**ADVANDB MCO2: Data Warehousing and Online Analytical Processing**

Amadora, Angelo John

De La Salle University, Manila

26 MJS Ave., Levitown Executive, Better Living Subdivision, Paranaque

(+63) 9178865800

angelo\_amadora@dlsu.edu.ph

Aquino, Kurt Neil

De La Salle University, Manila

254 Sitio Bagong Anyo, Pagsawitan, Santa Cruz, Laguna

(+63) 9273597974

kurt\_aquino@dlsu.edu.ph

Choy, Matthew Seaver

De La Salle University, Manila

109 Speaker Perez St.,

Quezon City

(+63) 9271273470

matthew\_choy@dlsu.edu.ph

1. **INTRODUCTION**

For second Major Course Output (MCO2) of the course ADVANDB, students will again use the CBMS dataset from the combined provinces of Palawan and Marinduque in order to design a dimensionality model for a data warehouse; to extract, transform and load the data from this dataset to the student’s newly created models; and finally, to formulate queries showing roll-up, drill-down, dice and slice operations on said models.

For this project, the group used the CBMS dataset and extracted specific data from it into a new model, which supports online analytical processing (OLAP). The group also developed a simple java application which allows the user to extract data from the created model as well as demonstrate various OLAP queries. The application focuses on analyzing and displaying the average household construction materials as well as other household factors such as water source, electricity status, house type, land ownership, etc. of households affected by natural disasters (e.g. storms, floods, earthquackes, etc.) per area/location. Other than household factors, the application also allows the user to extract detailed information about the status of households affected by calamities with regards to the frequency of the occurrence of certain calamities in an area as well as the average percentage of aids being sent per occurrence. Although not present in the application, the model can support even more in depth analysis with regards to the reasoning behind the various factors a household has (ex: Does a household with OFW members have a higher chance of having better household construction materials/factors, etc.) which was already covered in the group’s previous application created for the first Major Course Output (MCO1).

The data model used in this project can be used by various local government or non-government organizations focusing on disaster preparedness, awareness, as well as response.

1. **DIMENSIONAL MODEL**

The schema that has been created for this project mainly deals with location details, household factors, and calamity details as the primary facts; other household member details such as job status, job type, agricultural livelihood status, and worker type as additional constrraints (not entirely necessary for the general purpose of the application, but may serve as additional factors to be considered for in depth analysis). From the CBMS database, the tables: hpq\_hh, hpq\_mem, hpq\_crop, and hpq\_aquani, were used and transformed into the group’s constellation schema, with hpq\_hh holding the most facts. The schema model can be seen in Figure 1.

* 1. **Fact Tables**

There are three fact tables in the schema: fact\_household, fact\_calamity, and fact\_member.

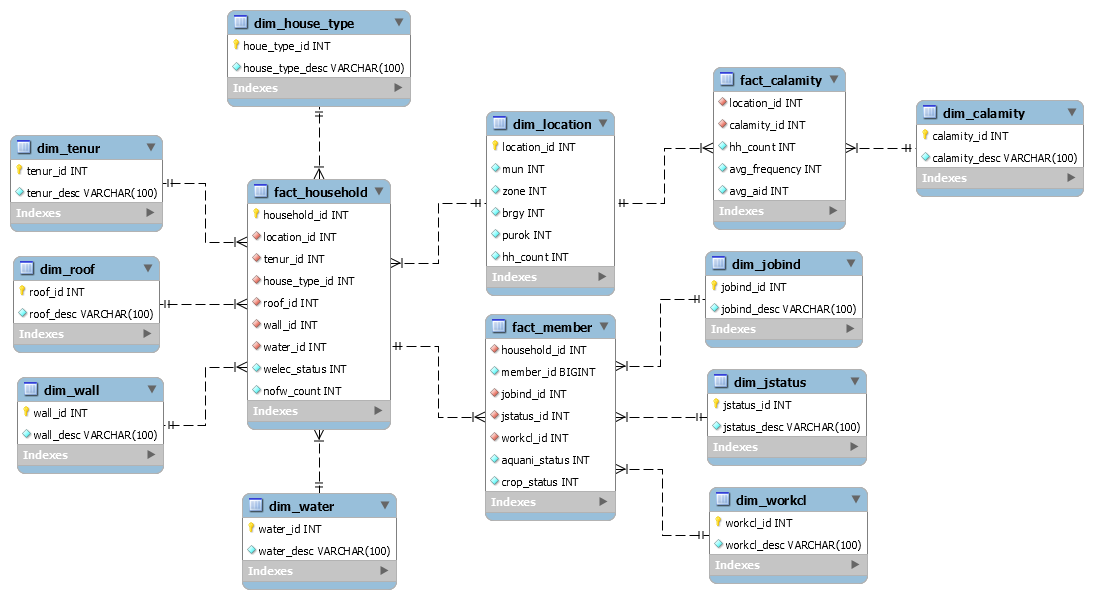
The fact\_household table contains the details of a household such as the household’s location, house type, tenur type, roof and wall construction material, water source, electricity status, and the number of OFW members.

