

Datamatiker 2. Semester 2024

Mads Dalsgaard

Fuglevænget 76a Grenaa 8500 - [Dsadfgtff@gmail.com](mailto:Dsadfgtff@gmail.com)

31/05/2024

## Forside



<b>Forside.....</b>	<b>1</b>
<b>Introduktion.....</b>	<b>3</b>
Beskrivelse af Raid logging & forklarelse af Augmentation.....	3
Prescience Buffen og Wowanalyser.....	4
<b>ER-Diagram.....</b>	<b>4</b>
<b>Overgang fra ER til relationel Model.....</b>	<b>5</b>
<b>Implementering.....</b>	<b>6</b>
<b>Begrebsliste:.....</b>	<b>7</b>
<b>Bilag 1.....</b>	<b>9</b>
<b>Bilag 2.....</b>	<b>10</b>
<b>Bilag 3.....</b>	<b>10</b>
<b>Bilag 4.....</b>	<b>11</b>
<b>Bilag 5.....</b>	<b>11</b>

## Introduktion

I denne opgave fokuserer jeg på dataen der indsamles når der bliver “raid logget” i MMORPG spillet World of Warcraft. Jeg har valgt at vælge dette emne, grundet at jeg er blandt nogle af de bedste i verden til at raide, og jeg elsker at kigge igennem den data vi indsamler når vi spiller, som giver mig og mine teammates en hjælpende hånd når vi raider. Med det sagt, synes jeg også det kunne være interessant at om jeg kunne bruge min viden om den dataindsamling jeg laver, og implementere det i et konsol program.

## Beskrivelse af Raid logging & forklarelse af Augmentation

En *raid* er et Player versus Environment sted som omhandler 20-40 spillere (Tanks, Healers og DPS') som kæmper én boss. Når der *raid logges* gemmes alt den data der sker under en fight: hvem døde, hvilken skade der blev gjort, hvilken skade der blevet taget osv<sup>1</sup>. Siden patch 10.1.5 blev en ny DPS class tilføjet (augmentation) som ikke gør skade som resten af DPS' classes, dens skade kommer igennem at give *buffs* til andre. Så hvis man skal kunne gøre god skade i en boss encounter, så skal man vide hvem man skal buffe. Man skulle tro man bare ville kunne kigge på dem der laver højst skade på én encounter og så det bare dem man skal give buffs til, men damage dataen kan altid svinge fra de første 30 sekunder og fra 01:00 inde i en encounter til bossen er død<sup>2</sup>.

---

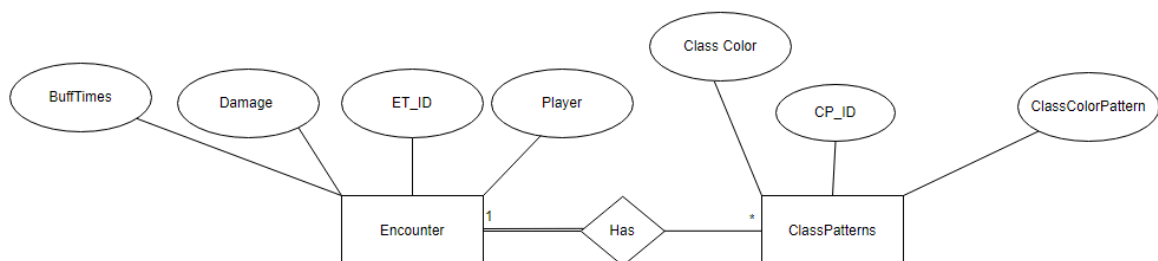
<sup>1</sup> Bilag 1 - Oversigt over skaden på en boss encounter.

<sup>2</sup> Bilag 2 - Første 30 sekunder versus 1 minut og fremadrettet

## Prescience Buffen og Wowanalyzer

Augmentation buffer personer med den “ability” som hedder Prescience<sup>3</sup>. Denne ability har 2 charges og 11 sekunder cooldown. Hver buffede person har den i 24 sekunder, ud over hver tredje person har den i 48 sekunder. Med den information og et raid log link, kan jeg igennem siden Wow Analyzer generere en tabel<sup>4</sup>, som viser hvem der gør mest skade under mine prescience intervaller og som burde blive buffed, som vil give en augmentation det maximale damage output, på en boss encounter på 1 minut og 30 sekunder.

