



PG_DWH TASK 2

Legal Notice: This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

Confidential

CONTENTS

1.

```
1  CREATE TABLE IF NOT EXISTS employee(  
2      id serial ,  
3      name varchar(100),  
4      status varchar(100)  
5  )  
6  )
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 34 msec.

Table created.

2.

```
8
9
10 -- first transaction
11 begin;
12 select txid_current();
13 v insert into public.employee ("name", status)
14 values ('Alice', 'Not fired');
15 v select *, xmin, xmax
16 from public.employee e;
17 commit;
18
19
20
21 -- first transaction
22 begin;
23 select *, xmin, xmax from public.employee e;
24 commit;
```

Data Output Messages Notifications

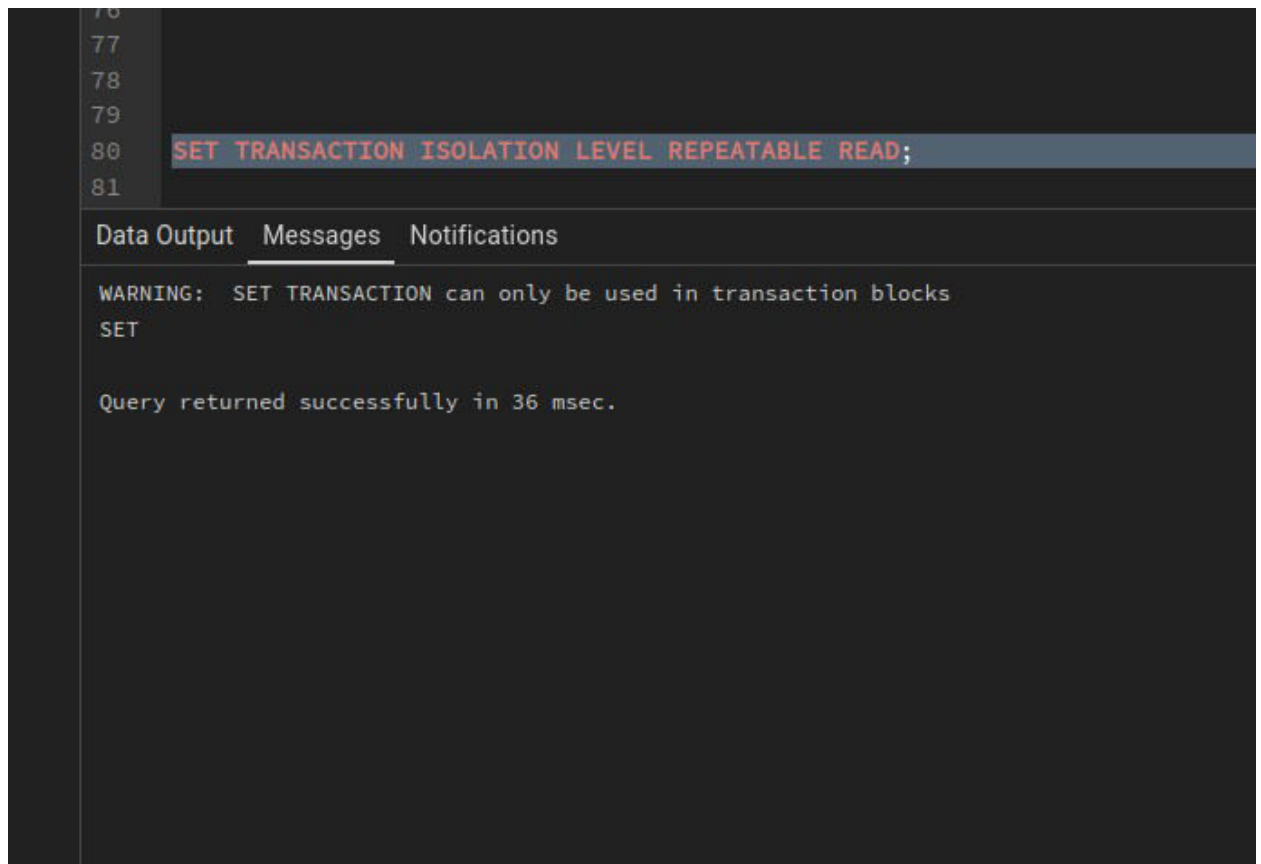
≡+ 📄 ▼ 📋 ▼ 🗑️ 🗑️ 📥 📥 📈 SQL

	Id integer	name character varying (100)	status character varying (100)	xmin xid	xmax xid
--	---------------	---------------------------------	-----------------------------------	-------------	-------------

All queries are executed as in lecture .

