

The Islamic University–Gaza
Research and Postgraduate Affairs
Faculty of Engineering
Master of Computer Engineering



الجامعة الإسلامية – غزة
شئون البحث العلمي والدراسات العليا
كلية الهندسة
ماجستير هندسة الحاسوب

Design and Implementation of a Data Warehouse Using Dynamic Materialized Views Selection Model: The Islamic University of Gaza as a Case Study

**تصميم وبناء مستودع بيانات باستخدام نموذج ديناميكي لعملية اختيار
التركيبات المجسمة للمستودع: الجامعة الإسلامية بغزة كدراسة حالة**

Belal W. Shbair

Supervised by

Dr. Wesam Ashour

Associate Professor at Islamic University of Gaza

**A thesis submitted in partial fulfillment of the
requirements for the degree of Master of
Computer Engineering**

August/2017

إقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Design and Implementation of a Data Warehouse Using Dynamic Materialized Views Selection Model: The Islamic University of Gaza as a Case Study

**تصميم وبناء مستودع بيانات واستخدام نموذج ديناميكي لعملية اختيار التركيبات
المجسمة للمستودع: الجامعة الإسلامية بغزة كدراسة حالة**

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وأن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل الآخرين لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

Declaration

I understand the nature of plagiarism, and I am aware of the University's policy on this.

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted by others elsewhere for any other degree or qualification.

| | | |
|-----------------|---------------------|-------------|
| Student's name: | Belal Waleed Shbair | اسم الطالب: |
| Signature: | Belal Shbair | التوقيع: |
| Date: | 23 /8/2017 | التاريخ: |



هاتف داخلي ١٩٥٠

مكتب نائب الرئيس للبحث العلمي والدراسات العليا

Ref: / ٣٥ / غ س. الرقم ج

Date: ٢٠١٧ / ٠٨ / التاريخ

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ بلال وليد شلاش شبير لنيل درجة الماجستير في كلية الهندسة قسم هندسة الحاسوب وموضوعها: تصميم وبناء مستودع بيانات باستخدام نموذج ديناميكي لعملية اختيار التركيبات المجسمة للمستودع: الجامعة الإسلامية بغزة كدراسة حالة

Design and Implementation of a Data Warehouse Using Dynamic Materialized Views "

"Selection Model: The Islamic University of Gaza as a Case Study

وبعد المناقشة التي تمت اليوم الأربعاء ١٠ ذو القعدة ١٤٣٧ هـ، الموافق ٢٠١٧/٠٨/٠٢ م الساعة العاشرة صباحاً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

| | | |
|-------|-----------------|-----------------------|
| | مشرفاً و رئيساً | د. وسام محمود عاشور |
| | مناقشاً داخلياً | د. محمد أحمد الحنجوري |
| | مناقشاً خارجياً | د. إيهاب صلاح زقوت |

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في كلية الهندسة / قسم: هندسة الحاسوب.

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

نائب الرئيس لشئون البحث العلمي والدراسات العليا

أ.د. عبد الرؤوف علي المناعمة



Abstract

Nowadays, business decision-making is moving towards advanced performance evaluation process using standard methods such as Key Performance Indicator (KPI). Performance evaluation process identifies a statistically significant connection between KPIs and related information set. However, with the volume of transactional data stored in a database in an Operational Data Store (ODS), traditional reporting process depending on joining multiple tables and logical views cannot be efficiently used to extract related information set for KPI due to high complexity and query cost. The major problem is how to generate the related information set with quality, which abstracts the huge volume of raw data. The scope of this thesis is to design an enterprise Data Warehouse (DW) for Islamic University of Gaza to extract a related information set for its KPIs as a base for Business Intelligence (BI) process. Additionally, we propose a novel and smart materialize views selection model which handles data warehouse storage and performance issues. Our model specifies, dynamically, which data marts and dimension tables should be materialized over time in peak intervals of usage of the data warehouse. Finally, this model is tested over TPC-H benchmark and that resulted high reduction of overall MVPP cost. Also we studied the effect of using DW in IUG and the results show reduction in execution time over 90% for most of views in DW, and it show additional reduction in execution time over 80% for most of views while materializing them.

الملخص

في الوقت الحاضر، تتحرك عملية اتخاذ القرار في مجال الأعمال نحو عملية تقييم الأداء في المؤسسات بأساليب متقدمة وذلك باستخدام أساليب قياسية مثل مؤشر الأداء الرئيسي (KPI). تحدد عملية تقييم الأداء وجود علاقة ذات دلالة إحصائية بين مؤشرات الأداء الرئيسية (KPIs) والمعلومات ذات الصلة. مع الحجم الكبير لبيانات المعاملات اليومية التي تخزن في قاعدة البيانات في مخزن البيانات التشغيلية (ODS)، تعتبر عملية استخراج التقارير التقليدية المعتمدة على ربط جداول أو تراكيبات (views) متعددة لا يمكن استخدامها بكفاءة لاستخراج المعلومات المطلوبة لمؤشر الأداء الرئيسي بسبب التعقيد الكبير وكلفة الاستعلام. المشكلة الرئيسية تكمن في كيفية توليد مؤشرات الأداء الرئيسية مع ضمان الجودة المناسبة في الوقت والتكلفة العملية في عملية تلخيص المعلومات من البيانات الخام الضخمة. نطاق هذه الأطروحة هو تصميم مستودع بيانات للجامعة الإسلامية في غزة لمعالجة وتخزين بيانات مؤشرات الأداء الرئيسية كأساس لمنظومة ذكاء الأعمال (BI). بالإضافة إلى ذلك، يقدم هذا البحث نموذجاً ديناميكياً لعملية تجسيم البيانات في مستودع البيانات المصمم للجامعة الذي بدوره يحسن من أدائه. يقوم النموذج المقترح وبشكل ديناميكي باستكشاف التراكيبات (views) التي يجب تجسيمها في أوقات ذروة استخدامها في مستودع البيانات. تم تجربة النموذج المقترح على مؤشر TPC-H مما نتج عنه تخفيض ملحوظ في تكلفة الـ MVPP الإجمالية له. كذلك تم استعراض تأثير استخدام مستودع البيانات في الجامعة الإسلامية على الوقت والتكلفة العملية لاستخراج التقارير، نتج عنه توفير في الوقت أكثر من 90% لمعظم التراكيبات في مستودع البيانات. بالإضافة إلى ذلك تم فحص تأثير تجسيم التراكيبات في مستودع البيانات مما نتج عنه توفير أكثر من 80% من الوقت المستهلك لمعظم التراكيبات.

Dedication

To my beloved father

To my beloved Mother

To my great brothers and sisters

To my wonderful country Palestine

Acknowledgment

First of all, I thank Allah for guiding me and taking care of me all the time. My life is so blessed because of his majesty.

I wish to express my thanks to my father Dr. Waleed Shbair for his support and motivation in my entire life, and especially during my study. Also I'm thankful to my beloved mother, for her prayers and motivation.

I would like to voice my sincere thanks and really appreciation to my advisor, Dr. Wesam Ashour, for his kindness in providing valuable advice, support, encouragement and guidance throughout the time to get this thesis in its current shape. I have been learned more than study by his coaching with kindness and very nice attitude.

Special thanks to all my family members and friends for their love, support and prayers.

Finally, I would like to voice my appreciation to Interpal for funding and supporting our project through its phases.

Table of Contents

| | |
|---|-------------|
| Declaration | II |
| Abstract | IV |
| المخلص | V |
| Dedication..... | VI |
| Acknowledgment | VII |
| Table of Contents..... | VIII |
| List of Tables..... | X |
| List of Figures | XI |
| List of Abbreviations..... | XIII |
| Chapter 1 Introduction | 2 |
| 1.1 Background..... | 2 |
| 1.2 Scope and Objectives..... | 5 |
| 1.3 Motivation..... | 5 |
| 1.4 Thesis Contribution..... | 6 |
| 1.5 Overview of Thesis | 6 |
| Chapter 2 Literature Review | 8 |
| 2.1 Data Warehouse | 8 |
| 2.2 OLAP vs OLTP | 9 |
| 2.3 Data Warehouse Design Approaches | 10 |
| 2.3.1 Bill Inmon Architecture..... | 10 |
| 2.3.2 Ralph Kimball Architecture | 12 |
| 2.3.3 Differences between Inmon and Kimball Approaches..... | 14 |
| 2.3.4 Deciding Factors..... | 15 |
| 2.4 Dimensional Model..... | 16 |
| 2.5 Definitions | 19 |
| 2.6 DW Case Studies for Higher Education Environments..... | 21 |
| 2.7 Materialized View Selection Algorithms..... | 22 |
| Chapter 3 Design and Implementation..... | 29 |
| 3.1 System Analysis..... | 29 |
| 3.1.1 IUG Data | 30 |
| 3.1.2 Functional Requirement | 30 |
| 3.1.3 Non Functional Requirement | 32 |
| 3.1.4 User Requirement..... | 32 |

| | |
|--|------------|
| 3.1.5 System Requirement..... | 32 |
| 3.2 System Design | 33 |
| 3.2.1 Waterfall Model | 33 |
| 3.2.2 Spiral Model | 34 |
| 3.2.3 Logical Models | 36 |
| 3.2.4 Facts and Dimensions Tables | 36 |
| 3.3 System Implementation | 64 |
| 3.3.1 Design of the Physical Database | 65 |
| 3.3.2 Design of the ETL Process | 66 |
| 3.4 System Verification and Maintenance | 74 |
| Chapter 4 Proposed Materialized View Selection Model | 76 |
| Chapter 5 Results and Discussion | 86 |
| 5.1 TPC Benchmark TM H (TPC-H)..... | 86 |
| 5.2 Legacy System versus Data Warehouse versus DW Materialized Views..... | 101 |
| Chapter 6 Conclusion and Future Works | 107 |
| 6.1 Conclusion | 107 |
| 6.2 Future Works | 108 |
| References | 110 |
| Appendix 1: Legacy System Vs Data Warehouse Vs Materialized Views | 114 |
| A1.1 Students Admission Data Mart..... | 114 |
| A1.2 Geographical Location Data Mart | 115 |
| A1.3 Transfers between Colleges Data Mart | 116 |
| A1.4 Exam Conflicts Data Mart..... | 117 |
| A1.5 SFD Students Registration Data Mart | 118 |
| A1.6 Cash Grants Data Mart | 119 |
| A1.7 Deferred Grant Data Mart | 120 |
| A1.8 Student Fund Summary Data Mart..... | 121 |
| A1.9 Financial Collection Data Mart | 122 |
| Appendix 2: TPC-H Schema Queries and Relational Algebra Trees | 123 |

List of Tables

| | |
|---|-----|
| Table (2.1): Differences between OLTP Database and OLAP Database..... | 10 |
| Table (3.1): Academic Years Dimension | 37 |
| Table (3.2): Academic Semesters Dimension | 37 |
| Table (3.3): Date Dimension | 38 |
| Table (3.4): Study Programs Dimension | 39 |
| Table (3.5): College Dimension | 39 |
| Table (3.6): Departments Dimension | 40 |
| Table (3.7): Courses Dimension | 40 |
| Table (3.8): High School Years Dimension | 41 |
| Table (3.9): High School Grades Dimension | 41 |
| Table (3.10): High School Sections Dimension | 42 |
| Table (3.11): Student States Dimension | 42 |
| Table (3.12): Student Levels Dimension | 43 |
| Table (3.13): Geographical Locations Dimension | 43 |
| Table (3.14): Students Dimension | 44 |
| Table (3.15): Grants Dimension | 45 |
| Table (3.16): Student Registration Hours Dimension | 45 |
| Table (4.1): Frequency Matrix of Last Five Years | 78 |
| Table (4.2): Detracted Frequency Matrix of Last Five Years | 79 |
| Table (4.3): Frequency Matrix of VIEW_5 Over Months | 83 |
| Table (4.4): Frequency Matrix of VIEW_5 Over Weeks | 83 |
| Table (5.1): The TPC-H Schema Table Size | 87 |
| Table (5.2): The Maintenance Cost of the Re-Optimized MVPP | 92 |
| Table (5.3): The Query Processing Cost of the Re-Optimized MVPP | 93 |
| Table (5.4): Usage Frequency Matrix of Queries Q1 through Q7 Over Financial Year's Quarter Weeks | 95 |
| Table (5.5): The Maintenance Cost of the MVPP in 1st Week of 1st Quarter | 97 |
| Table (5.6): The Query Processing Cost of the MVPP in 1st Week of 1st Quarter | 97 |
| Table (5.7): The Maintenance Cost of the MVPP in 2nd Week of 1st Quarter | 98 |
| Table (5.8): The Query Processing Cost of the MVPP in 2nd Week of 1st Quarter | 99 |
| Table (5.9): The Query Processing Cost, Maintenance Cost and Total Cost of 1st and 2nd Weeks of 1st Quarter | 99 |
| Table (5.10): High School Results Experimental Results | 101 |
| Table (5.11): Students Registration - Experimental Results | 103 |
| Table (5.12): Summary of the Comparison between IUG Legacy System, Data Warehouse, and Materialized Views | 104 |
| Table (A1.1): Students Admission - Experimental Results | 114 |
| Table (A1.2): Geographical Location- Experimental Results | 115 |
| Table (A1.3): Transfers between Colleges - Experimental Results | 116 |
| Table (A1.4): Exam Conflicts - Experimental Results | 117 |
| Table (A1.5): SFD Students Registration - Experimental Results | 118 |
| Table (A1.6): Cash Grants - Experimental Results | 119 |
| Table (A1.7): Deferred Grant - Experimental Results | 120 |
| Table (A1.8): Student Fund Summary - Experimental Results | 121 |
| Table (A1.9): Financial Collection - Experimental Results | 122 |

List of Figures

| | |
|--|----|
| Figure (1.1): Concept of Business Intelligence | 3 |
| Figure (1.2): Difference between Logical View and Materialized View | 4 |
| Figure (2.1): Bill Inmon Approach of Data Warehousing | 11 |
| Figure (2.2): The Ralph Kimball Approach..... | 13 |
| Figure (2.3): Example of Dimensional Model..... | 17 |
| Figure (2.4): Example of Hierarchy in Dimensional Model..... | 18 |
| Figure (2.5): Fact and Dimension Tables in a Dimensional Model..... | 19 |
| Figure (2.6): Factless Fact Table Example | 20 |
| Figure (3.1): Spiral Model of the Data Warehouse Life-cycle | 34 |
| Figure (3.2): High School Results Data Mart..... | 46 |
| Figure (3.3): Students Admission Data Mart..... | 47 |
| Figure (3.4): Students Admission Views | 48 |
| Figure (3.5): Students Registration Data Mart..... | 49 |
| Figure (3.6): Students Registration Views..... | 50 |
| Figure (3.7): Students Locations Data Mart | 51 |
| Figure (3.8): Students Locations Views | 51 |
| Figure (3.9): College Student GPA Data Mart | 52 |
| Figure (3.10): College Students GPA Views..... | 53 |
| Figure (3.11): College Transfers Data Mart | 54 |
| Figure (3.12): College Transfers Views | 55 |
| Figure (3.13): Exam Conflicts Data Mart..... | 56 |
| Figure (3.14): Exam Conflicts Views | 57 |
| Figure (3.15): Students Registration Data Mart (Student Fund) | 58 |
| Figure (3.16): Students Registration Views (Student Fund)..... | 59 |
| Figure (3.17): Cash Grant Data Mart..... | 60 |
| Figure (3.18): Cash Grant Views | 60 |
| Figure (3.19): Deferred Grant Data Mart..... | 61 |
| Figure (3.20): Deferred Grants View..... | 61 |
| Figure (3.21): Student Fund Totals Data Mart | 62 |
| Figure (3.22): Student Fund Totals Views..... | 62 |
| Figure (3.23): Financial Collection Data Mart..... | 63 |
| Figure (3.24): Financial Collection Views | 64 |
| Figure (3.25): DDL Generator in Oracle SQL Developer Data Modeler | 65 |
| Figure (3.26): Example of Loading Staging Tables | 67 |
| Figure (3.27): SCD Type 1 Before Changing a Record | 68 |
| Figure (3.28): SCD Type 1 After Changing a Record..... | 68 |
| Figure (3.29): SCD Type 2 Record Before Update | 69 |
| Figure (3.30): SCD Type 2 After Updating a Record..... | 69 |
| Figure (3.31): SCD Type 3 Example | 70 |
| Figure (3.32): SCD Type 6 After Record Update..... | 70 |
| Figure (3.33): SCD Type 1 Implementation in Talend Open Studio..... | 71 |
| Figure (3.34): SCD Type 1 Dimensions Load Process..... | 71 |
| Figure (3.35): Fact Table Loading Process..... | 72 |
| Figure (3.36): Example of a Staging Table..... | 72 |
| Figure (3.37): Data Mapper Lookups Example | 73 |
| Figure (3.38): Preparing Fact Table Data Using Data Mapper..... | 74 |

| | |
|--|------------|
| Figure (4.1): Usage Frequency Hierarchy | 81 |
| Figure (5.1): The TPC-H Schema..... | 86 |
| Figure (5.2): Relational Algebra Query Tree of Query Q1 | 89 |
| Figure (5.3): MVPP with Materialized View Nodes Selected by 2PO | 91 |
| Figure (5.4): TPC-H Usage Frequency Hierarchy | 94 |
| Figure (5.5): MVPP in 1st Week of 1st Quarter | 96 |
| Figure (5.6): MVPP in 2nd Week of 1st Quarter..... | 100 |
| Figure (5.7): High School Results - Legacy System Vs Data Warehouse | 102 |
| Figure (5.8): High School Results - Logical Vs Materialized | 102 |
| Figure (5.9): Students Registration - Legacy System Vs Data Warehouse | 103 |
| Figure (5.10): Students Registration - Logical Vs Materialized..... | 104 |
| Figure (A1.1): Students Admission - Legacy System Vs Data Warehouse | 114 |
| Figure (A1.2): Students Admission - Logical View Vs Materialized View..... | 114 |
| Figure (A1.3): Geographical Location- Legacy System Vs Data Warehouse..... | 115 |
| Figure (A1.4): Geographical Location- Logical View Vs Materialized View | 115 |
| Figure (A1.5): Transfers between Colleges - Legacy System Vs Data Warehouse | 116 |
| Figure (A1.6): Transfers between Colleges- Logical View Vs Materialized View | 116 |
| Figure (A1.7): SFD Exam Conflicts - Legacy System Vs Data Warehouse | 117 |
| Figure (A1.8): Exam Conflicts - Logical View Vs Materialized View | 117 |
| Figure (A1.9): SFD Students Registration - Legacy System Vs Data Warehouse..... | 118 |
| Figure (A1.10): SFD Students Registration - Logical View Vs Materialized View | 118 |
| Figure (A1.11): Cash Grants - Legacy System Vs Data Warehouse..... | 119 |
| Figure (A1.12): Cash Grants - Logical View Vs Materialized View | 119 |
| Figure (A1.13): Deferred Grant - Legacy System Vs Data Warehouse | 120 |
| Figure (A1.14): Deferred Grant - Logical View Vs Materialized View | 120 |
| Figure (A1.15): Student Fund Summary - Legacy System Vs Data Warehouse | 121 |
| Figure (A1.16): Student Fund Summary - Logical View Vs Materialized View..... | 121 |
| Figure (A1.17): Financial Collection - Legacy System Vs Data Warehouse..... | 122 |
| Figure (A1.18): Financial Collection - Logical View Vs Materialized View | 122 |
| Figure (A2.1): Relational Algebra Query Tree of Query Q2 | 123 |
| Figure (A2.2): Relational Algebra Query Tree of Query Q3 | 124 |
| Figure (A2.3): Relational Algebra Query Tree of Query Q4 | 125 |
| Figure (A2.4): Relational Algebra Query Tree of Query Q5 | 126 |
| Figure (A2.5): Relational Algebra Query Tree of Query Q6 | 127 |
| Figure (A2.6): Relational Algebra Query Tree of Query Q7 | 128 |

