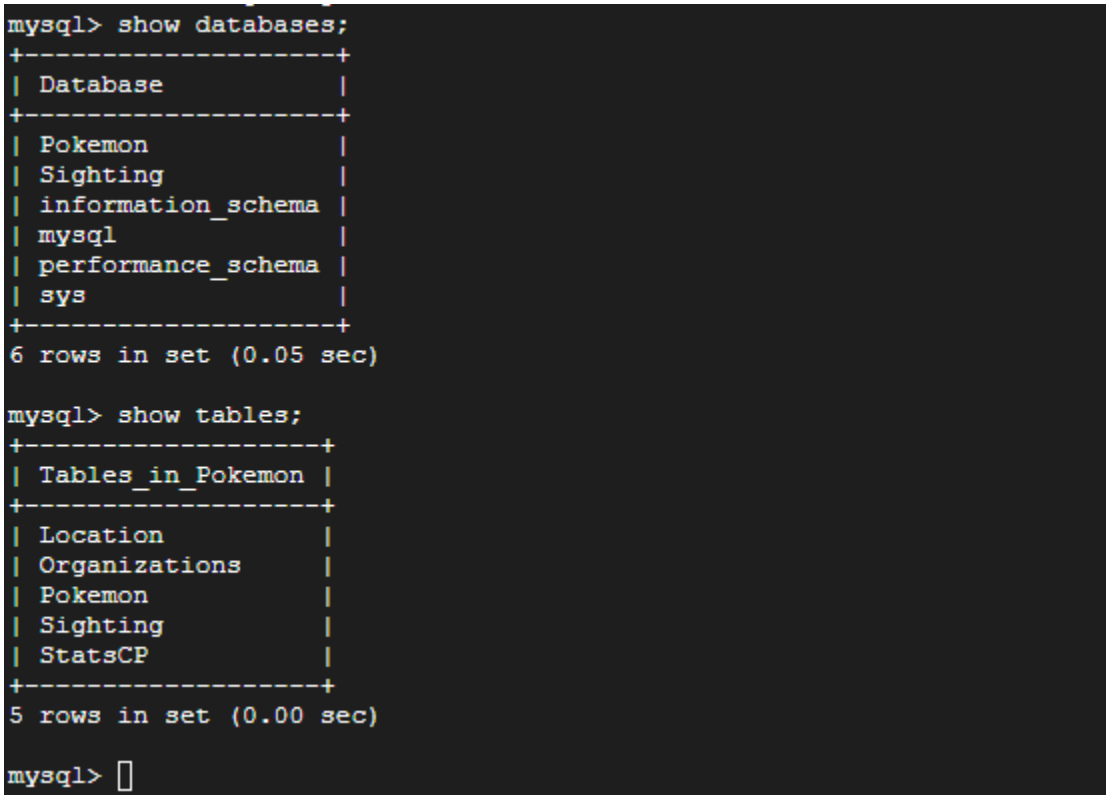
Part 1

Implementation of Database Table using GCP



DDL Commands

Tables:

```
CREATE TABLE Pokemon (
        pokemon_id INT PRIMARY KEY,
        pokemon_name VARCHAR(255),
        base_attack INT,
        base_defense INT,
        base_stamina INT,
        type VARCHAR(255),
        rarity VARCHAR(255),
);
```

```
CREATE TABLE StatsCP (
   base_attack INT,
   base_defense INT,
   base_stamina INT,
   max_cp INT,
   PRIMARY KEY (base_attack, base_defense, base_stamina)
);


CREATE TABLE Location (
        longitude DECIMAL(20, 18),
        latitude DECIMAL(15, 12),
        city VARCHAR(255),
        population_density DECIMAL(10,2),
        closeToWater BOOLEAN,
        PRIMARY KEY(longitude, latitude)
);

CREATE TABLE Sighting (
        sightingId VARCHAR(255),
        pokemon_id INT,
        longitude DECIMAL(20, 18),
        latitude DECIMAL(15, 12),
        appearedTimeOfDay VARCHAR(255),
        weather VARCHAR(255),
        temperature DECIMAL(4,2),
        windSpeed DECIMAL(4,2),
        PRIMARY KEY (sightingId),
        FOREIGN KEY (pokemon_id) REFERENCES Pokemon(pokemon_id) ON DELETE
CASCADE
);

CREATE TABLE User (
        userId INT PRIMARY KEY,
        password VARCHAR(255),
        role VARCHAR(255),
        organizationName VARCHAR(255),
        FOREIGN KEY organizationName  REFERENCES Organizations(organizationName)
ON DELETE CASCADE

);

CREATE TABLE Reports (
        reportId INT PRIMARY KEY,
```

```
        sightingId INT,
        userId INT,
        eventId INT,
        status VARCHAR(255),
        notes TEXT,
        time DATETIME,
        FOREIGN KEY sightingId REFERENCES Sighting(sightingId) ON DELETE CASCADE,
        FOREIGN KEY userId REFERENCES User(userId) ON DELETE CASCADE,
        FOREIGN KEY eventId REFERENCES Event(eventId) ON DELETE CASCADE

);

CREATE TABLE Events (
        eventId INT PRIMARY KEY,
        eventName VARCHAR(255),
        description TEXT,
        location VARCHAR(255),
        time DATETIME,
        participantCount INT,
        organizationName VARCHAR(255),
        FOREIGN KEY organizationName REFERENCES Organizations(organizationName)
