هدف اصلی، پیگیری و ثبت هر عملیات**INSERT**، **UPDATE**و **DELETE** روی جدول‌های `account` و `transactions`است تا امکان بررسی تاریخچه تغییرات و تحلیل فعالیت کاربران فراهم شود.

ساختار جدول و داده‌ها

- جدول اصلی `audit_log`است که در آن اطلاعات تغییرات ذخیره می‌شود.
- ستون‌ها شامل:

  - `table_name`نام جدولی که تغییر کرده.
  - `operation_type`نوع عملیات(`INSERT, UPDATE, DELETE`)
  - `record_id`= شناسه یا بخشی از رکورد تغییر یافته.
  - `old_values` و `new_values`مقادیر قبل و بعد از تغییر، به صورت **JSON** ذخیره می‌شوند.
  - `changed_by`کاربری که تغییر را اعمال کرده.
  - `changed_at`زمان دقیق تغییر.

- ایندکس‌ها روی جدول گذاشته شده تا جستجو و گزارش‌گیری سریع انجام شود.

منطق **Trigger** و فانکشن

- یک فانکشن `fn_audit_trigger`ایجاد شده که بعد از هر تغییر روی جدول‌ها اجرا می‌شود.
- رفتار فانکشن بر اساس نوع عملیات:
  1. **INSERT**مقدار قدیمی (`old_values`) خالی و مقدار جدید (`new_values`) ذخیره می‌شود.
  2. **UPDATE**مقادیر قدیمی و جدید هر دو ذخیره می‌شوند.
  3. **DELETE**مقدار قدیمی ذخیره می‌شود و مقدار جدید خالی خواهد بود.
- سپس یک رکورد شامل اطلاعات تغییر در جدول `audit_log`درج می‌شود.
- **Trigger**ها روی جدول‌های `account` و `transactions`تعریف شده‌اند تا این فانکشن پس از هر تغییر اجرا شود.

ویوهای گزارش‌گیری

- `vw_audit_account_changes`تغییرات حساب‌ها شامل موجودی و وضعیت آنها را به صورت خوانا نمایش می‌دهد.
- `vw_audit_statistics`آمار کلی از تغییرات در دیتابیس شامل تعداد عملیات، تعداد کاربران، اولین و آخرین تغییر را نشان می‌دهد.