## ER-Diagram



Et ER-Diagram er byggeplanen for den relationelle model og formålet med et ER-diagram er at vise forholdet mellem entiteter og for oprette de forskellige attributter. I mit ER diagram viser jeg de to entiteter som har hver deres attributter, og så den relation (has) som forbinder mine to entiteter. Fremgangsmåden jeg bruger, har været at angive de to entiteter og derefter give dem hver deres attributter og derefter skabe relationen mellem dem. Til sidst har jeg uddelt de to primære nøgler til hver entity, som identificerer en unik række i tabellen.

---

<sup>3</sup> Bilag 3 - Prescience

<sup>4</sup> Bilag 4 - Tabel genereret fra Wowanalyzer der viser buffing targets på en 1 minut og 30 sekunder Encounter

## Overgang fra ER til relationel Model

Formålet med en relationel model er at beskrive og strukturere data i en relationel database. Det er et mere konkret, brugervenligt og implementeringsspecifikt design. Når man skal få skabt en relationel model ud fra sit ER-Diagram, er der nogle regler man skal vide/følge f.eks. Flerværdiede attributter bliver til en tabel. Grundet af i mit ER diagram vi kun har at gøre med følgende:

Attributter

Entiteter - (Kun stærke ingen svage)

Relation

Så derfor kan jeg godt overse den regel som kun indebærer flerværdiede attributter. En regel vi benytter os af er, at en entitet bliver lavet til en tabel. Samt så bliver vores attributter til kolonner i deres entitets tabel. På vores relationelle model kan vi se vores attributter og entiteter samt relationen sat helt fint op. En relationel model består overordnet af 4 forskellige ting: attributter, tupler, relations grader og relationens tilstand. Dette har jeg prøvet at vise på Bilag 5<sup>5</sup>. Attributterne er vores forskellige kolonner, tuplerne viser de rækker der er, relationsgrad er antallet af kolonner, og relationensgrad er den data der er- eller bliver indsat i rækkerne.

---

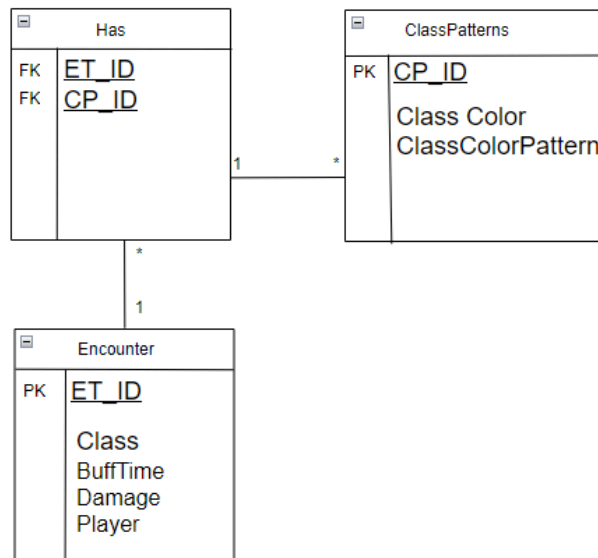
<sup>5</sup> Bilag 5 - Tabel over encounter entiteten

Datamatiker 2. Semester 2024

Mads Dalsgaard

Fuglevænget 76a Grenaa 8500 - [Dsadfgttf@gmail.com](mailto:Dsadfgttf@gmail.com)

31/05/2024



## Implementering

Dette er min C# kode, som både implementerer data ind i en database og opretter tabellen, samt det printer ud i konsollen de bedste targets til at buff samt tidspunkter, deres class og den respektive classcolor + dens specifikke MRT pattern. Det første vi gør er at vi connecter til databasen igennem en connectionstring, dette script kan køres på enhver computer så længe man indtaster selve sin databases navn i dette eksempel oprettede jeg bare databasen "zs". Herefter instantieres koden som opretter stringen "createTable" og denne linje skaber også vores tables i PGAdmin. Efter dette har vi alt den data som lægges ind i tabellen (relationens data). Nede i min "if" statement skaber vi reader.read som udskriver alt det vi har indtastet til vores lokale variable (output). Mine sidste to funktioner er dem der repræsenterer dataen og gør vi kan skabe en connection til vores output vha. datatyper som int eller string.

