

Group 3

Abigail Green

William Gray

Betapho Hannah

Melissa Hunfalvay

Assignment 6.1 ETL Team project

Memorandum

## MEMORANDUM

TO: Ms. Boss

FROM: Analytics Team 3

DATE: February 20<sup>th</sup>, 2021

The purpose of this memorandum is to document the process for merging and transforming the last three years of company data into a new data mart to facilitate analysis and output. Furthermore, to answer some questions related to the data and future examination and output of the data.

### The Entity Relationship Diagram (ERD)

To understand the data, an Entity Relationship Diagram (ERD) was created (see Figure 1). The purpose of the ERD is to provide the highest level of abstraction for the data model (Inmon & Lindstedt, 2015). In other words, the data is visualized and explained based on its relationship between the variables (or entities).

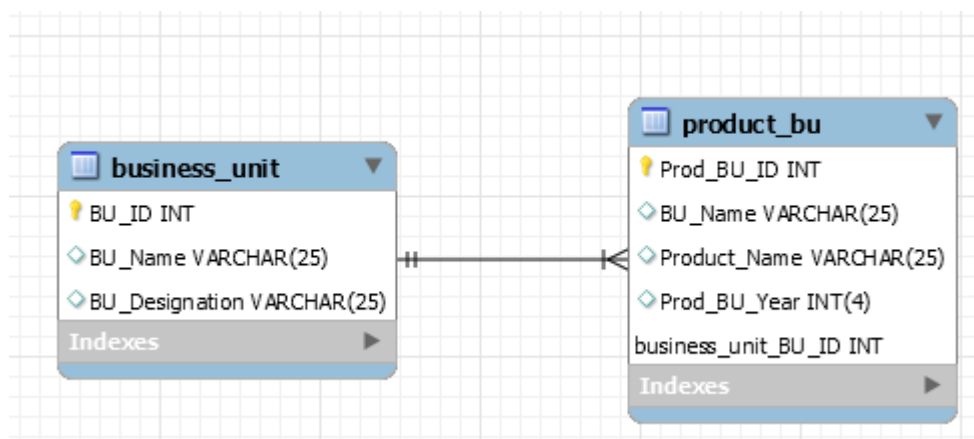


Figure 1: Entity Relationship Diagram (ERD) for Business Unit and Product Business

In the ERD are various structures. At the highest level is an 'Entity' defined as a board classification of data (Inmon & Lindstedt, 2015). The entities are business\_unit and product-bu.

Within each entity are attributes. An attribute is defined as a value of data that is distinguishable from other values (Inmon & Lindstedt, 2015). Attributes have their own unique data that are grouped together, for example, business name is an attribute (see Appendix A for more technical information).

Another unique feature of an ERD is a primary key. Primary keys are identifying attributes of data and enforce entity integrity by uniquely identifying entity instances (Condor, 2020). In the ERD, the primary key in the product\_bu is the Prod\_BU\_ID for example.

Another important feature of the ERD are foreign keys. Foreign keys enforce referential integrity by completing an association between two entities (Condor, 2020). In other words, for the tables (entities) to 'talk' to one another they need keys that open the door to the unique data inside each bucket or table. As we have two tables, we want them to work together, and a foreign key is used to link them for data from one table to be joined with data from the other. A foreign key in our ERD is business\_unit\_BU\_ID. Primary and foreign keys are the most basic components on which relational database theory is based (Condor, 2020).

The ERD describes the relationship between the entities and shows that for every product and year combination there must be one and only one business unit. Furthermore, every business unit must have at least one product. This relationship is shown by the line between the entities. A solid line represents a conceptual one relationship between the entities (Soper, 2013). The second symbol that looks like a crow's foot and extends to the product\_bu entity represents the conceptual notion of many (Soper, 2013). The double lines, known as hash lines, represent a

one-and-only one relationship (Soper, 2013). Therefore, we can interpret the relationship between the entities to be that for every product and year combination there must be one-and-only-one business unit and every business unit must have *at least one* product.

