cyan audit

Forensic Audit Logging for PostgreSQL

Moshe Jacobson

http://cyanaudit.neadwerx.com

The Situation

- Data is mysteriously wrong/missing
- Legal is asking for records

- Who, when, how?
- How to respond?

- CYA with proof!

Application-Level Logging

- Explicit
- Tedious
- Easy to miss something
- Not always consistent
- Increases development time

- Better alternative?

Database-Level Logging

- pg_audit https://github.com/jcasanov/pg_audit
- pgtrail http://code.google.com/p/pgtrail/
- tablelog http://pgfoundry.org/projects/tablelog/
- Audit trigger 91plus
  http://wiki.postgresql.org/wiki/Audit_trigger_91plus
- Half-baked home-grown solutions?
- I wanted something better.

Our Application

- 80,000 users
- 1TB database
- 450 tables, 3200 columns
- 14 million daily page requests
- 8.5 million daily database updates
- 99.999% uptime SLA

Wishlist

- Extension-based
- Space-efficient, organized logging
- Per-column control of logging
- Attach descriptions to events
- Scalability to years’ worth of logs
- Export / import between log & files
- Automated log maintenance
- Easy recovery from mistakes

Cyan Audit - Logged Data

- Timestamp
- Name of table & column modified
- Integer PK of row modified
  - You do have integer surrogate PKs, right??
- Application-level userid of responsible user
- Transaction ID
- Application-supplied description
- Operation type ('I', 'U', 'D')
- Old and new values (stored as text)

Installation – Part I

- Unpack extension tarball, “make install”
- Configure custom_variable_classes in postgresql.conf (9.1 only):

```
custom_variable_classes = 'cyanaudit'
```

- Create extension

```
db=# create schema cyanaudit;
db=# create extension cyanaudit schema cyanaudit;
```

- Set up logging triggers

```
db=# select cyanaudit.fn_update_audit_fields();
```

- Now you’re logging!

Installation – Part II

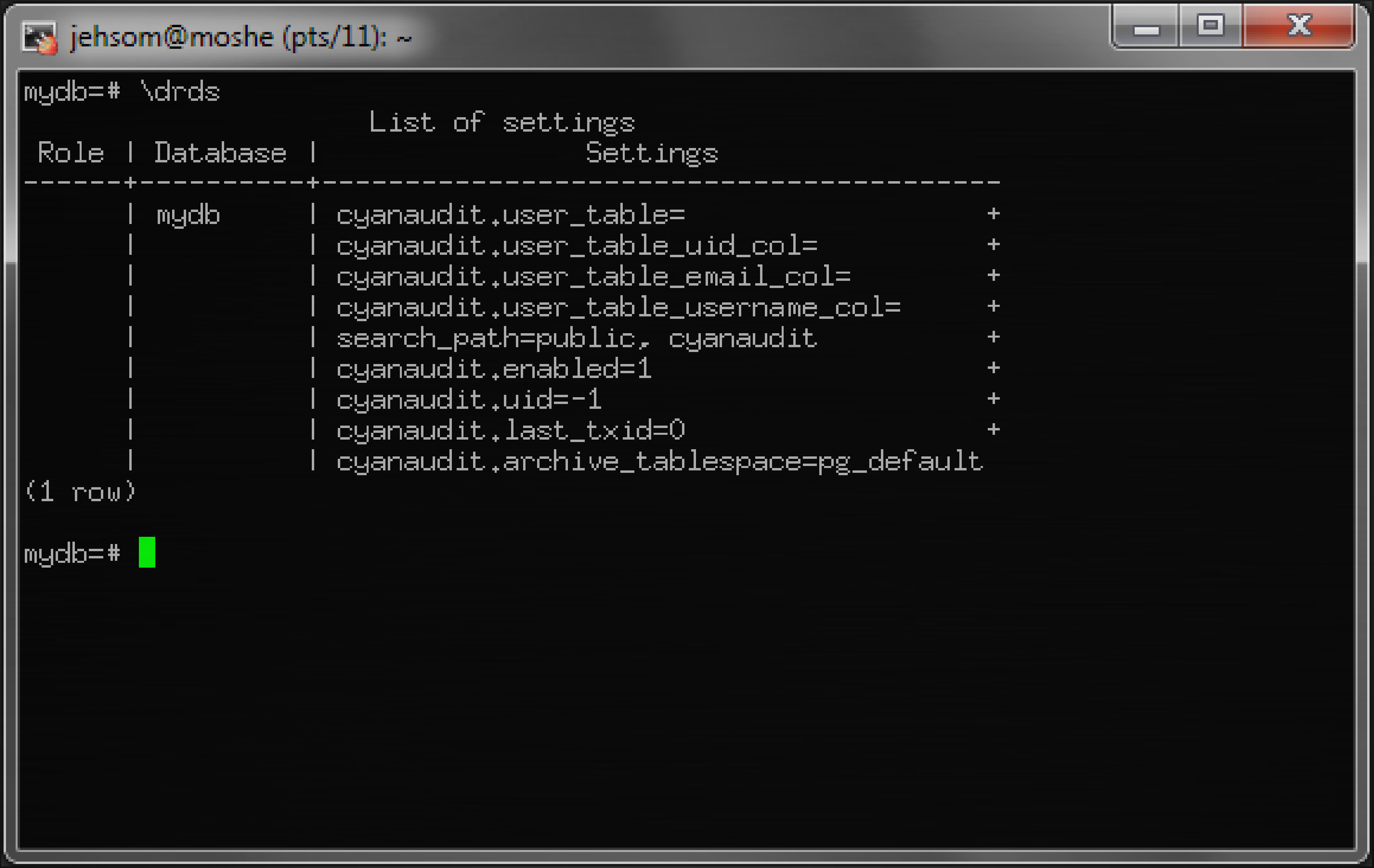
- Install cron jobs to rotate and archive logs
- Set your database-specific settings

```
alter database mydb
    set cyanaudit.archive_tablespace = 'big_slow_drive';
... set cyanaudit.user_table = 'users';
... set cyanaudit.user_table_uid_col = 'entity';
... set cyanaudit.user_table_username_col = 'username';
... set cyanaudit.user_table_email_col = 'email_address';
```

- Add cyanaudit schema to database search path

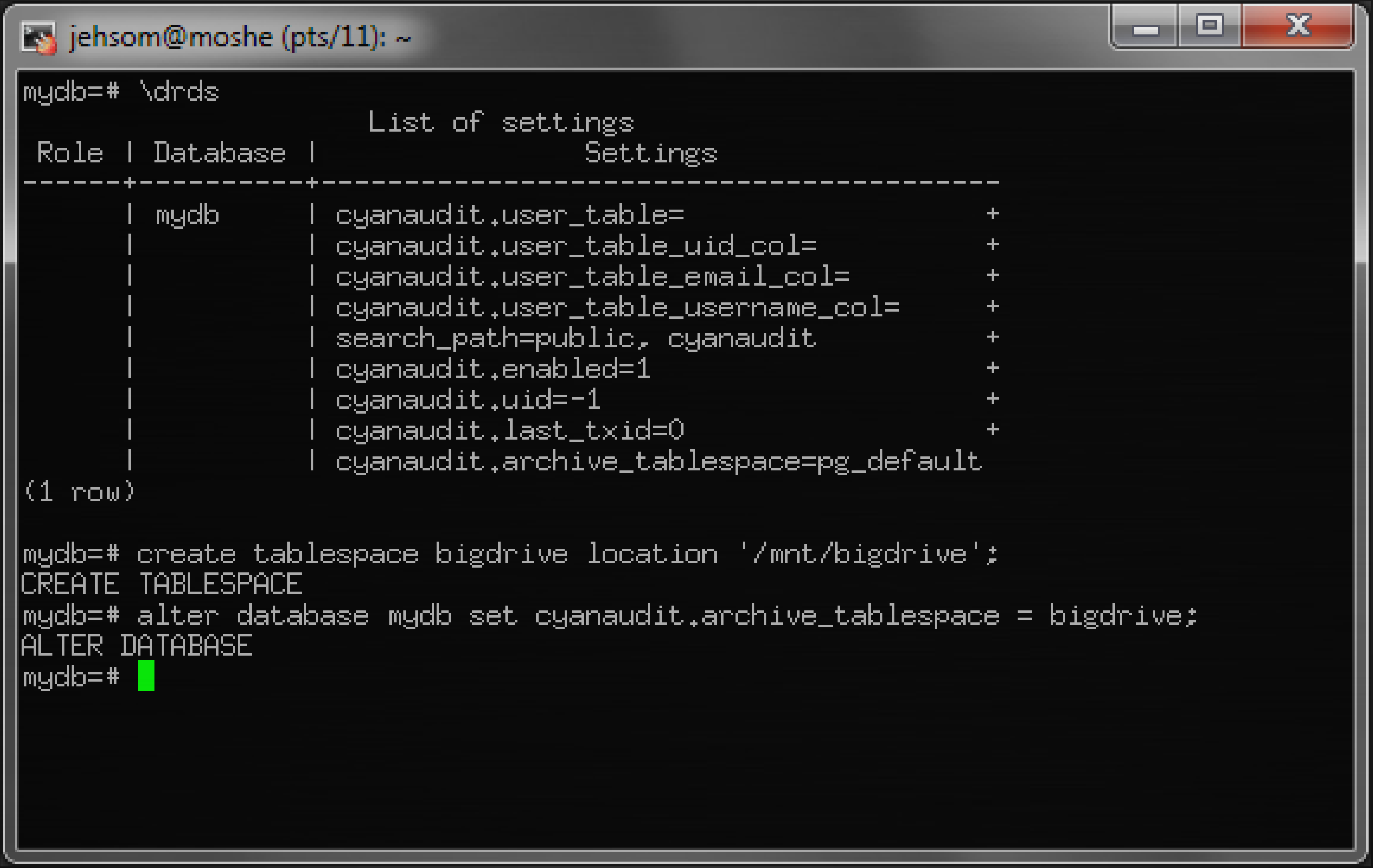
```
alter database mydb
    set search_path = public, cyanaudit;
```

Post-installation



```
mydb=# \drds
                     List of settings
 Role | Database |                Settings
------+----------+----------------------------------------
      | mydb     | cyanaudit.user_table=                   +
      |          | cyanaudit.user_table_uid_col=           +
      |          | cyanaudit.user_table_email_col=         +
      |          | cyanaudit.user_table_username_col=      +
      |          | search_path=public, cyanaudit           +
      |          | cyanaudit.enabled=1                     +
      |          | cyanaudit.uid=-1                        +
      |          | cyanaudit.last_txid=0                   +
      |          | cyanaudit.archive_tablespace=pg_default
(1 row)

mydb=#
```

Post-installation

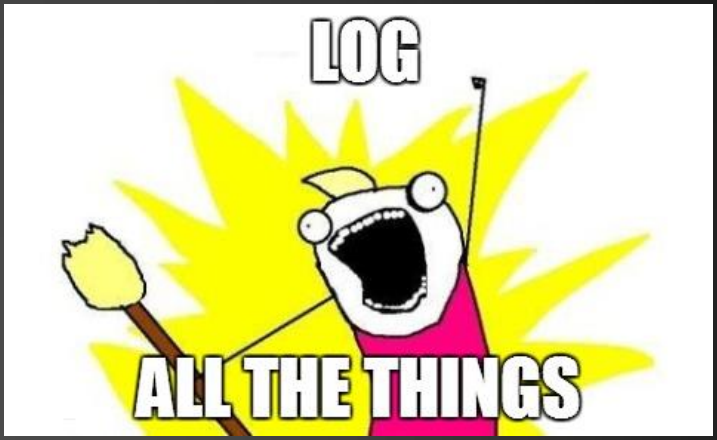


jehsom@moshe (pts/11): ~

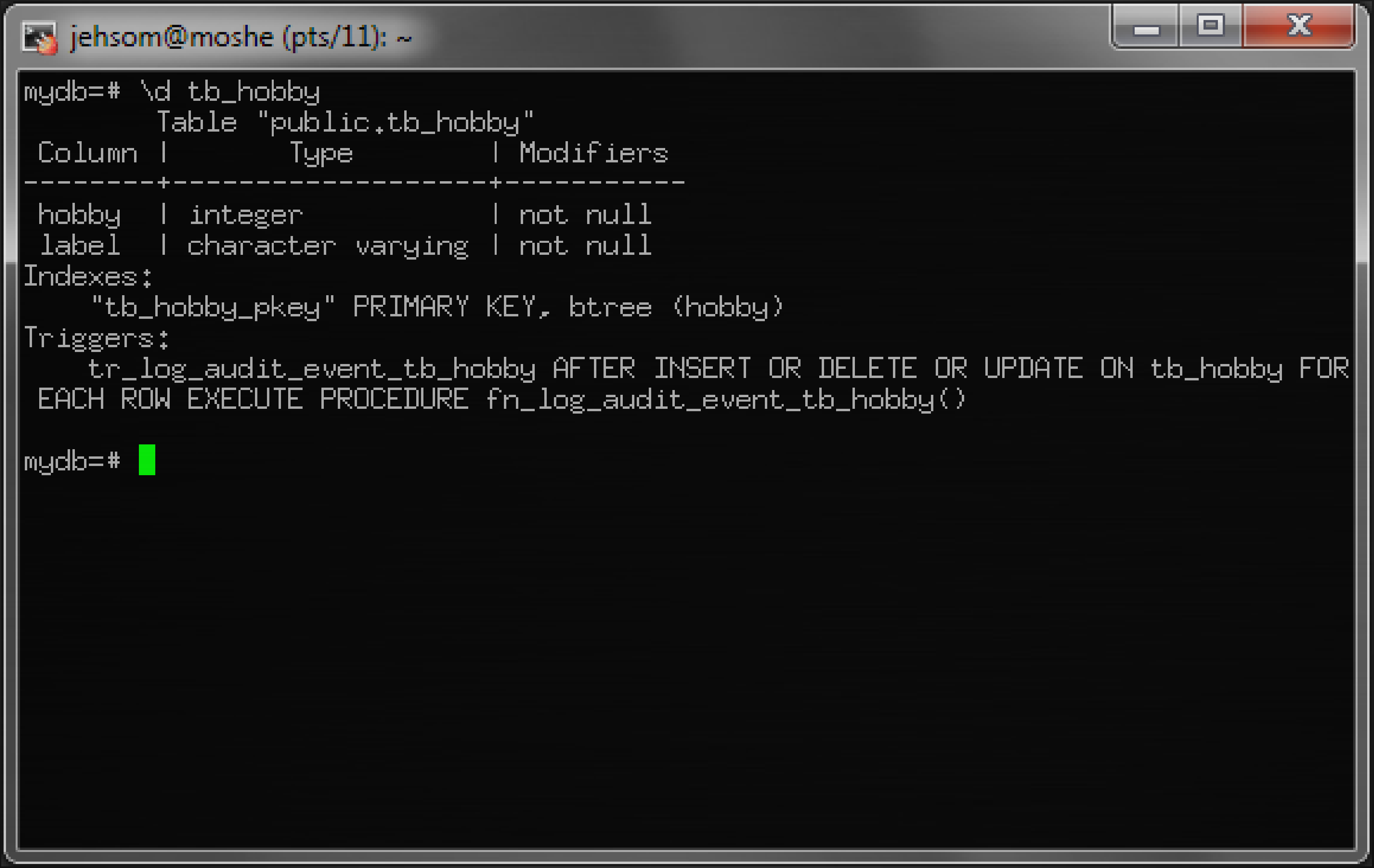
```
mydb=# \drds
                    List of settings
 Role | Database |                 Settings
------+----------+----------------------------------------
      | mydb     | cyanaudit.user_table=                 +
      |          | cyanaudit.user_table_uid_col=         +
      |          | cyanaudit.user_table_email_col=       +
      |          | cyanaudit.user_table_username_col=    +
      |          | search_path=public, cyanaudit         +
      |          | cyanaudit.enabled=1                   +
      |          | cyanaudit.uid=-1                      +
      |          | cyanaudit.last_txid=0                 +
      |          | cyanaudit.archive_tablespace=pg_default
(1 row)

mydb=# create tablespace bigdrive location '/mnt/bigdrive';
CREATE TABLESPACE
mydb=# alter database mydb set cyanaudit.archive_tablespace = bigdrive;
ALTER DATABASE
mydb=# 
```

Selecting what to log

- Upon installation, <u>all</u> fields are enabled
- Consider high traffic fields
- `tb_audit_field` has one row per table/column
- "`active`" boolean controls logging for a column
- `select fn_update_audit_fields()` reindexes fields after DDL
- Disable logging for a session: `set cyanaudit.enabled = 0`

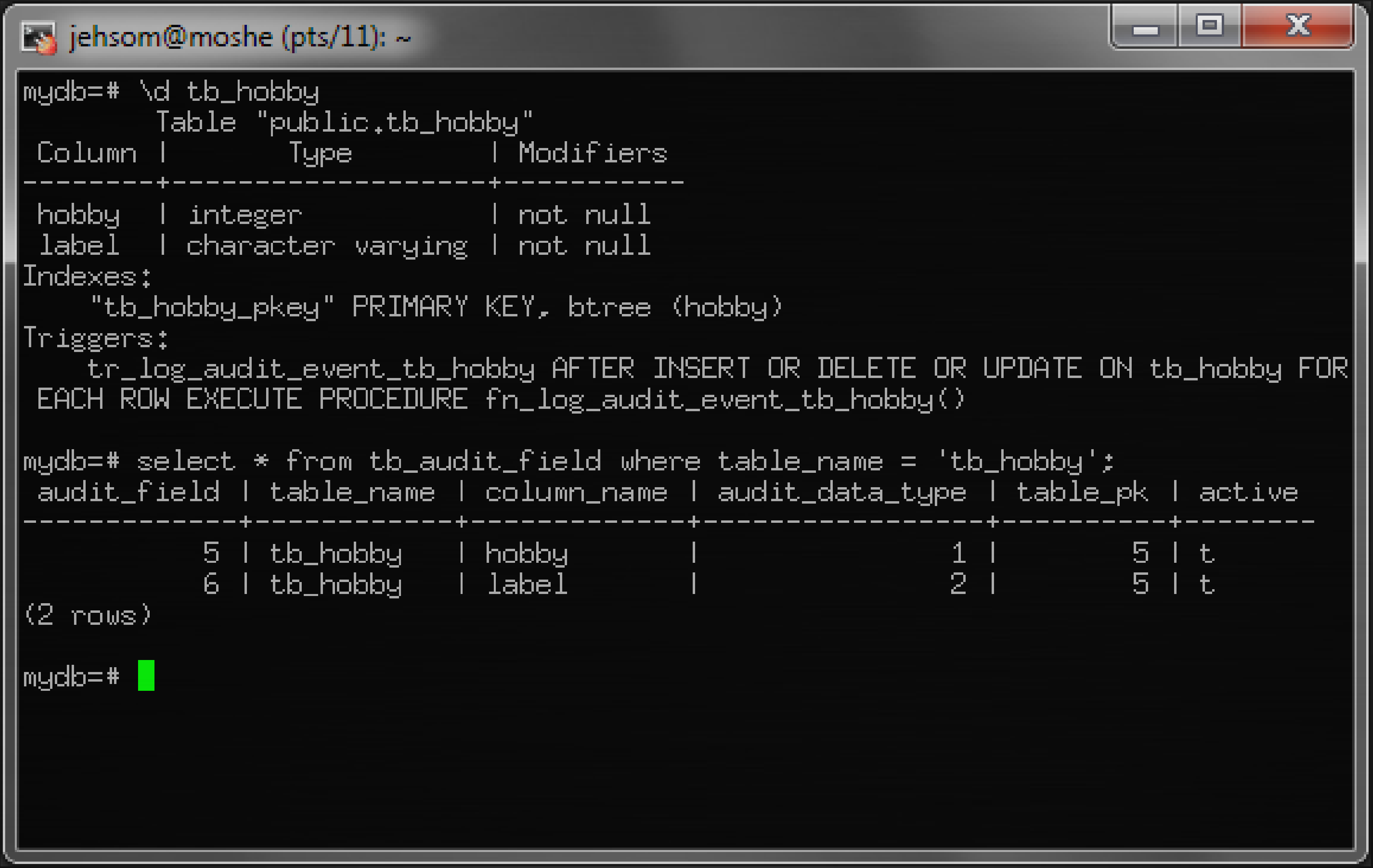


Selecting what to log


jehsom@moshe (pts/11): ~

```
mydb=# \d tb_hobby
         Table "public.tb_hobby"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 hobby  | integer           | not null
 label  | character varying | not null
Indexes:
    "tb_hobby_pkey" PRIMARY KEY, btree (hobby)
Triggers:
    tr_log_audit_event_tb_hobby AFTER INSERT OR DELETE OR UPDATE ON tb_hobby FOR
 EACH ROW EXECUTE PROCEDURE fn_log_audit_event_tb_hobby()

mydb=# 
```

