

Pontificia Universidad Javeriana

Diego Alejandro Negro

Jhoan Alberto Galeano Laya

Departamento de sistemas, Bases de datos

Andrés Oswaldo Calderón
26 de noviembre 2025

I. INTRODUCCION

En esta entrega se priorizan métricas que conectan directamente con decisiones de diseño y UX: duración por mapa, proximidad entre jugadores, trayectorias, hotspots y ritmo de movimiento. Cada consulta fue acompañada de un índice específico, y se publican vistas y una vista materializada para acelerar el consumo analítico.

II. CONSULTAS ANALITICAS

Se emplearon cinco consultas analíticas de las ocho propuestas del profesor, se presentan de la siguiente manera,

1. Duración promedio por mapa

```
doomdb2=# WITH per_game AS (  
doomdb2=#   SELECT game_id, map_id, (MAX(tic) - MIN(tic) + 1) AS duration_tics  
doomdb2=#   FROM telemetry_event  
doomdb2=#   GROUP BY game_id, map_id  
doomdb2=# )  
doomdb2=# SELECT map_id, AVG(duration_tics)::numeric(12,2) AS avg_duration_tics  
doomdb2=# FROM per_game  
doomdb2=# GROUP BY map_id  
doomdb2=# ORDER BY avg_duration_tics DESC  
doomdb2=# LIMIT 20;  
+-----+-----+  
| map_id | avg_duration_tics |  
+-----+-----+  
| 9b40375d-30ff-4dd0-b8f3-9ecfac834b5e | 133778.00 |  
+-----+-----+  
(1 row)  
  
Time: 15.587 ms  
doomdb2=#
```

Fig 1. Consulta promedio por mapa.

Se ejecuta el EXPLAIN ANALYZE antes de crear el index,

```
QUERY PLAN  
Limit (cost=2467.97..2467.98 rows=3 width=53) (actual time=56.807..56.809 rows=1.00 loops=1)  
 Buffers: shared hit=1828  
-> Sort (cost=2467.97..2467.98 rows=3 width=53) (actual time=56.805..56.806 rows=1.00 loops=1)  
   Sort Key: ((avg(per_game.duration_tics))::numeric(12,2)) DESC  
   Sort Method: quicksort Memory: 25kB  
   Buffers: shared hit=1828  
-> GroupAggregate (cost=2467.88..2467.95 rows=3 width=53) (actual time=56.647..56.648 rows=1.00 loops=1)  
   Group Key: per_game.map_id  
   Buffers: shared hit=1828  
-> Sort (cost=2467.88..2467.89 rows=3 width=45) (actual time=56.346..56.347 rows=3.00 loops=1)  
   Sort Key: per_game.map_id  
   Sort Method: quicksort Memory: 25kB  
   Buffers: shared hit=1828  
-> Subquery Scan on per_game (cost=2467.78..2467.86 rows=3 width=45) (actual time=56.026..56.028 rows=3.00 loops=1)  
   Buffers: shared hit=1828  
-> HashAggregate (cost=2467.78..2467.83 rows=3 width=82) (actual time=56.024..56.026 rows=3.00 loops=1)  
   Group Key: telemetry_event.game_id, telemetry_event.map_id  
   Batches: 1 Memory Usage: 32kB  
   Buffers: shared hit=1828  
-> Seq Scan on telemetry_event (cost=0.00..2147.89 rows=31989 width=82) (actual time=0.491..0.494 rows=31989.00 loops=1)  
   Buffers: shared hit=1828  
  
Planning:  
 Buffers: shared hit=11 dirtied=5  
Planning Time: 10.679 ms  
Execution Time: 58.580 ms  
  
(25 rows)
```

Fig 2. Explain analyze antes del indice.

Luego, se ejecuta el EXPLAIN ANALYZE luego de crear el index,

```

QUERY PLAN
Limit (cost=2467.97..2467.98 rows=3 width=53) (actual time=8.332..8.333 rows=1.00 loops=1)
  Buffers: shared hit=1828
  -> Sort (cost=2467.97..2467.98 rows=3 width=53) (actual time=8.332..8.332 rows=1.00 loops=1)
    Sort Key: ((avg(per_game.duration_tics)::numeric(12,2)) DESC
    Sort Method: quicksort Memory: 25kB
    Buffers: shared hit=1828
    -> GroupAggregate (cost=2467.88..2467.95 rows=3 width=53) (actual time=8.324..8.324 rows=1.00 loops=1)
      Group Key: per_game.map_id
      Buffers: shared hit=1828
      -> Sort (cost=2467.88..2467.89 rows=3 width=45) (actual time=8.317..8.318 rows=3.00 loops=1)
        Sort Key: per_game.map_id
        Sort Method: quicksort Memory: 25kB
        Buffers: shared hit=1828
        -> Subquery Scan on per_game (cost=2467.78..2467.86 rows=3 width=45) (actual time=8.309..8.311 rows=3.00 loops=1)
          Buffers: shared hit=1828
          -> HashAggregate (cost=2467.78..2467.83 rows=3 width=82) (actual time=8.309..8.309 rows=3.00 loops=1)
            Group Key: telemetry_event.game_id, telemetry_event.map_id
            Batches: 1 Memory Usage: 32kB
            Buffers: shared hit=1828
            -> Seq Scan on telemetry_event (cost=0.00..2147.89 rows=31989 width=82) (actual time=0.013..1.745 rows=31989.00 loops=1)
              Buffers: shared hit=1828
Planning:
  Buffers: shared hit=9
Planning Time: 0.329 ms
Execution Time: 8.394 ms
(25 rows)

```

Fig 3. Explain Analyze después del índice.

Para un Juego competitivo la duración de cada partida por mapa es KPI de balance, entonces, si un mapa concentra partidas demasiado largas o cortas, puede indicar problemas de flujo.

Para el diseño de la consulta se toma cada (game_id y map_id) y se calcula MAX(tic)-MIN(tic)+1. Al promediar por mapd_id se abstrae las diferencias entre partidas individuales y se obtiene una medida estable del tiempo típico del mapa.

El índice propuesto que es un avg_duration_tics alto señala mapas donde la progresión es lenta o el objetivo esta lejos, un valor bajo sugiere mapas tal ves resolubles por rutas dominantes.

En cuestión de decisión técnica, se creó un idx_tevt_game_map_tic para facilitar MIN/MAX(tic) por agrupación. En la toma realizada para el dataset la mejora de tiempo es pequeña.

