self.owner = owner
        super(Aircraft, self).__init__()
    def id(self):
        return self.aircraft_id

class AirportAircraft(Base):
    __tablename__ = 'airport_aircraft'
    columns = ['aircraft_id', 'airport_id']
    aircraft_id = Column(Integer, ForeignKey('aircraft.aircraft_id'),
primary_key=True)
    airport_id = Column(Integer, ForeignKey('airport.airport_id'),
primary_key=True)

    def __init__(self, aircraft_id, airport_id):
        self.aircraft_id = aircraft_id
        self.airport_id = airport_id
        super(AirportAircraft, self).__init__()
    def id(self):
        return self.aircraft_id

class Ticket(Base):
    __tablename__ = 'ticket'
    columns =
['ticket_id', 'price', 'seat_type', 'date', 'duration', 'aircraft_id', 'user_acc_id']
    ticket_id = Column(Integer, primary_key=True)
    price = Column(Integer)
```

```

seat_type    = Column(String)
date         = Column(String)
duration     = Column(Integer)
aircraft_id  = Column(Integer, ForeignKey('aircraft.aircraft_id'))
user_acc_id  = Column(Integer, ForeignKey('user_account.user_acc_id'))

def __init__(self, price, seat_type, date, duration, aircraft_id,
user_acc_id):
    self.price = price
    self.seat_type = seat_type
    self.date = date
    self.duration = duration
    self.aircraft_id = aircraft_id
    self.user_acc_id = user_acc_id
    super(Ticket, self).__init__()
def id(self):
    return self.ticket_id

class UserAccount(Base):
    __tablename__ = 'user_account'
    columns = ['user_acc_id', 'name', 'mailbox', 'bank_card']
    user_acc_id = Column(Integer, primary_key=True)
    name        = Column(String)
    mailbox     = Column(String)
    bank_card   = Column(String)

def __init__(self, name, mailbox, bank_card):
    self.name = name
    self.mailbox = mailbox
    self.bank_card = bank_card
    super(UserAccount, self).__init__()
def id(self):
    return self.user_acc_id

```

Результати та виконання операцій

Виведення таблиці 'Aircraft' в консоль

	aircraft_id	pilot	aircraft	name	owner
1.	1	Andrew Russel	Boeing 737-800	Bairbus	
2.	2	Ura	Airbas	Claster Master	
3.	4	Andrew Russel	Boeing 737-800	Airbuss	
4.	5	Ira	Boeing	Airbuss	
5.	6	Iu	Pw	Qs	
6.	7	Dx	Cl	Gq	
7.	8	Ky	Jm	Fm	
8.	9	Su	Sv	Kw	
9.	10	Fo	VfD	Dv	
10.	11	Yt	It	Xi	
11.	12	Mk	Ps	Ne	
12.	13	Ef	Dl	Gq	
13.	14	gwydsqsvdk	siznlsvbvl	sodwuhnbx1	
14.	15	xfmdhvjdyn	fsljrcztjn	nfozlfffvb	
15.	16	lbayfvpzrc	djxeqgrgmz	fjjbnlulks	

Додавання нового рядку до таблиці “Aircraft”