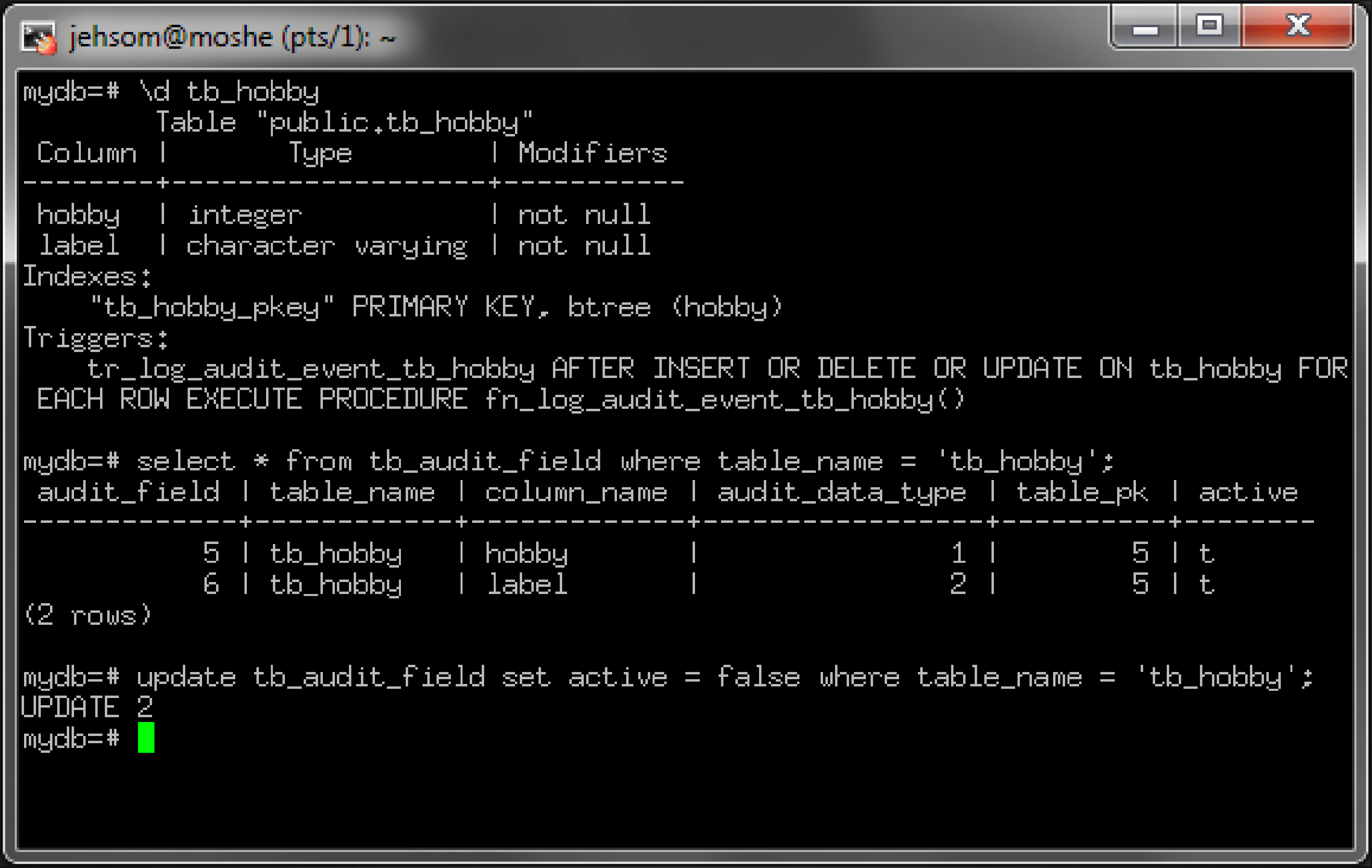
Selecting what to log



jehsom@moshe (pts/11): ~

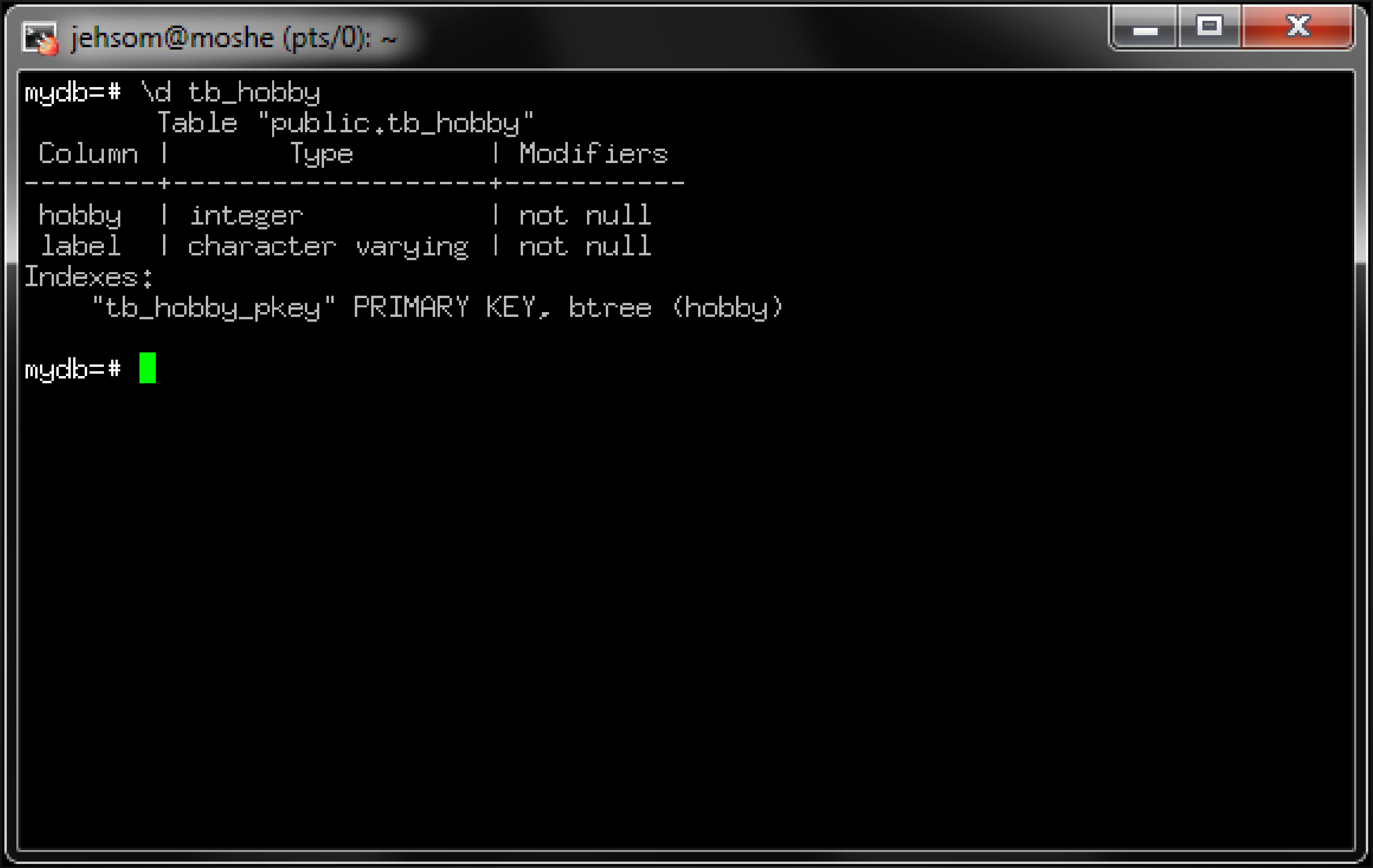
```
mydb=# \d tb_hobby
        Table "public.tb_hobby"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 hobby  | integer           | not null
 label  | character varying | not null
Indexes:
    "tb_hobby_pkey" PRIMARY KEY, btree (hobby)
Triggers:
    tr_log_audit_event_tb_hobby AFTER INSERT OR DELETE OR UPDATE ON tb_hobby FOR
 EACH ROW EXECUTE PROCEDURE fn_log_audit_event_tb_hobby()

mydb=# select * from tb_audit_field where table_name = 'tb_hobby';
 audit_field | table_name | column_name | audit_data_type | table_pk | active
-------------+------------+-------------+-----------------+----------+--------
           5 | tb_hobby   | hobby       |               1 |        5 | t
           6 | tb_hobby   | label       |               2 |        5 | t
(2 rows)

mydb=# 
```

Selecting what to log



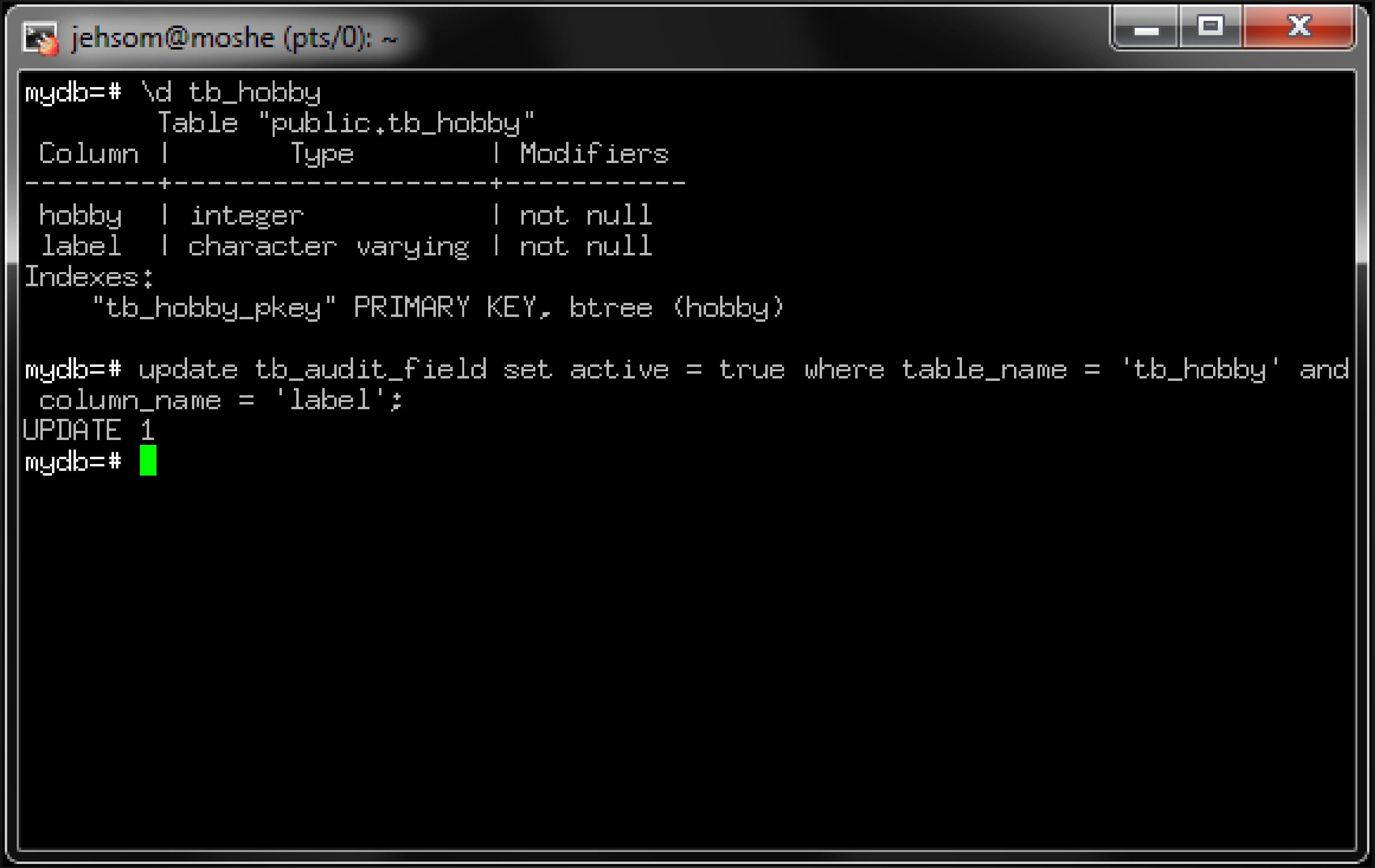
```
mydb=# \d tb_hobby
        Table "public.tb_hobby"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 hobby  | integer           | not null
 label  | character varying | not null
Indexes:
    "tb_hobby_pkey" PRIMARY KEY, btree (hobby)
Triggers:
    tr_log_audit_event_tb_hobby AFTER INSERT OR DELETE OR UPDATE ON tb_hobby FOR
 EACH ROW EXECUTE PROCEDURE fn_log_audit_event_tb_hobby()

mydb=# select * from tb_audit_field where table_name = 'tb_hobby';
 audit_field | table_name | column_name | audit_data_type | table_pk | active
-------------+------------+-------------+-----------------+----------+--------
           5 | tb_hobby   | hobby       |               1 |        5 | t
           6 | tb_hobby   | label       |               2 |        5 | t
(2 rows)

mydb=# update tb_audit_field set active = false where table_name = 'tb_hobby';
UPDATE 2
mydb=#
```

Selecting what to log


jehsom@moshe (pts/0): ~

```
mydb=# \d tb_hobby
         Table "public.tb_hobby"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 hobby  | integer           | not null
 label  | character varying | not null
Indexes:
    "tb_hobby_pkey" PRIMARY KEY, btree (hobby)

mydb=# 
```

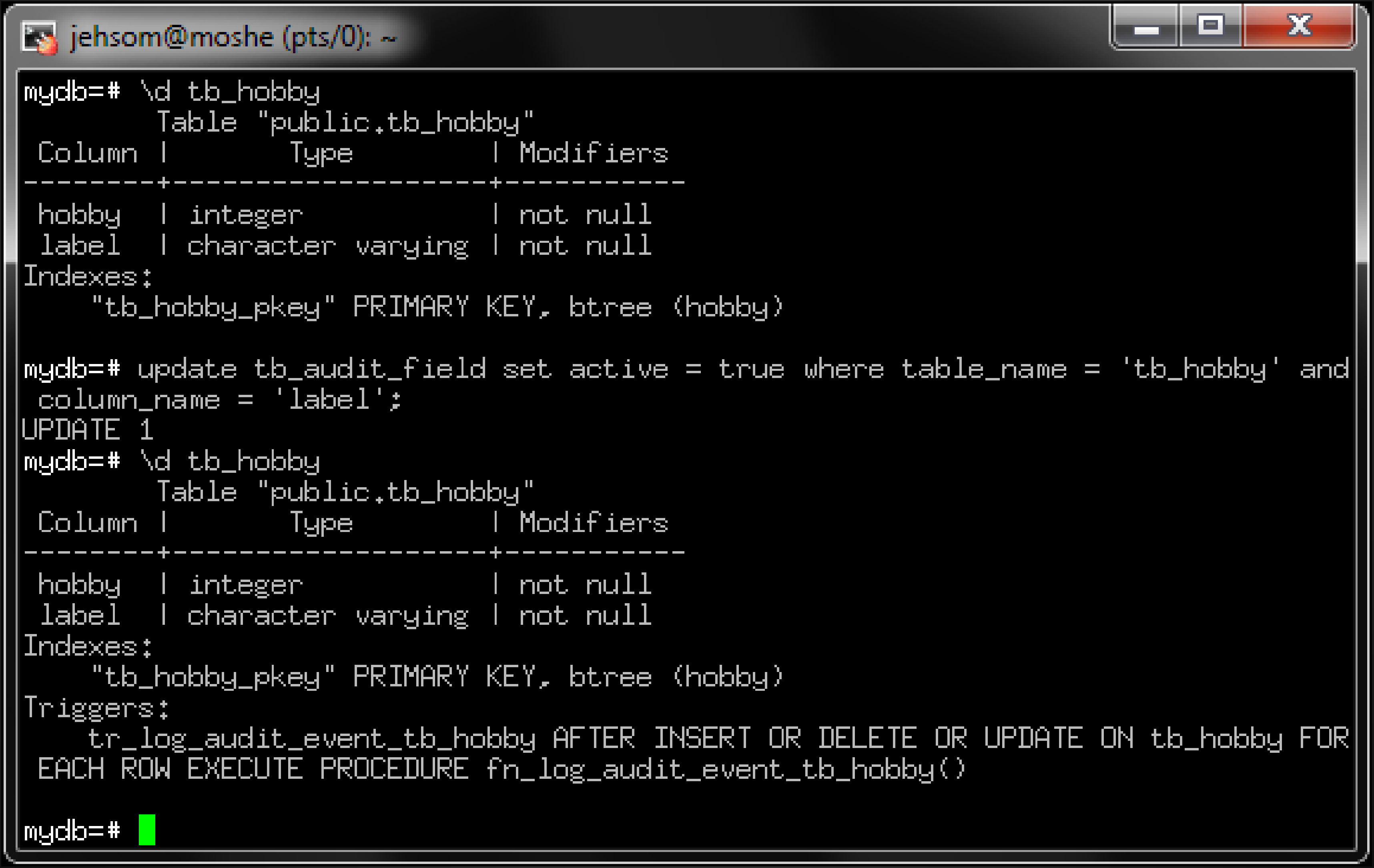
Selecting what to log



jehsom@moshe (pts/0): ~

```
mydb=# \d tb_hobby
         Table "public.tb_hobby"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 hobby  | integer           | not null
 label  | character varying | not null
Indexes:
    "tb_hobby_pkey" PRIMARY KEY, btree (hobby)

mydb=# update tb_audit_field set active = true where table_name = 'tb_hobby' and
 column_name = 'label';
UPDATE 1
mydb=# 
```

Selecting what to log



```
mydb=# \d tb_hobby
         Table "public.tb_hobby"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 hobby  | integer           | not null
 label  | character varying | not null
Indexes:
    "tb_hobby_pkey" PRIMARY KEY, btree (hobby)

mydb=# update tb_audit_field set active = true where table_name = 'tb_hobby' and
 column_name = 'label';
UPDATE 1
mydb=# \d tb_hobby
         Table "public.tb_hobby"
 Column |       Type        | Modifiers
--------+-------------------+-----------
 hobby  | integer           | not null
 label  | character varying | not null
Indexes:
    "tb_hobby_pkey" PRIMARY KEY, btree (hobby)
Triggers:
    tr_log_audit_event_tb_hobby AFTER INSERT OR DELETE OR UPDATE ON tb_hobby FOR
 EACH ROW EXECUTE PROCEDURE fn_log_audit_event_tb_hobby()

mydb=# 
```

Querying the audit log

View: `vw_audit_log`

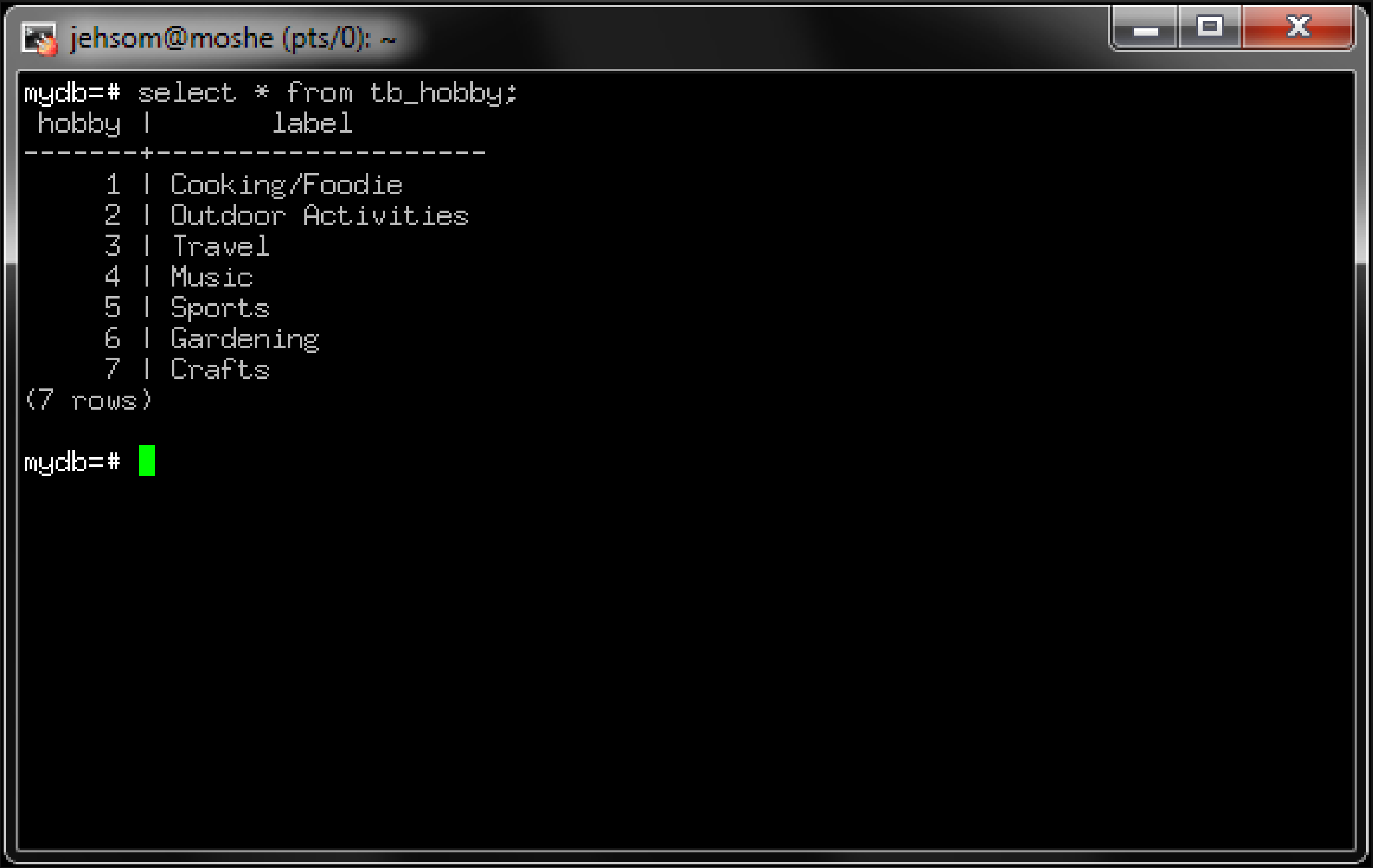
- Columns:

  `recorded | uid | user_email | txid | description | table_name | column_name | pk_val | op | old_value | new_value`

- Millions of rows accumulate quickly
  - Especially when you’re doing admin work and forget to turn off logging…
- Use indexed columns when querying:

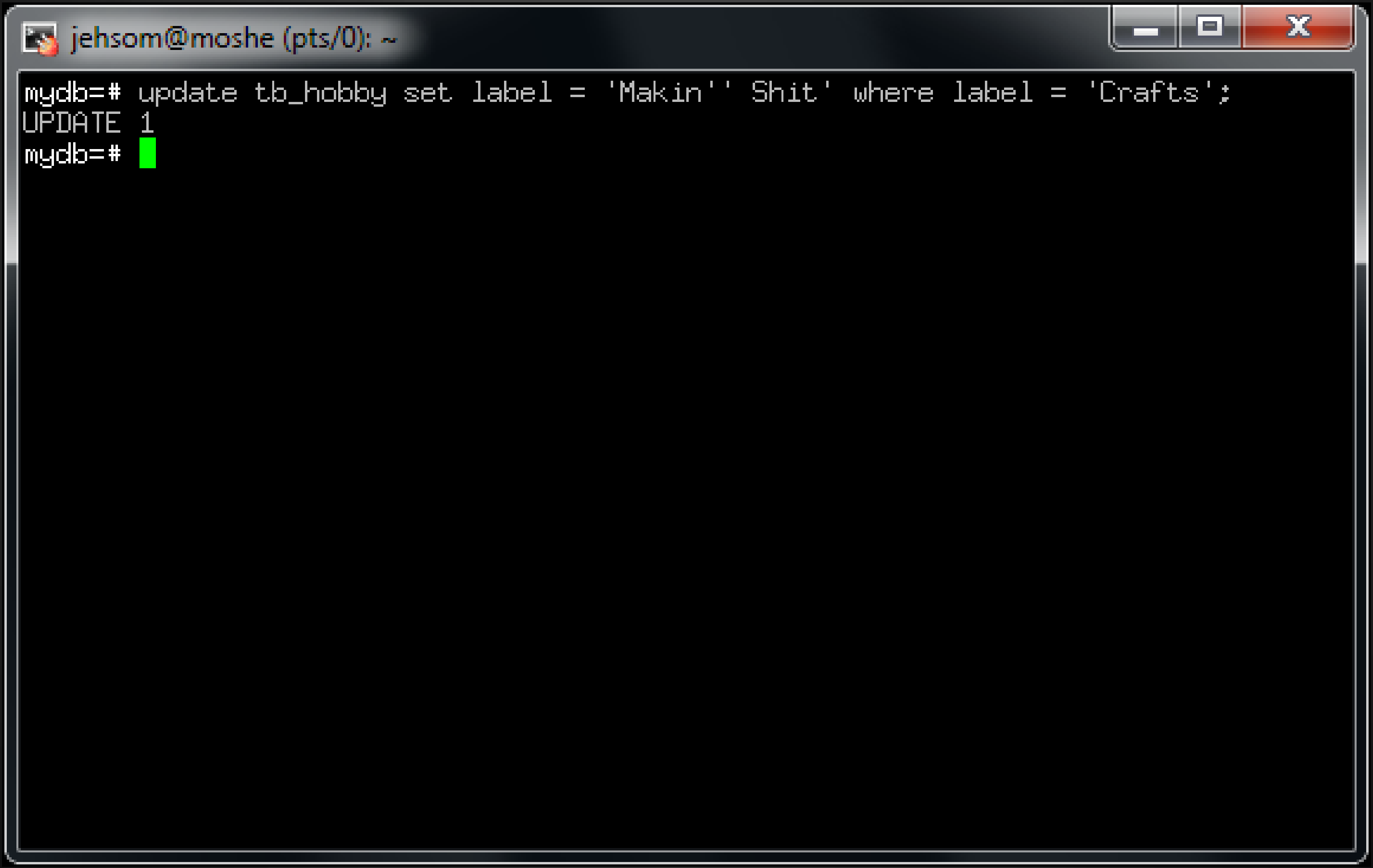
  `recorded, table_name + column_name, txid`

Example



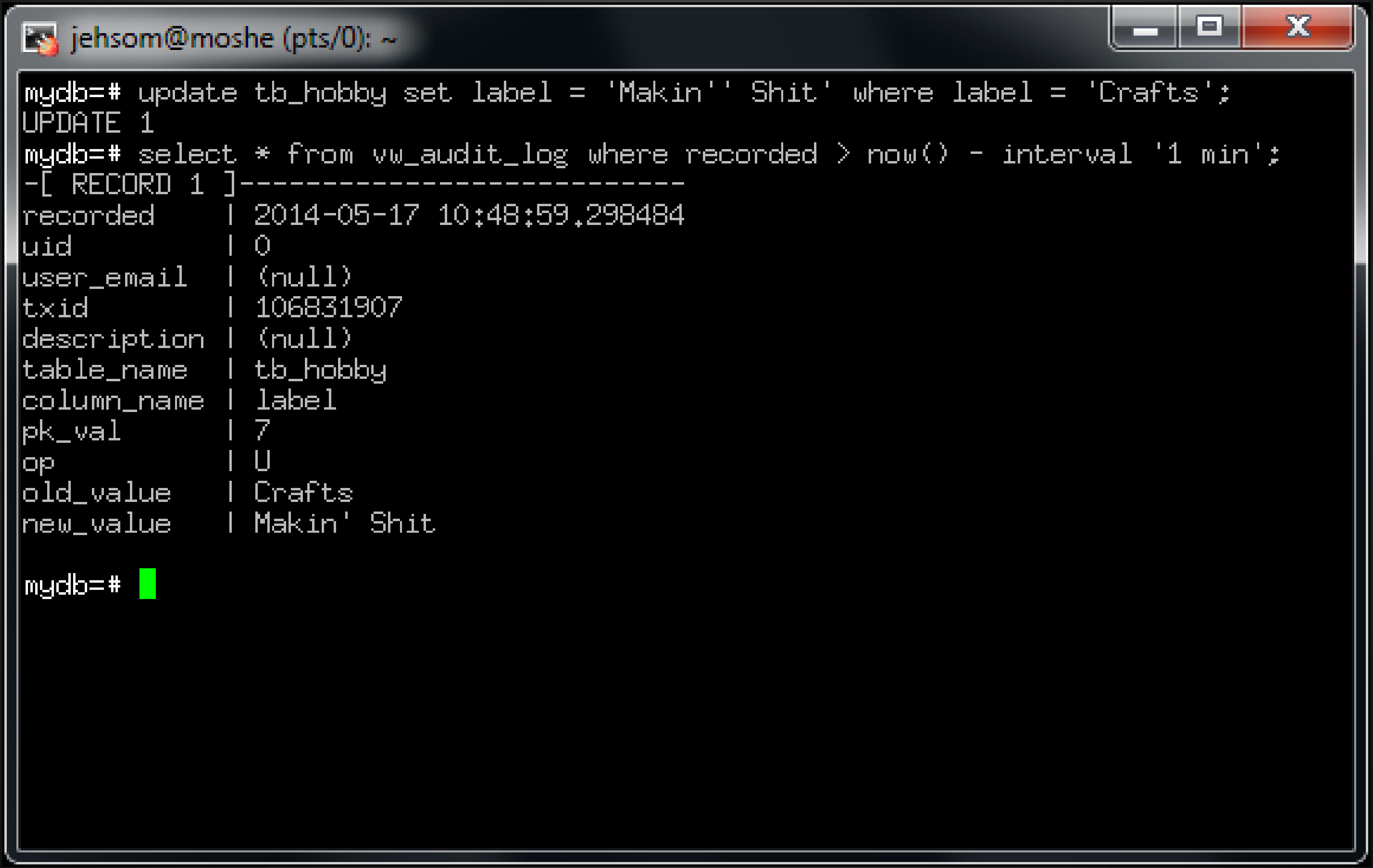
```
mydb=# select * from tb_hobby;
 hobby |        label
-------+---------------------
     1 | Cooking/Foodie
     2 | Outdoor Activities
     3 | Travel
     4 | Music
     5 | Sports
     6 | Gardening
     7 | Crafts
(7 rows)

mydb=# 
```

Example



```
mydb=# update tb_hobby set label = 'Makin'' Shit' where label = 'Crafts';
UPDATE 1
mydb=#
```

Example



```
mydb=# update tb_hobby set label = 'Makin'' Shit' where label = 'Crafts';
UPDATE 1
mydb=# select * from vw_audit_log where recorded > now() - interval '1 min';
-[ RECORD 1 ]---------------------------
recorded    | 2014-05-17 10:48:59.298484
uid         | 0
user_email  | (null)
txid        | 106831907
description | (null)
table_name  | tb_hobby
column_name | label
pk_val      | 7
op          | U
old_value   | Crafts
new_value   | Makin' Shit

mydb=# 
```

Reconstructing Queries

View:

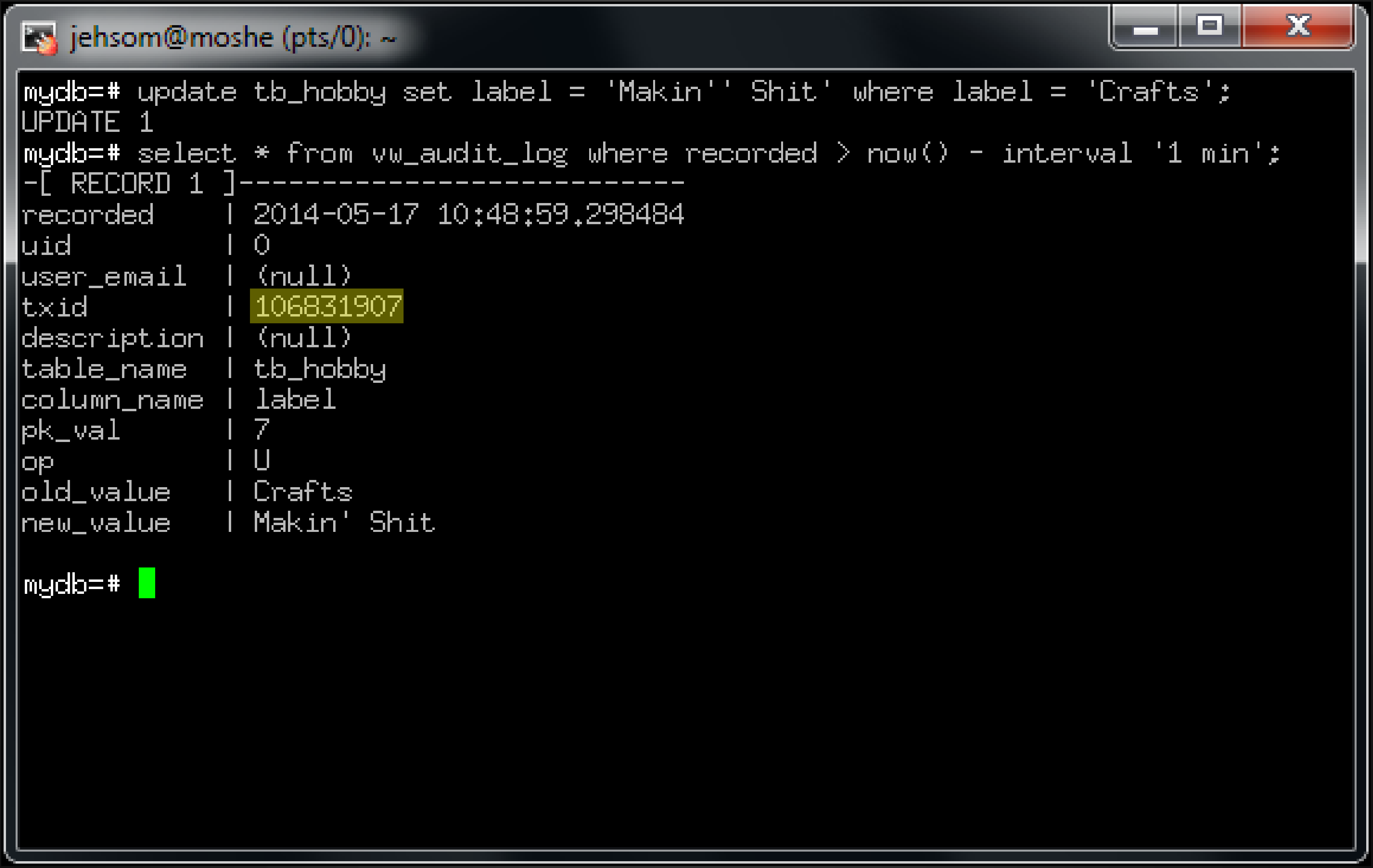
`vw_audit_transaction_statement`

Reconstructs queries effectively equivalent to original DML

Columns:

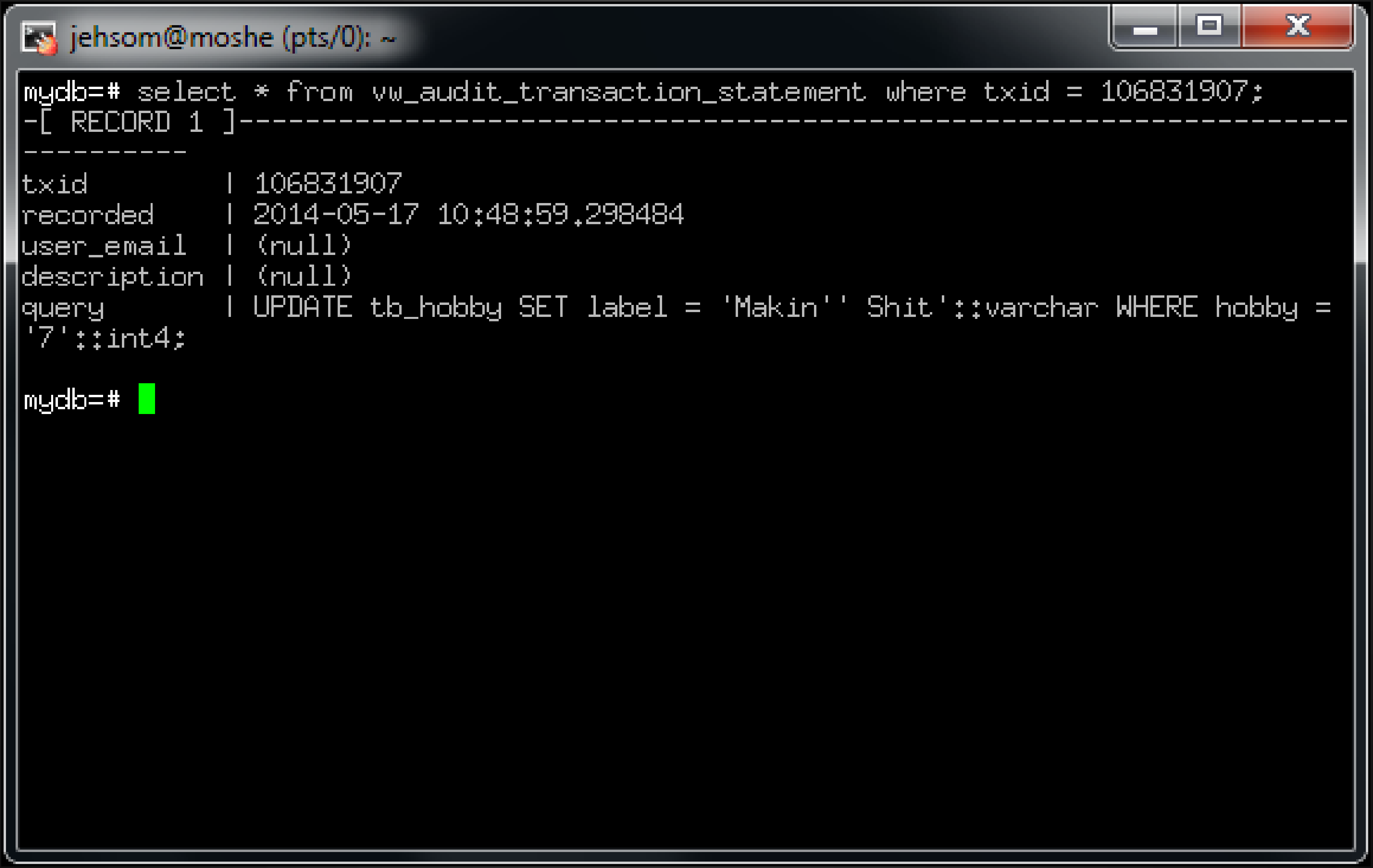
```
txid | recorded | email | description | query
```

Reconstructing Queries



```
mydb=# update tb_hobby set label = 'Makin'' Shit' where label = 'Crafts';
UPDATE 1
mydb=# select * from vw_audit_log where recorded > now() - interval '1 min';
-[ RECORD 1 ]---------------------------
recorded    | 2014-05-17 10:48:59.298484
uid         | 0
user_email  | (null)
txid        | 106831907
description | (null)
table_name  | tb_hobby
column_name | label
pk_val      | 7
op          | U
old_value   | Crafts
new_value   | Makin' Shit

mydb=#
```

Reconstructing Queries



jehsom@moshe (pts/0): ~

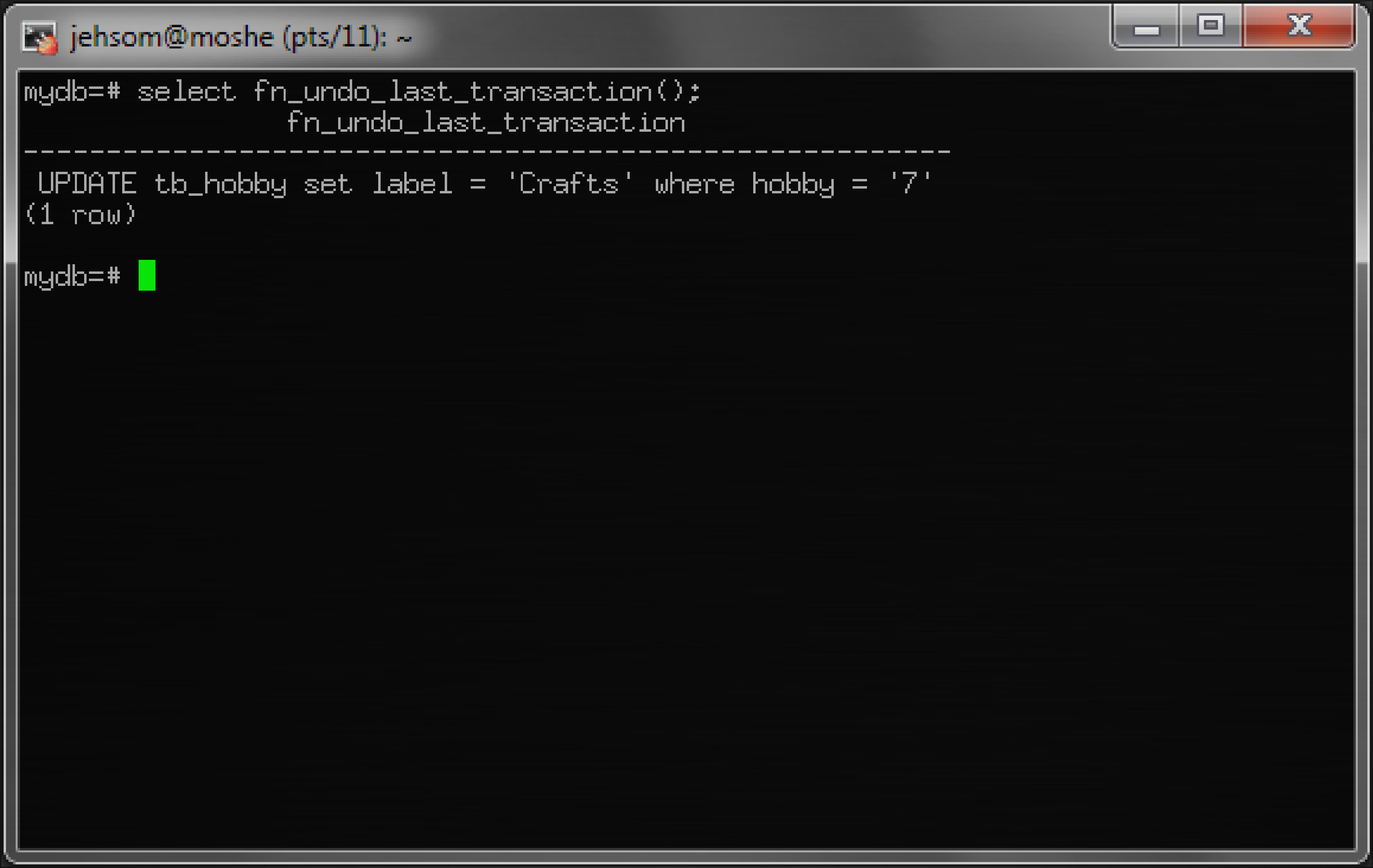
```
mydb=# select * from vw_audit_transaction_statement where txid = 106831907;
-[ RECORD 1 ]----------------------------------------------------------------
----------
txid        | 106831907
recorded    | 2014-05-17 10:48:59.298484
user_email  | (null)
description | (null)
query       | UPDATE tb_hobby SET label = 'Makin'' Shit'::varchar WHERE hobby =
'7'::int4;

mydb=#
```

When You F*** Up…

- We can reconstruct queries…
  Why not reverse them?