### **Extract. Transport. Load.**

Product related data, in the form of excel files, for the years 2012, 2013 and 2014 were used to create the ERD. Three steps were taken to transport the data and provide output. This process is visualized in Figure 2.

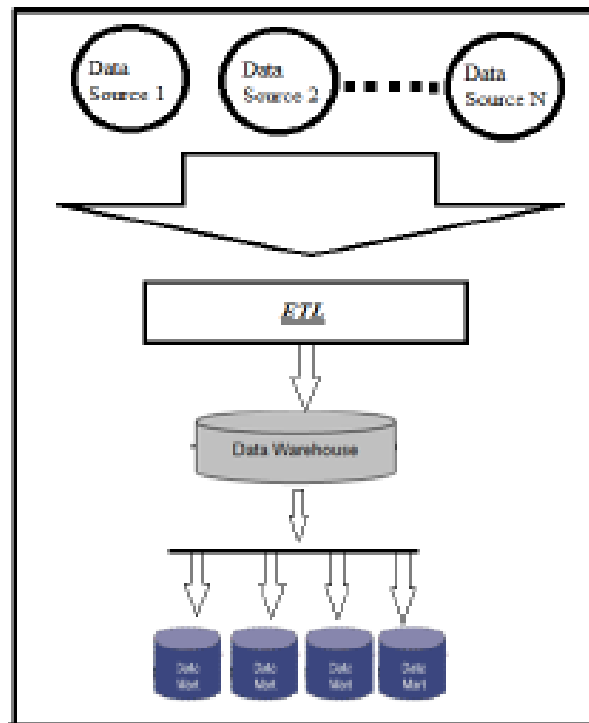


Figure 2: The Extract, Transport, Load (ETL) process. Image from Titirisca (2013)

This process is known as Extract, Transport and Load (ETL; Tritirisca, 2013, see Appendix B for technical details). During this process data is extracted from sources, in this case several excel spreadsheets, then brought into the data warehouse (Oracle, 2021). Once in the data

warehouse, an entity relationship diagram is created in order to explain a logical description of how the major subject areas fit together (Inmon & Lindstedt, 2015).

The first step is to extract all data from the three excel spreadsheets (see specific technical steps in Appendix B). Data was modified from the excel spreadsheets to csv files in preparation for step two.

In the second step and third steps the data is transported to My SQL workbench via the Table Data Import Wizard, then loaded. Data must be physically transported to the target system for further processing (Oracle, 2021).

Several business-related decisions were made during the Transport phase. For example, records with 0 quantity and/or 0 order total in the data were kept because as a business it is important to distinguish between years or months when a product was not offered and years or months where a product was not sold. Another note of importance is that integers in the final product do not reflect the actual Order Total. For instance, 466 is equivalent to \$4.66.

The ETL process can be a very time-consuming part of an analyst job (Inmon & Lindstedt, 2015). It can also be error prone with large data sets, which is in part why ETL tools are used for data warehouses (Oracle, 2021). The current data set, however, was manageable but these issues should be considered should our data become either larger, or more complex in the future.

## **Metadata**

To complete documentation and assist analysts using this work in the future we have included the meta data. Metadata is “data about data” (Inmon & Lindstedt, 2015). Metadata is

the descriptive data that defines important aspects of the database. Below is a description of the metadata in this database. This information is formatted as described by Inmon and Lindstedt (2015, p.190-191).

Table: business\_unit defined as the business characteristics associated with the business.

Relationship name: Every business unit must have at least one product. This is a 'must have' relationship with the product table. It is also a one-to-many relationship with products, in other words the business can have more than one product.

Attribute: BU\_ID

Description: Identification of the business unit

Characteristic: Integer

Key: Primary key

Attribute: BU\_Name

Description: The business name

Characteristic: VARCHAR (25)

Key: No

Attribute: BU\_Designation

Description: Umbrella designation used by the company to determine marketing strategy

Characteristic:        VARCHAR (25)

Key: No

Table: product\_bu defined as the characteristics associated with the product

Relationship name: Every product and year must have one and only one business unit. This is a ‘must have’ relationship with the business table. It is also a one-to-one relationship with business, in other words the product and year combination can have only one business unit association.