The fact\_calamity table contains the details of calamities that have occurred in certain areas such as the calamity type, the location where the calamity occurred, the number of households who were affected by the calamity, and the percentage of how often aid arrives when a specific calamity occurs.

The fact\_member tables contains the details of members from a certain household such as the household where the member belongs, the member’s job status, job type, worker status, and aquani and crop status in order to determine if the member focuses on an agricultural livelihood or not.

* 1. **Dimension Tables**

The dimensions used for the fact tables in the schema were derived from the columns from the selected tables in the CBMS database. The details regarding some of the dimension tables were taken from the CBMS database’s provided Data Dictionary.

For the fact\_household table, the dimensions were extracted from the columns of the hpq\_hh table from the CBMS data set. The dimensions used are as follows: “location” is based on the unique combinations of mun, zone, brgy, and purok of the household; “tenur” refers to the ownership type of the household which can either be owner-like possession of house and lot, rent house/room including lot, own house but rented lot, own house and rent-free lot with consent of owner, own house and rent-free lot without consent of owner, rent-free house and lot with consent of owner, rent-free house and lot without consent of owner, living in a public space with rent, or living in a public space without rent; “house\_type” refers to the type of house in which the household is built which can either be Single house, Duplex, Multi-unit residential (three units or more), Commercial/industrial/agricultural building/house, or Others (boat, cave, etc.); “roof” and “wall” refers to the household’s construction materials which can be any 2 combinations of Strong materials (tile, concrete, brick, stone, galvanized iron), Light materials (bamboo, sawali, cogon, nipa), Salvaged/makeshift materials, Mixed but predominantly strong materials, Mixed but predominantly light materials, Mixed but predominantly salvaged materials, or Not applicable; “water” refers to the household’s water source which can either be Own faucet and community water system, Shared faucet and community water system, Own use tubed/piped deep well, Shared tubed/piped deep well, tubed/piped shallow well, Dug well, Protected Spring, Unprotected Spring, Natural bodies of water (lake, rivers, etc.), Peddler, Bottled water, Others; “welec” simply refers to the presence of electricity within the household; and finally, “nofw” refers to the number of OFW members a household has.

**Figure 1 – Constellation Schema**

For the fact\_calamity table, the dimensions were extracted from the columns of hpq\_hh as well. Considering that details regarding the different calamities did not have its own table from the CBMS database, the group had to create a new one. The dimensions used are as follows: “calamity\_type” can either be Bagyo, Baha, Tagtuyot, Lindol, Pagsabog ng Bulcan, Armadong Digmaan, Landslide, Tsunami, Sunog, Forest Fire, or Others; “hh\_count” refers to the total number of distinct households affected by a certain calamity (calam\_hwmny); “avg\_frequency” refers to the average number of times a calamity occurs within a certain area (calam\_freq); and finally, “avg\_aid” which refers to the how often aid arrives when a specific calamity occurs (calam\_aid).

