به نام خدا

حدیث غفوری 9825413

سوال 1

university_small/postgres@localhost

Query Editor Query History

```
create table trader(
    trader_id char(10) primary key,
    trader_name varchar(20),
    birth_date date check(birth_date >= '1900-01-01'),
    Joined_date date check(Joined_date > birth_date),
    Salary numeric(8,2) check(Salary>= 1000)
);
```

Data Output Explain Messages Notifications

```
CREATE TABLE

Query returned successfully in 142 msec.
```

university_small/postgres@localhost

Query Editor Query History

```
insert into trader values(1111,'hadis','2001-05-04','2019-01-01',999);
```

Data Output Explain Messages Notifications

```
ERROR:  new row for relation "trader" violates check constraint "trader_salary_check"
DETAIL:  Failing row contains (1111      , hadis, 2001-05-04, 2019-01-01, 999.00).
SQL state: 23514
```

university_small/postgres@localhost

Query Editor Query History

```
insert into trader values(1111,'hadi','1899-05-04','2019-01-01',2000);
```

Data Output Explain Messages Notifications

```
ERROR:  new row for relation "trader" violates check constraint "trader_birth_date_check"
DETAIL:  Failing row contains (1111      , hadi, 1899-05-04, 2019-01-01, 2000.00).
SQL state: 23514
```

سوال3

dvdrental/postgres@localhost

Query Editor Query History

```
select concat(first_name,' ',last_name) as full_name,film.title
from customer inner join rental using(customer_id)
inner join payment using(rental_id)
inner join inventory using(inventory_id)
inner join film using(film_id)
where (payment_date - rental_date > INTERVAL '1 year 3 months 15 days' )
order by full_name;
```

Data Output Explain Messages Notifications

	full_name text	title character varying (255)
1	Aaron Selby	Shock Cabin
2	Aaron Selby	Network Peak
3	Aaron Selby	Fever Empire
4	Aaron Selby	Liaisons Sweet
5	Aaron Selby	Perfect Groove
6	Aaron Selby	Beauty Grease
7	Aaron Selby	Drumline Cyclone
8	Aaron Selby	Sleeping Suspects
9	Aaron Selby	Patient Sister
10	Aaron Selby	Valentine Vanishing

سوال 4

روش 1:

You could store them as BLOBs (or LONGBLOBs) and then retrieve the data out when you want to actually access the media files.

You can store audio files in any database that supports blobs, this should be possible in any SQL database and on non-sql databases as well.

However this is kind of inefficient because serving the files had some overhead. Storing the files in a filesystem or a storage like Amazon S3 is probably better

روش 2:

You could simply store the media files on a drive and store the metadata in the DB

Some advantages of using blobs to store files

Lower management overhead - use a single tool to backup / restore etc

No possibility for database and filesystem to be out of sync

Transactional capability (if needed)

Some disadvantages

blows up your database servers' RAM with useless rubbish it could be using to store rows, indexes etc

Makes your DB backups very large, hence less manageable

Not as convenient as a filesystem to serve to clients (e.g. with a web server)

Most of systems stores large numbers of big files stores them externally to the database. You store all of the queryable data for the file (title, artist, length, etc) in the database, along with a partial path to the file. When it's time to retrieve the file, you extract the file's path, prepend some file root (or URL) to it, and return that.

So, you'd have a "location" column, with a partial path in it, like "a/b/c/1000", which you then map to: "http://myserver/files/a/b/c/1000.mp3"

Make sure that you have an easy way to point the media database at a different server/directory, in case you need that for data recovery. Also, you might need a routine that re-syncs the database with the contents of the file archive.

Also, if you're going to have thousands of media files, don't store them all in one giant directory - that's a performance bottleneck on some file systems. Instead,break them up into multiple balanced sub-trees.