Attribute:        Prod\_BU\_ID

Description:    Identification of the product

Characteristic:        Integer

Key: Primary key

Attribute:        BU\_Name

Description:    Name of the business

Characteristic:        VARCHAR (25)

Key: No

Attribute:        Product\_Name

Description:    Name of the product

Characteristic: VARCHAR (25)

Key: No

Attribute: Product\_BU\_Year

Description: Year of the product

Characteristic: Integer (4)

Key: No

Attribute: business\_unit\_BU\_ID

Description: Identification of the business unit

Characteristic: Integer

Key: Foreign

### **Data Management**

Now we turn our attention to specific questions related to this data mart. Specifically, it was asked if this is the correct level of granularity for our data mart? This is a good start; however, we believe we can improve on the level of granularity in ways that will help our business. Our current data mart is focused on the total number of products ordered by year and whether those products are in a state of maturity or growth. In order to improve our data mart, we need a finer level of granularity which will allow more flexibility for analysis moving forward. To improve the granularity of the database we would need to ensure that we have State data for all prior and future years. Granularity refers to how detailed each individual datum is (Inmon &



Linstedt, 2015). Granularity is a business decision, and we recommend a finer level of granularity for our business because having a finer level of granularity in our database will lead to more informed decisions. Instead of making decisions on whether a product is in growth or maturity across the country or by region we could drill down to the state level. This will give us better insights into how certain products are performing and we could create a more localized plan.

We would also recommend keeping the per unit price in our data mart as well as the state. Keeping the per unit price in the data mart allows us to easily determine the best price point for each product. For example, if we sold 200 orange creepies in the midwest region in 2012, but then we increased the price by \$5 per unit in 2013 and saw a drop in quantity sold, we would be able to reevaluate our pricing structure for orange creepies. With only the order total stored in the data mart, we would have to divide the order total by the total quantity each time to get the per unit price before we could start a pricing analysis.

Furthermore, there are some inconsistencies in the historical data of our data mart. This means our data integrity is questionable. Data integrity refers to the completeness, accuracy, and reliability of all our data (Naeem, 2020). Data inconsistencies are generally problematic because it causes instability and makes querying more difficult (Inmon, 2002). While some years have data for month, region, and state other years do not. If we had consistent data across years, we could make a better assessment of which products are performing the best all the way down to the state level.

Additionally, missing data across years signals that the quality of data is questionable. According to The Global Data Management Community data quality refers to the planning,

implementation, and control of activities that apply quality management techniques to data, to assure it is fit for consumption and meets the needs of data consumers” (Knight, 2017). There are many qualities that characterize data quality, however, one that is consistent and features prominently is accuracy of the data (Wang & Strong, 1996).

To correct these issues and improve our database moving forward, we recommend creating data standards and processes that ensure a high quality of data. The GRAnD approach to data quality is one such example (Giorgini, Rizzi, Garzetti, 2005). GRAnD refers to a Goal-Oriented Approach to Requirement Analysis in Data Warehouses. According to Abai, Yahaya, Deraman (2013) the GRAnD approach can lead to high quality of data warehouse schema and high probability of fulfilling business goals.

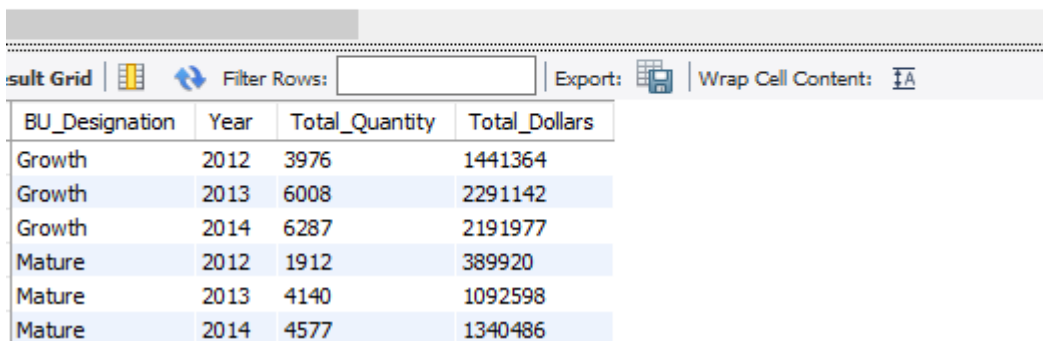
The business benefits of better data quality, data accuracy, and standard processes for our data mart are that we can trust we will be able to better assess marketing efforts across states and improve analysis of product growth or maturity.

