STU Bratislava, Fakulta informatiky a informačných technológií

Databázové systémy

Zadanie 6 – ORM

Meno: Andrii Rybak

AIS ID: 105840

Cvičiaci: Ing. Vladimír Kunštár

Akademicky rok: 2021/2022

Content

1. /patches

Initial:

```
WITH table1
     AS (SELECT NAME                                          AS
                patch_version,
                Cast(Extract(epoch FROM release_date) AS INTEGER) AS
                patch_start_date
         FROM   patches
         ORDER  BY patch_version ASC),
     table2
     AS (SELECT patch_version,
                patch_start_date,
                Lead(patch_start_date, 1)
                  OVER (
                    ORDER BY patch_version) patch_end_date
         FROM   table1)
SELECT patch_version,
       patch_start_date,
       patch_end_date,
       mts.id                          AS match_id,
       Round(mts.duration / 60.00, 2) AS match_duration
FROM   table2
       LEFT JOIN matches AS mts
              ON mts.start_time >= patch_start_date
                 AND mts.start_time < patch_end_date
```

	QUERY PLAN text
1	Nested Loop Left Join (cost=1.69..18560.07 rows=105556 width=76) (actual time=1917.938..19475.041 rows=50005 loops=1)
2	[...] Join Filter: ((mts.start_time >= ((date_part('epoch'::text, patches.release_date))::integer)) AND (mts.start_time < (lead(((date_part('epoch'::text, patches.release_date))::integer), 1) OVER (?))))
3	[...] Rows Removed by Join Filter: 900000
4	[...] -> WindowAgg (cost=1.69..2.21 rows=19 width=40) (actual time=0.554..1.326 rows=19 loops=1)
5	[...] -> Sort (cost=1.69..1.74 rows=19 width=36) (actual time=0.506..0.717 rows=19 loops=1)
6	[...] Sort Key: patches.name
7	[...] Sort Method: quicksort Memory: 25kB
8	[...] -> Seq Scan on patches (cost=0.00..1.28 rows=19 width=36) (actual time=0.041..0.257 rows=19 loops=1)
9	[...] -> Materialize (cost=0.00..1266.00 rows=50000 width=12) (actual time=0.011..525.040 rows=50000 loops=19)
10	[...] -> Seq Scan on matches mts (cost=0.00..1016.00 rows=50000 width=12) (actual time=0.022..487.312 rows=50000 loops=1)
11	Planning Time: 0.180 ms
12	Execution Time: 19953.505 ms

ORM generated:

```
SELECT anon_1.patch_version                AS anon_1_patch_version,
       anon_1.patch_start_date             AS anon_1_patch_start_date,
       anon_1.patch_end_date               AS anon_1_patch_end_date,
       matches.id                          AS match_id,
       Round(matches.duration / 60.0, 2) AS match_duration
FROM   (SELECT anon_2.patch_version                AS patch_version,
               anon_2.patch_start_date             AS patch_start_date,
               Lead(anon_2.patch_start_date, 1)
                 OVER (
                   ORDER BY anon_2.patch_version ) AS patch_end_date
        FROM   (SELECT patches.NAME
                       AS
                               patch_version,
                       Cast(Extract(epoch FROM patches.release_date) AS INTEGER)
                       AS
                               patch_start_date
                FROM   patches) AS anon_2) AS anon_1
       LEFT OUTER JOIN matches
                    ON matches.start_time >= anon_1.patch_start_date
                       AND matches.start_time < anon_1.patch_end_date
```

	QUERY PLAN text
1	Nested Loop Left Join (cost=1.59..18559.98 rows=105556 width=76) (actual time=2063.780..19368.389 rows=50005 loops=1)
2	[...] Join Filter: ((matches.start_time >= ((date_part('epoch'::text, patches.release_date))::integer)) AND (matches.start_time < (lead((date_part('epoch'::text, patches.release_date))::integer, 1) OVER (?))))
3	[...] Rows Removed by Join Filter: 900000
4	[...] -> WindowAgg (cost=1.59..2.12 rows=19 width=40) (actual time=0.682..1.583 rows=19 loops=1)
5	[...] -> Sort (cost=1.59..1.64 rows=19 width=40) (actual time=0.592..0.793 rows=19 loops=1)
6	[...] Sort Key: patches.name
7	[...] Sort Method: quicksort Memory: 25kB
8	[...] -> Seq Scan on patches (cost=0.00..1.19 rows=19 width=40) (actual time=0.037..0.300 rows=19 loops=1)
9	[...] -> Materialize (cost=0.00..1266.00 rows=50000 width=12) (actual time=0.011..520.889 rows=50000 loops=19)
10	[...] -> Seq Scan on matches (cost=0.00..1016.00 rows=50000 width=12) (actual time=0.019..524.267 rows=50000 loops=1)
11	Planning Time: 0.196 ms
12	Execution Time: 19834.763 ms

Summary

ORM generated almost the same query as original one, so it is impossible to get significantly different performance from it. The only difference is that in original query was used WITH to make subtable, but in the generated query was used nested SELECT. However, query plan has the same structure and almost the same values in both cases.

The most expensive operation is left join, which took ~19200 msec for all rows. Join operation took more than 90% of all query execution time.

2. /players/14944/game_exp

Initial:

```
SELECT pl.id,
       CASE
         WHEN pl.nick IS NULL THEN 'unknown'
         ELSE pl.nick
       end                                AS player_nick,
       hr.localized_name                  AS hero_localized_name,
       Round(mt.duration / 60.00, 2)      AS match_duration_minutes,
       Coalesce(mt_pl_dt.xp_hero, 0)
       + Coalesce(mt_pl_dt.xp_creep, 0)
       + Coalesce(mt_pl_dt.xp_other, 0)
       + Coalesce(mt_pl_dt.xp_roshan, 0) AS experiences_gained,
       mt_pl_dt.level                     AS level_gained,
       CASE
         WHEN player_slot >= 0
              AND player_slot <= 4
              AND radiant_win IS TRUE THEN true
         WHEN player_slot >= 128
              AND player_slot <= 132
              AND radiant_win IS FALSE THEN true
         ELSE false
       end                                AS winner,
       mt.id                              AS match_id
FROM   players AS pl
       JOIN matches_players_details AS mt_pl_dt
         ON mt_pl_dt.player_id = pl.id
       JOIN heroes AS hr
         ON mt_pl_dt.hero_id = hr.id
       JOIN matches AS mt
         ON mt_pl_dt.match_id = mt.id
WHERE  pl.id = 14944
ORDER  BY match_id ASC
```

	QUERY PLAN text
1	Gather Merge (cost=14228.08..14229.15 rows=9 width=91) (actual time=26.443..29.196 rows=13 loops=1)
2	[...] Workers Planned: 3
3	[...] Workers Launched: 3
4	[...] -> Sort (cost=13228.04..13228.05 rows=3 width=91) (actual time=19.724..19.764 rows=3 loops=4)
5	[...] Sort Key: mt.id
6	[...] Sort Method: quicksort Memory: 25kB
7	[...] Worker 0: Sort Method: quicksort Memory: 25kB
8	[...] Worker 1: Sort Method: quicksort Memory: 25kB
9	[...] Worker 2: Sort Method: quicksort Memory: 25kB
10	[...] -> Nested Loop (cost=4.25..13228.02 rows=3 width=91) (actual time=7.450..19.625 rows=3 loops=4)
11	[...] -> Hash Join (cost=3.96..13203.02 rows=3 width=53) (actual time=7.363..19.390 rows=3 loops=4)
12	[...] Hash Cond: (mt_pl_dt.hero_id = hr.id)
13	[...] -> Nested Loop (cost=0.42..13199.47 rows=3 width=47) (actual time=4.793..16.747 rows=3 loops=4)
14	[...] -> Parallel Seq Scan on matches_players_details mt_pl_dt (cost=0.00..13174.13 rows=3 width=36) (actual time=4.725..16.438 rows=3 loops=4)
15	[...] Filter: (player_id = 14944)
16	[...] Rows Removed by Filter: 124997
17	[...] -> Index Scan using players_pk on players pl (cost=0.42..8.44 rows=1 width=15) (actual time=0.020..0.033 rows=1 loops=13)
18	[...] Index Cond: (id = 14944)
19	[...] -> Hash (cost=2.13..2.13 rows=113 width=14) (actual time=2.407..2.419 rows=113 loops=4)
20	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
21	[...] -> Seq Scan on heroes hr (cost=0.00..2.13 rows=113 width=14) (actual time=0.049..1.211 rows=113 loops=4)
22	[...] -> Index Scan using matches_pk on matches mt (cost=0.29..8.31 rows=1 width=9) (actual time=0.026..0.028 rows=1 loops=13)
23	[...] Index Cond: (id = mt_pl_dt.match_id)
24	Planning Time: 0.581 ms
25	Execution Time: 29.667 ms

