**ADVANDB MCO1: Query Optimization**

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**

The first machine project on ADVANDB focuses on database level manipulation and optimizations with regards to query cost, time, and efficiency. The group’s application aims to present both of the optimized and non-optimized statistical data processed and gathered from the Palawan CMBS Data Sets while using the Data Dictionary as reference as to which tables and columns to process data from. The overall project’s goal is to differentiate between the efficiency and effectiveness of poorly or averagely made SQL statements and those that are properly made with corresponding database level optimizations.

1. **ORIGNAL QUERIES**
   1. **Average house materials of households with OFWs**

For displaying the result set of OFW households

SELECT id, nofw, roof, wall

FROM hpq\_hh

WHERE nofw > 0

Expected Output: 1,826 rows, with 4 columns (id, nofw, roof, wall)

For getting the average of strongly built OFW houses

SELECT numer.count, denom.count, numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw > 0) AS denom,

(SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw > 0

AND (roof = 1 or roof = 4)

AND (wall = 1 or wall = 4)) AS numer;

Expected Output: Single row, with 3 columns (number of strongly built OFW houses, total number of OFW houses, average between the 2)

For getting the average of weakly built OFW houses

SELECT numer.count, denom.count, numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw > 0) AS denom,

(SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw > 0

AND (roof = 2 or roof = 3 or roof = 5 or roof = 6 or roof = 7)

AND (wall = 2 or wall = 3 or wall = 5 or wall = 6 or wall = 7)) AS numer;

Expected Output: Single row, with 3 columns (number of weakly built OFW houses, total number of OFW houses, average between the 2)

For displaying the result set of non-OFW households

SELECT id, nofw, roof, wall

FROM hpq\_hh

WHERE nofw = 0

Expected Output: 98,344 rows, with 4 columns (id, nofw, roof, wall)

For getting the average of strongly built non-OFW houses

SELECT numer.count, denom.count, numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw > 0) AS denom,

(SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw = 0

AND (roof = 2 or roof = 3 or roof = 5 or roof = 6 or roof = 7)

AND (wall = 2 or wall = 3 or wall = 5 or wall = 6 or wall = 7)) AS numer;

Expected Output: Single row, with 3 columns (number of strongly built non-OFW houses, total number of non-OFW houses, average between the 2)

SELECT numer.count, denom.count, numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw > 0) AS denom,

(SELECT COUNT(\*) AS count

FROM hpq\_hh

WHERE nofw = 0

AND (roof = 1 or roof = 4)

AND (wall = 1 or wall = 4)) AS numer;

For getting the average of weakly built non-OFW houses

Expected Output: Single row, with 3 columns (number of weakly built non-OFW houses, total number of non-OFW houses, average between the 2)

* 1. **Average educational and work status of married people in a certain age**

For displaying the result set of all of the married people with a given age

SELECT id, civstat, age\_yr, jobind, educind

FROM hpq\_mem

WHERE age\_yr <= [input] AND civstat = 2 AND educind = [input] AND jobind = [input]

User inputs: int age, int educational status, int employment status

Expected output: number of rows depends on the number of hpq\_mem members within the specified age. 5 columns (id, civil status, age, educational status, employment status)

For getting the average number of people aged [input] who are married and are currently employed

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(id) AS count

FROM hpq\_mem

WHERE age\_yr <= [input]

AND civstat = 2) AS denom,

(SELECT COUNT(id) AS count

FROM hpq\_mem

WHERE age\_yr <= [input]

AND civstat = 2 AND jobind = 1) AS numer;

User inputs: int age, int educational status, int employment status

Expected output: Single row, with 3 columns (number of employed married people aged [input], total number of married people aged [input], average between the 2)

For getting the average number of people aged [input] who are married and are currently in education

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(id) AS count

FROM hpq\_mem

WHERE age\_yr <= [input]

AND civstat = 2) AS denom,

(SELECT COUNT(id) AS count

FROM hpq\_mem

WHERE age\_yr <= [input]

AND civstat = 2 AND educind = 1) AS numer;

User inputs: int age, int educational status, int employment status

Expected output: Single row, with 3 columns (number of educated married people aged [input], total number of married people aged [input], average between the 2)

For getting the average number of people aged [input] who are married and are neither in employment nor in education

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(id) AS count

FROM hpq\_mem

WHERE age\_yr <= [input]

AND civstat = 2) AS denom,

(SELECT COUNT(id) AS count

FROM hpq\_mem

WHERE age\_yr <= [input]

AND civstat = 2 AND jobind = 2 AND educind = 2) AS numer;

User inputs: int age, int educational status, int employment status

Expected output: Single row, with 3 columns (number of married people aged [input] neither in education nor employment, total number of married people aged [input], average between the 2)

* 1. **Average number of maids hired by OFWs vs those hire by non-OFWs**

For getting the result set of the OFW households and the number of their distinct maids.