ON DELETE CASCADE
);

CREATE TABLE Organizations (
        organizationName VARCHAR(255) PRIMARY KEY
);
```

1000 rows in three tables

```
5 rows in set (0.00 sec)

mysql> SELECT COUNT(*) FROM Location;
+----------+
| COUNT(*) |
+----------+
|   143919 |
+----------+
1 row in set (0.75 sec)

mysql> SELECT COUNT(*) FROM Pokemon
    -> ;
+----------+
| COUNT(*) |
+----------+
|     1007 |
+----------+
1 row in set (0.11 sec)

mysql> SELECT COUNT(*) FROM Sighting;
+----------+
| COUNT(*) |
+----------+
|   296021 |
+----------+
1 row in set (2.15 sec)

mysql>
```

Generalized/Plan SQL Queries:

1. IF 1 Filter SPECIFIED;
User defines X filter,

SELECT Pokemon, Longitude, Latitude
FROM Pokemon Natural Join StatsCP, Natural JOIN Sighting Natural Join Location
GROUP BY Pokemon.type
HAVING Distance >= (LongitudeUSER, LatitudeUSER) - (LongitudeSighting, LatitudeSighting)

2. IF X(i) Stats filters SPECIFIED;
User defines Xi filters,Y Distance
SELECT Pokemon, Longitude, Latitude
FROM Pokemon Natural Join StatsCP Natural JOIN Sighting Natural Join Location

GROUP BY X1,X2…Xi
HAVING Distance >= (LongitudeUSER, LatitudeUSER) - (LongitudeSighting, LatitudeSighting)
AND Xis in ranges of filters

3. IF X(i) Sighting filters SPECIFIED;
User defines Xi filters,Y Distance
SELECT Pokemon, Longitude, Latitude
FROM Pokemon Natural Join StatsCP Natural JOIN Sighting Natural Join Location
GROUP BY X1,X2…Xi
HAVING Distance >= (LongitudeUSER, LatitudeUSER) - (LongitudeSighting, LatitudeSighting)
AND Xis in ranges of filters

4. IF Pokemon SPECIFIED;
User defines X as pokemon name,

```
-- params: :X, :latitudeUSER, :longitudeUSER, :DistanceUSER (km)
SELECT p.pokemon_name    AS Pokemon, l.longitude, l.latitude,
  2 * 6371 * ASIN(
      SQRT(
        POWER(SIN(RADIANS(l.latitude  - :latitudeUSER)  / 2), 2) +
        COS(RADIANS(:latitudeUSER)) * COS(RADIANS(l.latitude)) *
        POWER(SIN(RADIANS(l.longitude - :longitudeUSER) / 2), 2)
      )
  ) AS actual
FROM Pokemon p NATURAL JOIN StatsCP NATURAL JOIN Sighting s NATURAL JOIN
Location l
WHERE p.pokemon_name = :X AND  actual < :DistanceUSER
ORDER BY actual;
```