List of Abbreviations

| | |
|----------------|---|
| 2PO | Two-Phase Optimization |
| 3NF | Third Normal Form |
| BI | Business Intelligence |
| CBDMVS | Clustering-Based Dynamic Materialized View Selection |
| CBO | Cost-Based Optimizer |
| CIF | Corporate Information Factory |
| CRM | Customer Relationship Management |
| DBMS | Database Management Systems |
| DDL | Data Description Language |
| DT | Dimension Table |
| DW | Data Warehouse |
| EMVSDIA | Efficient Materialized View Selection Dynamic Improvement Algorithm |
| ER | Entity Relationship |
| ERP | Enterprise Resource Planning |
| ETL | Extract-Transform-Load |
| FT | Fact Table |
| GPA | Grade Point Average |
| IDE | Integrated Development Environment |
| IUG | Islamic University of Gaza |
| KPI | Key Performance Indicator |
| MV | Materialized View |
| MVPP | Multiple View Processing Plan |
| MVS | Materialized View Selection |
| ODS | Operational Data Store |
| OLAP | On Line Analytical Processing |
| OLTP | On Line Transaction Processing |
| PLSQL | Procedural Language/Structured Query Language |
| SCD | Slowly Changing Dimensions |
| SCM | Supply Chain Management |
| SF | Scale Factor |
| SQL | Structured Query Language |
| TPC-H | Transaction Processing Performance Council-H |
| UNM | University of New Mexico |
| VRDS | View Relevance Driven Selection |

Chapter 1

Introduction

Chapter 1

Introduction

1.1 Background

Business Intelligence (BI) systems play major role in the organizations. It is providing the ability to measure, manage, and optimize business processes. Moreover, it support decision-making of organization's managers and improve organizational performance (Ramakrishnan, Jones, & Sidorova, 2012). BI facilitates the interaction between different sections of the organization, such as human resources, marketing, and finance, to help the business to extract metrics and measurements, and assist in decision-making and knowledge extraction.

Data Warehouse (DW) is the main component of business intelligence since it forms the central repository of BI (Devlin, 2010). Data warehousing strategies involve gathering data from organization's diverse source systems, typically from multiple online transaction processing (OLTP) databases, then data manipulated for BI purposes (Eckerson, 2003).

The Islamic University of Gaza (IUG) has more than 18,000 regular students in each academic semester (Al-Kordi, 2017). It has more than 120 academic programs which include about 1000 courses that are available for students (Shwede, 2017). IUG manages the academic and administrative operations of the students and the employees cooperating different departments in the campus such as finance, admission and registration, student affairs, and student fund. However, the electronic services in IUG was increased significantly over the years resulting huge amount of data in operational database. So that it must be obvious how complex it is to handle this volume of data that would have been accumulated over the years.

For the success of any organization, there must be some performance evaluation indicators which can be used in steering organizational decisions (Moullin, 2007). In IUG, there are different perspectives to define Key Performance Indicators (KPI) which can be sorted shortly as employees, students, colleges, and environment, etc. In this research, and because of time frame for our project, we focus on KPIs that will

help in decision-making for students' affairs which will help them academically and financially. We focus on students in this research since they are one of the main components in IUG, other components will be addressed in future works.

Students' success is associated with tracking their progress each semester, i.e., tracking if a student passed certain academic levels, meeting the academic needs of students, establishing and accomplishing short-term and long-term goals. Currently, only few questions like number of students served, or number of students receiving a passing grade in a course and so on determine measures taken to ensure progress of students, which are all based on data stored in transactional database. Such reports however do not enable deeper understanding, for instance, growth of a student in a given time period, comparing growth over years and factors contributing to this.

The key points of business intelligence (BI) or supporting decision-making process are the proper understanding of business users' and decision makers' requirements, and the good design and modeling historical and new data (Vizgaitytė & Skyrius, 2012). Figure 1.1 shows simplified overview of business intelligence. To extract knowledge and insights from raw data of organization's systems, business users use specific technologies, processes, tools and rules.

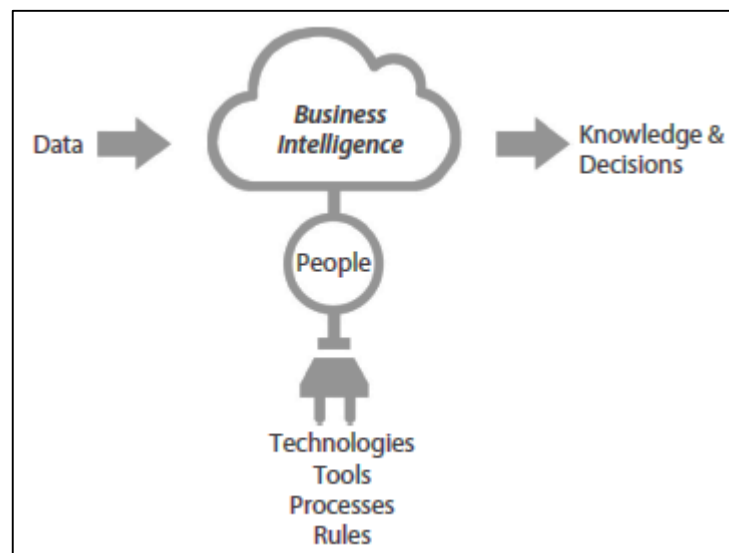


Figure (1.1): Concept of Business Intelligence

Source: (Vizgaitytė & Skyrius, 2012)

There are three stages to the business intelligence process in which a business gradually grows in analytical sophistication as business needs and demands (IBM Business Analytics, 2011). First, is IT-centric that totally focused on data collection and analytical tools selection. Second is to build a data model to support the reporting requirements of users which will help them track progress and success. Last is to build a predictive data model that will help the policy and decision makers by providing a better vision for the purpose of decision making.

The Islamic University of Gaza currently depends on logical views to build reports or to serve its applications that were developed for the academic or administrative offices. But in practice, this concept is not efficient enough to support analytics that need to be developed and to extract historical behavior of data change.

Reports that are built on top of logical views take a long time to run as views are not physically populated in the database, thus the data is gathered and populated in the time that report is requested.

For illustration, Figure 1.2 shows total time consumed to query total number of registered students for each of the last three semesters by logical view compared with total time consumed by materialized view (physically populated).

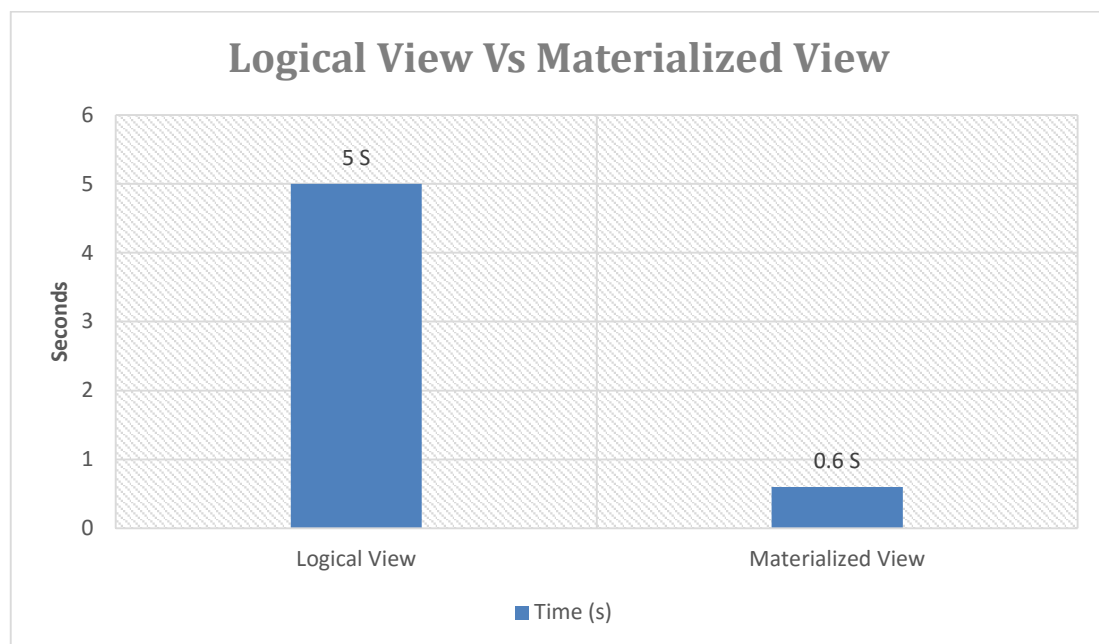


Figure (1.2): Difference between Logical View and Materialized View

It is obvious from Figure 1.2 that the time needed to run the materialized view is much less than that needed for logical view. The big difference in time was due to nature of logical views which need to be queried and calculations need to be performed on the fly. This process of querying an un-indexed view with hundreds of columns and performing calculations while generating the report consumes an undesirable amount of time.

1.2 Scope and Objectives

The scope of this thesis is to design and implement a robust and fast data warehouse for the Islamic University of Gaza. In current phase, it will hold analytical and historical data related to IUG departments that deal directly with students and help them academically and financially. These departments are Admission and Registration, Finance, and Student Fund departments. Other departments in IUG will be considered in future phases of the project. Our objectives include:

- Analyze data provided by following departments: Admission & Registration, Finance, and Student Fund.
- Design data warehouse for IUG that meets the requirements of Admission & Registration, Finance, and Student Fund departments.
- Propose our model for Materialized View Selection in Data Warehouse.
- Implement the Data Warehouse using our Materialized View Selection model.
- Test our proposed model on TPC benchmark™ H.

1.3 Motivation

The decision makers in IUG don't have a unique repository for analytical and historical data of the university. They are currently depending on analyzing data on demand based on raw transactional data, which costs a lot of time and effort and may not always be accurate since it is analyzed manually by individuals. Given the importance of the information for the IUG, the decision makers were motivated to deal with the problem of time loss and data inconsistency by implementing a data warehouse and to ensure that data is available in the time it is needed. Therefore, there

is a need to design a database that supports reporting, analytical and decision-making capabilities for executive offices at IUG.

1.4 Thesis Contribution

This thesis proposes a new Materialized View Selection (MVS) model integrated with our approach of designing and implementation of data warehouse for The Islamic University of Gaza. We used one of industry standard methods in order to design the data warehouse which is Kimball method following the guides in (Kimball & Ross, 2013).

Our model for view materialization depends on view usage frequency over time to predict materialization probability. It studies the usage behavior of each view over the academic/financial year, which is the academic year in our research. Then according to usage behavior, our model detects either the view should be materialized or not. The proposed model can be integrated with other materialized views selection algorithms as will be discussed on Results and Discussion chapter.

1.5 Overview of Thesis

The rest of this thesis is organized as follows: Chapter 2 describes a literature review on data warehouse and materialized views selection approaches. Chapter 3 will be the design and implementation process. Chapter 4 describes the proposed selection model in details. Chapter 5 presents the results of implementing our model over TPC-H benchmark showing the effectiveness of our model, and we show the value of using data warehouse in academic institutes. Chapter 6 concludes our work and provides directions for future work.

Chapter 2

Literature Review

Chapter 2

Literature Review

In this chapter, we focus on providing the information and concepts of the data warehouse and business intelligence that are relevant to our research.

2.1 Data Warehouse

Data warehouse is a repository that holds integrated data from different source systems which is retrieved and collected periodically, this data is stored in dimensional or normalized data store. Data warehouse usually stores historical data which can be used for analytics or business intelligence and data mining processes. It is typically refreshed in batches in defined intervals, so it is not required to be triggered for every transaction occur in running transactional systems (Rainardi, 2008).

A data warehouse for an organization is a repository for keeping measurements about the organization. Organizations, typically, have many systems to handle their operations internally or with its customers, i.e. human resources, finance, customer care, or sales. Each of these systems has its own operational data repository and it is very complex to derive analytical results of all these repositories based on operational data stores.

According to Bill Inmon, data warehousing is a set of procedures and processes of collecting and manipulating data to be stored in a managed database. Data is processed so that to be subject-oriented, integrated, time variant, and nonvolatile for the support of decision-making (Inmon, 2002).

“Subject-oriented” means that a data warehouse focuses on the high-level entities of the business, such as employees, courses, and accounts. This is in contrast to transactional systems, which deal with processes such as student registration or payment of invoices (Chan, 1999). “Integrated” means that the data are stored in consistent formats, with consistent naming conventions, domain constraints, physical attributes, and measurements. For example, an organization may have four or five unique coding schemes for ethnicity. In a data warehouse, there is only one coding

scheme. “Time-variant” means that data are associated with a point in time, such as a semester, financial year, or pay period. Finally, “nonvolatile” means that the data do not change once they are entered into the warehouse.

2.2 OLAP vs OLTP

Data warehouse also can be referred as Online Analytical Processing (OLAP) system since it helps decision makers in the data analysis scope. On the other hand, operational systems can be referred as Online Transaction Processing (OLTP) systems since they are handle daily organization’s transactions and processes. The difference can be summarized as OLTP system’s goal is to get data into databases, whereas the data warehouse is built to get information out of database.

In (Han, Kamber, & Pei, 2012) the researchers debate that OLTP system is customer-oriented, in contrast a data warehouse is market-oriented. It is not a good practice to combine both OLAP and OLTP tasks in one system, because OLAP database is optimized for querying and OLTP database is optimized for real time transactions. Data warehouse is based on dimensional design model which is much more effective for data retrieval while OLTP systems is based on relational design model. Furthermore, data warehouses may hold data from diverse of data sources with heterogeneous data formats which is manipulated before it is loaded in data warehouse. Organizations, typically, choose to separates OLTP and OLAP systems.

In (Poe, Klauer, & Brobst, 1996), the authors show that data in OLAP systems is used to extract knowledge and information through comparisons or by detecting patterns and trends. Such information cannot be easily discovered from OLTP system due to complexity. Furthermore, they argue the idea of data warehouse should be based on business requirements. Ralph Kimball (Kimball, 1996) agrees with that concept. Inmon in (Inmon, 2002) has a different approach for data warehouse development. He argues that despite OLTP systems are developed based on business requirements, data warehouse system is developed, then data marts are designed to cater individual business process needs. Inmon considers the data warehouse development lifecycle as data-driven and OLTP are requirements driven. On the other hand, Kimball approach

considers data warehouse development lifecycle is requirements-driven as well. In the next section, we discuss each approach in detail.

Santos and Ramos define data warehouse as a database that is managed independently of an operational database (OLTP) (Santos & Ramos, 2009). Table 2.1 illustrates that both types of databases serve different purposes. The operational database aims to handle near real time transactions (i.e. credit cards transactions) while the data warehouse is more dedicated for analyzing a bigger volume of data. All these differences must be taken into account in the process of selecting a new database.