SELECT DISTINCT(MAID.id), COUNT(MAID.id)

FROM (SELECT id

FROM hpq\_mem

WHERE reln = 8)

AS MAID

WHERE MAID.id IN (SELECT id FROM hpq\_hh where nofw > 0)

GROUP BY MAID.id;

Expected result: number of rows depends on the distinct count of OFW households with at least 1 maid. 2 columns (id, number of distinct maids)

For getting the average number of maids hired by OFW households.

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM db\_hpq.hpq\_mem mem, hpq\_hh hh

WHERE reln = 8 and mem.id = hh.id) AS denom,

(SELECT COUNT(MAID.id) AS count

FROM (SELECT id

FROM hpq\_mem

WHERE reln = 8)

AS MAID

WHERE MAID.id IN (

SELECT id

FROM hpq\_hh

WHERE nofw > 0)) AS numer;

Expected output: Single row, with 3 columns (number of maids hired by OFW households, total number of maids, average between the 2)

For getting the result set of the non-OFW households and the number of their distinct maids

SELECT DISTINCT(MAID.id), COUNT(MAID.id)

FROM (SELECT id

FROM hpq\_mem

WHERE reln = 8)

AS MAID

WHERE MAID.id IN (SELECT id FROM hpq\_hh where nofw = 0)

GROUP BY MAID.id;

Expected result: number of rows depends on the distinct count of non-OFW households with at least 1 maid. 2 columns (id, number of distinct maids)

For getting the average number of maids hired by non-OFW households.

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM db\_hpq.hpq\_mem mem, hpq\_hh hh

WHERE reln = 8 and mem.id = hh.id) AS denom,

(SELECT COUNT(MAID.id) AS count

FROM (SELECT id

FROM hpq\_mem

WHERE reln = 8)

AS MAID

WHERE MAID.id IN (

SELECT id

FROM hpq\_hh

WHERE nofw > 0)) AS numer;

Expected output: Single row, with 3 columns (number of maids hired by non-OFW households, total number of maids, average between the 2)

* 1. **Average number of deaths a calamity produces while considering its frequency**

For getting the result set of the households who had members pass away due to a calamity, while considering the calamity’s frequency.

SELECT HH.id, HH.calam[input], HH.calam[input]\_hwmny, D.mdeady

FROM hpq\_hh HH, hpq\_death D

WHERE calam[input] = 1

AND calam[input]\_hwmny <= [input]

AND HH.id = D.hpq\_hh\_id;

User input: int calamity type, int calamity frequency

Expected output: number of rows depends on the number of households with members who passed away due to a calamity, 4 columns (id, calamity type, calamity frequency, reason of death)

For getting the average number of deaths caused by the specified calamity given its frequency

SELECT numer.count, denom.count, numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM hpq\_hh HH, hpq\_death D

WHERE HH.id = D.hpq\_hh\_id) AS denom,

(SELECT COUNT(\*) AS count

FROM hpq\_hh HH, hpq\_death D

WHERE HH.calam[input] = 1

AND HH.calam[input]\_hwmny > 0

AND HH.id = D.hpq\_hh\_id) AS numer

User input: int calamity type, int calamity frequency

Expected output: Single row, 3 columns (number of deaths caused by the calamity, total number of deaths, average between the 2)

* 1. **Average house materials of households focusing on agricultural livelihood**

For getting the result set of the households and the distinct number of members participating in agricultural activity

SELECT MAIN.id, MAIN.roof, MAIN.wall,

COUNT(AQUA.hpq\_hh\_id),

COUNT(CROP.hpq\_hh\_id)

FROM hpq\_hh MAIN, hpq\_aquani AQUA, hpq\_crop CROP

WHERE MAIN.id = AQUA.hpq\_hh\_id

AND MAIN.id = CROP.hpq\_hh\_id

GROUP BY MAIN.id;