منطق ذخیره‌سازی**JSON**

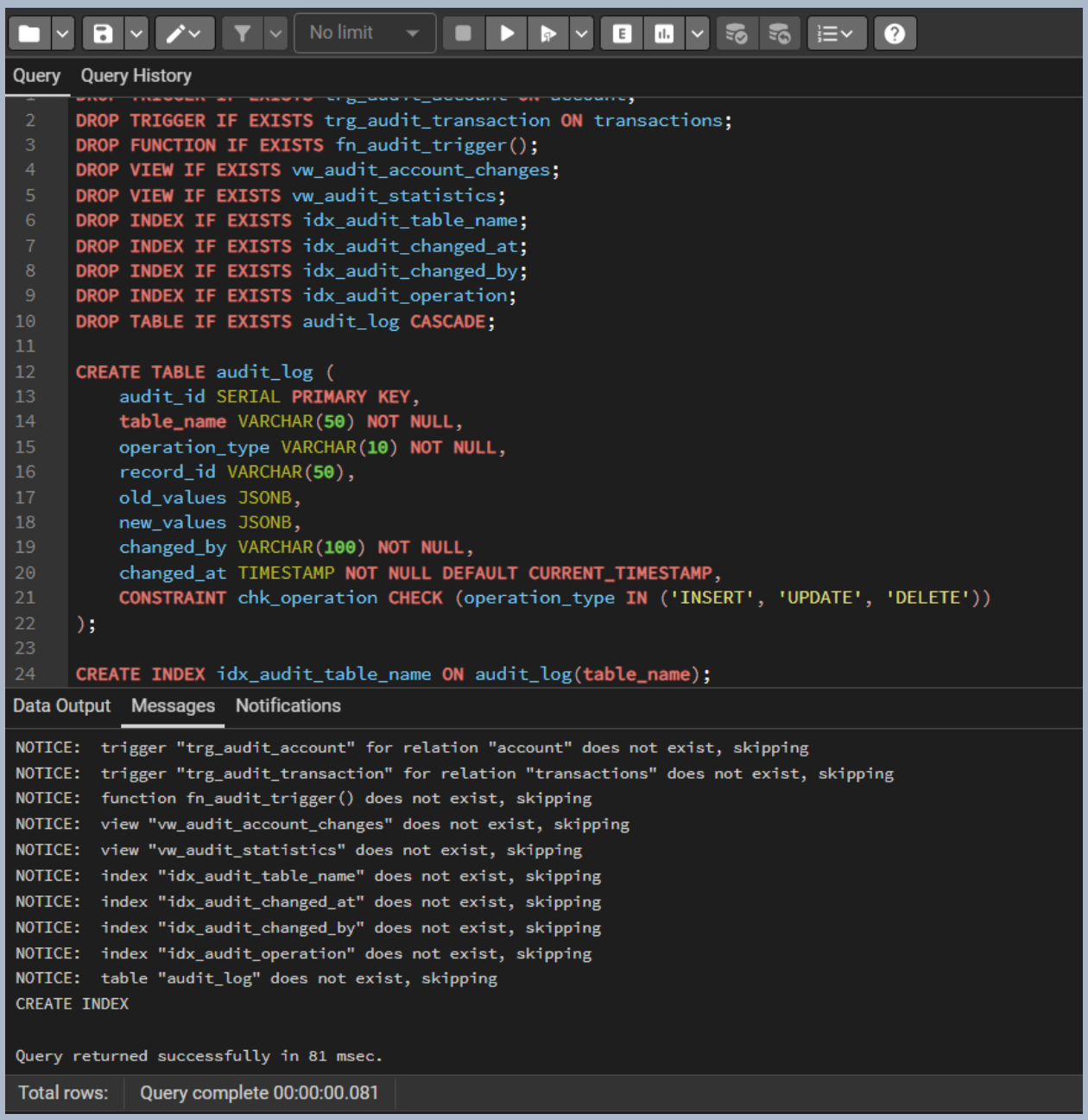
- استفاده از **JSONB** برای `old_values` و `new_values`باعث می‌شود که تمام ستون‌ها بدون نیاز به تغییر ساختار جدول **Audit** ثبت شوند.
- این روش انعطاف‌پذیر است و اگر ستون جدیدی به جدول اضافه شود، فانکشن به طور خودکار مقدار آن را در **JSON** ذخیره می‌کند.
- همچنین امکان مقایسه سریع تغییرات بین حالت قدیم و جدید فراهم و می‌توان با **SQL** مقادیر خاص را بررسی کرد.