For the fact\_member table, the dimensions were extracted from the columns of hpq\_mem. The dimensions used are as fofllows: “household\_id” refers to which household the member belongs to; “member\_id” refers to the member’s count from the household; “jobind” refers to the job status of the member which can either be Employed or Unemployed; “jstatus” refers to the member’s type of employment (if employed) which can either be Permanent, Short-term/seasonal/casual, or Worked on different jobs on day to day or week to week; “workcl” refers to the type of worker the member is (if employed) which can either be Working for private household, Working for private business/establishment/farm, Working for government/corporation, Self-employed without any employee, Employer in own family-operated or business, Working with pay on family-operated or business, Working without pay on family-operated or business; and finally “aquani\_status” and “crop\_status” which refers to the presence of the member’s household and member ID in the tables “hpq\_aquani” and “hpq\_crop” from the CBMs database in order to determine whether the member focuses on an agricultural livelihood.

* 1. **Schema and Issues**

A constellation schema was implemented, with the dimension tables revolving around fact\_household, fact\_calamity, and fact\_member. Both fact\_household and fact\_calamity tables reference the dim\_location table and the fact\_member table references fact\_household.

Since the original CBMS database mostly had one to one data references (meaning only one or two dimensions at a time are referenced at most), transforming these data into a constellation would allow for more than one analytical view and extraction for data. One way to transform the table into a multidimensional schema is to consider certain categorical columns from a table as individual dimension tables considering that most of them are descriptors (values in these columns represent certain information) and having the original table serve as a fact table referencing all the dimensions needed.

1. **ETL PROCESS**
   1. **ETL Tool and Code Description**

The group decided to create their own ETL tool using MySQL. Rather than extracting from the individual csv files containing the Palawan and Marinduque data sets, the group decided to perform the ETL process on a complete data dump containing both Palawan and Marinduque data sets. Instead of an actual ETL tool provided by MySQL, the group created a transformation and loading script instead, which SELECTs and transforms the required data from the CBMS database and then INSERTs them the group’s created schema.

* 1. **Extraction and Data Sources**

Data was extracted from the select tables as specified in Sections 2.1. and 2.2. from the original CBMS database. As for the categorical dimension tables in the newly created schema, IDs and their respective description labels were taken from the Data Dictionary provided with the CBMS database.

* 1. **Transformation Functions**

Data was extracted and transformed from the select tables as specified in Sections 2.1. and 2.2. from the original CBMS database.

For the fact\_household table, certain values had to be transformed in order to represent a hierarchal ranking for the purpose of handling certain OLAP queries. The location values had to be transformed as well into a set of unique combinations of mun, zone, brgy, and purok in order for the schema to support further dimensional referencing.

INSERT INTO constellation.fact\_household

SELECT H.id, L.location\_id, IFNULL(H.tenur, 0), H.house\_type,

CASE

WHEN H.roof = 1 THEN 1

WHEN H.roof = 2 THEN 5

WHEN H.roof = 3 THEN 3

WHEN H.roof = 4 THEN 2

WHEN H.roof = 5 THEN 6

WHEN H.roof = 6 THEN 4

WHEN H.roof = 7 THEN 7

END AS 'roof',

CASE

WHEN H.wall = 1 THEN 1

WHEN H.wall = 2 THEN 5

WHEN H.wall = 3 THEN 3

WHEN H.wall = 4 THEN 2

WHEN H.wall = 5 THEN 6

WHEN H.wall = 6 THEN 4

WHEN H.wall = 7 THEN 7

END AS 'wall',

H.water, IFNULL(H.welec, 2), H.nofw

FROM db\_hpq.hpq\_hh H

INNER JOIN constellation.dim\_location L

ON H.mun = L.mun AND H.zone = L.zone AND H.brgy = L.brgy AND H.purok = L.purok

GROUP BY H.id;

For the fact\_calamity table, all of the values had to be derived from groups of households within a certain location in order to normalize data. According to the CBMS database, calamities are listed and occurs per household rather than per location, which further supports the suggestion that a new set of locations should be created as a set of unique combinations of mun, zone, brgy, and purok. As all of the details of calamities are combined into a single row on separate columns from the hpq\_hh table in the CBMS database, the query below had to be done eleven times (number of unique types of calamities) with the calamity ID changing depending on the calamity being referenced. Other details transformed and included is the average occurrence frequency of a calamity in a certain location and the average arrival of an aid when a specific calamity occurs; these averages are calculated depending on the number of households who have reported to be affected by the same calamity.