ORM generated:

```
SELECT players.id                                        AS players_id,
       CASE
         WHEN ( players.nick IS NULL ) THEN 'unknown'
         ELSE players.nick
       END                                               AS player_nick,
       heroes.localized_name                             AS hero_localized_name,
       Round(matches.duration / 60.0, 2)                 AS
       match_duration_minutes,
       COALESCE(matches_players_details.xp_hero, 0)
       + COALESCE(matches_players_details.xp_creep, 0)
       + COALESCE(matches_players_details.xp_other, 0)
       + COALESCE(matches_players_details.xp_roshan, 0) AS experiences_gained,
       matches_players_details.level                     AS level_gained,
       CASE
         WHEN ( matches_players_details.player_slot >= 0
                AND matches_players_details.player_slot <= 4
                AND matches.radiant_win = true ) THEN true
         WHEN ( matches_players_details.player_slot >= 128
                AND matches_players_details.player_slot <= 132
                AND matches.radiant_win = false ) THEN true
         ELSE false
       END                                               AS winner,
       matches.id                                        AS match_id
FROM   players
       JOIN matches_players_details
         ON matches_players_details.player_id = players.id
       JOIN heroes
         ON matches_players_details.hero_id = heroes.id
       JOIN matches
         ON matches_players_details.match_id = matches.id
WHERE  players.id = 14944
ORDER  BY matches.id ASC
```

	QUERY PLAN text
1	Gather Merge (cost=14228.08..14229.15 rows=9 width=91) (actual time=23.102..25.890 rows=13 loops=1)
2	[...] Workers Planned: 3
3	[...] Workers Launched: 3
4	[...] -> Sort (cost=13228.04..13228.05 rows=3 width=91) (actual time=18.097..18.151 rows=3 loops=4)
5	[...] Sort Key: matches.id
6	[...] Sort Method: quicksort Memory: 25kB
7	[...] Worker 0: Sort Method: quicksort Memory: 25kB
8	[...] Worker 1: Sort Method: quicksort Memory: 25kB
9	[...] Worker 2: Sort Method: quicksort Memory: 25kB
10	[...] -> Nested Loop (cost=4.25..13228.02 rows=3 width=91) (actual time=9.444..18.005 rows=3 loops=4)
11	[...] -> Hash Join (cost=3.96..13203.02 rows=3 width=53) (actual time=9.370..17.772 rows=3 loops=4)
12	[...] Hash Cond: (matches_players_details.hero_id = heroes.id)
13	[...] -> Nested Loop (cost=0.42..13199.47 rows=3 width=47) (actual time=7.429..15.759 rows=3 loops=4)
14	[...] -> Parallel Seq Scan on matches_players_details (cost=0.00..13174.13 rows=3 width=36) (actual time=7.365..15.486 rows=3 loops=4)
15	[...] Filter: (player_id = 14944)
16	[...] Rows Removed by Filter: 124997
17	[...] -> Index Scan using players_pk on players (cost=0.42..8.44 rows=1 width=15) (actual time=0.021..0.033 rows=1 loops=13)
18	[...] Index Cond: (id = 14944)
19	[...] -> Hash (cost=2.13..2.13 rows=113 width=14) (actual time=2.430..2.439 rows=113 loops=3)
20	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
21	[...] -> Seq Scan on heroes (cost=0.00..2.13 rows=113 width=14) (actual time=0.042..1.210 rows=113 loops=3)
22	[...] -> Index Scan using matches_pk on matches (cost=0.29..8.31 rows=1 width=9) (actual time=0.026..0.028 rows=1 loops=13)
23	[...] Index Cond: (id = matches_players_details.match_id)
24	Planning Time: 0.558 ms
25	Execution Time: 26.314 ms

Summary

ORM generated absolutely the same query as original one, so it is impossible to get different performance from it.

Generally, the most expensive operations are JOIN, total cost of hash join is 13203 and the total cost of the query is 14229, while the execution time of initial query is 29.667 ms and the execution time of the ORM generated one – 26.314 ms. Such difference was caused by cache usage (the database cached some data after the initial query execution, so the ORM generated one was executed using this cached data)

3. /players/14944/game_objectives

Initial:

```
SELECT pl.id,
       CASE
         WHEN pl.nick IS NULL THEN 'unknown'
         ELSE pl.nick
       END                                  AS player_nick,
       hr.localized_name                    AS hero_localized_name,
       mt.id                                AS match_id,
       COALESCE(gm_obj.subtype, 'NO_ACTION') AS hero_action,
       Count(*)
FROM   players AS pl
       JOIN matches_players_details AS mt_pl_dt
         ON mt_pl_dt.player_id = pl.id
       JOIN heroes AS hr
         ON mt_pl_dt.hero_id = hr.id
       JOIN matches AS mt
         ON mt_pl_dt.match_id = mt.id
       LEFT JOIN game_objectives AS gm_obj
              ON mt_pl_dt.id = gm_obj.match_player_detail_id_1
WHERE  pl.id = 14944
GROUP  BY pl.id,
          hr.localized_name,
          mt.id,
          gm_obj.subtype
ORDER  BY match_id ASC
```

1	GroupAggregate (cost=34801.70..34802.28 rows=23 width=114) (actual time=8666.325..8666.910 rows=16 loops=1)
2	[...] Group Key: mt.id, pl.id, hr.localized_name, gm_obj.subtype
3	[...] -> Sort (cost=34801.70..34801.76 rows=23 width=53) (actual time=8666.272..8666.480 rows=18 loops=1)
4	[...] Sort Key: mt.id, hr.localized_name, gm_obj.subtype
5	[...] Sort Method: quicksort Memory: 26kB
6	[...] -> Nested Loop (cost=21591.84..34801.18 rows=23 width=53) (actual time=8652.414..8665.983 rows=18 loops=1)
7	[...] -> Index Scan using players_pk on players pl (cost=0.42..8.44 rows=1 width=15) (actual time=0.021..0.042 rows=1 loops=1)
8	[...] Index Cond: (id = 14944)
9	[...] -> Gather (cost=21591.42..34792.51 rows=23 width=42) (actual time=8652.355..8668.750 rows=18 loops=1)
10	[...] Workers Planned: 3
11	[...] Workers Launched: 3
12	[...] -> Parallel Hash Left Join (cost=20591.42..33790.21 rows=7 width=42) (actual time=8647.120..8658.855 rows=4 loops=4)
13	[...] Hash Cond: (mt_pl_dt.id = gm_obj.match_player_detail_id_1)
14	[...] -> Nested Loop (cost=3.83..13202.60 rows=3 width=22) (actual time=10.547..25.551 rows=4 loops=3)
15	[...] -> Hash Join (cost=3.54..13177.68 rows=3 width=22) (actual time=10.432..25.168 rows=4 loops=3)
16	[...] Hash Cond: (mt_pl_dt.hero_id = hr.id)
17	[...] -> Parallel Seq Scan on matches_players_details mt_pl_dt (cost=0.00..13174.13 rows=3 width=16) (actual time=6.929..21.554 rows=4 loops=3)
18	[...] Filter: (player_id = 14944)
19	[...] Rows Removed by Filter: 166662
20	[...] -> Hash (cost=2.13..2.13 rows=113 width=14) (actual time=3.398..3.409 rows=113 loops=3)
21	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
22	[...] -> Seq Scan on heroes hr (cost=0.00..2.13 rows=113 width=14) (actual time=0.060..1.707 rows=113 loops=3)
23	[...] -> Index Only Scan using matches_pk on matches mt (cost=0.29..8.31 rows=1 width=4) (actual time=0.041..0.043 rows=1 loops=13)
24	[...] Index Cond: (id = mt_pl_dt.match_id)
25	[...] Heap Fetches: 13
26	[...] -> Parallel Hash (cost=15856.15..15856.15 rows=378515 width=28) (actual time=8629.937..8629.946 rows=293349 loops=4)
27	[...] Buckets: 2097152 Batches: 1 Memory Usage: 60704kB
28	[...] -> Parallel Seq Scan on game_objectives gm_obj (cost=0.00..15856.15 rows=378515 width=28) (actual time=0.039..4339.424 rows=293349 loops=4)
29	Planning Time: 0.809 ms
30	Execution Time: 8670.962 ms

ORM generated:

```
SELECT players.id                                   AS players_id,
       CASE
```

```
        WHEN ( players.nick IS NULL ) THEN 'unknown'
        ELSE players.nick
      END                                          AS player_nick,
      heroes.localized_name                        AS hero_localized_name,
      matches.id                                   AS match_id,
      COALESCE(game_objectives.subtype, 'NO_ACTION') AS hero_action,
      Count(*)                                     AS count
FROM  players
      JOIN matches_players_details
        ON matches_players_details.player_id = players.id
      JOIN heroes
        ON matches_players_details.hero_id = heroes.id
      JOIN matches
        ON matches_players_details.match_id = matches.id
      LEFT OUTER JOIN game_objectives
                   ON matches_players_details.id =
                      game_objectives.match_player_detail_id_1
WHERE players.id = 14944
GROUP BY players.id,
         heroes.localized_name,
         matches.id,
         game_objectives.subtype
ORDER BY matches.id ASC
```