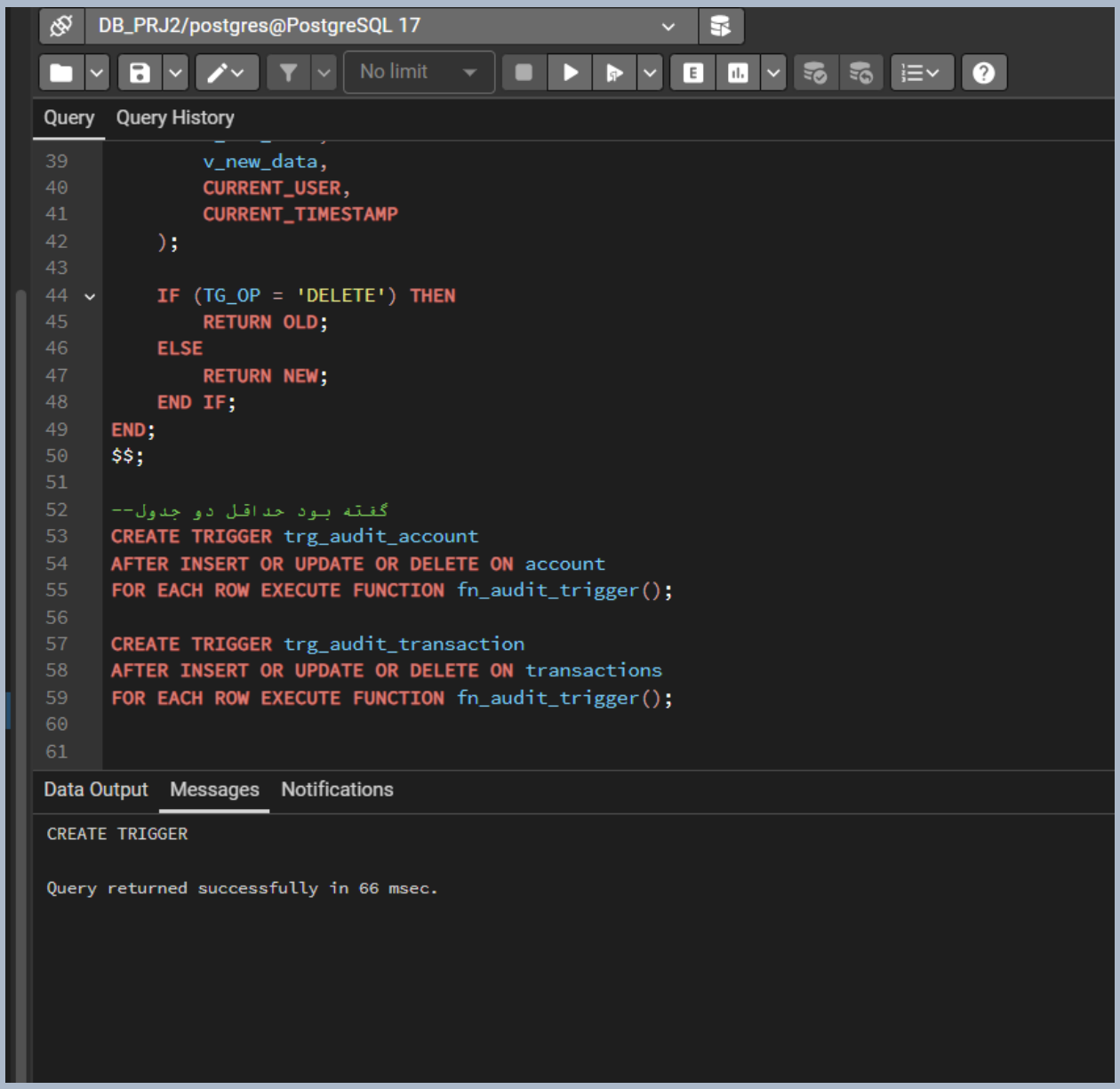
فرآیند هنگام اجرای عملیات

- **وقتی یک موجودی حساب تغییر می‌کند:**
  1. **عملیات UPDATE روی جدول `account` انجام می‌شود.**
  2. **Trigger فعال می‌شود و فانکشن `fn_audit_trigger` اجرا می‌شود.**
  3. **اطلاعات قبل و بعد تغییر به صورت JSON در `audit_log` ذخیره می‌شوند.**
- **وقتی یک تراکنش جدید ثبت می‌شود:**
  1. **عملیات INSERT روی جدول `transactions` انجام می‌شود.**
  2. **Trigger اجرا می‌شود و رکورد جدید تراکنش در `audit_log` ثبت می‌شود.**
- **هر تغییر با اطلاعات کاربر و زمان دقیق ثبت می‌شود، حتی اگر تغییر توسط سیستم یا کاربر دیگر باشد.**