INSERT INTO constellation.fact\_calamity

SELECT L.location\_id, <calamity\_id>, COUNT(L.location\_id), AVG(H.calam<calamity id>\_hwmny),

2 - AVG(H.calam<calamity id>\_\_aid)

FROM db\_hpq.hpq\_hh H

INNER JOIN constellation.dim\_location L

ON H.mun = L.mun AND H.zone = L.zone AND H.brgy = L.brgy AND H.purok = L.purok

WHERE H.calam<calamity id> = 1

GROUP BY L.location\_id;

For the fact\_member table, a basic extraction is performed with the exception of referencing other tables (hpq\_crop and hpq\_aquani) other than hpq\_mem for the extraction of additional member data. Considering that some values necessary from the CBMS database are left null/empty, they are transformed into their corresponding values from their respective dimension tables.

INSERT INTO constellation.fact\_member

SELECT M.id, M.memno,

IFNULL(M.jobind, 0) AS 'jobind',

IFNULL(M.jstatus, 0) AS 'jstatus',

IFNULL(M.workcl, 0) AS 'workcl',

IF(A.id IS NOT NULL, TRUE, FALSE) AS 'aquani\_status',

IF(C.id IS NOT NULL, TRUE, FALSE) AS 'crop\_status'

FROM (SELECT id, memno, jobind, jstatus, workcl FROM db\_hpq.hpq\_mem) M

LEFT JOIN (SELECT hpq\_hh\_id, id FROM db\_hpq.hpq\_aquani) AS A

ON M.id = A.hpq\_hh\_id AND M.memno = A.id

LEFT JOIN (SELECT hpq\_hh\_id, id FROM db\_hpq.hpq\_crop) AS C

ON M.id = C.hpq\_hh\_id AND M.memno = C.id

WHERE M.id IN (SELECT household\_id FROM constellation.fact\_household)

ORDER BY M.id;

For the dim\_location table, as mentioned earlier, a new set of unique combinations of mun, zone, brgy, and purok is created with a corresponding id (auto-incremented by MySQL). Other information included in the dim\_location table is the number of households situated in that certain area.

INSERT INTO constellation.dim\_location(mun, zone, brgy, purok, hh\_count)

SELECT DISTINCT mun, zone, brgy, purok, COUNT(id) AS 'hh\_count'

FROM db\_hpq.hpq\_hh

GROUP BY mun, zone, brgy, purok;

Cubes were created in order to demonstrate cube specific result sets and OLAP differences as compared to the regular queries.

Two cubes were created, both of which provide detailed information about the status of households affected by calamities with regards to the frequency of the occurrence of certain calamities in an area as well as the average percentage of aid being sent per occurrence within an area, and the percentage of strongly, weakly, and mixed built households. The only difference between the two is that the first one is a simplified cube which uses the created dim\_location set in order to reference locations and the second one uses a more specific data set as it compares various combinations of mun, zone, brgy, and purok. Only the former will be discussed in the paper as both queries require multiple UNIONs for every combination of GROUP BYs WITH ROLLUP in order for MySQL to simulate a cube

CREATE TABLE cube\_calamity

SELECT L.location\_id, DC.calamity\_desc,

SUM(C.avg\_frequency) AS 'frequency\_sum',

AVG(C.avg\_aid) AS 'aid\_avg',

IF(STRONG.count IS NULL, 0, STRONG.count/TOTAL.count) AS 'strong\_avg',

IF(WEAK.count IS NULL, 0, WEAK.count/TOTAL.count) AS 'weak\_avg',

CASE

WHEN STRONG.count IS NULL AND WEAK.count IS NOT NULL THEN (TOTAL.count - WEAK.count)/TOTAL.count