```
Enter data for insert in table: aircraft
Enter pilot: Ura
Enter name: Bombardier CRJ-200
Enter owner: Bairbus
Enter airport_id: 15
```

	aircraft_id	pilot	aircraft	name	owner
1.	1	Andrew Russel		Boeing 737-800	Bairbus
2.	2	Ura		Airbas	Claster Master
3.	4	Andrew Russel		Boeing 737-800	Airbuss
4.	5	Ira		Boeing	Airbuss
5.	6	Iu		Pw	Qs
6.	7	Dx		Cl	Gq
7.	8	Ky		Jm	Fm
8.	9	Su		Sv	Kw
9.	10	Fo		VfD	Dv
10.	11	Yt		It	Xi
11.	12	Mk		Ps	Ne
12.	13	Ef		Dl	Gq
13.	14	gwydsqsvdk		siznlsvbvl	sodwuhnbxl
14.	15	xfmdhvjdyn		fsljrcztjn	nfozlffffvb
15.	16	lbayfvpzrc		djxeqgrgmz	fjjbnlulks
16.	17	Ura	Bombardier CRJ-200		Bairbus

В результаті дані також оновилися в зв'язаній таблиці “Airport/Aircraft”

	aircraft_id	airport_id
1.	1	15
2.	4	17
3.	5	30
4.	5	13
5.	6	12
6.	6	14
7.	7	16
8.	8	14
9.	9	17
10.	9	31
11.	9	12
12.	10	12
13.	11	17
14.	14	25
15.	15	33
16.	16	12
17.	17	15

Оновлення рядку у таблиці “Users Account”

До

	user_acc_id	name	user_account	mailbox	bank_card
1.	2	admin		admin@mainbox.com	1234567812345678
2.	3	Ura		ura@gmail.com	1025458412535412
3.	4	Hm		Db	202
4.	5	Jg		Fr	928
5.	6	Hv		Yd	675
6.	7	Ffl		fk	2345
7.	8	Ws		Xg	119
8.	9	De		Fc	158
9.	10	hgztekugk		ijigdfimeq	1436
10.	11	hhjyfxqypl		ifymvjtfnz	1680

Після