فرآیند تست و بررسی صحت سیستم(برای توضیح بالا)

- **برای اطمینان از عملکرد سیستم، تست‌هایی با DO block نوشته شد:**
  - **تست تغییر موجودی حساب :موجودی یک حساب افزایش پیدا می‌کند و سیستم بررسی می‌کند که رکورد Audit ثبت شده باشد.**
  - **تست ثبت تراکنش جدید :تراکنش INSERT می‌شود و بررسی می‌شود که trigger فعال شده و رکورد Audit ثبت شده باشد.**
- **این تست‌ها تضمین می‌کنند که سیستم در شرایط واقعی درست عمل می‌کند.**


No limit
Query
Query History
DROP TRIGGER IF EXISTS trg_audit_transaction ON transactions;
DROP FUNCTION IF EXISTS fn_audit_trigger();
DROP VIEW IF EXISTS vw_audit_account_changes;
DROP VIEW IF EXISTS vw_audit_statistics;
DROP INDEX IF EXISTS idx_audit_table_name;
DROP INDEX IF EXISTS idx_audit_changed_at;
DROP INDEX IF EXISTS idx_audit_changed_by;
DROP INDEX IF EXISTS idx_audit_operation;
DROP TABLE IF EXISTS audit_log CASCADE;
CREATE TABLE audit_log (
audit_id SERIAL PRIMARY KEY,
table_name VARCHAR(50) NOT NULL,
operation_type VARCHAR(10) NOT NULL,
record_id VARCHAR(50),
old_values JSONB,
new_values JSONB,
changed_by VARCHAR(100) NOT NULL,
changed_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
CONSTRAINT chk_operation CHECK (operation_type IN ('INSERT', 'UPDATE', 'DELETE'))
);
CREATE INDEX idx_audit_table_name ON audit_log(table_name);
Data Output
Messages
Notifications
NOTICE: trigger "trg_audit_account" for relation "account" does not exist, skipping
NOTICE: trigger "trg_audit_transaction" for relation "transactions" does not exist, skipping
NOTICE: function fn_audit_trigger() does not exist, skipping
NOTICE: view "vw_audit_account_changes" does not exist, skipping
NOTICE: view "vw_audit_statistics" does not exist, skipping
NOTICE: index "idx_audit_table_name" does not exist, skipping
NOTICE: index "idx_audit_changed_at" does not exist, skipping
NOTICE: index "idx_audit_changed_by" does not exist, skipping
NOTICE: index "idx_audit_operation" does not exist, skipping
NOTICE: table "audit_log" does not exist, skipping
CREATE INDEX
Query returned successfully in 81 msec.
Total rows:
Query complete 00:00:00.081


DB_PRJ2/postgres@PostgreSQL 17
No limit
Query
Query History
v_new_data,
CURRENT_USER,
CURRENT_TIMESTAMP
);
IF (TG_OP = 'DELETE') THEN
RETURN OLD;
ELSE
RETURN NEW;
END IF;
END;
$$;
--گفته بود حداقل دو جدول
CREATE TRIGGER trg_audit_account
AFTER INSERT OR UPDATE OR DELETE ON account
FOR EACH ROW EXECUTE FUNCTION fn_audit_trigger();
CREATE TRIGGER trg_audit_transaction
AFTER INSERT OR UPDATE OR DELETE ON transactions
FOR EACH ROW EXECUTE FUNCTION fn_audit_trigger();
Data Output
Messages
Notifications
CREATE TRIGGER
Query returned successfully in 66 msec.