WHEN STRONG.count IS NOT NULL AND WEAK.count IS NULL THEN (TOTAL.count - STRONG.count)/TOTAL.count

WHEN STRONG.count IS NOT NULL AND WEAK.count IS NOT NULL THEN (TOTAL.count - (STRONG.count + WEAK.count))/TOTAL.count

END AS 'others\_avg'

FROM (dim\_location L, fact\_calamity C,

(SELECT location\_id, COUNT(\*) as 'count'

FROM fact\_household

GROUP BY location\_id) AS TOTAL)

INNER JOIN dim\_calamity AS DC ON C.calamity\_id = DC.calamity\_id

LEFT JOIN (SELECT location\_id, COUNT(\*) as 'count'

FROM fact\_household

WHERE (roof\_id = 1 OR roof\_id = 2)

AND (wall\_id = 1 OR wall\_id = 2)

GROUP BY location\_id) AS STRONG

ON L.location\_id = STRONG.location\_id

LEFT JOIN (SELECT location\_id, COUNT(\*) as 'count'

FROM fact\_household

WHERE (roof\_id = 3 OR roof\_id = 4 OR roof\_id = 5 OR roof\_id = 6 OR roof\_id = 7)

AND (wall\_id = 3 OR wall\_id = 4 OR wall\_id = 5 OR wall\_id = 6 OR wall\_id = 7)

GROUP BY location\_id) AS WEAK

ON L.location\_id = WEAK.location\_id

WHERE L.location\_id = C.location\_id

AND L.location\_id = TOTAL.location\_id

GROUP BY L.location\_id, DC.calamity\_desc WITH ROLLUP

UNION

SELECT L.location\_id, DC.calamity\_desc,

SUM(C.avg\_frequency) AS 'frequency\_sum',

AVG(C.avg\_aid) AS 'aid\_avg',

IF(STRONG.count IS NULL, 0, STRONG.count/TOTAL.count) AS 'strong\_avg',

IF(WEAK.count IS NULL, 0, WEAK.count/TOTAL.count) AS 'weak\_avg',

CASE

WHEN STRONG.count IS NULL AND WEAK.count IS NOT NULL THEN (TOTAL.count - WEAK.count)/TOTAL.count

WHEN STRONG.count IS NOT NULL AND WEAK.count IS NULL THEN (TOTAL.count - STRONG.count)/TOTAL.count

WHEN STRONG.count IS NOT NULL AND WEAK.count IS NOT NULL THEN (TOTAL.count - (STRONG.count + WEAK.count))/TOTAL.count

END AS 'others\_avg'

FROM (dim\_location L, fact\_calamity C,

(SELECT location\_id, COUNT(\*) as 'count'

FROM fact\_household

GROUP BY location\_id) AS TOTAL)

INNER JOIN dim\_calamity AS DC ON C.calamity\_id = DC.calamity\_id

LEFT JOIN (SELECT location\_id, COUNT(\*) as 'count'

FROM fact\_household

WHERE (roof\_id = 1 OR roof\_id = 2)

AND (wall\_id = 1 OR wall\_id = 2)

GROUP BY location\_id) AS STRONG

ON L.location\_id = STRONG.location\_id

LEFT JOIN (SELECT location\_id, COUNT(\*) as 'count'

FROM fact\_household

WHERE (roof\_id = 3 OR roof\_id = 4 OR roof\_id = 5 OR roof\_id = 6 OR roof\_id = 7)

AND (wall\_id = 3 OR wall\_id = 4 OR wall\_id = 5 OR wall\_id = 6 OR wall\_id = 7)

GROUP BY location\_id) AS WEAK

ON L.location\_id = WEAK.location\_id

WHERE L.location\_id = C.location\_id

AND L.location\_id = TOTAL.location\_id

GROUP BY DC.calamity\_desc, L.location\_id WITH ROLLUP;

This query provides 6917 rows, 10 of which are presented in Table 1.