Expected result: number of rows depends on the distinct count of households participating in at least 1 agricultural activity. 3 columns (id, number of members joined in aquani, number of members joined in crop)

For getting the average number of strongly built houses of agricultural households

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM db\_hpq.hpq\_hh

WHERE id IN (SELECT hpq\_hh\_id

FROM db\_hpq.hpq\_aquani)

OR id IN (SELECT hpq\_hh\_id

FROM db\_hpq.hpq\_crop)) AS denom,

(SELECT COUNT(\*) AS count

FROM db\_hpq.hpq\_hh

WHERE (roof = 1 OR roof = 4)

AND (wall = 1 OR wall = 4)

AND (id IN (SELECT hpq\_hh\_id FROM db\_hpq.hpq\_aquani)

OR id IN (SELECT hpq\_hh\_id FROM db\_hpq.hpq\_crop))) AS numer

Expected output: Single row, 3 columns (number of strongly built houses, total number of agricultural households, average between the 2)

For getting the average number of weakly built houses of agricultural households

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(\*) AS count

FROM db\_hpq.hpq\_hh

WHERE id IN (SELECT hpq\_hh\_id

FROM db\_hpq.hpq\_aquani)

OR id IN (SELECT hpq\_hh\_id

FROM db\_hpq.hpq\_crop)) AS denom,

(SELECT COUNT(\*) AS count

FROM db\_hpq.hpq\_hh

WHERE (roof = 2 OR roof = 3 OR roof = 5 OR roof = 6 OR roof = 7)

AND (wall = 2 OR wall = 3 OR wall = 5 OR wall = 6 OR wall = 7)

AND (id IN (SELECT hpq\_hh\_id FROM db\_hpq.hpq\_aquani)

OR id IN (SELECT hpq\_hh\_id FROM db\_hpq.hpq\_crop))) AS numer

Expected output: Single row, 3 columns (number of weakly built houses, total number of agricultural households, average between the 2)

* 1. **Average number of crop types non-OFWs produce as compared to OFWs while considering both own agricultural land and are members of ARCDP**

For getting the result set of the number of distinct crop types per OFW household, ARCDP members

SELECT MAIN.id, MAIN.nofw, COUNT(DISTINCT(CROP.croptype))

FROM hpq\_hh MAIN, hpq\_mem MEM,

hpq\_arcdp\_mem ARCDP, hpq\_crop CROP

WHERE MAIN.id = MEM.id

AND MEM.id = ARCDP.hpq\_hh\_id

AND ARCDP.hpq\_hh\_id = CROP.hpq\_hh\_id

AND MAIN.nofw > 0

GROUP BY MAIN.id;

Expected output: number of rows depends on the distinct number of OFW households and are members of ARCDP, 3 columns (id, nofw, number of distinct crop types)

For getting the average number of OFW, ARCDP members

SELECT numer.count, denom.count, numer.count/denom.count

FROM(SELECT COUNT(DISTINCT(CROP.hpq\_hh\_id))

FROM hpq\_crop CROP, hpq\_arcdp\_mem ARCDP

WHERE CROP.hpq\_hh\_id = ARCDP.hpq\_hh\_id) AS denom,

(SELECT COUNT(DISTINCT(CROP.hpq\_hh\_id))

FROM hpq\_hh MAIN, hpq\_mem MEM, hpq\_crop CROP, hpq\_arcdp\_mem ARCDP

WHERE MAIN.id = MEM.id

AND MEM.id = ARCDP.hpq\_hh\_id

AND ARCDP.hpq\_hh\_id = CROP.hpq\_hh\_id

AND MAIN.nofw > 0) AS numer

Expected output: single row, 3 columns (number of OFW, ARCDP members, total ARCDP members, average between the 2)

For getting the result set of the number of distinct crop types per non-OFW household, ARCDP members

SELECT MAIN.id, MAIN.nofw, COUNT(DISTINCT(CROP.croptype))

FROM hpq\_hh MAIN, hpq\_mem MEM,

hpq\_arcdp\_mem ARCDP, hpq\_crop CROP

WHERE MAIN.id = MEM.id

AND MEM.id = ARCDP.hpq\_hh\_id

AND ARCDP.hpq\_hh\_id = CROP.hpq\_hh\_id

AND MAIN.nofw = 0

GROUP BY MAIN.id;