Query | Query History

```
12      (old_values->>'balance')::NUMERIC AS old_balance,
13      (new_values->>'balance')::NUMERIC AS new_balance,
14      (new_values->>'balance')::NUMERIC - (old_values->>'balance')::NUMERIC AS balance_change,
15      (old_values->>'status') AS old_status,
16      (new_values->>'status') AS new_status
17  FROM audit_log
18  WHERE table_name = 'account'
19    AND operation_type <> 'DELETE'
20  ORDER BY changed_at DESC;
21
22  -- آمار کلی Audit
23  CREATE OR REPLACE VIEW vw_audit_statistics AS
24  SELECT
25      table_name,
26      operation_type,
27      COUNT(*) AS operation_count,
28      COUNT(DISTINCT changed_by) AS unique_users,
29      MIN(changed_at) AS first_change,
30      MAX(changed_at) AS last_change
31  FROM audit_log
32  GROUP BY table_name, operation_type
33  ORDER BY table_name, operation_type;
34
```

Data Output | Messages | Notifications

```
CREATE VIEW

Query returned successfully in 67 msec.
```

Query | Query History

```sql
--تست ها
-- تغییر موجودی حساب
DO $$
DECLARE
    v_audit_count_before INTEGER;
    v_audit_count_after INTEGER;
BEGIN
    SELECT COUNT(*) INTO v_audit_count_before FROM audit_log WHERE table_name = 'account';

    UPDATE account
    SET balance = balance + 100000
    WHERE account_number = '1001000100010001';

    SELECT COUNT(*) INTO v_audit_count_after FROM audit_log WHERE table_name = 'account';

    IF v_audit_count_after > v_audit_count_before THEN
        RAISE NOTICE 'تغییر موجودی ثبت شد';
    ELSE
        RAISE EXCEPTION 'تست موجودی شکست خورد';
    END IF;
END $$;
```

Data Output | Messages | Notifications

```
NOTICE:  تغییر موجودی ثبت شد
DO

Query returned successfully in 67 msec.
```

Query | Query History

```sql
-- درج تراکنش جدید
DO $$
BEGIN
    INSERT INTO transactions (
        transaction_type, amount, description,
        source_account, destination_account, status
    )
    VALUES (
        'TRANSFER', 50000, 'تست Audit System',
        '1001000100010001', '2002000200020001', 'COMPLETED'
    );

    IF EXISTS (
        SELECT 1 FROM audit_log
        WHERE table_name = 'transactions'
        AND operation_type = 'INSERT'
        AND changed_at >= CURRENT_TIMESTAMP - INTERVAL '5 seconds'
    ) THEN
        RAISE NOTICE 'تراکنش ثبت شد';
    ELSE
        RAISE EXCEPTION 'تست تراکنش شکست خورد';
    END IF;
END $$;
```

Data Output | Messages | Notifications

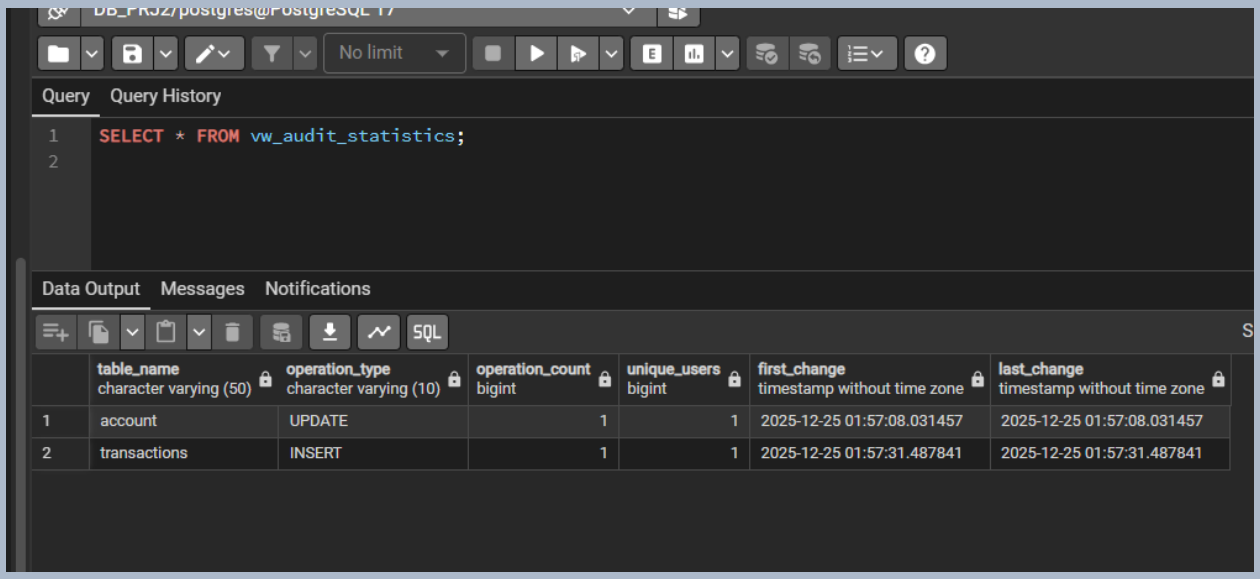
```
NOTICE:  تراکنش ثبت شد
DO

Query returned successfully in 59 msec.
```

```sql
SELECT * FROM vw_audit_account_changes;
```