* 1. **Loading and Issues**

The loading of the extracted and transformed data from the CBMS database into the group’s created schema is done via the INSERT INTO functions as specified in Section 3.3. of this paper.

The only issues the group has faced is the matching of data types from the extracted and transformed result set and the table attribute data types of the group’s created schema. Foreign keys and primary keys were implemented in the created schema in order to ensure referential and key consistency.

1. **OLAP QUERIES**
   1. **Base Query**

The base query used in the application gets the average roof and wall materials, water source, and electricity status per location, and per house type. This query is done in order to determine if certain house types are built with better materials as compared to others.

SELECT L.location\_id, HT.house\_type\_desc,

COUNT(DISTINCT(H.household\_id)) AS 'hh\_count',

AVG(H.roof\_id) AS 'roof\_avg',

AVG (H.wall\_id) AS 'wall\_avg',

AVG(H.water\_id) AS 'water\_avg',

2 - AVG(H.welec\_status) AS 'welec\_avg'

FROM (fact\_household H)

INNER JOIN dim\_location AS L ON H.location\_id = L.location\_id

LEFT JOIN dim\_house\_type AS HT ON H.house\_type\_id = HT.house\_type\_id

GROUP BY L.location\_id, HT.house\_type\_desc

ORDER BY L.location\_id ASC, HT.house\_type\_desc ASC;

This query provides 4532 rows, 10 of which are presented in Table 2.

* 1. **Rollup Query**

The rollup query used in the application gets the average roof and wall materials, water source, and electricity status per location. Same as the previous query, this query is done in order to determine if certain households are built with better materials as compared to others within certain locations.

SELECT L.location\_id,

COUNT(DISTINCT(H.household\_id)) AS 'hh\_count',

AVG(H.roof\_id) AS 'roof\_avg',

AVG (H.wall\_id) AS 'wall\_avg',

AVG(H.water\_id) AS 'water\_avg',

2 - AVG(H.welec\_status) AS 'welec\_avg'

FROM (fact\_household H)

INNER JOIN dim\_location AS L ON H.location\_id = L.location\_id

GROUP BY L.location\_id

This query provides 3078 rows, 10 of which are presented in Table 3.

* 1. **Drill Down Query**

From the base query, the query can be further drilled down in two ways.

The first drill down will be by tenur. So adding to the purpose of the base query, this is done in order to determine if certain types of land ownerships affects the quality of household factors a household has.

SELECT L.location\_id, HT.house\_type\_desc, T.tenur\_desc,

COUNT(DISTINCT(H.household\_id)) AS 'hh\_count',

AVG(H.roof\_id) AS 'roof\_avg',

AVG (H.wall\_id) AS 'wall\_avg',

AVG(H.water\_id) AS 'water\_avg',

2 - AVG(H.welec\_status) AS 'welec\_avg'

FROM (fact\_household H)

INNER JOIN dim\_location AS L ON H.location\_id = L.location\_id

LEFT JOIN dim\_house\_type AS HT ON H.house\_type\_id = HT.house\_type\_id

LEFT JOIN dim\_tenur AS T ON H.tenur\_id = T.tenur\_id

GROUP BY L.location\_id, HT.house\_type\_desc, T.tenur\_desc

ORDER BY L.location\_id ASC, HT.house\_type\_desc ASC, T.tenur\_desc ASC;

This query provides 12136 rows, 10 of which are presented in Table 4.

The next drill down will be the most specific, making aggregate functions unnecessary. The query will be further drilled down by household\_id, therefore showing the very base results of the query.

SELECT L.location\_id, HT.house\_type\_desc, T.tenur\_desc, H.household\_id, R.roof\_desc, W.wall\_desc, WA.water\_desc, H.welec\_status

FROM (fact\_household H)

INNER JOIN dim\_location AS L ON H.location\_id = L.location\_id

LEFT JOIN dim\_house\_type AS HT ON H.house\_type\_id = HT.house\_type\_id

LEFT JOIN dim\_tenur AS T ON H.tenur\_id = T.tenur\_id

LEFT JOIN dim\_roof AS R ON H.roof\_id = R.roof\_id

LEFT JOIN dim\_wall AS W ON H.wall\_id = W.wall\_id

LEFT JOIN dim\_water AS WA ON H.water\_id = WA.water\_id

GROUP BY L.location\_id, HT.house\_type\_desc, T.tenur\_desc, H.household\_id

ORDER BY L.location\_id ASC, HT.house\_type\_desc ASC, T.tenur\_desc ASC, H.household\_id ASC;

This query provides 148227 rows, 10 of which are presented in Table 5.

