A Database System To Support a Self-Service Email Marketing System

Author:
Ali Kiyan

Assessor:
Dr Mohamad Saraee

April , 2017

Introduction

EffectMail-EFM is a self-service email marketing software solution providing free and paid plans for enterprises. In this essay, various areas regarding database designing will be covered such as using database-independent documentation technique the *entity relationship diagram*, views, stored procedure, system functions, user-defined function, triggers and SSIS packages for getting files from user. Possible security solutions and backup policy for the system will be discussed. Business Intelligence techniques such as SSRS and SSIS will be used for reporting and adding packages to database respectively.Also probable concurrent transaction scenarios and the way they can be tackled will be discussed as concurrency control for efm.

In this essay, SQL Statements are just for demonstration purposes and the comprehensive list of statements for designing the whole schema can be found in the attached text file with this essay.

1.1 Preliminary database design for EffectMail

The formal way in which we express data relationships to a DBMS is known as *data model* and the data model that has been used for efm is relational data model.It is worth mentioning that Relational databases have a certain amount of data redundancy which is in the form of copies of primary keys (or candidate key) acting as a foreign keys.

1.1.1 Entity relationship modeling

First, for have a better udenderstanding of data relationships and characteristics of the given domain, an independent-DBMS documentation technique called *entity relationship* diagram with Chen notation has been used. This diagram provide better overall understanding of domain of interest.

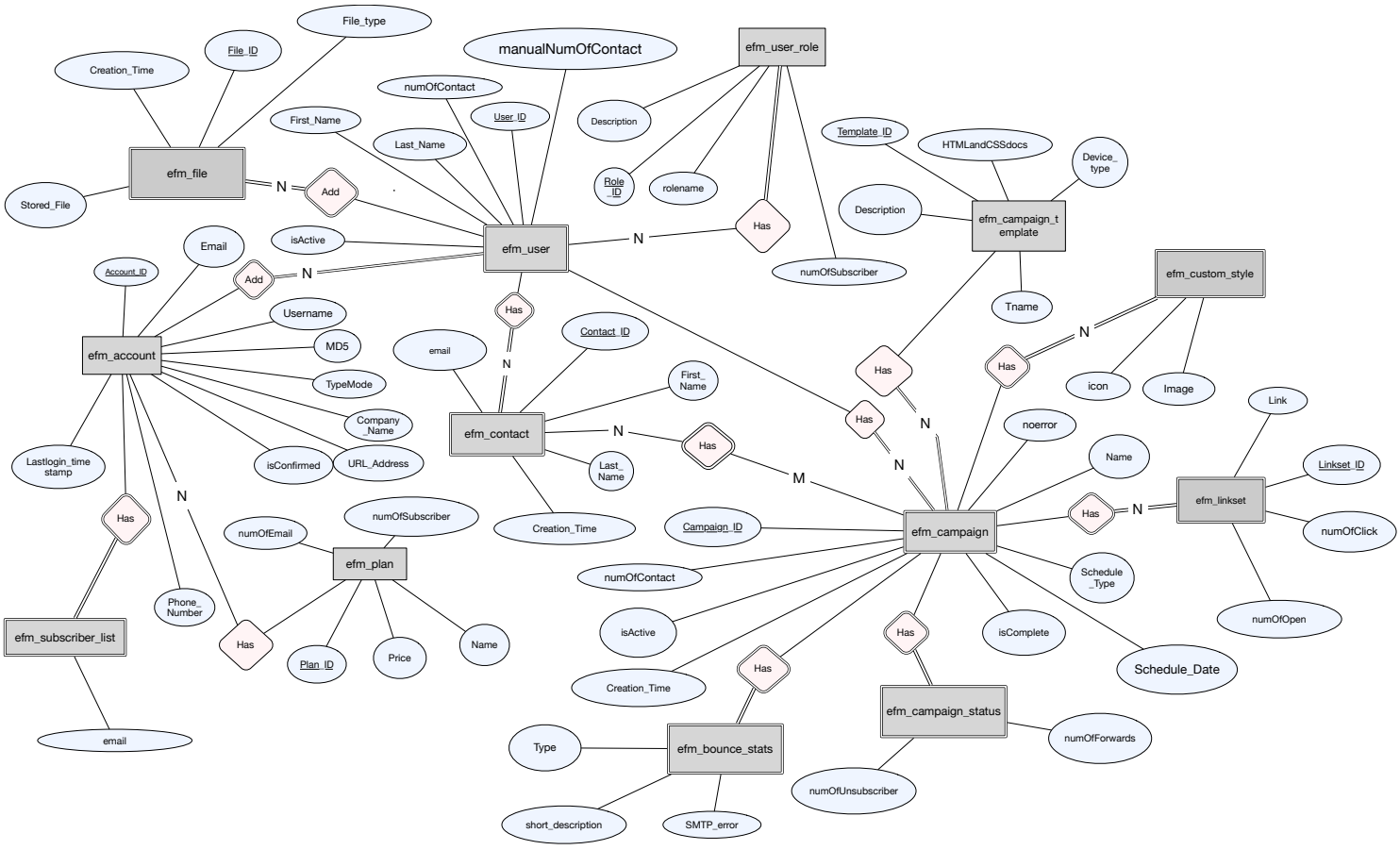


Figure 1.1 Entity relationship diagram for EffectMail

1.1.1.1 Relationships

Each relationship between entities can be divided to 3 following type :

1. Many-to-One relationship
2. One-to-One relationship
3. Many-to-many relationship

For example each account can have many users but each user can only belong to one account so the relationship between account and user entity will be one-to-many relationship while in relationship between account and subscriber list each account have one subscriber list and each subscriber list belongs to one single account so the relationship is one-to-one in this case.

Determining the multiplicity determines how key are transferred between tables. In one-to-many relationship the primary key from one table is transferred to many tabled as foreign key and in one-to-one relationships a key from more important entity is transferred to the other. In many-to-many relationships an intermediate table containing both primary keys of both tables will be considered. This table contains two group of attribute. One, primary key(s) from the first table and two, primary key(s) from the second table and all attributes in this table are considered as both primary key and foreign key.
The relationship between contacts and campaigns is many-to-many so in designing the schema, another intermediate table will be added containing primary key of both tables.
Notice :

- For avoiding recurring statements, we will not describing the whole attributes and their datatype, keys,Relationship type and how they are chosen in each case.
- Since both entity and table share the same concept in different area (ERD and Schema) ,as of now they are used interchangeably.
- The domain keys are the keys that have a meaning in a domain of interest and can specify each tuple. (sometimes one key can specify a tuple and sometime multiple of them)
  These keys can be used if they do not occupy large amount of storage and if they occupy large space another type of key (AI) will be used which will be later discussed in this report.
- The default implementation for relationships is one to many, but one-to-one relations can deployed by setting the value of foreign key as unique in the child table and considering the key from the more important table.

1.1.1.2 Designing entities, entities' attributes and entities' interconnections

For account table, MD5 is for getting hashed password input from user and lastlogin_ timestamp gets the timestamp of the server for having the last login time(Temporal database concepts considered in some table throughout the database design). It has Typemode attribute for getting the value of premium or free or other type of account in the future (upward compatibility). For understanding whether an account is activated or not isConfirmed boolean flag can be used and front-end developer can just check this value for understanding which users are active. valid company name, phone number and URL of the company are needed as well but they can be provided later so these attributes are nullable.

Email attribute is the one that appear in the from section in the campaign because multiple users can be added by account but in campaign the email of certain account should be shown. Each account can have subscriber list for sending newsletter or new offers to people who want this service, this is different than contacts which define by user and related to campaign. Each account can have one plan and in this table first check the isConfirmed flag in account table is checked and if it was true, it would allow account to have plan id of premium plans.

In user table, is active determines whether a user is active and last name attribute will be used in campaign to specify which user has sent certain campaign to contacts. User table can upload one or multiple contact CSV file which will be stored in file table and will be imported as SSIS package to contact table later. The user can upload the contact manually by adding them to contact table and spCountSubscribers stored procedure can return the total count of this contacts with subscriberCount output or trsubscriberCounter trigger can populated the value of numOfContact automatically whenever the user adds another contact to the contact table. In other case, manualNumOfContact can be taken manually from user for checking whether the user is eligible to use special backend tool. The account can give user different roles which limit their permission and this permissions defines in user role table. In this table different roles with different limitations can be imposed. 4 different roles has been defined which can be expanded in the future but for security reason they can only be added by database admin.

	Role_ID	rolename	description	numOfSubscriber	numOfemail
1	1	free	Account_Manager	100	500
2	3	minimumpaid	Marketing_Exexecutive	500	700
3	4	mediumpaid	Junior_Assistant	1000	600
4	5	paid	Marketing_Assistant	2000	1000000

figure 1.2 different roles for users

The user can have campaigns and can search through history of campaign using campaign id or datetime which later will discussed more in depth. The campaign has schedule type for scheduling a campaign for a certain date and schedule date stores the date which campaign is going to be sent. All the information in pop-up window of campaign can be either in campaign table itself or by joining it with related tables (user and account). Each campaign has a campaign template which contains HTML and CSS Codes for a template page and device type which denotes this template is related to which type of device like tablet or phone. The User can manipulate the outline of the campaign by adding SVG social buttons, images and they are stored in custom style table.If the user is happy with campaign and pop-up summary, campaign can be sent and if it is sent to all recipient the no error flag will be set. After sending the campaign there are various ways to keep track of the campaign and monitor status of the campaign.

Campaign status table store data about unsubscribed contacts and forwarded times of campaign.

Each Campaign can have one or more contact and the relationship between campaign and contact is considered many-to-many, so in some cases a campaign can use contacts of other users as well and as mentioned, an intermediate table with both tables primary keys is considered for implementing this. (We have used many-to-many relationship for more flexibility of database and if each user had to access only his/her contacts there should not be any intermediate table and the relationship between contact and campaign would be one-to-many.)

One of the handy features of efm is providing a heat map for keeping track of the mouse of the user.

Heat-mapping analysis can enhance UX. Although it can provide hover maps in which the mouse hovering pattern is represented, efm is more focused on a representation for showing all the clicks especially clicks on link sets.

Figure 1.3 Heatmaps can boost user experience
(picture from http://pixelmarketing.net)

Information from heat map, namely number of clicks and opens for each link will be stored in link set table. In this table each link in the campaign can have two attributes of numOfClick and numOfOpen.

Depending on the mail server that has been used there are certain codes which you will get from recipient mail server denoting whether the mail submission has been successful. An email bounce signifies failed e-mails and in this case there are two type of bounces.

1. Soft Bounce :

It indicates that the email address is valid and reached to the recipient mail server but it bounced back for other reason.

2. Hard Bounce :

It indicates that there is a permanent issue with recipient email address. It means, either recipient email address is not valid and it has been permanently rejected by the server.

Bounce stats table stores these type of data for each emails providing the delivery status of a campaign.

In following figure 1.4 bounce stats table of efm has been shown and for better implementation type 0 defined as soft bounce and type 1 defined as hard bounce.

	Campaign_ID	Type	short_description	SMTP_error
1	1	0	The message was refused because the mailbox nam...	553
2	2	1	Cryptographic algorithm not supported	576
3	4	1	Wrong protocol version	555
4	5	NULL	This response is what results in a 'Delivered' event w...	250

Figure 1.4 Bounce stats table in efm

In efm service, in plan the policy of company is either has unlimited e-mails for paid plans or 500 e-mails for free plans so a bit datatype could have been used but for upwards compatibilities purposes int datatype has been used for providing more options in future.

1.2 Designing database schema

1.2.1 Defining datatypes

Choosing appropriate datatype not only improve storage and overall performance, but it also improves data integrity by ensuring correct input has been inserted into database (even for security purpose)for example, bounce code for SMTP bounce code are always 3 digit code so CHAR(3) has been used as a integrity constraint as well as a place holder.

To avoid stating the obvious, not all the attributes will be covered.

- Since phone number is always 10 digit and a 10 digit datatype has been considered.
- bit datatype is equivalent of boolean in T-SQL.
- Varchar has been used for those character name with varied name.
- Timestamp is for getting a time of the server and since it is not human readable time format, at times, we need to using conversion system functions like CONVERT to change the timestamp format to a date time format which can be compared with other attributes. (for example this happened at searching through campaign by date section).
- Nvarchar is for unicode name where there is a possibility of other languages.
- Text is the same as Varchar(max) and both can store up to 2GB
- Small money has been used for price and can store up to 4 byte values.
- Datetime has been used with millisecond precision.

Suggestion: according to MSDN timestamp datatype is deprecated and it is better to use rowversion but since this datatypee has been used in other databases and for readability purpose, timestamp datatype has been used.

1.2.2 Integrity Enhancement

Integrity Control and Enhancement consists of constraints that are imposed to prevent inconsistency. In efm 4 type of constraints have been employed :

3. Required data :

Due to the needs of the client, not null or nullable variable should be employed for example in account table the account should have username to login to or shown on from part of email.

4. Entity integrity :

The primary key of the table should be unique and Entity Integrity ensures that primary key is unique by setting it as not null attribute and making it a unique attribute.

5. Referential integrity :

Referential Integrity means that the value must refer to a valid primary key or it can be null and SQL will reject any attempt to UPDATE or INSERT a value that violating this constraint.
In UPDATE and DELETE operation in parent table SQL acts differently according to referential action that can be modified on ON UPDATE and ON DELETE subclass of FOREIGN KEY clause or it can be modified using GUI in foreign key section.
There are 4 type of option regarding the action to be employed in case of deleting a primary key in the parent table and its effect on child table :

- CASCADE
  Both primary key and foreign key are deleted
- SET NULL
  Put a null value on child table's foreign key
- SET DEFAULT
  Put the default value of foreign key
- NO ACTION
  Rejecting any deletion from parent table

In efm NO ACTION option has been used for all the foreign keys

1.2.3 Auto key

As mentioned earlier, domain keys can unique the tuples of a table by having a unique semantic in the domain of interesting and sometimes they can be a combination of different attributes to specify a certain tuple. In this cases, database offer a technical key called “auto key”.
Auto key is an integer number that can be incremented or decremented to provide a unique number for a tuple. Auto key can be used instead of combination of keys or those keys that occupy large amount of space.

Suggestion :

Some may argue that using technical keys instead of domain keys can compromise the readability which is a true statements but they offer better performance as well.

1.2.4 Designing

With attributes and their datatypes, relationship between them, primary key and foreign keys , database can be created by writing T-SQL statements. All of the statements are available in T-SQL text file. Sometimes for amendment SQL Server GUI can be used for alter Tables and their connections.

1.2.5 Normalisation

Normalisation is database design techniques examining the relations (functional dependencies) between attributes and even entities. In other words, Normalisation uses different tests to help identify optimal database design.

There are two main approaches for using normalisation. The first one is using a buttom-up standalone technique which is how to normalize data from the beginning and is used in the creation of efm service. The second approach on the other hand, shows how normalisation can be used as a validation technique to check the structure of the database and relations and which may have been created by a top-down approach before.

It is important that the database is normalized because :

- It suggest the minimal attributes to comprehensively support the sets of data for the enterprise
- It suggest a design that has minimal redundancy in the way that in the way that each attributes only describes once except for foreign keys (which are essential for joining database's tables)
- It prevents inconsistency throughout the database
- Updates of the stored data on the database are done by minimal operations
- They cause final reduction in the file storage space

Levels of Normalisation

Normalisation consists of 7 different levels and before mentioning these level there are some concepts that should be covered.

Figure 1.5 Functional

Functional dependency is mentioned when A and B are attributes (Or can be group of attributes) and B is functionally dependent on A if each value of A only associates with one value of B.

Determinant

In the functional dependency the left side of the the relationship is determinant.

Full Functional dependency

In the above relationship, B is fully functionally dependent on A if B is functionally dependent on A, but not dependent on any proper subset of A.

Transitive Dependency

If A ® B and B ® C, then C is transitively dependent on A through B.

Unnormalised Form (UNF)

A table that contains one or more repeating group

First Normal Form (1NF)

The intersection of each row and column contains one and only one value in other words there should not be any multi-valued in the table.
Multi-valued attributes can be transferred to another table with one-to-many relationship.

Second Normal Form (2NF)

A relation that is 1NF and every non primary-key attribute is fully functionally dependent on the primary key.

Third Normal Form (3NF)

A relation that is 1NF and 2NF and no non-primary-key attribute is transitively dependent on the primary key.

There are more 4 levels of normalisation after 3NF, namely BCNF, 4NF and 5NF and DKNF but due to some reason in most of real word database designs they are not deployed.
Firstly, they have impact on overall performance of the system. Secondly, They may facilitate some statements(for example UPDATE) but in some common statements they are relatively slow (for example, READ) and all in all they function better at UPDATE and unsatisfactory in retrieval statements. They are not provide maximum processing efficiency and sometimes due to the problems mentioned the database design is denormalised from a higher level to a lower one.
In efm database design is only normalised until 3NF.

Suggestion :

Some database designers believe that it is better to normalized the data until it starts to have major problems and the denormalise to have the most optimal database design.

Look up tables can be used for static lists like city or country names and can be used in multiple places and sometimes they referred to as data dictionary. They provide a higher level of abstraction with overhead as well.

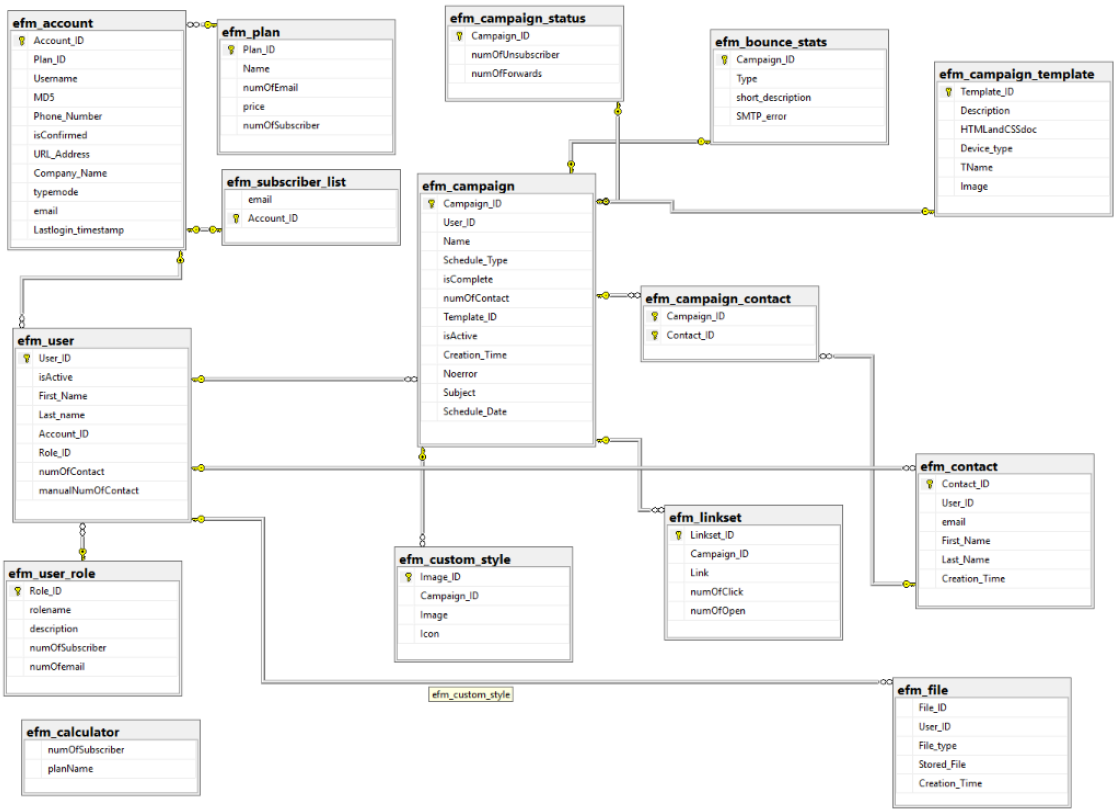


Figure 1.6 efm Schema

Figure 1.3. is final schema for efm service after normalisation to 3NF and adding Auto key instead of lengthy and multiple primary keys. Now each level of normalisation is discussed in efm service.

Notice :

- Due to the experience of the designer of efm database, top-down normalisation approach has been selected for normalisation. For example if bottom-up had been selected, in account table, all the attributes of plans table would have been in the account table and then due to the 3NF and transitive dependency of attributes of plan table on plan plan name, it had to be separated in a table.
- Due to the mentioned characteristics of "Auto Increments" ,they have been used throughout the database design.
- It is worth mentioning that a relation with single value primary key is automatically 2NF and since all of the tables of efm has single-value primary key, they are all 2NF.

In Account table, phone_Number has the potential of being multi-valued attribute and should have been separated in another table but efm ask their clients to just provide one phone number but Subscribers' email could be multi valued so they are separated in another table called efm_subscriber_list. (1NF)
URL_address of the company can be transitively accessed through Company name but there is a chance that the URL of company change so due to better performance new table for URL_address is ignored.
plan table does not have any multi-valued column (1NF) and there are no transitively dependent attributes to primary key (3NF).

In the user table all attributes all single-valued (1NF) and there are no transitively dependent attributes.(3NF)
User_role does not have any multi-valued attribute (1NF) and all of non-primary attributes are not transitively dependent on role_ID.(3NF)
In the file table all attributes all single-valued (1NF) and there are no transitively dependent attributes.(3NF)
In the contact table all attributes all single-valued (1NF) and there are no transitively dependent attributes.(3NF)
Campaign table does not have multi-valued attributes (1NF) and any determiner non-primary attribute.(3NF)
Campaign_template table does not have multi-valued attributes (1NF) and transitively dependent attribute(3NF)

campaign status and bounce status were both two multi-valued attributes which transferred to another table respectively and both of them do not have any multi value attributes and any transitively dependent attribute(3NF)
Custom_style table does not have any multi-valued attribute(1NF) and there is no transitively dependent attribute(3NF)
Linkset table does not have any multi-valued attribute(1NF) and there is no transitively dependent attribute(3NF)
Lastly, calculator does not need primary key due to its characteristics so 2NF and 3NF are not arguable at this point and there is no multi-valued attribute so this table is 1NF.

1.3 Other features of efm

1.3.1 Stored Procedures, Views, Triggers, User-defined Functions and System Functions in efm

Stored Procedure

Efm allows searching through the history of campaigns by entering the date value and returning all the campaigns before that date. For Example the following expression returns all the campaigns before the mentioned date and since this command is used every day by IT staff an stored procedure has been considered and in the stored procedure a view has been used to have all the information related to campaign at once.
spcampaignhistorysearch '2018-01-01 00:00:00'

The key point is that when client enters the date through GUI is not the same format as the Creation_Time which is Timestamp so for comparing these two one of the should be converted to the other's datatype. The respective T-SQL code would be like this :

CONVERT(datetime,Creation_Time,121)

In this code, 121 is ODBC Canonical standard for providing the right format for Creation_Time to fit in datetime datatype.

Campaign can be searched by their id (using an input in stored procedure) and this is provide by spcampaignsearchById.

There is a handy stored procedure for facilitating the activation/deactivation of users by getting the userID and activation type. Since activation of user is common in efm , spuseractivator is going to be used numerous times. The following syntax set user with id 6 :
Spuseractivator 6,1

Searching through subscribers is a requirement for efm service, and this is met by spSearchSubscribersByEmail stored procedure which takes an Email as an argument an then search whether that is exists in contact table. (Note that we have another table called subscriber list which is different than contact table and subscriber list is the list of people who have left their email for newsletters of company).

The number of contacts (subscribers) are important to efm company and there are different ways to have that total number and as mentioned we used two ways. First by having spCountSubscribers stored procedure that can pass the total number with @subscriberCount output variable and another way is by using trigger which will be discussed later i trigger section.

When the user click send button in the front end part, three parameter should be stored and post to database. 1.Account ID 2.UserID 3.Campaign ID. These three parameters are sent to spPopup stored procedure as argument and the store procedure provides, from email, last name of the sender of email and title of contact and number of email that the campaign and send them back to front end part to be used in pop up window.

View

Subscriber list is different than the email list but sometimes it is handy to have them in one place for example for sending an important news to people who have somehow interested in company either by putting their email on subscriber list or by including in on user's contact list, so efm has considered a view combining both tables.

There is another view that allows having all the the users even without those without roles to have an understanding about all the users.

There is another view as campaign_full view which joins all tables related to campaign to keep all the data related to campaign table in one place and it can be used in different place like in other stored procedure.

Triggers

Triggere are defined within the table and in efm for number of contact two triggers have been defined. One trigger for getting the number of contact from contact table and inserting it to

user table(trsubscriberCounter) and one trigger for getting the number of emails of campaign which is extracted from efm_campain_contact table(trcampaignsubscribercounter). These two table functions almost the same and they update the numofcontact attribute when some contact is inserted. (It uses **for insert** key word and inserted temporary table which is created whenever something adds to the related table)

truserSubscriberCounter is a trigger that can populate the numOfConcat value of each user whenever that user adds a contact in contact table which is handy because it can be used as another feature to for example users that have more than 2000 can use certain tools.
For security reason, the acccount should be confirmed and if its not confirmed the trconfcheck trigger does not allow insertion to the table and roll back the transaction.

There is another trigger called TRplan trigger which controls the number of subscribers and number of emails that each user according to his/her plan can have.

USER DEFINED FUNCTION

For the calculator part the user key in the number of subscribers and the user defined function returns proper plan name and due to policy of the company that the number of subscribers should be limited to 2000 subscribers per month, proper datatype has been chosen.

For checking the user defined function if it is table valued function it can be on from clause in select statement or if it is scalar function name it can be in select statement or where clause.In the calculator it can be checked by following code :

Select Planpricecalculator(345) ==> minimumplan

The user enters the number of subscribers, so there should be a table that stores that data and then after calculating the plan store it in database. This table called efm_calculator and since it is not necessarily related to any other tables it can be an isolated table.
There no join and any connection in this table so in this special case the primary key can be left out.
Notice : Calculator can be implemented by programming (without using database) and an isolated table has been created just as a possibility.

Notice :

For security purposes it is better to encrypt stored procedures so they cannot be viewed thereafter with option WITH ENCRYPTION when defining an stored procedure but since in this assignment stored procedure should be viewed, WITH ENCRYPTION option has not been used.

Notice :

cout(emial) could be deployed without using output variable but it could not be accessed elsewhere so output put variable has been used and in case of using output variable in your stored procedure, you should declare a variable as output as a placeholder of the output of your stored procedure. The T-SQL code would be like this for printing out the number of the entire contact list.

```
DECLARE @TotalSubscriberCount INT
Execute spCountSubscribers @subscriberCount = @TotalSubscriberCount out
print  @TotalSubscriberCount
```

SUGGESTION :

Sometimes it is a good practice to use system store procedures to have better control over our system. Here “sp_depends Table name” can be used to check whether there are dependencies and if there are some stop for example deletion of that table.

1.3.2 Security

Security is of paramount importance and efm provides security in various ways.
Since efm can have different database users it can be secured by granting or denying databased permissions to different database users and then those roles can be assigned to different database users, but there is only one DBA at the efm for the time being so that person is sysadmin and have all the permission. In future, if the company hires more database users, it can be more secured by limiting the access of database principals. This can be done by GUI in security folder or T- SQL CODE A sample code would be like this :

```
USE efm
GO
DENY SELECT ON efm_campaign TO [SECOND-ROLE]
GO
```

second role can be assigned to any multiple users.

You can give each department users which are not familiar with T-SQL queries a view to tables that contains all the joins and conditions to first provide them with related information and second restrict their access to view all rows which is commonly called row level security. Column level security can be deployed as well by eliminating the column from select clause. You can restrict users to aggregating data as well by using aggregating function in select clause.

Encryption can be used in different parts of database like in stored procedure or in the time of performing a backup.

Sql Injection can be a threat for the database if the entered query is handled hard-coded and dynamically.
There are two different approaches to tackle this issue :

1. Using parametrized queries
2. Using stored procedure

Using parameterized query to prevent sql injection

In this approach instead of using like ‘ in the hard-coded part of query, @sampleparameter can be superseded and then by declaring this parameter we can have the entered value in the @sampleparametr parameter. The way it implements is that an escaped statement executed by SQL server system stored procedure called “sp_executesql”

for example in case of example the the statement would be like this :
exec sp_executesql N’Select * from sample where samplename like @sampleparameter’ ,
N’@sampleparameter nvarchar (41)’ ,
@ProductName=N’ escaped enterned value’

Using stored procedure to prevent sql injection

This approach is like the previous one but instead of calling a system stored procedure, we define the stored procedure with parameter and then the SQL Server will escape the entered string and treat it as a value not statement
spsample

```
Create procedure spsample
@sampleparmeter nvarchar(50)
as
begin
   Select * from sample where samplename  like @sampleparameter
end
```

1.3.3 Concurrency Control

The efm is designed to minimum the concurrency issue and there is a lot any many-to-many relationships in efm schema and most of the time a table is not changed by multiple users at the same time but still there is a fair chance that two transaction run by a user or account and in campaign_contact table which is intermediate table for many-to-many relationship there is possibility of concurrency issues since multiple users can change data of this table.
In case of querying the database or updating table's values in transactions there would be different occasion where data could be inconsistent.
Concurrency problems can occur throughout the efm database if two transactions try to change a data.
In Dirty read issue the value that has been queried by select statement can be changed by rolled back transaction which results in misleading value.
In efm, when a user is adding or updating a contact or a new campaign or new custom style there is a possibility that another transaction is reading from the same tables at the same time. In this case, if the first transaction is rolled back, the data that is read by transaction 2 simultaneously is incorrect.
Lost update is happening when two transactions are trying to update the same value and the data can be inconsistent since the transaction that commits later is overwriting its value disregarding the updated value of second transaction.
In nonrepeatable read issue, a transaction reads a data and at the same time the second transaction updates the data for example change the campaign name and its error status of campaign and then if the first user wants to read the table for the second time it gets different value in the same transaction.
In phantom read the first transaction is reading a range of rows and at the same time another transaction add another row in between the range of read rows by the first transaction and then the first transaction gets different rows in the same transaction. (same as the nonrepeatable read but in row row level)

All of these issues can be tackled by an isolation level of sql server which is illustrated in table 1-3 .
It is worth mentioning that as we of higher in isolation level then number of concurrent users will plummet but in snapshot isolation level unlike the other levels there is no lock in transactions and it is implemented by versioning in tempdb so the number of concurrent user is improved in comparison serializable isolation table.
Due to the importance of consistency of the data in efm, snapshot isolation has been selected so that concurrency issue become minimum and the number of concurrent users become maximum.

Table 1-3-3 Concurrency Issues and isolation levels in SQL Server

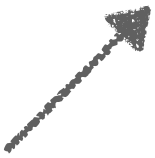
Isolation Lavel / Concurrency Issue	Dirty Reads	Lost Update	Nonrepeatable Reads	Phantom Reads
Read Uncommitted	✓	✓	✓	✓
Read Committed	✗	✓	✓	✓
Repeatable Read	✗	✗	✗	✓
Snapshot	✗	✗	✗	✗
Serializable	✗	✗	✗	✗

As mentioned before all the concurrency issues happens when two or more transactions are running simultaneously for example, in dirty read issue in read uncommitted isolation level the syntax for two transactions is as below.
Transaction one tries to update the template id for campaign with ID of 4 and then for an error it is rolled back. Meanwhile, another transaction tries to read the template ID for campaign with the ID of 4 and in this case with following syntax dirty read will happen because the template ID will go back to its previous value when the transaction one is rolled back.

TRANSACTION ONE

BEGIN TRANSACTION

Update efm_campaign SET
Template_ID=2
WHERE Campaign_ID = 4
.
.
.
ROLLBACK TRANSACTION



TRANSACTION TWO

SET ISOLATION LEVEL READ UNCOMMITED

SELECT Template_ID from campaign where Campain_ID =4

In this case transaction selects template ID of 2 but the real consistent value is the the default number for template ID which is 1.

1.4 Business Intelligence

Business intelligence is an umbrella term for referring to collecting and analyzing data and technologies used in this process which is facilitate corporate decision makings. Efm uses business intelligence techniques to enhance its functional capabilities. It uses SSIS packages for importing files to database and SSRS for getting a report from certain tables.

1.4.1 Importing Contacts using backend tool (SSIS Package)

In efm service, users can upload their files into the database and the manualNumOfContact attribute is checked by front-end software team to see whether they are eligible for using SSIS packages or not. (It could also check numOfcontact as an extra feature.)
In case of eligibility, their uploaded CSV file imported to contact table.
Notice : For demonstration purpose, the content of contact table has been removed and user_ID is not provided in CSV file so imported contact by SSIS package can be visible in contact table.
There are multiple ways of Importing data CSV file into database. Two of them demonstrated as following:

1. Importing data using SQL Data Tools

First New Integration Service Project should be created in SQL Server data tool and in Control Flow tab for simplicity just data flow task add and in data flow from SSIS Toolbox flat file source and OLE DB Destination(Microsoft API for allowing access of data from different sources)are added.
In Flat File editor, new connection is created and in browse section, CSV Contact is selected.
On the left hand side of the same window there are 2 options that need to modified. In column section comma is chosen for delimiter and in advance section encoding for each attributes would be unicode string. Then Flat file is matched to OLE DB destination.
In OLE DB new connection is established by specifying server name and using SQL server authentication, providing password and choosing efm as database. Next, Destination table efm_contact is selected. Then the mapping can be chosen for checking. Ultimately, run the package and the CSV data will be imported.

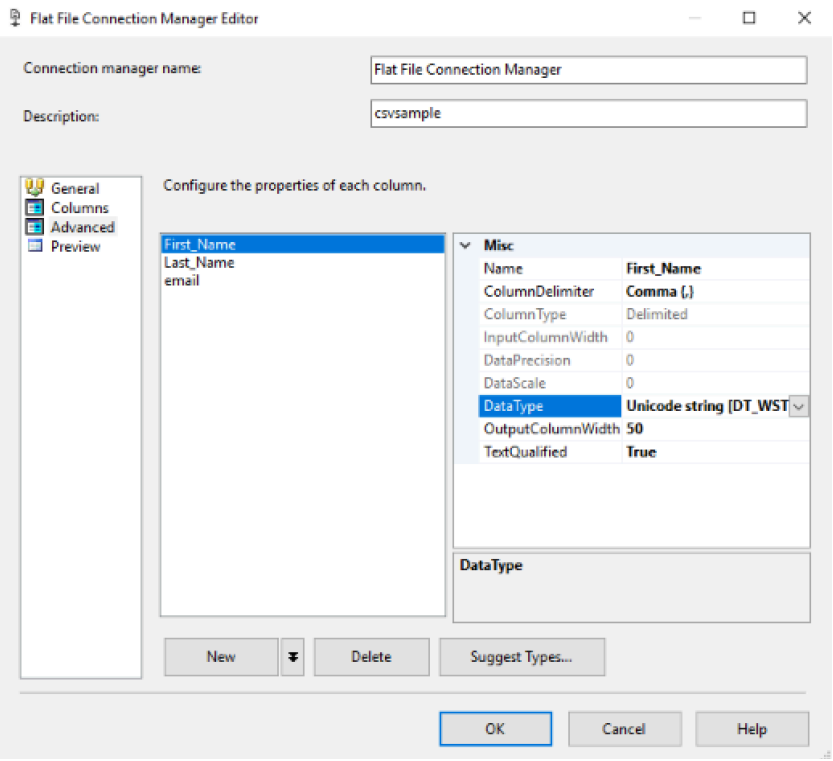


Figure 1.7 Selecting unicode encoding for

Notice : The encoding is important in this section because the imported CSV file is in unicode format so either the related attribute in efm_table should be unicode as well or the data conversion in data flow should be added.

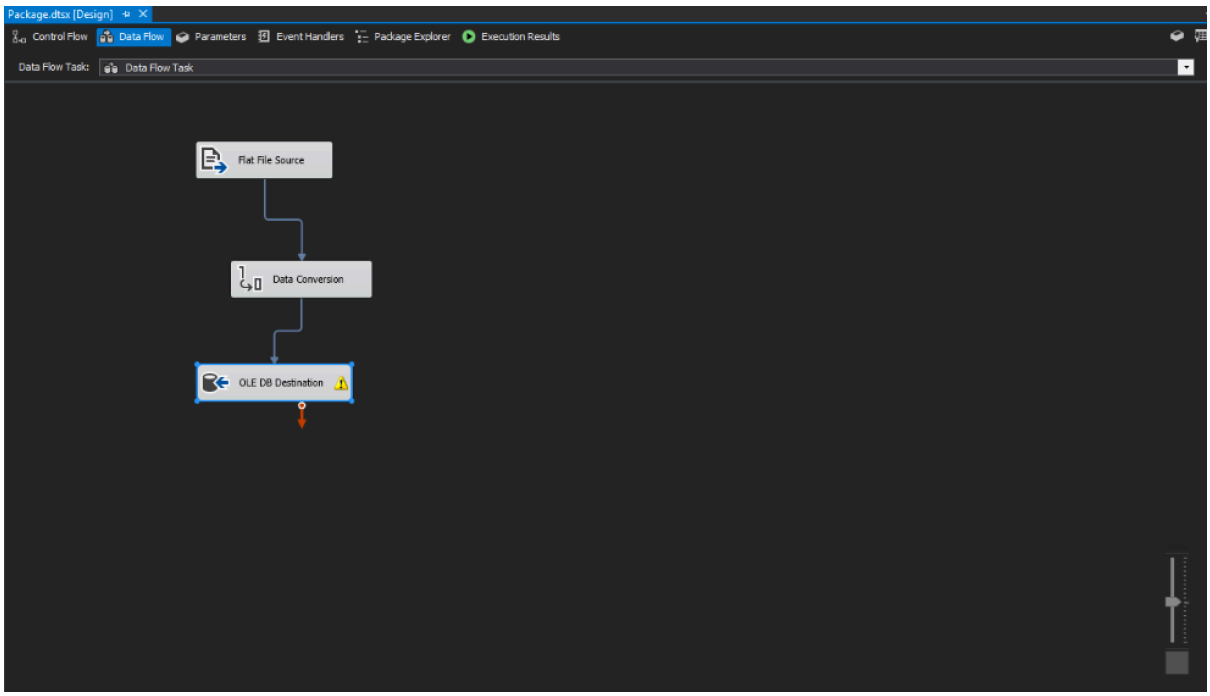


Figure 1.8 Adding Data

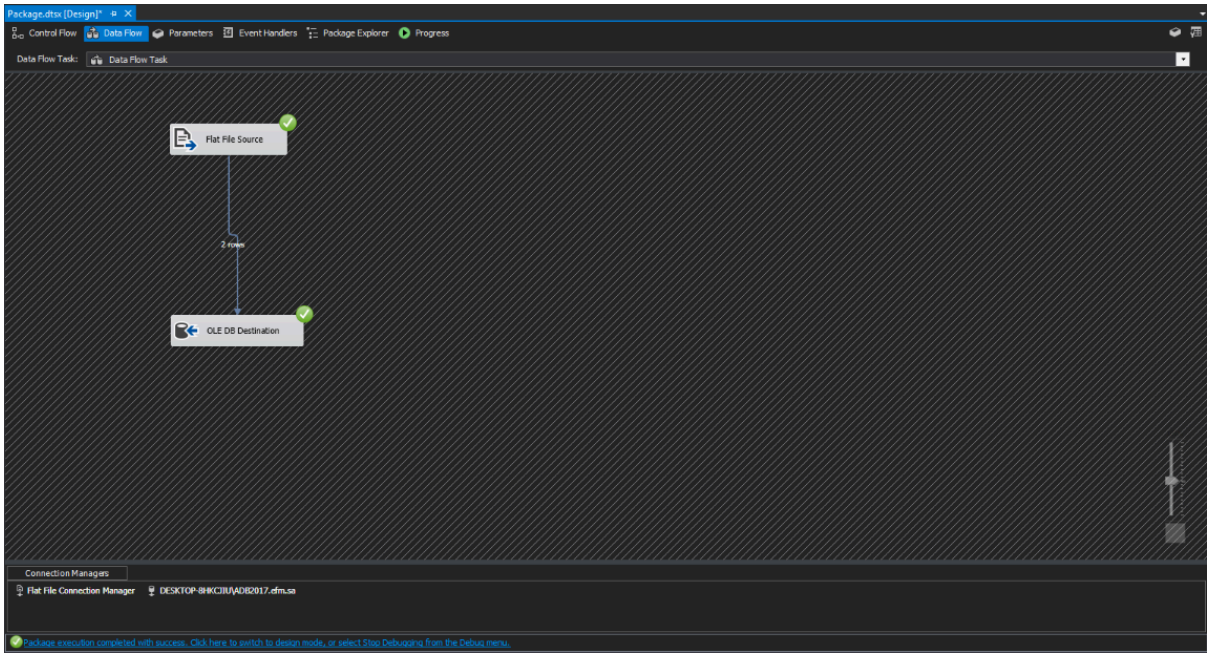


Figure 1.9 Completed package using unicode

Note that the Rows with “Null” values in User_ID are imported by SSIS package and NULL is just selected for emphasising that these contacts are imported by SSIS package otherwise user_ID should be provide in CSV file and therefore there was not NULL in user_ID column

Contactcsvsample.csv

```
1  First_Name,Last_Name,email
2  Jasmin,Dean,Jasmin.dean@test.com
3  Liam,Well,liam.well@gmail.com
```

	Contact_ID	User_ID	email	First_Name	Last_Name	Creation_Time
1	3	1	AliKyand@gmail.com	David	Murphy	0x0000000000004651
2	4	NULL	Jasmin.dean@test.com	Jasmin	Dean	0x0000000000004652
3	5	NULL	liam.well@gmail.com	Liam	Well	0x0000000000004653

Figure 1.10 Appended value of selected csv file into efm_contact table

2. Importing data using SQL Server Import and export wizard.

In this method, the backend staff right clicks on database and in Task selects import data and then chooses the flat file as data source. In this window delimiter and encoding for each row can be modified. In case of csv file comma as delimiter and used non unicode encoding for attributes is selected.

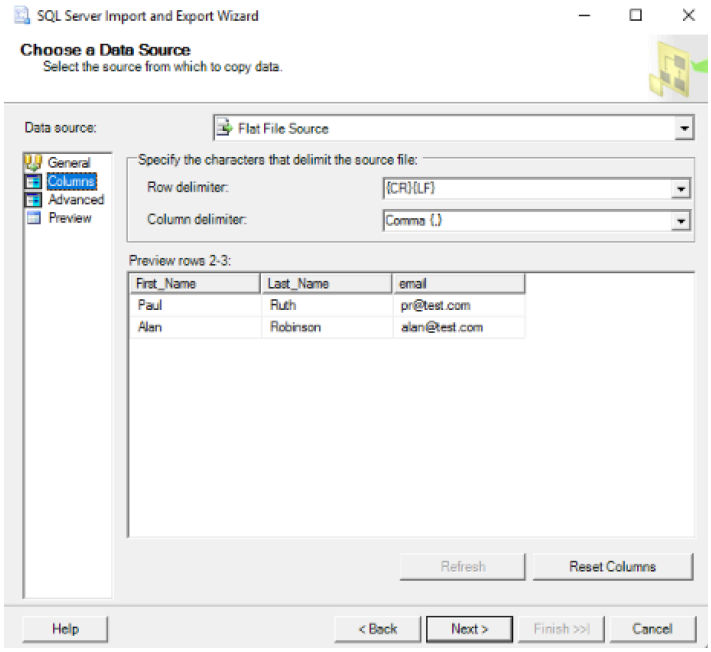


Figure 1.11 Choosing csv file

Native SQL Server and server name is selected and the authentication and database is correctly is filled.

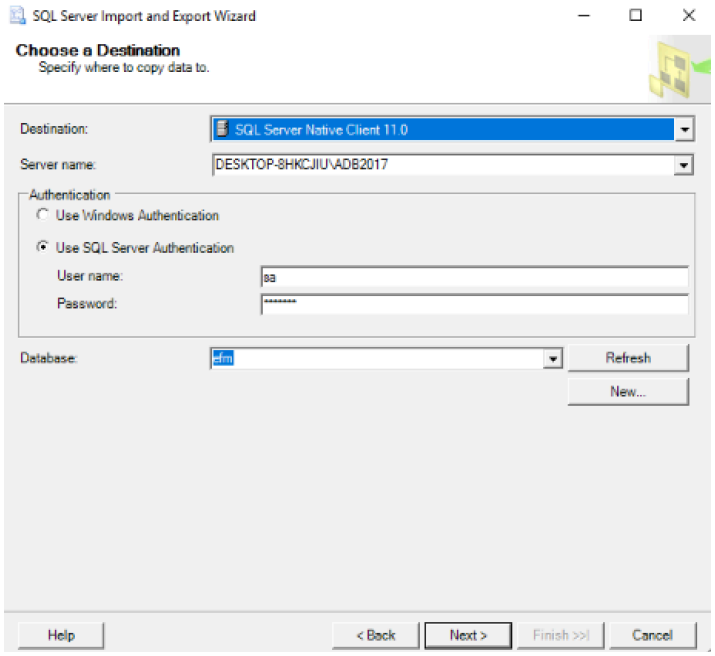


Figure 1.12 Destination Table for importing CSV file

In the next window for appending to existing table of contact, destination efm_contact is chosen. Then in edit mapping append rows to the destination table is checked.

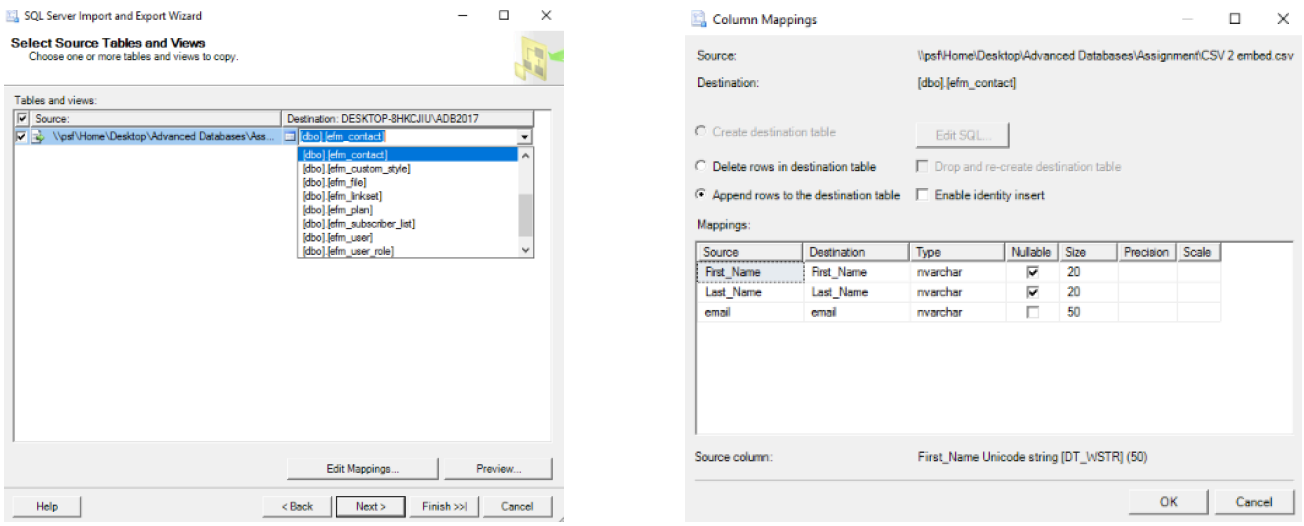


Figure 1.13Appending CSV contents to the existing table

After running this package, All the contents for CSV file is appended to User's Contact list and by outputting the content of efm_contact this can be proved.

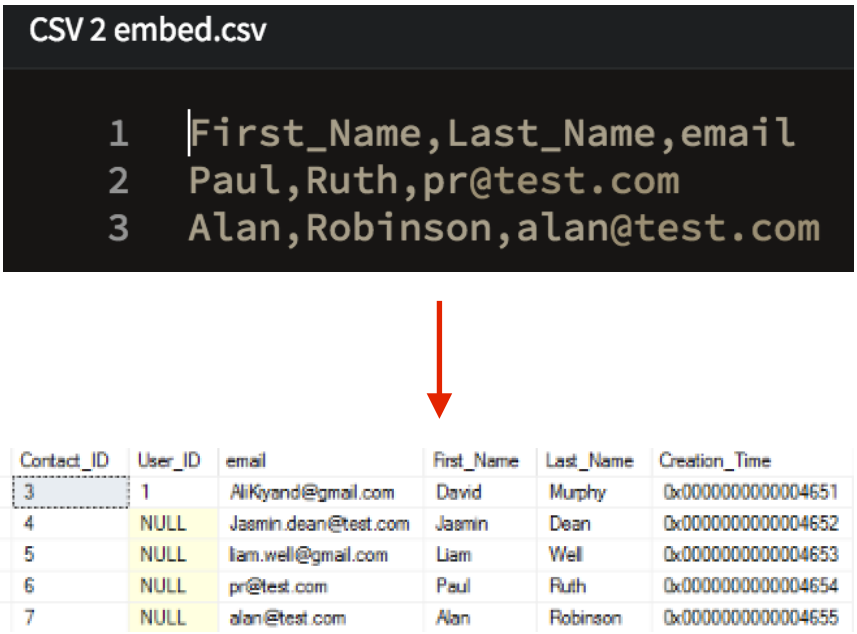


Figure 1.14 Appended CSV data at the end of efm_contact table

Null user denotes that the related content is imported by backend tool for demonstration purpose and in reality user_ID should be provided.

1.4.2 Using SSRS for reporting

efm uses business intelligence platform SSRS for reporting campaign statistics and Users' information. First a new Report Server Project and report is created. Either shared or embedded data source can be used and efm is using embedded data source.

1.4.2.1 Reporting users' information

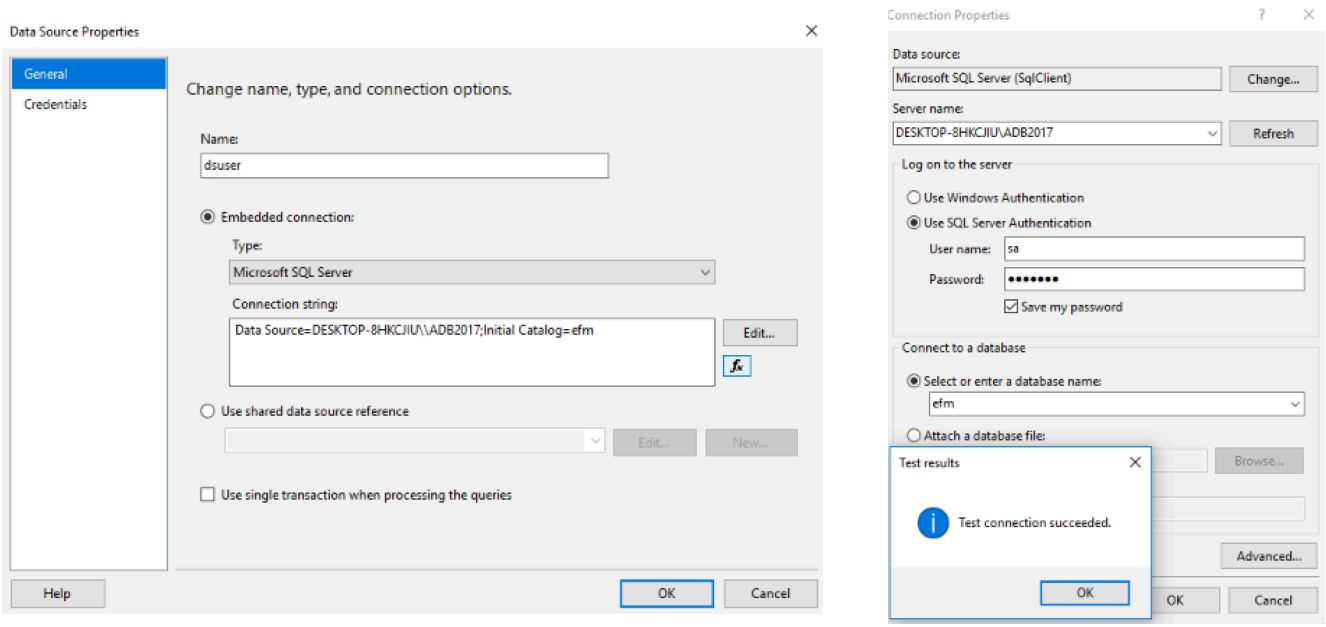


Figure 1.15 Creating embedded data source in user report

Next data set is created. The values has been keyed in as below and in the query section the the needed information about the users has been queried.

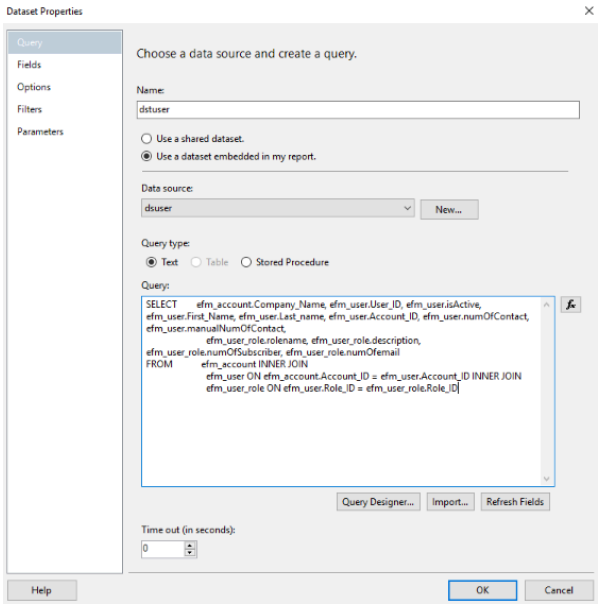


Figure 1.16 Data set Window in user report

Then, from the tool box pane, table dragged to designing window and and now all the tables that added in data set can be added in the table.

Lastly, the window can be customized and previewed and then saved.

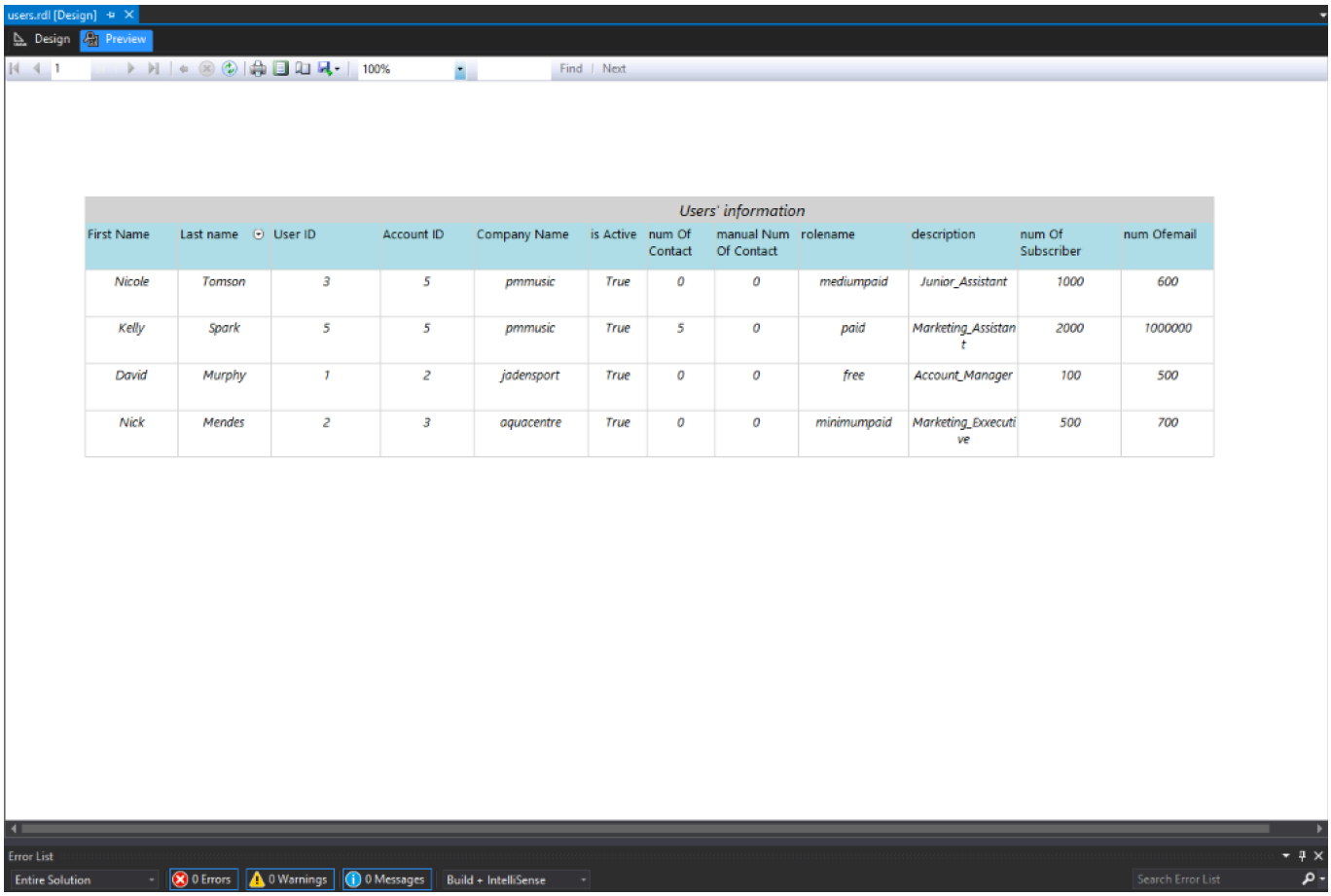


Figure 1.17 Users' information report layout

1.4.2.2 Reporting Campaign statistics

Most of the steps are like the previous report except for some parts.
The query part which contains more tables since campaign statistics are spread through different tables.
In campaign report the time that the campaign has created is shown by CONVERT system function in query statement for changing timestamp value to human readable datetime format. campaign reports.
Lastly, the time of the report is logged and can be seen at the top right hand side of the report.

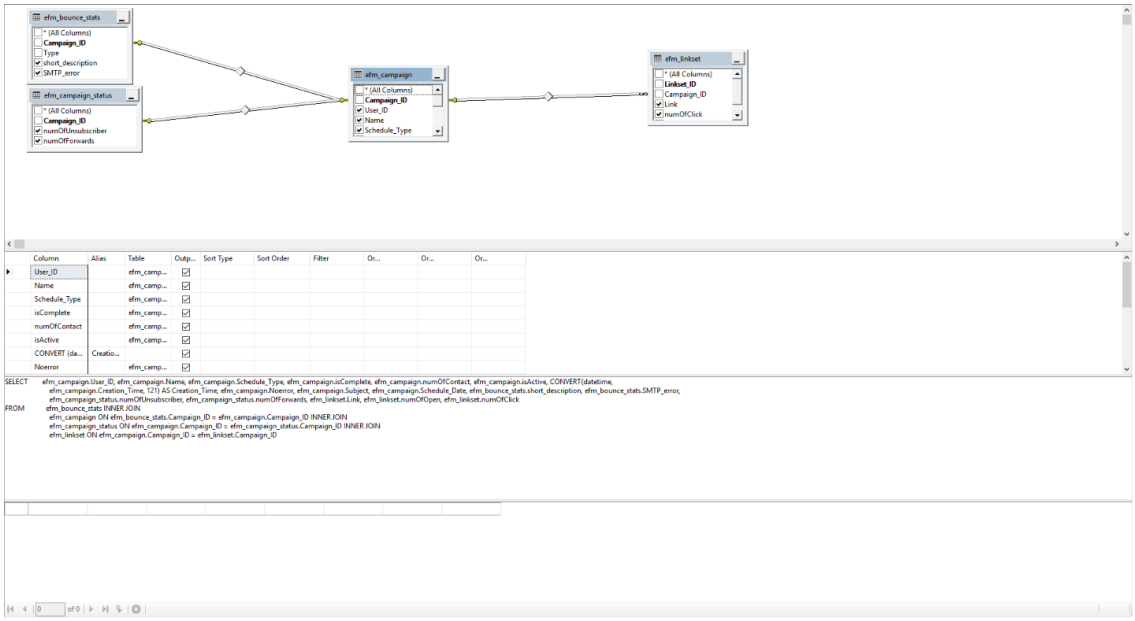


Figure 1.18 Designing the query for campaign statistics' dataset

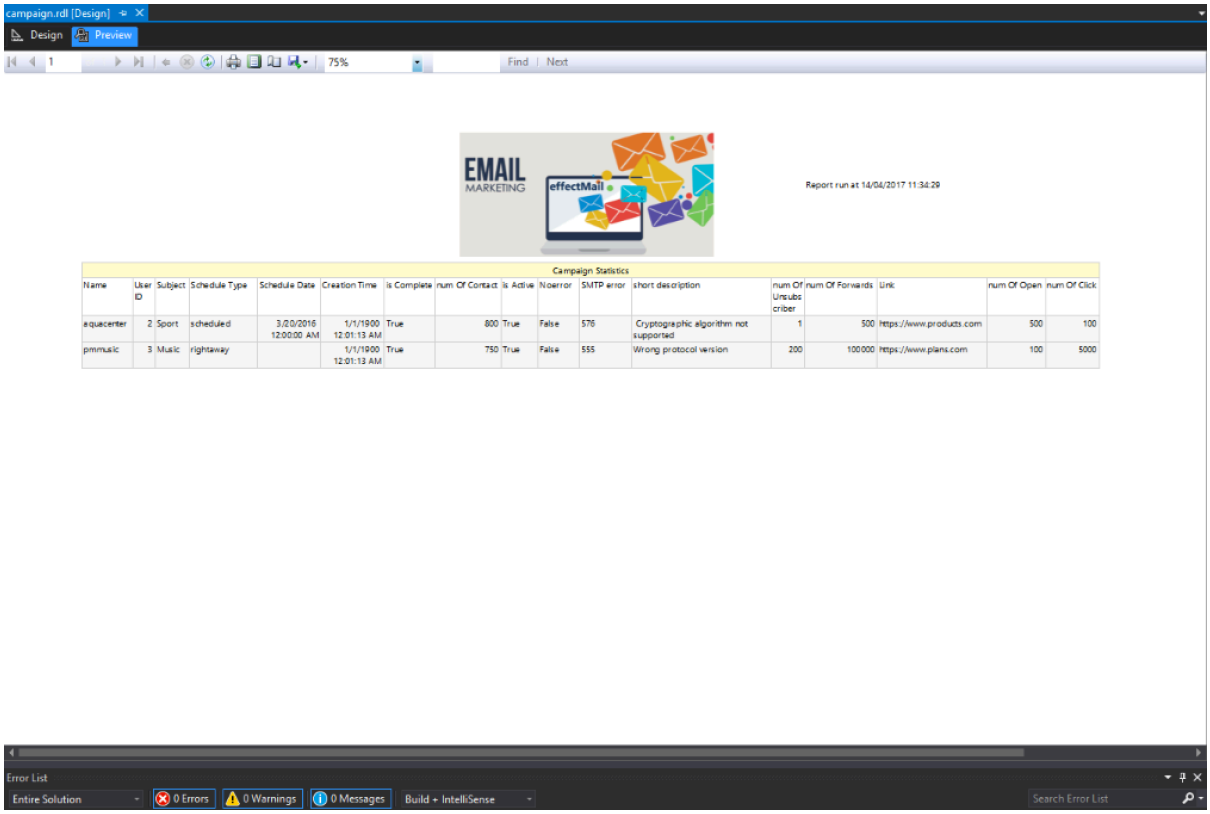


Figure 19 Campaign statistics

At the end, the report service should be established by running report service configuration manager and copying the report server URI like the following picture and pasting it into Target server URL of the project and then after running the report, it can be available on local reporting server and can be seen in the browser. (Although there are different ways of outputting it with SSIS in PDF or email or Sharepoint)

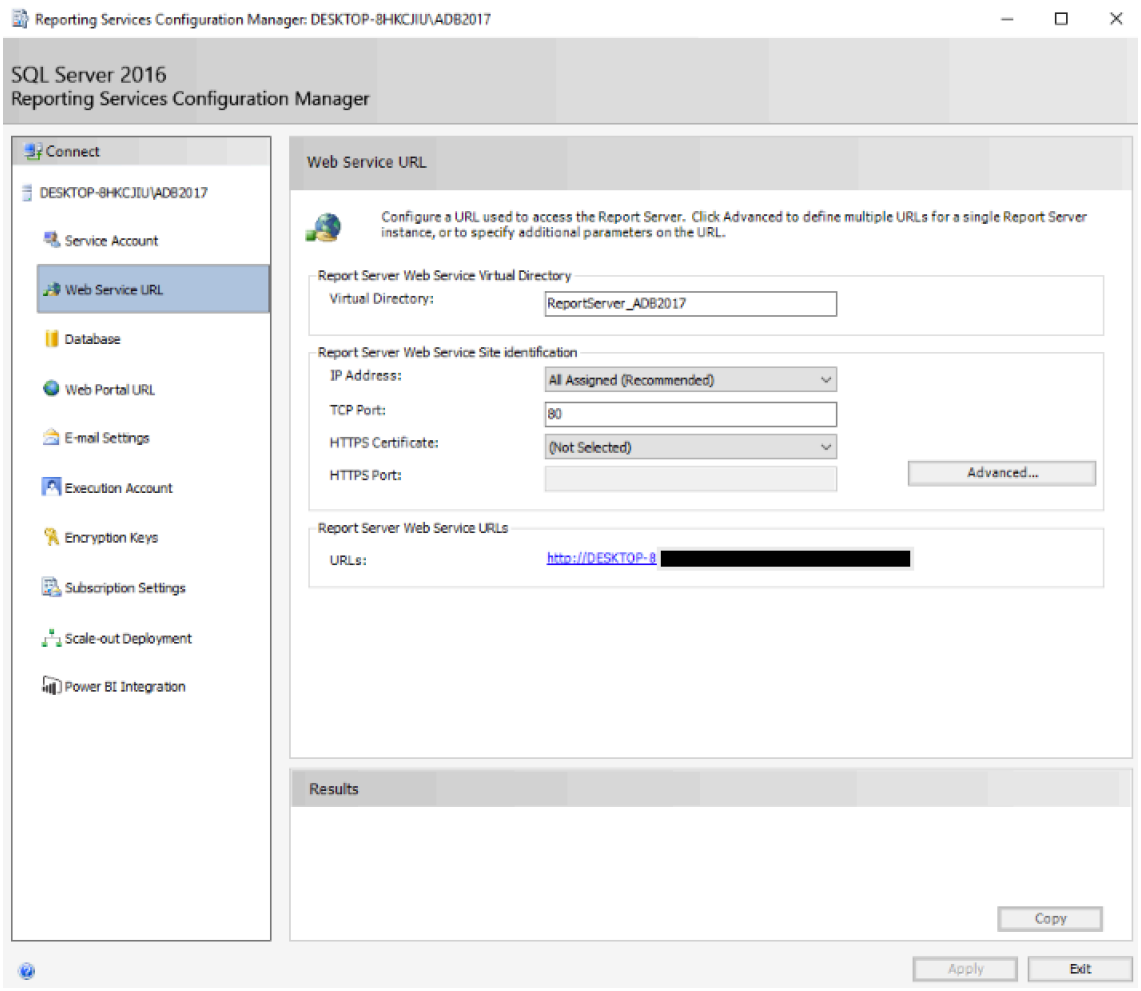


Figure 1.20 Reporting services configuration manager for running report server and its URL

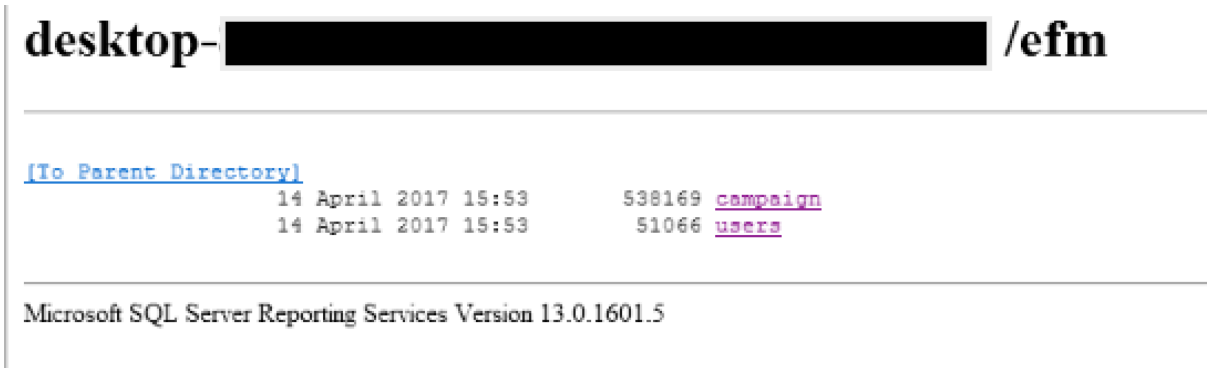
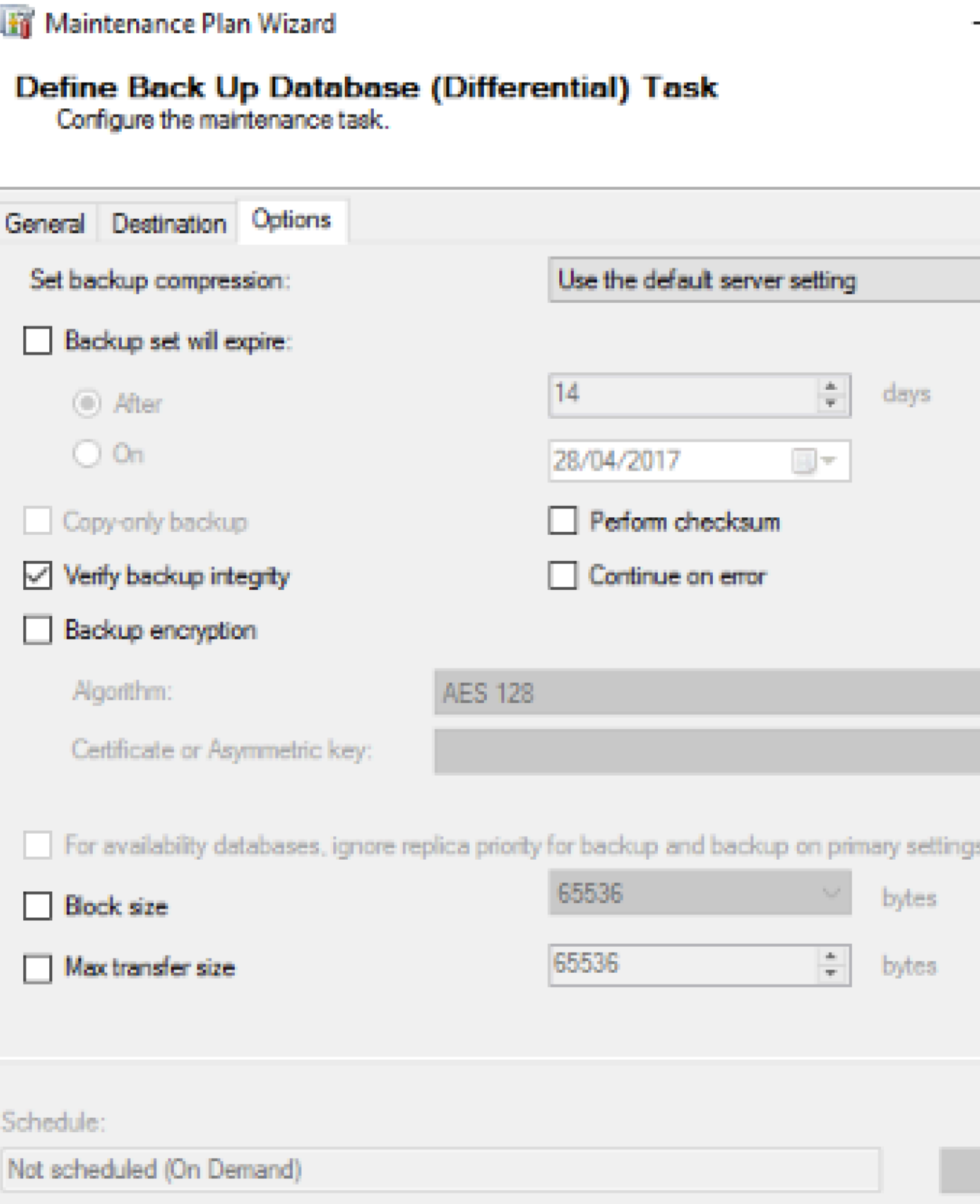


Figure 1.21 The output report result in the browser


Maintenance Plan Wizard
Define Back Up Database (Differential) Task
Configure the maintenance task.
General
Destination
Options
Set backup compression:
Use the default server setting
Backup set will expire:
After
14
days
On
28/04/2017
Copy-only backup
Perform checksum
Verify backup integrity
Continue on error
Backup encryption
Algorithm:
AES 128
Certificate or Asymmetric key:
For availability databases, ignore replica priority for backup and backup on primary settings
Block size
65536
bytes
Max transfer size
65536
bytes
Schedule:
Not scheduled (On Demand)

1.5 Backup

Efm takes backup and recovery seriously for availability purposes. When the designing of the project is finished, Full backup will performed and then for maintainability purposes it does a differential every 4 night at 10 PM and transactional backup every night at 10 pm. This jobs ensures that data is available through the database lifecycle.
Since creating both job are almost the same one of them (Differential backup) will be explained and difference will be explained in the Transaction log backup).
After opening Maintenance Plan Wizard in SSMS, the setting would be like this.

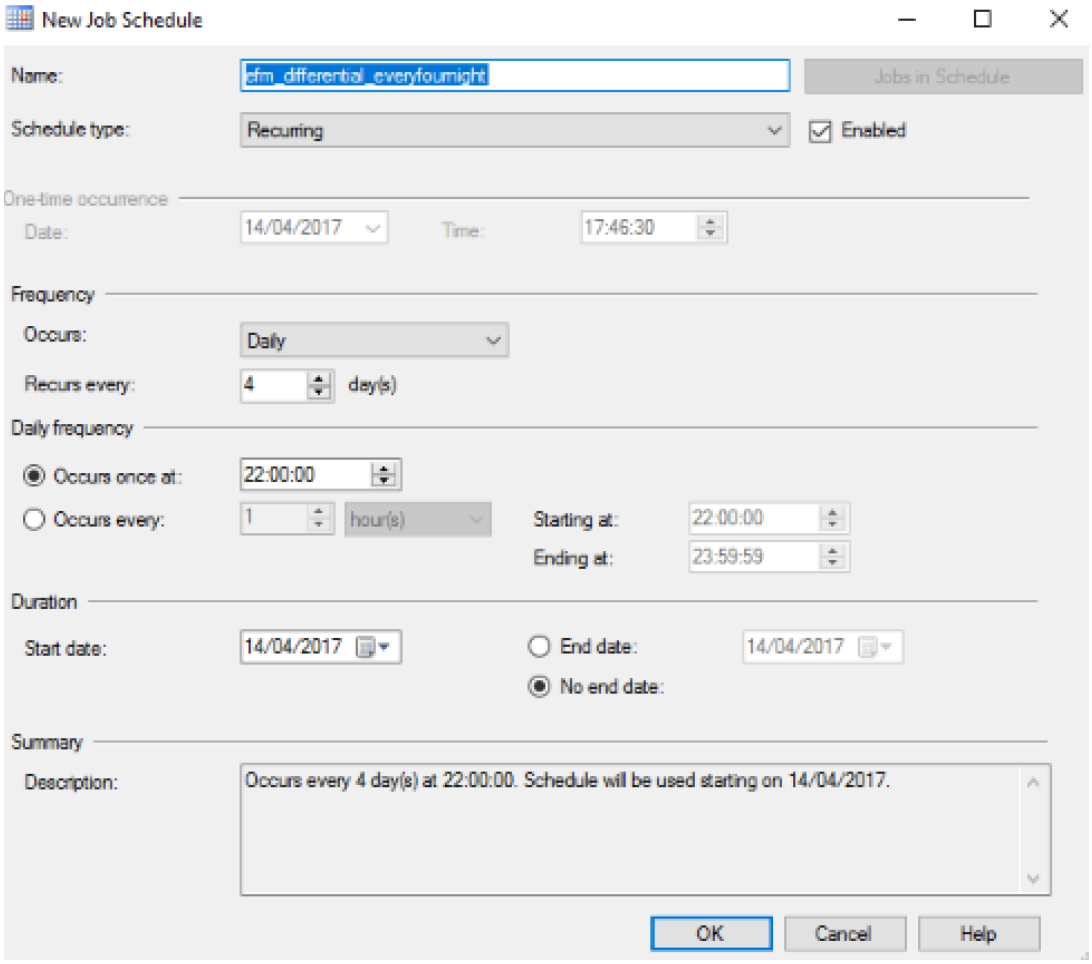


Figure 1.22 SSMS job Schedule for Differential Backup

Then differential backup and efm database will be selected and the setting will be as the following picture.