Expected output: number of rows depends on the distinct number of non-OFW households and are members of ARCDP, 3 columns (id, nofw, number of distinct crop types)

For getting the average number of non-OFW, ARCDP members

SELECT numer.count, denom.count, numer.count/denom.count

FROM(SELECT COUNT(DISTINCT(CROP.hpq\_hh\_id))

FROM hpq\_crop CROP, hpq\_arcdp\_mem ARCDP

WHERE CROP.hpq\_hh\_id = ARCDP.hpq\_hh\_id) AS denom,

(SELECT COUNT(DISTINCT(CROP.hpq\_hh\_id))

FROM hpq\_hh MAIN, hpq\_mem MEM, hpq\_crop CROP, hpq\_arcdp\_mem ARCDP

WHERE MAIN.id = MEM.id

AND MEM.id = ARCDP.hpq\_hh\_id

AND ARCDP.hpq\_hh\_id = CROP.hpq\_hh\_id

AND MAIN.nofw = 0) AS numer

Expected output: single row, 3 columns (number of non-OFW, ARCDP members, total ARCDP members, average between the 2)

* 1. **Most participated beneficiaries in average per household**

For getting the result set of the distinct households who has participated in at least 1 of the 3 beneficiaries (Cash for Work, Food for Work, Food for School) and their respective counts

SELECT MAIN.id, CSHWRK.cshwrkCnt, FUDWRK.fudwrkCnt, FUDSCH.fudschCnt

FROM hpq\_hh MAIN, hpq\_mem MEM

LEFT JOIN (SELECT MAIN.id AS 'cshwrkID',

COUNT(DISTINCT(CSHWRK.id)) AS 'cshwrkCnt'

FROM hpq\_hh MAIN, hpq\_mem MEM, hpq\_cshforwrk\_mem CSHWRK

WHERE MAIN.id = MEM.id

AND MEM.id = CSHWRK.hpq\_hh\_id

GROUP BY MAIN.id) AS CSHWRK

ON MEM.id = CSHWRK.cshwrkID

LEFT JOIN (SELECT MAIN.id AS 'fudwrkID',

COUNT(DISTINCT(FUDWRK.id)) AS 'fudwrkCnt'

FROMhpq\_hh MAIN, hpq\_mem MEM, hpq\_fudforwrk\_mem FUDWRK

WHERE MAIN.id = MEM.id

AND MEM.id = FUDWRK.hpq\_hh\_id

GROUP BY MAIN.id) AS FUDWRK

ON MEM.id = FUDWRK.fudwrkID

LEFT JOIN (SELECT MAIN.id AS 'fudschID',

COUNT(DISTINCT(FUDSCH.id)) AS 'fudschCnt'

FROM hpq\_hh MAIN, hpq\_mem MEM, hpq\_fudforsch\_mem FUDSCH

WHERE MAIN.id = MEM.id

AND MEM.id = FUDSCH.hpq\_hh\_id

GROUP BY MAIN.id) AS FUDSCH

ON MEM.id = FUDSCH.fudschID

WHERE MAIN.id = MEM.id

AND !(CSHWRK.cshwrkCnt is null

AND FUDWRK.fudwrkCnt is null

AND FUDSCH.fudschCnt is null)

GROUP BY MAIN.id;

Expected output: number of rows depends on the distinct households who has joined at least 1 of the 3 beneficiaries, 4 columns (id, number of Cash for Works joined, number of Food for Works joined, number of Food for Schools joined)

For getting the total number of distinct households who has joined at least 1 of the 3 beneficiaries

SELECT COUNT(DISTINCT(MAIN.id))

FROM hpq\_hh MAIN, hpq\_mem MEM

WHERE MAIN.id = MEM.id

AND (MEM.id IN (SELECT hpq\_hh\_id FROM hpq\_cshforwrk\_mem)

OR MEM.id IN (SELECT hpq\_hh\_id FROM hpq\_fudforwrk\_mem)

OR MEM.id IN (SELECT hpq\_hh\_id FROM hpq\_fudforsch\_mem));

Expected output: single row, single column (id count)

For getting the result set of distinct households who has joined Cash for Work

SELECT MAIN.id, COUNT(DISTINCT(CSHWRK.id))