Specific Queries + 15 Results

First Query: finds number of each water type pokemon's sightings within 100 miles of Urbana-Champaign

```
SELECT Pokemon.pokemon_name, COUNT(Pokemon.pokemon_name)
FROM Pokemon NATURAL JOIN Sighting
WHERE Pokemon.type = 'Water'
AND 100 >=
(SELECT ST_Distance_Sphere(
   POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
```

POINT(Sighting.longitude, Sighting.latitude) -- Sighting (lon, lat)
) / 1609.34)
GROUP BY Pokemon.pokemon_name;

```
mysql> SELECT Pokemon.pokemon_name, COUNT(Pokemon.pokemon_name)
    -> FROM Pokemon NATURAL JOIN Sighting
    -> WHERE Pokemon.type = 'Water'
    -> AND 100 >=
    -> (SELECT ST_Distance_Sphere(
    ->     POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
    ->     POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
    -> ) / 1609.34)
    -> GROUP BY Pokemon.pokemon_name
    -> ;
+--------------+-----------------------------+
| pokemon_name | COUNT(Pokemon.pokemon_name) |
+--------------+-----------------------------+
| Krabby       |                           3 |
| Seel         |                           1 |
| Goldeen      |                           2 |
| Horsea       |                           1 |
| Poliwag      |                           1 |
+--------------+-----------------------------+
5 rows in set (9.51 sec)
```

Less than 15 rows in output

Second Query: finds average distance of each water type pokemon's sightings within 100 miles of Urbana-Champaign

SELECT Pokemon.pokemon_name, AVG(ST_Distance_Sphere(
POINT(-88.2073, 40.1106), -- Urbana-Champaign (lon, lat)
POINT(Sighting.longitude, Sighting.latitude) -- Sighting (lon, lat)
) / 1609.34)
FROM Pokemon NATURAL JOIN Sighting
WHERE Pokemon.type = 'Water'
AND 100 >= (SELECT ST_Distance_Sphere(
POINT(-88.2073, 40.1106), -- Urbana-Champaign (lon, lat)
POINT(Sighting.longitude, Sighting.latitude) -- Sighting (lon, lat)
) / 1609.34)
GROUP BY Pokemon.pokemon_name;

```
mysql> SELECT Pokemon.pokemon_name, AVG(ST_Distance_Sphere(
    ->     POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
    ->     POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
    -> ) / 1609.34)
    -> FROM Pokemon NATURAL JOIN Sighting
    -> WHERE Pokemon.type = 'Water'
    -> AND 100 >= (SELECT ST_Distance_Sphere(
    ->     POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
    ->     POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
    -> ) / 1609.34)
    -> GROUP BY Pokemon.pokemon_name
    -> ;
+--------------+-----------------------------------------------------------------------------------------------------------------------------------+
| pokemon_name | AVG(ST_Distance_Sphere(
    POINT(-88.2073, 40.1106),
    POINT(Sighting.longitude, Sighting.latitude)
) / 1609.34) |
+--------------+-----------------------------------------------------------------------------------------------------------------------------------+
| Krabby       |                                                                                                  98.71722803754314 |
| Seel         |                                                                                                  98.73747065903535 |
| Goldeen      |                                                                                                  87.33306333227473 |
| Horsea       |                                                                                                  98.71891370230698 |
| Poliwag      |                                                                                                  87.29944236020435 |
+--------------+-----------------------------------------------------------------------------------------------------------------------------------+
5 rows in set (9.11 sec)
```

