**Challenge**.

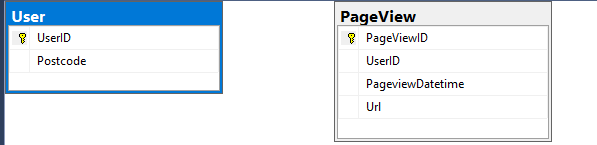
**Proposed Solution**

To Approach this challenge, I created 3 Different databases developer in SQL Server 2019 and an SSIS to cover the ETL Part:

# 1)**WebAgency db**

This will be the transactional database where we have all the data we want to analyze.

In this DB we have 2 tables:



For testing purpose in this Database I created 2 stored procedure that will allow us to insert a random number of data in order to perform testing with our ETL process.

**These two procedures are called:**

**[Test].[InsertUserDummyData] <**@NumberOfDummyUsers optional int>

This procedure will insert the specified amount of data into the User Table

**[Test].[InsertPageViewsDummyData] <@**NumberOfDummyPageViews optional int>

This procedure will insert a specified number of record into the PageView Table

# 2) **DWHExport** db

I created this db with the only purpose to export the data from the Transactional Db. I expect to have this database on the same instance of the source one.

In fact, it does not contain any table but only one procedure and one view that will allow us to export the needed data

**View:**

**[Export].[VwUser]**

This view will allow us to export the data of all the users

**Procedure**:

**[Export].[GetPageView] <**@StartTime optional datetime>

I decided to create a procedure to export the PageView Data because I wanted to parametrize the query and allow us to put a start date to decide from which date export the data.

If the parameter is not provided, we will export all the table

# **3) WebAgencyDWH**

Here we will have all the information about logs and the data lineage, the staging and the warehouse layer.

Usually, I keep these two layers separate but for this project as we have a small number of tables from my point of view there is no a downside to keep both in the same db.

In this db we will find 4 different schemas:

1) **Control:** used for the logging tables and procedures

2) **Stage:** used for the staging tables and procedures

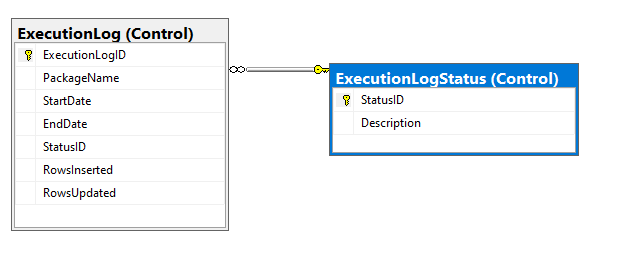
4) **Dimension:** used for the dimension tables and procedures

2) **Fact:** used for the Fact tables and procedures

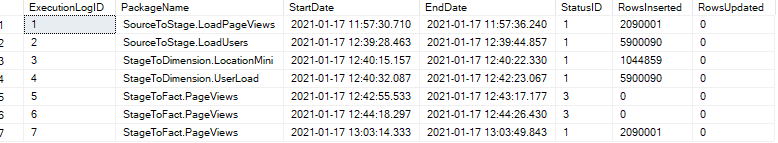
**Control schema**:

As I said here, we have all our logging tables.

We have two tables used to keep track of what happens each time we run an ETL package:



Example of ExecutionLog table contenent:



**Under this schema we have three stored procedures**:

**[Control].[StartLog]** < @Status int,@ExecutionLogID VARCHAR(200),@RowsInserted int Optional, @RowsUpdated int Optional>

This procedure allows us to start a log for the SSIS package

**[Control].[EndLog]** < @Status int,@ExecutionLogID VARCHAR(200),@RowsInserted int Optional, @RowsUpdated int Optional>

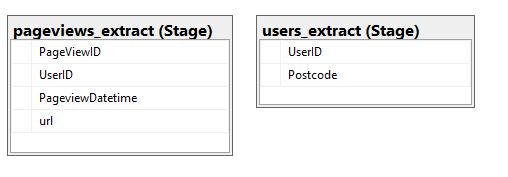
This procedure allows us to end a log.

