Bonus Lab – Advanced Database Features

Author: Kaldanova Lina
Course: Database
Instructor: Kuralbaev Aibek

1. Transaction Isolation & ACID Compliance

1.1 process_transfer

The process_transfer procedure uses an **internal BEGIN…EXCEPTION block**, which acts like a **SAVEPOINT**.
This ensures:

only the transfer logic is rolled back when an error occurs;

the outer transaction remains intact;

all operations are atomic, consistent, isolated, and durable.

The procedure validates:

account existence

customer status

currency match

balance sufficiency

daily limit (amount_kzt)

All failures are logged into audit_log with detailed context.

1.2 process_salary_batch

Key ACID features:

Advisory Lock

pg_try_advisory_lock() ensures **serialized execution** of salary batches per company account.

If another batch is in progress:

ERR_BATCH_LOCKED: another salary batch in progress

Per-employee SAVEPOINT

Each payment is wrapped in a sub-block:

```
BEGIN
   ...
EXCEPTION
```

```
    -- this payment is logged and skipped
END;
```

This means:

- 1 invalid employee DOES NOT stop the entire batch
- Good payroll records still complete successfully

Batch-level atomicity

Successful and failed payments are aggregated into:

- successful_count
- failed_count
- failed_details (JSONB array)

The final update to account balances is performed in a single operation.

2. Views with Window Functions

2.1 customer_balance_summary

Includes:

- per-customer total balance (using SUM() OVER (PARTITION BY ...))
- daily limit utilization (%)
- ranking customers by total balance (RANK())

2.2 daily_transaction_report

Uses:

- running totals (SUM() OVER)
- day-over-day growth using LAG()
- average daily volume

2.3 suspicious_activity_view

Security barrier ensures restricted access.
The view flags:

- unusually large amounts
- 10 transactions per hour
- transactions under 1 minute apart

3. Index Strategy

The indexing strategy focuses on accelerating the most frequent and most expensive operations in the system:

Account lookups:
A B-tree index on account_number ensures instant access during transfers and salary batches.

Customer search:
A functional index on lower(email) speeds up case-insensitive email queries.

Transaction reporting:
A composite index on (from_account_id, status, type, created_at) makes analytical views fast and allows PostgreSQL to use index-only scans.

Audit log analysis:
A hash index on table_name and a GIN index on JSONB fields enable quick filtering and searching through large audit logs.

4. EXPLAIN ANALYZE Outputs

4.1 account_number lookup

```
Index Scan using idx_acc_num_btree on accounts
  Index Cond: (account_number = 'KZ09601A241001007211')
Execution Time: 0.080 ms
```

4.2 lower(email) lookup

```
Index Scan using idx_customers_email_lower on customers
  Index Cond: (lower(email) = 'lina.kaldanova@gmail.com')
Execution Time: 0.060 ms
```

4.3 transactions reporting

```
Index Scan using idx_transactions_from_status_type_created_inc
  Index Cond: (from_account_id = 3 AND status='completed' AND type='transfer')
Execution Time: 0.090 ms
```

4.4 audit log filtering

```
Bitmap Index Scan on idx_audit_log_jsonb_gin
  Recheck Cond: ((old_values || new_values) @> '{"code":"ERR_INVALID_AMOUNT"}')
Execution Time: 0.140 ms
```