```
Enter data for update table: user_account
Enter user_acc_id: 10
Enter name: Andrey
Enter mailbox: mybox@mainbox.com
Enter bank_card: 1284369475569871
```

	user_acc_id	name	user_account	mailbox	bank_card
1.	2	admin		admin@mainbox.com	1234567812345678
2.	3	Ura		ura@gmail.com	1025458412535412
3.	4	Hm		Db	202
4.	5	Jg		Fr	928
5.	6	Hv		Yd	675
6.	7	Ffl		fk	2345
7.	8	Ws		Xg	119
8.	9	De		Fc	158
9.	10	Andrey		mybox@mainbox.com	1284369475569871
10.	11	hhjyfxqypl		ifymvjtfnz	1680

Видалення рядку з таблиці “Ticket”

До

	ticket_id	price	seat_type	date	duration	aircraft_id	user_acc_id
1.	1	4500	B	2022-10-10 17:17:40	05:45:00	4	2
2.	2	285	C	2022-11-18 00:59:00	05:00:00	5	2
3.	3	668	C	2022-12-06 19:01:00	02:00:00	5	4
4.	4	884	E	2022-12-07 14:50:00	21:00:00	7	2
5.	5	234	A	2022-11-25 21:40:00	13:00:00	2	6
6.	6	230	D	2022-11-14 09:55:00	13:00:00	11	3
7.	7	238	E	2018-01-11 04:24:00	03:41:20	13	5
8.	8	894	E	1979-09-10 16:37:36	19:31:11	15	6

Після

Enter data for delete from table: ticket
Enter ticket_id for delete: 6

	ticket_id	price	seat_type	date	duration	aircraft_id	user_acc_id
1.	1	4500	B	2022-10-10 17:17:40	05:45:00	4	2
2.	2	285	C	2022-11-18 00:59:00	05:00:00	5	2
3.	3	668	C	2022-12-06 19:01:00	02:00:00	5	4
4.	4	884	E	2022-12-07 14:50:00	21:00:00	7	2
5.	5	234	A	2022-11-25 21:40:00	13:00:00	2	6
6.	7	238	E	2018-01-11 04:24:00	03:41:20	13	5
7.	8	894	E	1979-09-10 16:37:36	19:31:11	15	6

Додавання нових випадкових даних до кожної таблиці БД.

Select an option:
1. Show all tables
2. Show table
3. Add new record
4. Update record
5. Delete record
6. Add new random records
7. Exit

Enter your choice: 6
Enter count of records: 3
Done

	airport_id	name	location
1.	12	venst	madrid
2.	13	venst	madrid
3.	14	Kyiv City Airport	Kyiv, Ukraine
4.	15	Xj	On
5.	16	2df	fg
6.	17	Xw	Qk
7.	18	Vf	Ey
8.	19	Vw	Eg
9.	20	Ur	Ki
10.	23	dtmbxulxol	kzueernzmc
11.	24	rjxitjqxhg	elktvqvrbt
12.	25	tmtazfzicf	dkybeulapc
13.	26	hpdtdpzfp	ifnosryssj
14.	27	zuzsowgegt	doiwrerpka
15.	28	wrfquxktix	owgsiqlecle
16.	29	iheseqhprb	qvcadducfj
17.	30	awpbpukgpp	uwjsgvehnz
18.	31	yvgojuzbkd	dhqwrkxstb
19.	32	anccalnryp	qvfwysddm
20.	33	nhcfwbiyoc	orktemrirw
21.	34	fswwckrlxr	pjhklzdyhe
22.	35	sbozessgqe	kioeikzvjjg
23.	36	rngpxfrbzb	uwyqhekoow
24.	37	jrwvlxoqpv	avxfphrl1l
25.	38	lothexhitf	thblpchbsj
26.	39	tkrfdgmpu	bexbapundr
27.	40	pntdwcflbs	ogfrrmdyjc
28.	41	dxhnhzhrjs	ncdwlktbvm
29.	42	ocgvwyphex	tsanjrwtnj
30.	43	ydvxrbecun	tzdpsowjis

===== aircraft =====				
	aircraft_id	pilot	name	owner
1.	1	Andrew Russel	Boeing 737-800	Bairbus
2.	2	Ura	Airbas	Claster Master
3.	4	Andrew Russel	Boeing 737-800	Airbuss
4.	5	Ira	Boeing	Airbuss
5.	6	Iu	Pw	Qs
6.	7	Dx	Cl	Gq
7.	8	Ky	Jm	Fm
8.	9	Su	Sv	Kw
9.	10	Fo	VfD	Dv
10.	11	Yt	It	Xi
11.	12	Mk	Ps	Ne
12.	13	Ef	Dl	Gq
13.	14	gwydsqsvdk	siznlsvbvl	sodwuhnbx1
14.	15	xfmdhvjdyn	fsljrcztjn	nfozlfffvb
15.	16	lbayfvpzrc	djxeqgrgmz	fjjbnlulks
16.	17	Ura	Bombardier CRJ-200	Bairbus
17.	18	cxftpzuii	lgmkyhoxjf	bnkorgoywn
18.	19	tcjcyzcodf	tgshbjwzqcn	gbwspvhkkc
19.	20	wvmglpvxwr	dmgeubkxgb	hfegkafwvb

===== airport_aircraft =====		
	aircraft_id	airport_id
1.	1	15
2.	4	17
3.	5	13
4.	5	30
5.	5	15
6.	6	14
7.	6	12
8.	7	16
9.	8	14
10.	9	12
11.	9	17
12.	9	31
13.	10	12
14.	11	17
15.	14	25
16.	15	24
17.	15	33
18.	16	12
19.	17	14
20.	17	15
21.	18	13
22.	19	34
23.	20	36

===== ticket =====							
	ticket_id	price	seat_type	date	duration	aircraft_id	user_acc_id
1.	1	4500	B	2022-10-10 17:17:40	05:45:00	4	2
2.	2	285	C	2022-11-18 00:59:00	05:00:00	5	2
3.	3	668	C	2022-12-06 19:01:00	02:00:00	5	4
4.	4	884	E	2022-12-07 14:50:00	21:00:00	7	2
5.	5	234	A	2022-11-25 21:40:00	13:00:00	2	6
6.	7	238	E	2018-01-11 04:24:00	03:41:20	13	5
7.	8	894	E	1979-09-10 16:37:36	19:31:11	15	6
8.	9	3327	D	2003-01-11 17:53:50	16:45:40	1	6
9.	10	1083	B	1999-04-17 21:21:58	15:23:07	17	5
10.	11	2017	B	1992-07-20 21:20:44	13:38:03	11	8

===== user_account =====				
	user_acc_id	name	mailbox	bank_card
1.	2	admin	admin@mainbox.com	1234567812345678
2.	3	Ura	ura@gmail.com	1025458412535412
3.	4	Hm	Db	202
4.	5	Jg	Fr	928
5.	6	Hv	Yd	675
6.	7	Ffl	fk	2345
7.	8	Ws	Xg	119
8.	9	De	Fc	158
9.	10	Andrey	mybox@mainbox.com	1284369475569871
10.	11	hhjyfxqypl	ifymvjtfnz	1680
11.	12	zputvswubg	doihtbtfcti	2895
12.	13	zvczbogskn	qdmxcrsqzs	130
13.	14	mpfbcjodrs	fqwslwaomp	4701

Завдання 2

BTree

Для дослідження індексу була створена таблиця, яка має дві колонки: str типу varchar і str_indexed типу varchar. Колонка str_indexed проіндексована як BTree.

Заповнимо обидва стовбця 500.000 випадковими даними.