```
Query  Query History
36
37 -- third transaction
38 begin;
39 ✓ select *, xmin, xmax
40 from public.employee e;
41 ✓ select *, xmin, xmax
42 from public.employee e;
43 commit;
44
45
46
47 -- first transaction
48 begin;
49 ✓ select *, xmin, xmax from public.employee
50 e;
51 commit;
52
53
54 -- second transaction
55 begin;
56 select txid_current();
57 ✓ delete from public.employee
58 where id = 1;
59 ✓ select *, xmin, xmax,cmin,cmax
60 from public.employee e;
61 commit;
62
63
64
65
66 -- third transaction
67 begin;
68 select txid_current();
69 ✓ update public.employee
70 set status = 'Fired'
71 where id = 2;
72 ✓ select *, xmin, xmax
73 from public.employee e;
74
75 commit;
```

3.



```
76
77
78
79
80 SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
81
```

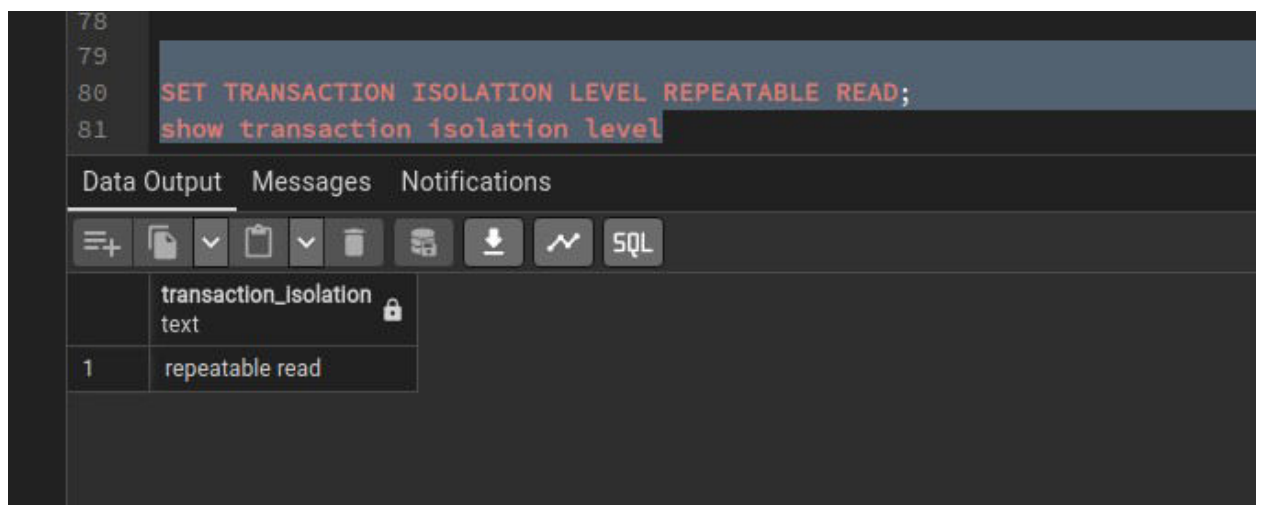
Data Output Messages Notifications

WARNING: SET TRANSACTION can only be used in transaction blocks
SET

Query returned successfully in 36 msec.

4.

Set Transaction isolation level to repeatable read.