Datamatiker 2. Semester 2024

Mads Dalsgaard

Fuglevænget 76a Grenaa 8500 - [Dsadfgttf@gmail.com](mailto:Dsadfgttf@gmail.com)

31/05/2024

```
static void Main(string[] args)
{
    string connectionString = "Host=localhost;Username=postgres;Password=12345678;Database=zs";
    NpgsqlDataSource dataSource = NpgsqlDataSource.Create(connectionString);
    string createTable = "CREATE TABLE IF NOT EXISTS encounter (PT_ID integer NOT NULL GENERATED ALWAYS AS IDENTITY(INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1), class character varying(50), bufftime integer, damage integer);";

    NpgsqlCommand cmd = dataSource.CreateCommand(createTable);

    cmd.ExecuteNonQuery();

    createTable = "CREATE TABLE IF NOT EXISTS classpatterns (CP_ID integer NOT NULL GENERATED ALWAYS AS IDENTITY(INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1), classcolor character varying(50), classcolorpattern character varying(50));";

    cmd = dataSource.CreateCommand(createTable);

    cmd.ExecuteNonQuery();

    Add2("pistachio", "|cfaad372|r", connectionString);
    Add2("pistachio", "|cfaad372|r", connectionString);
    Add2("Purple", "|cfa330c9A|r ", connectionString);
    Add2("light purple", "|cff8788ee|r", connectionString);
    Add2("brown", "|cfa69b6d|r ", connectionString);
    Add2("blue", "|cff9970d|r ", connectionString);
    Add2("yellow", "|cffff468|r ", connectionString);
    Add2("white", "|cff9970d|r", connectionString);
    Add2("light purple", "|cff8788ee|r", connectionString);
    Add2("blue", "|cff3fc7eb|r ", connectionString);

    Add("hunter", "Penkek", 24450000, 0, connectionString);
    Add("hunter", "Sprikey", 26770000, 0, connectionString);
    Add("Demonhunter", "Maevbean", 14220000, 11, connectionString);
    Add("warlock", "Echazarlock", 19660000, 22, connectionString);
    Add("Warrior", "Sponngiewarr", 18980000, 33, connectionString);
    Add("Shaman", "Wäter", 24450000, 44, connectionString);
    Add("Rogue", "imtilt", 4220000, 55, connectionString);
    Add("Priest", "Weems", 9590000, 66, connectionString);
    Add("warlock", "Echazarlock", 4220000, 77, connectionString);
    Add("Rage", "Doffusmagus", 3470000, 88, connectionString);

    Console.WriteLine("Would you like to see the data of buffing targets? (Yes or No)");
}
```

```
if(Console.ReadLine()=="Yes")
{
    NpgsqlCommand selectCmd = dataSource.CreateCommand("SELECT * FROM encounter");
    using (NpgsqlDataReader reader = selectCmd.ExecuteReader())
    {
        while (reader.Read())
        {
            string output = reader.GetInt32(0).ToString() + " " + reader.GetString(1) + " " + reader.GetInt32(2).ToString() + " " + reader.GetInt32(3).ToString() + " " + reader.GetString(4);
            Console.WriteLine(output);
        }
    }

    selectCmd = dataSource.CreateCommand("SELECT * FROM classpatterns");
    using (NpgsqlDataReader reader = selectCmd.ExecuteReader())
    {
        while (reader.Read())
        {
            string output = reader.GetInt32(0).ToString() + " " + reader.GetString(1) + " " + reader.GetString(2);
            Console.WriteLine(output);
        }
    }

    Console.ReadLine();
}
else
{
    //Lukker konsollen
}

10 references
static void Add(string playerClass, string name, int dps, int time, string connectionString)
{
    NpgsqlDataSource dataSource = NpgsqlDataSource.Create(connectionString);
    NpgsqlCommand cmd = dataSource.CreateCommand("INSERT INTO encounter (class, players, damage, bufftime) VALUES ('" + playerClass + "', '" + name + "', '" + dps + "', '" + time + "')");

    cmd.ExecuteNonQuery();
}

10 references
static void Add2(string classcolor, string classcolorpattern, string connectionString)
{
    NpgsqlDataSource dataSource = NpgsqlDataSource.Create(connectionString);
    NpgsqlCommand cmd = dataSource.CreateCommand("INSERT INTO classpatterns (classcolor, classcolorpattern) VALUES ('" + classcolor + "', '" + classcolorpattern + "')");

    cmd.ExecuteNonQuery();
}
```

## Begrebsliste:

raid: En gruppe af folk som arbejder sammen for at besejre bosser på forskellige sværhedsgrader, en raid består typisk af 9-12 bosser og raid gruppen kan have en størrelse på 20-40 mennesker. (højeste sværhedsgrad er det maximum 20).

raidlog: En raidlog indeholder alt det data som gemmes inden for en boss encounter, alt fra hvem der dør, hvem der gør mest skade, hvem der tager mest skade, hvilket abilites folk har som skader osv.