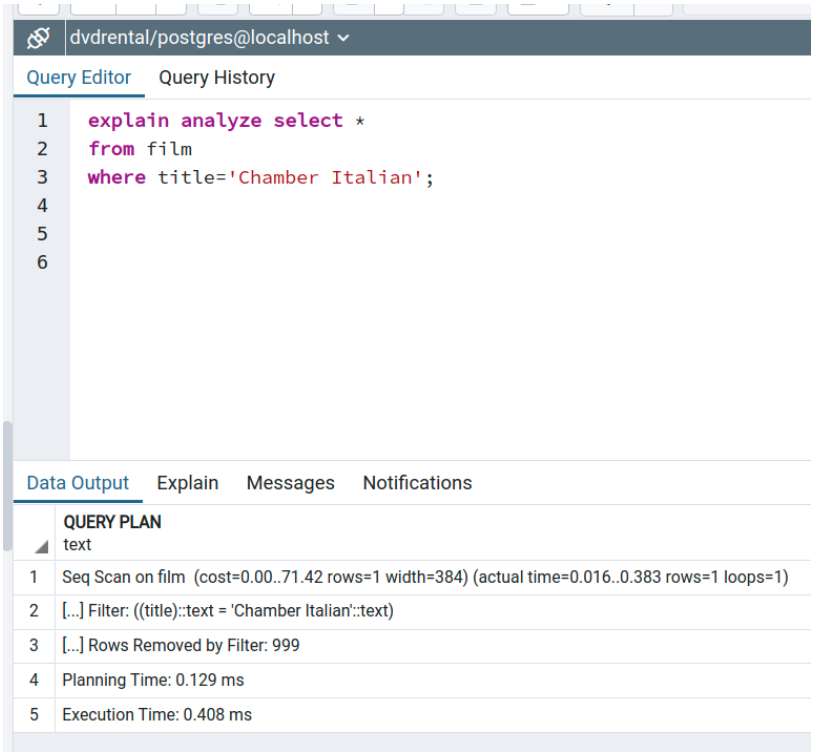
Store them as external files. Then save the path in a varchar field. Putting large binary blobs into a relational database is generally very inefficient - they only use up space and slow things down as caches are filled are unusable. And there's nothing to be gained - the blobs themselves cannot be searched. You might want to save media meta data into the the database though.

Save the files in storage. Store the unambiguous filename (the filename starting at the effective root) of the file in the database. When you need the file, retrieve the name form the database, then retrieve the file from storage.

(Storing the file as a BLOB usually means saving it as a temporary file after retrieving the BLOB, wasting a lot of time - and a lot of database storage.)

سوال 5

اجرای دستور بدون index



اجرای دستور با index

طبق نکته ی زیر چون اینجا داریم شرط مساوی را چک میکنیم بهترین روش استفاده از hash است.

Hash indexes can handle only simple equality comparison (=). It means that whenever an indexed column is involved in a comparison using the equal(=) operator, the query planner will consider using a hash index.

طبق تصاویر :

زمان اجرای دستور با index خیلی کمتر است نسبت به زمانی که index نداشتیم.

Query Editor Query History

```
-- create index idx_title
-- on film using hash (title) ;

 explain analyze select *
 from film
 where title='Chamber Italian';
```

Data Output Explain Messages Notifications

	QUERY PLAN text
1	Index Scan using idx_title on film (cost=0.00..8.02 rows=1 width=384) (actual time=0.015..0.016 rows=1 loops=1)
2	[...] Index Cond: ((title)::text = 'Chamber Italian'::text)
3	Planning Time: 2.090 ms
4	Execution Time: 0.065 ms

سوال 6

adventure_works/postgres@localhost

Query Editor Query History

```
select
    tablename,
    indexname,
    indexdef
from
    pg_indexes
where
    schemaname = 'sales'
order by
    tablename,
    indexname;
```