Perhaps more importantly, we do not want to come to false conclusions based on bad data. Continuing the same path with our current data mart could lead to false assumptions that guide us in the wrong direction and hurt overall business objectives. For example, a product could be growing in a specific state but maturing for the region. If we had state level data for every year, we could better assess the performance on the state level and create a more localized marketing plan. Additionally, if we had per unit price information for each product, it would allow us to make better pricing decisions. We recommend a finer level of granularity for the data mart, keeping both the State and Per Unit Price fields after accounting for data quality issues.

## Data Analysis

A specific request was made to answer this business question: “We believe our Growth segment should show at least 10% year over year growth in either quantity sold or order total. We also believe our Mature segment should remain pretty much the same in terms of quantity and order totals. If you give the final data file produced to Ramon can he run queries to answer this?” Yes, Ramon can run queries to obtain these insights and to illustrate that point, we have run those queries. First, we used the final select statement which we used to generate our csv file and inserted the results into a new SQL table called g3\_final\_output. Then grouped the data by BU\_Designation and Year and calculated the total sum of orders and the total dollar amount for each designation and year. The code and the resulting output are pictured below:

```
1 • SELECT BU_Designation, Year, SUM(Sum_of_Quantity) as Total_Quantity,  
2     SUM(Sum_of_Order_Total) as Total_Dollars  
3     FROM g3_final_output  
4     GROUP BY BU_Designation, Year;
```



BU_Designation	Year	Total_Quantity	Total_Dollars
Growth	2012	3976	1441364
Growth	2013	6008	2291142
Growth	2014	6287	2191977
Mature	2012	1912	389920
Mature	2013	4140	1092598
Mature	2014	4577	1340486

Figure 5: MySQL code run on the final data file and its output to help answer Ramon’s questions

After running the SQL query, Ramon has the totals he needs, but because each year’s total quantity and dollars are not in their own columns, Ramon will have to go one step further and calculate the year-over-year growth for quantity and dollars by hand. For only three years of

data, this may be the easiest option for Ramon. He can take the total quantity from 2013, subtract the total quantity from 2012 and then divide by the total quantity from 2012 to get the year-over-year growth in quantity from 2012 to 2013. This same equation would apply for 2013 to 2014 and for growth in total dollars. Ramon would find that for growth products, quantity and order total increased by over 50% from 2012 to 2013 but increased by 4% and decreased by 4% respectively from 2013 to 2014. Mature products on the other hand experienced over 100% growth from 2012 to 2013 in both quantity and order total, and experienced 11% growth in quantity and 23% growth in order total from 2013 to 2014.

The data being structured this way is conducive for Ramon to filter things out or group the data in a different way in this sort of analysis. If he decides he wants to take out the month of January or change his mind to show growth month-over-month for example, that's easy to do with a "tall" data structure like the one we gave him. However, if Ramon needs to calculate year-over-year percentages easily and without using multiple steps, a "wide" data structure would be more suitable. In this case, it would look like having columns for each year's quantity total and order total. With those columns, Ramon could run the query in one step by grouping on business unit designation and then creating columns for year-over-year growth from 2012 to 2013 and from 2013 to 2014 right in SQL.

### **Using a different ETL structure?**

Finally, regarding your question on the Extract, Transform and Load (ETL) process. You mentioned that our database folks have suggested we use a different format for the ETL. Data layout has a significant impact on overall business performance, which is why it is very important to choose the right data layout that will meet our needs. Susie's concerns regarding the

layout that has columns for each year's data are valid and we would recommend going with the existing layout.