Less than 15 rows in output

Third Query: finds number of pokemon sightings with the constraint of 1000 <= max_cp <= 2000 and within 100 miles of Urbana-Champaign

```
SELECT Pokemon.pokemon_name, MAX(StatsCP.max_cp) as max_cp,
COUNT(Pokemon.pokemon_name)
FROM Pokemon
JOIN StatsCP
ON Pokemon.base_attack = StatsCP.base_attack AND Pokemon.base_defense =
StatsCP.base_defense AND Pokemon.base_stamina = StatsCP.base_stamina
NATURAL JOIN Sighting
WHERE StatsCP.max_cp >= 1000 AND StatsCP.max_cp <= 2000
AND 100 >= (SELECT ST_Distance_Sphere(
   POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
   POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
) / 1609.34)
GROUP BY Pokemon.pokemon_name;
```

```
mysql> SELECT Pokemon.pokemon_name, MAX(StatsCP.max_cp) as max_cp, COUNT(Pokemon.pokemon_name)
    -> FROM Pokemon
    -> JOIN StatsCP
    -> ON Pokemon.base_attack = StatsCP.base_attack AND Pokemon.base_defense = StatsCP.base_defense AND Pokemon.base_stamina = StatsCP.base_stamina
    -> NATURAL JOIN Sighting
    -> WHERE StatsCP.max_cp >= 1000 AND StatsCP.max_cp <= 2000
    -> AND 100 >= (SELECT ST_Distance_Sphere(
    ->     POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
    ->     POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
    -> ) / 1609.34)
    -> GROUP BY Pokemon.pokemon_name;
+--------------+--------+---------------------------+
| pokemon_name | max_cp | COUNT(Pokemon.pokemon_name) |
+--------------+--------+---------------------------+
| Pidgeotto    |   1366 |                         3 |
| Raticate     |   1950 |                         3 |
| Oddish       |   1405 |                         2 |
| Paras        |   1164 |                         3 |
| Venonat      |   1148 |                         9 |
| Bellsprout   |   1182 |                         3 |
| Seel         |   1111 |                         1 |
| Gastly       |   1406 |                         1 |
| Drowzee      |   1190 |                        10 |
| Krabby       |   1785 |                         3 |
| Horsea       |   1208 |                         1 |
| Goldeen      |   1318 |                         2 |
| Eevee        |   1225 |                        13 |
| Dratini      |   1149 |                         1 |
+--------------+--------+---------------------------+
14 rows in set (1 min 14.04 sec)
```

Less than 15 rows in output

Fourth Query: finding pokemon that have been sighted within 100 miles of Urbana-Champaign and within 100 miles of San Francisco

```
((SELECT Pokemon.pokemon_name
FROM Pokemon NATURAL JOIN Sighting
WHERE 100 >= (SELECT ST_Distance_Sphere(
    POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
    POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
) / 1609.34))
INTERSECT
(SELECT Pokemon.pokemon_name
FROM Pokemon NATURAL JOIN Sighting
WHERE 100 >= (SELECT ST_Distance_Sphere(
    POINT(-99.1332, 19.4326),  -- Mexico City (lon, lat)
    POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
) / 1609.34)))
LIMIT 15;
```

```
mysql> ((SELECT Pokemon.pokemon_name
    -> FROM Pokemon NATURAL JOIN Sighting
    -> WHERE 100 >= (SELECT ST_Distance_Sphere(
    ->     POINT(-88.2073, 40.1106),  -- Urbana-Champaign (lon, lat)
    ->     POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
    -> ) / 1609.34))
    -> INTERSECT
    -> (SELECT Pokemon.pokemon_name
    -> FROM Pokemon NATURAL JOIN Sighting
    -> WHERE 100 >= (SELECT ST_Distance_Sphere(
    ->     POINT(-99.1332, 19.4326),  -- Mexico City (lon, lat)
    ->     POINT(Sighting.longitude, Sighting.latitude)  -- Sighting (lon, lat)
    -> ) / 1609.34)))
    -> LIMIT 15
    -> ;
+--------------+
| pokemon_name |
+--------------+
| Eevee        |
| Weedle       |
| Pidgey       |
| Oddish       |
| Rattata      |
| Venonat      |
| Dratini      |
| Drowzee      |
| Raticate     |
| Bellsprout   |
| Krabby       |
| Metapod      |
| Pidgeotto    |
| Spearow      |
| Paras        |
+--------------+
15 rows in set (14.18 sec)
```

Limited to 15 rows in output

Part 2. Indexing

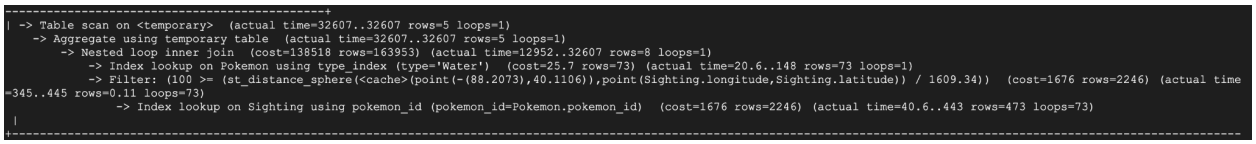
Query 1

```
| -> Table scan on <temporary>  (actual time=5276..5276 rows=5 loops=1)
    -> Aggregate using temporary table  (actual time=5276..5276 rows=5 loops=1)
        -> Nested loop inner join  (cost=319315 rows=29549) (actual time=1175..5275 rows=8 loops=1)
            -> Filter: ((100 >= (st_distance_sphere(<cache>(point(-(88.2073),40.1106)),point(Sighting.longitude,Sighting.latitude)) / 1609.34)) and (Sighting.pokemon_id is not nu
ll))  (cost=31213 rows=295489) (actual time=218..5255 rows=170 loops=1)
                -> Table scan on Sighting  (cost=31213 rows=295489) (actual time=0.22..4506 rows=296021 loops=1)
            -> Filter: (Pokemon.`type` = 'Water')  (cost=0.875 rows=0.1) (actual time=0.12..0.12 rows=0.0471 loops=170)
                -> Single-row index lookup on Pokemon using PRIMARY (pokemon_id=Sighting.pokemon_id)  (cost=0.875 rows=1) (actual time=0.113..0.113 rows=1 loops=170)
 |
```

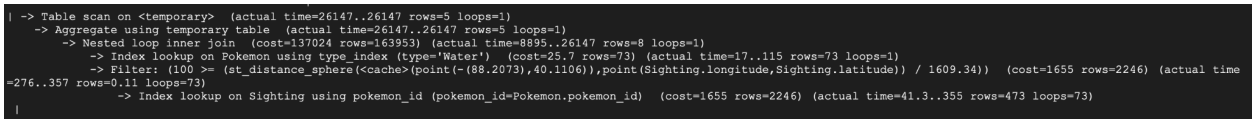
Original

```
| -> Table scan on <temporary>  (actual time=7166..7166 rows=5 loops=1)
    -> Aggregate using temporary table  (actual time=7166..7166 rows=5 loops=1)
        -> Nested loop inner join  (cost=245765 rows=29549) (actual time=1481..7166 rows=8 loops=1)
            -> Filter: ((100 >= (st_distance_sphere(<cache>(point(-(88.2073),40.1106)),point(Sighting.longitude,Sighting.latitude)) / 1609.34)) and (Sighting.pokemon_id is not nu
ll))  (cost=31536 rows=295489) (actual time=474..7161 rows=170 loops=1)
                -> Table scan on Sighting  (cost=31536 rows=295489) (actual time=94.9..6432 rows=296021 loops=1)
            -> Filter: (Pokemon.`type` = 'Water')  (cost=0.625 rows=0.1) (actual time=0.0268..0.0268 rows=0.0471 loops=170)
                -> Single-row index lookup on Pokemon using PRIMARY (pokemon_id=Sighting.pokemon_id)  (cost=0.625 rows=1) (actual time=0.0202..0.0203 rows=1 loops=170)
 |
```

