Pri pouziti 30 ATM, 30 Traderu, 60 Uctu, 90 karet, a 1500 transakci lze dosahnout velkeho zrychleni pri nasledujici zmene (rozdil bude jeste vice signifikantni pri vetsim objemu dat).

```
EXPLAIN ANALYZE
UPDATE account a
SET balance = (balance + balance * 0.01)
WHERE a.id_account in (
    SELECT a.id_account
    FROM (                SELECT c.id_account
                          FROM transaction t
                            JOIN card c ON
                                        t.id_card = c.id_account
                          WHERE t.date >= '2017/05/01' AND t.date <= '2017/06/01'
                          GROUP BY c.id_card
                          HAVING COUNT(c) >= 3) as eligible_cards
    WHERE a.id_account = eligible_cards.id_account);
```

	QUERY PLAN
1	Update on account a (cost=0.00..2318.90 rows=30 width=179) (actual time=11.968..11.968 rows=0 loops=1)
2	-> Seq Scan on account a (cost=0.00..2318.90 rows=30 width=179) (actual time=11.966..11.966 rows=0 loops=1)
3	Filter: (SubPlan 1)
4	Rows Removed by Filter: 60
5	SubPlan 1
6	-> Subquery Scan on eligible_cards (cost=0.14..77.05 rows=2 width=0) (actual time=0.199..0.199 rows=0 loops=60)
7	-> GroupAggregate (cost=0.14..77.03 rows=2 width=69) (actual time=0.198..0.198 rows=0 loops=60)
8	Filter: (count(c.*) >= 3)
9	-> Nested Loop (cost=0.14..76.99 rows=2 width=69) (actual time=0.198..0.198 rows=0 loops=60)
10	-> Index Scan using card_pkey on card c (cost=0.14..13.72 rows=2 width=69) (actual time=0.047..0.058 rows=2 loops=60)
11	Filter: (id_account = a.id_account)
12	Rows Removed by Filter: 88
13	-> Materialize (cost=0.00..63.26 rows=1 width=8) (actual time=0.093..0.093 rows=0 loops=90)
14	-> Seq Scan on transaction t (cost=0.00..63.25 rows=1 width=8) (actual time=0.218..0.218 rows=0 loops=38)
15	Filter: ((date >= '2017-05-01'::date) AND (date <= '2017-06-01'::date) AND (id_card = a.id_account))
16	Rows Removed by Filter: 1500
17	Total runtime: 12.026 ms

```
EXPLAIN ANALYZE
UPDATE account a
SET balance = (balance + balance * 0.01)
WHERE EXISTS (
    SELECT NULL
    FROM (                SELECT c.id_account
                          FROM transaction t
                            JOIN card c ON
                                        t.id_card = c.id_account
                          WHERE t.date >= '2017/05/01' AND t.date <= '2017/06/01'
                          GROUP BY c.id_card
                          HAVING COUNT(c) >= 3) as eligible_cards
    WHERE a.id_account = eligible_cards.id_account);
```

	QUERY PLAN
1	Update on account a (cost=61.80..64.58 rows=1 width=211) (actual time=0.303..0.303 rows=0 loops=1)
2	-> Hash Semi Join (cost=61.80..64.58 rows=1 width=211) (actual time=0.302..0.302 rows=0 loops=1)
3	Hash Cond: (a.id_account = eligible_cards.id_account)
4	-> Seq Scan on account a (cost=0.00..2.60 rows=60 width=179) (actual time=0.007..0.007 rows=1 loops=1)
5	-> Hash (cost=61.79..61.79 rows=1 width=40) (actual time=0.281..0.281 rows=0 loops=1)
6	Buckets: 1024 Batches: 1 Memory Usage: 0kB
7	-> Subquery Scan on eligible_cards (cost=61.77..61.79 rows=1 width=40) (actual time=0.281..0.281 rows=0 loops=1)
8	-> HashAggregate (cost=61.77..61.78 rows=1 width=69) (actual time=0.280..0.280 rows=0 loops=1)
9	Filter: (count(c.*) >= 3)
10	-> Hash Join (cost=59.51..61.76 rows=1 width=69) (actual time=0.279..0.279 rows=0 loops=1)
11	Hash Cond: (c.id_account = t.id_card)
12	-> Seq Scan on card c (cost=0.00..1.90 rows=90 width=69) (actual time=0.007..0.007 rows=1 loops=1)
13	-> Hash (cost=59.50..59.50 rows=1 width=8) (actual time=0.264..0.264 rows=0 loops=1)
14	Buckets: 1024 Batches: 1 Memory Usage: 0kB
15	-> Seq Scan on transaction t (cost=0.00..59.50 rows=1 width=8) (actual time=0.264..0.264 rows=0 loops=1)
16	Filter: ((date >= '2017-05-01'::date) AND (date <= '2017-06-01'::date))
17	Rows Removed by Filter: 1500
18	Total runtime: 0.374 ms