* 1. **Slice/Dice Query**

Slice and dice operations can be performed in all instances of the query (base, rollup, drilldown). While taking into consideration that the focus of the group’s application is calamity affected areas, calamity types as well as specific locations can be used as constraints for the slice and dice operations. Checking whether certain locations receive disaster response/aid can also be another constraint to consider. Further slicing and dicing can be performed with the various member factors of households as well (ex: OFW based households, agricultural livelihood base households, etc.). For this example, the result set of the average factors of households affected by “Bagyo” (calamity\_id = 1), within the Municipality 1 (mun = 1), and did not receive disaster response/aid (avg\_aid = 0) will be presented

SELECT L.location\_id,

COUNT(DISTINCT(H.household\_id)) AS 'hh\_count',

AVG(H.roof\_id) AS 'roof\_avg',

AVG (H.wall\_id) AS 'wall\_avg',

AVG(H.water\_id) AS 'water\_avg',

2 - AVG(H.welec\_status) AS 'welec\_avg'

FROM (fact\_household H)

INNER JOIN dim\_location AS C ON H.location\_id = C.location\_id

INNER JOIN dim\_location AS L ON H.location\_id = L.location\_id

WHERE C.calamity\_id = 1

AND C.avg\_aid = 0

AND L.mun = 1

GROUP BY L.location\_id

This query provides 134 rows, 10 of which are presented in Table 6.

1. **RESULTS AND ANALYSIS**
   1. **Dimensional Model**

Considering that the newly created schema was already processed during the ETL in order to support OLAP functions, it no longer requires as much joins as compared to querying from the original database when performing OLAP functions.

Consider the simple query getting all of the unique locations affected by the calamity “Bagyo”. For the created schema the query will be as follows:

SELECT location\_id

FROM constellations.fact\_calamity

WHERE calamity\_id = 1;

This results with 2284 rows with an average of 0.038 seconds execution time per 5 runs. While on the original database:

SELECT L.location\_id

FROM db\_hpq.hpq\_hh H

INNER JOIN constellation.dim\_location L

ON H.mun = L.mun AND H.zone = L.zone AND H.brgy = L.brgy AND H.purok = L.purok

WHERE calam1 = 1

GROUP BY L.location\_id;

Which resulted with 2284 rows as well but with an average of 19.1828 seconds execution time per 5 runs.

Going for the more complicated queries, using the query from Section 4.3. of this paper on the original database:

SELECT L.location\_id, HT.house\_type\_desc, T.tenur\_desc, H.household\_id, R.roof\_desc, W.wall\_desc, WA.water\_desc, H.welec\_status

FROM (fact\_household H)

INNER JOIN dim\_location AS L ON H.location\_id = L.location\_id

LEFT JOIN dim\_house\_type AS HT ON H.house\_type\_id = HT.house\_type\_id

LEFT JOIN dim\_tenur AS T ON H.tenur\_id = T.tenur\_id

LEFT JOIN dim\_roof AS R ON H.roof\_id = R.roof\_id

LEFT JOIN dim\_wall AS W ON H.wall\_id = W.wall\_id

LEFT JOIN dim\_water AS WA ON H.water\_id = WA.water\_id

GROUP BY L.location\_id, HT.house\_type\_desc, T.tenur\_desc, H.household\_id

ORDER BY L.location\_id ASC, HT.house\_type\_desc ASC, T.tenur\_desc ASC, H.household\_id ASC;

Resulted in 148277 rows with an average of 7.52818 seconds execution time per 5 runs. While performing the equivalent query on the original database:

SELECT L.location\_id, HT.house\_type\_desc, T.tenur\_desc, H.id, R.roof\_desc, W.wall\_desc, WA.water\_desc, H.welec

FROM (db\_hpq.hpq\_hh H)

INNER JOIN constellation.dim\_location L

ON H.mun = L.mun AND H.zone = L.zone AND H.brgy = L.brgy AND H.purok = L.purok

LEFT JOIN dim\_house\_type AS HT ON H.house\_type = HT.house\_type\_id

LEFT JOIN dim\_tenur AS T ON H.tenur = T.tenur\_id

LEFT JOIN dim\_roof AS R ON H.roof = R.roof\_id

LEFT JOIN dim\_wall AS W ON H.wall = W.wall\_id

LEFT JOIN dim\_water AS WA ON H.water = WA.water\_id

GROUP BY L.location\_id, HT.house\_type\_desc, T.tenur\_desc, H.id

ORDER BY L.location\_id ASC, HT.house\_type\_desc ASC, T.tenur\_desc ASC, H.id ASC

Resulted in 148277 rows as well but with an average 55.428 seconds of execution time per 5 runs

By providing test cases of varied complexity, it can be concluded that the usage of the data warehousing schema is more efficient than the usage of the original schema, due to the lower normalization required, which needs lesser joins and processes only the necessary data.

* 1. **ETL Process**

In order to check if the ETL process has been performed properly, the group compared certain queries on both the original CBMS database and on the created schema. Ideally, both should have the same result sets, with the exception of marginal differences.

Using the same queries used in Section 5.1. of this paper,

SELECT location\_id

FROM constellation.fact\_calamity

WHERE calamity\_id = 1;

On the created schema and

SELECT L.location\_id

FROM db\_hpq.hpq\_hh H

INNER JOIN constellation.dim\_location L

ON H.mun = L.mun AND H.zone = L.zone AND H.brgy = L.brgy AND H.purok = L.purok

WHERE calam1 = 1

GROUP BY L.location\_id;

On the original CBMS database, provided the same results. After performing the same tests on the remaining queries, the only noticeable difference between the original and the created one is the number of household\_ids present in each respective databases.

Performing the query below on both database

SELECT COUNT(household\_id)

FROM constellation.fact\_household;

The original database returned with 148240 rows and 148227 on the created one. After further analysis, the following query was performed on the original database in order to check for duplicates

SELECT COUNT(DISTINCT(id))

FROM db\_hpq.hpq\_hh;

This now resulted with 148227 rows, similar to that of the created schema. This is because duplicate rows were handled during the ETL process as such, the new schema did not allow these entries to be processed at all, as compared to the original database which allowed for duplicate key inputs.

1. **CONCLUSION**

Before applying Online Analytical Processing (OLAP) queries on a database, it is important to ensure that the schema being used is properly designed to handle and accommodate the processes. That being said, Extract, Transform, and Load (ETL) processes are heavily involved when it comes to creating applications which utilizes OLAP queries as the efficiency and effectiveness of the application solely depends on the precision and the accuracy of the transformation of the extracted/mined data into an OLAP supported schema.

The ETL process must be accurate in a sense that the data should similar, if not exact, from the original mined database. The only marginal differences accepted are the transformed values unaccounted for from the original database.

Overall, the group was able to implement an OLAP application which utilizes an OLAP based on a constellation schema generated from the original CBMS data set via ETL.

1. **REFERENCES**

[1]Faroult, S. (2007, March 29). Emulating Analytic (AKA Ranking) Functions with MySQL. Retrieved March 27, 2016, from http://www.onlamp.com/pub/a/mysql/2007/03/29/emulating-analytic-aka-ranking-functions-with-mysql.html

[2] OLTP vs. OLAP. (n.d.). Retrieved March 27, 2016, from http://datawarehouse4u.info/OLTP-vs-OLAP.html

[3] What is the definition of OLAP? (n.d.). Retrieved March 27, 2016, from http://olap.com/olap-definition/