Table (2.1): Differences between OLTP Database and OLAP Database

| OLTP Database | OLAP Warehouse |
|-------------------------------------|---|
| Operational Purposes | Records history |
| Read/Write access | Read-only access |
| Pre-defined transactions access | Periodic reports and ad hoc access |
| Access to a small amount of records | Access to a huge amount of records |
| Refresh of data near real time | Scheduled data loads |
| Optimized structure for updates | Optimized structure for processing issues |

Source: (Santos & Ramos, 2009)

2.3 Data Warehouse Design Approaches

Bill Inmon and Ralph Kimball are two pioneers that built two different approaches for data warehousing. In this section, we explain each approach and we show the differences between them.

2.3.1 Bill Inmon Architecture

Bill Inmon proposed a top-down approach for data warehousing shown in Figure 2.1, known as corporate information factory (CIF) (Inmon, 2002). The components of a CIF include a data warehouse which is built in third normalized form and individual de-normalized data marts which are populated from the data warehouse. These data

marts cater to individual business process needs. Reporting cubes are built as required on top of the data marts.

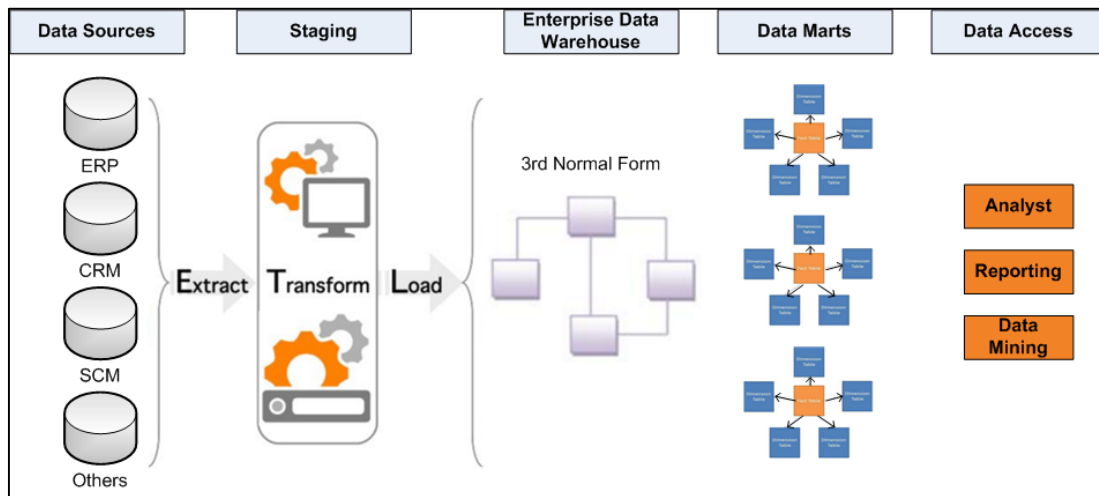


Figure (2.1): Bill Inmon Approach of Data Warehousing

Source: (Zahra, 2015)

In Figure 2.1, feeding the data warehouse starts by Extract, Load, and Transform (ETL) process from different sources. First, data is extracted and transformed in the staging area. Next, after data had been ready it is loaded into centralized enterprise data warehouse in 3rd normal form (3NF). Finally, once data is loaded, data marts are designed according to business requirements. The data warehouse as defined by Bill Inmon, contains enterprise data without any redundancy at the lowest level of detail i.e., transactional data in 3NF.

(Rangarajan, 2016) summarizes the advantages and disadvantages of Inmon's approach as follows:

The key advantages of the Inmon approach are:

- The data warehouse truly serves as the single source of truth for the enterprise, as it is the only source for the data marts and all the data in the data warehouse is integrated.
- Data update anomalies are avoided because of very low redundancy. This makes ETL process easier and less prone to failure.

- The business processes can be understood easily, as the logical model represents the detailed business entities.
- Very flexible, as the business requirements change or source data changes, it is easy to update the data warehouse as one thing is in only one place.
- Can handle varied reporting needs across the enterprise.

Here are some of the disadvantages of Inmon method:

- The model and implementation can become complex over time as it involves more tables and joins.
- The need for resources who are experts in data modeling and of the business itself. These type of resources can be hard to find and are often expensive.
- The initial set-up and delivery will take more time, and management needs to be aware of this.
- More ETL work is needed as the data marts are built from the data warehouse.
- A fairly large team of specialists need to be around to successfully manage the environment (Breslin, 2004).

2.3.2 Ralph Kimball Architecture

The second approach presented by Ralph Kimball (Kimball & Ross, 2013), known as dimensional data warehouse architecture, considered a bottom-up design as shown in Figure 2.2.

In this approach business users have a simple dimensional structure at first, and when combined together it will create a broad Data Warehouse. The key factor of dimensional modeling is the simplicity. Kimball and Ross in (Kimball & Ross, The Data Warehouse Toolkit: The definitive guide to dimensional modeling, 2013) argue that using dimensional modeling in data warehouse makes data easier to understand even for business users not only for data warehouse experts.

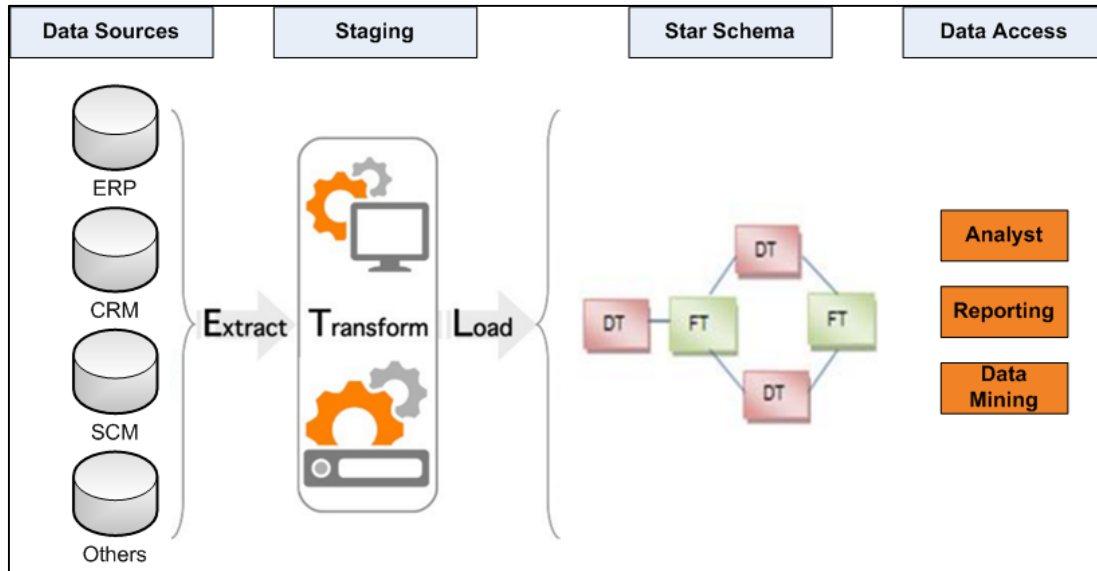


Figure (2.2): The Ralph Kimball Approach

Source: (Zahra, 2015)

Since Kimball approach is bottom-up approach, data warehousing in Figure 2.2 starts by ETL process for one or more data marts of business procedures. Data first extracted from different sources of organization systems, then it is transformed in the staging area for each data mart separately. Next, it is loaded to corresponding fact/dimension tables in the data mart. Finally, reports are designed to fetch data directly from data marts.

The major idea of Kimball approach is to build the data warehouse incrementally over time by combining data marts.

(Rangarajan, 2016) summarizes the advantages and disadvantages of Inmon's approach as follows:

Here are some of the advantages of the Kimball method:

- Quick to set-up and build, and the first phase of the data warehousing project will be delivered quickly.
- The star schema can be easily understood by the business users and is easy to use for reporting. Most BI tools work well with star schema.

- The foot print of the data warehousing environment is small; it occupies less space in the database and it makes the management of the system fairly easier.
- The performance of the star schema model is very good. The database engine will perform a 'star join' where a Cartesian product will be created using all of the dimension values and the fact table will be queried finally for the selective rows. This is known to be a very effective database operation.
- A small team of developers and architects is enough to keep the data warehouse performing effectively (Breslin, 2004).
- Works really well for department-wise metrics and KPI tracking, as the data marts are geared towards department-wise or business process-wise reporting.
- Drill-across, where a BI tool goes across multiple star schemas to generate a report can be successfully accomplished using confirmed dimensions.

Here are some of the disadvantages of the Kimball method:

- The essence of the 'one source of truth' is lost, as data is not fully integrated before serving reporting needs.
- Redundant data can cause data update anomalies over time.
- Adding columns to the fact table can cause performance issues. This is because the fact tables are designed to be very deep. If new columns are to be added, the size of the fact table becomes much larger and will not perform well. This makes the dimensional model hard to change as the business requirements change.
- Cannot handle all the enterprise reporting needs because the model is oriented towards business processes rather than the enterprise as a whole.
- Integration of legacy data into the data warehouse can be a complex process.

2.3.3 Differences between Inmon and Kimball Approaches

Now in the case of Inmon's approach of data warehousing, the architecture suggests that a 3NF data warehouse to be built as the first step, which would contain all the data in the organization, and then build a data marts layer to support the reporting layer based on data warehouse architecture.

In the case of Kimball's approach however, the idea is to build the data marts layer right after the staging layer. These data marts cater to individual business processes identified. All the data marts together then form the data warehouse as defined by Kimball. The common dimensions between the business processes are however shared between them, without building a separate version of it, to maintain a single version of truth and make it simple to update. These are called conformed dimensions. Using this approach, we do not need a second staging layer and since the data marts are specific to a business process, reports can be generated out of it, without waiting for rest of data marts to be designed and implemented.

2.3.4 Deciding Factors

Now that we have seen the pros and cons of the Kimball and Inmon approaches, a question arises. Which approach should be used when? This question is faced by data warehouse architects every time they start building a data warehouse. Here are the deciding factors that can help an architect choose between the two:

Reporting Requirements – If the reporting requirements are strategic and enterprise-wide and integrated reporting is needed, then Inmon works best. If the reporting requirements are tactical and business process/team oriented, then Kimball works best.

Project Urgency – If the organization has enough time to wait for the first delivery of the data warehouse (for example, 4 to 9 months), then Inmon approach can be followed. If there is very little time for the data warehouse to be up and running (for example, 2 to 3 months) then the Kimball approach is best (Breslin, 2004).

Future Staffing Plan – If the company can afford to have a large sized team of specialists to maintain the data warehouse, then the Inmon method can be pursued. If the future plan for the team is to be small, then Kimball is more suited.

Frequency of Changes – If the reporting requirements are expected to change more rapidly and the source systems are known to be volatile, then the Inmon approach works better, as it is more flexible. If the requirements and source systems are relatively stable, the Kimball method can be used.

Organization Culture – If the sponsors of the data warehouse and the managers of the firm understand the value proposition of the data warehouse and are willing to accept long-lasting value from the data warehouse investment, the Inmon approach is better. If the sponsors do not care about the concepts but want a solution to get better at reporting, then the Kimball approach is enough.

Design Approach for IUG

Considering reasons like reports in IUG are business oriented, time-frame of the project is limited, stability of the requirements and the ability to deliver reports quickly since IUG is established for 3 decades till now, it has been decided that the Kimball methodology would be used for designing a data warehouse for IUG.

2.4 Dimensional Model

With the rapid growth in data and with high insistence to analyze and understand it to derive some information out of it, the historical data must be stored in a form that it can be analyzed quickly. It is also important to extract some important statistics and various KPIs of the business. But this cannot be effectively implemented using the entity relationship models of operational data sources (ODS). The data needs to be reorganized into a dimensional model.

As defined in (Ballard, Farrell, Gupta, Mazuela, & Vohnik, 2006), the dimensional modeling approach facilitates generating analytical reports by improving its performance. Dimensional models provide remarkable better performance than Entity Relationship (ER) models especially for large queries. Furthermore, dimensional models are easier to understand since each model designed for a defined subject. It formed by two main components, a fact table which holds the subject's measures, and a set of tables referred to the fact table which hold descriptive data about the fact dimensions. Figure 2.3 visualize the concept. The FACT table in the center contain foreign keys to other dimension tables which are PRODUCT, CUSTOMER, REGION, and TIME and also it has Sales and Profit facts. Finally the fact table and dimensions forms a star schema which is described in Section 2.3.4.

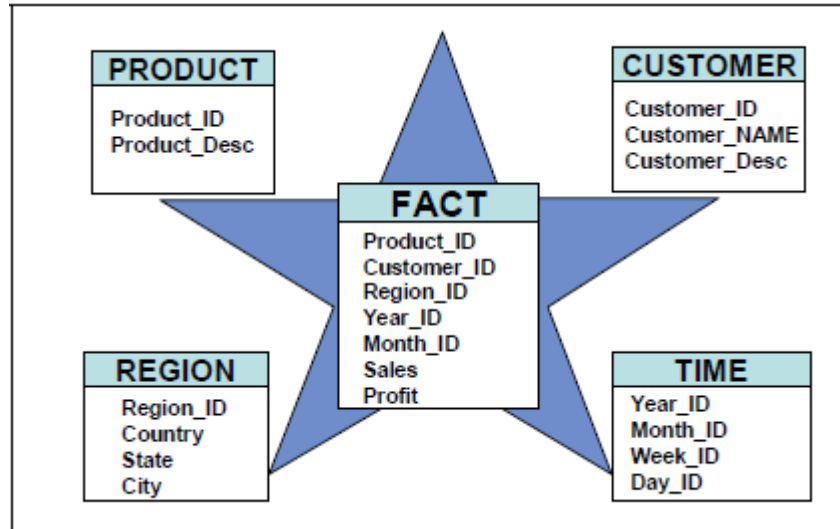


Figure (2.3): Example of Dimensional Model

Source: (Ballard, Farrell, Gupta, Mazuela, & Vohnik, 2006)

The benefits of dimensional modelling technique can be summarized as follow:

- **Better data navigation and presentation:** Data is modeled in a way that is easy to understand even for business users. This enables them to easily design their own reports.
- **Easy and low-cost maintenance:** The data is stored in the same way as it is presented unlike in the case of relational databases where in most cases views are built in order to build any reports. This increases the maintenance cost in the case of relational databases or ODS.
- **Better performance:** Most reports require summarized data which results in a slower performance due to on-the-fly calculations in the case of non-indexed views in ODS. In the case of dimensional modeling, summarized tables are built as required. It also allows the ability to store data history in a manner that is easy to query and build reports on. Such a design delivers faster query performance and to drill down and drill across hierarchies.

On the other hand, in ER model tables are normalized in a form where there is no data redundancy called Third Normal Form (3NF). Tables in 3NF have attributes where all are related to a primary key in the entity. A primary key can be just one attribute or a composite key consisting of two or more attributes.

A third normal form ER model is good design to handle transactional data. Examples for transactional data are courses taken by students in each term, students' admission to IUG and so on. This design is good for performing quick transactions such as inserts, updates and deletes because the tables usually have a small number of fields with foreign keys to other tables. This design is inefficient for reporting purpose, the reason behind this is that, to generate a simple report one would end up joining scores of tables in a 3NF environment which ends up being very time and resource (hardware) intensive. Apart from this, the model becomes very complex very quickly as a result of which it becomes very difficult to understand and navigate the model even for a developer let alone a business user.

Hierarchies

As reported by (Moody & Kortink, 2000), hierarchies are an extremely important concept in dimensional modelling, and form the primary basis for deriving dimensional models from ER models. Most dimension tables contain embedded hierarchies. A hierarchy in an Entity Relationship model is any sequence of entities joined together by one-to-many relationships, all aligned in the same direction. Figure 2.4 shows a hierarchy extracted from the example data model, with State at the top and Sale Item at the bottom.

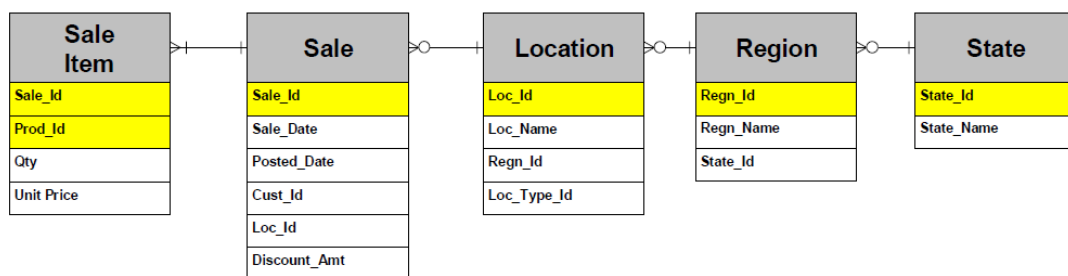


Figure (2.4): Example of Hierarchy in Dimensional Model

Source: (Moody & Kortink, 2000)

In hierarchical terminology, it is stated by (Moody & Kortink, 2000) as:

- State is the parent of Region
- Region is the child of State

- Sale Item, Sale, Location and Region are all “descendants” of State
- Sale, Location, Region and State are all “ancestors” of Sale Item

2.5 Definitions

In this section we explain common terms that are used in this thesis which related to data warehousing as defined by Kimball in (Kimball & Ross, 2013) and (Kimball & Ross, 2010):

A **Fact Table** is the container of subject’s measurements and it contain foreign keys to the dimension tables in the schema as shown in Figure 2.5. Students Payment Fact in the figure has three measurements which are total transactions number, students count, and total payment amount. These measurements are calculated for each academic semester, department, and teller which are the dimensions for the fact table.

