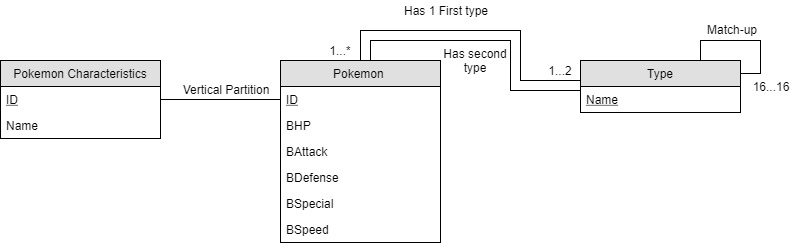
**Database Final Deliverable**

Cameron Rutherford, Ben Morledge-Hampton, Tyler Stitt, Tom Dale

12.14.2017





**Queries (both SQL and English):**

**1)**

SELECT Temp.ID ,PC.Name, Temp.HP, Temp.Attack, Temp.Defense, Temp.Speed, Temp.Special, Temp.Type1, Temp.Type2, Temp.Mul, Temp.SumStats

FROM (

SELECT PS.\*, GTM.Mul, (PS.HP + PS.Attack + PS.Defense + PS.Speed + PS.Special) AS SumStats, ROW\_NUMBER() over (partition by PS.Type1, PS.Type2 ORDER BY (PS.HP + PS.Attack + PS.Defense + PS.Speed + PS.Special) DESC) as rn

FROM PokemonStats PS, [dbo].[goodTypesInto] (46) AS GTM

WHERE PS.Type1 = GTM.A1 AND PS.Type2 = GTM.A2

) AS Temp INNER JOIN PokeCharacteristics AS PC ON PC.ID = Temp.ID

WHERE Temp.rn = 1

ORDER BY Mul DESC, Temp.SumStats DESC

**1 in English)**

Give me all the details of the best Pokémon of each type (determined by raw stat sum) who have a good type matchup against a certain Pokémon. Order it by the best type matchups, and then by the Pokémon’s stats.

**2)**

SELECT PC.\*

FROM PokeCharacteristics AS PC

WHERE PC.Name LIKE '%pok%’

**2 in English)**

Give all Pokémon characteristics from a single Pokémon that has a similar name to the one given by user.

3)

CREATE FUNCTION [goodTypesInto]

(

@AttackingPoke int

)

RETURNS TABLE

RETURN

SELECT DISTINCT M1.AttackTypeID AS A1, M2.AttackTypeID AS A2, ( (M2.Multiplier \* M4.Multiplier) + (M3.Multiplier\*M1.Multiplier) ) AS Mul

FROM PokemonStats PS, TypeMatchups M1, TypeMatchups M2, TypeMatchups M3, TypeMatchups M4

WHERE PS.ID = @AttackingPoke AND M1.TargetTypeID = PS.Type1 AND M2.TargetTypeID = PS.Type1 AND M3.TargetTypeID =

PS.Type2 AND M4.TargetTypeID = PS.Type2 AND M1.AttackTypeID = M3.AttackTypeID AND M2.AttackTypeID = M4.AttackTypeID AND

( (M2.Multiplier \* M4.Multiplier) + (M3.Multiplier\*M1.Multiplier) ) >= 3 AND M1.AttackTypeID <> M2.AttackTypeID

2

GO

**3 English)**

Function called by other queries that calculates an overall multiplier for each Pokémon when fighting against a user given Pokémon. Only Pokémon with multiplier 3 or higher are returned.

**Database Design**

Primary keys were created just fine, but unfortunately, we had issues enforcing some foreign keys. If two foreign keys in the same table referred to the same primary key in a different table, an error was given which prevented saving the changes. Furthermore, if a foreign key was created which was already itself a primary key, SQL server manager did not allow that either. We unfortunately do not have the time to investigate solutions to these problems, but seeing as the database works with our queries as is, we believe it is ok. The relationships which were not enforced are listed below:

Foreign Key Primary Key

PokemonStats.Type2 TypeNames.ID

TypeMatchups.AttackTypeID TypeNames.ID

TypeMatchups.TargetTypeID TypeNames.ID