```
create table strs(  
    str varchar,  
    str_indexed varchar  
);  
CREATE INDEX strs_indexed ON strs using btree (str_indexed);  
  
insert into strs SELECT  
    md5(random()::text),  
    md5(random()::text)  
from (  
    SELECT * FROM generate_series(1,500000) AS id  
) AS ser;
```

Тепер за допомогою Select виберемо і відсортуємо всі значення за спаданням і зрівняємо час виконання кожної з команд.

```
1 select * from strs order by str desc;  
2
```

Data Output	Messages	Notifications
Successfully run. Total query runtime: 2 secs 430 msec. 500000 rows affected.		

```
1 select * from strs order by str_indexed desc;  
2
```

Data Output	Messages	Notifications
Successfully run. Total query runtime: 400 msec. 500000 rows affected.		

Різниця при використанні індекса очевидна, ми отримуємо таку швидкодію через те, що дані в індексі впорядковані через незменшення, а сторінки одного рівня пов'язані між собою двонаправленим списком. Тому отримати впорядкований набір даних ми можемо просто проходячи по списку в одну або в іншу сторону, не повертаючись щоразу до кореня.

BRIN

Створимо та заповнимо таблицю з запису швидкості вітра кожної секунди протягом 2 років.

```
CREATE TABLE wind_log (  
    id int,  
    time_st timestamp without time zone,  
    speed int  
);  
  
INSERT INTO wind_log  
(  
    id,  
    time_st,  
    speed  
)  
VALUES  
(  
    round(random()*1000)::int,  
    generate_series('2020-01-01'::timestamp, '2022-01-01'::timestamp, '1 second'),  
    round(random()*50)::int  
);
```

Порахуємо середню температуру за 15 жовтня

```
1 SELECT AVG(speed)  
2 FROM wind_log  
3 WHERE time_st>='2021-10-15' AND time_st<'2021-10-16';
```

Data Output			Messages	Notifications
	avg			
	numeric	🔒		
1	25.0618287037037037			
Total rows: 1 of 1			Query complete 00:00:06.943	

Без індекса отримали час виконання: 6.943 секунди

Створимо Breen індекс:

```
DROP INDEX IF EXISTS average_speed;  
CREATE INDEX average_speed ON wind_log  
USING BRIN (time_st) WITH (pages_per_range = 128);
```

і запусимо аналогічний select

```
20 SELECT AVG(speed)
21 FROM wind_log
22 WHERE time_st>='2021-10-15' AND time_st<'2021-10-16';
```

Data Output Messages Notifications

	avg numeric	
1	25.0618287037037037	

Total rows: 1 of 1 Query complete 00:00:01.767

Отримали час виконання: 1.767 секунди

Створимо BTree індекс:

```
DROP INDEX IF EXISTS average_speed;
CREATE INDEX average_speed ON wind_log
USING btree (time_st);
```

і запусимо аналогічний select

```
20 SELECT AVG(speed)
21 FROM wind_log
22 WHERE time_st>='2021-10-15' AND time_st<'2021-10-16';
```

Data Output Messages Notifications

	avg numeric	
1	25.0618287037037037	

Total rows: 1 of 1 Query complete 00:00:00.161

Отримали час виконання: 0.161 секунди

Висновок

Додавання індексів значно підвищує швидкодію, але збільшує використання пам'яті, також варто пам'ятати, що для різних операцій треба обирати різні типи індекса, наприклад для надвеликих таблиць чудово підійде BTREE, в той же час BRIN добре працює для тих стовпців, значення в яких корелюють з їх фізичним розташуванням у таблиці.

Завдання 3

Умова для тригера – before insert, delete

Створення 2 додаткових таблиць для більш наглядної роботи тригерів

```
CREATE TABLE IF NOT EXISTS public.users_stats
```

```
(  
    total_sign_up integer NOT NULL,  
    last_time_sign_up timestamp NOT NULL  
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public.users_stats  
    OWNER to postgres;
```

```
--
```

```
INSERT INTO public.users_stats(  
    total_sign_up, last_time_sign_up)  
    VALUES (0, current_timestamp);
```

```
CREATE TABLE IF NOT EXISTS public.deleted_users
```

```
(  
    id integer NOT NULL  
)
```

```
TABLESPACE pg_default;
```

```
ALTER TABLE IF EXISTS public.deleted_users  
    OWNER to postgres;
```

Тригери

before insert

```
CREATE OR REPLACE FUNCTION insert_func()
```

```
    RETURNS trigger AS $insert_func$
```

```
BEGIN
```

```
    UPDATE users_stats SET total_sign_up = total_sign_up + 1, last_time_sign_up  
= current_timestamp
```

```
    WHERE id = 1;
```

```
    RETURN NULL;
```

```
END;
```

```
$insert_func$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER UsersInsertTrigger
```

```
    BEFORE INSERT ON user_account
```

```
    FOR EACH ROW EXECUTE FUNCTION insert_func();
```

before delete

```
CREATE OR REPLACE FUNCTION delete_func()
```

```
    RETURNS trigger AS $delete_func$
```

```
BEGIN
```

```
    INSERT INTO deleted_users (id) VALUES (OLD.user_acc_id);
```

```
    RETURN NULL;
```

```
END;
```

```
$delete_func$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER UsersDeleteTrigger
```

```
    BEFORE DELETE ON user_account
```

```
    FOR EACH ROW EXECUTE FUNCTION delete_func();
```

Принцип роботи

Тригер спрацьовує перед додаванням або видаленням в таблиці 'user_account'.

При додаванні нового рядка в таблицю 'user_account', таблиця 'users_stats' оновлює свій зміст, а саме інкрементує значення total_sign_up та оновлює часову мітку на теперішнє значення.

При видаленні рядка в таблиці 'user_account', в таблицю 'deleted_users' заноситься новий зміст, а саме додається видалене значення user_acc_id в колонку deleted_users.id

Приклад

Таблиці до змін:

users_stats

	total_sign_up integer	last_time_sign_up timestamp without time zone
1	0	2023-01-10 15:52:27.921518

deleted_users

	id integer
--	---------------

Введемо три записи до кожної таблиці

```
Select an option:
  1. Show all tables
  2. Show table
  3. Add new record
  4. Update record
  5. Delete record
  6. Add new random records
  7. Exit

Enter your choice: 6
Enter count of records: 3
```

Таблиця 'users_stats' оновилася

	total_sign_up integer	last_time_sign_up timestamp without time zone	id [PK] integer
1	3	2023-01-10 16:18:26.993799	1

Видалимо три записи з таблиці 'User Account'

```
Select an option:
1. Show all tables
2. Show table
3. Add new record
4. Update record
5. Delete record
6. Add new random records
7. Exit

Enter your choice: 5

Select a table:
1. Airport
2. Aircraft
3. Airport/Aircraft
4. Ticket
5. Users Account
6. Exit

Enter your choice: 5
```

```
Enter data for delete from table: user_account
Enter user_acc_id for delete: 16


Enter data for delete from table: user_account
Enter user_acc_id for delete: 10

Enter data for delete from table: user_account
Enter user_acc_id for delete: 17
```

Таблиця 'User_Account' після видалення

	user_acc_id	name	mailbox	bank_card
1.	2	admin	admin@mainbox.com	1234567812345678
2.	3	Ura	ura@gmail.com	1025458412535412
3.	4	Hm	Db	202
4.	5	Jg	Fr	928
5.	6	Hv	Yd	675
6.	7	Ff1	fk	2345
7.	8	Ws	Xg	119
8.	9	De	Fc	158
10.	11	hhjyfxqypl	ifymvjtfnz	1680
11.	12	zputvswubg	doihtbcti	2895
12.	13	zvczboggkn	qdmxcrsqzs	130
13.	14	mpfbcjodrs	fqwslwaomp	4701

Таблиця 'deleted_users' після спрацювання тригерів

	id integer 
1	10
2	16
3	17