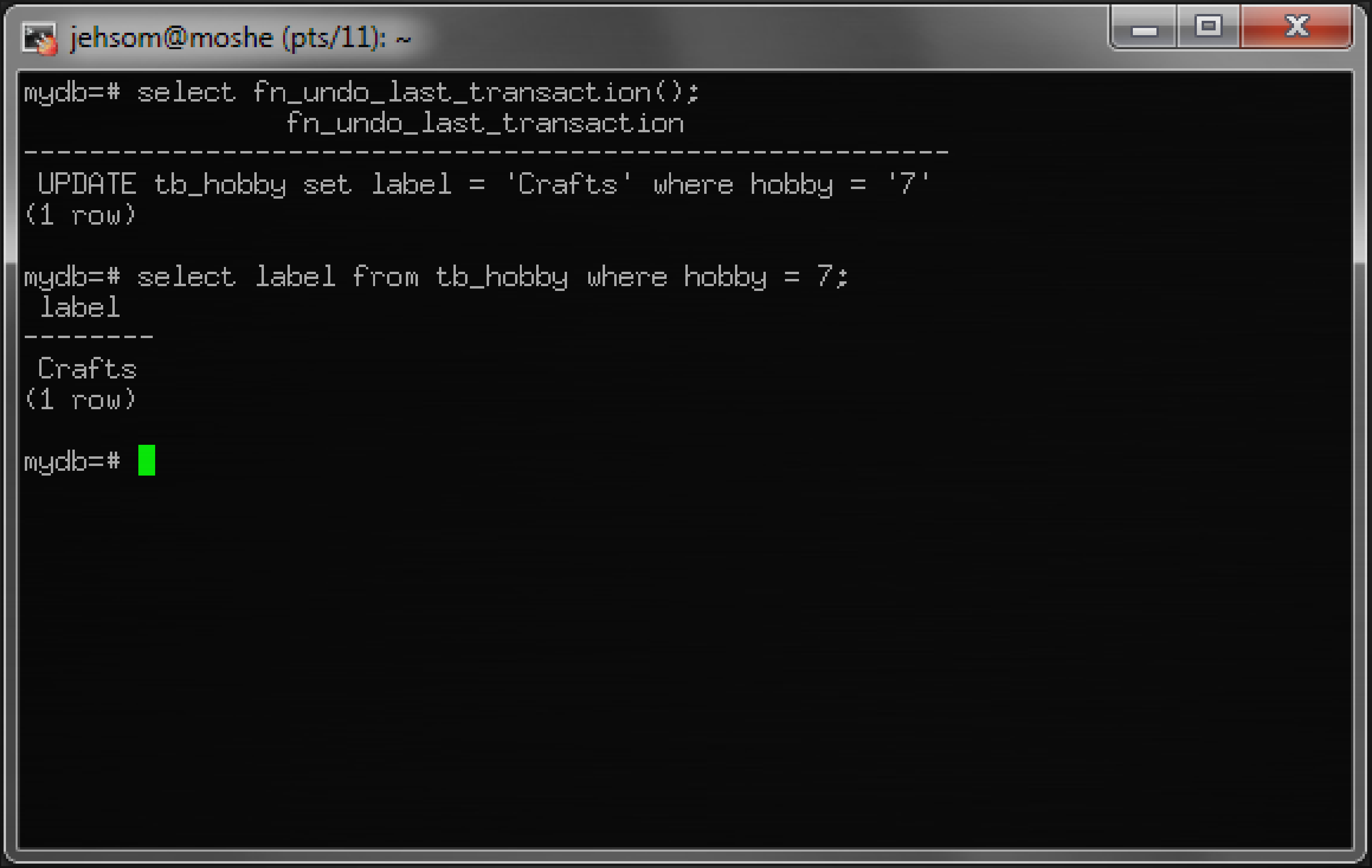
- `fn_undo_transaction(txid)`
  Undoes recorded changes for txid
- `fn_get_last_audit_txid()`
  Gives txid of last logged transaction
- `select fn_undo_last_transaction()`
  Combines two functions above.

When You F*** Up



```
mydb=# select fn_undo_last_transaction();
                 fn_undo_last_transaction
-------------------------------------------------------------
 UPDATE tb_hobby set label = 'Crafts' where hobby = '7'
(1 row)

mydb=# 
```

When You F*** Up

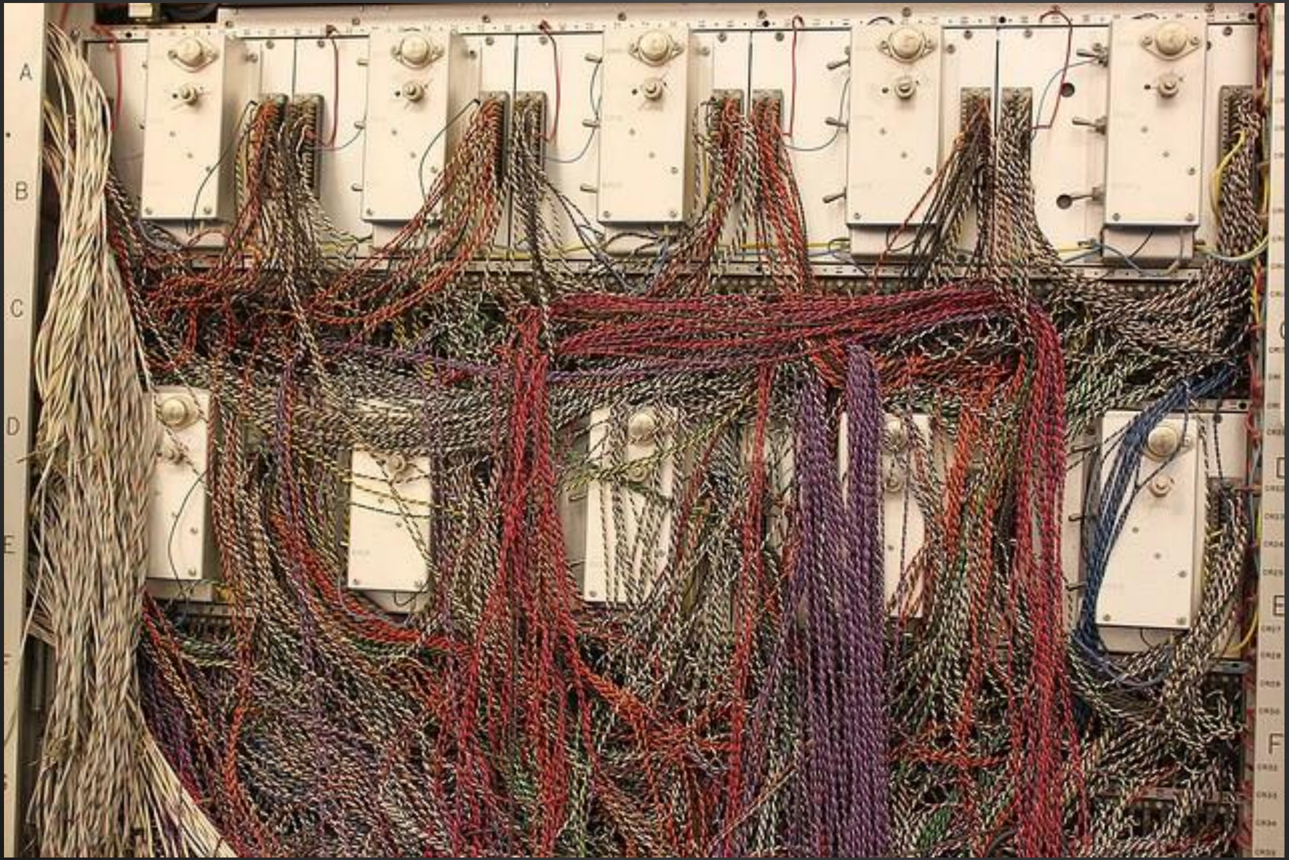


jehsom@moshe (pts/11): ~

```
mydb=# select fn_undo_last_transaction();
                  fn_undo_last_transaction
-----------------------------------------------------------------
 UPDATE tb_hobby set label = 'Crafts' where hobby = '7'
(1 row)

mydb=# select label from tb_hobby where hobby = 7;
 label
---------
 Crafts
(1 row)

mydb=# 
```

Application Integration

How DBAs see application code:



Application Integration

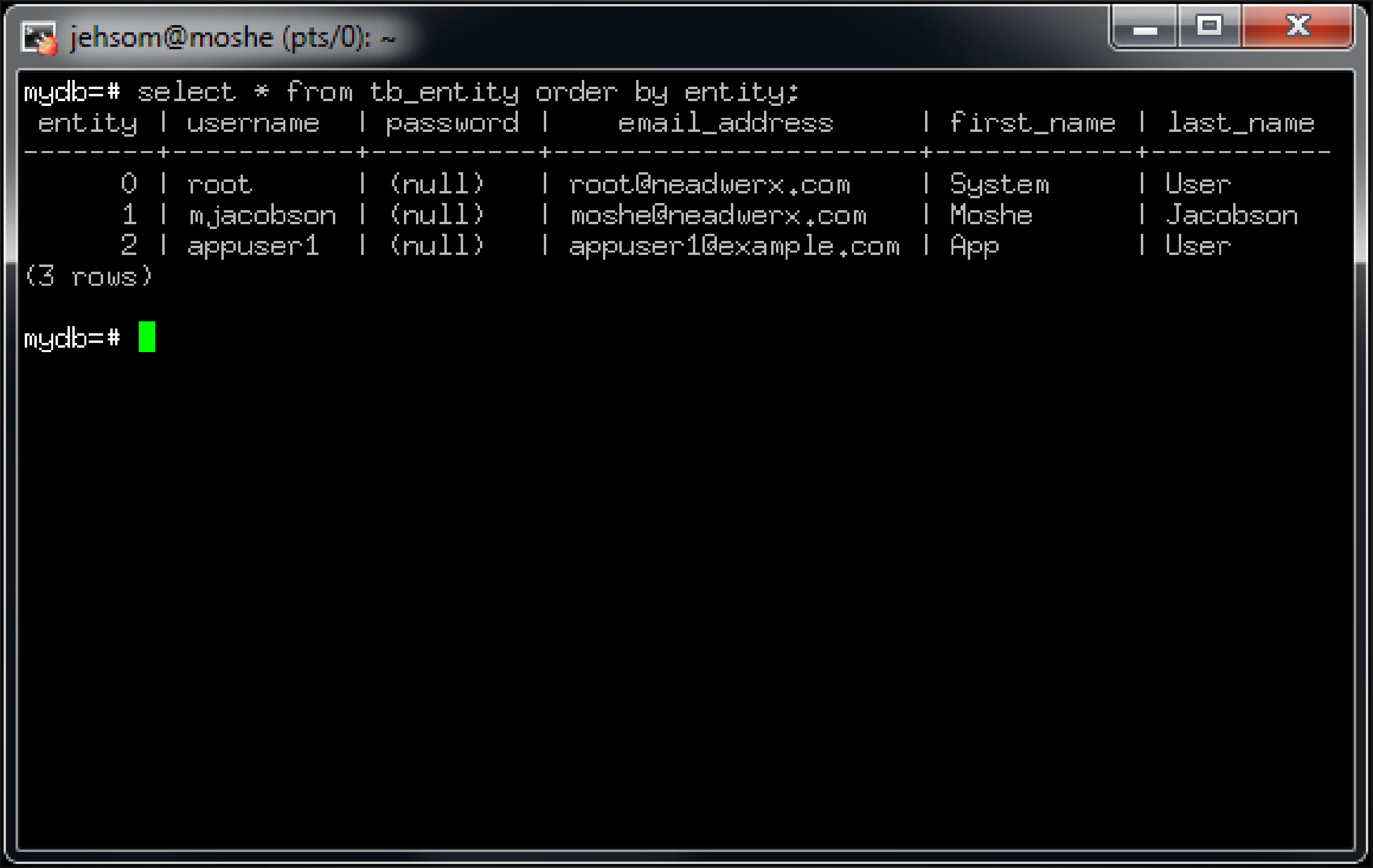
- Don't want to? Don't have to!

- Two modifications if you want:
  - Attach UIDs to transactions
  - Attach descriptions to transactions

Attaching UIDs to DML

- `fn_set_audit_uid(uid)`
- Match `current_user` to `user_table_username_col`
- Otherwise, assume 0

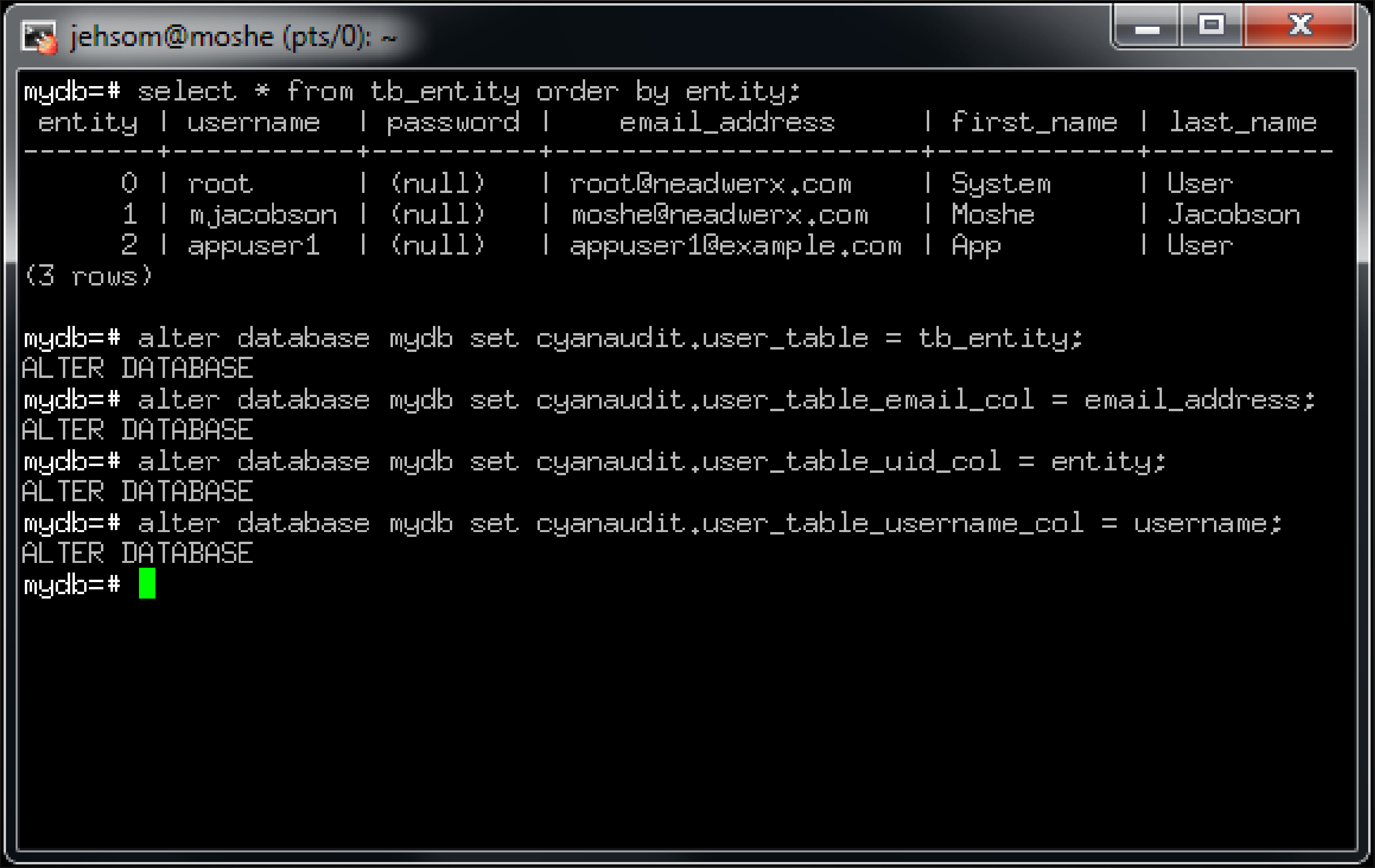
Attaching UIDs to DML



jehsom@moshe (pts/0): ~

```
mydb=# select * from tb_entity order by entity;
 entity | username  | password |    email_address      | first_name | last_name
--------+-----------+----------+-----------------------+------------+-----------
      0 | root      | (null)   | root@neadwerx.com     | System     | User
      1 | mjacobson | (null)   | moshe@neadwerx.com    | Moshe      | Jacobson
      2 | appuser1  | (null)   | appuser1@example.com  | App        | User
(3 rows)

mydb=#
```

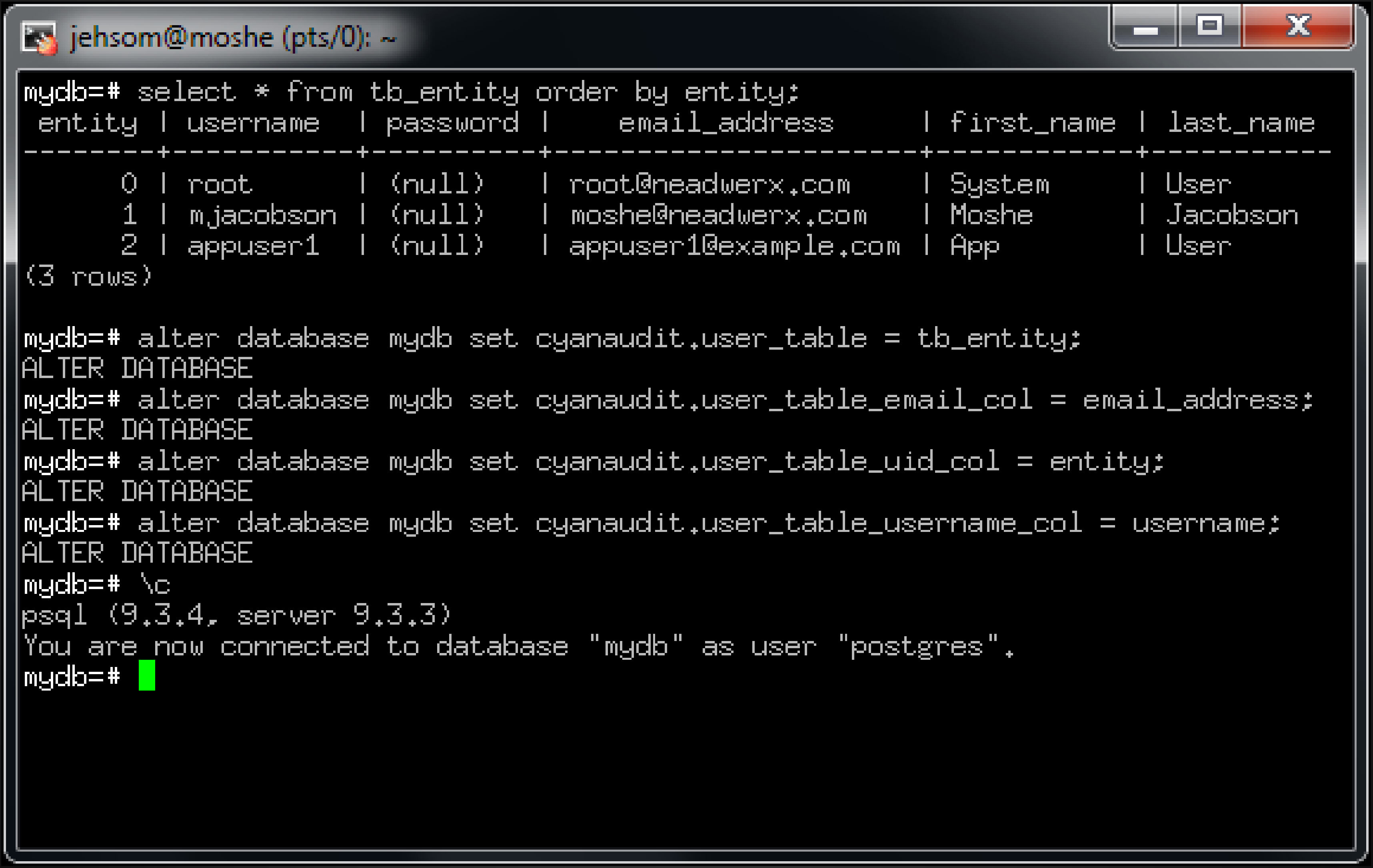
Attaching UIDs to DML



jehsom@moshe (pts/0): ~

```
mydb=# select * from tb_entity order by entity;
 entity | username  | password |     email_address      | first_name | last_name
--------+-----------+----------+------------------------+------------+-----------
      0 | root      | (null)   | root@neadwerx.com      | System     | User
      1 | mjacobson | (null)   | moshe@neadwerx.com     | Moshe      | Jacobson
      2 | appuser1  | (null)   | appuser1@example.com   | App        | User
(3 rows)

mydb=# alter database mydb set cyanaudit.user_table = tb_entity;
ALTER DATABASE
mydb=# alter database mydb set cyanaudit.user_table_email_col = email_address;
ALTER DATABASE
mydb=# alter database mydb set cyanaudit.user_table_uid_col = entity;
ALTER DATABASE
mydb=# alter database mydb set cyanaudit.user_table_username_col = username;
ALTER DATABASE
mydb=# 
```