```
78
79
80 SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
81 show transaction isolation level
```

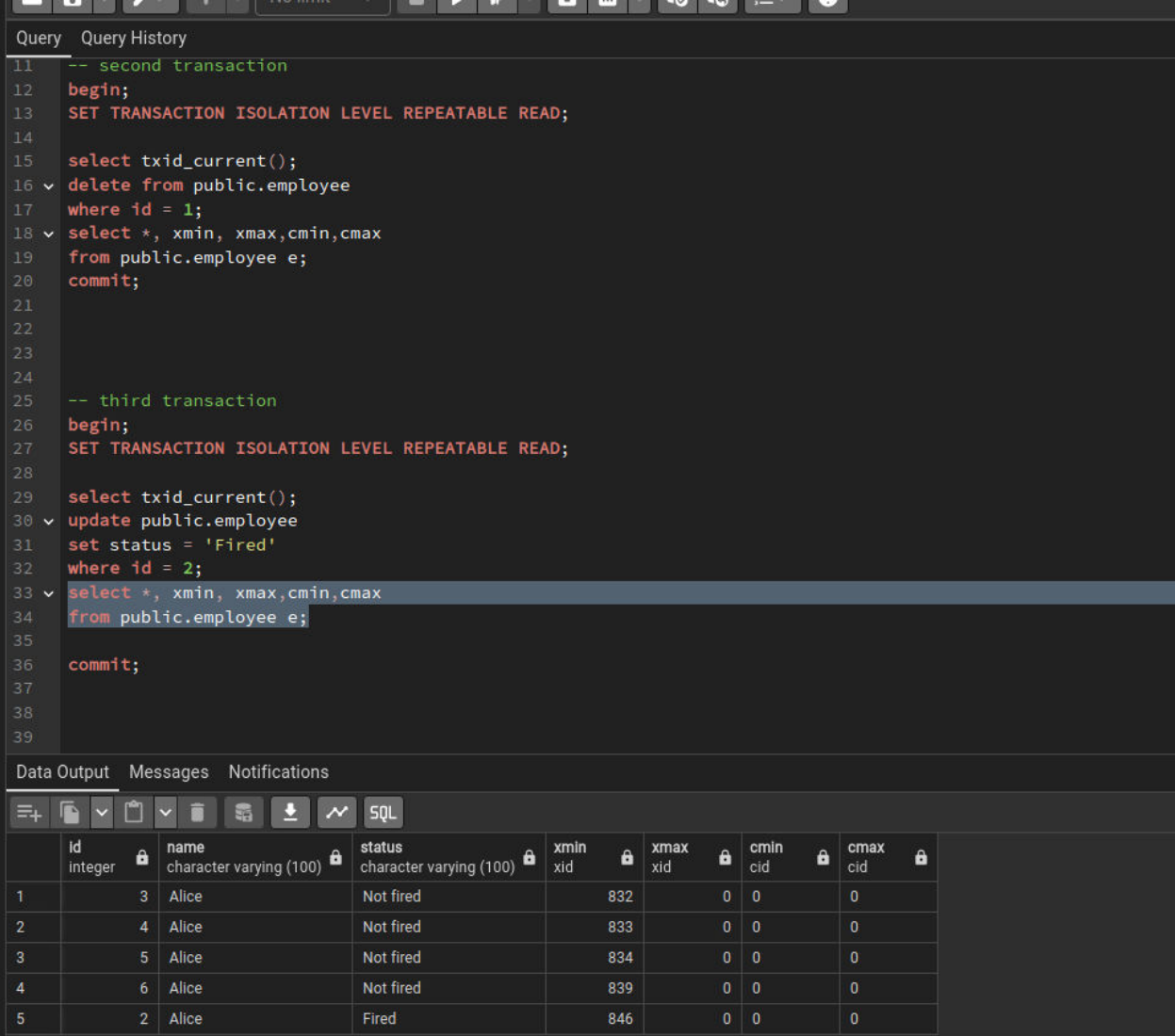
Data Output Messages Notifications

transaction_isolation text

1	repeatable read
---	-----------------

Transaction isolation level is repeatable read.

5.



The screenshot shows a SQL IDE interface with a query editor and a data output table. The query editor contains two transactions. The first transaction starts with a comment '-- second transaction', followed by 'begin;', 'SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;', 'select txid_current();', 'delete from public.employee where id = 1;', 'select *, xmin, xmax, cmin, cmax from public.employee e;', and 'commit;'. The second transaction starts with a comment '-- third transaction', followed by 'begin;', 'SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;', 'select txid_current();', 'update public.employee set status = 'Fired' where id = 2;', 'select *, xmin, xmax, cmin, cmax from public.employee e;', and 'commit;'. The data output table has columns: id (integer), name (character varying (100)), status (character varying (100)), xmin (xid), xmax (xid), cmin (cid), and cmax (cid). The table contains 5 rows of data.