FROM hpq\_hh MAIN, hpq\_mem MEM,

hpq\_cshforwrk\_mem CSHWRK

WHERE MAIN.id = MEM.id

AND MEM.id = CSHWRK.hpq\_hh\_id

GROUP BY MAIN.id;

Expected output: number of rows depends on the distinct households who has joined Cash for Work, 2 columns (id, number of Cash for Works joined)

For getting the result set of distinct households who has joined Food for Work

SELECT MAIN.id, COUNT(DISTINCT(FUDWRK.id))

FROM hpq\_hh MAIN, hpq\_mem MEM,

hpq\_fudforwrk\_mem FUDWRK

WHERE MAIN.id = MEM.id

AND MEM.id = FUDWRK.hpq\_hh\_id

GROUP BY MAIN.id;

Expected output: number of rows depends on the distinct households who has joined Food for Work, 2 columns (id, number of Food for Works joined)

For getting the result set of distinct households who has joined Food for School

SELECT MAIN.id, COUNT(DISTINCT(FUDSCH.id))

FROM hpq\_hh MAIN, hpq\_mem MEM,

hpq\_fudforsch\_mem FUDSCH

WHERE MAIN.id = MEM.id

AND MEM.id = FUDSCH.hpq\_hh\_id

GROUP BY MAIN.id;

Expected output: number of rows depends on the distinct households who has joined Food for School, 2 columns (id, number of Food for Schools joined)

1. **OPTIMIZATION METHODS APPLIED**
   1. **Average house materials of households with OFWs**
      1. **Heuristics**

SELECT NUMER.count, DENOM.count, NUMER.count/DENOM.count

FROM (SELECT COUNT(id) as count FROM hpq\_hh WHERE nofw = 0) DENOM ,

(SELECT COUNT(id) as count FROM (SELECT id,nofw FROM hpq\_hh WHERE (roof = 1 or roof = 4)

AND (wall = 1 or wall = 4)) temp WHERE nofw = 0) NUMER;

**3.1.2. Indices**

CREATE INDEX hpq\_hh\_nofw ON hpq\_hh(nofw);

CREATE INDEX hpq\_hh\_wall ON hpq\_hh(wall);

CREATE INDEX hpq\_hh\_roof ON hpq\_hh(roof);

* + 1. **Views**

CREATE VIEW nOFWView AS

SELECT id,nofw,roof,wall

FROM hpq\_hh;

**3.1.4. Stored Procedure**

DELIMITER $$

CREATE PROCEDURE strongWalls1(

IN strongwall1 INT,

IN strongwall2 INT,

IN isOFW INT

)

BEGIN

SELECT NUMER.count, DENOM.count, NUMER.count/DENOM.count

FROM (SELECT count(id) as count FROM nOFWView WHERE nofw = isOFW) DENOM ,

(SELECT count(id) as count

FROM (SELECT id,nofw

FROM nOFWView

WHERE (roof = strongwall1 or roof = strongwall2)

AND (wall = strongwall1 or wall = strongwall2) ) temp

WHERE nofw = isOFW) NUMER;

END$$

DELIMITER ;

CALL strongWalls1(1,4,0);

* 1. **Average educational and work status of married people in a certain age** 
     1. **Heuristics**

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(age\_yr) as count

FROM (SELECT age\_yr

FROM hpq\_mem

WHERE civstat = 2) temp

WHERE age\_yr <= 24) as denom,

(SELECT COUNT(age\_yr) as count

FROM (SELECT age\_yr FROM hpq\_mem

WHERE civstat = 2

AND educind = 2 AND jobind = 2) temp1

WHERE age\_yr <= 24) as numer

* + 1. **Indices**

CREATE INDEX hpq\_mem\_civstat ON hpq\_mem(civstat);

CREATE INDEX hpq\_mem\_ageyr ON hpq\_mem(age\_yr);

CREATE INDEX hpq\_mem\_educ ON hpq\_mem(educind);

CREATE INDEX hpq\_mem\_job ON hpq\_mem(jobind);

* + 1. **Views**

CREATE VIEW civstatView AS

SELECT age\_yr,civstat,educind,jobind

FROM hpq\_mem;

CREATE VIEW civstatView1 AS

SELECT age\_yr,civstat

FROM hpq\_mem;

* + 1. **Stored Procedure**

DELIMITER $$

CREATE PROCEDURE civQuery(

IN civstatP INT,

IN educP INT,

IN jobindP INT,

IN ageP INT

)