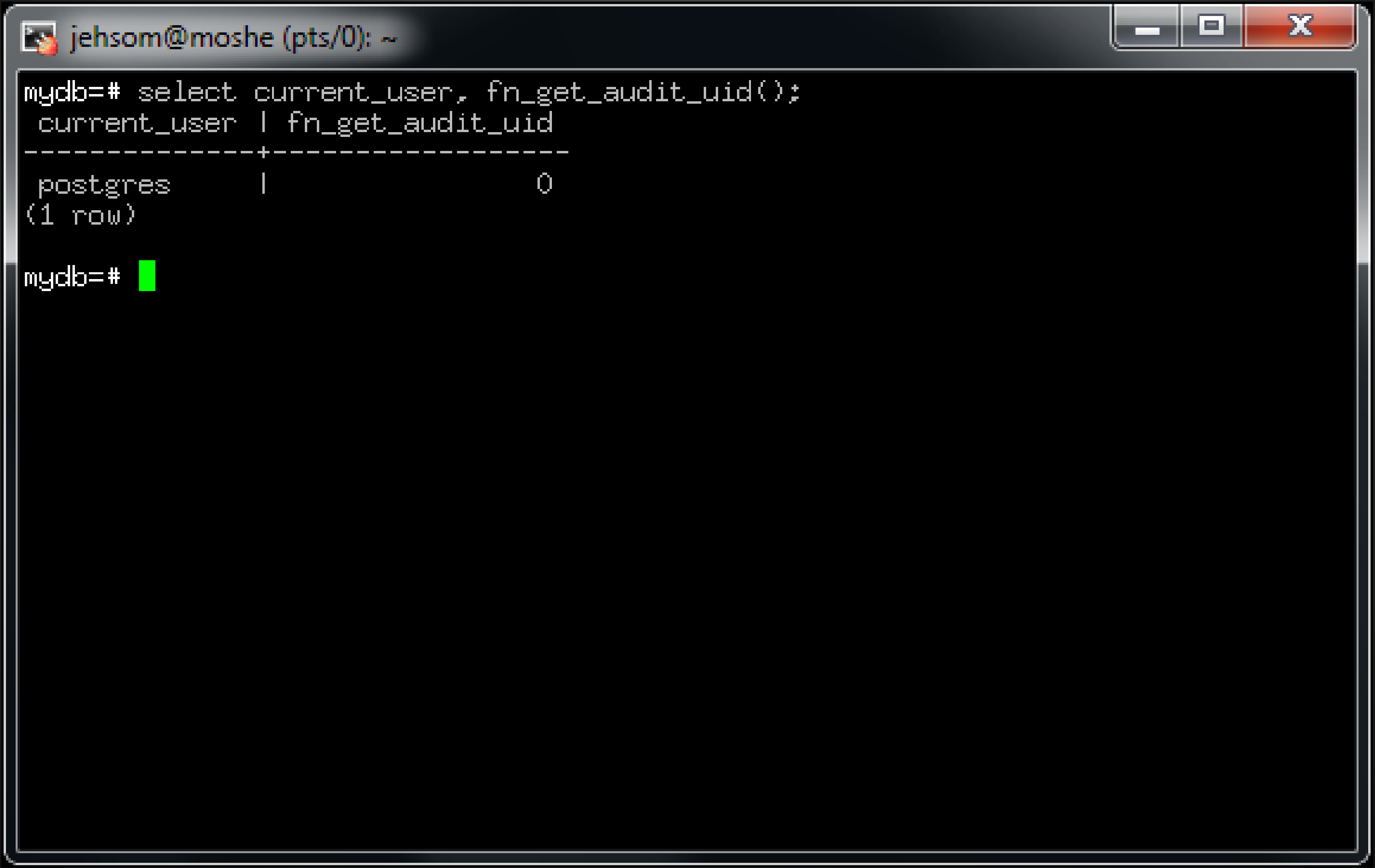
Attaching UIDs to DML



jehsom@moshe (pts/0): ~

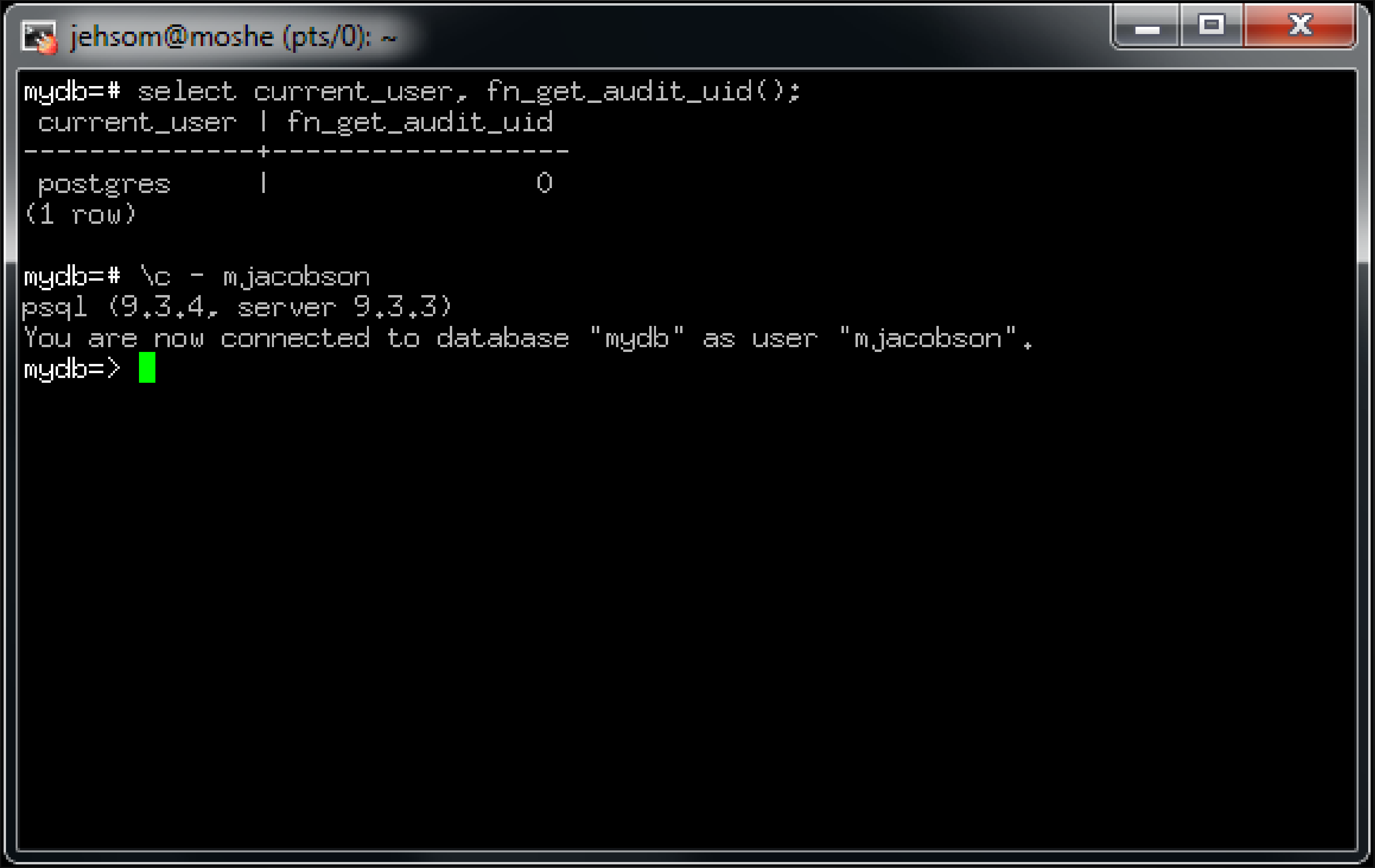
```
mydb=# select * from tb_entity order by entity;
 entity | username   | password |     email_address      | first_name | last_name
--------+------------+----------+------------------------+------------+-----------
      0 | root       | (null)   | root@neadwerx.com      | System     | User
      1 | mjacobson  | (null)   | moshe@neadwerx.com     | Moshe      | Jacobson
      2 | appuser1   | (null)   | appuser1@example.com   | App        | User
(3 rows)

mydb=# alter database mydb set cyanaudit.user_table = tb_entity;
ALTER DATABASE
mydb=# alter database mydb set cyanaudit.user_table_email_col = email_address;
ALTER DATABASE
mydb=# alter database mydb set cyanaudit.user_table_uid_col = entity;
ALTER DATABASE
mydb=# alter database mydb set cyanaudit.user_table_username_col = username;
ALTER DATABASE
mydb=# \c
psql (9.3.4, server 9.3.3)
You are now connected to database "mydb" as user "postgres".
mydb=#
```

Attaching UIDs to DML



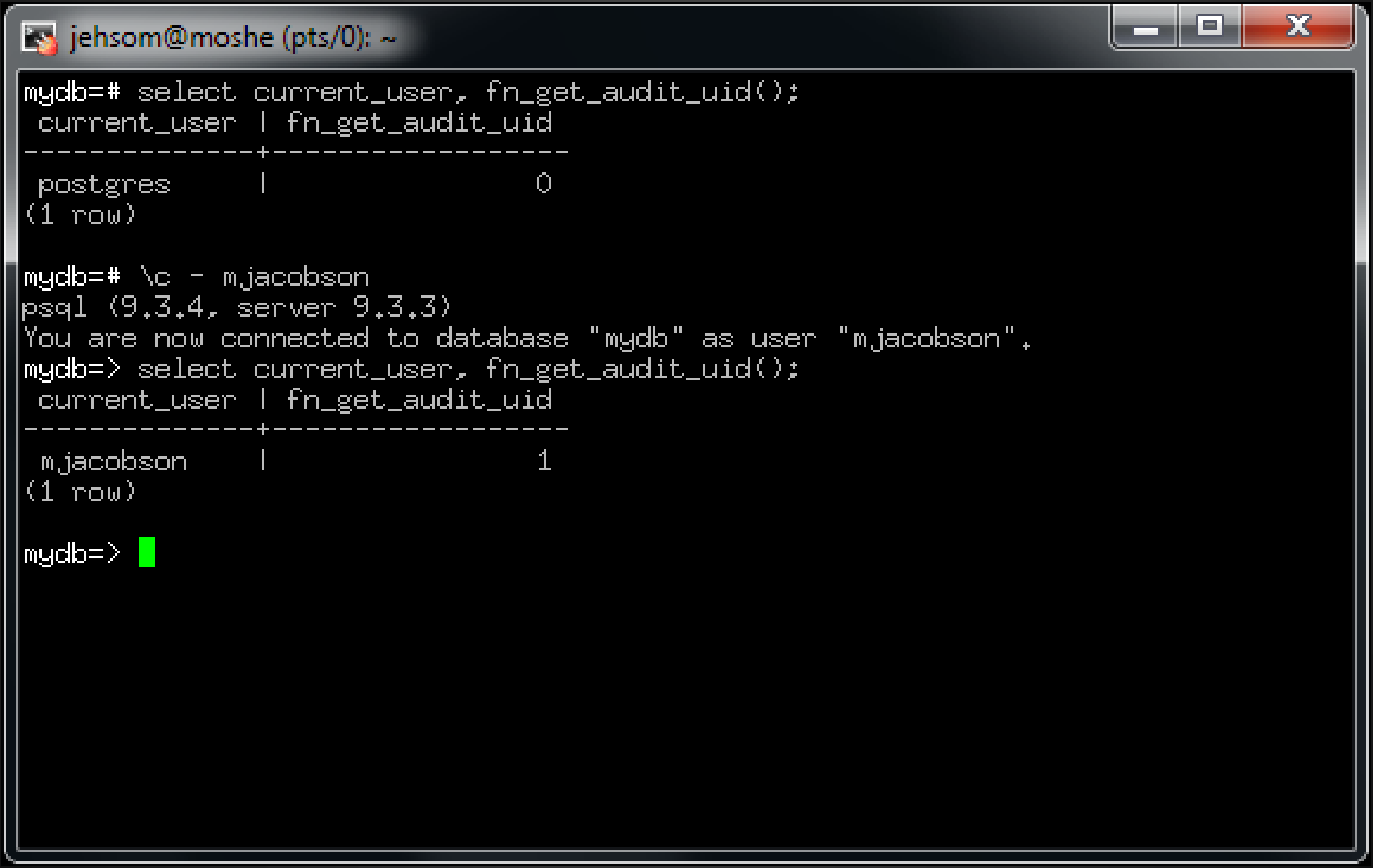
```
mydb=# select current_user, fn_get_audit_uid();
 current_user | fn_get_audit_uid
--------------+------------------
 postgres     |                0
(1 row)

mydb=#
```

Attaching UIDs to DML



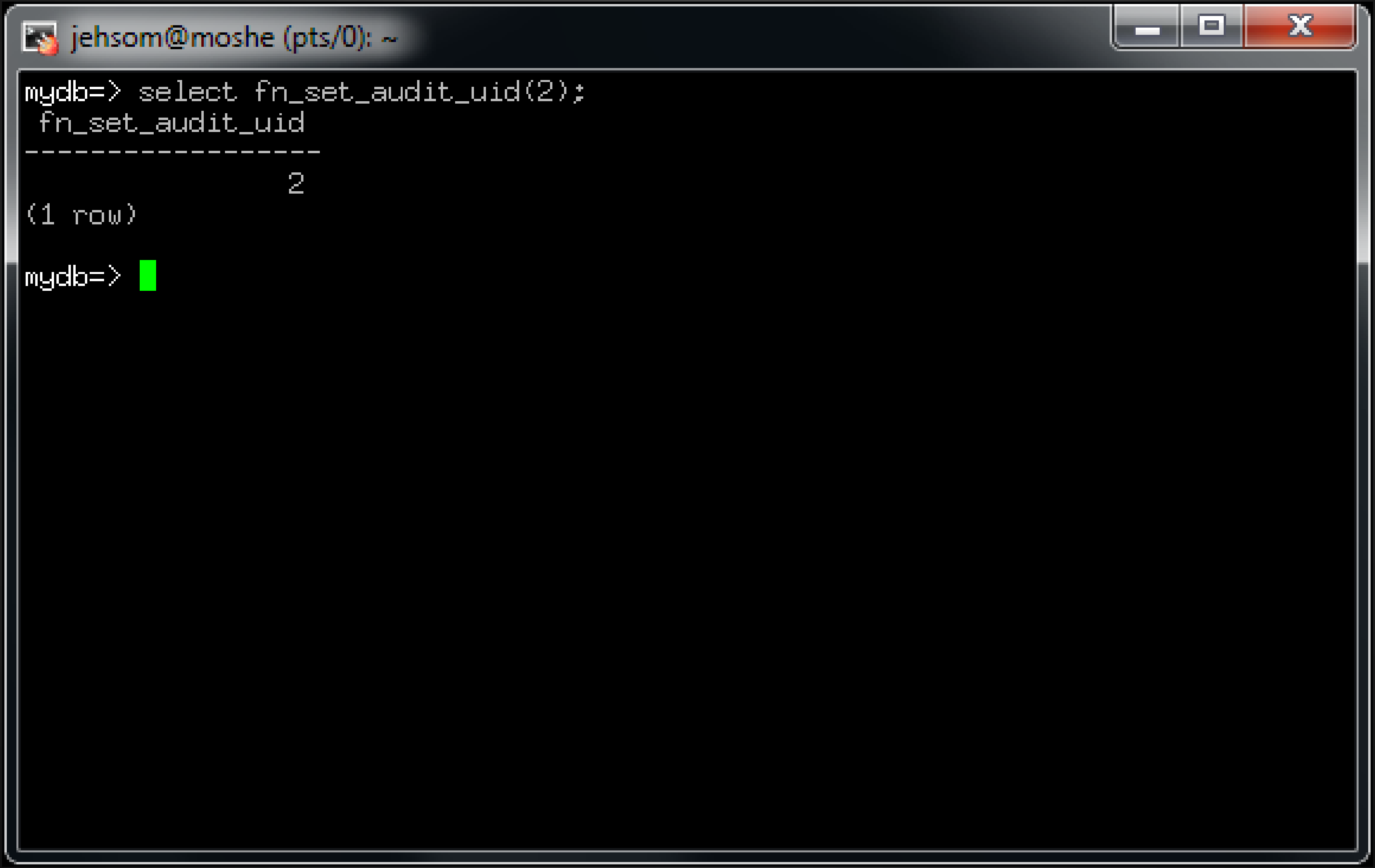
```
mydb=# select current_user, fn_get_audit_uid();
 current_user | fn_get_audit_uid
--------------+------------------
 postgres     |                0
(1 row)

mydb=# \c - mjacobson
psql (9.3.4, server 9.3.3)
You are now connected to database "mydb" as user "mjacobson".
mydb=>
```

Attaching UIDs to DML



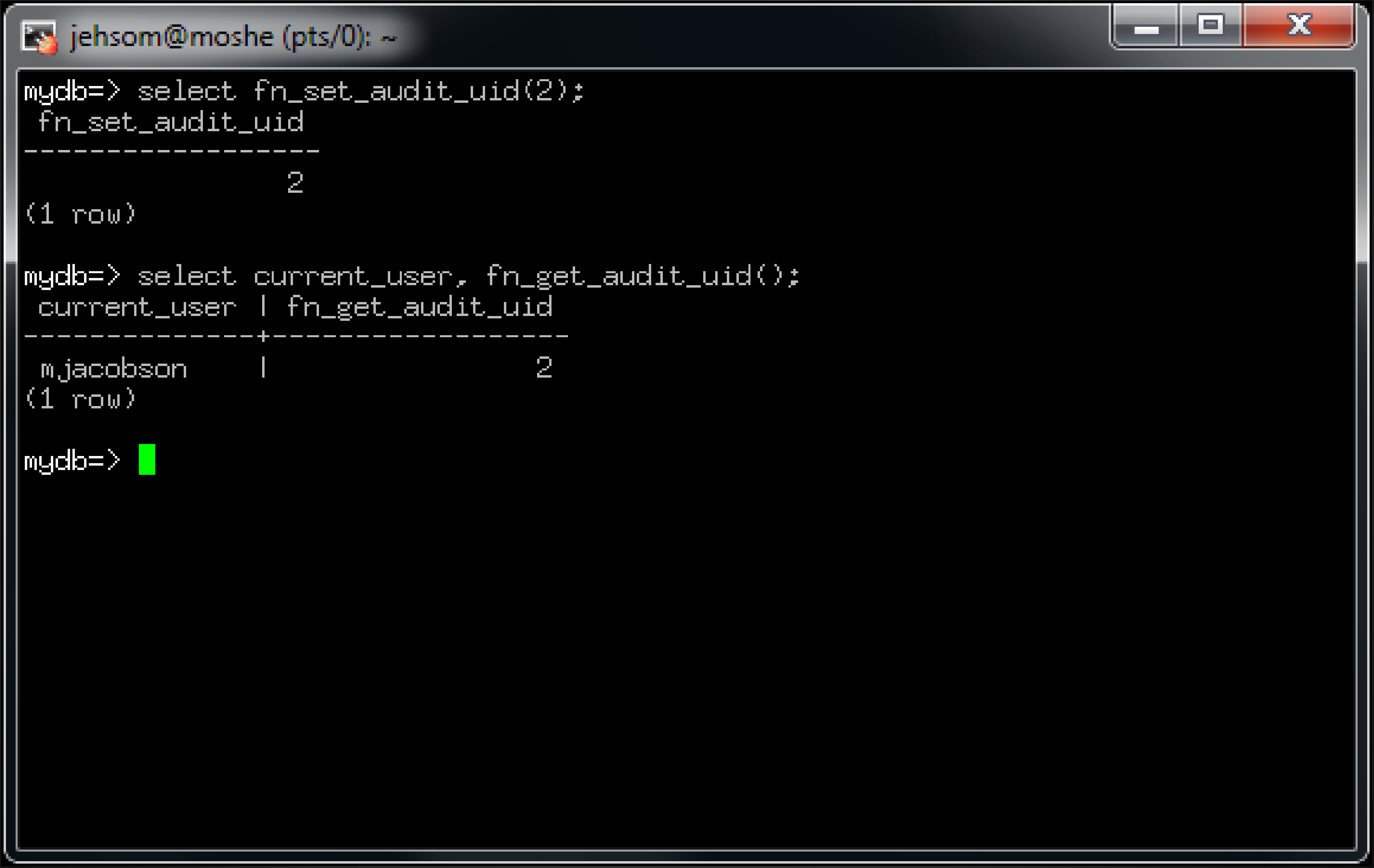
```
mydb=# select current_user, fn_get_audit_uid();
 current_user | fn_get_audit_uid
--------------+------------------
 postgres     |                0
(1 row)

mydb=# \c - mjacobson
psql (9.3.4, server 9.3.3)
You are now connected to database "mydb" as user "mjacobson".
mydb=> select current_user, fn_get_audit_uid();
 current_user | fn_get_audit_uid
--------------+------------------
 mjacobson    |                1
(1 row)

mydb=> 
```

