

## Laboratory work 7

1. Create an index on the actual\_departure column in the flights table.

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the PostgreSQL 17 database structure. The right pane is the Query Editor, showing a successful execution of the SQL command:

```
create index idx_act_dep on flights(actual_departure);
```

The Messages tab indicates the query was returned successfully in 106 msec. The status bar at the bottom right shows "Query complete 00:00:00.106".

2. Create a unique index to ensure flight\_no and scheduled\_departure combinations are unique.

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the PostgreSQL 17 database structure. The right pane is the Query Editor, showing a successful execution of the SQL command:

```
create unique index idx_uniq_flight on flights (flight_no,scheduled_departure);
```

The Messages tab indicates the query was returned successfully in 91 msec. The status bar at the bottom right shows "Query complete 00:00:00.091".

### 3. Create a composite index on the departure\_airport\_id and arrival\_airport\_id columns.

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the database structure. The right pane is the Query Editor, showing the command to create a composite index:

```
create index idx_airport on flights(departure_airport_id, arrival_airport_id);
```

The Data Output tab shows the result of the query:

CREATE INDEX

Query returned successfully in 93 msec.

Total rows: Query complete 00:00:00.093 CRLF Ln 1, Col 80

A green message bar at the bottom right indicates: **✓ Query returned successfully in 93 msec. X**

### 4. Evaluate the difference in query performance with and without indexes. Measure performance differences.

The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer. The right pane is the Query Editor, displaying the EXPLAIN ANALYZE output for a query:

```
explain analyze
select * from flights where actual_departure > '2025-11-10' ;
```

The Data Output tab shows the results of the query, including the execution plan:

Showing rows: 1 to 4 Page No: 1 of 1

QUERY PLAN

1	Index Scan using idx_act_dep on flights (cost=0.28..8.29 rows=1 width=63) (actual time=0.005..0.006 rows=0 loops=...)
2	Index Cond: (actual_departure > '2025-11-10'::date)
3	Planning Time: 6.988 ms
4	Execution Time: 0.040 ms

Total rows: 4 Query complete 00:00:00.104 CRLF Ln 2, Col 62

```

explain analyze
select * from flights where actual_departure > '2025-11-10';

```

**QUERY PLAN**

1	Index Scan using idx_act_dep on flights (cost=0.28..8.29 rows=1 width=63) (actual time=0.008..0.008 rows=0 loops=...
2	Index Cond: (actual_departure > '2025-11-10'::date)
3	Planning Time: 0.251 ms
4	Execution Time: 0.032 ms

Successfully run. Total query runtime: 122 msec. 4 rows affected.

## 5. Use EXPLAIN ANALYZE to check index usage in a query filtering by departure\_airport and arrival\_airport.

```

explain analyze
select * from flights where departure_airport_id = 1 and arrival_airport_id = 1;

```

**QUERY PLAN**

1	Bitmap Heap Scan on flights (cost=4.31..12.95 rows=3 width=63) (actual time=0.794..0.803 rows=3 loops=1)
2	Recheck Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 1))
3	Heap Blocks: exact=3
4	-> Bitmap Index Scan on idx_airport (cost=0.00..4.31 rows=3 width=0) (actual time=0.126..0.126 rows=3 loop...
5	Index Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 1))
6	Planning Time: 0.326 ms
7	Execution Time: 2.658 ms

Total rows: 7 Query complete 00:00:00.120

6. Create a unique index for the passport\_number of the Passengers table. Check if the index was created or not. Insert into the table two new passengers.

Explain in your own words what is going on in the output?

create

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'Tables' section of the 'airline' table, there is a 'Columns' section containing 'airline\_id' and 'airline\_code'. In the main query editor window, the following SQL command is run:

```
create unique index idx_passport_uni on passengers(passport_number);
```

The 'Messages' tab shows the result of the query:

```
CREATE INDEX
Query returned successfully in 132 msec.
```

A green message bar at the bottom right indicates:

✓ Query returned successfully in 132 msec. CRLF Ln 1, Col 70

Check

The screenshot shows the pgAdmin 4 interface again. In the Object Explorer, under the 'Tables' section of the 'airline' table, there is a 'Columns' section containing 'airline\_id' and 'airline\_code'. In the main query editor window, the following SQL command is run:

```
select indexname, indexdef from pg_indexes where tablename = 'passengers';
```

The 'Data Output' tab displays the results of the query:

indexname	indexdef
passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
uq_passport_number	CREATE UNIQUE INDEX uq_passport_number ON public.passengers USING btree (passport_number)
idx_passport_uni	CREATE UNIQUE INDEX idx_passport_uni ON public.passengers USING btree (passport_number)

A green message bar at the bottom right indicates:

✓ Query returned successfully in 132 msec. CRLF Ln 1, Col 70

## Insert 2 passengers

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer pane, which lists servers, databases, schemas, and tables. The current connection is to the 'airport\_db' database. In the center is the Query Editor pane, showing a SQL query and its execution results.

```
1 insert into passengers
2 (passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship,
3 country_of_residence, passport_number, created_at, update_at)
4 values
5 (201, 'Kuralay', 'Kassym', '2001-06-15', 'female', 'Kazakhstan', 'Kazakhstan', '4657385', '2
6 (202, 'Dana', 'Kairat', '1998-02-20', 'female', 'Kazakhstan', 'Kazakhstan', '4657385', '2024
```

The Messages tab displays an error message:

ERROR: повторяющееся значение ключа нарушает ограничение уникальности "uq\_passport\_number"  
Ключ "(passport\_number)=(4657385)" уже существует.

The Notifications tab displays an additional error message:

ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "uq\_passport\_number"  
SQL state: 23505  
Detail: Ключ "(passport\_number)=(4657385)" уже существует.

At the bottom of the Query Editor, it says "Total rows: 200 Query complete 00:00:00.117".

**The unique index allows only one unique passport number per passenger, when I inserted 2 new passengers, it gave an error, because this 2 passengers have the same passport number. This happens because unique index prevents duplicate passport numbers in the table.**

7. Create an index for the Passengers table. Use for that first name, last name, date of birth and country of citizenship. Then, write a SQL query to find a passenger who was born in Philippines and was born in 1984 and check if the query uses indexes or not. Give the explanation of the results.

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Servers (1)
  - PostgreSQL 17
    - Databases (3)
      - airport\_db
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Publications
        - Schemas (1)
          - public
            - Aggregates
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Operators
            - Procedures
            - Sequences
          - Tables (10)
            - airline
              - Columns (6)
                - airline\_id

airport\_db/postgres@PostgreSQL 17\*

Query Query History

```
CREATE INDEX idx_passengerinfo ON passengers(first_name, last_name, date_of_birth, country_of_citizenship)
```

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 109 msec.

Total rows: 5 Query complete 00:00:00.109 CRLF Ln 1, Col 92

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Servers (1)
  - PostgreSQL 17
    - Databases (3)
      - airport\_db
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Publications
        - Schemas (1)
          - public
            - Aggregates
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Operators
            - Procedures
            - Sequences
          - Tables (10)
            - airline
              - Columns (6)
                - airline\_id

airport\_db/postgres@PostgreSQL 17\*

Query Query History

```
explain analyze
select * from passengers where country_of_citizenship = 'Philippines' and date_of_birth between '1984-01-01' and '1984-12-31'
```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

QUERY PLAN

text
1 Seq Scan on passengers (cost=0.00..6.50 rows=1 width=64) (actual time=0.041..0.102 rows=1 loops=1)
2 Filter: ((date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '1984-12-31'::date) AND ((country_of_citizenship)::text = 'Philippines'::text))
3 Rows Removed by Filter: 199
4 Planning Time: 1.491 ms
5 Execution Time: 0.132 ms

Total rows: 5 Query complete 00:00:00.104 CRLF Ln 2, Col 128

The query used seq scan , so it checked every row in the table to find this passengers. It didn't use index that was created,because index is used from left to right,starting with the

first columns that were defined in the index, also the table is small, so it would be faster than using the index , to scan all rows.

## 8. Write a SQL query to list indexes for table Passengers. After delete the created indexes.

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'airport\_db' database, the 'Tables' node is expanded, showing 'airline' and its 'Columns' (including 'airline\_id'). In the main window, a query is run:

```
select * from pg_indexes where tablename = 'passenger';
```

The results show four indexes for the 'passenger' table:

schemaname	tablename	indexname	tablespace	indexdef
public	passenger	passenger_pkey	[null]	CREATE UNIQUE INDEX passenger_pkey ON public.passenger
public	passenger	uq_passport_number	[null]	CREATE UNIQUE INDEX uq_passport_number ON public.passenger
public	passenger	idx_passport_uni	[null]	CREATE UNIQUE INDEX idx_passport_uni ON public.passenger
public	passenger	idx_passengerinfo	[null]	CREATE INDEX idx_passengerinfo ON public.passenger USING t

Message bar: Successfully run. Total query runtime: 162 msec. 4 rows affected.

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'airport\_db' database, the 'Tables' node is expanded, showing 'airline' and its 'Columns' (including 'airline\_id'). In the main window, two indexes are dropped:

```
drop index idx_passport_uni;
drop index idx_passengerinfo;
```

The results show the operation was successful:

Query returned successfully in 122 msec.

Message bar: Query returned successfully in 122 msec.