A nivel general, las partidas en el mapa analizado duran mas o menos lo mismo, esto hablar de un flujo estable, es decir, el mapa no se vuelve impredecible, si en algún momento se quieren tener partidas mas cortas o mas rápidas, hay que intervenir en puntos de espera o rutas largas.

2. Proximidad promedio entre pares de jugadores

```

doomdb2=# WITH pairs AS (
doomdb2=#   SELECT t1.game_id,
doomdb2=#         LEAST(t1.player_id, t2.player_id) AS p_a,
doomdb2=#         GREATEST(t1.player_id, t2.player_id) AS p_b,
doomdb2=#         sqrt( power(t1.pos_x - t2.pos_x,2)
doomdb2=#           + power(t1.pos_y - t2.pos_y,2)
doomdb2=#           + power(t1.pos_z - t2.pos_z,2) ) AS dist
doomdb2=#   FROM telemetry_event t1
doomdb2=#   JOIN telemetry_event t2
doomdb2=#     ON t1.game_id = t2.game_id
doomdb2=#    AND t1.player_id < t2.player_id
doomdb2=#    AND t2.tic BETWEEN t1.tic-1 AND t1.tic+1
doomdb2=# )
doomdb2=# SELECT game_id, p_a, p_b,
doomdb2=#         AVG(dist)::numeric(12,2) AS avg_distance,
doomdb2=#         COUNT(*) FILTER (WHERE dist <= 256) AS close_tics
doomdb2=# FROM pairs
doomdb2=# GROUP BY game_id, p_a, p_b
doomdb2=# ORDER BY close_tics DESC, avg_distance ASC
doomdb2=# LIMIT 20;
 game_id | p_a | p_b | avg_distance | close_tics
-----+-----+-----+-----+-----
(0 rows)

```

Fig 4. Consulta promedio entre pares de jugadores.

Primero, se ejecuta EXPLAIN ANALYZE antes de crear el index,

```
| Planning:
|
| Buffers: shared hit=16 dirtied=3
|
| Planning Time: 3.412 ms
|
| Execution Time: 147.909 ms
|
```

Fig 5. Explain Analyze antes de crear el índice.

Luego, se ejecuta el EXPLAIN ANALYZE luego de crear el index,

```
| Planning:
|
| Buffers: shared hit=33 read=1
|
| Planning Time: 1.569 ms
|
| Execution Time: 144.814 ms
|
+-----+
|-----+
(36 rows)
```

Fig 6. Explain Analyze después de crear el índice.

La consulta para la proximidad promedio entre pares de jugadores tiene un Auto-join por mismo game_id y tics cercanos, entonces, calcula la distancia entre cada par (p_a, p_b) en la misma ventana temporal y reporta promedio y numero de tics cercanos.

El Explain Analyze tanto antes como después muestra tiempos prácticamente iguales, (147 ms vs 145 ms) lo que tiene sentido, ya que el patros en un auto-join por rango.

El índice creado es idx_tevt_game_tic_player (game_id, tic, player_id) ayuda a posicionar por (game, tic) antes de combinar jugadores, aun así, si no hay pares cercanos la consulta seguirá devolviendo 0, el índice solo reduce trabajo cuando existen coincidencias.

A nivel general la consulta esta bien, el resultado de cero es un hallazgo del dataset, no un error en general.

Como conclusión con el criterio que se colocó, no aparecen parejas por decirlo pegadas por mucho tiempo. Si se aumentara la ventana temporal o se quitara la condición de tics contiguos, aparece señal.

3. Trayectoria mínima y máxima por jugador.

```
doomdb=# WITH steps AS (
doomdb=# SELECT game_id, player_id, tic,
doomdb=# pos_x, pos_y, pos_z,
doomdb=# LAG(pos_x) OVER (PARTITION BY game_id, player_id ORDER BY tic) AS px_prev,
doomdb=# LAG(pos_y) OVER (PARTITION BY game_id, player_id ORDER BY tic) AS py_prev,
doomdb=# LAG(pos_z) OVER (PARTITION BY game_id, player_id ORDER BY tic) AS pz_prev
doomdb=# FROM telemetry_event
doomdb=# ),
doomdb=# per_game AS (
doomdb=# SELECT game_id, player_id,
doomdb=# SUM( CASE WHEN px_prev IS NULL THEN 0
doomdb=# ELSE sqrt( power(pos_x - px_prev,2)
doomdb=# + power(pos_y - py_prev,2)
doomdb=# + power(pos_z - pz_prev,2) )
doomdb=# END ) AS total_distance
doomdb=# FROM steps
doomdb=# GROUP BY game_id, player_id
doomdb=# )
doomdb=# SELECT player_id,
doomdb=# MIN(total_distance)::numeric(12,2) AS min_traj,
doomdb=# MAX(total_distance)::numeric(12,2) AS max_traj
doomdb=# FROM per_game
doomdb=# GROUP BY player_id
doomdb=# ORDER BY max_traj DESC
doomdb=# LIMIT 20;
```

player_id	min_traj	max_traj
60fc2e54-7379-40a6-bf5a-fdc4fd21db82	16440975.15	16471348.83
dc2c7785-1775-402b-b416-462cc7ef3807	16442346.84	16442346.84

(2 rows)

Fig 6. Consulta promedio por mapa.