Attaching UIDs to DML


jehsom@moshe (pts/0): ~

```
mydb=> select fn_set_audit_uid(2);
 fn_set_audit_uid
------------------
                2
(1 row)

mydb=> 
```

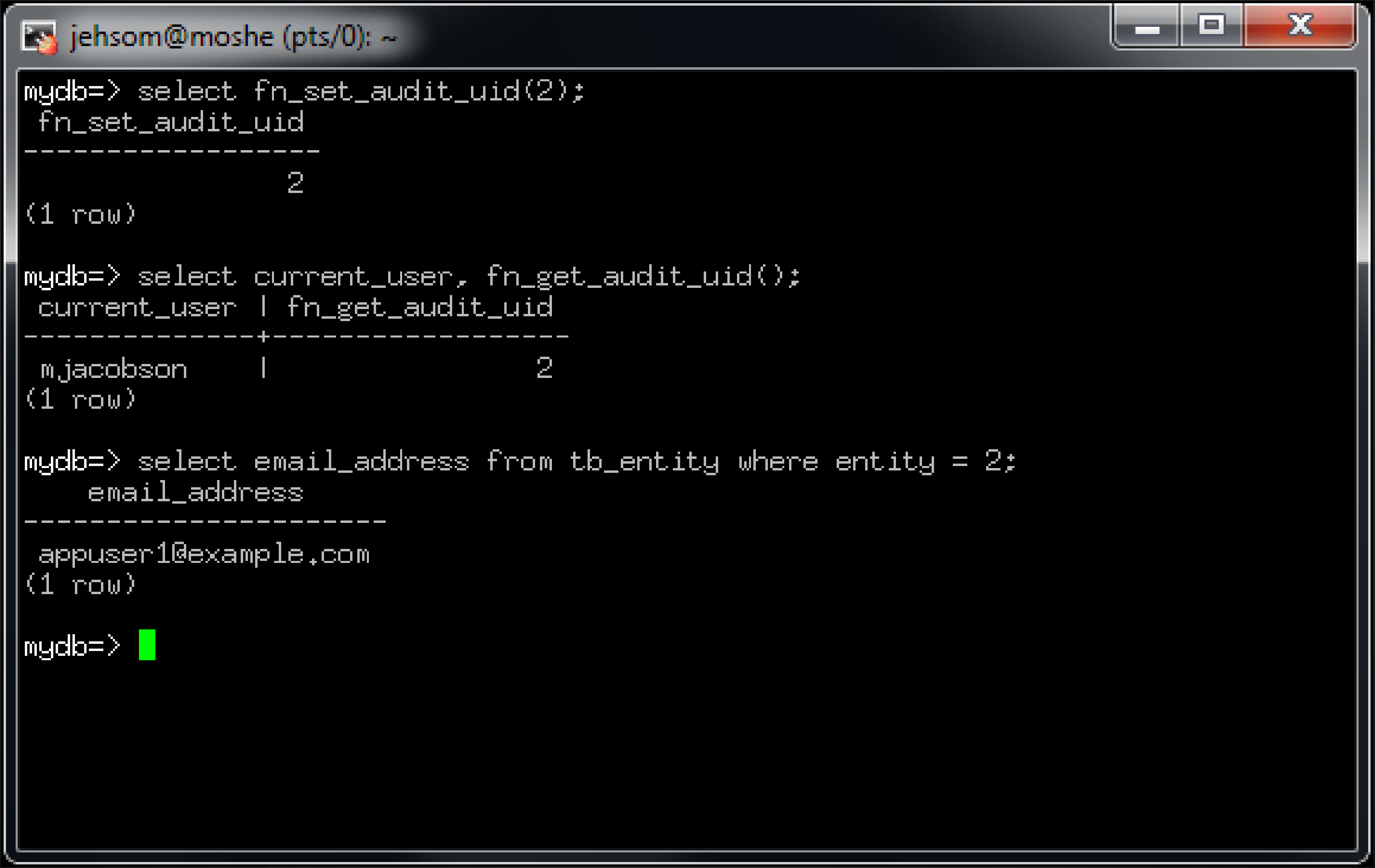
Attaching UIDs to DML



jehsom@moshe (pts/0): ~

```
mydb=> select fn_set_audit_uid(2);
 fn_set_audit_uid
------------------
                2
(1 row)

mydb=> select current_user, fn_get_audit_uid();
 current_user | fn_get_audit_uid
--------------+------------------
 mjacobson    |                2
(1 row)

mydb=> 
```

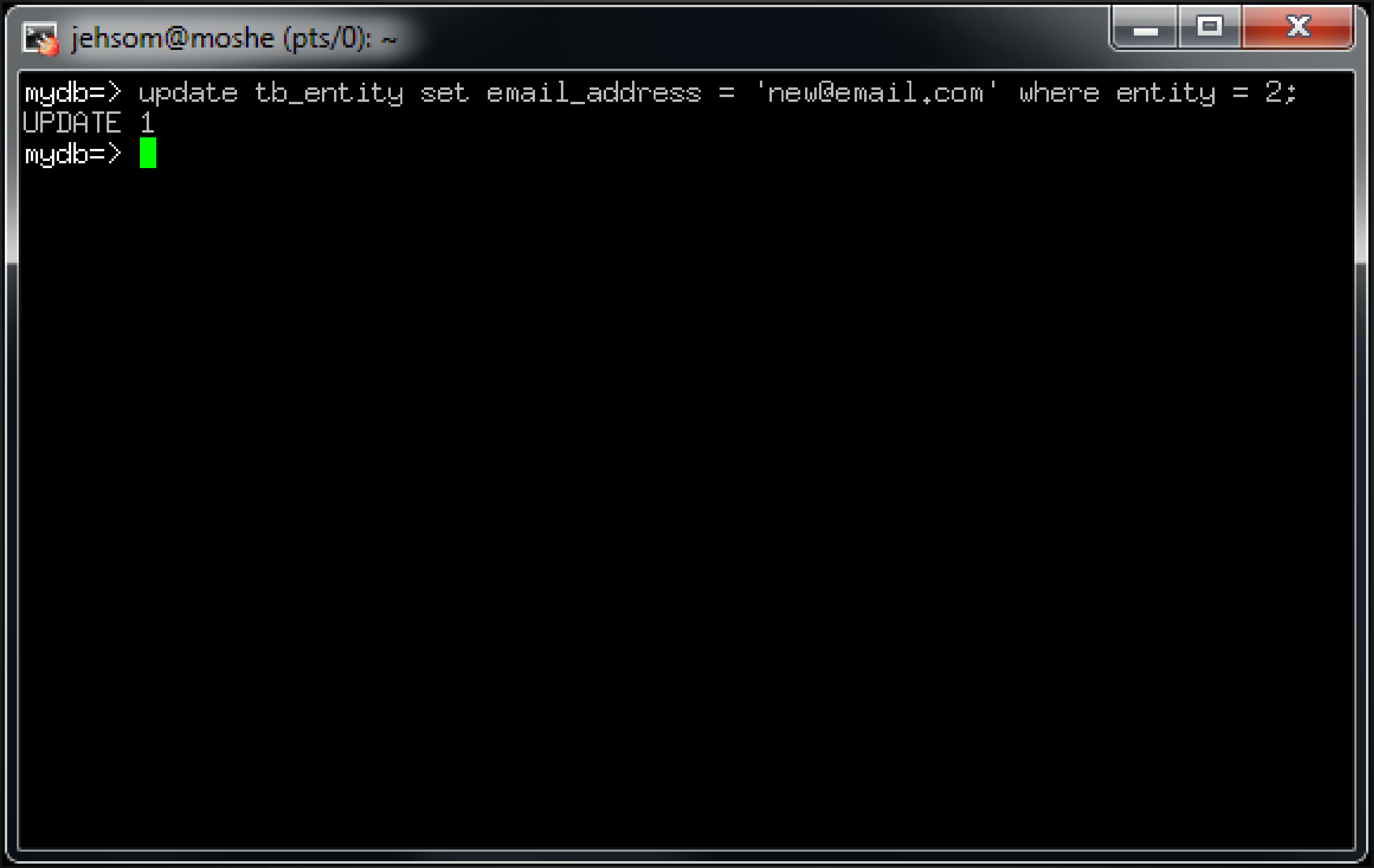
Attaching UIDs to DML



jehsom@moshe (pts/0): ~

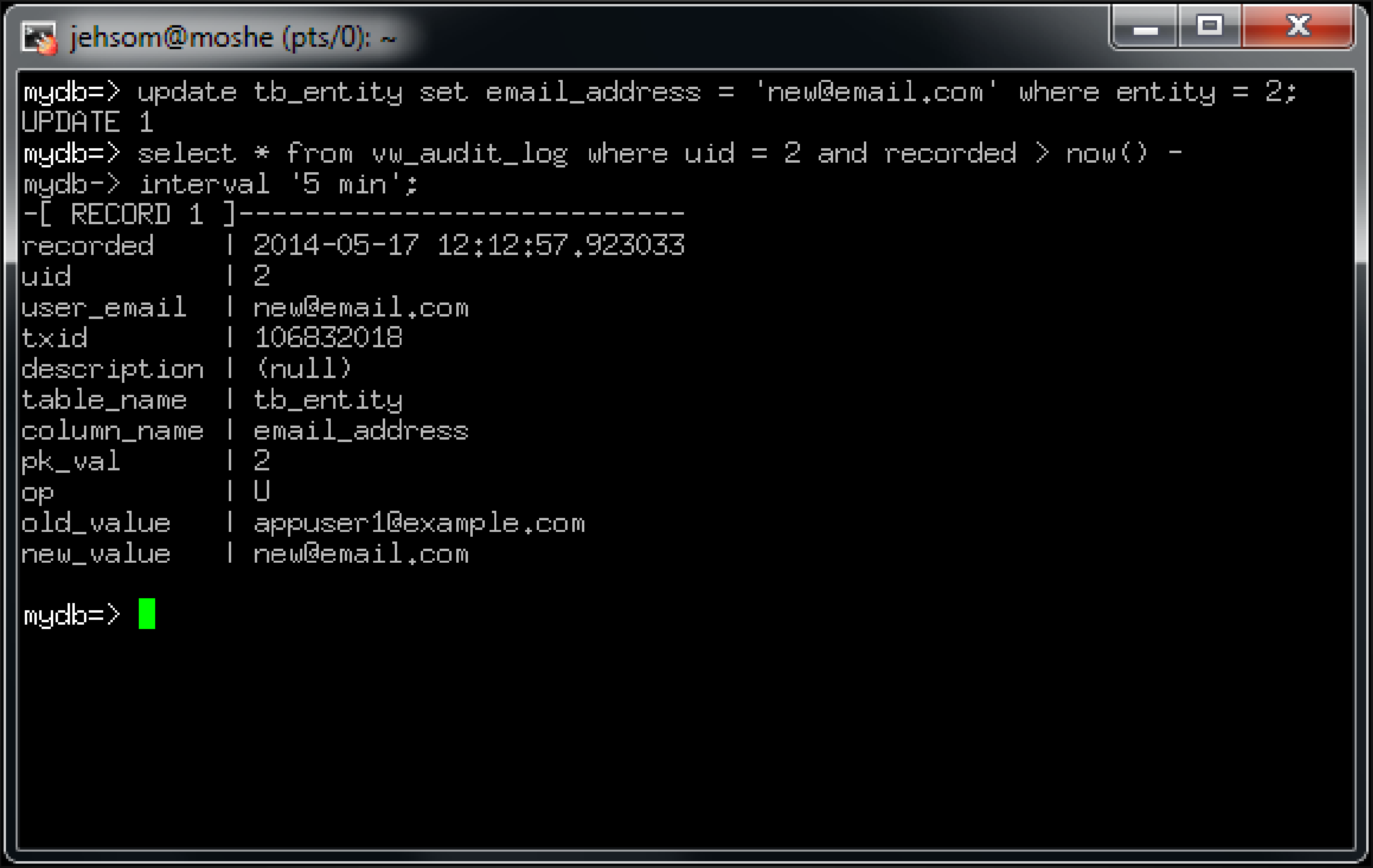
```
mydb=> select fn_set_audit_uid(2);
 fn_set_audit_uid
------------------
                2
(1 row)

mydb=> select current_user, fn_get_audit_uid();
 current_user | fn_get_audit_uid
--------------+------------------
 mjacobson    |                2
(1 row)

mydb=> select email_address from tb_entity where entity = 2;
    email_address
----------------------
 appuser1@example.com
(1 row)

mydb=> 
```

Attaching UIDs to DML



```
mydb=> update tb_entity set email_address = 'new@email.com' where entity = 2;
UPDATE 1
mydb=> 
```

Attaching UIDs to DML



```
mydb=> update tb_entity set email_address = 'new@email.com' where entity = 2;
UPDATE 1
mydb=> select * from vw_audit_log where uid = 2 and recorded > now() -
mydb-> interval '5 min';
-[ RECORD 1 ]---------------------------
recorded    | 2014-05-17 12:12:57.923033
uid         | 2
user_email  | new@email.com
txid        | 106832018
description | (null)
table_name  | tb_entity
column_name | email_address
pk_val      | 2
op          | U
old_value   | appuser1@example.com
new_value   | new@email.com

mydb=> 
```

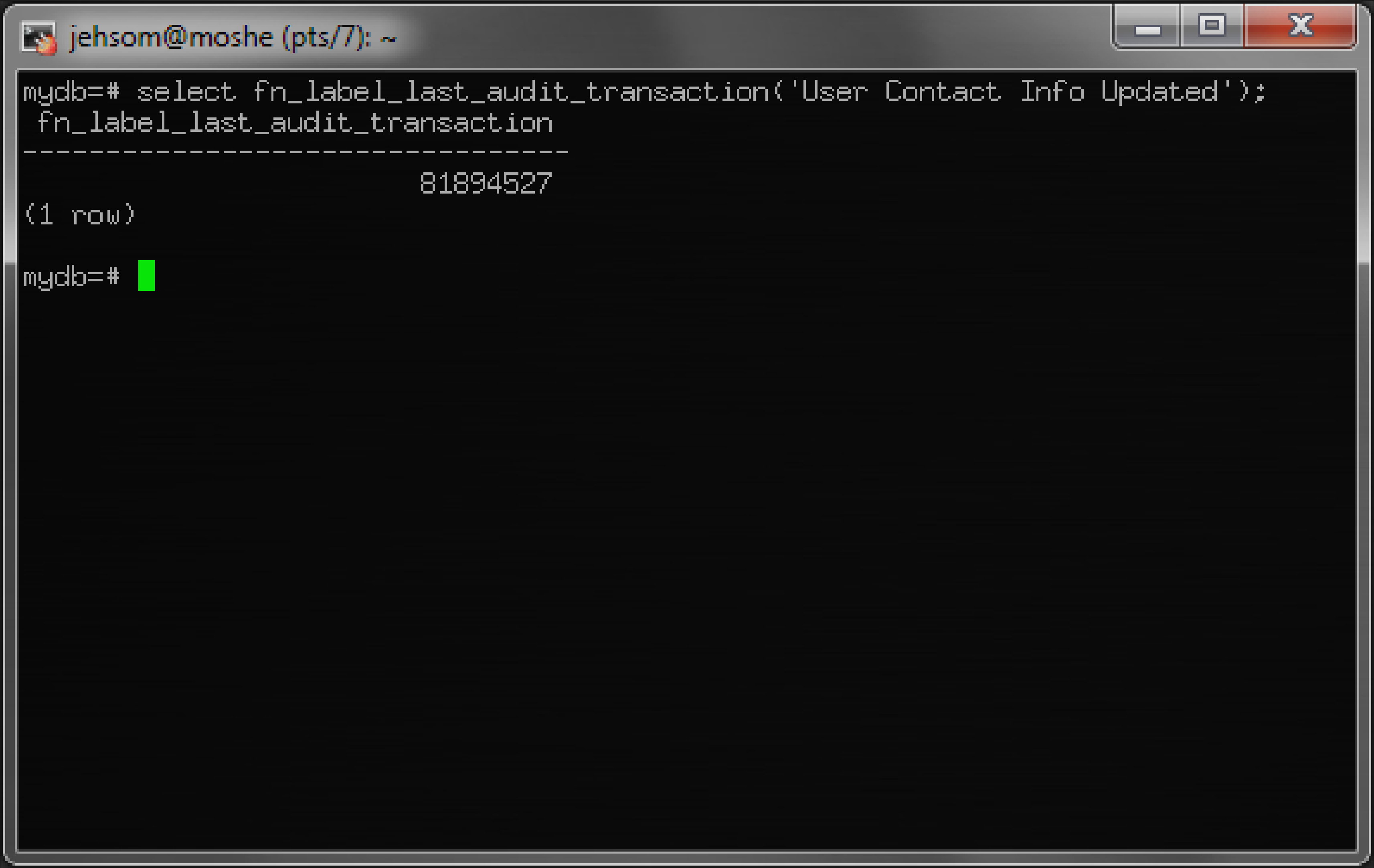
Labeling transactions

- Not everyone understands the schema.
- Let's help them out.

- Two functions for labeling transactions:

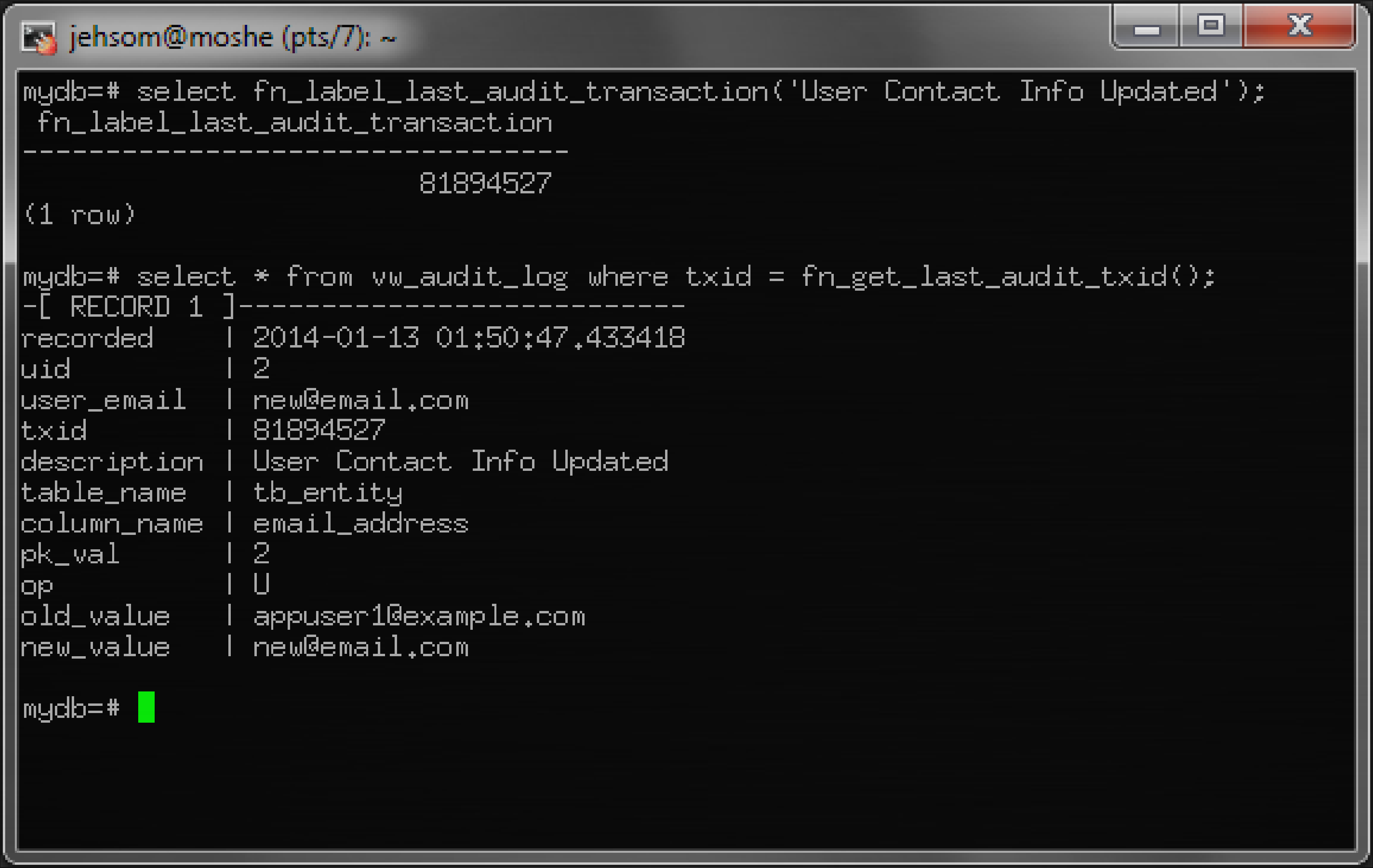
```
fn_label_audit_transaction(label, txid)
fn_label_last_audit_transaction(label)
```

Labeling transactions



```
mydb=# select fn_label_last_audit_transaction('User Contact Info Updated');
 fn_label_last_audit_transaction
---------------------------------
                        81894527
(1 row)

mydb=# 
```

Labeling transactions



jehsom@moshe (pts/7): ~

```
mydb=# select fn_label_last_audit_transaction('User Contact Info Updated');
 fn_label_last_audit_transaction
---------------------------------
                        81894527
(1 row)

mydb=# select * from vw_audit_log where txid = fn_get_last_audit_txid();
-[ RECORD 1 ]---------------------------
recorded    | 2014-01-13 01:50:47.433418
uid         | 2
user_email  | new@email.com
txid        | 81894527
description | User Contact Info Updated
table_name  | tb_entity
column_name | email_address
pk_val      | 2
op          | U
old_value   | appuser1@example.com
new_value   | new@email.com

mydb=#
```

Log Rotation/Archival

- You’re gonna run out of space eventually.
- What is the solution?