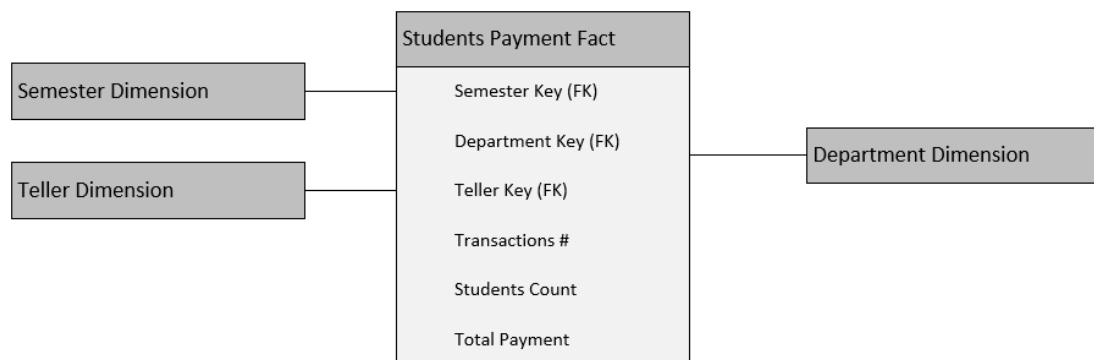


Figure (2.5): Fact and Dimension Tables in a Dimensional Model

Dimension Tables are integral companions to a fact table. It usually has fewer record than fact tables, but can be wide with many large text columns. Dimension tables describe the “who, what, where, when, how, and why” associated with the facts in fact table. Every dimension table has a single primary key column. This primary key is embedded as a foreign key in any associated fact table.

Conformed Dimensions are standardized tables modeled once and shared across multiple fact tables in the same schema or even a different data mart. The main advantage of using conformed dimensions is to save storage space. It is also easier to maintain and refresh one table versus multiple versions of the same table.

The **Grain** is the lowest level of detail in a fact or dimension table. For example, a dimension such as Date (with Year and Quarter hierarchies) has a granularity at the quarter level but does not have information for individual days or months. Alternately, a Date dimension table (with Year, Quarter, and Month hierarchies) has granularity at the Month level, but does not contain information at the day level (IBM Knowledge Center, 2017).

A **Factless Fact Table** is a fact table which does not contain a measurement but a set of dimensions' references. Usually it is used to capture an event. For example, Figure 2.6 illustrates a factless fact table about student's attendance of an admission event such as a high school visit, college fair, alumni interview or campus overnight.

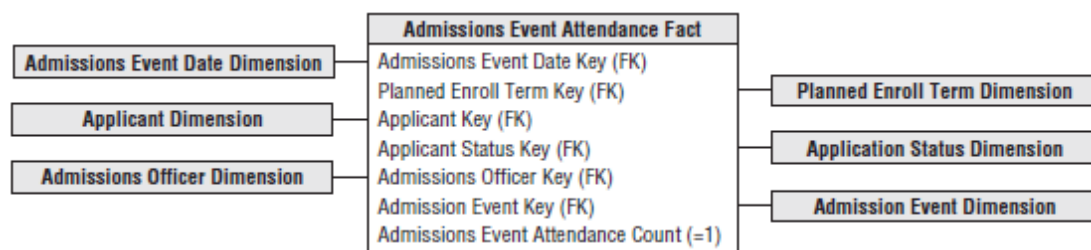


Figure (2.6): Factless Fact Table Example

Source: (Kimball & Ross, 2013)

The **Business Key** or **Natural Key** identifies a business entity. Examples include student_id, course_id and program_id.

The **Primary Key** uniquely identifies a record in a table. A primary key can consist of a single field or multiple fields and cannot be a NULL value.

The **Foreign Key** is a single field or multiple fields which uniquely identifies a record in another table.

The **Surrogate Key** uniquely identifies a record in a dimension table. It is usually ETL generated and provides the means to maintain data warehouse information when dimensions change. One simple way improve performance of queries is to use surrogate keys. Surrogate keys can be derived from the existing natural keys or it can be a simple integer. As an example, a surrogate key can be a

composite key, being the combination, student_id + academic_period or just an integer value generated by the ETL program while a record is being inserted the table. Using integer surrogate keys means a thinner fact table and the thinner the fact table, the better the performance.

The **Star Schema** is a dimensional design for a relational database. In a star schema, related dimensions are grouped as columns in dimension tables, and the facts are stored as columns in a fact table. The star schema gets its name from its appearance: when drawn with the fact table in the center, it looks like a star or asterisk.

The **Snow Flake** schema is a variation on the star schema. When principles of normalization are applied to a dimension table, the result is called a snow flake schema.

The **Multiple View Processing Plan (MVPP)** is a directed acyclic graph which presents the processing plan of a set of analytical queries

2.6 DW Case Studies for Higher Education Environments

In (Leonard, 2011), the researcher designed and implemented a small data warehouse as proof of concept of using data warehouse in higher education environment. He studied the components of a theoretical, small scale, and downsized enterprise data warehouse within the context of higher education environment.

However, another thesis (Ganapavarapu, 2014) argues the need of data warehouse for higher education institutes. The researcher adopted Kimball's approach to design small prototype of a data warehouse for University of New Mexico (UNM). He started by designing DW phase, then go through implementation and ETL process. Finally, he illustrated the effect of using DW in UNM by comparing the consumed time of processing UNM reports in both ODS and DW databases.

(Ribeiro, 2016) analysis NOVA Information Management School academic service's data to design a data warehouse, the purpose of its DW is to discover information from data. The researcher also adopted Kimball's approach in his thesis through designing process.

In (Suknović, Čupić, Martić, & Krulj, 2005), the authors designed simple data warehouse for University of Belgrade students' services. They did not define their approach through designing process (Inmon/Kimball). The second part of their research is about data mining and knowledge extraction

2.7 Materialized View Selection Algorithms

Materialized views (MVs) comprise pre computed and summarized information with the aim of answering most queries posed on data warehouse thereby saving of query processing time and storage. With the increase of attributes in each dimension there is need of increase of pre calculation of MVs. In this regard there is the increase of work load, needs to decrease the response time and storage. There are many view selection algorithms proposed. On the other hand, continuous updating in the base table have to be reflected in the dimension table. As the information in DW is in the form of Facts, it is not possible to change whole DW instead changes to be accommodated at only affected part of DW. For this updating many views maintenance algorithms are proposed.

The Multiple View Processing Plan (MVPP) based algorithms are one of the significant solutions to address the materialized view selection problem in data warehouses. A heuristic algorithm has been designed based on MVPP in (Yang, Karlapalem, & Li, 1997), which find a solution based on individual optimal query plans. This approach depends on analyzing the queries to derive common intermediate results which can be shared among the queries. The MVPP helped in this design to select a set of views to materialize by reusing temporary results from the execution of global queries.

Gupta proposed a theoretical framework for materialized view selection, and illustrated the cost model of materialized view selection under the space constraints using greedy algorithm. The cost model is designed to achieve minimum query response time and minimum view maintenance cost (Gupta, 1997). It checks a small part of the space, to make the views meet the space constraints and the time requirements, but the performance of this method is bad.

(Himanshu & Mumick, 1998) developed algorithms to select a collection of views in data warehouse to materialize so as to minimize the total query response time under the constraint of a given total view maintenance time .

Another model for selecting views to materialize based on candidate selection and enumeration techniques is presented in (Agrawal, Chaudhuri, & Narasayya, 2000) by a team of Microsoft researchers. This approach selects materialized views and indexes by searching over the reduced space of candidate materialized views at a fraction of enumeration cost.

Another approach to handle materialized view selection is the use of evolutionary algorithms that based on multiple global processing plans of queries. Lee et al. proposed a solution based on genetic algorithm to solve maintenance-cost view selection problem assuming unlimited amount of storage space (Lee, 2001). This solution computes a near optimal set of views that used to search for a near optimal solution. However, the performance of selection process is slow.

Zhang' et al. show an evolutionary algorithm which combined with heuristic algorithm to preserve gains of both methods and achieve better performance than each of them individually (Zhang, Yao, & Yang, 2001). This approach shows that applying an evolutionary algorithm, to either global processing plan optimization or materialized view selection for a given global processing plan, can reduce the total query and maintenance cost significantly.

(Valluri, Vadapalli, & Karlapalem , 2002) proposed the definition of view correlation and view correlation matrix, and they also proposed the costs models and algorithms of view correlation, which based on that one view selection may affect the interests of other views, thereby affect the total query cost and maintenance cost. Additionally, they designed View Relevance Driven Selection (VRDS) algorithm for materialized view selection to minimize total processing cost including query processing and view maintenance cost, this selection algorithm is based on AND-OR Graph. This algorithm performed better than greedy algorithms and MVPP based heuristic algorithms when there is space constraint and update frequency is high.

Panos Kalnis et al. in (Kalnis, Mamoulis, & Papadias, 2002) proposed the application of randomized search heuristics. They adapted random sampling, iterative improvement, simulated annealing and two-phase optimization to find fast a sub-optimal set of views under space or maintenance cost constraints. The proposed method provided near-optimal solutions in limited time, being robust to data and query skew.

Yu et al. presented a new constrained evolutionary algorithm for the maintenance-cost view-selection problem (Yu, Yao, Choi, & Gou, 2003). The proposed algorithm is based on constraint handling technique which is stochastic ranking procedure. They evaluated their algorithm against both heuristic and evolutionary algorithm and showed that the proposed algorithm can significantly provide better solution in term of minimization of query processing cost.

In (Wang & Zhang, 2005), they proposed a modified genetic algorithm for the selection of a set of views for materialization so that the sum of query processing cost and view maintenance cost is minimized.

A general AND-OR view graph was considered in (Gupta & Mumick, 2005). They have presented greedy polynomial-time heuristics to minimize the total view maintenance cost of selected views to be materialized under a disk space constraint.

Kamel Aouiche et al. proposed clustering based materialized selection solution. This algorithm uses workload approach (Aouiche, Jouve, & Darmont, 2006). The researchers exploit a query clustering involving similarity and dissimilarity measures defined on the workload queries, in order to capture the relationships existing between the candidate views derived from this workload. These candidate views are merged to resolve multiple queries. This research was a proof on concept to show that the idea of using data mining techniques for data warehouse auto-administration is a promising approach.

In (Gou, Yu, & Lu, 2006) an efficient materialized view selection approach under disk space constraint has been proposed, it uses A* algorithm to find better set

of views. Unfortunately, the proposed approach works great only when the space constraint is low.

ASVMRT (Algorithm for Selection of Views to Materialize using Reduced Table) was presented in (Yang & Chung, 2006). This approach use clustering method to reduce tables in the data warehouse based on attribute-values density and considered the combination of reduced tables as materialized views instead of a combination of the original tables. As a result of this algorithm, a faster computation time, reduced storage space, and better performance than former algorithms were reported. However, maintenance cost of reduced table was not considered in this approach.

Jiratta Phuboon-ob and Raweewan Auepanwiriyaikul in (Phuboon-ob & Auepanwiriyaikul, 2007) proposed a two phase optimization (2PO) method which was a combination of simulated annealing and iterative improvement with the use of MVPP. Their experiments show that 2PO outperform the original algorithms in terms of query processing cost and view maintenance cost.

Simulated annealing is used in another study for materialized view selection (Derakhshan, Stantic, Korn, & Dehne, 2008) in which MVPP is considered as input. The researchers show that parallel simulated annealing has been resulted better quality of the selected views set to be materialized and a significant improvement in query processing cost and view maintenance cost. However, in some cases this approach trapped in local minimum.

Gang Zhao proposed the CBDMVS algorithm (Clustering-Based Dynamic Materialized View Selection Algorithm) which utilize clustering technique to materialized views, then dynamically adjusts materialized view set (Gong & Zhao, 2008). In this approach, similarity function is used for clustering. Then these materialized view clusters are dynamically adjusted. The algorithm selects the materialized view set which has relatively higher frequency responses performance to variety types of query. In this algorithm when updating is done only to the required materialized views but not whole MV set, which greatly reduces the computational cost.

A dynamic approach EMVSDIA (Efficient Materialized View Selection Dynamic Improvement Algorithm) is proposed in (Lijuan, Xuebin, Linshuang, & Qian, 2009). EMVSDIA is a two-phase algorithm which dynamically select a set of views to materialize online. This approach has provided reduced search space and time consumption.

Yogeshree D. Choudhari and Dr. S. K. Shrivastava proposed the cluster based approach for selection of materialized views (Choudhari & Shrivastava, 2012). The procedure uses the clustering of the views. This algorithm uses the record generator. Then System finds set of all possible queries resolved on generated records. Then based on the access frequency, set of queries are optimized. Further using the cluster area and threshold, the MV's are made. These are divided further into three types 1) Single query to Multi table MV. 2) Single query to single table MV. 3) Multiple queries to single table MV. This framework decreases the query response time.

Another dynamic approach to view selection has been proposed in (Suchyukorn, 2013) which have determined the existing materialized views that are affected by adding new queries rather than all existing resources and so have re-optimized MVPP and have improve the total query processing cost of it.

In (Suchyukorn & Auepanwiriyaikul, 2013), the researchers have used merging of incoming query as the global common sub-expressions of the previous merging to avoid a huge search space which some combination would not be considered. Since the global optimization is not acquired in this method, they rewrote some queries by using common sub-expression among queries to gain more optimal query processing cost.

From the above mentioned works, it is found that the research works have been provided different approaches for the selection of views to materialize considering view maintenance cost and storage space. There is lack in researches in selection of views to be materialized based on dynamic threshold so far. Thus in this thesis, on Chapter 4, we focus on designing a novel approach for dynamic selection of views to be materialized based on views' usage frequencies.

In the next chapter, we will show in detail the steps of designing and implementing the data warehouse for the Islamic University of Gaza.

Chapter 3

Design and Implementation

Chapter 3

Design and Implementation

This chapter focuses on the process of designing and implementing the data warehouse for the Islamic University of Gaza. We explain the approach we have followed in the process of building the data warehouse. The data warehousing process will be based on four main phases of software development which are: system analysis, system design, system development, and system verification and maintenance. This chapter explains each phase in detail.

We used two types of data gathering methodologies to define business requirements: Introspection and Interviews.

Introspection: we identified the common functionalities and requirements for the project based on our experience and knowledge, these helped us to proceed to the next level.

Interviews: we made many interviews with many directors and decision makers in the IUG. In these interviews we discussed the derived business requirements from our introspection. Additionally, we gathered the stakeholders' specific requirements to recognize their needs for better business intelligence experience, which will help them in proper decision making.

3.1 System Analysis

Analysis is the process of studying a procedure in order to identify its goals and purposes and to create systems and procedures that will achieve them in an efficient way. In this phase, we analyzed IUG transactional data which has been accumulated over the years. The overall goal of this phase is to understand the main modules of the IUG platform and to identify all business users who need to access the data warehouse.

Also, the business analysis in this phase helps to understand the relations between the users and the business operations. In this phase, data in which users are currently uses is identified, and how they would like to use it. System analysis depends

on the feedback of business users which will resulting business entities, relationships between the entities, and hierarchies.

In our case, analysis involved a detailed study of the current procedures of reporting in IUG, leading to specifications of the new system. During the analysis, we studied the procedures of admission & registration, student fund, and finance departments to design its data marts for data warehouse. Interviews and Introspection are the base tools used for system analysis.

The business requirements were gathered based on many interviews with the different business users. Answers from these users will generate the requirements needed for further development of the data warehouse.

3.1.1 IUG Data

The data available within the IUG is very huge since it was accumulated over years. IUG need to make best use of this data to make fast and right decisions in right time. In our research we focused on providing a data warehouse solution, which will help mainly in decision-making process related to the students. Within the scope of this project, we looked at developing data mart for some of the department that is core for students. They are the Admission & Registration, Student Fund, and Finance departments.

3.1.2 Functional Requirement

In general, requirements are partitioned into functional requirements and non-functional requirements.

Functional requirements are associated with specific functions, tasks or behaviors the system must support, it can be in any format but has to be in line with the business requirements.

To determine the functional requirements for this project, we asked interviewees to show us their current user experience of generating the reports. Additionally, we noted their comments to get better user experience, performance, and finally business intelligence. The identified functional requirements are stated below; the function

requirement can be reviewed at different stages of the project in order to cater for new discoveries during the project design.

Admission & Registration Requirements

- The users need to be able to analyze high school students' results according to the succeeded students' percentage in each section.
- The users need to be able to show high school students admission percentage in each college.
- The users need to be able to query historical data about total succeeded high school students each year compared with admitted students in IUG.
- The users need to be able to analyze students' transfer transactions between colleges and departments during their study.
- The users need to be able to analyze students' registration process over semesters and to be able to compare it.
- The users need to be able to show exams conflicts over the semesters and colleges.
- The users need to be able to analyze students' performance over semesters in each college.

Student Fund Requirements

- The users need to be able to analyze students' registration according to their need levels.
- The users need to be able to query total cash grants paid to students over semesters.
- The users need to be able to analyze grants that given to students according to its type (External grants, Internal Grants, or Loans).
- The users need to be able to analyze deferred grants execution over semesters.

Finance Requirements

- The users need to be able to analyze student's financial profile
- The users need to be able to query total financial collection for each teller in IUG.
- The users need to be able to analyze financial collection over semester days and weeks for each college.

3.1.3 Non Functional Requirement

- The system should be integrated with current IUG platform
- The front end application should be web enabled and no installation is required on users' system
- User level permission is required in order to protect the integrity of the data and restrict user's accessibility to data
- The system should perform very well at all times and should be easy to recover after system down time.
- The system should be able to keep up to-date information at all time.