Luego de crear el index,

```
QUERY PLAN
Limit (cost=7338.11..7338.13 rows=6 width=69) (actual time=58.855..58.857 rows=2.00 loops=1)
  Buffers: shared hit=976
  -> Sort (cost=7338.11..7338.13 rows=6 width=69) (actual time=58.854..58.855 rows=2.00 loops=1)
    Sort Key: ((max(per_game.total_distance))::numeric(12,2)) DESC
    Sort Method: quicksort Memory: 25kB
    Buffers: shared hit=976
    -> GroupAggregate (cost=7337.86..7338.04 rows=6 width=69) (actual time=58.844..58.848 rows=2.00 loops=1)
      Group Key: per_game.player_id
      Buffers: shared hit=976
      -> Sort (cost=7337.86..7337.87 rows=6 width=69) (actual time=58.829..58.830 rows=3.00 loops=1)
        Sort Key: per_game.player_id
        Sort Method: quicksort Memory: 25kB
        Buffers: shared hit=976
        -> Subquery Scan on per_game (cost=1619.81..7337.78 rows=6 width=69) (actual time=27.889..58.815 rows=3.00 loops=1)
          Buffers: shared hit=976
          -> GroupAggregate (cost=1619.81..7337.72 rows=6 width=69) (actual time=27.888..58.812 rows=3.00 loops=1)
            Group Key: telemetry_event.game_id, telemetry_event.player_id
            Buffers: shared hit=976
            -> WindowAgg (cost=1619.81..6888.18 rows=31989 width=130) (actual time=7.881..49.795 rows=31989.00 loops=1)
              Window: #1 AS (PARTITION BY telemetry_event.game_id, telemetry_event.player_id ORDER BY telemetry_event.tic)
              Storage: Memory Maximum Storage: 17kB
              Buffers: shared hit=976
              -> Incremental Sort (cost=1619.68..5258.37 rows=31989 width=106) (actual time=6.989..33.821 rows=31989.00 loops=1)
                Sort Key: telemetry_event.game_id, telemetry_event.player_id, telemetry_event.tic
                Presorted Key: telemetry_event.game_id
                Full-sort Groups: 3 Sort Method: quicksort Average Memory: 33kB Peak Memory: 33kB
                Pre-sorted Groups: 3 Sort Method: quicksort Average Memory: 1717kB Peak Memory: 1717kB
                Buffers: shared hit=976
                -> Index Scan using idx_text_game on telemetry_event (cost=0.29..2718.33 rows=31989 width=106) (actual time=0.019..9.553 rows=31989.00 loops=1)
                  Index Searches: 1
                  Buffers: shared hit=976
Planning:
  Buffers: shared hit=20
Planning Time: 0.554 ms
Execution Time: 59.163 ms
```

Fig 7. Explain Analyze después de crear el índice.

La consulta creada para la trayectoria mínima y máxima por jugador usa LAG() particionado por (game_id, player_id) y ordenado por tic para medir distancia paso a paso, suma por jugador, luego reporta min y max de esa distancia total por player_id.

Antes de aplicar el index analyze el motor deber ordenar por (game_id, player_id, tic) para cada partición, luego de aplicar el índice idx_text_game_player_tic (game_id, player_id, tic) es el orden que exigen las funciones de ventana, en las capturas obtenidas el tiempo cae o permanece estable.

En conclusión, hay jugadores que recorren bastante mas que otros, esto muestra que el mapa permite distintos estilos.

4. Hotspot (grid 250×250) por episodio y mapa.

```
doomdb2=# WITH grid AS (  
doomdb2=#   SELECT  
doomdb2=#     game_id, map_id,  
doomdb2=#     FLOOR(pos_x/250.0)::int AS cell_x,  
doomdb2=#     FLOOR(pos_y/250.0)::int AS cell_y,  
doomdb2=#     COUNT(*) AS hits  
doomdb2=#   FROM telemetry_event  
doomdb2=#   GROUP BY game_id, map_id, FLOOR(pos_x/250.0), FLOOR(pos_y/250.0)  
doomdb2=# ),  
doomdb2=# ranked AS (  
doomdb2=#   SELECT *,  
doomdb2=#     ROW_NUMBER() OVER (PARTITION BY game_id, map_id ORDER BY hits DESC) AS rk  
doomdb2=#   FROM grid  
doomdb2=# )  
doomdb2=# SELECT game_id, map_id, cell_x, cell_y, hits  
doomdb2=# FROM ranked  
doomdb2=# WHERE rk = 1  
doomdb2=# ORDER BY hits DESC  
doomdb2=# LIMIT 20;  
+-----+-----+-----+-----+-----+  
| game_id | map_id | cell_x | cell_y | hits |  
+-----+-----+-----+-----+-----+  
| 5793d4a6-692a-4fa2-8629-588444f113cd | 9b40375d-30ff-4dd0-b8f3-9ecfac834b5e | 4 | 4 | 502 |  
| 5ac83ebe-94d2-4099-8f69-55de98245fea | 9b40375d-30ff-4dd0-b8f3-9ecfac834b5e | 4 | 4 | 502 |  
| fecfb2bc-6cef-4084-b952-11c1ca339daa | 9b40375d-30ff-4dd0-b8f3-9ecfac834b5e | 4 | 4 | 502 |  
+-----+-----+-----+-----+-----+  
(3 rows)
```

Fig 8. Consulta Hotspot.

Primero, se ejecuta el EXPLAIN ANALYZE antes de crear el index,

```
| Planning:  
|  
| Buffers: shared hit=9  
|  
| Planning Time: 0.497 ms  
|  
| Execution Time: 18.340 ms  
|  
+-----+  
-----+  
(30 rows)
```

Fig 9. Explain Analyze antes de crear el índice.

Luego, el EXPLAIN ANALYZE luego de crear el index,

```
| Planning:  
|  
| Buffers: shared hit=18  
|  
| Planning Time: 0.381 ms  
|  
| Execution Time: 12.962 ms  
|  
+-----+  
-----+
```

Fig 10. Explain Analyze después de crear el índice.

Para la creación de la consulta se agrupa por celdas discretizadas dentro de (game_id, map_id) y cuenta hits por celda, luego toma el topo -1 por (game, map).

La creación del index idx_tevt_grid_250 ON telemetry_event(map_id, game_id, floor(pos_x/250.0), floor(pos_y/250.0)) permite agrupar y ordenar.

Como conclusión, se tiene una consulta y resultados coherentes, un índice funcional adecuado al patrón de agregación por grid.

En conclusión, general, se repite la misma zona “caliente”, es decir, en esa zona se concentra la acción.