```
11 -- second transaction
12 begin;
13 SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
14
15 select txid_current();
16 delete from public.employee
17 where id = 1;
18 select *, xmin, xmax, cmin, cmax
19 from public.employee e;
20 commit;
21
22
23
24
25 -- third transaction
26 begin;
27 SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
28
29 select txid_current();
30 update public.employee
31 set status = 'Fired'
32 where id = 2;
33 select *, xmin, xmax, cmin, cmax
34 from public.employee e;
35
36 commit;
37
38
39
```

	id integer	name character varying (100)	status character varying (100)	xmin xid	xmax xid	cmin cid	cmax cid
1	3	Alice	Not fired	832	0	0	0
2	4	Alice	Not fired	833	0	0	0
3	5	Alice	Not fired	834	0	0	0
4	6	Alice	Not fired	839	0	0	0
5	2	Alice	Fired	846	0	0	0

All queries are executed with isolation level repeatable read.

6.

```
137
138
139
140 BEGIN;
141
142 SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
143
144
145 ✓ update public.employee
146    set status = 'Fired'
147   where id = 2;
148 COMMIT;
149
150
```

Data Output	Messages	Notifications
UPDATE 1		
Query returned successfully in 59 msec.		

First transaction works well as expected ,

```
33
34 BEGIN;
35
36 SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
37
38
39 ✓ update public.employee
40 set status = 'Fired'
41 where id = 2;
42 COMMIT;
```

Data Output Messages Notifications

Total rows: 6 of 6 Waiting for the query to complete... 00:00:16.925 Ln 51, Col 14

Second transaction waiting for the first to be complete then it will be executed. When I commit first one I got this error.

```
42
43
44 BEGIN;
45
46 SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
47
48
49 ✓ update public.employee
50 set status = 'Fired'
51 where id = 2;
52 COMMIT;
```

Data Output Messages Notifications

ERROR: could not serialize access due to concurrent update

SQL state: 40001

Total rows: 6 of 6 Query complete 00:01:19.252 Ln 51, Col 14

This means Postgres locked access due to concurrent update.

7.

```
151
152
153
154
155 BEGIN;
156
157 SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
158
159
160 ✓ update public.employee
161    set status = 'Fired'
162    where id = 2;
163 COMMIT;
164
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 38 msec.

Total rows: 6 of 6 Query complete 00:00:00.038 Ln 162, Col 14

We successfully update our table in first transaction and when I run second one it's also waiting for the first one to be committed.

```
52 COMMIT;
53
54
55
56
57
58
59 BEGIN;
60
61 SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
62
63
64 ✓ update public.employee
65    set status = 'Fired'
66    where id = 2;
67 COMMIT;
68
69
70
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 5 min 21 secs.

Total rows: 6 of 6 Query complete 00:05:21.931 Ln 66, Col 14

And after I commit first one , second one also executed successfully and updated row.