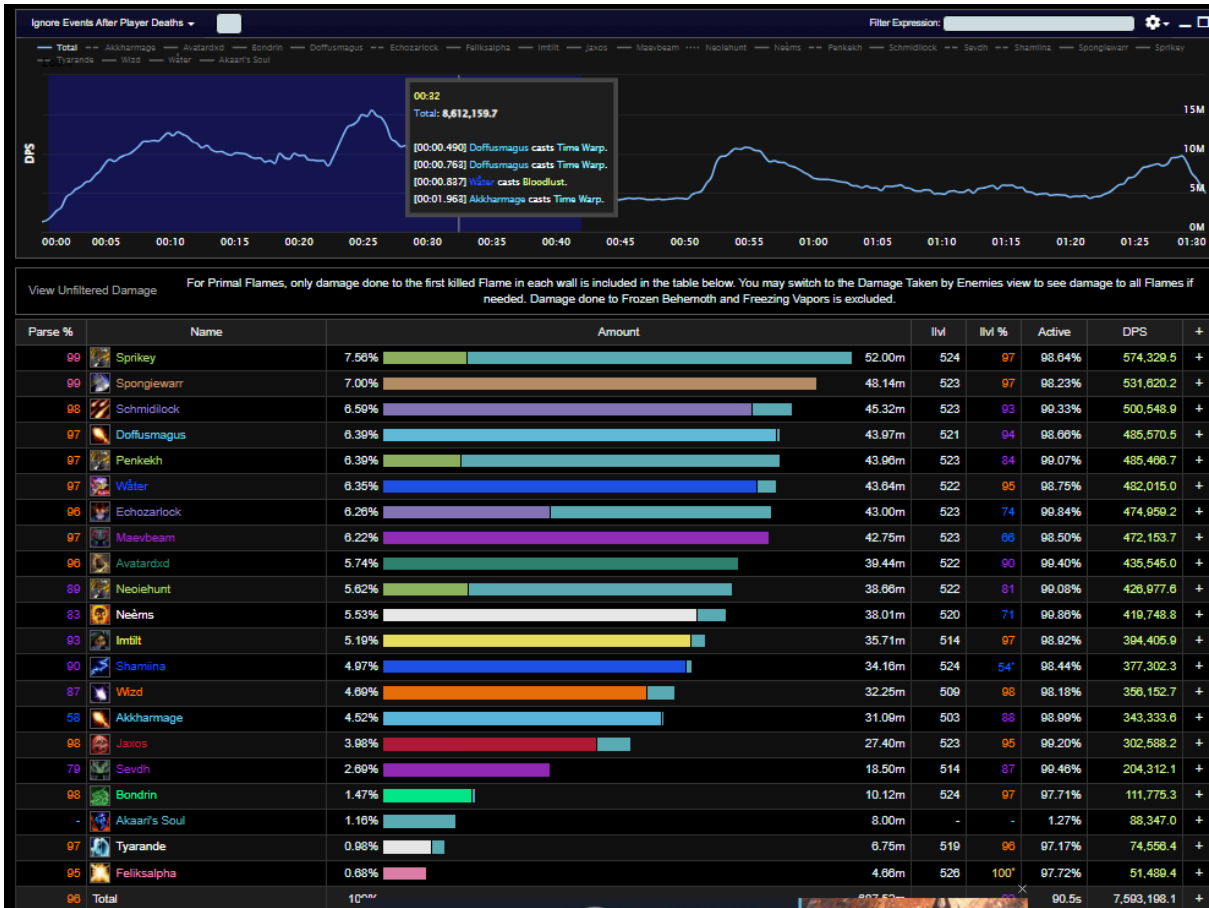
Datamatiker 2. Semester 2024

Mads Dalsgaard

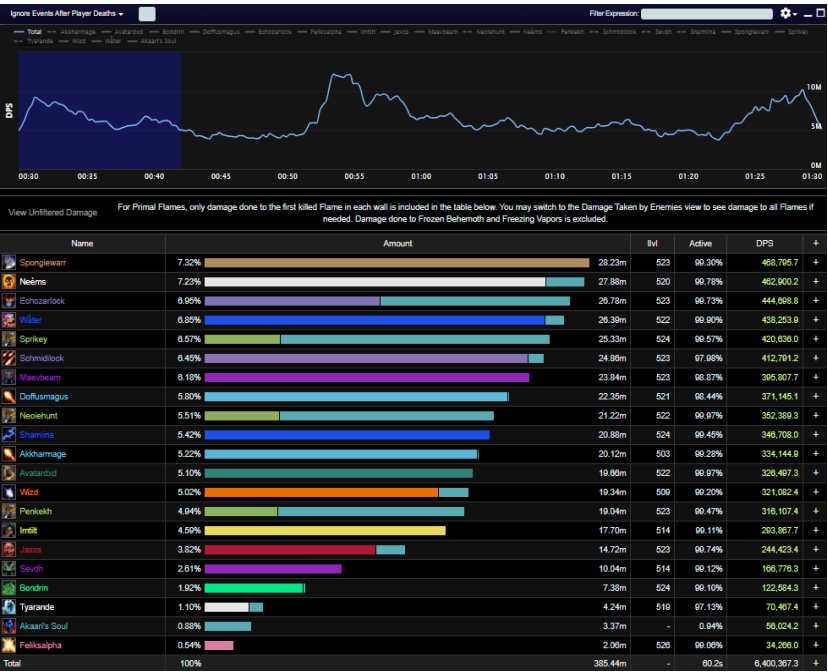
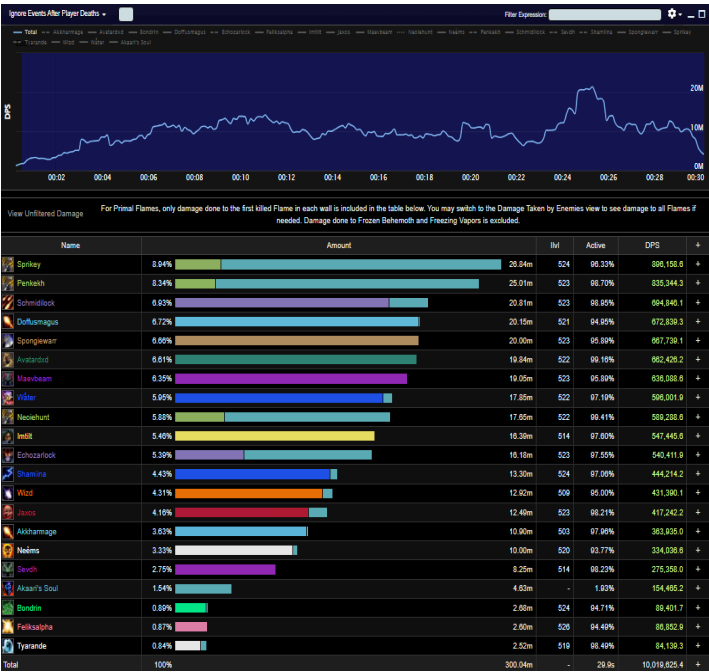
Fuglevænget 76a Grenaa 8500 - [Dsadfgttf@gmail.com](mailto:Dsadfgttf@gmail.com)

31/05/2024


Bilag 1



Bilag 2



Bilag 3



**Prescience**  
Talent  
25 yd range  
Instant  
12 sec recharge  
2 Charges  
Requires Evoker

Grant an ally the gift of foresight, increasing their critical strike chance by 3% [Fate Mirror: and occasionally copying their damage and healing spells at 15% power]for 18 sec.

Affects the nearest ally within 25 yds, preferring damage dealers, if you do not have an ally targeted.

## Bilag 4

1	Sprikey	Hunter	26770000	00:00
2	Penkekh	Hunter	24450000	00:00
3	Maevbeam	Demon Hunter	14220000	00:11
4	Echozarlock	Warlock	19660000	00:22
5	Sponmgiewarr	Warrior	18980000	00:33
6	Wäter	Shaman	12080000	00:44
7	imtilt	Rogue	4220000	00:55
8	Neëms	Priest	9590000	01:06
9	Echozarlock	Warlock	4220000	01:17
10	Doffusmagus	Mage	3470000	01:28

## Bilag 5

4

Attributter	ET_ID	BuffTime	Damage	Players
Tupler				
Relationsgrad				
Relationens tilstand				