Having a column in the data mart for each year's quantity and order total will make it easier to answer the question of growth year-over-year. Answering the growth question is still possible with our data mart structured as is, it just requires an additional step as explained in Ramon's analysis. You could take the results obtained in Figure 5, export them to Excel, and then calculate year-over-year percentages of growth with the existing data mart structure. Another option would be to take the results of Figure 5, insert them into a new table, and reshape that data from "tall" to "wide". In the structure proposed by Bobby, you could easily create columns that will show percentage year-over-year growth directly in SQL. However, the structure Bobby supports limits us in two very important ways that will not allow us to be agile moving forward.

One of the ways that the proposed data mart structure limits us is that it makes it harder to do any other type of analysis, which is why Susie is hesitant to change the existing structure. Right now, the primary business objective may be to grow our business from year to year, but at some point, we may want to do a head-to-head analysis of two different products for example. If we want to know whether we've sold more Purple Pain or Red Hot Chili Peppers in the Southeast region, we'll have to add up the quantity columns for each year. An even more complicated example would be if we wanted to do a monthly analysis where we compared growth from January to February across all years. This would be a challenge with columns for each year but would be much more straightforward with the existing data structure. The existing format also provides more agility for filtering and grouping the data.

The existing data mart structure allows us to filter 2012 and 2014 data out of our analysis if we want to do a deep dive of 2013 sales. Additionally, having each year's numbers as their own columns will introduce Null values into the dataset. For example, Green Lightning was part of the Sugar business unit in 2012, but part of the Lunchtime business unit in the years thereafter. In the proposed data mart structure, we will have rows for Green Lightning in the Sugar business unit for each month in 2012 that have values populated for the Sum of Quantity for 2012 and Sum of Order Total for 2012 columns and Null values for the other years' columns and vice versa, which would make calculations difficult if we just wanted to know about the growth of Green Lightning from year to year regardless of which business unit it was in. The most significant way in which the new proposed structure limits us in analysis moving forward is with unwieldy column creation.

Currently, we have three years' worth of data in our data mart so creating six columns for each of the three years' order total and quantity total is reasonable. However, as we continue to grow as a company and continue our business for years to come, we will continue to add years' worth of data to our data mart. This will require that we add columns to our data mart every time we add more data. In addition, the columns for each year are structured as a summary once that year is closed. If we are adding data to our data mart in real time month by month, we will have to recalculate the 2015 order total column and 2015 quantity total column each time we add a new month's data. The current data mart is set up in a way that will allow us to add a row for each Product, Region, Year, and Month as the data is made available to us.

Our recommendation is to keep the data mart structured the way it is because the additional granularity at the row level allows for better filtering and grouping, more agility to do different analyses and change our KPIs, and it will allow us to keep our data more current. If we

must wait until an entire year's worth of sales are complete before we can add them to our data mart, that could be detrimental to us. If we see sales information in the data mart at the end of each month, we can make mid-year adjustments to pricing and marketing that will allow us to respond to market challenges.

## References

- Abai, N.H.Z, Yahaya, J. H. Deraman, A. (2013). User requirement analysis in data warehouse design: A review. *Procedia Technology*, 11, p.801-806.
- Condor, D.P. (2020). Database Design. Primary and Foreign Keys. Retrieved from:  
<https://condor.depaul.edu/gandrus/240IT/accesspages/primary-foreign-keys.htm>
- Giorgini, P. Rizzi, S. and Garzetti, M. (2005). GRAnD: Oriented Approach to Requirement Analysis in Data Warehouses. *DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, p. 1–31.
- Inmon, W.H. (2002). *Building the Data Warehouse*, 3rd Ed. John Wiley & Sons, New York, NY.
- Inmon, W.H. & Lindstedt, D. (2015). *Data Architecture: A Primer for the Data Scientist*. Elsevier, Morgan Kauffman. New York, NY.
- Knight, M. (2017). What is data quality? Dataversity: <https://www.dataversity.net/what-is-data-quality/>
- Naeem, T. (2020, November 30). *Data Integrity in a Database - Why Is It Important*. Astera.  
<https://www.astera.com/type/blog/data-integrity-in-a-database/>
- Oracle Corporation (2021). Overview of ETL in Data Warehouses. Retrieved from:  
[https://docs.oracle.com/cd/B19306\\_01/server.102/b14223/etlover.htm](https://docs.oracle.com/cd/B19306_01/server.102/b14223/etlover.htm)
- Soper, D. (2013). Database lesson #4 of 8 - *Data Modeling and the ER Model*. [Video]. YouTube.  
<https://www.youtube.com/watch?v=IfaqkiHpIjo>



Titirisca, A. (2013). ETL as a necessity for business architectures. *Database Systems Journal* IV(2), p. 3-14.

