

EPAM Systems, RD Dep., RD Dep.

POSTGRESQL DB FOR DWH AND ETL BUILDING

Transaction

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®.

- 1. Create a table called employee with columns id serial, name varchar, status varchar.
- 2. Replicate the example given in the lecture with the code below:

first transaction	first transaction
begin;	begin;
<pre>select txid_current();</pre>	select *, xmin, xmax from public.employee
	e;
insert into public.employee ("name", status)	
values ('Alice', 'Not fired');	commit;
select *, xmin, xmax	
from public.employee e;	
commit;	
second transaction	second transaction
begin;	begin;
select *, xmin, xmax	select txid_current();
from public.employee e;	
	delete from public.employee
select *, xmin, xmax	where id = 1;
from public.employee e;	
	select *, xmin, xmax
commit;	from public.employee e;
	commit;
insert into public.employee ("name", status)	
values ('Alice', 'Not fired');	
third transaction	third transaction
begin;	begin;
select *, xmin, xmax	select txid_current();
from public.employee e;	
	update public.employee
select *, xmin, xmax	set status = 'Fired'
from public.employee e;	where id = 2;
commit;	select *, xmin, xmax
	from public.employee e;

Confidential 2

Transaction <epam>

commit;

- 3. Run the command set transaction isolation level repeatable read.
- 4. Check your current isolation level in each session with show transaction isolation level.
- 5. Recreate employee table and redo the second task but modify the code so that select statements would now include **cmin** and **cmax** system columns. What changed?
- 6. * Try to cause a *serialization anomaly* on the employee table (add more data if necessary). Change your isolation level to serializable and try to cause *serialization anomaly* one more time. What happened?
- 7. * Set your isolation level to read committed. Try to cause a *lost update* database anomaly on the employee table (add more data if necessary). What happened? What do you think are the downsides of the approach that Postgres took to handle this anomaly?

Confidential 3