Data Output Explain Messages Notifications

	tablename name	indexname name	indexdef text
1	countryregioncurrency	PK_CountryRegionCurrency_CountryRegionCode_CurrencyCode	CREATE UNIQUE INDEX "PK_CountryRegionCurrency_CountryRegionCode_CurrencyCode" ON sales.countryregioncu
2	creditcard	PK_CreditCard_CreditCardID	CREATE UNIQUE INDEX "PK_CreditCard_CreditCardID" ON sales.creditcard USING btree (creditcardid)
3	currency	PK_Currency_CurrencyCode	CREATE UNIQUE INDEX "PK_Currency_CurrencyCode" ON sales.currency USING btree (currencycode)
4	currencyrate	PK_CurrencyRate_CurrencyRateID	CREATE UNIQUE INDEX "PK_CurrencyRate_CurrencyRateID" ON sales.currencyrate USING btree (currencyrateid)
5	customer	PK_Customer_CustomerID	CREATE UNIQUE INDEX "PK_Customer_CustomerID" ON sales.customer USING btree (customerid)
6	personcreditcard	PK_PersonCreditCard_BusinessEntityID_CreditCardID	CREATE UNIQUE INDEX "PK_PersonCreditCard_BusinessEntityID_CreditCardID" ON sales.personcreditcard USING b
7	salesorderdetail	PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID	CREATE UNIQUE INDEX "PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID" ON sales.salesorderdetail USING b
8	salesorderheader	PK_SalesOrderHeader_SalesOrderID	CREATE UNIQUE INDEX "PK_SalesOrderHeader_SalesOrderID" ON sales.salesorderheader USING btree (salesorderi
9	salesorderheadersalesreason	PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID	CREATE UNIQUE INDEX "PK_SalesOrderHeaderSalesReason_SalesOrderID_SalesReasonID" ON sales.salesorderhea
10	salesperson	PK_SalesPerson_BusinessEntityID	CREATE UNIQUE INDEX "PK_SalesPerson_BusinessEntityID" ON sales.salesperson USING btree (businessentityid)
11	salespersonquotahistory	PK_SalesPersonQuotaHistory_BusinessEntityID_QuotaDate	CREATE UNIQUE INDEX "PK_SalesPersonQuotaHistory_BusinessEntityID_QuotaDate" ON sales.salespersonquotahis

سوال 7

Query Editor Query History

```
-- create role role1;
grant select on instructor to role1 with grant option;
```

Data Output Explain Messages Notifications

```
GRANT

Query returned successfully in 76 msec.
```

Query Editor Query History

```
-- create role role2;
grant role1 to role2;
-- grant insert,delete,update on instructor,course to role2;
```

Data Output Explain Messages Notifications

```
GRANT ROLE

Query returned successfully in 66 msec.
```

Query Editor Query History

```
-- create role role2;
-- grant role1 to role2;
grant insert,delete,update on instructor,course to role2;
```

Data Output Explain Messages Notifications

```
GRANT

Query returned successfully in 66 msec.
```

Query Editor Query History

```
-- create role role2;
-- grant role1 to role2;
grant insert,delete,update on instructor,course to role2;
```

Data Output Explain Messages Notifications

```
GRANT

Query returned successfully in 66 msec.
```

Query Editor Query History

```
--  create role role3;
--  grant update (dept_name) on student to role3;
revoke update (dept_name) on student from role3;
```

Data Output Explain Messages Notifications

```
REVOKE

Query returned successfully in 71 msec.
```

Query Editor Query History

```
-- create role role4;
 create view specific_students as
 (
    select *
    from takes inner join student using(id)
    where dept_name in('Comp. Sci.','Elec. Eng.')
 );
```

Data Output Explain Messages Notifications

```
CREATE VIEW

Query returned successfully in 104 msec.
```

سوال 8

Query Editor Query History

```
set balance = balance -200
where name ='Amir';
---where id=1;


update accounts
set balance = balance +200
where name ='Ali';
---where id=2;

select id,name,balance
from accounts
commit;
```

Data Output Explain Messages Notifications

	id [PK] integer	name character varying (100)	balance numeric (15,3)
1	1	Amir	800.000
2	2	Ali	1200.000

Query Editor Query History

```


-- update accounts
-- set balance = balance +200
-- where name ='Ali';
---where id=2;

-- select id,name,balance
-- from accounts
-- commit;
rollback;


```

Data Output Explain Messages Notifications

```
ROLLBACK

Query returned successfully in 89 msec.
```

Query Editor Query History

```
7
8
9   -- update accounts
10  -- set balance = balance +200
11  -- where name ='Ali';
12  ---where id=2;
13
14  -- commit;
15  -- rollback;
16  select id,name,balance
17   from accounts
18
19
```

Data Output Explain Messages Notifications

	id [PK] integer	name character varying (100)	balance numeric (15,3)
1	1	Amir	1000.000
2	2	Ali	1000.000

C:

وقتی قبل از commit کردن rollback کنیم کل تراکنش برگردانده میشود و لاگ هایی که مربوط به قسمت Begin که شروع تراکنش را مشخص میکردند پاک میشوند.