5. Distancia total y velocidad promedio por jugador

```
doomdb2=# WITH steps AS (  
doomdb2=#   SELECT game_id, player_id, tic,  
doomdb2=#     pos_x, pos_y, pos_z,  
doomdb2=#     LAG(pos_x) OVER (PARTITION BY game_id, player_id ORDER BY tic) AS px_prev,  
doomdb2=#     LAG(pos_y) OVER (PARTITION BY game_id, player_id ORDER BY tic) AS py_prev,  
doomdb2=#     LAG(pos_z) OVER (PARTITION BY game_id, player_id ORDER BY tic) AS pz_prev,  
doomdb2=#     LAG(tic) OVER (PARTITION BY game_id, player_id ORDER BY tic) AS tic_prev  
doomdb2=# FROM telemetry_event  
doomdb2=# ),  
doomdb2=# measures AS (  
doomdb2=#   SELECT player_id,  
doomdb2=#     SUM( CASE WHEN px_prev IS NULL THEN 0  
doomdb2=#       ELSE sqrt( power(pos_x - px_prev,2)  
doomdb2=#         + power(pos_y - py_prev,2)  
doomdb2=#         + power(pos_z - pz_prev,2) )  
doomdb2=#       END ) AS total_distance,  
doomdb2=#     SUM( GREATEST(tic - COALESCE(tic_prev,tic), 1) ) AS total_tics  
doomdb2=# FROM steps  
doomdb2=# GROUP BY player_id  
doomdb2=# )  
doomdb2=# SELECT player_id,  
doomdb2=#   total_distance::numeric(12,2) AS total_distance,  
doomdb2=#   (total_distance / NULLIF(total_tics,0))::numeric(12,4) AS avg_speed_per_tic  
doomdb2=# FROM measures  
doomdb2=# ORDER BY total_distance DESC  
doomdb2=# LIMIT 20;
```

player_id	total_distance	avg_speed_per_tic
69fc2e54-7379-40a6-bf5a-fdc4fd21db82	32912323.98	122.6443
dc2c7785-1775-402b-b416-462cc7ef3807	16442346.84	122.5413

(2 rows)

Time: 52.072 ms

Fig 11. Consulta Distancia total y velocidad.

Primero, se ejecuta el EXPLAIN ANALYZE antes de crear el index,

```
QUERY PLAN  
Limit (cost=7577.83..7577.83 rows=2 width=69) (actual time=59.508..59.510 rows=2 loops=1)  
  Buffers: shared hit=976  
  -> Sort (cost=7577.83..7577.83 rows=2 width=69) (actual time=59.508..59.508 rows=2 loops=1)  
    Sort Key: ((measures.total_distance)::numeric(12,2)) DESC  
    Sort Method: quicksort Memory: 25kB  
    Buffers: shared hit=976  
    -> Subquery Scan on measures (cost=7577.76..7577.82 rows=2 width=69) (actual time=59.500..59.503 rows=2 loops=1)  
      Buffers: shared hit=976  
      -> HashAggregate (cost=7577.76..7577.78 rows=2 width=77) (actual time=59.480..59.485 rows=2 loops=1)  
        Group Key: telemetry_event.player_id  
        Batches: 1 Memory Usage: 32kB  
        Buffers: shared hit=976  
        -> WindowAgg (cost=1619.88..6138.25 rows=31989 width=138) (actual time=9.479..49.846 rows=31989 loops=1)  
          Window: w1 AS (PARTITION BY telemetry_event.game_id, telemetry_event.player_id ORDER BY telemetry_event.tic)  
          Storage: Memory Maximum Storage: 17kB  
          Buffers: shared hit=976  
          -> Incremental Sort (cost=1619.76..5258.86 rows=31989 width=106) (actual time=9.467..32.498 rows=31989 loops=1)  
            Sort Key: telemetry_event.game_id  
            Full-sort Groups: 3 Sort Method: quicksort Average Memory: 33kB Peak Memory: 33kB  
            Pre-sorted Groups: 3 Sort Method: quicksort Average Memory: 17176kB Peak Memory: 17176kB  
            Buffers: shared hit=976  
            -> Index Scan using idx_tevt_game on telemetry_event (cost=0.29..2718.52 rows=31989 width=106) (actual time=8.015..11.729 rows=31989 loops=1)  
              Index Searches: 1  
              Buffers: shared hit=976  
Planning:  
  Buffers: shared hit=9  
  Planning Time: 0.482 ms  
  Execution Time: 59.607 ms  
(29 rows)
```

Fig 12. Explain Analyze antes de crear el índice.

Luego, el EXPLAIN ANALYZE luego de crear el index,

```

QUERY PLAN
Limit (cost=7577.66..7577.67 rows=2 width=69) (actual time=52.028..52.030 rows=2.00 loops=1)
  Buffers: shared hit=976
  -> Sort (cost=7577.66..7577.67 rows=2 width=69) (actual time=52.027..52.029 rows=2.00 loops=1)
    Sort Key: ((measures.total_distance)::numeric(12,2)) DESC
    Sort Method: quicksort Memory: 25kB
    Buffers: shared hit=976
    -> Subquery Scan on measures (cost=7577.59..7577.65 rows=2 width=69) (actual time=52.018..52.021 rows=2.00 loops=1)
      Buffers: shared hit=976
      -> HashAggregate (cost=7577.59..7577.62 rows=2 width=77) (actual time=52.004..52.006 rows=2.00 loops=1)
        Group Key: telemetry_event.player_id
        Batches: 1 Memory Usage: 32kB
        Buffers: shared hit=976
        -> WindowAgg (cost=1619.02..6138.09 rows=31989 width=138) (actual time=5.993..42.804 rows=31989.00 loops=1)
          Window: w1 AS (PARTITION BY telemetry_event.game_id, telemetry_event.player_id ORDER BY telemetry_event.tic)
          Storage: Memory Maximum Storage: 17kB
          Buffers: shared hit=976
          -> Incremental Sort (cost=1619.08..5358.39 rows=31989 width=106) (actual time=5.984..25.963 rows=31989.00 loops=1)
            Sort Key: telemetry_event.game_id, telemetry_event.player_id, telemetry_event.tic
            Presorted Key: telemetry_event.game_id
            Full-sort Groups: 3 Sort Method: quicksort Average Memory: 33kB Peak Memory: 33kB
            Pre-sorted Groups: 3 Sort Method: quicksort Average Memory: 171kB Peak Memory: 171kB
            Buffers: shared hit=976
            -> Index Scan using idx_tevt_game on telemetry_event (cost=0.29..2718.35 rows=31989 width=106) (actual time=0.016..8.282 rows=31989.00 loops=1)
              Index Searches: 1
              Buffers: shared hit=976
Planning:
  Buffers: shared hit=20
Planning Time: 0.625 ms
Execution Time: 52.210 ms
(29 rows)

```

Fig 13. Explain Analyze después de crear el índice.

Para esta consulta se aplica el mismo concepto que la consulta número 3, pero además se generan los tics efectivos ($\text{GREATEST}(\text{tic}-\text{tic_prev}, 1)$) y se produce `total_distance` y `avg_speed_per_tic` igual a `distance/tics`.

En las capturas se observa que al crear el índice `idx_tevt_game_player_tic` el motor lee en orden y evita sorts altos, en la captura se observa que se estabilizar y los tiempos son similares y normal por tamaño actual.