[**Control].[LastExportedPageViewDate]** <>

This procedure will take the last exported date from the Fact Table allowing us to have an Incremental load when we export the data and speed-up the process

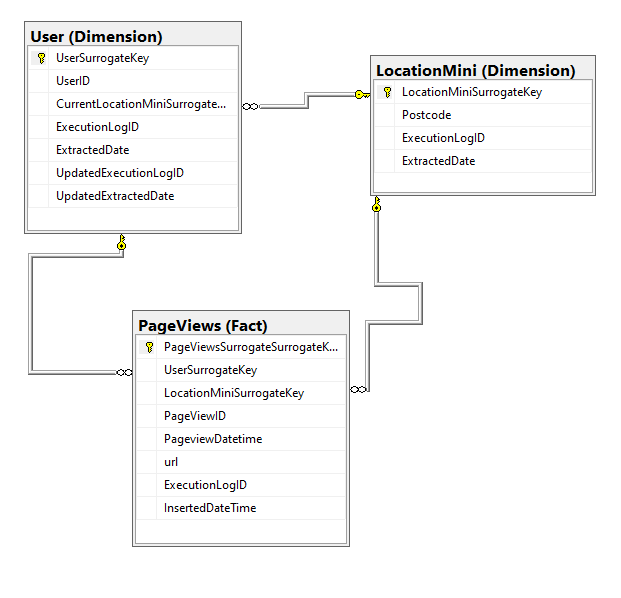
**Stage schema**:

In this schema we have the following two tables in which we will insert all the data from the source system:



**Dimension and Fact Schema**:

Tables:



The reason behind this model is to achieve the goal of keeping track of the PostCode of the User both current location and at the time he viewed the page. I wanted to achieve this goal keeping the model as simple as possible and reducing the amount of the data we will need to store at the minimum.

At first glance when I approached the problem, I was going to define a different model using only one SCD Type 2 dimension in which historicize all the data always keeping the old ones and inserting new data in the dimension, but after I realized that we have many users, and these users are likely to change often their Postcode I decided to treat this dimension as a Rapidly changing dimension.

What we are doing with this approach is just to insert all the possible Postcode into the LocationMini dimension Table and all the users into the user dimension plus their current location into the CurrentLocationMiniSurrogate column which is linked to the LocationMiniDimension.

Each load we will update the CurrentLocationMiniSurrogate column in the user table with the last location of the user.

While every time we insert a new record in the **fact table**, we will insert the Foreign key of the user from which will be able to retrieve the current location and then we will also insert the current location at the of the user into the LocationMiniSurrogateKey Fact Colum.

In this way if a user that had already a page view change the location we will be able to retrieve the current Location joining the User dimension with the LocationMini Dimension while if we want to retrieve the original value we will join the directly fact table with the LocationMini Dimension.

**Procedures under this schema**:

**[Dimension].[LocationMiniToDimension]**< @ExecutionLogID optional int>

This procedure loads only the new postcode from the staging table to the Dimension

**[Dimension].[UserToDimension]** <@ExevutionLogID optional int>

This procedure for each user in the Staging table check if it exists in the dimension, if it exists and the location changed updates it, if the user do not exists then it will insert a new record.

**[Fact].[PopulatePageViewsFactTable**]<@ExevutionLogID optional int>

This procedure populates the fact table

**Views**:

**Fact.PageViewsOriginalLocation :** Retrieve the number of the Page viewed grouped by the postcode with the original location from the users

**Fact.PageViewsMosRecentPostCode :** Retrieve the number of the Page viewed grouped by the postcode using the current location

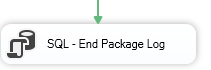
# **SSIS Process**

We have 7 packages in our ETL process and in every package we have always the same structure of logging

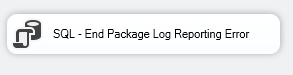
In each package will find the following tasks:

1. 

Inserts a new row into the Control.ExecutionLog Table

2)

Updates the row created by the previous task reporting success

3)

This task will be only executed if we have an error in the process and it will update the Control.ExecutionLog table reporting error

Packages Explenations:

1)**SourceToStage.LoadUsers**:

* Truncates the stage table
* Does a full load of the user source table into the User Staging one

2)**SourceToStage.LoadPageViews**:

* + Truncates the fact stage table
  + Get the last extracted date in order to perform the incremental load
  + Loads only the new record from the Source table to the Staging Fact table

3) **StageToDimension.LocationMini**:

* + Executes the procedure to load the new postcodes into the LocationMini dimension table

4) **StageToDimension.UserLoad**

* + Executes the procedure that load all the new user from and update the current location for the existing ones

5) **StageToFact.PageViews**

* + Executes the procedure to load the fact records from the staging table

6) **DailyProcess.GetUsers && HourlyProcess.GetPages**

* + These are the process we want to execute daily or hourly. To do that we use the SQL Server agent jobs but there are other ways as well using for example cronjobs or other solutions. Here the example of the jobs:

