BUILDING AGGREGATES IN THE DATA WAREHOUSE: A CASE STUDY OF BIRTH, DECEASED AND PROPERTY REGISTRATION E-GOVERNANCE DATA

Pushpal Desai

(M.Sc. (I.T.) Programme, VNSGU, Surat, India)

ABSTRACT

In this paper, the concept of aggregates in the data warehouse is discussed. The proposed method to create aggregate in data warehouse and its implementation using Microsoft SQL Server Integration Services is discussed. The results obtained from aggregates are presented. The results indicate that aggregates can be very efficient compare to querying data from base fact table of the data warehouse.

Keywords: Aggregates, Data Warehouse, Microsoft SQL Server Integration Service.

I. INTRODUCTION

An Aggregate is a supplemented data structure that helps make things go faster in the data warehouse [3]. Aggregates are very important part of any data warehouse implementation. An aggregate is a number that is calculated from amounts in many detail records. An aggregate is often the sum of many numbers, although it can also be derived using other arithmetic operations or even from a count of the number of items in a group [1]. An aggregate is a value formed by combining values from a given dimension or set of dimensions to create a single value [1]. By implementing aggregate in the data warehouse, we can store summarized data from the detailed data that are available in the OLTP systems. Once we create different aggregates in the data warehouse, retrieving information from the aggregate is much more efficient compare to detailed data [1]. There are several advantages of creating aggregates in data warehouse. Typically, Aggregates contains fewer rows than the base tables. Therefore, when end user executes query against the aggregate’s fact table instead of the data warehouse fact table, the response time is quite high. So, aggregates are very effective in improving query performance in data warehouse [2]. Typically, data warehouse contains large amount of data with millions of records. In data warehouse environment several users tries to executes complex queries from the data warehouse and that may take lot of time. The use of pre
calculated aggregates can greatly improve the query execution time and efficiency the data warehouse [4].

II. METHODOLOGY

The Aggregate transformation allows us to combine information from multiple records from the source data and convert into a single value [1].

![Flowchart showing the proposed methodology to create Aggregates]

To create aggregate, first we need to specify source data and then select the input columns from the source data. We need to specify operations on the input columns and the possible operations on input columns are “group by”, “minimum”, “maximum”, “sum”, “average”, “count”, “count distinct”, etc… After specify these settings, we can create aggregate in the data warehouse and store them for future analysis tasks by the management. The proposed methodology to create aggregates is depicted in the Figure 1. The aggregate transformations are implemented on different data by considering the common business requirements.

The SQL Server Integration Service provides aggregate transformation to develop various aggregates [1]. For example, In “Birth Data”, aggregate based on “RegistrationYear”, “ReligionID” and “Sex” fields was developed. Based on these fields, aggregate of “Average Birth Weight” was developed. The Figure 2 shows settings for aggregate transformation settings in the SQL Server Integration Services.
Similarly, aggregate for “Average Deceased Age” considering “Registration Year”, “Deceased Religion” and “Deceased Sex” fields was developed. The settings for “Deceased Age Aggregate” transformation are shown in the Figure 3.

Similarly, aggregates for “Property Database” considering average “Property Age” in various wards and property types was developed. The settings for the property age aggregate transformation are shown in the Figure 4.
III. RESULTS

The SQL Server Integration Services package execution on Birth Data source records generated “151” records. The execution flow and result is shown in the Figure 5 and Figure 6 respectively.

This aggregate summarized data for “Average Child Birth Weight” attribute. It considers various fields such as Gender, Year and Religion. Hence, whenever, Average Child Birth Weight data is required, query can be efficiently executed against aggregate. This query execution will very efficient as aggregate contains only 151 records and query execution does not affect base fact table.
Similarly, we executed SSIS package for creating “Average Deceased Age” attributed. The execution flow and its result are shown in the Figure 7 and Figure 8 respectively.

![Figure 6: Result of Child’s Birth Weight Aggregate transformation](image)

This aggregate considers other fields such as Gender, Religion and Year. This aggregate can be efficiently used, whenever; Average Decease Age information is required.

![Figure 7: Execution of Deceased Age Aggregate transformation](image)
The execution of SSIS package for “Average Property Age” resulted in “768” rows from 1,47,1859 records stored in base fact table.

This aggregate contains other important fields such as Property Type and Ward Number. So this aggregate can be very efficiently used whenever “Average Property Age” is required as query execution will be against only 768 records.
IV. CONCLUSION

The results clearly indicate that aggregates are crucial part of any data warehouse deployment. The deployment and use of aggregates greatly improves the efficiency of data warehouse queries. The practical implementation indicates that queries executed against aggregates are highly efficient because aggregates contain far less records compare to base fact tables.

V. ACKNOWLEDGEMENT AND LIMITATIONS

All results are based on data provided by the municipal corporation for the research purpose only. Hence results may change, if data warehouse concepts are applied on actual data sets.

VI. REFERENCES

(1) Brion Larson, Delivering Business Intelligence with Microsoft SQL Server 2008.