1	GroupAggregate (cost=34801.70..34802.28 rows=23 width=114) (actual time=6131.572..6132.098 rows=16 loops=1)
2	[...] Group Key: matches.id, players.id, heroes.localized_name, game_objectives.subtype
3	[...] -> Sort (cost=34801.70..34801.76 rows=23 width=53) (actual time=6131.506..6131.681 rows=18 loops=1)
4	[...] Sort Key: matches.id, heroes.localized_name, game_objectives.subtype
5	[...] Sort Method: quicksort Memory: 26kB
6	[...] -> Nested Loop (cost=21591.84..34801.18 rows=23 width=53) (actual time=6114.820..6131.269 rows=18 loops=1)
7	[...] -> Index Scan using players_pk on players (cost=0.42..8.44 rows=1 width=15) (actual time=0.034..0.056 rows=1 loops=1)
8	[...] Index Cond: (id = 14944)
9	[...] -> Gather (cost=21591.42..34792.51 rows=23 width=42) (actual time=6114.742..6132.696 rows=18 loops=1)
10	[...] Workers Planned: 3
11	[...] Workers Launched: 3
12	[...] -> Parallel Hash Left Join (cost=20591.42..33790.21 rows=7 width=42) (actual time=6115.111..6124.173 rows=4 loops=4)
13	[...] Hash Cond: (matches_players_details.id = game_objectives.match_player_detail_id_1)
14	[...] -> Nested Loop (cost=3.83..13202.60 rows=3 width=22) (actual time=9.026..17.652 rows=3 loops=4)
15	[...] -> Hash Join (cost=3.54..13177.68 rows=3 width=22) (actual time=8.913..17.386 rows=3 loops=4)
16	[...] Hash Cond: (matches_players_details.hero_id = heroes.id)
17	[...] -> Parallel Seq Scan on matches_players_details (cost=0.00..13174.13 rows=3 width=16) (actual time=6.490..14.889 rows=3 loops=4)
18	[...] Filter: (player_id = 14944)
19	[...] Rows Removed by Filter: 124997
20	[...] -> Hash (cost=2.13..2.13 rows=113 width=14) (actual time=2.323..2.332 rows=113 loops=4)
21	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
22	[...] -> Seq Scan on heroes (cost=0.00..2.13 rows=113 width=14) (actual time=0.056..1.154 rows=113 loops=4)
23	[...] -> Index Only Scan using matches_pk on matches (cost=0.29..8.31 rows=1 width=4) (actual time=0.040..0.043 rows=1 loops=13)
24	[...] Index Cond: (id = matches_players_details.match_id)
25	[...] Heap Fetches: 13
26	[...] -> Parallel Hash (cost=15856.15..15856.15 rows=378515 width=28) (actual time=6102.164..6102.173 rows=293349 loops=4)
27	[...] Buckets: 2097152 Batches: 1 Memory Usage: 60704kB
28	[...] -> Parallel Seq Scan on game_objectives (cost=0.00..15856.15 rows=378515 width=28) (actual time=0.041..3041.573 rows=293349 loops=4)
29	Planning Time: 1.538 ms
30	Execution Time: 6134.625 ms

Summary

ORM generated absolutely the same query as original one, so it is impossible to get different performance from it.

Generally, the most expensive operations are JOIN. The execution time of initial query is 8670 ms and the execution time of the ORM generated one – 6134 ms. Such difference was caused by cache usage (the database cached some data after the initial query execution, so the ORM generated one was executed using this cached data) Total cost is 34802 of every query.

4. /players/14944/abilities

Initial:

```
WITH table1
     AS (SELECT pl.id,
                CASE
                  WHEN pl.nick IS NULL THEN 'unknown'
                  ELSE pl.nick
                END                              AS player_nick,
                hr.localized_name                AS hero_localized_name,
                mt.id                            AS match_id,
                ab.NAME                          AS ability_name,
                Max(ab_upg.level)
                  OVER (
                    partition BY mt.id, ab.NAME) AS upgrade_level
         FROM   players AS pl
                JOIN matches_players_details AS mt_pl_dt
                  ON mt_pl_dt.player_id = pl.id
                JOIN heroes AS hr
                  ON mt_pl_dt.hero_id = hr.id
                JOIN matches AS mt
                  ON mt_pl_dt.match_id = mt.id
                JOIN ability_upgrades AS ab_upg
                  ON mt_pl_dt.id = ab_upg.match_player_detail_id
                JOIN abilities AS ab
                  ON ab_upg.ability_id = ab.id
         WHERE  pl.id = 14944
         GROUP  BY pl.id,
                   hr.localized_name,
                   mt.id,
                   ab.NAME,
                   ab_upg.level
         ORDER  BY match_id ASC)
SELECT id,
       player_nick,
       hero_localized_name,
       match_id,
       ability_name,
       upgrade_level,
       Count(*)
FROM   table1
GROUP  BY id,
          player_nick,
          hero_localized_name,
          match_id,
          ability_name,
          upgrade_level
ORDER  BY match_id ASC
```

1	GroupAggregate (cost=99777.76..99783.13 rows=179 width=84) (actual time=37759.004..37766.384 rows=63 loops=1)
2	[...] Group Key: table1.match_id, table1.id, table1.player_nick, table1.hero_localized_name, table1.ability_name, table1.upgrade_level
3	[...] -> Sort (cost=99777.76..99778.21 rows=179 width=76) (actual time=37758.928..37762.054 rows=239 loops=1)
4	[...] Sort Key: table1.match_id, table1.id, table1.player_nick, table1.hero_localized_name, table1.ability_name, table1.upgrade_level
5	[...] Sort Method: quicksort Memory: 49kB
6	[...] -> Subquery Scan on table1 (cost=99765.69..99771.06 rows=179 width=76) (actual time=37745.065..37756.554 rows=239 loops=1)
7	[...] -> WindowAgg (cost=99765.69..99769.27 rows=179 width=80) (actual time=37745.046..37752.100 rows=239 loops=1)
8	[...] -> Sort (cost=99765.69..99766.14 rows=179 width=55) (actual time=37744.961..37747.170 rows=239 loops=1)
9	[...] Sort Key: mt.id, ab.name
10	[...] Sort Method: quicksort Memory: 49kB
11	[...] -> Group (cost=99756.31..99758.99 rows=179 width=55) (actual time=37735.358..37742.361 rows=239 loops=1)
12	[...] Group Key: mt.id, pl.id, hr.localized_name, ab.name, ab_upg.level
13	[...] -> Sort (cost=99756.31..99756.76 rows=179 width=55) (actual time=37735.324..37737.607 rows=239 loops=1)
14	[...] Sort Key: mt.id, hr.localized_name, ab.name, ab_upg.level
15	[...] Sort Method: quicksort Memory: 49kB
16	[...] -> Nested Loop (cost=14178.69..99749.61 rows=179 width=55) (actual time=13095.390..37732.522 rows=239 loops=1)
17	[...] -> Index Scan using players_pk on players pl (cost=0.42..8.44 rows=1 width=15) (actual time=0.021..0.052 rows=1 loops=1)
18	[...] Index Cond: (id = 14944)
19	[...] -> Gather (cost=14178.27..99739.38 rows=179 width=44) (actual time=13095.338..37727.796 rows=239 loops=1)
20	[...] Workers Planned: 4
21	[...] Workers Launched: 4
22	[...] -> Nested Loop (cost=13178.27..98721.48 rows=45 width=44) (actual time=19046.027..37724.405 rows=48 loops=5)
23	[...] -> Nested Loop (cost=13178.00..98708.32 rows=45 width=26) (actual time=19045.949..37722.057 rows=48 loops=5)
24	[...] -> Hash Join (cost=13177.71..98334.48 rows=45 width=26) (actual time=19045.809..37719.555 rows=48 loops=5)
25	[...] Hash Cond: (mt_pl_dt.hero_id = hr.id)
26	[...] -> Parallel Hash Join (cost=13174.17..98330.82 rows=45 width=20) (actual time=19042.653..37715.366 rows=48 loops=5)
27	[...] Hash Cond: (ab_upg.match_player_detail_id = mt_pl_dt.id)
28	[...] -> Parallel Seq Scan on ability_upgrades ab_upg (cost=0.00..79290.00 rows=2234900 width=12) (actual time=0.025..18686.349 rows=1787920 loops=5)
29	[...] -> Parallel Hash (cost=13174.13..13174.13 rows=3 width=16) (actual time=14.153..14.163 rows=3 loops=5)
30	[...] Buckets: 1024 Batches: 1 Memory Usage: 136kB
31	[...] -> Parallel Seq Scan on matches_players_details mt_pl_dt (cost=0.00..13174.13 rows=3 width=16) (actual time=5.535..13.966 rows=3 loops=5)
32	[...] Filter: (player_id = 14944)
33	[...] Rows Removed by Filter: 99997
34	[...] -> Hash (cost=2.13..2.13 rows=113 width=14) (actual time=3.002..3.012 rows=113 loops=5)
35	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
36	[...] -> Seq Scan on heroes hr (cost=0.00..2.13 rows=113 width=14) (actual time=0.043..1.508 rows=113 loops=5)
37	[...] -> Index Only Scan using matches_pk on matches mt (cost=0.29..8.31 rows=1 width=4) (actual time=0.018..0.018 rows=1 loops=239)
38	[...] Index Cond: (id = mt_pl_dt.match_id)
39	[...] Heap Fetches: 239
40	[...] -> Index Scan using abilities_pk on abilities ab (cost=0.28..0.29 rows=1 width=26) (actual time=0.016..0.016 rows=1 loops=239)
41	[...] Index Cond: (id = ab_upg.ability_id)
42	Planning Time: 1.398 ms
43	Execution Time: 37768.122 ms