XCOMP introduces a complete micro-size disk subsystem with more...

- MORE STORAGE
- MORE SPEED
- MORE VALUE
- MORE SUPPORT

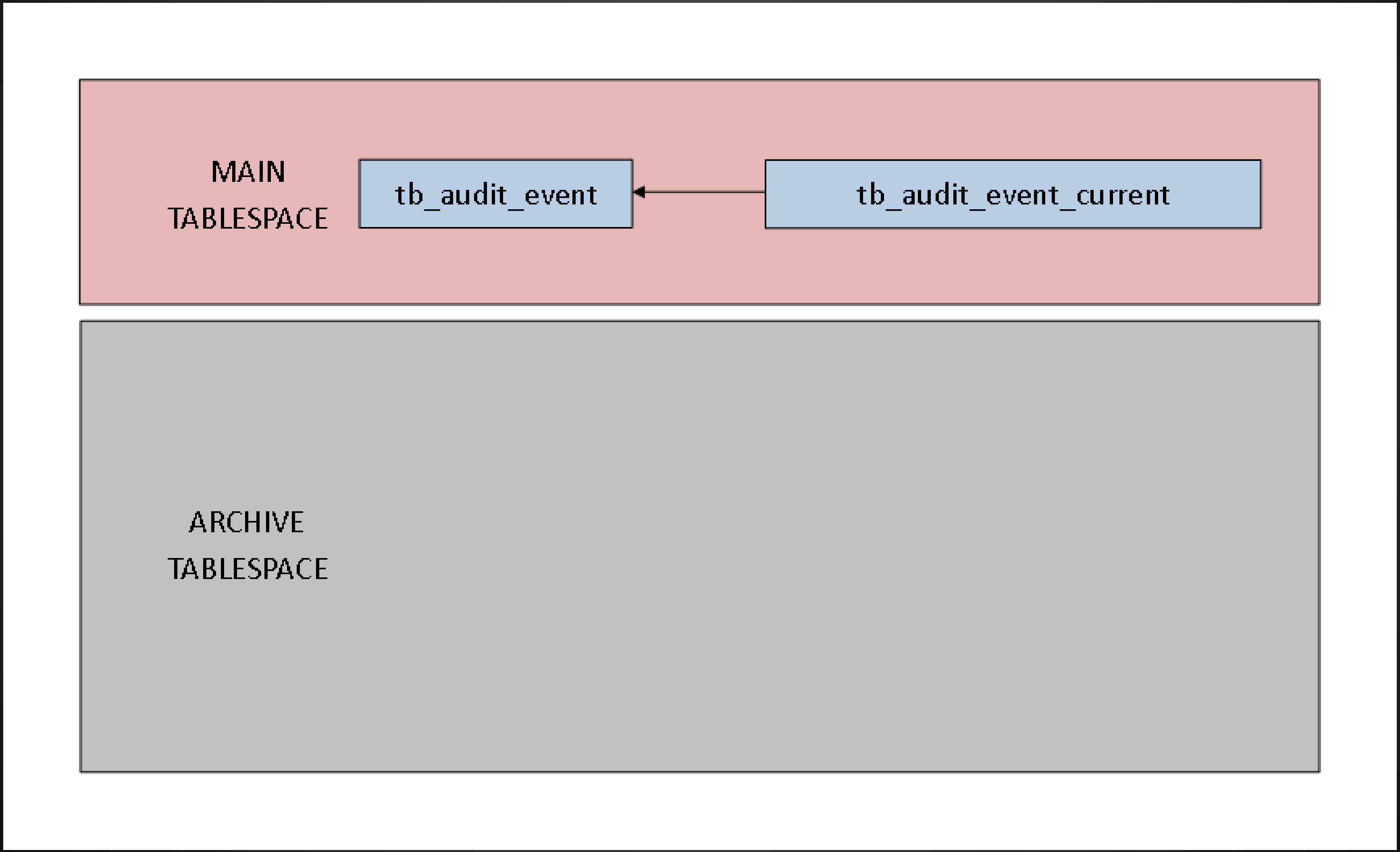
The XCOMP subsystem is now [illegible] S100 users . . . 10 megabytes of storage; 5 megabytes [illegible] Compare the price and [illegible] 8-inch

MORE SOFTWARE
Included with the system is software for testing, formatting, I/O drivers for CP/M®, plus an automatic CP/M driver attach program. Support software and drivers for MP/M® and Oasis® are also available. The sophisticated formatting program assigns alternate sectors for any weak sectors detected during formatting, assuring the lowest possible error rate — at least ten times better than floppies.

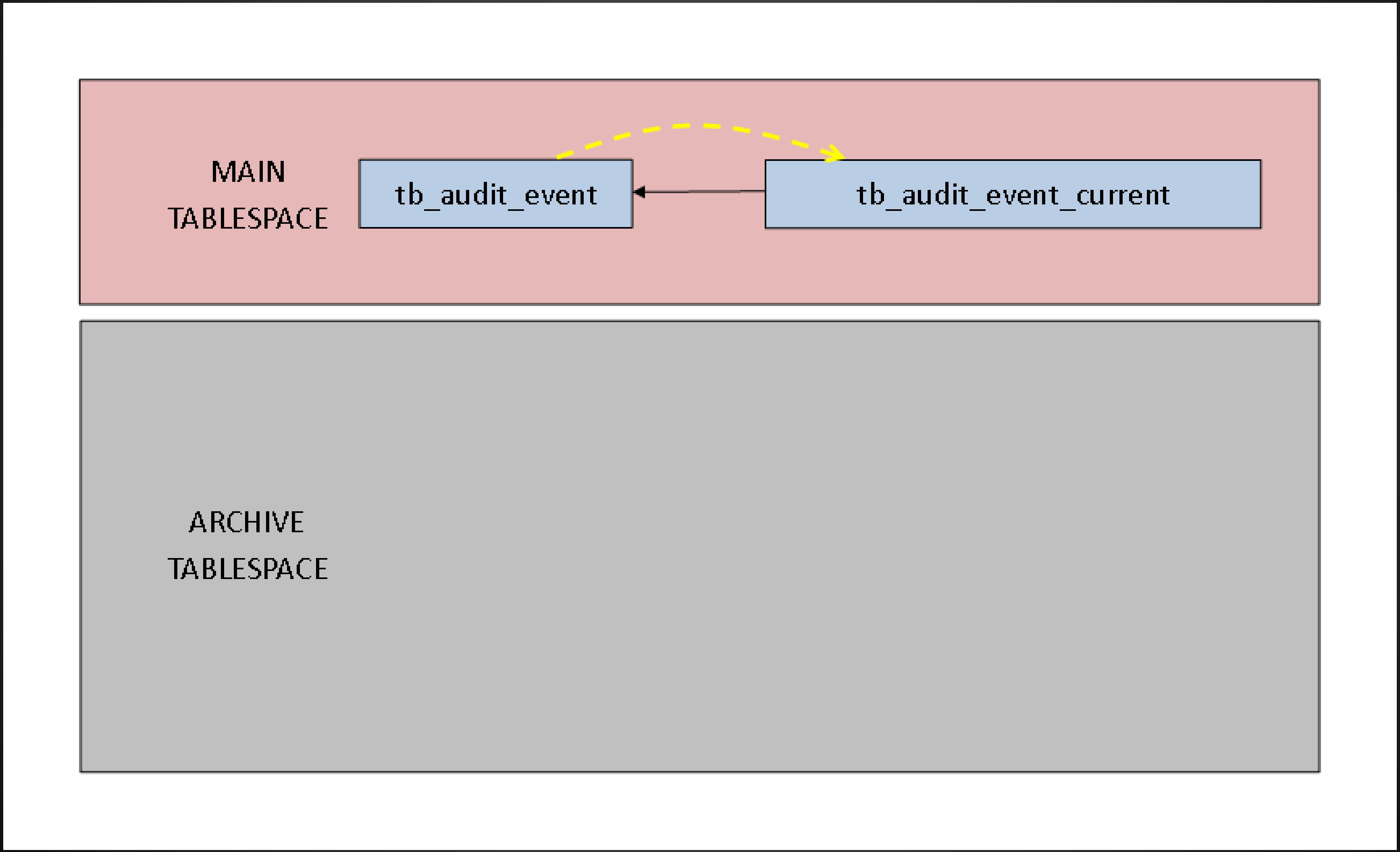
WARRANTY
The system has a full one-year warranty on parts and [illegible]