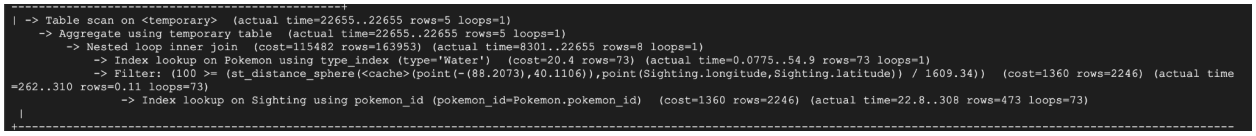
Type Index



Type, maxCP Indexes

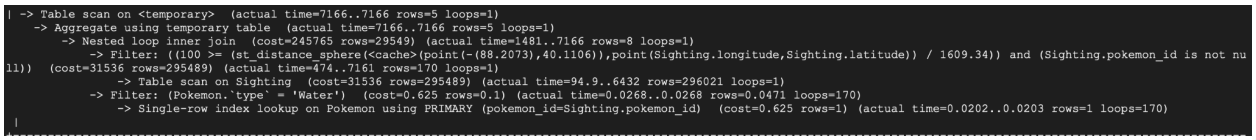


Type, maxcp, base_attack Indexes

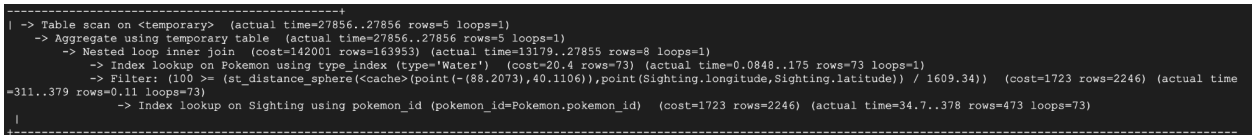


Type, maxcp, base_attack, base_defense Indexes

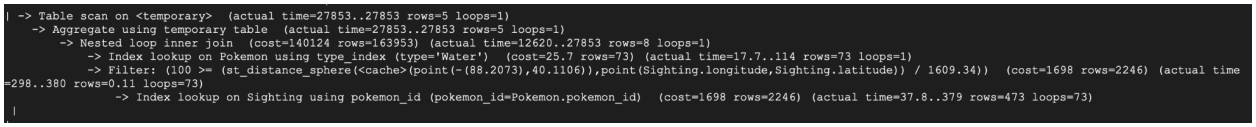
Query 2



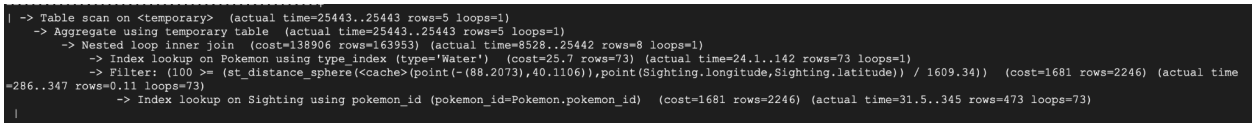
Original



Type, maxcp Indexes

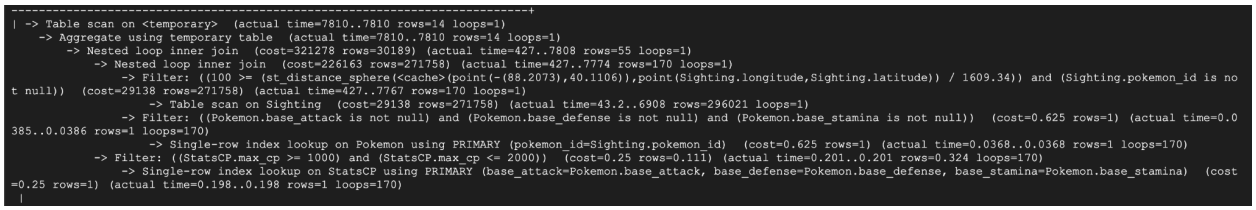


Type, maxcp, base_attack Indexes

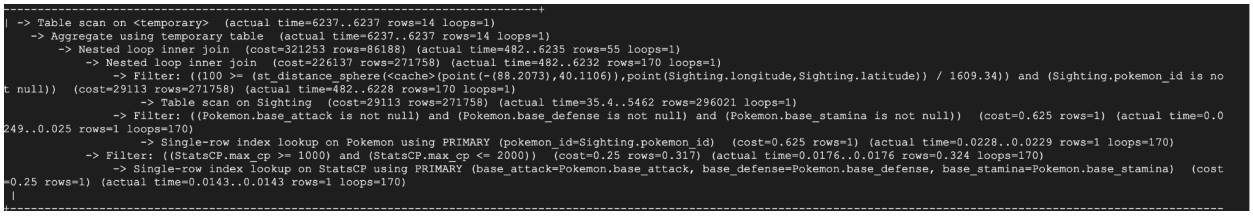


Type, maxcp, base_attack, base_defense Indexes

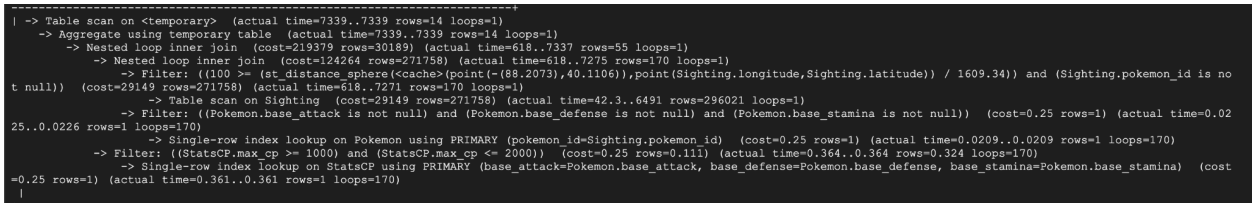
Query 3



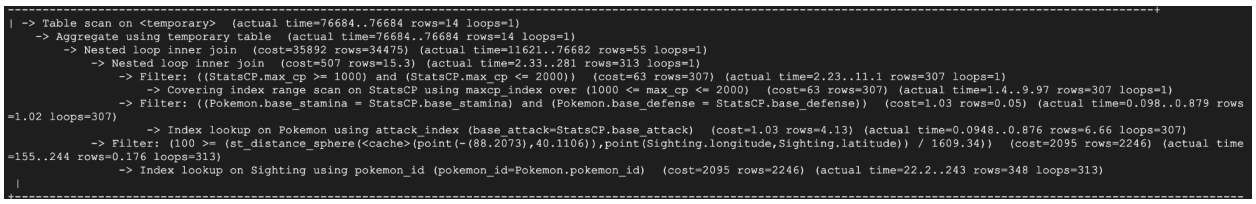
Original



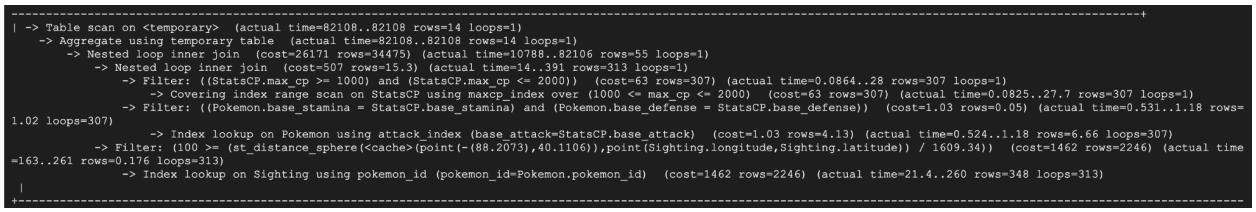
Type, maxcp Indexes



Type index

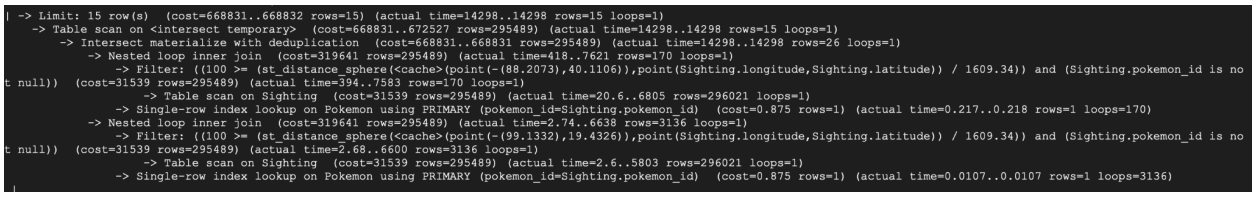


Type, maxcp, base_attack Indexes

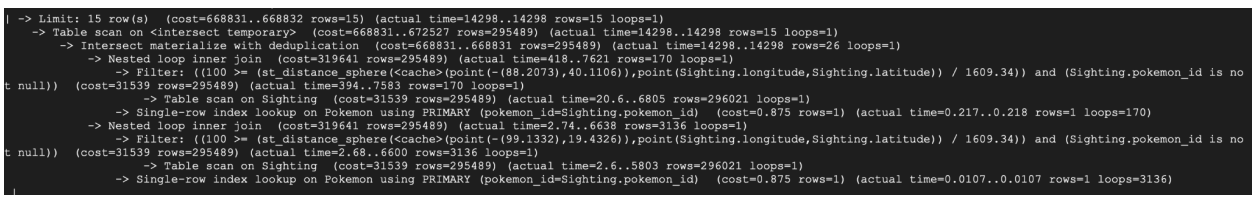


Type, maxcp, base_attack, base_defense Indexes

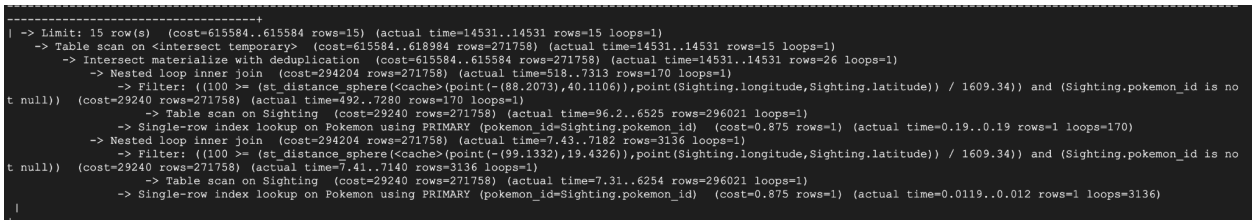
Query 4



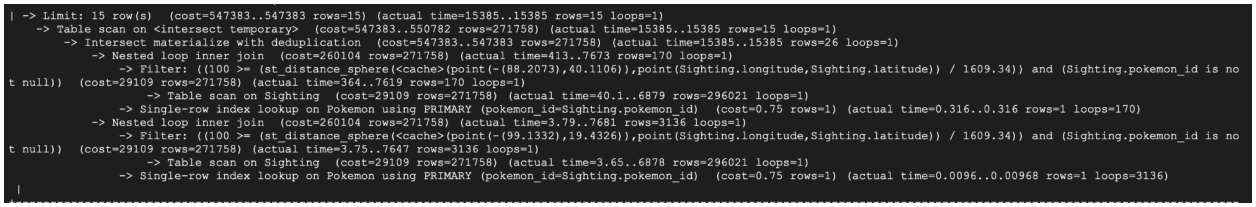
Original



Type index