ORM generated:

```
SELECT anon_1.id                   AS anon_1_id,
       anon_1.player_nick          AS anon_1_player_nick,
       anon_1.hero_localized_name AS anon_1_hero_localized_name,
       anon_1.match_id             AS anon_1_match_id,
       anon_1.ability_name         AS anon_1_ability_name,
       anon_1.upgrade_level        AS anon_1_upgrade_level,
       Count(*)                    AS count
FROM   (SELECT players.id                                    AS id,
               CASE
                 WHEN ( players.nick IS NULL ) THEN 'unknown'
                 ELSE players.nick
               END                                           AS player_nick,
               heroes.localized_name                         AS
               hero_localized_name,
               matches.id                                    AS match_id,
               abilities.NAME                                AS ability_name,
               Max(ability_upgrades.level)
                 OVER (
                   partition BY matches.id, abilities.NAME) AS upgrade_level
        FROM   players
               JOIN matches_players_details
                 ON matches_players_details.player_id = players.id
               JOIN heroes
                 ON matches_players_details.hero_id = heroes.id
               JOIN matches
                 ON matches_players_details.match_id = matches.id
               JOIN ability_upgrades
                 ON matches_players_details.id =
                    ability_upgrades.match_player_detail_id
               JOIN abilities
                 ON ability_upgrades.ability_id = abilities.id
        WHERE  players.id = 14944
        GROUP  BY players.id,
                  heroes.localized_name,
                  matches.id,
                  abilities.NAME,
                  ability_upgrades.level
        ORDER  BY matches.id ASC) AS anon_1
GROUP  BY anon_1.id,
          anon_1.player_nick,
          anon_1.hero_localized_name,
          anon_1.match_id,
          anon_1.ability_name,
          anon_1.upgrade_level
ORDER  BY anon_1.match_id ASC
```

1	GroupAggregate (cost=99777.76..99783.13 rows=179 width=84) (actual time=37478.410..37483.983 rows=63 loops=1)
2	[...] Group Key: anon_1.match_id, anon_1.id, anon_1.player_nick, anon_1.hero_localized_name, anon_1.ability_name, anon_1.upgrade_level
3	[...] -> Sort (cost=99777.76..99778.21 rows=179 width=76) (actual time=37478.324..37480.762 rows=239 loops=1)
4	[...] Sort Key: anon_1.match_id, anon_1.id, anon_1.player_nick, anon_1.hero_localized_name, anon_1.ability_name, anon_1.upgrade_level
5	[...] Sort Method: quicksort Memory: 49kB
6	[...] -> Subquery Scan on anon_1 (cost=99765.69..99771.06 rows=179 width=76) (actual time=37464.098..37475.900 rows=239 loops=1)
7	[...] -> WindowAgg (cost=99765.69..99769.27 rows=179 width=80) (actual time=37464.078..37471.245 rows=239 loops=1)
8	[...] -> Sort (cost=99765.69..99766.14 rows=179 width=55) (actual time=37464.001..37466.221 rows=239 loops=1)
9	[...] Sort Key: matches.id, abilities.name
10	[...] Sort Method: quicksort Memory: 49kB
11	[...] -> Group (cost=99756.31..99758.99 rows=179 width=55) (actual time=37454.758..37461.460 rows=239 loops=1)
12	[...] Group Key: matches.id, players.id, heroes.localized_name, abilities.name, ability_upgrades.level
13	[...] -> Sort (cost=99756.31..99756.76 rows=179 width=55) (actual time=37454.730..37456.914 rows=239 loops=1)
14	[...] Sort Key: matches.id, heroes.localized_name, abilities.name, ability_upgrades.level
15	[...] Sort Method: quicksort Memory: 49kB
16	[...] -> Nested Loop (cost=14178.69..99749.61 rows=179 width=55) (actual time=13650.112..37452.019 rows=239 loops=1)
17	[...] -> Index Scan using players_pk on players (cost=0.42..8.44 rows=1 width=15) (actual time=0.022..0.053 rows=1 loops=1)
18	[...] Index Cond: (id = 14944)
19	[...] -> Gather (cost=14178.27..99739.38 rows=179 width=44) (actual time=13650.058..37447.340 rows=239 loops=1)
20	[...] Workers Planned: 4
21	[...] Workers Launched: 4
22	[...] -> Nested Loop (cost=13178.27..98721.48 rows=45 width=44) (actual time=15970.125..37444.745 rows=48 loops=5)
23	[...] -> Nested Loop (cost=13178.00..98708.32 rows=45 width=26) (actual time=15970.044..37442.420 rows=48 loops=5)
24	[...] -> Hash Join (cost=13177.71..98334.48 rows=45 width=26) (actual time=15969.923..37439.917 rows=48 loops=5)
25	[...] Hash Cond: (matches_players_details.hero_id = heroes.id)
26	[...] -> Parallel Hash Join (cost=13174.17..98330.82 rows=45 width=20) (actual time=15967.249..37436.213 rows=48 loops=5)
27	[...] Hash Cond: (ability_upgrades.match_player_detail_id = matches_players_details.id)
28	[...] -> Parallel Seq Scan on ability_upgrades (cost=0.00..79290.00 rows=2234900 width=12) (actual time=0.027..18544.646 rows=1787920 loops=5)
29	[...] -> Parallel Hash (cost=13174.13..13174.13 rows=3 width=16) (actual time=12.835..12.844 rows=3 loops=5)
30	[...] Buckets: 1024 Batches: 1 Memory Usage: 168kB
31	[...] -> Parallel Seq Scan on matches_players_details (cost=0.00..13174.13 rows=3 width=16) (actual time=5.649..12.669 rows=3 loops=5)
32	[...] Filter: (player_id = 14944)
33	[...] Rows Removed by Filter: 99997
34	[...] -> Hash (cost=2.13..2.13 rows=113 width=14) (actual time=2.529..2.538 rows=113 loops=5)
35	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
36	[...] -> Seq Scan on heroes (cost=0.00..2.13 rows=113 width=14) (actual time=0.041..1.285 rows=113 loops=5)
37	[...] -> Index Only Scan using matches_pk on matches (cost=0.29..8.31 rows=1 width=4) (actual time=0.018..0.018 rows=1 loops=239)
38	[...] Index Cond: (id = matches_players_details.match_id)
39	[...] Heap Fetches: 239
40	[...] -> Index Scan using abilities_pk on abilities (cost=0.28..0.29 rows=1 width=26) (actual time=0.015..0.015 rows=1 loops=239)
41	[...] Index Cond: (id = ability_upgrades.ability_id)
42	Planning Time: 1.397 ms
43	Execution Time: 37485.279 ms

Summary

ORM generated query is similar to the original one, so it is impossible to get different performance from it. The only difference is that in original query was used WITH to make subtable, but in the generated query was used nested SELECT. However, query plan has the same structure and almost the same values in both cases.

Generally, the most expensive operations are JOIN. The execution time of initial query is 37768 ms and the execution time of the ORM generated one – 37485 ms. The difference in execution time can be considered as statistical deviation. Total cost is 99783 of every query.

5. /matches/21421/top_purchases/

Initial:

```
SELECT *
FROM   (
               SELECT   *,
                        Row_number() OVER (partition BY localized_name ORDER BY hero_id ASC, count
DESC, item_name ASC) AS row_num
               FROM     (
                                SELECT   m.id AS match_id,
                                         m_p_d.hero_id,
                                         her.localized_name,
                                         p_log.item_id,
                                         itms.NAME AS item_name,
                                         Count(*)
                                FROM     purchase_logs           AS p_log
                                JOIN     matches_players_details AS m_p_d
                                ON       p_log.match_player_detail_id = m_p_d.id
                                JOIN     heroes AS her
                                ON       m_p_d.hero_id = her.id
                                JOIN     matches AS m
                                ON       m_p_d.match_id = m.id
                                JOIN     items AS itms
                                ON       p_log.item_id = itms.id
                                WHERE    ((
                                                          m_p_d.player_slot >= 0
                                                  AND     m_p_d.player_slot <= 4
                                                  AND     m.radiant_win IS true)
                                         OR       (
                                                          m_p_d.player_slot >= 128
                                                  AND     m_p_d.player_slot <= 132
                                                  AND     m.radiant_win IS false))
                                AND      m.id = 21421
                                GROUP BY m.id,
                                         m_p_d.hero_id,
                                         her.localized_name,
                                         p_log.item_id,
                                         itms.NAME) table1
               ORDER BY hero_id ASC,
                        count DESC,
                        item_name ASC) table2
WHERE  row_num <= 5
```

```
1  Subquery Scan on table2  (cost=156879.44..156882.41 rows=66 width=52) (actual time=80571.414..80575.520 rows=25 loops=1)
2  [...] Filter: (table2.row_num <= 5)
3  [...] Rows Removed by Filter: 88
4  [...] -> Sort  (cost=156879.44..156879.93 rows=198 width=52) (actual time=80571.387..80573.231 rows=113 loops=1)
5  [...] Sort Key: table1.hero_id, table1.count DESC, table1.item_name
6  [...] Sort Method: quicksort  Memory: 37kB
7  [...] -> WindowAgg  (cost=156866.94..156871.89 rows=198 width=52) (actual time=80563.946..80569.384 rows=113 loops=1)
8  [...] -> Sort  (cost=156866.94..156867.43 rows=198 width=44) (actual time=80563.872..80565.721 rows=113 loops=1)
9  [...] Sort Key: table1.localized_name, table1.hero_id, table1.count DESC, table1.item_name
10 [...] Sort Method: quicksort  Memory: 34kB
11 [...] -> Subquery Scan on table1  (cost=156851.96..156859.38 rows=198 width=44) (actual time=80551.134..80561.730 rows=113 loops=1)
12 [...] -> GroupAggregate  (cost=156851.96..156857.40 rows=198 width=44) (actual time=80551.115..80558.470 rows=113 loops=1)
13 [...] Group Key: m.id, m_p_d.hero_id, her.localized_name, p_log.item_id, itms.name
14 [...] -> Sort  (cost=156851.96..156852.45 rows=198 width=36) (actual time=80551.049..80553.674 rows=188 loops=1)
15 [...] Sort Key: m_p_d.hero_id, her.localized_name, p_log.item_id, itms.name
16 [...] Sort Method: quicksort  Memory: 39kB
17 [...] -> Nested Loop  (cost=1036.40..156844.40 rows=198 width=36) (actual time=35543.823..80548.969 rows=188 loops=1)
18 [...] Join Filter: (((m_p_d.player_slot >= 0) AND (m_p_d.player_slot <= 4) AND (m.radiant_win IS TRUE)) OR ((m_p_d.player_slot >= 128) AND (m_p_d.player_slot <= 132) AND (m.radiant_win IS FALSE)))
19 [...] Rows Removed by Join Filter: 165
20 [...] -> Gather  (cost=1036.11..156831.06 rows=255 width=40) (actual time=35543.598..80533.533 rows=353 loops=1)
21 [...] Workers Planned: 4
22 [...] Workers Launched: 4
23 [...] -> Hash Join  (cost=36.11..155805.56 rows=64 width=40) (actual time=53539.918..80538.094 rows=71 loops=5)
24 [...] Hash Cond: (p_log.item_id = itms.id)
25 [...] -> Hash Join  (cost=28.08..155797.36 rows=64 width=26) (actual time=53534.386..80531.145 rows=71 loops=5)
26 [...] Hash Cond: (m_p_d.hero_id = her.id)
27 [...] -> Hash Join  (cost=24.54..155793.64 rows=64 width=16) (actual time=53532.013..80527.382 rows=71 loops=5)
28 [...] Hash Cond: (p_log.match_player_detail_id = m_p_d.id)
29 [...] -> Parallel Seq Scan on purchase_logs p_log  (cost=0.00..143829.36 rows=4548436 width=8) (actual time=0.019..40225.434 rows=3638749 loops=5)
30 [...] -> Hash  (cost=24.45..24.45 rows=7 width=16) (actual time=0.300..0.312 rows=10 loops=5)
31 [...] Buckets: 1024  Batches: 1  Memory Usage: 9kB
32 [...] -> Index Scan using idx_match_id_player_id on matches_players_details m_p_d  (cost=0.42..24.45 rows=7 width=16) (actual time=0.035..0.185 rows=10 loops=5)
33 [...] Index Cond: (match_id = 21421)
34 [...] Filter: (((player_slot >= 0) AND (player_slot <= 4)) OR ((player_slot >= 128) AND (player_slot <= 132)))
35 [...] -> Hash  (cost=2.13..2.13 rows=113 width=14) (actual time=2.308..2.320 rows=113 loops=5)
36 [...] Buckets: 1024  Batches: 1  Memory Usage: 14kB
37 [...] -> Seq Scan on heroes her  (cost=0.00..2.13 rows=113 width=14) (actual time=0.029..1.168 rows=113 loops=5)
38 [...] -> Hash  (cost=4.68..4.68 rows=268 width=18) (actual time=5.407..5.420 rows=268 loops=5)
39 [...] Buckets: 1024  Batches: 1  Memory Usage: 22kB
40 [...] -> Seq Scan on items itms  (cost=0.00..4.68 rows=268 width=18) (actual time=0.040..2.714 rows=268 loops=5)
41 [...] -> Materialize  (cost=0.29..8.31 rows=1 width=5) (actual time=0.010..0.014 rows=1 loops=353)
42 [...] -> Index Scan using matches_pk on matches m  (cost=0.29..8.31 rows=1 width=5) (actual time=0.041..0.063 rows=1 loops=1)
43 [...] Index Cond: (id = 21421)
44 [...] Filter: ((radiant_win IS TRUE) OR (radiant_win IS FALSE))
45 Planning Time: 1.078 ms
46 Execution Time: 80576.700 ms
```