BEGIN

SELECT numer.count, denom.count, Numer.count/denom.count

FROM (SELECT COUNT(age\_yr) as count

FROM (SELECT age\_yr FROM civstatView1

WHERE civstat = civstatP) temp

WHERE age\_yr <= ageP ) as denom,

(SELECT COUNT(age\_yr) as count

FROM (SELECT age\_yr

FROM civstatView WHERE civstat = civstatP AND educind = educP

AND jobind = jobindP) temp1

WHERE age\_yr <= ageP ) as numer;

END$$

DELIMITER ;

CALL civQuery(2,2,2,24);

* 1. **Average number of maids hired by OFWs vs those hire by non-OFWs**
     1. **Indexing + Heuristics + View**

CREATE INDEX hpq\_reln\_1 ON hpq\_mem(reln);

CREATE INDEX hpq\_nofw\_1 ON hpq\_hh(nofw);

CREATE VIEW hhMaid1 AS

SELECT id, reln

FROM hpq\_mem;

CREATE VIEW hhMaid2 AS

SELECT id, nofw

FROM hpq\_hh;

SELECT DISTINCT(MAID.id), COUNT(MAID.id)

FROM (SELECT id

FROM hhMaid1

WHERE reln = 8) AS MAID

WHERE MAID.id IN (SELECT id

FROM hpq\_hh where nofw > 0)

GROUP BY MAID.id;

* 1. **Average number of deaths a calamity produces while considering its frequency**
     1. **Indices**

CREATE INDEX hpq\_hh\_calam2\_hwmny ON hpq\_hh(calam2\_hwmny);

CREATE INDEX hpq\_death\_mdeady ON hpq\_death(mdeady);

* + 1. **Views**

CREATE VIEW HHView AS

SELECT id, calam2\_hwmny FROM hpq\_hh;

CREATE VIEW HDView AS

SELECT id, mdeady from hpq\_death;

* 1. **Average house materials of households focusing on agricultural livelihood**
     1. **Indices**

CREATE INDEX hpq\_hh\_roof ON hpq\_hh(roof);

CREATE INDEX hpq\_hh\_wall ON hpq\_hh(wall);

* + 1. **Views**

CREATE VIEW HAView AS

SELECT hpq\_hh\_id FROM hpq\_aquani;

CREATE VIEW HCView AS

SELECT hpq\_hh\_id from hpq\_crop;

CREATE VIEW HHView AS

SELECT id, roof, wall from hpq\_hh;

* + 1. **Stored Procedure**

DELIMITER $$

CREATE PROCEDURE strongWalls1(

IN strongwall1 INT,

IN strongwall2 INT

)

BEGIN

SELECT MAIN.id, MAIN.roof, MAIN.wall,

COUNT(AQUA.hpq\_hh\_id),

COUNT(CROP.hpq\_hh\_id)

FROM hpq\_hh MAIN, hpq\_aquani AQUA, hpq\_crop CROP

WHERE (MAIN.roof = strongwall1 OR MAIN.roof = strongwall2)

AND (MAIN.wall = strongwall1 OR MAIN.wall = strongwall2)

AND MAIN.id = AQUA.hpq\_hh\_id

AND MAIN.id = CROP.hpq\_hh\_id

GROUP BY MAIN.id;

END$$

DELIMITER ;

CALL strongWalls1(1,4);

* 1. **Average number of crop types non-OFWs produce as compared to OFWs while considering both own agricultural land and are members of ARCDP**
     1. **Indices**

CREATE INDEX hpq\_hh\_nofw ON hpq\_hh(nofw);

CREATE INDEX hpq\_crop\_croptype ON hpq\_crop(croptype);

* + 1. **Views**

CREATE VIEW HAMView AS

SELECT hpq\_hh\_id

FROM hpq\_arcdp\_mem;

CREATE VIEW HMView AS

SELECT id

FROM hpq\_mem;

CREATE VIEW HCView AS

SELECT croptype, hpq\_hh\_id

FROM hpq\_crop;

* 1. **Most participated beneficiaries in average per household**
     1. **Indices**

CREATE INDEX hpq\_mem\_id ON hpq\_mem(id);

CREATE INDEX hpq\_hh\_id ON hpq\_hh(id);

* + 1. **Views**

CREATE View HHView as

SELECT id from hpq\_hh;