Query Editor Query History

```
1   begin;
2   update accounts
3   set balance = balance -150
4   where name ='Ali';
5
6   update accounts
7   set balance = balance +150
8   where name ='Amir';
9   rollback;
10
11  -- select id,name,balance
```

Data Output Explain Messages Notifications

```
ROLLBACK

Query returned successfully in 60 msec.
```

Query Editor Query History

```
3   set balance = balance -150
4   where name ='Ali';
5   ---where id=2;
6
7   update accounts
8   set balance = balance +150
9   where name ='Amir';
10  ---where id=1;
11  rollback;
12  commit;
13  select id,name,balance
14  from accounts
15
```

Data Output Explain Messages Notifications

	id [PK] integer	name character varying (100)	balance numeric (15,3)
1	1	Amir	1000.000
2	2	Ali	1000.000

طبق جدول زیر میتونیم معانی ستون ها در جدول lock را بفهمیم:

Table 48-61. pg_locks Columns

Name	Type	References	Description
locktype	text		Type of the lockable object: relation, extend, page, tuple, transactionid, virtualxid, object, userlock, or advisory
database	oid	pg_database.oid	OID of the database in which the lock target exists, or zero if the target is a shared object, or null if the target is a transaction ID
relation	oid	pg_class.oid	OID of the relation targeted by the lock, or null if the target is not a relation or part of a relation
page	integer		Page number targeted by the lock within the relation, or null if the target is not a relation page or tuple
tuple	smallint		Tuple number targeted by the lock within the page, or null if the target is not a tuple
virtualxid	text		Virtual ID of the transaction targeted by the lock, or null if the target is not a virtual transaction ID
transactionid	xid		ID of the transaction targeted by the lock, or null if the target is not a transaction ID
classid	oid	pg_class.oid	OID of the system catalog containing the lock target, or null if the target is not a general database object
objid	oid	any OID column	OID of the lock target within its system catalog, or null if the target is not a general database object
objsubid	smallint		Column number targeted by the lock (the classid and objid refer to the table itself), or zero if the target is some other general database object, or null if the target is not a general database object
virtualtransaction	text		Virtual ID of the transaction that is holding or awaiting this lock
pid	integer		Process ID of the server process holding or awaiting this lock, or null if the lock is held by a prepared transaction
mode	text		Name of the lock mode held or desired by this process (see Section 13.3.1 and Section 13.2.3)
granted	boolean		True if lock is held, false if lock is awaited
fastpath	boolean		True if lock was taken via fast path, false if taken via main lock table

relation	pg_attribute							7/38	AccessShareLock
relation	accounts_pkey							7/38	AccessShareLock
relation	accounts_pkey							7/38	RowExclusiveLock
relation	accounts							7/38	AccessShareLock
relation	accounts							7/38	RowExclusiveLock
relation	pg_depend							7/38	AccessShareLock
relation	pg_depend_depender_index							7/38	AccessShareLock
relation	pg_attrdef_oid_index							7/38	AccessShareLock
relation	pg_index							7/38	AccessShareLock
relation	pg_namespace_nspname_index							7/38	AccessShareLock
relation	pg_namespace_oid_index							7/38	AccessShareLock
relation	pg_namespace							7/38	AccessShareLock
relation	pg_index_indrelid_index							7/38	AccessShareLock
relation	pg_constraint_oid_index							7/38	AccessShareLock
relation	pg_index_indexrelid_index							7/38	AccessShareLock
relation	pg_constraint_contypid_index							7/38	AccessShareLock
relation	pg_attrdef							7/38	AccessShareLock
relation	pg_attrdef_adrelid_adnum_index							7/38	AccessShareLock
relation	pg_constraint_conrelid_contypid_con…							7/38	AccessShareLock
relation	pg_depend_reference_index							7/38	AccessShareLock

Lock type	Target relation	Page	Tuple	vXID (target)	XID (target)	Class	Object ID	vXID (owner)	Mode	(
relation	pg_locks							5/1331	AccessShareLock	1
relation	pg_constraint_conname_nsp_index							7/38	AccessShareLock	1
relation	pg_constraint_conparentid_index							7/38	AccessShareLock	1
relation	pg_class_tblspc_relfilenode_index							7/38	AccessShareLock	1
relation	pg_class_relname_nsp_index							7/38	AccessShareLock	1
relation	pg_class_oid_index							7/38	AccessShareLock	1
relation	pg_constraint							7/38	AccessShareLock	1
relation	pg_class							7/38	AccessShareLock	1
relation	pg_type_typname_nsp_index							7/38	AccessShareLock	1
relation	pg_type_oid_index							7/38	AccessShareLock	1
relation	pg_attribute_relid_attnum_index							7/38	AccessShareLock	1
relation	pg_attribute_relid_attnam_index							7/38	AccessShareLock	1
relation	pg_type							7/38	AccessShareLock	1
relation	pg_attribute							7/38	AccessShareLock	1

به کمک کویری هم میشه این جدول را مشاهده کرد:

	locktype text	database oid	relation oid	page integer	tuple smallint	virtualxid text	transactionid xid	classid oid	objid oid	objsubid smallint	virtualtransaction text	pid integer	mode text	granted boolean
1	relation	18641	2665	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
2	relation	18641	2664	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
3	relation	18641	2579	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
4	relation	18641	3455	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
5	relation	18641	2663	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
6	relation	18641	2662	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
7	relation	18641	2606	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
8	relation	18641	1259	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
9	relation	18641	2704	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
10	relation	18641	2703	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true

Data Output Explain Messages Notifications

	locktype text	database oid	relation oid	page integer	tuple smallint	virtualxid text	transactionid xid	classid oid	objid oid	objsubid smallint	virtualtransaction text	pid integer	mode text	granted boolean
10	relation	18641	2703	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
11	relation	18641	2659	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
12	relation	18641	2658	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
13	relation	18641	1247	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
14	relation	18641	1249	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
15	relation	18641	19028	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
16	relation	18641	19028	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	RowE...	true
17	relation	18641	19025	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
18	relation	18641	19025	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	RowE...	true
19	virtualxid	[null]	[null]	[null]	[null]	7/38	[null]	[null]	[null]	[null]	7/38	19162	Exclu...	true

Data Output Explain Messages Notifications

	locktype text	database oid	relation oid	page integer	tuple smallint	virtualxid text	transactionid xid	classid oid	objid oid	objsubid smallint	virtualtransaction text	pid integer	mode text	granted boolean
21	relation	18641	2608	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
22	relation	0	2672	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
23	relation	18641	2673	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
24	relation	18641	2657	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
25	relation	18641	2610	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
26	relation	18641	2684	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
27	relation	18641	2685	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
28	relation	18641	12143	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
29	relation	18641	2615	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
30	relation	18641	2678	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true

Data Output | Explain | Messages | Notifications

	locktype text	database oid	relation oid	page integer	tuple smallint	virtualxid text	transactionid xid	classid oid	objid oid	objsubid smallint	virtualtransaction text	pid integer	mode text	granted boolean
33	relation	18641	2666	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
34	relation	18641	2604	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
35	transacti...	[null]	[null]	[null]	[null]	[null]	3773	[null]	[null]	[null]	7/38	19162	Exclu...	true
36	relation	18641	2656	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
37	relation	0	2676	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
38	relation	0	2671	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
39	relation	18641	12248	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
40	relation	0	1260	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
41	relation	0	1262	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true
42	relation	18641	2674	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/38	19162	Acces...	true

همان طور که از جدول میشه مشاهده کرد وقتی یک دستور begin را میزنیم یک سری لاگ نمایش داده میشود که مربوط به lock ها و اماده سازی ها برای انجام تراکنش است. و یک نوعی از lock داریم به نام transaction که یک ایدی و یک mode به نام ROW EXCLUSIVE دارد که

ROW EXCLUSIVE

Conflicts with the SHARE, SHARE ROW EXCLUSIVE, EXCLUSIVE, and ACCESS EXCLUSIVE lock modes.

The commands UPDATE, DELETE, and INSERT acquire this lock mode on the target table (in addition to ACCESS SHARE locks on any other referenced tables). In general, this lock mode will be acquired by any command that modifies data in a table.

بقیه ی lock ها هم از نوع Access share هستند

ACCESS SHARE

Conflicts with the ACCESS EXCLUSIVE lock mode only.

The SELECT command acquires a lock of this mode on referenced tables. In general, any query that only reads a table and does not modify it will acquire this lock mode.