XCOMP

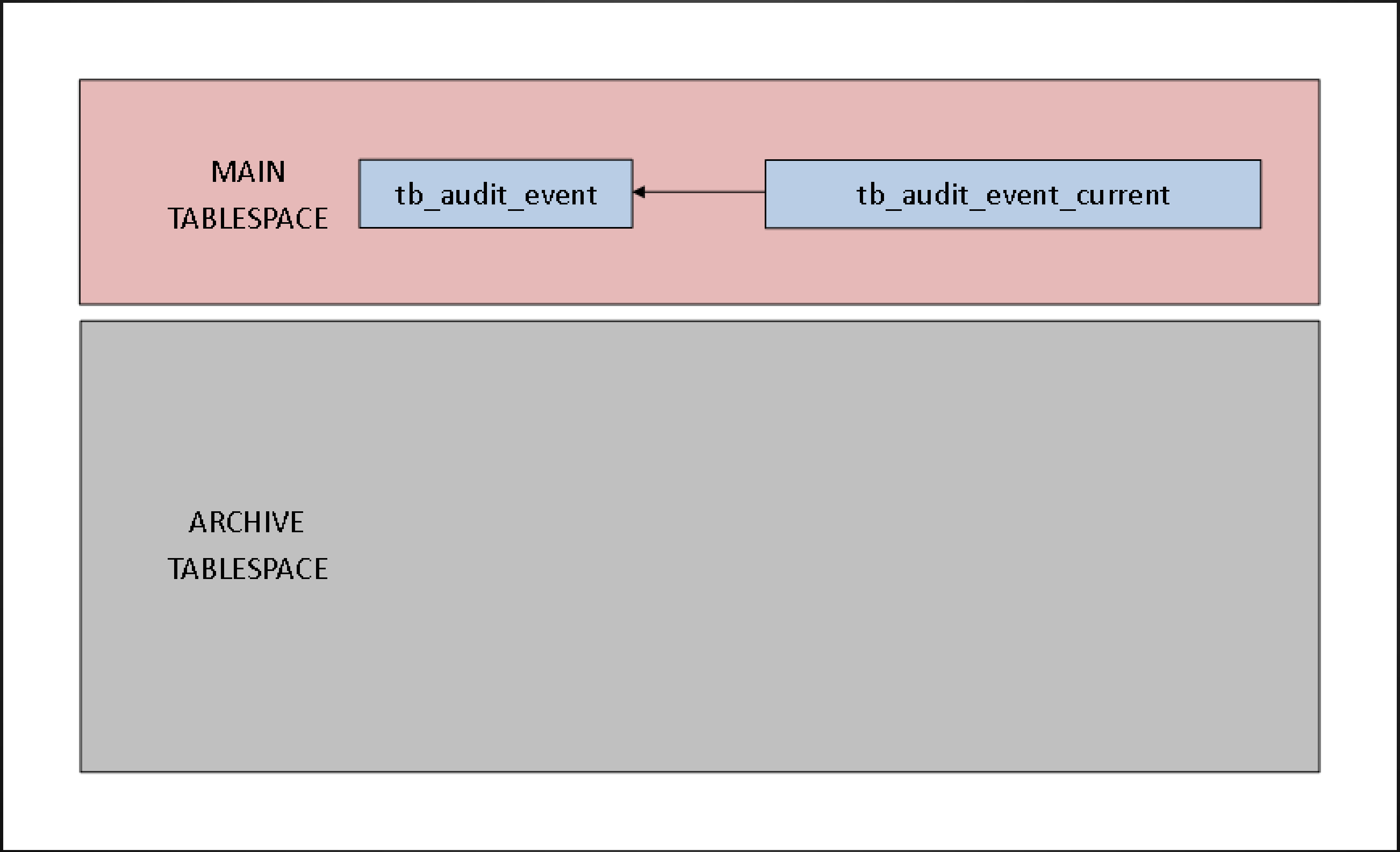
Log Rotation/Archival



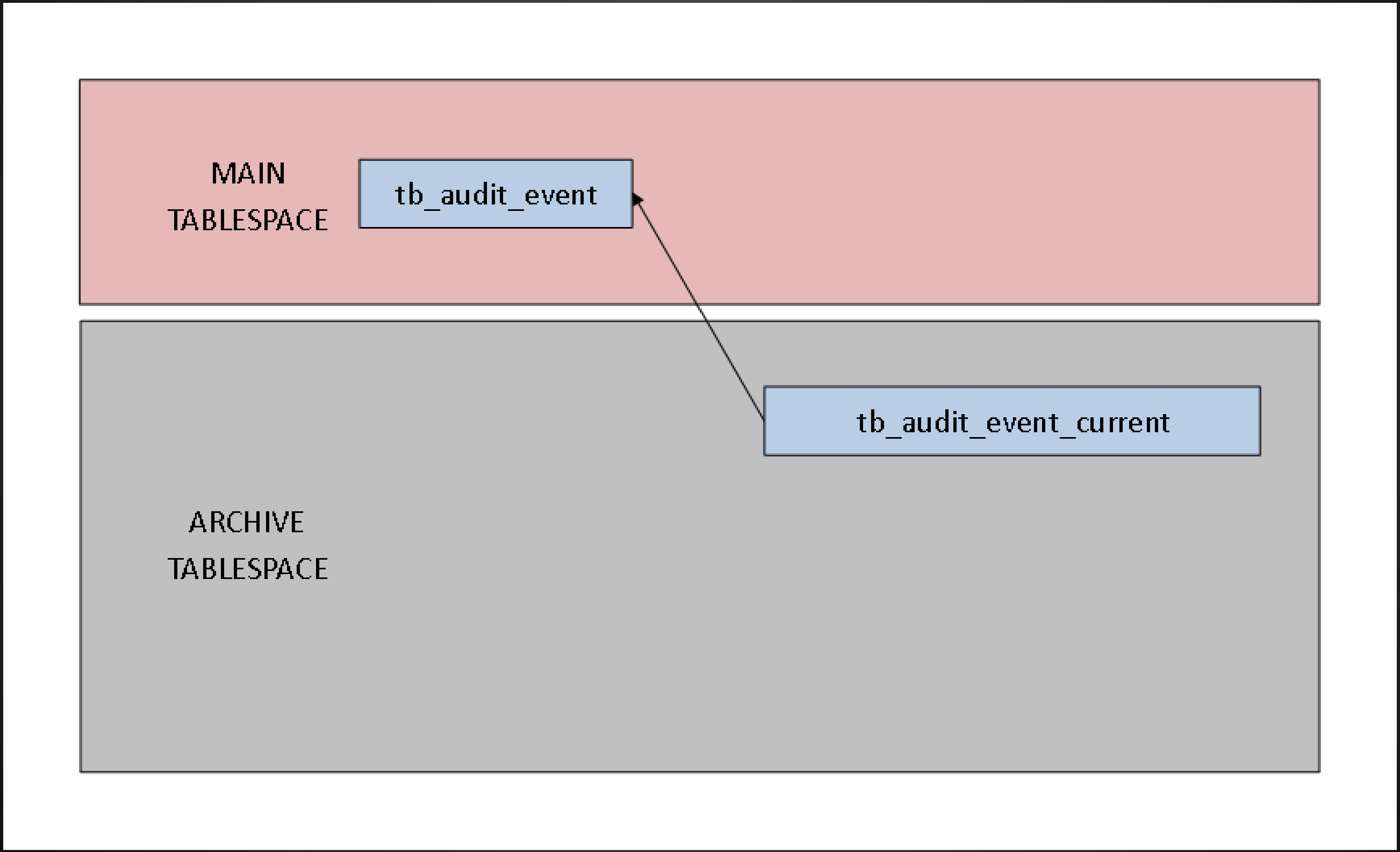
Log Rotation/Archival



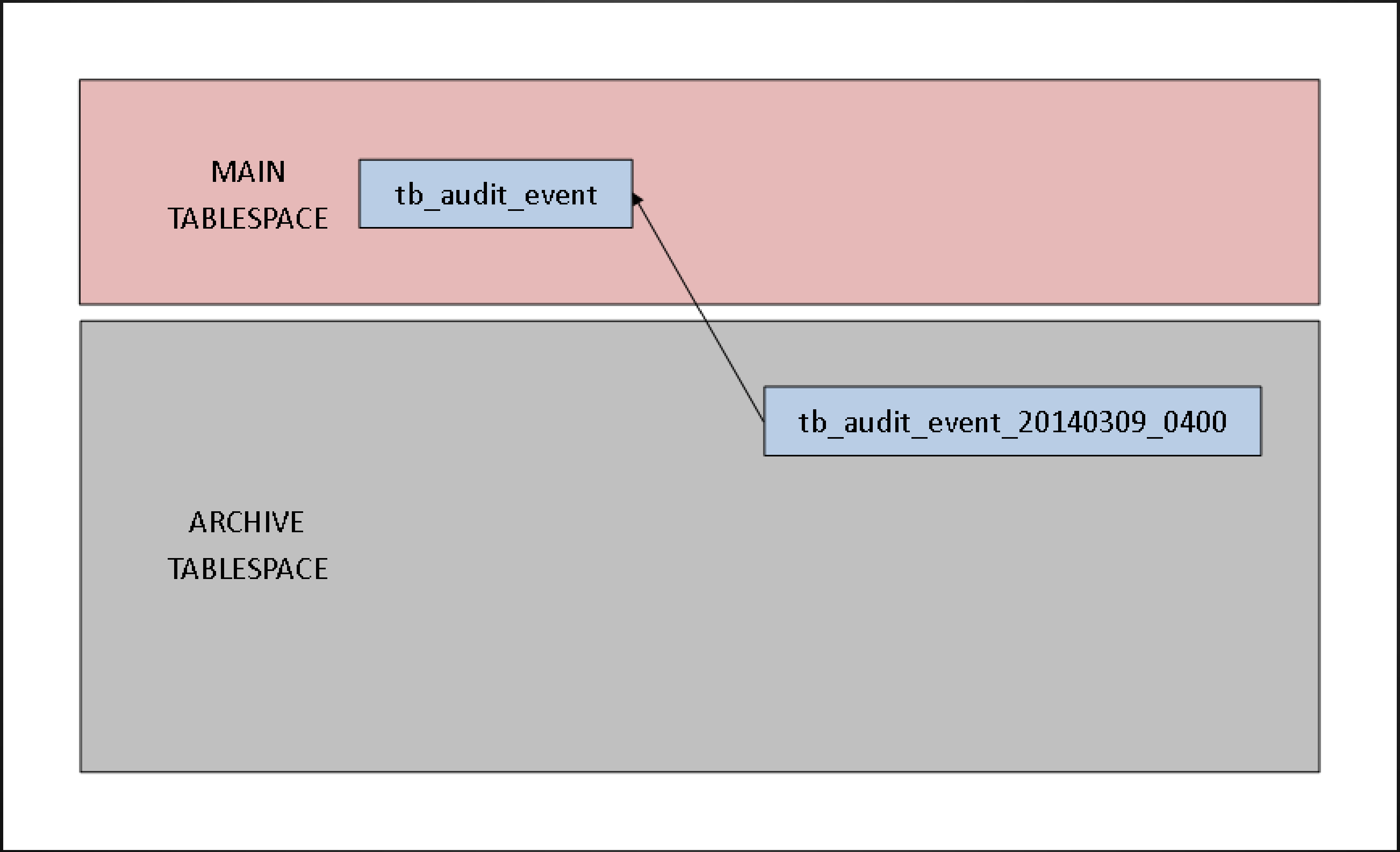
Log Rotation/Archival



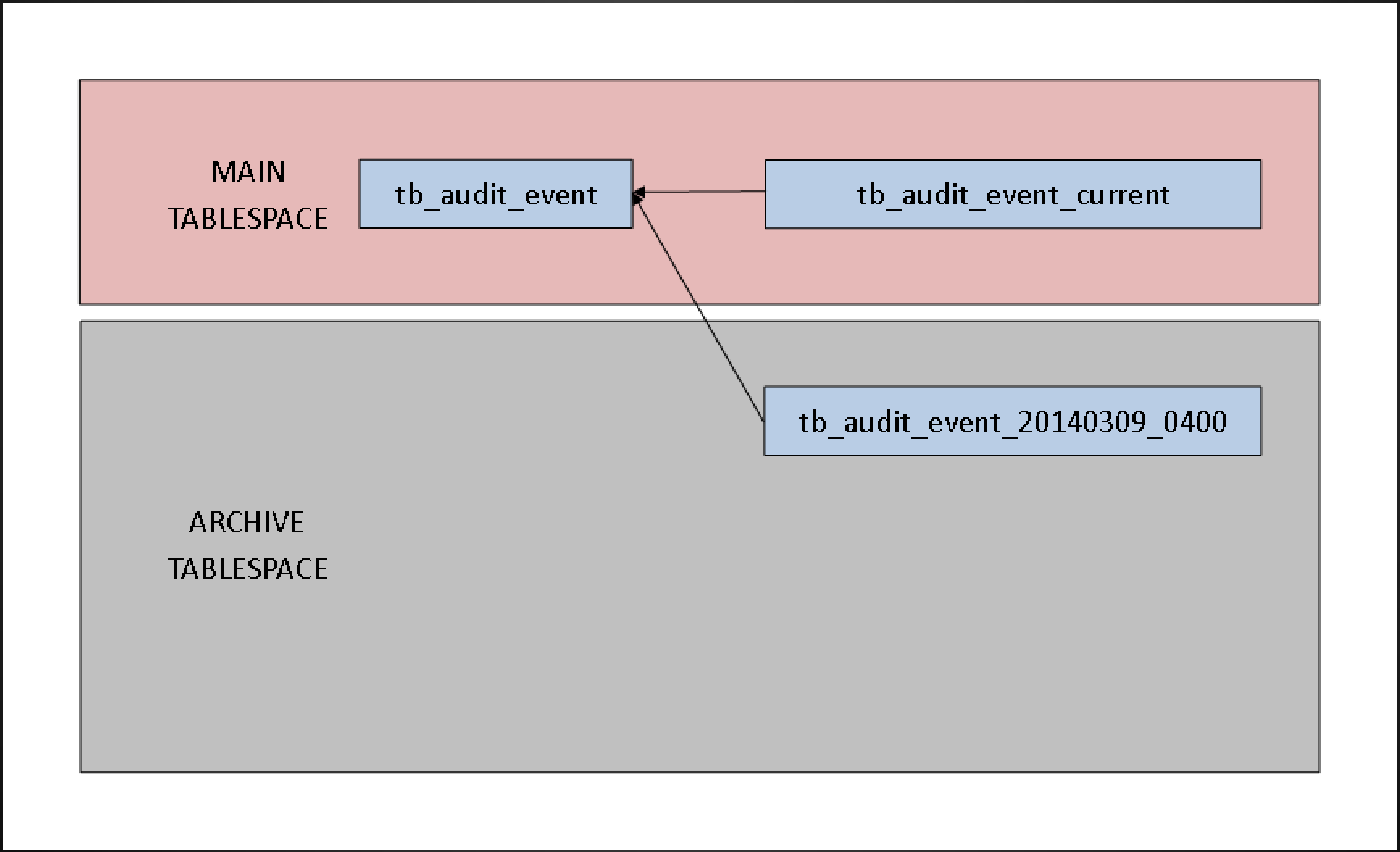
Log Rotation/Archival



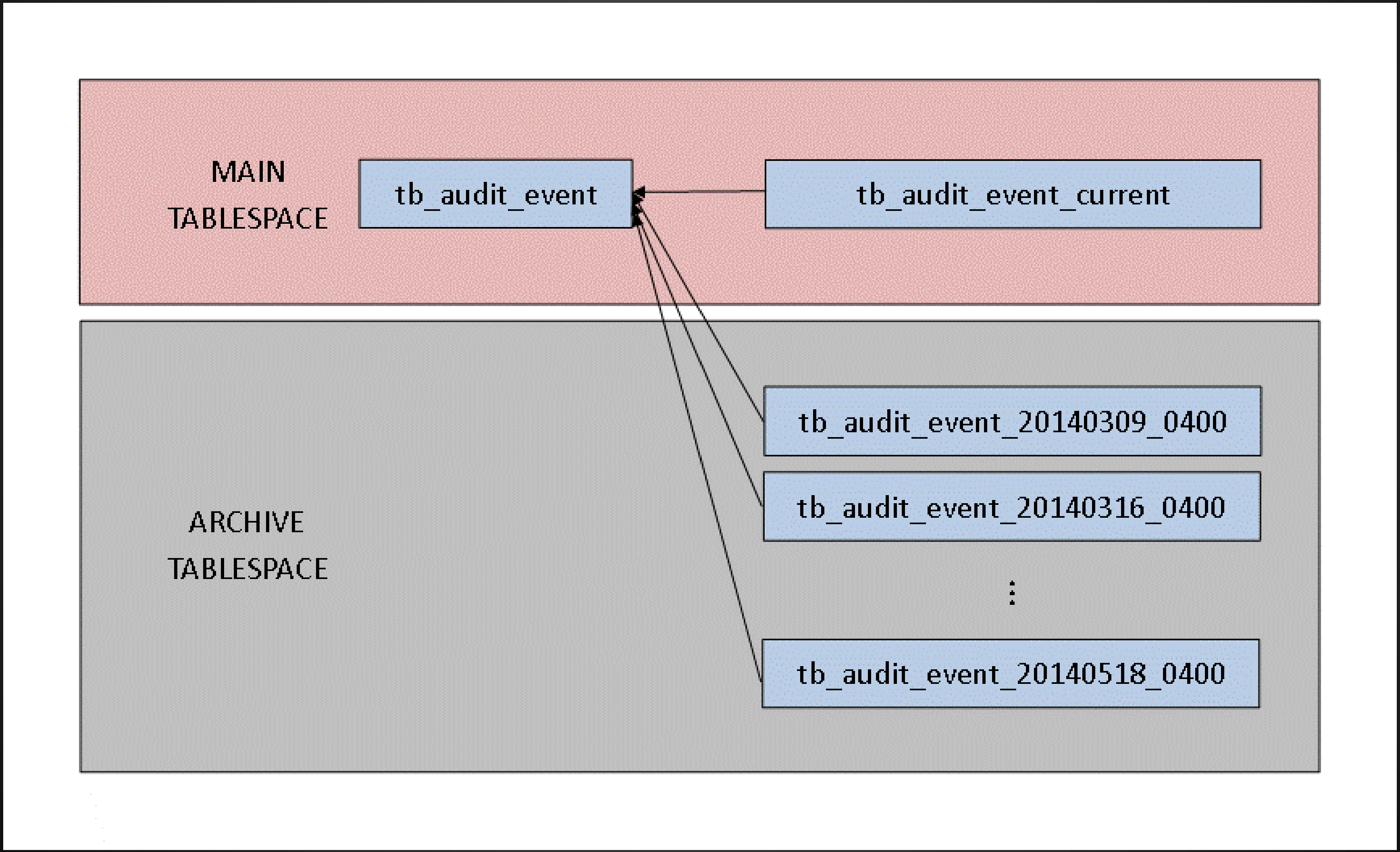
Log Rotation/Archival



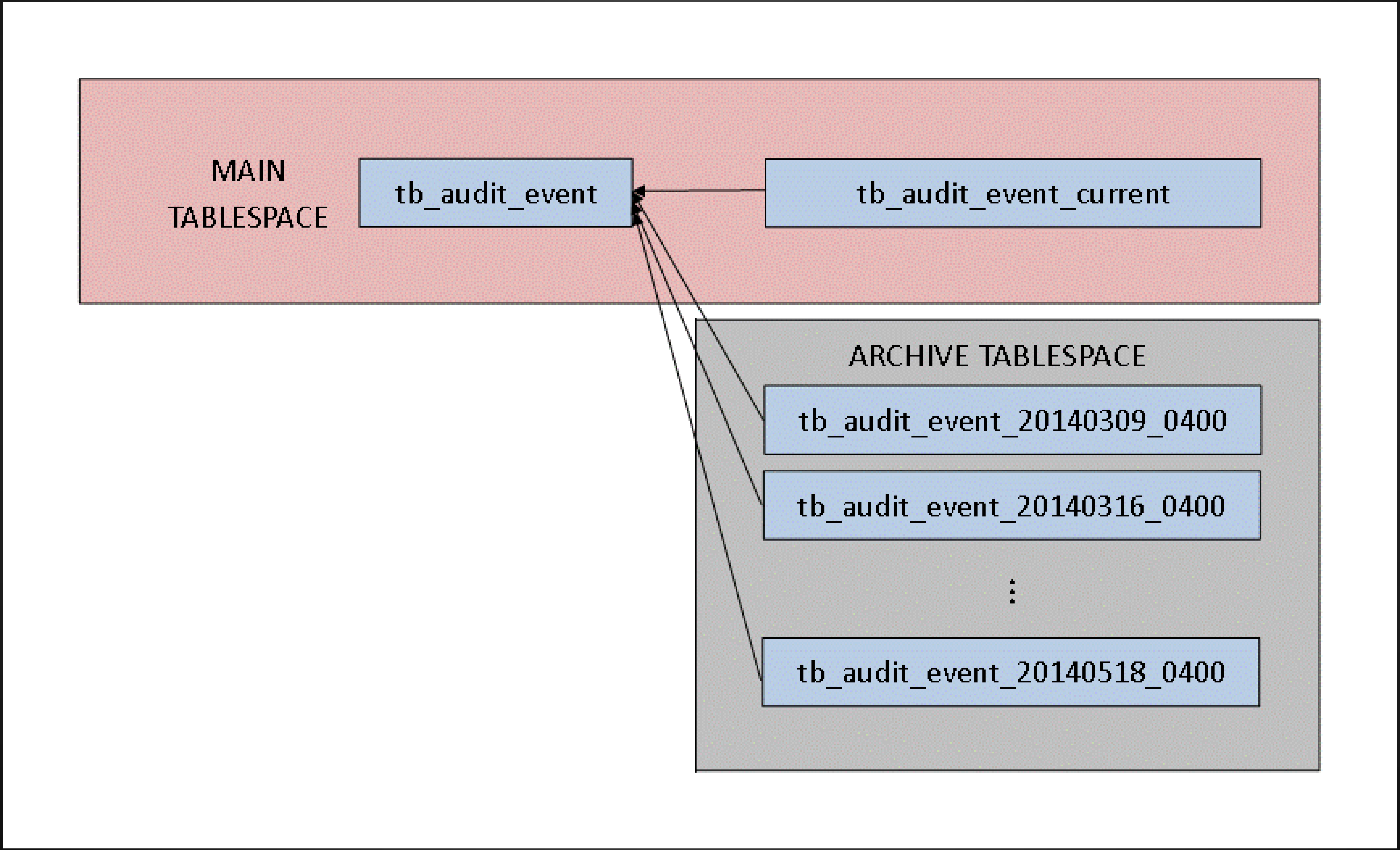
Log Rotation/Archival



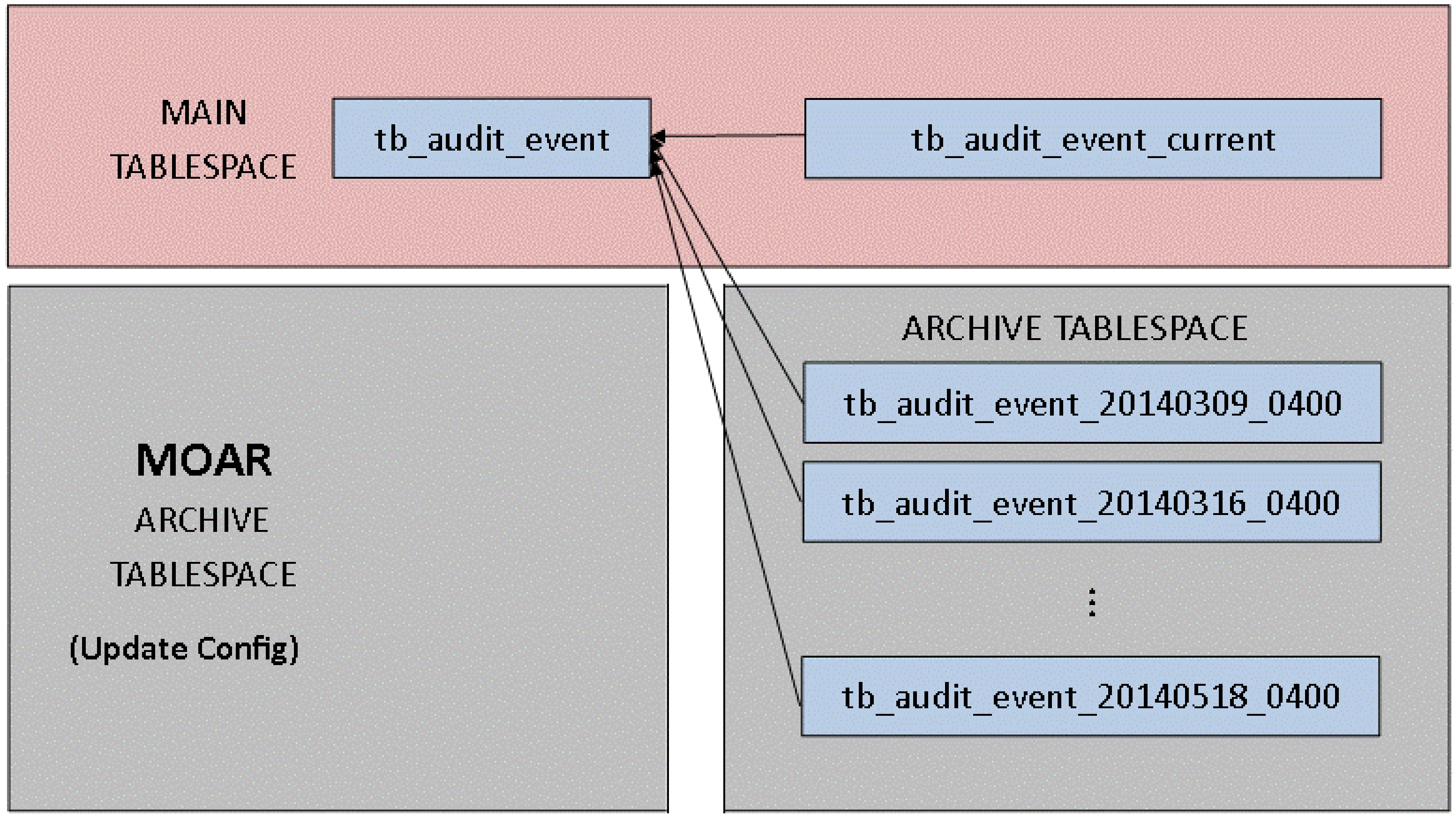
Log Rotation/Archival


MAIN
TABLESPACE
tb_audit_event
tb_audit_event_current
ARCHIVE
TABLESPACE
tb_audit_event_20140309_0400
tb_audit_event_20140316_0400
⋮
tb_audit_event_20140518_0400

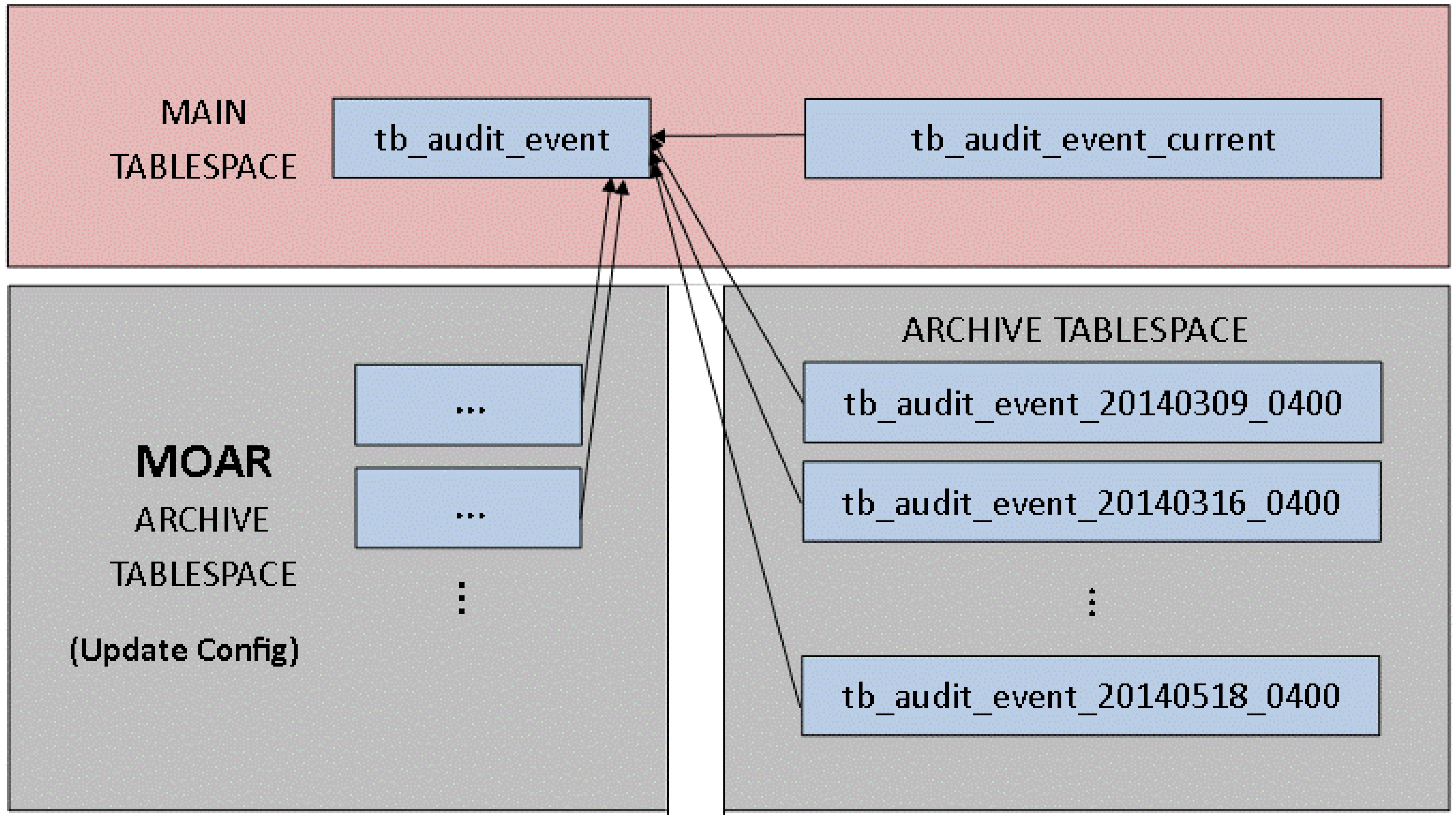
Log Rotation/Archival


MAIN TABLESPACE
tb_audit_event
tb_audit_event_current
ARCHIVE TABLESPACE
tb_audit_event_20140309_0400
tb_audit_event_20140316_0400
⋮
tb_audit_event_20140518_0400

Log Rotation/Archival



Log Rotation/Archival



Log Rotation/Archival

- `cyanaudit_log_rotate.pl`
  Log entries since last rotation become a new child partition of parent table tb_audit_event.

- `cyanaudit_dump.pl`
  Back up audit data, remove old tables.

- `cyanaudit_restore.pl`
  Restore dumps created with cyanaudit_dump.pl

Wishlist – Nailed it!

- Extension-based
- Space-efficient, organized logging
- Per-column control of logging
- Attach descriptions to events
- Scalability to years’ worth of logs
- Export / import between log & files
- Automated log maintenance
- Easy recovery from mistakes
- Plus: Released under PostgreSQL license

Cyan Audit Caveats

- PostgreSQL version compatibility:
  - >= 9.3.3: All features supported
  - < 9.3.3: No DDL triggers. After any DDL you must `select fn_update_audit_fields()`
  - < 9.2.0: Must modify postgresql.conf with `custom_variable_classes = cyanaudit`
  - < 9.1.7: Not supported
- Logs only tables with integer PK.
- Logs only public schema.
- Truncates are not logged.
- Does not store original SQL.

Cyan Audit Challenges

- Proper behavior with pg_dump/pg_restore
- Log tables using OID as PK
- Log tables in other schemas than public
- Amazon RDB – non-extension version?
- Automatic testing
- Leverage 9.4’s logical replication
- Wide use, inclusion with PostgreSQL core! YEAAH!

Questions? Comments?

Moshe Jacobson moshe@neadwerx.com

Download: http://cyanaudit.neadwerx.com

Thanks to Nead Werx, my employer, for sponsoring the development of Cyan Audit.