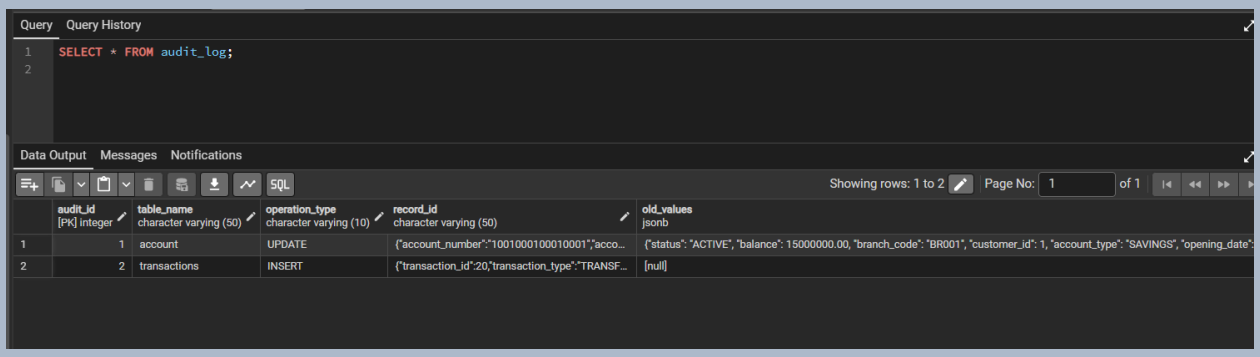
Data Output | Messages | Notifications

Showing rows: 1 to 1 | Page No: 1

	audit_id integer	operation_type character varying (10)	changed_by character varying (100)	changed_at timestamp without time zone	account_number text	old_balance numeric	new_balance numeric	balance_change numeric	old_status text	new_status text
1	1	UPDATE	postgres	2025-12-25 01:57:08.031457	10010001000100...	15000000.00	15100000.00	100000.00	ACTIVE	ACTIVE



	table_name character varying (50)	operation_type character varying (10)	operation_count bigint	unique_users bigint	first_change timestamp without time zone	last_change timestamp without time zone
1	account	UPDATE	1	1	2025-12-25 01:57:08.031457	2025-12-25 01:57:08.031457
2	transactions	INSERT	1	1	2025-12-25 01:57:31.487841	2025-12-25 01:57:31.487841



	audit_id [PK] integer	table_name character varying (50)	operation_type character varying (10)	record_id character varying (50)	old_values jsonb
1	1	account	UPDATE	{"account_number":"1001000100010001","acco...	{"status": "ACTIVE", "balance": 15000000.00, "branch_code": "BR001", "customer_id": 1, "account_type": "SAVINGS", "opening_date":
2	2	transactions	INSERT	{"transaction_id":20,"transaction_type":"TRANSF...	[null]