Wang, R.Y. & Strong, D.M. (1996). Beyond accuracy: What data quality means to data customers. *Journal of Management Information Systems*, 12,(4), pp.5-33.

## **Appendix A**

### **Entity Relationship Diagram**

Appendix A provides some more technical information on the ERD.

Within the table (or entities) of the business unit there are three attributes (BU\_ID, BU\_Name, BU\_Designation). The BU-ID is the primary key, which is used within the entity of the business\_unit that is an attribute which uniquely identifies the entity (business\_unit, Inmon & Linstedt, 2015).

There are five attributes in the product\_bu entity. The attributes are Prod\_BU\_ID, B-Name, Product\_name, Prod\_BU\_Year and business\_unit\_BU\_ID. The primary key is Prod\_BU\_ID.

## Appendix B

### Extract. Transport. Load.

The following notes are technical steps used in the ETL process for this data.

First, the excel files needed to be changed into a format that allows for upload into Structured Query Language (SQL). One example of this was to change the 2012\_product\_data\_students.csv: Per-Unit Price and Order Total to Per\_Unit\_Price and Order\_Total

To extract the data, we used MySQL Workbench and clicked on Table Data Import Wizard. We navigated to the files on our computer and clicked next. On the Select Destination page we selected “Create new table”, renamed the table to 2012\_product\_data and clicked Next. On the Configure Import Settings page, we kept all the defaults with Month, Per-Unit Price, Quantity, and Order Total being read in as Ints and Country, Region, State, and Product being read in as Text then clicked Next. We clicked Next on the Import Data page, clicked Next again, and Finish. We verified the upload was correctly received as we saw that 500 records were imported (see Figure 3).

#### Import Results

File C:\Users\agreen169\Desktop\DATA620\Week6\2012\_product\_data\_students.csv was imported in 1225.735 s  
Table business.2012\_product\_data was created  
500 records imported

Figure 3: Screenshot of Table Data Import Wizard in MySQL Workbench which shows that 2012\_product\_data was imported successfully with 500 records

This process was then repeated for the 2013 and 2014 product data sets. A SQL script called G3\_2\_create\_product\_data\_table.sql to extract all rows, but only columns necessary for the final csv from each of the three 201x\_product\_data files and union them into one product\_data table. This completed the Extract portion of the ETL process.

Once in the correct format, and the files were extracted (uploaded) to SQL they were then unioned (or joined, step 2) into one table. During this second step, only the necessary columns were extracted for the final exported data file.

Finally, the data was prepared for Load or exporting. To do this another SQL script was written called G3\_3\_create\_output\_file.sql. This script collects in all the fields for the output file, does a group by. The script sums the quantity and order totals and puts commas between all the fields to facilitate a csv export.

After running the third script, the command line was opened and a series of commands to export to a csv file were performed. First, a change directory was used (C:\Program Files\MySQL\MySQL Workbench 8.0 CE), this enabled navigation to the MySQL working directory. Then the SQL command with credentials to export the file was input into the command lines so that a csv file could be output (see Figure 4). When prompted a password was entered and the data was exported into one csv file.

```
C:\Program Files\MySQL\MySQL Workbench 8.0 CE>mysql.exe -h data620-9041-agreen169.mysql.database.azure.com -P 3306 -u ad  
minuser@data620-9041-agreen169 -p --batch <C:\Users\agreen169\Desktop\DATA620\Week6\G3_3_create_output_file.sql> C:\User  
s\agreen169\Desktop\G3_output_final.csv  
Enter password: *****
```

Figure 4: Command line code used to export the data file