Type, maxcp indexes



Type, maxcp, base_attack, base_defense Indexes

Final Index Design Report

After analyzing the queries, we implemented indexes on the following attributes:

1. Pokemon.type
    a. CREATE INDEX type_index ON Pokemon(type)
2. StatsCP.max_cp
    a. CREATE INDEX maxcp_index ON StatsCP(max_cp)
3. Pokemon.base_attack
    a. CREATE INDEX attack_index ON Pokemon(base_attack)
4. Pokemon.base_defense
    a. CREATE INDEX defense_index ON Pokemon(base_defense)

Rationale and Query Analysis

1. Pokemon.type – This index directly benefits Query #1, Query #4 and Query #2, which filter Pokémon by type (WHERE Pokemon.type = 'Water'). By indexing this column, the database can quickly locate only the relevant Pokémon instead of scanning the entire table.

2. StatsCP.max_cp – This index is used in Query #3, which filters Pokémon based on max_cp (WHERE StatsCP.max_cp >= 1000 AND StatsCP.max_cp <= 2000). Indexing max_cp allows the database to efficiently find rows within the specified range without scanning all entries in StatsCP. Although it was only specified/joined in Query #3, it did improve the costs of Query 1-4 by a slight amount, due to StatsCP sharing most of its attributes with the Pokemon table, improving the cache/Pokemon lookups.

3. Pokemon.base_attack and Pokemon.base_defense – These attributes are used to join Pokemon with StatsCP in Query #3. Indexing them improves the join performance by allowing the database to quickly match Pokémon with their corresponding stats. Although it was only specified in Query #3, it did improve the costs of Query 1-4 by 10 fold, due to the Pokemon table being used in every single query.

Observations

After applying these indexes, the performance improvement was very good for all queries, particularly because Queries 1,2,3,4 involves the indexed attributes in join and where clauses. The implemented indexes helped with filtering and joins, but the main computational cost remains calculating distances for each sighting row.