3.1.4 User Requirement

- Ability to generate report with little effort
- Ability to get the aggregate report and drill down for further details
- Ability to export data in any format which let them able to manipulate it.
- System reliability at all time.

3.1.5 System Requirement

For the purpose of this project, we looked at using Microsoft Windows Operating system. The main application, we will be using ORACLE 11g as a database engine which is an object-relational database management system produced and marketed by Oracle Corporation. The application has the relational database management system that is capable of storing all the data required for the data warehouse. The process of ETL is built using Talend Open Studio which is a software integration vendor. Talend Open Studio was the first commercial open source software vendor of data integration software.

The front end application for this project would be the BI Publisher from Oracle Corporation. It can be used to design interactive reports according to user's request and can also design different gauges and dashboards. It has the capability to design different charts and graphs. After designing reports, it can be easily integrated with IUG platform using BI Publisher web services.

To run the above application, the operating system would be from Windows 2012 and above, minimum of 8 GB memory, including the data 100GB of hard drive space is required. The speed of processor could be from 3.0 MHz duo core. The other system unit components are required to support the operating system and the application for this project.

3.2 System Design

In this phase, the purpose is to obtain the system specifications from user requirements which are gathered from stakeholders. Some of data warehousing components are built based on these specifications such as data extractor, data transformation, and data integration tools. System specifications are based on logical and physical design of data marts in the data warehouse.

As discussed previously in the literature review, there are two approaches for data warehousing which are Bill Inmon and Ralph Kimball approaches. We have adopted the Ralph Kimball approach. The requirements collected from each department is translated to data marts, and the resulting data marts presents the final version of data warehouse.

A Development Methodology describes the expected evolution and management of the engineering system. One of the most important principles of Systems Engineering is evaluating a system from a Life-Cycle perspective. Establishing a methodology will also provide a strategy for the project manager and the project team as they execute the data warehouse project throughout all phases of development (Burton & Green, 2016).

3.2.1 Waterfall Model

The waterfall model is a linear sequence comprised of the following basic stages:

- Requirements Definition
- System Design
- Detailed Design
- Integration and Testing

- Operations and Maintenance

This model is used when the system requirements and objectives are known and clearly specified.

3.2.2 Spiral Model

The Spiral model is a sequence of waterfall models which corresponds to a risk oriented iterative enhancement, and it recognizes that requirements are not always available and clear when the system is first implemented.

Since designing and building a data warehouse is an iterative process, the spiral method is the best development methodology for our purpose. Figure 3.1 shows one waterfall series in a recommended spiral model of a data warehouse life-cycle.

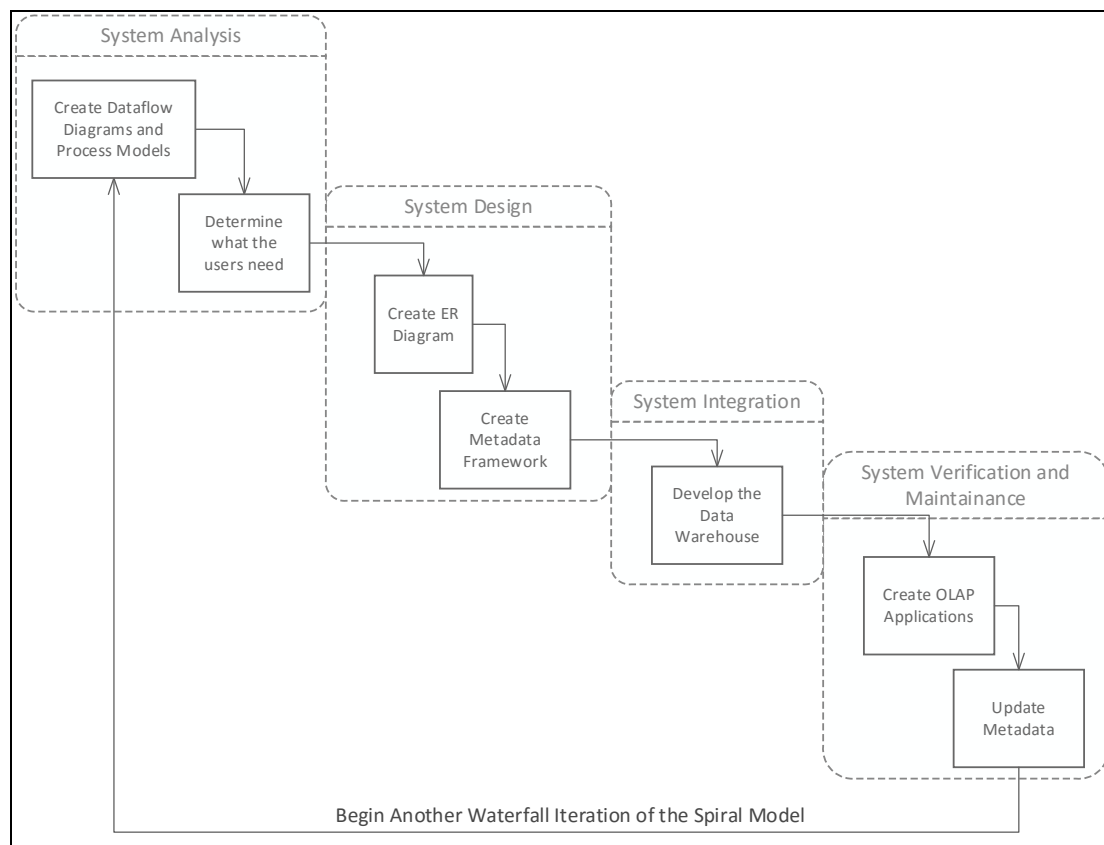


Figure (3.1): Spiral Model of the Data Warehouse Life-cycle

Source: (Burton & Green, 2016)

We start our data mart design by specifying the measure, the measures are the foundation and feedback information that the decision makers require. We reconcile these requirements with what is available in the source system (OLTP). For the purpose of this project, we used the star schema for the data warehouse design. The star schema is a relational database schema used to hold measures and dimensions in a data mart. The measures are stored in a fact table and the dimensions are stored in dimension tables. For each data mart, there is only one fact table surrounded by the dimension tables, hence the name star schema.

As mentioned in literature review Figure 2.3, the center of the star is formed by the fact table. The fact table has a column for the measure and the column for each dimension containing the foreign key for a member of that dimension. The key for this table is formed by concatenate all of the foreign key fields. The primary key for the fact table is usually referred to as composite key. It contains the measures, hence the name “Fact”

The dimensions are stored in dimension tables. The dimension table has a column for the unique identifier of a member of the dimension, usually an integer of a short character value. It has another column for a description.

One of the most important parts of the data warehouse is the extracting, transforming and loading of data from the operational transactional databases to the data warehouse itself to make best use of the data. The main two processes of data warehouse are data load and data access. The design of the system was very robust in order for the aim to be achieved. The loading of the data warehouse was done through the use of ETL process, Figure 2.2 show data warehouse architecture that chosen for this project.

The design of the databases started with the principle and theories of database design and the rule that support business need. We started the process of data warehouse design with the logical design.

3.2.3 Logical Models

The logical model is a representation of the data in a way that can be presented to the business to serve as a road map for the physical implementation. The main elements of a logical model are entities, attributes, and relationships. We started the design of the data marts through the fact and dimension tables. All database design starts with logical design.

3.2.4 Facts and Dimensions Tables

Fact table contains the measurements associated with a specific business process. A record in a fact table is a measurement, and a measurement event can always produce a fact table record. These events usually have numeric measurements that quantify the magnitude of the events. These numbers are called facts; they are also referring to as measure in the analysis services.

Dimensions are the foundation of the dimensional model, describing the objects of the business such as student, college, course and other dimension table to be used in this design of the data mart.

According to Ralph Kimball the dimension serves as the nouns of the DW/BI system. They describe the surrounding measurement events. The business processes (facts) are the action of the business in which the dimension participates. Each dimension table links to all the business processes in which it participates

Data marts represent a unit or departmental process within an organization. Data mart is the collection of fact table and its dimension tables. Using the bottom up data warehouse design, combination of the data mart would form the data warehouse.

According to our system analysis, the final data warehouse design has 11 fact tables and 15 dimension tables. Many dimension tables are shared between different fact tables. In this section we describe dimension table first, then we illustrate each of data marts in data warehouse. In all following tables we show columns names, data type of each column, and the description and purpose of each one.

Academic Years Dimension (ACD_ACADEMIC_YEAR_DIM)

This dimension contains simple data about all academic years of IUG, the columns are described in Table 3.1. The number next to VARCHAR data type denotes to the size of that column.

Table (3.1): Academic Years Dimension

| Column Name | Data Type | Description |
|-------------|-------------|--|
| YEAR_NO | NUMERIC | Academic year number formed of 4 digits, example: 2017 |
| YEAR_TITLE | VARCHAR(60) | Descriptive title of the academic year. |
| START_DATE | DATE | First day of the year |
| END_DATE | DATE | Last day of the year |

Academic Semesters Dimension (ACD_ACADEMIC_SEMESTERS_DIM)

Academic semesters are loaded into ACD_ACADEMIC_SEMESTERS_DIM dimension table. Table is described in Table 3.2.

Table (3.2): Academic Semesters Dimension

| Column Name | Data Type | Description |
|----------------|--------------|--|
| SEMESTER_NO | NUMERIC | Academic semester number as identified by IUG legacy system. Contains 5 digits, first 4 digits represent academic year and the last digit represent semester no (1 for first, 2 for second, 3 for third) |
| SEMESTER_TITLE | VARCHAR(100) | Descriptive title of the semester |
| YEAR_NO | NUMERIC | Academic year number formed of 4 digits, example: 2017 |
| YEAR_TITLE | VARCHAR(60) | Descriptive title of the academic year. |
| START_DATE | DATE | First day of the semester |
| END_DATE | DATE | Last day of the semester |

Date Dimension (GEN_DATE_DIM)

The date dimension provides extra data about a date. It includes information such as the full date, day of week, day name, and month name and it is a general dimension that should exist in every data warehouse. This dimension is provided by Kimball group (The Microsoft Data Warehouse Toolkit, 2nd Edition, 2017) and described in Table 3.3 below.

Table (3.3): Date Dimension

| Column Name | Data Type | Description |
|----------------------|-------------|---|
| DATE_KEY | NUMERIC | Primary key which contains 8 digits in form “yyyymmdd” of the date, example: 20170115 |
| SEMESTER_NO | NUMERIC | Foreign key to Semesters Dimension |
| FULL_DATE | DATE | Full date as date object |
| DATE_NAME | VARCHAR(11) | Date in simple format |
| DAY_OF_WEEK | NUMERIC | Day of the week. Below is the day that corresponds to each day of week: 1 – Sunday 2 – Monday 3 – Tuesday 4 – Wednesday 5 – Thursday 6 – Friday 7 – Saturday |
| DAY_NAME_OF_WEEK | VARCHAR(10) | Name of the day in week: i.e. “Saturday”, “Friday” |
| DAY_OF_MONTH | NUMERIC | Day of the month. May be a number between 1-31 |
| DAY_OF_YEAR | NUMERIC | Day of the month. May be a number between 1-366 |
| WEEKDAY_WEEKEND | VARCHAR(10) | Classifies the day as “Weekday” or “Weekend” |
| WEEK_OF_YEAR | NUMERIC | Number of the week in a year |
| MONTH_NAME | VARCHAR(10) | Month name i.e. “January” |
| MONTH_OF_YEAR | NUMERIC | Number of the month in a year |
| IS_LAST_DAY_OF_MONTH | NUMERIC | Classifies the day if it is the last day of the month (“Y”) or not (“N”) |
| CALENDAR_QUARTER | NUMERIC | Quarter number in a year |

| | | |
|----------------------------|-------------|--|
| CALENDAR_YEAR | NUMERIC | Year number |
| CALENDAR_YEAR_MONTH | VARCHAR(10) | Year and month number separated by a “-” (i.e. “1990-01”) |
| CALENDAR_YEAR_QTR | VARCHAR(10) | Year number and quarter separated by a “-” (i.e. “1990-Q1”) |
| ACADEMIC_MONTH_OF_SEMESTER | NUMERIC | Number of the academic month in a semester |
| ACADEMIC_WEEK_OF_SEMESTER | NUMERIC | Number of the academic week in a semester |
| ACADEMIC_DAY_OF_SEMESTER | NUMERIC | Number of the academic day in a semester |

Study Programs Dimension (ACD_STUDY_PROGRAM_DIM)

This dimension is very simple since it only store academic program key number and its title as described in Table 3.4.

Table (3.4): Study Programs Dimension

| Column Name | Data Type | Description |
|--------------|-------------|---|
| PROGRAM_NO | VARCHAR(1) | Academic program primary key ‘B’: Bachelor ‘M’: Master ‘D’: doctorates |
| PROGRAM_NAME | VARCHAR(60) | Program title |

Colleges Dimension (ACD_COLLEGE_DIM)

This dimension store college name, and for which academic program it is related. Additionally, it has an attribute to indicate if the college is active or closed. Table 3.5 shows the columns.

Table (3.5): College Dimension

| Column Name | Data Type | Description |
|-----------------|-------------|----------------------------------|
| COLLEGE_NO | NUMERIC | Primary key |
| COLLEGE_AR_NAME | VARCHAR(60) | College name in Arabic language |
| COLLEGE_EN_NAME | VARCHAR(60) | College name in English language |

| | | |
|-----------|------------|--|
| IS_ACTIVE | VARCHAR(1) | ‘Y’ means it is active and ‘N’ in case if the college is not available |
|-----------|------------|--|

Departments Dimension (ACD_DEPARTMENT_DIM)

This table contains department details. A department must be related to a college as shown in Table 3.6.

Table (3.6): Departments Dimension

| Column Name | Data Type | Description |
|--------------------|-------------|---|
| DEPARTMENT_NO | NUMERIC | Primary key |
| DEPARTMENT_AR_NAME | VARCHAR(60) | Department name in Arabic language |
| DEPARTMENT_EN_NAME | VARCHAR(60) | Department name in English language |
| COLLEGE_NO | NUMERIC | Foreign key to College |
| IS_ACTIVE | VARCHAR(1) | ‘Y’ means it is active and ‘N’ in case if the department is not available |

Courses Dimension (ACD_SUBJECT_DIM)

Table 3.7 shows the description of each column in Courses dimension table, this table contains courses’ related information.

Table (3.7): Courses Dimension

| Column Name | Data Type | Description |
|-----------------|-------------|---|
| SUBJECT_NO | NUMERIC | Primary key |
| SUBJECT_CODE | VARCHAR(10) | Manually identified code for subject contain 5 letters and 5 digits |
| SUBJECT_AR_NAME | VARCHAR(60) | Course name in Arabic language |
| SUBJECT_EN_NAME | VARCHAR(60) | Course name in English language |
| SUBJECT_TYPE | NUMERIC | Identify if the subject is required or optional |

| | | |
|------------------|------------|--|
| DEPARTMENT_NO | NUMERIC | Foreign key to Department |
| ACADEMIC_HOURS | NUMERIC | The academic hours in the plan |
| FINANCIAL_HOURS | NUMERIC | Financial hours to be paid for |
| SUBJECT_LANGUAGE | VARCHAR(2) | Subject language: 'AR' for Arabic and 'EN' for English |

High School Years Dimension (ACD_HIGHSCHOOL_YEARS_DIM)

In this table, we define high school years. It contains key for each year in date format “yyyy” and a descriptive title for this year as shown in Table 3.8.

Table (3.8): High School Years Dimension

| Column Name | Data Type | Description |
|-------------|-------------|--|
| YEAR_NO | NUMERIC | Primary key which formed by 4 digits of high school year |
| YEAR_NAME | VARCHAR(60) | Descriptive title of the year |

High School Grades Dimension (ACD_HIGHSCHOOL_GRADE_DIM)

The high school grades table stores some extra information about grades range of high school students. This dimension is described in detail in Table 3.9.

Table (3.9): High School Grades Dimension

| Column Name | Data Type | Description |
|-------------|-------------|--|
| GRADE_NO | VARCHAR(2) | Primary key which must be one of the following: 'A+' for grades between 95 and 100 'A' for grades between 90 and 94.9 'B+' for grades between 85 and 89.9 'B' for grades between 80 and 84.9 'C+' for grades between 75 and 79.9 'C' for grades between 70 and 74.9 'D+' for grades between 65 and 69.9 'D' for grades between 60 and 64.9 'E+' for grades between 55 and 59.9 'E' for grades between 50 and 54.9 'F' for grades less than 50 |
| GRADE_TITLE | VARCHAR(60) | Descriptive title of the grade |

| | | |
|------------|---------|---|
| FROM_GRADE | NUMERIC | Minimum grade as described in GRADE_NO column |
| TO_GRADE | NUMERIC | Maximum grade as described in GRADE_NO column |

High School Sections Dimension (ACD_HIGHSCHOOL_BRANCH_DIM)

This is a simple dimension table, Table 3.10, which store a serial key of predefined high school section in IUG legacy system, and a descriptive title i.e. “Scientific Section”.

Table (3.10): High School Sections Dimension

| Column Name | Data Type | Description |
|-------------|-------------|--|
| BRANCH_NO | NUMERIC | Primary key / serial number of section |
| BRANCH_NAME | VARCHAR(60) | Descriptive title of the section |

Student States Dimension (ACD_STUDENT_STATES_DIM)

In this dimension table, a unique status code and a name for the status are defined as described in Table 3.11.