Como conclusión general, la velocidad promedio es muy parecida entre jugadores, lo que cambia mas es cuanto caminan, es decir, las reglas del movimiento son consistentes y lo que marca la diferencia es la ruta elegida.

II. EXPLAIN ANALYZE PARA TRES CONSULTAS ANALITICAS CREADAS. (antes de los index).

1. Trayectoria por jugador.

```

QUERY PLAN
Limit (cost=4177.42..4179.92 rows=1000 width=130) (actual time=21.486..21.562 rows=1000.00 loops=1)
  Buffers: shared hit=2154
  -> Sort (cost=4177.42..4190.75 rows=5332 width=130) (actual time=21.485..21.519 rows=1000.00 loops=1)
    Sort Key: t.tic
    Sort Method: top-N heapsort Memory: 306kB
    Buffers: shared hit=2154
    -> WindowAgg (cost=3728.07..3885.07 rows=5332 width=130) (actual time=15.726..20.402 rows=10663.00 loops=1)
      Window: w1 AS (PARTITION BY t.game_id, t.player_id ORDER BY t.tic)
      Storage: Memory Maximum Storage: 17kB
      Buffers: shared hit=2154
      -> Sort (cost=3738.00..3751.77 rows=5332 width=106) (actual time=15.714..15.948 rows=10663.00 loops=1)
        Sort Key: t.game_id, t.player_id, t.tic
        Sort Method: quicksort Memory: 171kB
        Buffers: shared hit=2154
        -> Nested Loop (cost=2388.18..3488.38 rows=5332 width=106) (actual time=11.283..13.955 rows=10663.00 loops=1)
          Buffers: shared hit=2154
          -> Limit (cost=2387.90..2387.90 rows=1 width=82) (actual time=11.234..11.236 rows=1.00 loops=1)
            Buffers: shared hit=1828
            -> Sort (cost=2387.90..2387.91 rows=6 width=82) (actual time=11.233..11.234 rows=1.00 loops=1)
              Sort Key: (count(*)) DESC
              Sort Method: top-N heapsort Memory: 25kB
              Buffers: shared hit=1828
              -> HashAggregate (cost=2387.81..2387.87 rows=6 width=82) (actual time=11.225..11.226 rows=3.00 loops=1)
                Group Key: telemetry_event.game_id, telemetry_event.player_id
                Batches: 1 Memory Usage: 32kB
                Buffers: shared hit=1828
                -> Seq Scan on telemetry_event (cost=0.00..2147.89 rows=31989 width=74) (actual time=0.026..2.164 rows=31989.00 loops=1)
                  Buffers: shared hit=1828
          -> Index Scan using idx_tevt_game on telemetry_event t (cost=0.29..967.15 rows=5332 width=106) (actual time=0.044..1.634 rows=10663.00 loops=1)
            Index Cond: ((game_id)::text = (telemetry_event.game_id)::text)
            Filter: ((telemetry_event.player_id)::text = (player_id)::text)
            Index Searches: 1
            Buffers: shared hit=326
Planning Time: 0.958 ms
Execution Time: 21.935 ms
(35 rows)

```

Fig 14. Explain Analyze consulta trayectoria por jugador.

2. Heatmap.

```
QUERY PLAN
Limit (cost=3090.99..3091.12 rows=50 width=106) (actual time=18.703..18.708 rows=50.00 loops=1)
  Buffers: shared hit=1828
  -> Sort (cost=3090.99..3099.18 rows=3274 width=106) (actual time=18.701..18.704 rows=50.00 loops=1)
    Sort Key: (count(*)) DESC
    Sort Method: top-N heapsort Memory: 36kB
    Buffers: shared hit=1828
    -> HashAggregate (cost=2867.64..2982.23 rows=3274 width=106) (actual time=18.595..18.652 rows=405.00 loops=1)
      Group Key: game_id, map_id, floor((pos_x / '250'::double precision)), floor((pos_y / '250'::double precision))
      Batches: 1 Memory Usage: 153kB
      Buffers: shared hit=1828
      -> Seq Scan on telemetry_event (cost=0.00..2467.78 rows=31989 width=106) (actual time=0.026..9.742 rows=31989.00 loops=1)
        Buffers: shared hit=1828
Planning Time: 0.268 ms
Execution Time: 19.605 ms

(14 rows)

Time: 20.734 ms
```

Fig 15. Explain Analyze consulta Heatmap.

3. duración por mapa.

```
QUERY PLAN
Limit (cost=2467.97..2467.98 rows=3 width=53) (actual time=10.580..10.581 rows=1.00 loops=1)
  Buffers: shared hit=1828
  -> Sort (cost=2467.97..2467.98 rows=3 width=53) (actual time=10.579..10.580 rows=1.00 loops=1)
    Sort Key: ((avg(per_game.duration_tics))::numeric(12,2)) DESC
    Sort Method: quicksort Memory: 25kB
    Buffers: shared hit=1828
    -> GroupAggregate (cost=2467.88..2467.95 rows=3 width=53) (actual time=10.573..10.574 rows=1.00 loops=1)
      Group Key: per_game.map_id
      Buffers: shared hit=1828
      -> Sort (cost=2467.88..2467.89 rows=3 width=45) (actual time=10.565..10.566 rows=3.00 loops=1)
        Sort Key: per_game.map_id
        Sort Method: quicksort Memory: 25kB
        Buffers: shared hit=1828
        -> Subquery Scan on per_game (cost=2467.78..2467.86 rows=3 width=45) (actual time=10.553..10.555 rows=3.00 loops=1)
          Buffers: shared hit=1828
          -> HashAggregate (cost=2467.78..2467.83 rows=3 width=82) (actual time=10.552..10.553 rows=3.00 loops=1)
            Group Key: telemetry_event.game_id, telemetry_event.map_id
            Batches: 1 Memory Usage: 32kB
            Buffers: shared hit=1828
            -> Seq Scan on telemetry_event (cost=0.00..2147.89 rows=31989 width=82) (actual time=0.022..1.646 rows=31989.00 loops=1)
              Buffers: shared hit=1828
Planning Time: 0.313 ms
Execution Time: 10.639 ms

(23 rows)

Time: 11.482 ms
```

Fig 16. Explain Analyze consulta duración por mapa.

Al ejecutar el EXPLAIN ANALYZE antes de crear los índices se observa una trayectoria por jugador con WindowAgg y en duración por mapa una subconsulta por (game, map). En general hay mucho seq scan.

Ahora, se crean los index,

```
Time: 11.482 ms
doomdb2=# CREATE INDEX IF NOT EXISTS idx_tevt_game_player_tic
doomdb2=# ON telemetry_event (game_id, player_id, tic);
NOTICE: relation "idx_tevt_game_player_tic" already exists, skipping
CREATE INDEX
Time: 4.965 ms
doomdb2=#
doomdb2=#
doomdb2=# CREATE INDEX IF NOT EXISTS idx_tevt_grid_250
doomdb2=# ON telemetry_event (map_id, game_id, floor(pos_x/250.0), floor(pos_y/250.0));
CREATE INDEX
Time: 115.814 ms
doomdb2=#
doomdb2=# CREATE INDEX IF NOT EXISTS idx_tevt_game_map_tic
doomdb2=# ON telemetry_event (game_id, map_id, tic);
NOTICE: relation "idx_tevt_game_map_tic" already exists, skipping
CREATE INDEX
Time: 1.776 ms
doomdb2=#
```

Fig 17. creación de índices.

- idx_tevt_game_player_tic (game_id, player_id, tic), ventanas y filtros por jugador en el tiempo.

- idx_tevt_grid_250 (map_id, game_id, floor(pos_x/250.0), floor(pos_y/250.0)), agregados por celda 250×250.
- idx_tevt_game_map_tic (game_id, map_id, tic), extremos MIN/MAX(tic) por {game,map} y recorridos temporales por mapa.

Esto, debido a que cada índice refleja la clave de partición mas el orden de la consulta, de este modo el optimizador puede evitar sorts o reducir lecturas.

Se vuelve a ejecutar cada uno de los EXPLAIN ANALYZE para cada una de las consultas,

1. Trayectoria por jugador.

```

QUERY PLAN
Limit (cost=4177.42..4179.92 rows=1000 width=130) (actual time=24.396..24.465 rows=1000.00 loops=1)
  Buffers: shared hit=2154
  -> Sort (cost=4177.42..4190.75 rows=5332 width=130) (actual time=24.395..24.425 rows=1000.00 loops=1)
    Sort Key: t.tic
    Sort Method: top-N heapsort  Memory: 306kB
    Buffers: shared hit=2154
    -> WindowAgg (cost=3738.47..3885.07 rows=5332 width=130) (actual time=18.365..23.049 rows=10663.00 loops=1)
      Window: w1 AS (PARTITION BY t.game_id, t.player_id ORDER BY t.tic)
      Storage: Memory Maximum Storage: 17kB
      Buffers: shared hit=2154
      -> Sort (cost=3738.44..3751.77 rows=5332 width=106) (actual time=18.352..18.596 rows=10663.00 loops=1)
        Sort Key: t.game_id, t.player_id, t.tic
        Sort Method: quicksort  Memory: 1717kB
        Buffers: shared hit=2154
        -> Nested Loop (cost=2388.18..3408.38 rows=5332 width=106) (actual time=10.074..15.201 rows=10663.00 loops=1)
          Buffers: shared hit=2154
          -> Limit (cost=2387.90..2387.90 rows=1 width=82) (actual time=10.019..10.023 rows=1.00 loops=1)
            Buffers: shared hit=1828
            -> Sort (cost=2387.90..2387.91 rows=6 width=82) (actual time=10.018..10.021 rows=1.00 loops=1)
              Sort Key: (count(*)) DESC
              Sort Method: top-N heapsort  Memory: 25kB
              Buffers: shared hit=1828
              -> HashAggregate (cost=2387.81..2387.87 rows=6 width=82) (actual time=10.010..10.012 rows=3.00 loops=1)
                Group Key: telemetry_event.game_id, telemetry_event.player_id
                Batches: 1 Memory Usage: 32kB
                Buffers: shared hit=1828
                -> Seq Scan on telemetry_event (cost=0.00..2147.89 rows=31989 width=74) (actual time=0.010..2.404 rows=31989.00 loops=1)
                  Buffers: shared hit=1828
          -> Index Scan using idx_tevt_game on telemetry_event t (cost=0.29..967.15 rows=5332 width=106) (actual time=0.048..3.664 rows=10663.00 loops=1)
            Index Cond: ((game_id)::text = (telemetry_event.game_id)::text)
            Filter: ((telemetry_event.player_id)::text = (player_id)::text)
            Index Searches: 1
            Buffers: shared hit=326
Planning:
  Buffers: shared hit=32 read=1
Planning Time: 1.778 ms
Execution Time: 24.813 ms
(37 rows)
Time: 27.248 ms

```

Fig 18. Explain Analyze después consulta trayectoria por jugador.

2. Heatmap.

```

QUERY PLAN
Limit (cost=3090.99..3091.12 rows=50 width=106) (actual time=23.563..23.569 rows=50.00 loops=1)
  Buffers: shared hit=1828
  -> Sort (cost=3090.99..3099.18 rows=3274 width=106) (actual time=23.561..23.563 rows=50.00 loops=1)
    Sort Key: (count(*)) DESC
    Sort Method: top-N heapsort  Memory: 36kB
    Buffers: shared hit=1828
    -> HashAggregate (cost=2867.64..2982.23 rows=3274 width=106) (actual time=23.391..23.489 rows=405.00 loops=1)
      Group Keys: game_id, map_id, floor((pos_x / '250'::double precision)), floor((pos_y / '250'::double precision))
      Batches: 1 Memory Usage: 153kB
      Buffers: shared hit=1828
      -> Seq Scan on telemetry_event (cost=0.00..2467.78 rows=31989 width=106) (actual time=0.017..15.276 rows=31989.00 loops=1)
        Buffers: shared hit=1828
Planning:
  Buffers: shared hit=4
Planning Time: 0.449 ms
Execution Time: 23.662 ms
(16 rows)

```

Fig 19. Explain Analyze después consulta Heatmap.

3. duración por mapa.

```
QUERY PLAN
Limit: (cost=2467.97..2467.98 rows=3 width=53) (actual time=10.690..10.693 rows=1.00 loops=1)
 Buffers: shared hit=1828
-> Sort (cost=2467.97..2467.98 rows=3 width=53) (actual time=10.690..10.691 rows=1.00 loops=1)
   Sort Key: ((avg(per_game.duration_tic))::numeric(12,2)) DESC
   Sort Method: quicksort Memory: 25kB
   Buffers: shared hit=1828
-> GroupAggregate (cost=2467.88..2467.95 rows=3 width=53) (actual time=10.685..10.686 rows=1.00 loops=1)
   Group Key: per_game.map_id
   Buffers: shared hit=1828
-> Sort (cost=2467.88..2467.89 rows=3 width=45) (actual time=10.675..10.676 rows=3.00 loops=1)
   Sort Key: per_game.map_id
   Sort Method: quicksort Memory: 25kB
   Buffers: shared hit=1828
-> Subquery Scan on per_game (cost=2467.78..2467.86 rows=3 width=45) (actual time=10.661..10.663 rows=3.00 loops=1)
   Buffers: shared hit=1828
-> HashAggregate (cost=2467.78..2467.83 rows=3 width=82) (actual time=10.659..10.661 rows=3.00 loops=1)
   Group Key: telemetry_event.game_id, telemetry_event.map_id
   Batches: 1 Memory Usage: 32kB
   Buffers: shared hit=1828
-> Seq Scan on telemetry_event (cost=0.00..2147.89 rows=31989 width=82) (actual time=0.016..2.032 rows=31989.00 loops=1)
   Buffers: shared hit=1828

Planning Time: 0.252 ms
Execution Time: 10.735 ms
(23 rows)
```

Fig 20. Explain Analyze después consulta duración por mapa.

Luego de ejecutar el EXPLAIN ANALYZE después de crear los índices se observa que,

- Trayectoria por jugador: comienza a usar Index Scan using idx_tevt_game_player_tic.
- Heatmap: sigue el Seq Scan (barre 31k filas), pero el índice funcional queda listo para crecer; en tus medidas bajó ~20.7 ms → ~23.6 ms (variación normal por caché/ruido).
- Duración por mapa: plan estable y rápido (~10–11 ms).

Esto, para demostrar que los índices si cambian la ruta del plan, se obtiene menos lectura de paginas compartidas, aparece uso de los índices creados y baja el trabajo de ordenamiento en memoria.

III. VISTAS POR JUGADOR. (View_player_measures).

```
Time: 6.321 ms
doomdb2=# -- pequeña comprobación
doomdb2=# SELECT * FROM view_player_measures ORDER BY total_distance DESC LIMIT 10;
+-----+-----+-----+
| player_id | total_distance | avg_speed_per_tic |
+-----+-----+-----+
| 60fc2e54-7379-40a6-bf5a-fdc4fd21db82 | 32912323.98 | 122.6443 |
| dc2c7785-1775-402b-b416-462cc7ef3807 | 16442346.84 | 122.5413 |
+-----+-----+-----+
(2 rows)

Time: 47.041 ms
```

Fig 21. Vista por jugador.

La vista creada por medidas por jugador, precaculca por player_id la total_distance y avg_speed_per_tic, con esto se puede comprobar mostrando los 2 jugadores con distancias y velocidades coherentes.

Esta vista sirve para recalcular CTEs largos cada vez que se quiera un ranking o comparar jugadores o armar un top.

La velocidad promedio se mantiene pareja, lo que cambia es la distancia recorrida, esto valida que el sistema de movimiento es consistente y que la diferencia está en las rutas.

IV. Vista top hotspot por (game,map) en grid 250.

```
CREATE VIEW
Time: 1.889 ms
doomdb2=# -- comprobación
doomdb2=# SELECT * FROM view_hotspots_250 LIMIT 10;
```

game_id	map_id	cell_x	cell_y	hits
5793d4a6-692a-4fa2-8629-588444f113cd	9b40375d-30ff-4dd0-b8f3-9ecfac834b5e	4	4	502
5ac83ebe-94d2-4099-8f69-55de98245fea	9b40375d-30ff-4dd0-b8f3-9ecfac834b5e	4	4	502
fecfb2bc-6cef-4084-b952-11c1ca339daa	9b40375d-30ff-4dd0-b8f3-9ecfac834b5e	4	4	502

(3 rows)

Fig 22. Vista top hotspot.

La vista creada para el hotspot devuelve por (game_id, map_id) la celda 250x250 con mas hits, en la validación muestra 3 juegos con el mismo pad_id y el mismo hotspot (4,4), lo que es posible dado que la actividad se concentra en esa zona del mapa y el índice soporta esa vista.

Esta vista sirve para detectar puntos de concentración para equilibrar coberturas sin hacer barridos manuales, como resultado aparece la misma celda repetida como top, es decir, es mantiene en esa zona.

V. Vista materializada: segmentos de trayectoria.

```
CREATE MATERIALIZED VIEW
doomdb2=# SELECT *
doomdb2=# FROM mv_trajectory_segments
doomdb2=# WHERE game_id = '5793d4a6-692a-4fa2-8629-588444f113cd'
doomdb2=# AND player_id = '60fc2e54-7379-40a6-bf5a-fdc4fd21db82'
doomdb2=# ORDER BY tic
doomdb2=# LIMIT 30;
```

game_id	player_id	tic	created_at	pos_x	pos_y	pos_z	step_distance	step_speed
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	237	2025-11-07 23:31:45	-96	784	56	0	0
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	272	2025-11-07 23:31:46	-89	882	36	180.26464980241042	2.864704280068869
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	307	2025-11-07 23:31:47	-86	1091	32	209.05980000573042	5.973137144494405
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	337	2025-11-08 08:36:58	-96	784	56	388.09901006007794	10.269967002002598
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	342	2025-11-07 23:31:48	-57	995	19	217.74867144197016	43.54813428839403
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	372	2025-11-08 08:36:59	-97	985	16	41.348053217188775	1.3780017739062924
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	377	2025-11-07 23:31:49	-64	1061	24	83.24061508662703	16.648123017325407
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	395	2025-11-08 13:47:40	448	752	128	606.9934101783972	33.721856121022064
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	395	2025-11-08 10:13:52	448	752	128	0	0
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	403	2025-11-08 06:49:45	-96	784	56	549.6762683616603	68.70953354520753
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	407	2025-11-08 08:37:00	-97	1064	24	281.82441342083905	70.45610335520976
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	412	2025-11-07 23:31:50	-66	1105	32	52.01922721455981	10.403845442911962
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	430	2025-11-08 13:47:41	495	747	128	672.3845625830504	37.354697921280575
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	430	2025-11-08 10:13:53	495	747	128	0	0
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	438	2025-11-08 06:49:46	-101	1015	16	663.0113121206907	82.87641401508634
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	442	2025-11-08 08:37:01	-100	1056	24	41.78516483155236	10.44629120788809
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	447	2025-11-07 23:31:51	-76	1370	56	316.53751752359466	63.30750350471893
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	465	2025-11-08 13:47:42	522	747	128	866.554672250978	48.141926236165446
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	465	2025-11-08 11:49:32	522	747	128	0	0
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	473	2025-11-08 06:49:47	-106	1086	24	721.1941406174163	90.14926857717704
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	477	2025-11-08 08:37:02	-97	1108	32	18.466185312619388	4.616506328154807
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	482	2025-11-07 23:31:52	-46	1385	56	290.5202230482415	58.1040446096408305
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	500	2025-11-08 11:49:33	697	720	128	999.72896326950454	55.540497959419184
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	501	2025-11-08 13:47:44	-122	721	56	822.159351950703	822.159351950703
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	508	2025-11-08 06:49:48	-86	1121	32	402.33319524990725	57.47617074998675
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	512	2025-11-08 08:37:03	-73	1371	56	251.485586060606873	62.87139651701718
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	517	2025-11-07 23:31:53	-48	1404	56	41.4060483080968905	8.280096617793701
5793d4a6-692a-4fa2-8629-588444f113cd	60fc2e54-7379-40a6-bf5a-fdc4fd21db82	535	2025-11-08 11:49:34	637	669	128	1007.29042408527333	55.96057915840518