CREATE View HMView AS

SELECT id from hpq\_mem;

1. **RESULTS AND ANALYSIS**
   1. **Average house materials of households with OFWs**

|  |  |
| --- | --- |
| **Method** | **Seconds(Average per 10 runs)** |
| BASE | 2.516 |
| HEURISTICS | 1.703 |
| INDICES | 1.687 |
| VIEWS | 1.619 |

* 1. **Average educational and work status of married people in a certain age**

|  |  |
| --- | --- |
| **Method** | **Seconds(Average per 10 runs)** |
| BASE | 2.516 |
| HEURISTICS | 1.703 |
| INDICES | 1.687 |
| VIEWS | 1.619 |

Basically, the group filtered the rows from the original table and generated a temporary table with only those with civstat of 2 are present. Since age\_yr’s value depends on user input, it is left out of the subquery. This query counts age\_yr only since the results don’t need any other value. This way, since there are lesser projections present the faster the query.

Indeces are not the best solution to optimize this query due to hpq\_mem having the indeces of the keys being searched. Hence, before projecting smaller columns, MySQL looks at the full blown Index of hpq\_mem and looks for the key values in the subquery.

* 1. **Average number of maids hired by OFWs vs those hire by non-OFWs**

|  |  |
| --- | --- |
| **Method** | **Seconds(Average per 10 runs)** |
| BASE | 0.034 |
| OPTIMIZED | 0.012 |

The Query speeds up due to the tables being preprocessed beforehand as a View in MySQL, Since reln and nofw were turned into an index already it uses the index as a searh key intead of looking up the whole table.

* 1. **Average number of deaths a calamity produces while considering its frequency**

|  |  |
| --- | --- |
| **Method** | **Seconds(Average per 10 runs)** |
| BASE | 0.530 |
| INDICES | 0.015 |
| VIEWS | 0.016 |

* 1. **Average house materials of households**

**focusing on agricultural livelihood**

|  |  |
| --- | --- |
| **Method** | **Seconds(Average per 10 runs)** |
| BASE | 0.421 |
| INDICES | 0.312 |
| VIEWS | 0.094 |
| STORED PROCEDURE | 0.406 |

* 1. **Average number of crop types non-OFWs produce as compared to OFWs while considering both own agricultural land and are members of ARCD**

|  |  |
| --- | --- |
| **Method** | **Seconds(Average per 10 runs)** |
| BASE | 0.016 |
| INDICES | 0.015 |
| VIEWS | 0.016 |

* 1. **Most participated beneficiaries in average per household**

|  |  |
| --- | --- |
| **Method** | **Seconds(Average per 10 runs)** |
| BASE | 14.844 |
| INDICES | 15.375 |
| VIEWS | 6.672 |

1. **CONCLUSION**

After conducting the testing and analysis, the group has concluded that the most efficient method(s) of optimizing queries depends the required result set and the data required to be manipulated/accessed by the query itself.

When optimizing a query on a database level, the developer should first plan ahead on which data sets would become necessary for updating, retrieving, and inserting data, in order to properly hypothesize which techniques or methods would best optimize the query, in terms of cost, time, and disk read/writes. The group has also experienced that during the optimization phase of the database, applying relational algebra on base queries was the easiest method, and at most times produces positive results.

The developers should note that different optimizations may give out varying results. Some queries, when applied a given method, may even slow down as compared to the base query. This is why planning ahead is also a factor in optimizing queries.

In order to easily analyze which data are necessary for the developer’s respective queries, having the data dictionary of the database would be most recommended.

1. **REFERENCES**

13.5 SQL Syntax for Prepared Statements. (n.d.). Retrieved on February 28, 2016, from http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-prepared-statements.html

How to Create Index in MySQL. (n.d.). Retrieved on February 28, 2016, from www.w3schools/sql/sql\_create\_index.asp

MySQL Create View tutorial with examples. (n.d.). Retrieved on February 28, 2016, from www.mysqltutorial.org/create-sql-views-mysql.aspx

Sams Teach Yourself SQL in 24 Hours | Copyright | InformIT. (n.d.). Retrieved February 28, 2016, from http://www.informit.com/library/content.aspx?b=STY\_Sql\_24hours

SQL Indexes. (n.d.). Retrieved February 28, 2016, from http://www.tutorialspoint.com/sql/sql-indexes.htm