Table (3.11): Student States Dimension

| Column Name | Data Type | Description |
|-------------|-------------|--|
| STATUS_CODE | VARCHAR(1) | Primary key / a letter which represent the student academic status i.e. ‘R’ for Regular students |
| STATUS_NAME | VARCHAR(60) | Descriptive title of the status |

Student Levels Dimension (ACD_STUDENT_LEVEL_DIM)

The table ACD_STUDENT_LEVEL_DIM, Table 3.12, contains details about academic levels in IUG. A student in IUG starts his academic life in level 1, and if he passed certain academic hours he moves to next level.

Table (3.12): Student Levels Dimension

| Column Name | Data Type | Description |
|-------------|-------------|---|
| LEVEL_NO | NUMERIC | Primary key and a number which represent student's academic level sequence, i.e. 1 for first level, 2 for second level and so on. |
| LEVEL_NAME | VARCHAR(60) | Descriptive title of the level |

Geographical Locations Dimension (ACD_NEIGHBORHOOD_DIM)

ACD_NEIGHBORHOOD_DIM table as described in Table 3.13, it lists all predefined neighborhoods in IUG system. Additionally, it links neighborhoods with its cities and governorates as shown below.

Table (3.13): Geographical Locations Dimension

| Column Name | Data Type | Description |
|-------------------|-------------|--|
| NEIGHBORHOOD_ID | NUMERIC | Primary key and a serial number of predefined locations in IUG legacy system |
| NEIGHBORHOOD_NAME | VARCHAR(60) | Neighborhood name |
| CITY_ID | NUMERIC | City id in which this neighborhood is related for |
| CITY_NAME | VARCHAR(60) | City name |
| GOVERNORATE_ID | NUMERIC | Governorate id that identified in IUG system |
| GOVERNORATE_NAME | VARCHAR(60) | Governorate name |

Students Dimension (ACD_STUDENT_DIM)

In this dimension we store instance for each student in IUG since its establishment, some metadata stored in this dimension and described in detail in Table 3.14.

Table (3.14): Students Dimension

| Column Name | Data Type | Description |
|--------------------|--------------|---|
| STUDENT_NO | NUMERIC | Primary key and a number which represent student's academic identifier. Contains 9 digits |
| STUDENT_AR_NAME | VARCHAR(100) | Student's full name in Arabic language |
| STUDENT_EN_NAME | VARCHAR(100) | Student's full name in English language |
| STUDENT_ID | VARCHAR(30) | Student's national ID |
| RESPONDER_ID | VARCHAR(30) | Student responder national ID |
| GENDER | VARCHAR(1) | Gender of the student: 'M' if Male and 'F' for Female |
| NEIGHBORHOOD_ID | NUMERIC | Student location, foreign key to ACD_NEIGHBORHOOD_DIM |
| ADMISSION_SEMESTER | NUMERIC | The semester when student was admitted |
| ADMISSION_TYPE | VARCHAR(1) | Admission type specify previous study type in which student was accepted at |
| STUDY_PROGRAM | VARCHAR(1) | Academic program of the student, foreign key to States dimension |
| DEPARTMENT_NO | NUMERIC | Student's department |
| ACADEMIC_STATUS | VARCHAR(1) | Student status which specify for example if he is Regular or Graduated student |
| STUDENT_LEVEL | NUMERIC | Academic level for student, a value between 1 and 6 |
| HIGHSCHOOL_SCORE | NUMERIC | Student score in high school |
| HIGHSCHOOL_BRANCH | VARCHAR(2) | Student's high school section |

| | | |
|-----------------|---------|---|
| IS_GRADUATED | NUMERIC | A flag for graduation status of student, 1: graduated, 0: not graduated yet |
| GRADUATION_DATE | DATE | Student's graduation date if exists |

Grants Dimension (SFD_GRANT_DIM)

In this dimension, data warehouse store basic information about IUG grants which are grant number, grant title, and its type as shown in Table 3.15. IUG grants is one of its tools to help the students financially to complete their studies.

Table (3.15): Grants Dimension

| Column Name | Data Type | Description |
|-------------|-------------|--|
| GRANT_NO | NUMERIC | Primary key and a number which represent a Grant unique identifier |
| GRANT_NAME | VARCHAR(60) | Grant name as identified in IUG |
| GRANT_TYPE | NUMERIC | Grant type one of the following: 1: Internal Grant,2: External Grant,3: Loan |

Student Registration Hours Dimension (ACD_REGISTRATION_HOURS_DIM)

Student Registration Hours dimension contains details about registration hours' ranges, which used to classify students' registration based on how many hours they already registered in specific semester. This dimension table described in Table 3.16.

Table (3.16): Student Registration Hours Dimension

| Column Name | Data Type | Description |
|-------------|-------------|--|
| SERIAL_NO | NUMERIC | Primary key |
| LABEL | VARCHAR(60) | Descriptive label for hours range, i.e. "3-6 Registration Hours" |
| MIN_VALUE | NUMERIC | Meta data which represent minimum registration hours value of this range |

| | | |
|-----------|---------|--|
| MAX_VALUE | NUMERIC | Meta data which represent maximum registration hours value of this range |
|-----------|---------|--|

In the following section, we describe data marts in our design of data warehouse. As mentioned before data mart formed as a star schema of fact table and referred by dimension tables. For each data mart, we extract all possible query structures in the form of “logical view”. Thus, the end user can build his reports and metrics based on these logical views. The purpose of these views is to optimize data warehouse performance using our materialized view selection model that proposed in the next chapter.

4.2.2.1 Admission and Registration Data Marts

In this section, we describe each data mart that designed to handle Admission and Registration department business processes in IUG.

High School Results Data Mart

Figure 3.2 shows high school results fact table and its related dimensions as a star schema. The fact table holds two measures in each year for each high school section and divided to 9 grades that defined in Table 3.9, the two measures are students count and their average GPA in high school. As shown in Figure 3.2, fact table has three foreign keys to High School Years, High School Branch, and High School Grade dimension tables which identifies its granularity.

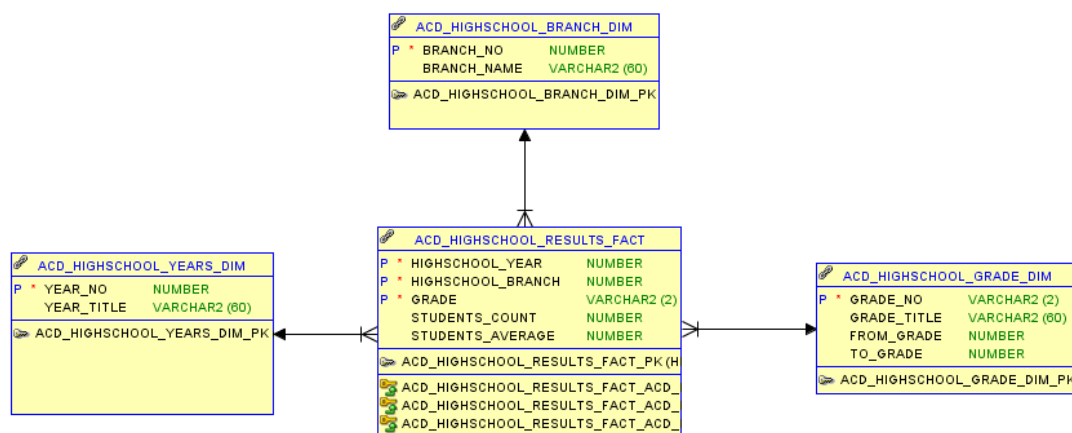


Figure (3.2): High School Results Data Mart

Students Admission Data Mart

Students Admission Data Mart is designed to calculate two measures, which are admitted students count and their average GPA in high school. The granularity of the fact table identified as 1 row per year, per high school section, per high school grade, per department as shown in Figure 3.3. The fact table is referred by Academic Year, High School Section, High School Grade, and Department dimensions.

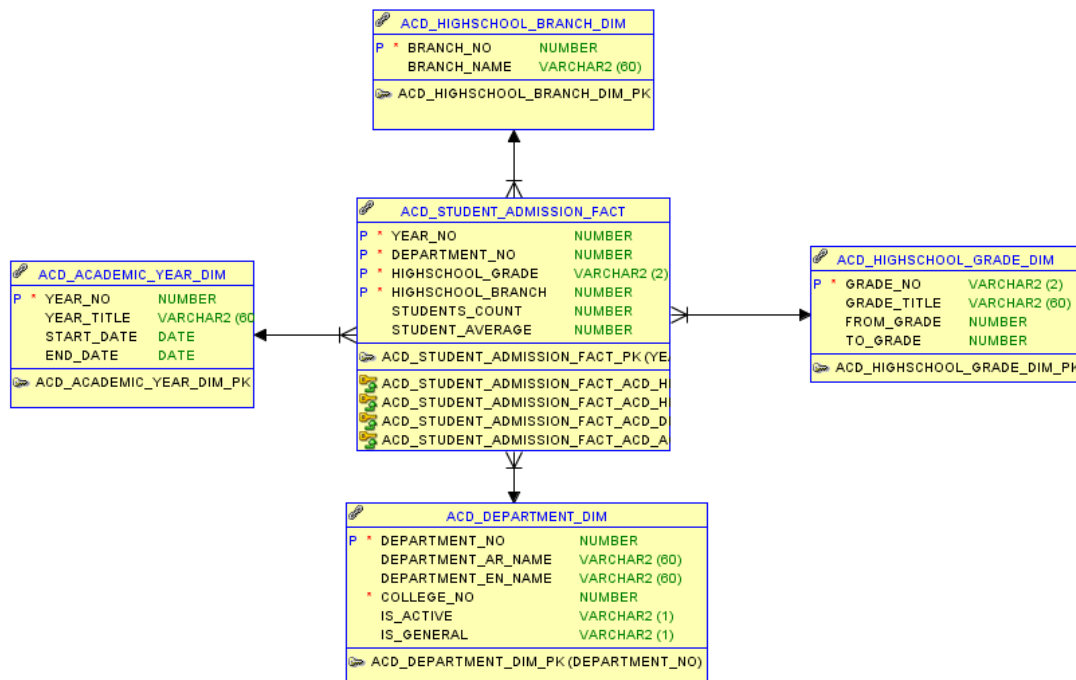


Figure (3.3): Students Admission Data Mart

This data mart used to calculate students count and average GPA in three cases:

- Total admitted students count and their average GPA calculated for each Section and Grade.
- Total admitted students count and their average GPA calculated for each Department.
- Total admitted students count and their average GPA calculated for each College.

We predefined the calculation queries for these three cases in logical views as shown in Figure 3.4. First we build a master view for this data mart titled

“V_ACD_ADMISSION_FACT”, this view joins fact table with all dimension tables and its hierarchies. Next we build Section and Department views, named “V_ACD_ADMISSION_BRANCH” and “V_ACD_ADMISSION_DEPARTMENT” respectively, based on “V_ACD_ADMISSION_FACT”. Finally, College view is build based on “V_ACD_ADMISSION_DEPARTMENT”.



Figure (3.4): Students Admission Views

Students Registration Data Mart

This data mart has one simple measure which is registered students count in a semester. This measure divided over five dimensions listed as: Semester, Students Status, Gender, Academic Program, and Registration Hours Range Dimensions as shown in Figure 3.5 which identify the fact granularity.

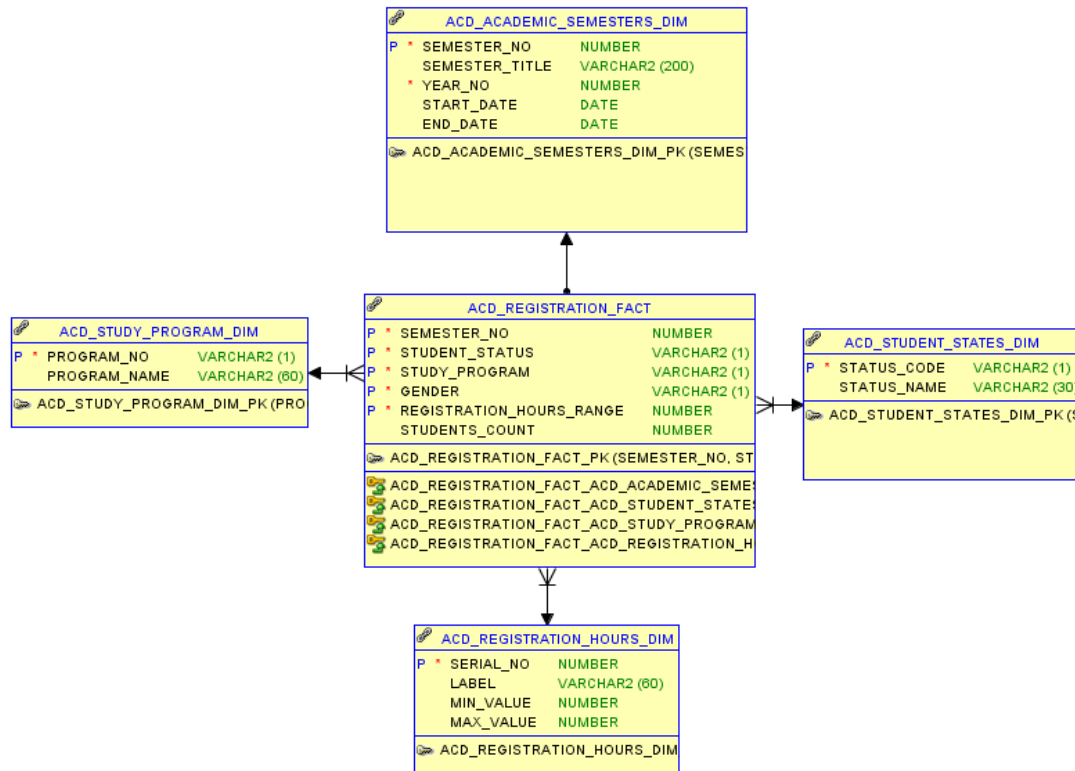


Figure (3.5): Students Registration Data Mart

Based on business requirements, users need to calculate total registered students count in academic semesters in two ways:

- Total registered students count for each Academic Program and for each Gender, finally categorized based on available Registration Hours Range.
- Total registered students count for each Academic Program and Gender.

Thus, according to business requirements we built logical views shown in Figure 3.6. “V_ACD_REGISTRATION_FACT” is the master view that joins the fact table

with all four dimension tables. Next, based on this master view we build “V_ACD_REGISTRATION_HOURS” which address the first calculation required. Then finally the “V_ACD_REGISTRATION_SEMESTERS” view calculates total registered students in academic Semesters for each Academic Program and Gender.

| V_ACD_REGISTRATION_FACT | |
|----------------------------|----------------|
| STATUS_CODE | VARCHAR2 (1) |
| STATUS_NAME | VARCHAR2 (30) |
| PROGRAM_NO | VARCHAR2 (1) |
| PROGRAM_NAME | VARCHAR2 (60) |
| REGISTRATION_HOURS_RANGE | NUMBER |
| REGISTRATION_HOURS_LABEL | VARCHAR2 (60) |
| YEAR_NO | NUMBER |
| SEMESTER_NO | NUMBER |
| SEMESTER_TITLE | VARCHAR2 (200) |
| GENDER | VARCHAR2 (1) |
| STUDENTS_COUNT | NUMBER |
| ACD_STUDENT_STATES_DIM | |
| ACD_STUDY_PROGRAM_DIM | |
| ACD_REGISTRATION_FACT | |
| ACD_REGISTRATION_HOURS_DIM | |
| ACD_ACADEMIC_SEMESTERS_DIM | |

| V_ACD_REGISTRATION_HOURS | |
|--------------------------|----------------|
| PROGRAM_NO | VARCHAR2 (1) |
| PROGRAM_NAME | VARCHAR2 (60) |
| YEAR_NO | NUMBER |
| SEMESTER_NO | NUMBER |
| SEMESTER_TITLE | VARCHAR2 (200) |
| GENDER | VARCHAR2 (1) |
| REGISTRATION_HOURS_RANGE | NUMBER |
| REGISTRATION_HOURS_LABEL | VARCHAR2 (60) |
| STUDENTS_COUNT | UNKNOWN |
| V_ACD_REGISTRATION_FACT | |

| V_ACD_REGISTRATION_SEMESTER | |
|-----------------------------|----------------|
| PROGRAM_NO | VARCHAR2 (1) |
| PROGRAM_NAME | VARCHAR2 (60) |
| YEAR_NO | NUMBER |
| SEMESTER_NO | NUMBER |
| SEMESTER_TITLE | VARCHAR2 (200) |
| GENDER | VARCHAR2 (1) |
| STUDENTS_COUNT | UNKNOWN |
| V_ACD_REGISTRATION_HOURS | |

Figure (3.6): Students Registration Views

Registered Students Geographical Location Data Mart

IUG need to investigate the location of students each semester to be able to facilitate the transportation for students. So, this fact table holds total registered students count in every semester for each college regarding their locations. The fact table shown in Figure 3.7 has three foreign keys to Neighborhood, Semester, and College dimensions.