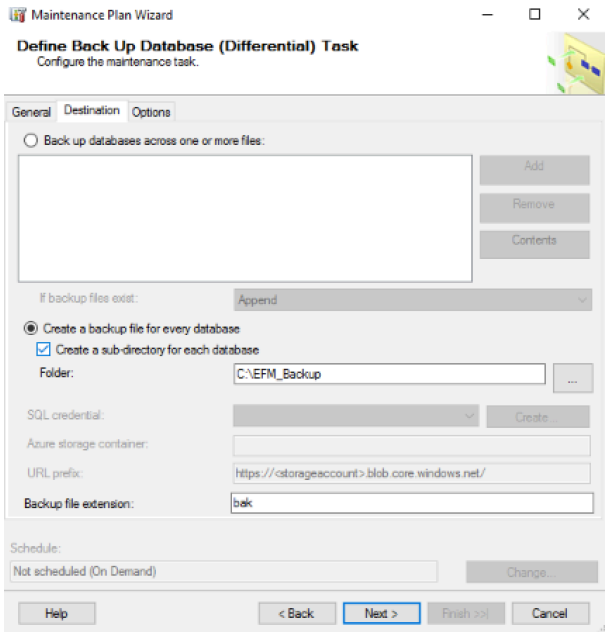


Figure 1.23 Different settings for Differential Backup

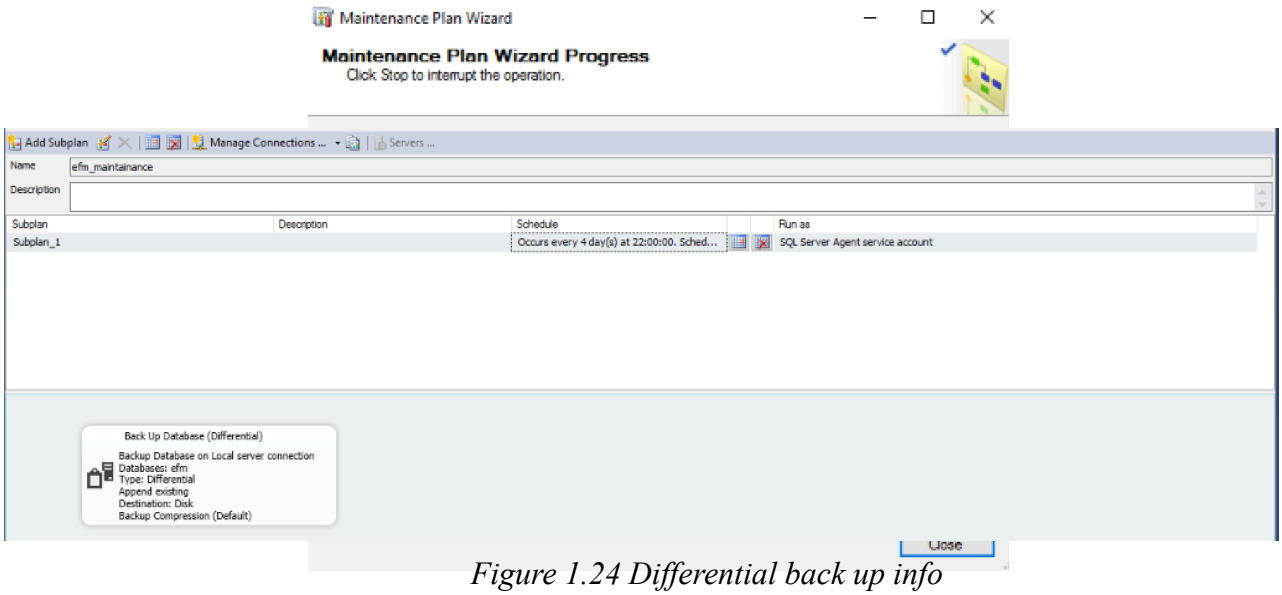


Figure 1.24 Differential back up info

The Backup job for transaction log would be the same except in the following settings.

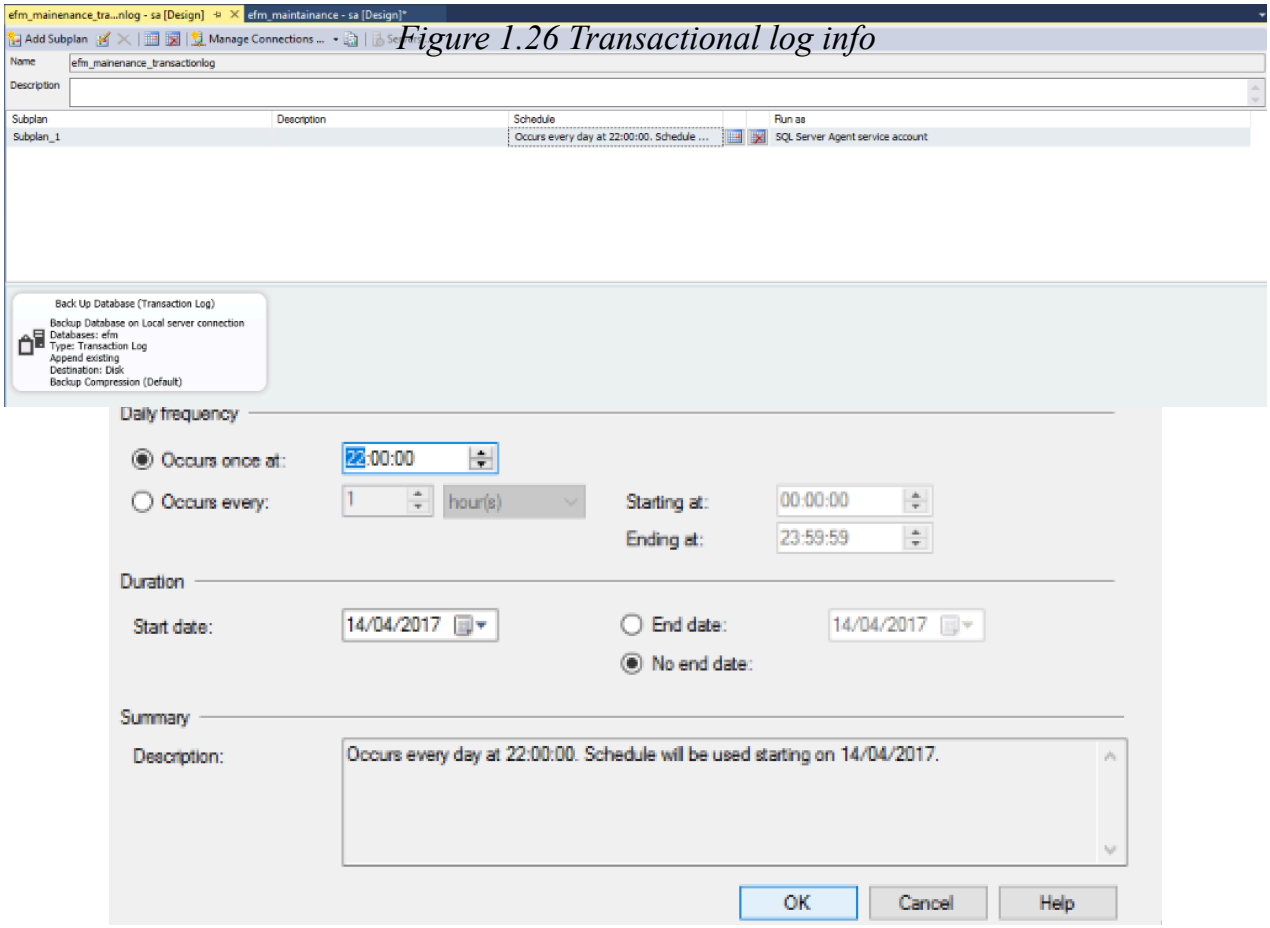


Figure 1.26 Transactional log info

It is
worth mentioning that there is another way to create scheduled jobs in SSMS by adding new job on object explorer and using this T-SQL statement to create first a full backup and then differential backup.

```
BACKUP DATABASE efm
   TO DISK = N'C:\EFM_Backup\efm.bak'
   WITH INIT,
NAME = N'efm full backup'

GO

BACKUP DATABASE efm
   TO DISK = N'C:\EFM_Backup\efm.bak'
   WITH DIFFERENTIAL,
NAME = N'efm differential backup'
GO
```

and for transaction log the statement would be :

```
BACKUP LOG efm

TO DISK = N'C:\EFM_Backup\efm.trn'

NAME = N'efm transaction log backup'

GO
```

and the other steps are almost the same as before.

1.6 Restore

Depending on the type of RPO[1] policy of a company different timing can be set for automated backup. In elm service as mentioned beside the full backup every 4 night there is a differential backup and every night there is a transactional backup. In the event of disaster in different scenarios we can restore these backups.
The process of restoring the database is almost the same for three type of backup files but it is important that a tail log backup should be performed before any other step and for restoring .trn file or transaction log file the database should be in restoring mode.

By right click on preferable database and choosing restore from task, restore window prompts. In this window we can choose the type of backup that are available or we can use new timeline feature in SQL Server 2012 and newer. Then for restoring a single file RESTORE WITH NORECOVERY should be selected in option tab and for restoring

[1] Recovery Point Objective

multiple backup files RESTORE WITH RECOVERY should be picked. To ensure that all the connection are closed in recovery time we can chose related option in option tab.
To ensure that the backup file has been restored successfully we can choose with checksum option and for ensuring that backup is not corrupted and able to be restored we can use RESTORE VERIFYONLY option.

References

Connolly, T., Carolyn, B. (2015). *Database systems A Practical Approach to Design, Implementation, and Management.* 6th edn. Edinburgh Gate, Harlow, Essex: Pearson Education Limited.

L.HARRINGTON, J.(2002). *Relational database design*. 2nd edn. London: Academic Press.

http://pixelmarketing.net (Accessed: April 2017).