ORM generated:

```
SELECT anon_1.match_id        AS anon_1_match_id,
       anon_1.hero_id         AS anon_1_hero_id,
       anon_1.localized_name AS anon_1_localized_name,
       anon_1.item_id         AS anon_1_item_id,
       anon_1.item_name       AS anon_1_item_name,
       anon_1.count           AS anon_1_count,
       anon_1.row_num         AS anon_1_row_num
FROM    (SELECT anon_2.match_id
                AS
                match_id,
             anon_2.hero_id
                AS hero_id,
             anon_2.localized_name
                AS localized_name,
             anon_2.item_id
                AS item_id,
             anon_2.item_name
                AS item_name,
             anon_2.count
                AS count,
             Row_number()
               OVER (
```

```
                partition BY anon_2.localized_name
                ORDER BY anon_2.hero_id ASC, anon_2.count DESC,
              anon_2.item_name ASC
              ) AS
            row_num
      FROM   (SELECT matches.id                       AS match_id,
                     matches_players_details.hero_id AS hero_id,
                     heroes.localized_name            AS localized_name,
                     purchase_logs.item_id            AS item_id,
                     items.NAME                       AS item_name,
                     Count(*)                         AS count
              FROM   purchase_logs
                     JOIN matches_players_details
                       ON purchase_logs.match_player_detail_id =
                          matches_players_details.id
                     JOIN heroes
                       ON matches_players_details.hero_id = heroes.id
                     JOIN matches
                       ON matches_players_details.match_id = matches.id
                     JOIN items
                       ON purchase_logs.item_id = items.id
              WHERE  ( matches_players_details.player_slot >= 0
                       AND matches_players_details.player_slot <= 4
                       AND matches.radiant_win = true
                        OR matches_players_details.player_slot >= 128
                           AND matches_players_details.player_slot <= 132
                           AND matches.radiant_win = false )
                     AND matches.id = 21421
              GROUP  BY matches.id,
                        matches_players_details.hero_id,
                        heroes.localized_name,
                        purchase_logs.item_id,
                        items.NAME) AS anon_2
      ORDER  BY anon_2.hero_id ASC,
                anon_2.count DESC,
                anon_2.item_name ASC) AS anon_1
WHERE  anon_1.row_num <= 5
```

```
1  Subquery Scan on anon_1  (cost=156879.44..156882.41 rows=66 width=52) (actual time=79008.151..79010.480 rows=25 loops=1)
2  [...] Filter: (anon_1.row_num <= 5)
3  [...] Rows Removed by Filter: 88
4  [...] ->  Sort  (cost=156879.44..156879.93 rows=198 width=52) (actual time=79008.123..79009.191 rows=113 loops=1)
5  [...] Sort Key: anon_2.hero_id, anon_2.count DESC, anon_2.item_name
6  [...] Sort Method: quicksort  Memory: 37kB
7  [...] ->  WindowAgg  (cost=156866.94..156871.89 rows=198 width=52) (actual time=79003.412..79006.856 rows=113 loops=1)
8  [...] ->  Sort  (cost=156866.94..156867.43 rows=198 width=44) (actual time=79003.348..79004.434 rows=113 loops=1)
9  [...] Sort Key: anon_2.localized_name, anon_2.hero_id, anon_2.count DESC, anon_2.item_name
10 [...] Sort Method: quicksort  Memory: 34kB
11 [...] ->  Subquery Scan on anon_2  (cost=156851.96..156859.38 rows=198 width=44) (actual time=78995.140..79002.082 rows=113 loops=1)
12 [...] ->  GroupAggregate  (cost=156851.96..156857.40 rows=198 width=44) (actual time=78995.115..78999.920 rows=113 loops=1)
13 [...] Group Key: matches.id, matches_players_details.hero_id, heroes.localized_name, purchase_logs.item_id, items.name
14 [...] ->  Sort  (cost=156851.96..156852.45 rows=198 width=36) (actual time=78995.062..78996.850 rows=188 loops=1)
15 [...] Sort Key: matches_players_details.hero_id, heroes.localized_name, purchase_logs.item_id, items.name
16 [...] Sort Method: quicksort  Memory: 39kB
17 [...] ->  Nested Loop  (cost=1036.40..156844.40 rows=198 width=36) (actual time=78974.088..78993.032 rows=188 loops=1)
18 [...] Join Filter: (((matches_players_details.player_slot >= 0) AND (matches_players_details.player_slot <= 4) AND matches.radiant_win) OR ((matches_players_details.player_slot >= 128) AND (matches_players_details.player_slot <= 132) AND (NOT matches.radiant_win)))
19 [...] Rows Removed by Join Filter: 165
20 [...] ->  Gather  (cost=1036.11..156831.06 rows=255 width=40) (actual time=78973.967..78977.564 rows=353 loops=1)
21 [...] Workers Planned: 4
22 [...] Workers Launched: 4
23 [...] ->  Hash Join  (cost=36.11..155805.56 rows=64 width=40) (actual time=51782.093..78969.177 rows=71 loops=5)
24 [...] Hash Cond: (purchase_logs.item_id = items.id)
25 [...] ->  Hash Join  (cost=28.08..155797.36 rows=64 width=26) (actual time=51775.774..78961.422 rows=71 loops=5)
26 [...] Hash Cond: (matches_players_details.hero_id = heroes.id)
27 [...] ->  Hash Join  (cost=24.54..155793.64 rows=64 width=16) (actual time=51773.056..78957.285 rows=71 loops=5)
28 [...] Hash Cond: (purchase_logs.match_player_detail_id = matches_players_details.id)
29 [...] ->  Parallel Seq Scan on purchase_logs  (cost=0.00..143829.36 rows=4548436 width=8) (actual time=0.021..39400.331 rows=3638749 loops=5)
30 [...] ->  Hash  (cost=24.45..24.45 rows=7 width=16) (actual time=0.313..0.328 rows=10 loops=5)
31 [...] Buckets: 1024  Batches: 1  Memory Usage: 9kB
32 [...] ->  Index Scan using idx_match_id_player_id on matches_players_details  (cost=0.42..24.45 rows=7 width=16) (actual time=0.037..0.171 rows=10 loops=5)
33 [...] Index Cond: (match_id = 21421)
34 [...] Filter: (((player_slot >= 0) AND (player_slot <= 4)) OR ((player_slot >= 128) AND (player_slot <= 132)))
35 [...] ->  Hash  (cost=2.13..2.13 rows=113 width=14) (actual time=2.642..2.652 rows=113 loops=5)
36 [...] Buckets: 1024  Batches: 1  Memory Usage: 14kB
37 [...] ->  Seq Scan on heroes  (cost=0.00..2.13 rows=113 width=14) (actual time=0.031..1.319 rows=113 loops=5)
38 [...] ->  Hash  (cost=4.68..4.68 rows=268 width=18) (actual time=6.186..6.195 rows=268 loops=5)
39 [...] Buckets: 1024  Batches: 1  Memory Usage: 22kB
40 [...] ->  Seq Scan on items  (cost=0.00..4.68 rows=268 width=18) (actual time=0.036..3.111 rows=268 loops=5)
41 [...] ->  Materialize  (cost=0.29..8.31 rows=1 width=5) (actual time=0.010..0.014 rows=1 loops=353)
42 [...] ->  Index Scan using matches_pk on matches  (cost=0.29..8.31 rows=1 width=5) (actual time=0.045..0.071 rows=1 loops=1)
43 [...] Index Cond: (id = 21421)
44 [...] Filter: (radiant_win OR (NOT radiant_win))
45 Planning Time: 1.887 ms
46 Execution Time: 79011.339 ms
```

Summary

ORM generated query is identical to the original one, so it is impossible to get different performance from it.

Generally, the most expensive operations are JOIN. The execution time of initial query is 80576 ms and the execution time of the ORM generated one – 79011 ms. The difference in execution time can be considered as statistical deviation. Total cost is 156882 of every query.

6. /abilities/5004/usage/

Initial:

```
SELECT DISTINCT
ON (
                                hero_id, winner) ability_id,
                NAME,
                hero_id,
                hero_name,
                winner,
                                Concat(Substr(percent_time, 1, Length(percent_time) - 1), '0-
', Substr(percent_time, 1, Length(percent_time) - 1), '9') AS rozsah,
                Count(*)
FROM            (
                       SELECT ab_up.ability_id,
                              ab.NAME,
                              her.id             AS hero_id,
                              her.localized_name AS hero_name,
                              CASE
                                     WHEN m_p_d.player_slot >= 0
                                     AND    m_p_d.player_slot <= 4
                                     AND    m.radiant_win IS true THEN true
                                     WHEN m_p_d.player_slot >= 128
                                     AND    m_p_d.player_slot <= 132
                                     AND    m.radiant_win IS false THEN true
                                     ELSE false
                              END                                          AS winner,
                              cast((ab_up.time * 100 / m.duration) AS varchar) AS percent_time
                       FROM   ability_upgrades                             AS ab_up
                       JOIN   abilities                                    AS ab
                       ON     ab.id = ab_up.ability_id
                       JOIN   matches_players_details AS m_p_d
                       ON     ab_up.match_player_detail_id = m_p_d.id
                       JOIN   heroes AS her
                       ON     her.id = m_p_d.hero_id
                       JOIN   matches AS m
                       ON     m.id = m_p_d.match_id
                       WHERE  ab_up.ability_id = 5004) AS table1
GROUP BY        ability_id,
                NAME,
                hero_id,
                hero_name,
                winner,
                rozsah
ORDER BY        hero_id ASC,
                winner DESC,
                count DESC
```

```
1  Unique  (cost=115593.91..115879.98 rows=38142 width=81) (actual time=6171.420..6172.101 rows=3 loops=1)
2  [...] -> Sort  (cost=115593.91..115689.27 rows=38142 width=81) (actual time=6171.399..6171.725 rows=34 loops=1)
3  [...] Sort Key: her.id, (CASE WHEN ((m_p_d.player_slot >= 0) AND (m_p_d.player_slot <= 4) AND (m.radiant_win IS TRUE)) THEN true WHEN ((m_p_d.player_slot >= 128) AND (m_
4  [...] Sort Method: quicksort  Memory: 29kB
5  [...] -> HashAggregate  (cost=109735.48..112691.48 rows=38142 width=81) (actual time=6170.456..6171.022 rows=34 loops=1)
6  [...] Group Key: ab_up.ability_id, ab.name, her.id, CASE WHEN ((m_p_d.player_slot >= 0) AND (m_p_d.player_slot <= 4) AND (m.radiant_win IS TRUE)) THEN true WHEN ((m_p_d.p
7  [...] -> Nested Loop  (cost=17431.85..109163.35 rows=38142 width=73) (actual time=4095.025..5680.302 rows=37068 loops=1)
8  [...] -> Index Scan using abilities_pk on abilities ab  (cost=0.28..8.29 rows=1 width=26) (actual time=0.020..0.047 rows=1 loops=1)
9  [...] Index Cond: (id = 5004)
10 [...] -> Gather  (cost=17431.57..106199.05 rows=38142 width=31) (actual time=4094.936..4701.195 rows=37068 loops=1)
11 [...] Workers Planned: 4
12 [...] Workers Launched: 4
13 [...] -> Hash Join  (cost=16431.57..101384.85 rows=9536 width=31) (actual time=4098.314..4906.661 rows=7414 loops=5)
14 [...] Hash Cond: (m_p_d.match_id = m.id)
15 [...] -> Hash Join  (cost=14790.57..99718.82 rows=9536 width=30) (actual time=2689.824..3333.870 rows=7414 loops=5)
16 [...] Hash Cond: (m_p_d.hero_id = her.id)
17 [...] -> Parallel Hash Join  (cost=14787.03..99689.31 rows=9536 width=20) (actual time=2686.901..3162.270 rows=7414 loops=5)
18 [...] Hash Cond: (ab_up.match_player_detail_id = m_p_d.id)
19 [...] -> Parallel Seq Scan on ability_upgrades ab_up  (cost=0.00..84877.25 rows=9536 width=12) (actual time=0.225..273.528 rows=7414 loops=5)
20 [...] Filter: (ability_id = 5004)
21 [...] Rows Removed by Filter: 1780506
22 [...] -> Parallel Hash  (cost=12770.90..12770.90 rows=161290 width=16) (actual time=2676.390..2676.399 rows=100000 loops=5)
23 [...] Buckets: 524288  Batches: 1  Memory Usage: 27648kB
24 [...] -> Parallel Seq Scan on matches_players_details m_p_d  (cost=0.00..12770.90 rows=161290 width=16) (actual time=0.038..1259.065 rows=100000 loops=5)
25 [...] -> Hash  (cost=2.13..2.13 rows=113 width=14) (actual time=2.830..2.840 rows=113 loops=5)
26 [...] Buckets: 1024  Batches: 1  Memory Usage: 14kB
27 [...] -> Seq Scan on heroes her  (cost=0.00..2.13 rows=113 width=14) (actual time=0.055..1.413 rows=113 loops=5)
28 [...] -> Hash  (cost=1016.00..1016.00 rows=50000 width=9) (actual time=1407.865..1407.874 rows=50000 loops=5)
29 [...] Buckets: 65536  Batches: 1  Memory Usage: 2661kB
30 [...] -> Seq Scan on matches m  (cost=0.00..1016.00 rows=50000 width=9) (actual time=0.039..705.985 rows=50000 loops=5)
31 Planning Time: 0.955 ms
32 Execution Time: 6173.752 ms
```

ORM generated:

```
SELECT DISTINCT ON ( anon_1.hero_id, anon_1.winner)
anon_1.ability_id                                                    AS
anon_1_ability_id,
anon_1.NAME
AS anon_1_name,
anon_1.hero_id
AS anon_1_hero_id,
anon_1.hero_name
AS anon_1_hero_name,
anon_1.winner
AS anon_1_winner,
Concat(
Substr(anon_1.percent_time, 1,
Length(anon_1.percent_time) - 1), '0-',
                    Substr(anon_1.percent_time, 1,
                    Length(anon_1.percent_time) - 1), '9') AS rozsah,
                    Count(*)
AS
count
FROM    (SELECT ability_upgrades.ability_id
                AS
                       ability_id,
                abilities.NAME
                AS
                       NAME,
                heroes.id
                AS
                       hero_id,
                heroes.localized_name
                AS
```

```
                hero_name,
            CASE
              WHEN ( matches_players_details.player_slot >= 0
                     AND matches_players_details.player_slot <= 4
                     AND matches.radiant_win = true ) THEN true
              WHEN ( matches_players_details.player_slot >= 128
                     AND matches_players_details.player_slot <= 132
                     AND matches.radiant_win = false ) THEN true
              ELSE false
            END
            AS
                winner,
            Cast(( ability_upgrades.time * 100 ) / matches.duration AS
                 VARCHAR) AS
            percent_time
      FROM  ability_upgrades
            JOIN abilities
              ON abilities.id = ability_upgrades.ability_id
            JOIN matches_players_details
              ON ability_upgrades.match_player_detail_id =
                 matches_players_details.id
            JOIN heroes
              ON heroes.id = matches_players_details.hero_id
            JOIN matches
              ON matches.id = matches_players_details.match_id
      WHERE ability_upgrades.ability_id = 5004) AS anon_1
GROUP BY anon_1.ability_id,
         anon_1.NAME,
         anon_1.hero_id,
         anon_1.hero_name,
         anon_1.winner,
         rozsah
ORDER BY anon_1.hero_id ASC,
         anon_1.winner DESC,
         Count(*) DESC
```

```
1  Unique  (cost=115593.91..115879.98 rows=38142 width=81) (actual time=5683.688..5684.369 rows=3 loops=1)
2  [...] -> Sort  (cost=115593.91..115689.27 rows=38142 width=81) (actual time=5683.663..5683.992 rows=34 loops=1)
3  [...] Sort Key: heroes.id, (CASE WHEN ((matches_players_details.player_slot >= 0) AND (matches_players_details.player_slot <= 4) AND matches.radiant_win) THE
4  [...] Sort Method: quicksort  Memory: 29kB
5  [...] -> HashAggregate  (cost=109735.48..112691.48 rows=38142 width=81) (actual time=5682.666..5683.264 rows=34 loops=1)
6  [...] Group Key: ability_upgrades.ability_id, abilities.name, heroes.id, CASE WHEN ((matches_players_details.player_slot >= 0) AND (matches_players_details.playe
7  [...] -> Nested Loop  (cost=17431.85..109163.35 rows=38142 width=73) (actual time=3773.402..5208.659 rows=37068 loops=1)
8  [...] -> Index Scan using abilities_pk on abilities  (cost=0.28..8.29 rows=1 width=26) (actual time=0.026..0.048 rows=1 loops=1)
9  [...] Index Cond: (id = 5004)
10 [...] -> Gather  (cost=17431.57..106199.05 rows=38142 width=31) (actual time=3773.311..4298.772 rows=37068 loops=1)
11 [...] Workers Planned: 4
12 [...] Workers Launched: 4
13 [...] -> Hash Join  (cost=16431.57..101384.85 rows=9536 width=31) (actual time=3789.464..4701.823 rows=7414 loops=5)
14 [...] Hash Cond: (matches_players_details.match_id = matches.id)
15 [...] -> Hash Join  (cost=14790.57..99718.82 rows=9536 width=30) (actual time=2559.020..3291.747 rows=7414 loops=5)
16 [...] Hash Cond: (matches_players_details.hero_id = heroes.id)
17 [...] -> Parallel Hash Join  (cost=14787.03..99689.31 rows=9536 width=20) (actual time=2556.555..3084.755 rows=7414 loops=5)
18 [...] Hash Cond: (ability_upgrades.match_player_detail_id = matches_players_details.id)
19 [...] -> Parallel Seq Scan on ability_upgrades  (cost=0.00..84877.25 rows=9536 width=12) (actual time=0.155..343.324 rows=7414 loops=5)
20 [...] Filter: (ability_id = 5004)
21 [...] Rows Removed by Filter: 1780506
22 [...] -> Parallel Hash  (cost=12770.90..12770.90 rows=161290 width=16) (actual time=2527.334..2527.344 rows=100000 loops=5)
23 [...] Buckets: 524288  Batches: 1  Memory Usage: 27616kB
24 [...] -> Parallel Seq Scan on matches_players_details  (cost=0.00..12770.90 rows=161290 width=16) (actual time=0.026..1232.559 rows=100000 loops=5)
25 [...] -> Hash  (cost=2.13..2.13 rows=113 width=14) (actual time=2.391..2.401 rows=113 loops=5)
26 [...] Buckets: 1024  Batches: 1  Memory Usage: 14kB
27 [...] -> Seq Scan on heroes  (cost=0.00..2.13 rows=113 width=14) (actual time=0.037..1.200 rows=113 loops=5)
28 [...] -> Hash  (cost=1016.00..1016.00 rows=50000 width=9) (actual time=1230.025..1230.037 rows=50000 loops=5)
29 [...] Buckets: 65536  Batches: 1  Memory Usage: 2661kB
30 [...] -> Seq Scan on matches  (cost=0.00..1016.00 rows=50000 width=9) (actual time=0.030..613.422 rows=50000 loops=5)
31 Planning Time: 1.478 ms
32 Execution Time: 5686.164 ms
```

Summary

ORM generated query is absolutely identical to the original one, so it is impossible to get different performance from it.

Generally, the most expensive operations are JOIN. The execution time of initial query is 6173 ms and the execution time of the ORM generated one – 5686 ms. The difference in execution time can be considered as statistical deviation. Total cost is 115879 of every query.

7. /statistics/tower_kills/

Initial:

```
SELECT hero_id,
       heroes.localized_name,
       max
FROM   (SELECT hero_id,
               Max(count)
        FROM   (SELECT match_id,
                       hero_id,
                       Count(*)
                FROM  (SELECT Row_number()
                              OVER (
                                ORDER BY m_p_d.match_id ASC, g_ob.time ASC)
                            - Row_number()
                                          OVER (
                                            partition BY m_p_d.match_id,
                                          m_p_d.hero_id
                                            ORDER BY m_p_d.match_id ASC,
                                          g_ob.time
                                          ASC) AS
                            row_num,
                            m_p_d.match_id,
                            m_p_d.hero_id,
                            g_ob.time
                       FROM   game_objectives AS g_ob
                              JOIN matches_players_details AS m_p_d
                                ON g_ob.match_player_detail_id_1 = m_p_d.id
                       WHERE  g_ob.subtype = 'CHAT_MESSAGE_TOWER_KILL'
                              AND match_player_detail_id_1 IS NOT NULL
                       ORDER  BY m_p_d.match_id ASC,
                                 g_ob.time ASC) table1
                GROUP  BY row_num,
                          match_id,
                          hero_id) table2
        GROUP  BY hero_id) table3
       JOIN heroes
         ON hero_id = heroes.id
ORDER  BY max DESC,
          localized_name ASC
```

```
1   Sort  (cost=154372.02..154372.30 rows=113 width=22) (actual time=74401.741..74402.752 rows=110 loops=1)
2   [...] Sort Key: (max((count(*)))) DESC, heroes.localized_name
3   [...] Sort Method: quicksort  Memory: 33kB
4   [...] -> Hash Join  (cost=154363.63..154368.16 rows=113 width=22) (actual time=74397.471..74400.601 rows=110 loops=1)
5   [...] Hash Cond: (m_p_d.hero_id = heroes.id)
6   [...] -> HashAggregate  (cost=154360.08..154362.08 rows=200 width=12) (actual time=74395.130..74396.187 rows=110 loops=1)
7   [...] Group Key: m_p_d.hero_id
8   [...] -> HashAggregate  (cost=153337.56..153746.57 rows=40901 width=24) (actual time=66964.124..70706.248 rows=375232 loops=1)
9   [...] Group Key: (row_number() OVER (?) - (row_number() OVER (?))), m_p_d.match_id, m_p_d.hero_id
10  [...] -> WindowAgg  (cost=136070.32..145211.80 rows=406288 width=20) (actual time=47862.636..61933.743 rows=475710 loops=1)
11  [...] -> Sort  (cost=136070.32..137086.04 rows=406288 width=20) (actual time=47862.578..52345.890 rows=475710 loops=1)
12  [...] Sort Key: m_p_d.match_id, g_ob."time"
13  [...] Sort Method: quicksort  Memory: 49453kB
14  [...] -> WindowAgg  (cost=88063.04..98220.24 rows=406288 width=20) (actual time=28618.384..43004.493 rows=475710 loops=1)
15  [...] -> Sort  (cost=88063.04..89078.76 rows=406288 width=12) (actual time=28618.331..33170.081 rows=475710 loops=1)
16  [...] Sort Key: m_p_d.match_id, m_p_d.hero_id, g_ob."time"
17  [...] Sort Method: quicksort  Memory: 34587kB
18  [...] -> Hash Join  (cost=22408.00..50212.96 rows=406288 width=12) (actual time=9819.090..23837.236 rows=475710 loops=1)
19  [...] Hash Cond: (g_ob.match_player_detail_id_1 = m_p_d.id)
20  [...] -> Seq Scan on game_objectives g_ob  (cost=0.00..26738.45 rows=406288 width=8) (actual time=0.021..4863.542 rows=475710 loops=1)
21  [...] Filter: ((match_player_detail_id_1 IS NOT NULL) AND (subtype = 'CHAT_MESSAGE_TOWER_KILL'::text))
22  [...] Rows Removed by Filter: 697686
23  [...] -> Hash  (cost=16158.00..16158.00 rows=500000 width=12) (actual time=9818.479..9818.488 rows=500000 loops=1)
24  [...] Buckets: 524288  Batches: 1  Memory Usage: 25581kB
25  [...] -> Seq Scan on matches_players_details m_p_d  (cost=0.00..16158.00 rows=500000 width=12) (actual time=0.018..4862.904 rows=500000 loops=1)
26  [...] -> Hash  (cost=2.13..2.13 rows=113 width=14) (actual time=2.284..2.293 rows=113 loops=1)
27  [...] Buckets: 1024  Batches: 1  Memory Usage: 14kB
28  [...] -> Seq Scan on heroes  (cost=0.00..2.13 rows=113 width=14) (actual time=0.023..1.158 rows=113 loops=1)
29  Planning Time: 0.519 ms
30  Execution Time: 74414.133 ms
```

ORM generated:

```
SELECT    anon_1.hero_id         AS anon_1_hero_id,
          heroes.localized_name AS heroes_localized_name,
          anon_1.max_1           AS anon_1_max_1
FROM      (
                    SELECT    anon_2.hero_id     AS hero_id,
                              Max(anon_2.count) AS max_1
                    FROM      (
                                        SELECT    anon_3.match_id AS match_id,
                                                  anon_3.hero_id  AS hero_id,
                                                  Count(*)        AS count
                                        FROM      (
                                                            SELECT    Row_number() OVER (ORDER BY matches_p
layers_details.match_id ASC, game_objectives.time ASC) - Row_number() OVER (partition BY matches_pla
yers_details.match_id, matches_players_details.hero_id ORDER BY matches_players_details.match_id ASC
, game_objectives.time ASC) AS row_num,
                                                                      matches_players_details.match_id

                                                                                                        AS ma
tch_id,
                                                                      matches_players_details.hero_id

                                                                                                       AS her
o_id,
                                                                      game_objectives.time

                                                                                                 AS time
                                                            FROM      game_objectives
                                                            JOIN      matches_players_details
                                                            ON        game_objectives.match_player_detail_i
```

```
d_1 = matches_players_details.id
                                                                WHERE     game_objectives.subtype = 'CHAT_MESSA
GE_TOWER_KILL'
                                                                AND       game_objectives.match_player_detail_i
d_1 IS NOT NULL
                                                                ORDER BY matches_players_details.match_id ASC,
                                                                         game_objectives.time ASC) AS anon_3
                                        GROUP BY anon_3.row_num,
                                                 anon_3.match_id,
                                                 anon_3.hero_id) AS anon_2
                        GROUP BY anon_2.hero_id) AS anon_1
JOIN      heroes
ON        anon_1.hero_id = heroes.id
ORDER BY anon_1.max_1 DESC,
         heroes.localized_name ASC
```

1	Sort (cost=154372.02..154372.30 rows=113 width=22) (actual time=74236.698..74237.704 rows=110 loops=1)
2	[...] Sort Key: (max((count(*)))) DESC, heroes.localized_name
3	[...] Sort Method: quicksort Memory: 33kB
4	[...] -> Hash Join (cost=154363.63..154368.16 rows=113 width=22) (actual time=74232.408..74235.556 rows=110 loops=1)
5	[...] Hash Cond: (matches_players_details.hero_id = heroes.id)
6	[...] -> HashAggregate (cost=154360.08..154362.08 rows=200 width=12) (actual time=74230.088..74231.133 rows=110 loops=1)
7	[...] Group Key: matches_players_details.hero_id
8	[...] -> HashAggregate (cost=153337.56..153746.57 rows=40901 width=24) (actual time=66822.224..70534.322 rows=375232 loops=1)
9	[...] Group Key: (row_number() OVER (?) - (row_number() OVER (?))), matches_players_details.match_id, matches_players_details.hero_id
10	[...] -> WindowAgg (cost=136070.32..145211.80 rows=406288 width=20) (actual time=47719.200..61799.598 rows=475710 loops=1)
11	[...] -> Sort (cost=136070.32..137086.04 rows=406288 width=20) (actual time=47719.147..52240.456 rows=475710 loops=1)
12	[...] Sort Key: matches_players_details.match_id, game_objectives."time"
13	[...] Sort Method: quicksort Memory: 49453kB
14	[...] -> WindowAgg (cost=88063.04..98220.24 rows=406288 width=20) (actual time=28690.451..42879.486 rows=475710 loops=1)
15	[...] -> Sort (cost=88063.04..89078.76 rows=406288 width=12) (actual time=28690.399..33179.427 rows=475710 loops=1)
16	[...] Sort Key: matches_players_details.match_id, matches_players_details.hero_id, game_objectives."time"
17	[...] Sort Method: quicksort Memory: 34587kB
18	[...] -> Hash Join (cost=22408.00..50212.96 rows=406288 width=12) (actual time=9905.236..23919.604 rows=475710 loops=1)
19	[...] Hash Cond: (game_objectives.match_player_detail_id_1 = matches_players_details.id)
20	[...] -> Seq Scan on game_objectives (cost=0.00..26738.45 rows=406288 width=8) (actual time=0.020..4803.601 rows=475710 loops=1)
21	[...] Filter: ((match_player_detail_id_1 IS NOT NULL) AND (subtype = 'CHAT_MESSAGE_TOWER_KILL'::text))
22	[...] Rows Removed by Filter: 697686
23	[...] -> Hash (cost=16158.00..16158.00 rows=500000 width=12) (actual time=9904.645..9904.654 rows=500000 loops=1)
24	[...] Buckets: 524288 Batches: 1 Memory Usage: 25581kB
25	[...] -> Seq Scan on matches_players_details (cost=0.00..16158.00 rows=500000 width=12) (actual time=0.018..4890.741 rows=500000 loops=1)
26	[...] -> Hash (cost=2.13..2.13 rows=113 width=14) (actual time=2.283..2.292 rows=113 loops=1)
27	[...] Buckets: 1024 Batches: 1 Memory Usage: 14kB
28	[...] -> Seq Scan on heroes (cost=0.00..2.13 rows=113 width=14) (actual time=0.022..1.166 rows=113 loops=1)
29	Planning Time: 0.513 ms
30	Execution Time: 74239.906 ms

Summary

ORM generated query is absolutely identical to the original one, so it is impossible to get different performance from it.

Generally, the most expensive operations are JOIN. The execution time of initial query is 74414 ms and the execution time of the ORM generated one – 74239 ms. The difference in execution time can be considered as statistical deviation. Total cost is 154372 of every query.