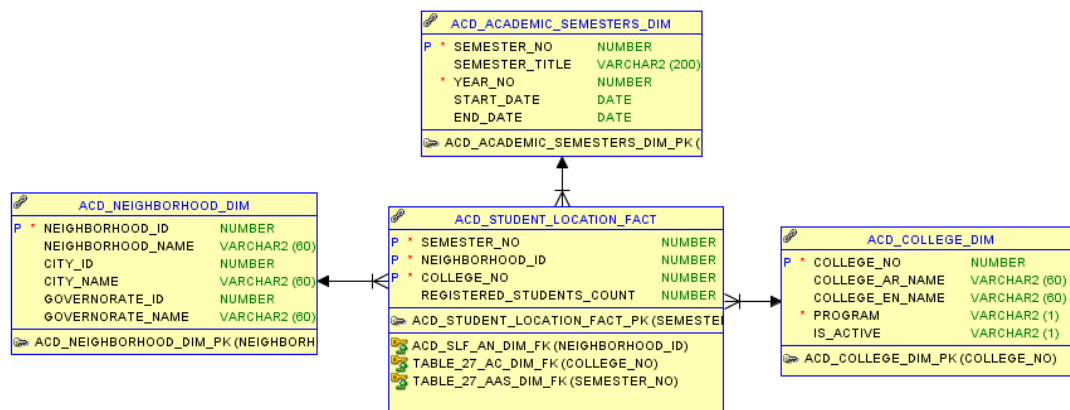


Figure (3.7): Students Locations Data Mart

A master view that joins the fact table with dimension tables is build and considered as a base for “V_ACD_NEIGHBORHOODS” logical view as shown in Figure 3.8. This latter view calculates total registered students count regardless all colleges. This view was the base to calculate total registered students count for each city that presented in “V_ACD_CITY_REGISTRATION” view. Finally, “V_ACD_GOVERNORATE_REGISTRATION” calculates total students count registered categorized by Governorates in IUG. All these three views present a hierarchy.

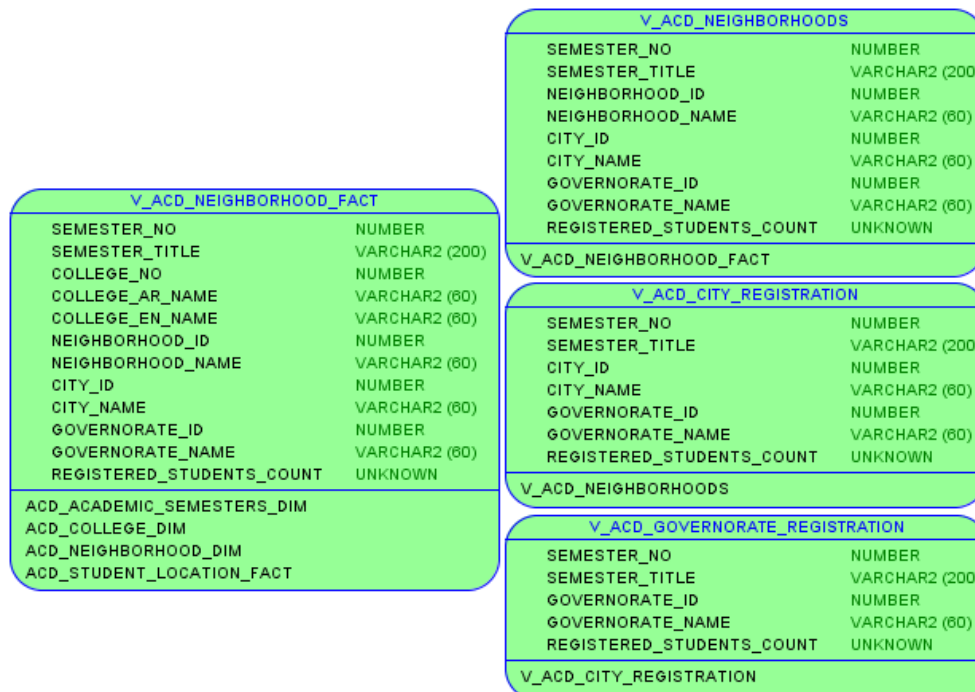


Figure (3.8): Students Locations Views

College Students GPA Data Mart

Each college in IUG use Students' GPA as main indicator for its performance. However, this data mart stores five measures which are students count, their average GPA in the semester, their average success hours of the semester, average success hours for all semesters, and finally average GPA for students as shown in Figure 3.9. The fact table is referred by four dimensions which are Academic Year, Semester, Department, and Academic Level. We use SUCCESS_HOURS and SEMESTER_SUCCESS_HOURS measures to scale SEMESTER_GPA and GPA measures in case of hierarchal calculations. We should note that academic level added to this data mart just for future use, it is not considered as business requirement currently.

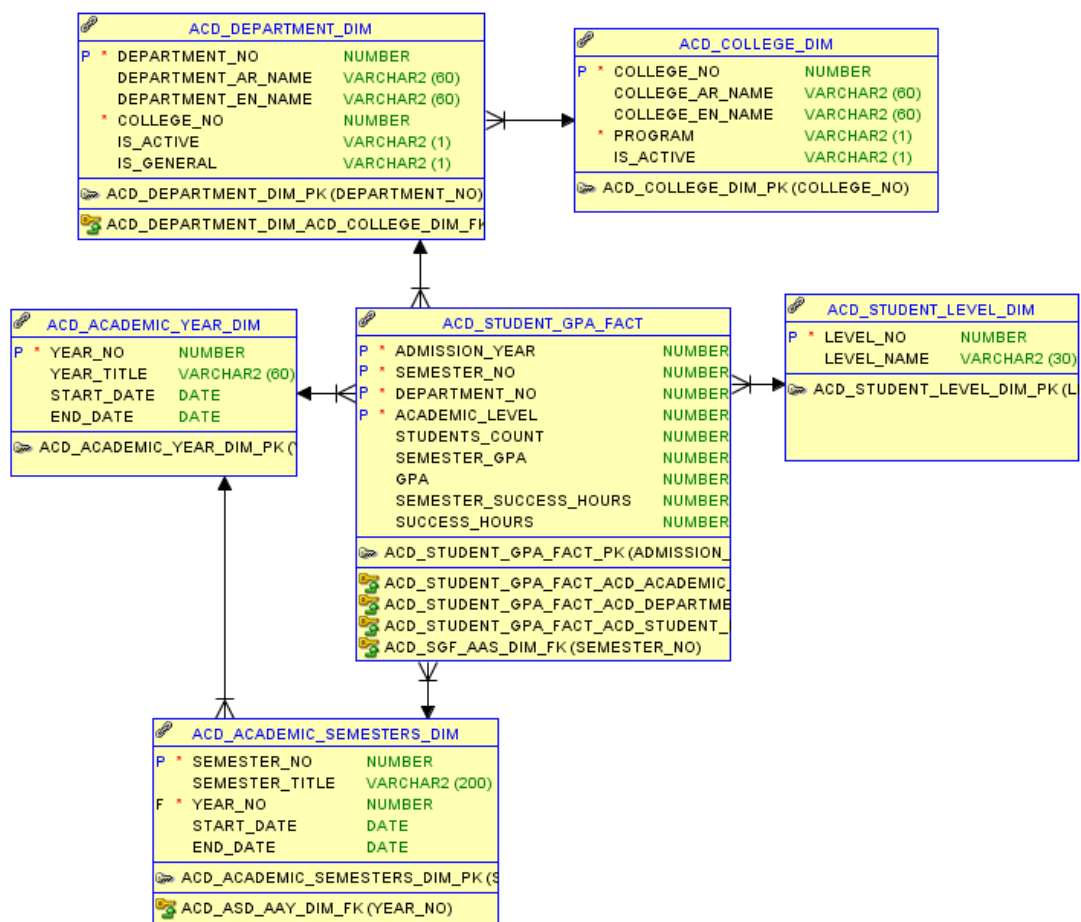


Figure (3.9): College Student GPA Data Mart

From this data mart, we extracted four logical views which are calculated in advance shown in Figure 3.10. These can be summarized as follow:

- “V_ACD_STUDENT_GPA_FACT” shows total students count and their GPA for every Admission Year and Semester separated by Departments.
- “V_ACD_GPA_COLLEGE” calculates total students count and their GPA for each Semester and each College.

| V_ACD_STUDENT_GPA_FACT | |
|------------------------|---------------|
| YEAR_NO | NUMBER |
| YEAR_TITLE | VARCHAR2 (60) |
| LEVEL_NO | NUMBER |
| LEVEL_NAME | VARCHAR2 (30) |
| COLLEGE_NO | NUMBER |
| COLLEGE_AR_NAME | VARCHAR2 (60) |
| COLLEGE_EN_NAME | VARCHAR2 (60) |
| DEPARTMENT_NO | NUMBER |
| DEPARTMENT_AR_NAME | VARCHAR2 (60) |
| DEPARTMENT_EN_NAME | VARCHAR2 (60) |
| STUDENTS_COUNT | NUMBER |
| GPA | NUMBER |
| ACD_ACADEMIC_YEAR_DIM | |
| ACD_STUDENT_LEVEL_DIM | |
| ACD_COLLEGE_DIM | |
| ACD_DEPARTMENT_DIM | |
| ACD_STUDENT_GPA_FACT | |

| V_ACD_GPA_COLLEGE | |
|------------------------|---------------|
| YEAR_NO | NUMBER |
| YEAR_TITLE | VARCHAR2 (60) |
| LEVEL_NO | NUMBER |
| LEVEL_NAME | VARCHAR2 (30) |
| COLLEGE_NO | NUMBER |
| COLLEGE_AR_NAME | VARCHAR2 (60) |
| COLLEGE_EN_NAME | VARCHAR2 (60) |
| STUDENTS_COUNT | UNKNOWN |
| GPA | UNKNOWN |
| V_ACD_STUDENT_GPA_FACT | |

Figure (3.10): College Students GPA Views

Transfers between Colleges Data Mart

This data mart holds a fact table which calculate total transferred students count between colleges and departments. The fact table shown in Figure 3.11 stores transferred students’ count in each semester, for each academic level, for each department, and for which department they are transfer for. This presents its granularity. The figure below illustrates the relationship between the fact table and dimension tables,

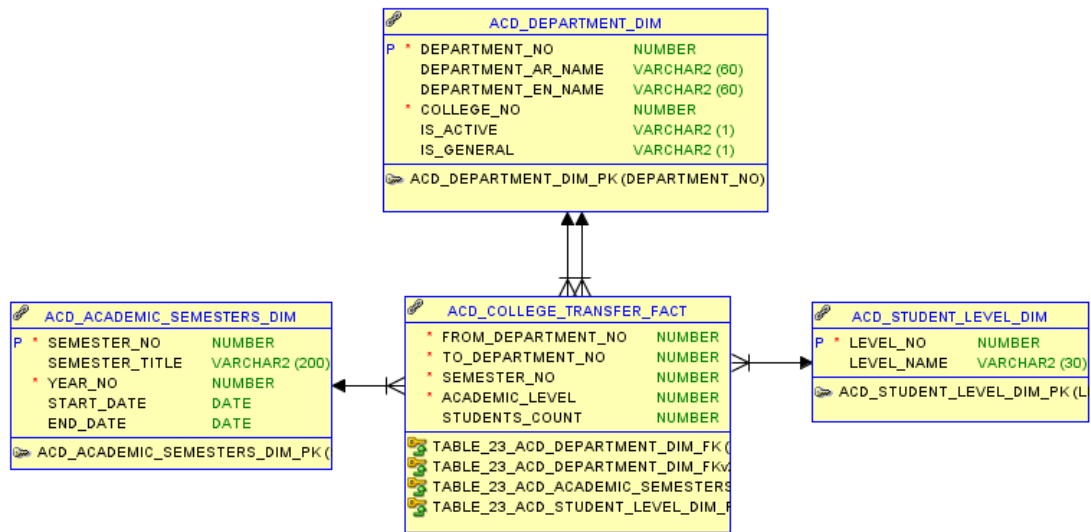


Figure (3.11): College Transfers Data Mart

Colleges and Academic Affairs in IUG need to study why students are transfer from college/department to another during their study. Thus, they may use one of the following logical views which are identified in Figure 3.12:

- “V_ACD_COLLEGE_TRANSFER_FACT” shows total transferred students count in every semester for each level, and from which department they were transferred and to which department they are going.
- “V_ACD_CLG_TSFR_DEPARTMENT” calculates total transferred students count in every year, and from which department they were transferred and to which department they are going.
- “V_ACD_CLG_TSFR_COLLEGE” calculates total transferred students count in every year, and from which college they were transferred and to which college they are going.

| V_ACD_COLLEGE_TRANSFER_FACT | |
|-----------------------------|----------------|
| FROM_DEPARTMENT_NO | NUMBER |
| FROM_DEPARTMENT_AR_NAME | VARCHAR2 (60) |
| FROM_DEPARTMENT_EN_NAME | VARCHAR2 (60) |
| FROM_COLLEGE_NO | NUMBER |
| FROM_COLLEGE_AR_NAME | VARCHAR2 (60) |
| FROM_COLLEGE_EN_NAME | VARCHAR2 (60) |
| TO_DEPARTMENT_NO | NUMBER |
| TO_DEPARTMENT_AR_NAME | VARCHAR2 (60) |
| TO_DEPARTMENT_EN_NAME | VARCHAR2 (60) |
| TO_COLLEGE_NO | NUMBER |
| TO_COLLEGE_AR_NAME | VARCHAR2 (60) |
| TO_COLLEGE_EN_NAME | VARCHAR2 (60) |
| YEAR_NO | NUMBER |
| SEMESTER_NO | NUMBER |
| SEMESTER_TITLE | VARCHAR2 (200) |
| LEVEL_NO | NUMBER |
| LEVEL_NAME | VARCHAR2 (30) |
| STUDENTS_COUNT | NUMBER |
| ACD_COLLEGE_TRANSFER_FACT | |
| ACD_DEPARTMENT_DIM | |
| ACD_COLLEGE_DIM | |
| ACD_ACADEMIC_SEMESTERS_DIM | |
| ACD_STUDENT_LEVEL_DIM | |

| V_ACD_CLG_TSFR_DEPARTMENT | |
|-----------------------------|---------------|
| FROM_DEPARTMENT_NO | NUMBER |
| FROM_DEPARTMENT_AR_NAME | VARCHAR2 (60) |
| FROM_DEPARTMENT_EN_NAME | VARCHAR2 (60) |
| FROM_COLLEGE_NO | NUMBER |
| FROM_COLLEGE_AR_NAME | VARCHAR2 (60) |
| FROM_COLLEGE_EN_NAME | VARCHAR2 (60) |
| TO_DEPARTMENT_NO | NUMBER |
| TO_DEPARTMENT_AR_NAME | VARCHAR2 (60) |
| TO_DEPARTMENT_EN_NAME | VARCHAR2 (60) |
| TO_COLLEGE_NO | NUMBER |
| TO_COLLEGE_AR_NAME | VARCHAR2 (60) |
| TO_COLLEGE_EN_NAME | VARCHAR2 (60) |
| YEAR_NO | NUMBER |
| STUDENTS_COUNT | UNKNOWN |
| V_ACD_COLLEGE_TRANSFER_FACT | |

| V_ACD_CLG_TSFR_COLLEGE | |
|---------------------------|---------------|
| FROM_COLLEGE_NO | NUMBER |
| FROM_COLLEGE_AR_NAME | VARCHAR2 (60) |
| FROM_COLLEGE_EN_NAME | VARCHAR2 (60) |
| TO_COLLEGE_NO | NUMBER |
| TO_COLLEGE_AR_NAME | VARCHAR2 (60) |
| TO_COLLEGE_EN_NAME | VARCHAR2 (60) |
| YEAR_NO | NUMBER |
| STUDENTS_COUNT | UNKNOWN |
| V_ACD_CLG_TSFR_DEPARTMENT | |

Figure (3.12): College Transfers Views

Exam Conflicts Data Mart

Exam Conflicts data mart holds total conflicts count occurred in each course for each semester. The fact table in Figure 3.13 has four foreign keys to Department, Academic Semester, and Subject dimensions. The measure TOTAL_CONFLICTS represents conflicts occurred between SUBJECT_NO and SUBJECT_NO2 in SEMESTER_NO which represent fact table granularity.