Query Editor Query History

```
begin;
-- update accounts
-- set balance = balance -150
-- where name ='Ali';

-- update accounts
--  set balance = balance +150
--  where name ='Amir';
rollback;

select id,name,balance
```

Data Output Explain Messages Notifications

```
ROLLBACK

Query returned successfully in 34 msec.
```

بعد از اجرای دستور rollback

```
select * from pg_locks ;
```

Data Output Explain Messages Notifications

	locktype text	database oid	relation oid	page integer	tuple smallint	virtualxid text	transactionid xid	classid oid	objid oid	objsubid smallint	virtualtransaction text	pid integer	mode text	granted boolean	fa bo
1	relation	18641	12143	[null]	[null]	[null]	[null]	[null]	[null]	[null]	7/51	19162	Acces...	true	tru
2	virtualxid	[null]	[null]	[null]	[null]	7/51	[null]	[null]	[null]	[null]	7/51	19162	Exclu...	true	tru

همه ی لاگ های مربوط به تراکنش پاک میشوند و فقط virtualxid و خود relation اصلی میمانند.

دلیل تغییرات لاک به این دلیل است که دیگر هم زمانی نداریم که دیتابیس بخاهد از یک lock استفاده کند چون وقتی یک تراکنش داریم برای جلوگیری از data inconsistency و ایجاد بن بست و ... باید از یک متغیر به نام lock استفاده کنیم .وقتی که تراکنش را به جای اولیش برمیگردانیم دیگر نیازی نیست این متغیرها نگه داشته شوند وگزارش شوند.

سوال 9

Query Editor Query History

```
create or replace function showCustomers(region varchar(20))
returns table(first_name varchar(20), last_name varchar(20), title varchar(100))
language plpgsql
as
$$
begin
return query
select c1.first_name,c1.last_name,f1.title
from customer c1 inner join rental using(customer_id)
inner join inventory using(inventory_id)
inner join film f1 using(film_id)
```

Data Output Explain Messages Notifications

```
CREATE FUNCTION

Query returned successfully in 343 msec.
```

Query Editor Query History

```
-- where rental_date = (
-- select max(rental_date)
-- from customer c2 inner join rental using(customer_id)
-- where c1.customer_id = c2.customer_id
-- group by customer_id
-- )
--  and address.district = showCustomers.region;
-- end;
-- $$

 select * from showCustomers('Texas')
```

Data Output Explain Messages Notifications

	first_name character varying	last_name character varying	title character varying
1	Jennifer	Davis	Masked Bubble
2	Kim	Cruz	Rouge Squad
3	Richard	Mccrary	Rocketeer Mother
4	Bryan	Hardison	Strictly Scarface
5	Ian	Still	Mockingbird Hollywood

Query Editor Query History

```
-- )
--  and address.district = showCustomers.region;
-- end;
-- $$
-- select * from showCustomers('Texas')

 select * from showCustomers('Kaduna');
```

Data Output Explain Messages Notifications

	first_name character varying	last_name character varying	title character varying
1	Carol	Garcia	Banger Pinocchio
2	Constance	Reid	Glory Tracy

Query Editor Query History

```
-- )
--  and address.district = showCustomers.region;
-- end;
-- $$
-- select * from showCustomers('Texas')

 select * from showCustomers('Alberta');
```

Data Output Explain Messages Notifications

	first_name character varying	last_name character varying	title character varying

سوال 10

```
select title,rating from film
where title in ('Airport Pollock','Bright Encounters');
```

Data Output Explain Messages Notifications

	title character varying (255)	rating mpaa_rating
1	Airport Pollock	R
2	Bright Encounters	PG-13

dvdrental/postgres@localhost

Query Editor Query History

```
create or replace procedure switch_rating(
    film1 varchar,
    film2 varchar
)
language plpgsql
as $$
declare
     temp_rating1 mpaa_rating;
    temp_rating2 mpaa_rating;
begin

```

Data Output Explain Messages Notifications

```
CREATE PROCEDURE

Query returned successfully in 76 msec.
```

dvdrental/postgres@localhost

Query Editor Query History

```
-- end;$$


call switch_rating('Airport Pollock','Bright Encounters');


-- select title,rating from film
-- where title in ('Airport Pollock','Bright Encounters');


```

Data Output Explain Messages Notifications

```
CALL

Query returned successfully in 66 msec.
```

بعد از اجرای call

dvdrental/postgres@localhost

Query Editor Query History

```
-- end;$$


-- call switch_rating('Airport Pollock','Bright Encounters');


 select title,rating from film
where title in ('Airport Pollock','Bright Encounters');
```

Data Output Explain Messages Notifications

	title character varying (255)	rating mpaa_rating
1	Airport Pollock	PG-13
2	Bright Encounters	R