(30 rows)

Fig 23. Comprobación de datos.

```

doomdb2=# SELECT COUNT(*)
doomdb2=# FROM mv_trajectory_segments
doomdb2=# WHERE game_id = '5793d4a6-692a-4fa2-8629-588444f113cd'
doomdb2=# AND player_id = '60fc2e54-7379-40a6-bf5a-fdc4fd21db82';
count
-----
10663
(1 row)

doomdb2=# -- (ya viste que da 10663)

```

Fig 24. Cardinalidad y evidencia que si hay datos.

```

Limit (cost=0.41..247.59 rows=1000 width=130) (actual time=0.057..0.263 rows=1000.00 loops=1)
  Buffers: shared hit=37
  -> Index Scan using idx_mv_game_player_tic on mv_trajectory_segments (cost=0.41..1757.11 rows=7197 width=130) (actual time=0.056..0.224 rows=1000.00 loops=1)
    Index Cond: (((game_id)::text = '5793d4a6-692a-4fa2-8629-588444f113cd'::text) AND ((player_id)::text = '60fc2e54-7379-40a6-bf5a-fdc4fd21db82'::text))
    Index Searches: 1
    Buffers: shared hit=37
  Planning Time: 0.140 ms
  Execution Time: 0.302 ms
(8 rows)

```

Fig 25. Rendimiento en Explain Analyze.

FIG. RENDIMIENTO EN EXPLAIN ANALYZE.

Muchas visualizaciones interactúan segmento a segmentos la línea de trayectoria, hetampa de velocidad por tic, recalcula LAG(), entonces materializar reduce latencia de consulta.

La vista materializada creada pre-calcula los segmentos entre tics consecutivos por (game_id, player_id) con su step_distance y step_speed.

Esta vista expone los segmentos ya calculados por (game_id, player_id, tic) con distancia por paso y speed, es necesario materializar para que los cálculos que usan ventanas y agregaciones se puedan tener persistidos y esto acelera las consultas de trayectoria y depuración.

El uso de la vista es filtrar por un jugador o juego y ordenar por tic para ver el recorrido de principio a fin.

Como se observa en las imágenes, la vista materializada mv_trajectory_segments almacena los pasos de trayectoria precomputados, con el índice (game_id, player_id_tic) las consultas filtradas por jugador y partida responden milisegundos.

- Creación de índices.

```

Time: 0.348 ms
doomdb=# CREATE INDEX IF NOT EXISTS idx_tevt_game_player_tic ON telemetry_event (game_id, player_id, tic);
NOTICE: relation "idx_tevt_game_player_tic" already exists, skipping
CREATE INDEX
Time: 1.297 ms
doomdb=# CREATE INDEX IF NOT EXISTS idx_tevt_game_map_tic ON telemetry_event (game_id, map_id, tic);
NOTICE: relation "idx_tevt_game_map_tic" already exists, skipping
CREATE INDEX
Time: 1.232 ms
doomdb=# CREATE INDEX IF NOT EXISTS idx_tevt_grid_250 ON telemetry_event (map_id, game_id, floor(pos_x/250.0), floor(pos_y/250.0));
NOTICE: relation "idx_tevt_grid_250" already exists, skipping
CREATE INDEX
Time: 1.542 ms
doomdb=# CREATE INDEX IF NOT EXISTS idx_mv_game_player_tic ON mv_trajectory_segments (game_id, player_id, tic);
NOTICE: relation "idx_mv_game_player_tic" already exists, skipping
CREATE INDEX
Time: 1.262 ms
doomdb=# ANALYZE telemetry_event;
ANALYZE
Time: 113.880 ms
doomdb=# ANALYZE mv_trajectory_segments;
ANALYZE
Time: 52.016 ms

```

Fig 26. creación de índices.

En cuestión al desempeño, los índices creados (por jugador/tiempo, por grid y por juego-map-tic) ayudan a que el motor elija planes de ejecución mas directos, las vistas dejan listas las métricas principales y la vista materializada permite revisar trayectorias sin recalcular todo cada vez.

Con la creación de los índices se tiene que son exactamente las columnas por las que se filtra y se orden en las consultas analíticas, esto permite que postgres salte a las filas que importan sin leer todo.

A nivel general se nota que los planes de ejecución empiezan a usar index Sca/Bitmap en vez de full scans y los tiempos se vuelven mas estables. Como conclusión general, con estos índices las consultas criticas quedan con una psita y escalan mejor y son más predecibles.

VI. CONCLUSIONES.

- En nivel general, el mapa es estable en duración, tiene un punto que concentra la actividad, permite estilos de juego distintos y mantiene una velocidad de movimiento constante.
- Se diseñaron los índices a la medida de cada consulta, aunque el dataset no es el mas grande, los tiempos son similares.
- Se realizaron vistas y una MV para convertir análisis en activos reutilizables y bajar la latencia de consumo.
- El proyecto permitió tener diferentes conocimientos de la asignatura, también se pudieron plantear diferentes formas de resolverlo y el poder jugar la extensión de DOOM lo hizo más interesante.