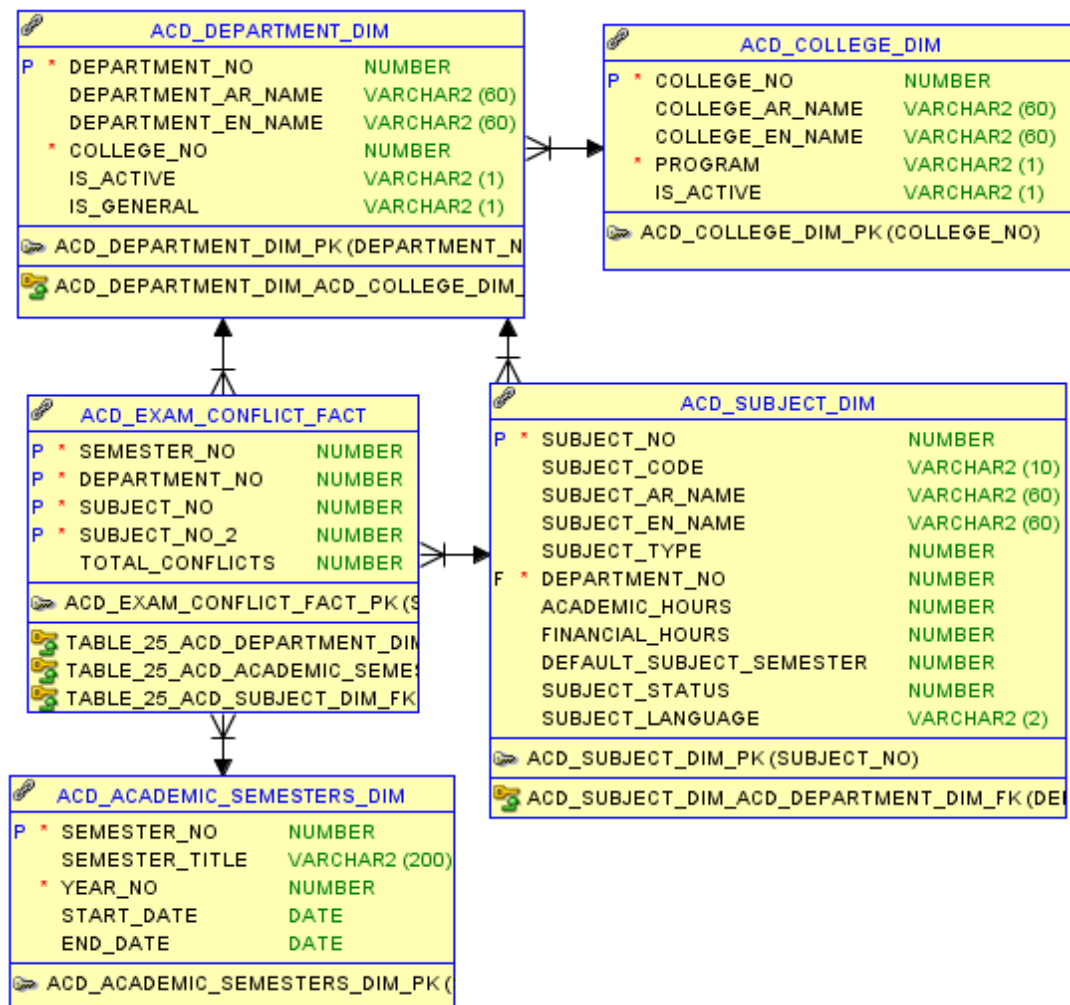


Figure (3.13): Exam Conflicts Data Mart

The master view “V_ACD_EXAM_CONFLICT_FACT” shown in Figure 3.14 joins fact table with three dimension tables Department, Semester, and Subject. For each row in fact table it represented by two records in this master view, the first row is an image for SUBJECT_NO metadata and the second row is an image of SUBJECT_NO_2 metadata. The total conflicts count is same for both records. Additionally, conflicts count could be categorized by Subject Type, Department, or College by using “V_ACD_EXM_CFLT_SUBJECT_TYPE”, “V_ACD_EXM_CFLT_DEPARTMENT”, or “V_ACD_EXM_CFLT_COLLEGE” respectively.



Figure (3.14): Exam Conflicts Views

4.2.2.2 Student Fund Data Marts

Student Fund has four data marts can be listed briefly as:

- Students Registration Data Mart
- Cash Grants Data Mart
- Deferred Grant Data Mart
- Student Fund Totals Data Mart

In this section, we address each data mart and describe it in detail.

Students Registration Data Mart (Student Fund)

Regardless Students Registration Data Mart for Admission and Registration department which described in section 1, this data mart categorize students according to their financial need level. Students in IUG may submit an application for grants to either help them in their study fees, or to let them able to register specific amount of study hours based on a Guarantee. After auditing students' applications, each student is classified to a need level from 'A' to 'F'. Level 'A' means students are in critical need for financial help, and 'F' means students can pay their fees by themselves.

The data mart in this case has a fact table with five measures as shown in Figure 3.15, the measures are total REGISTERED STUDENTS count, total REGULAR STUDENTS count, total REMAINING FEES in students' financial profile, total GUARANTEED STUDENTS count, and total FEES of GUARANTEED STUDENTS. The granularity of fact table identified as 1 row per registration day, per semester, per department, per need level, per gender. That means this fact table updated daily in each semester, and contains measures for each department and academic level and gender. As shown in the figure below it referred by three dimension which are Date, Semester, and Department dimensions.

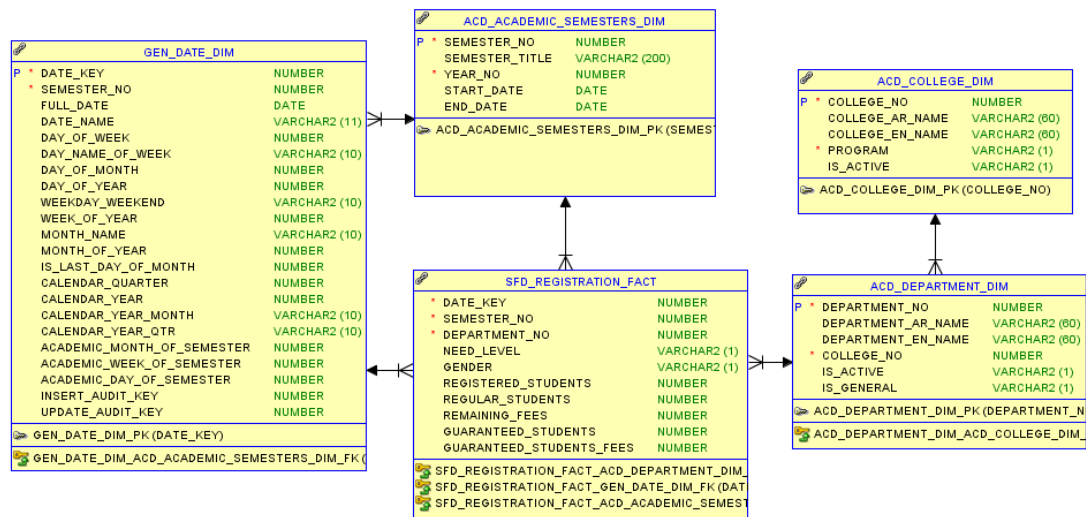


Figure (3.15): Students Registration Data Mart (Student Fund)

Based on the described data mart we built the main view “V_SFD_REGISTRATION__FACT” as shown in Figure 3.16. The view joins the fact table with all dimensions in data mart which can be used to query date range. We defined two other views:

- “V_SFD_REGISTRATION_COLLEGE” that used to calculate measures for semester weeks for each college.
- “V_SFD_REGISTRATION_NEED” which used to calculate measures for semester weeks for each need level between ‘A’ and ‘F’.

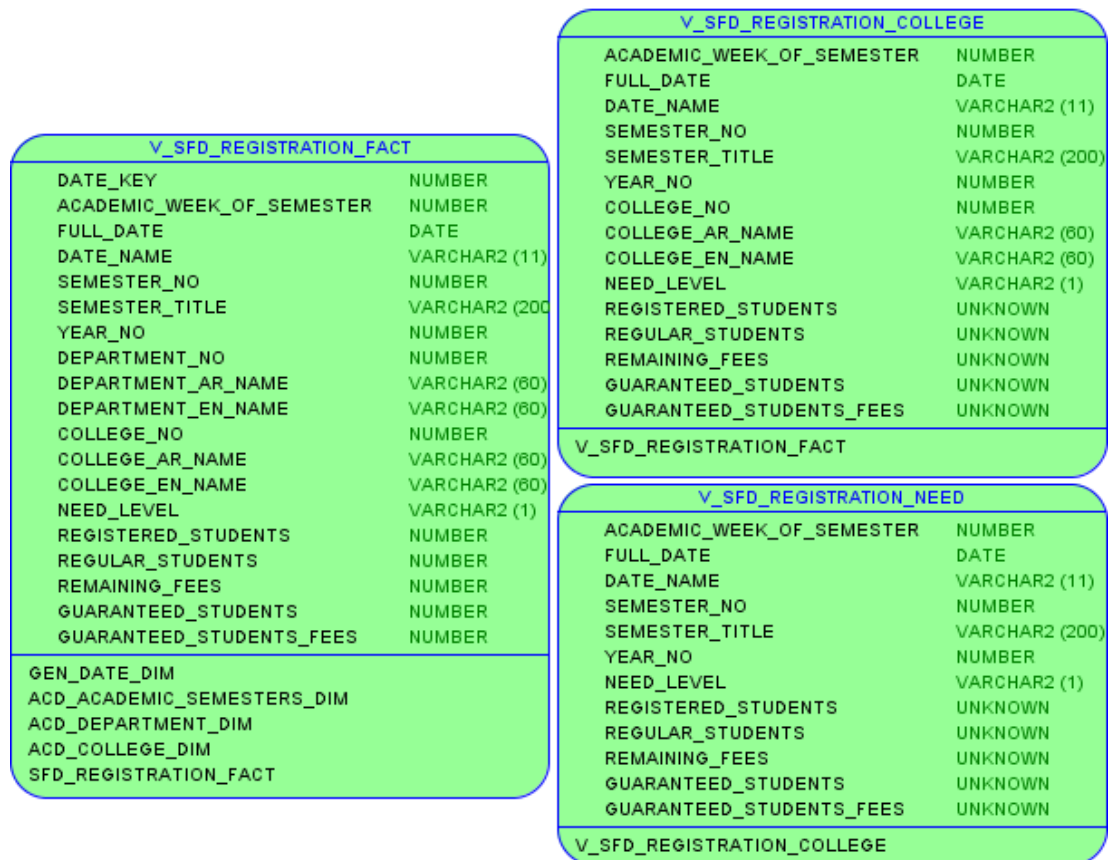


Figure (3.16): Students Registration Views (Student Fund)

Cash Grants Data Mart

Both Student Fund and Finance departments need to trace how many students are get benefit of cash grants in daily basis. Also they need to query total cash grants amount paid for students. For this, we designed Cash Grants Data Mart which it holds

two measures, total students count and total cash amount every day, and categorized by semesters and grants as shown in Figure 3.17. The fact table has three foreign keys for Date, Semester, and Grant dimension tables.

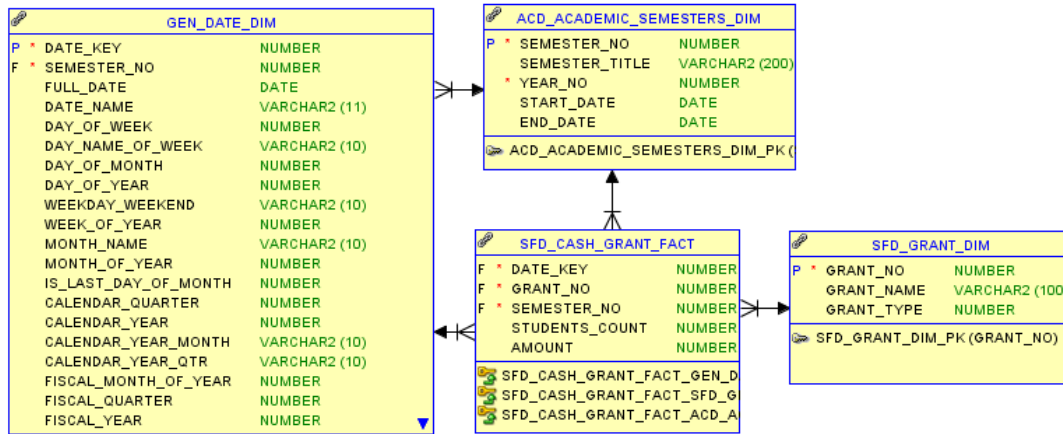


Figure (3.17): Cash Grant Data Mart

We build “V_SFD_CASH_GRANT_FACT” view by joining fact table “SFD_CASH_GRANT_FACT” with all three dimension tables as illustrated in Figure 3.18. This view used to get measures in daily basis. Additionally, we designed “V_SFD_CASH_GRANT_SEMESTER” view to summarize measures over semesters.

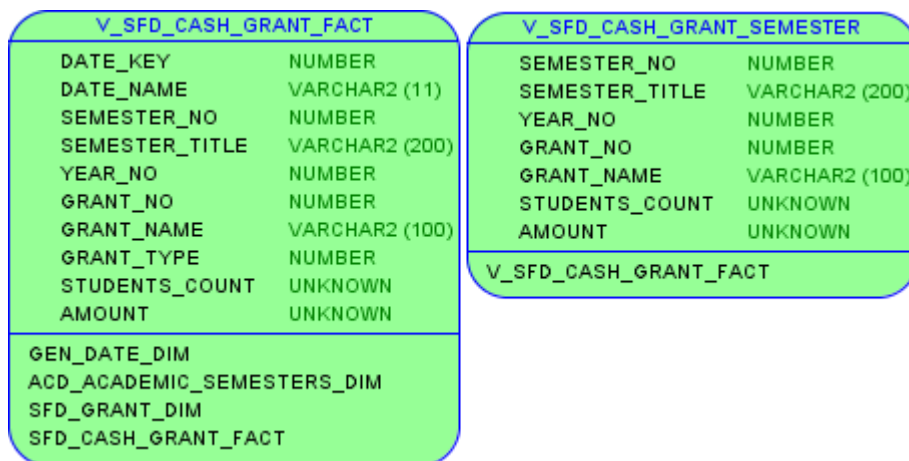


Figure (3.18): Cash Grant Views

Deferred Grant Data Mart

Deferred Grants in IUG are loans granted in the beginning of the semester to students which they must pay it later during the semester. Deferred Grant Data Mart has a fact table of three measures as shown in Figure 3.19, total amount granted in a semester, total amount paid back by students, and total amount must be paid. Rows in the fact table are unique for each semester and grant which defines its granularity.

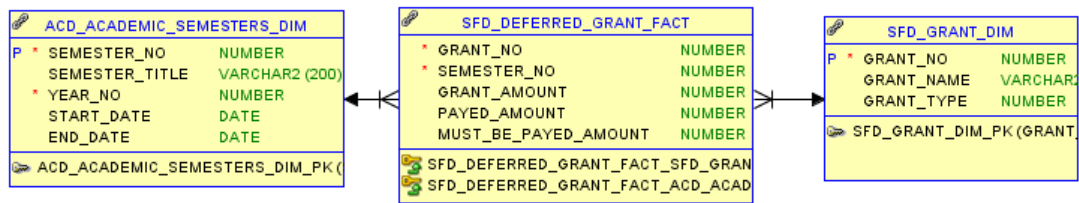


Figure (3.19): Deferred Grant Data Mart

This data mart has one simple view “V_SFD_DEFERRED_GRANT_FACT” which join the fact table with all two dimension tables (Academic Semester and Grant) as shown in Figure 3.20

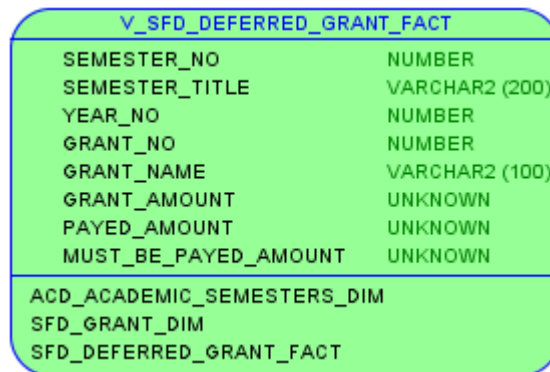


Figure (3.20): Deferred Grants View

Student Fund Summary Data Mart

This data mart holds summary about student fund grants. Its fact table contain total students count and their total grant amount that executed every day categorized by grant no, semester, and academic program. Dimensions in this data mart are Grant, Academic Program, Semester, and Date as shown in Figure 3.21.

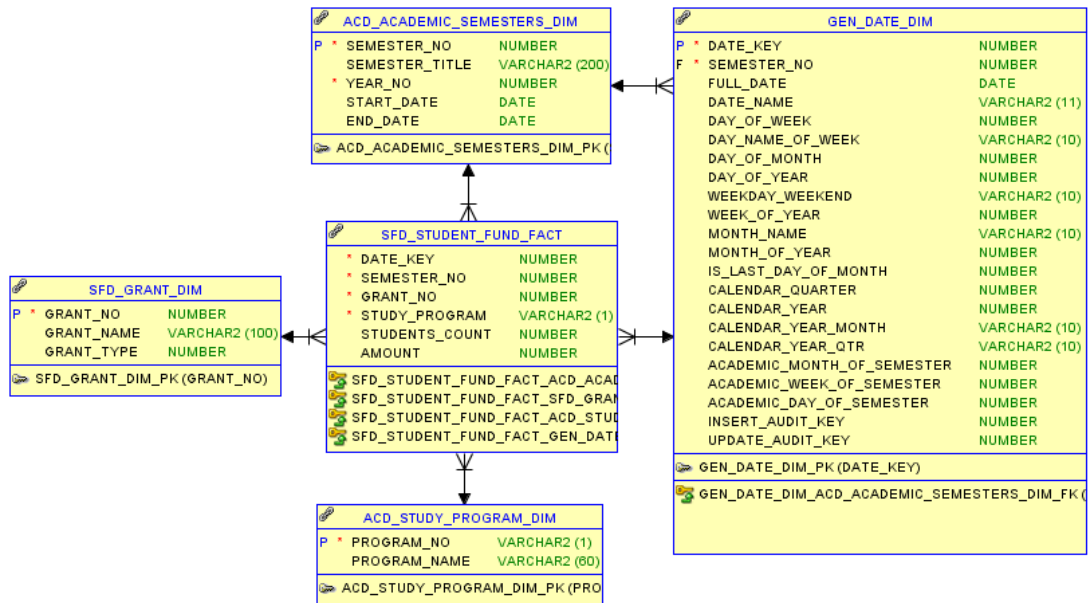


Figure (3.21): Student Fund Totals Data Mart

Based on this data mart, user can query grant execution in daily basis or weekly by using “V_SFD_STUDENT_FUND_FACT” view shown in Figure 3.22. Depending on this view, we designed “V_SFD_STUDENT_FUND_SEMESTER” which calculate granted students count and grants amount for each semester.

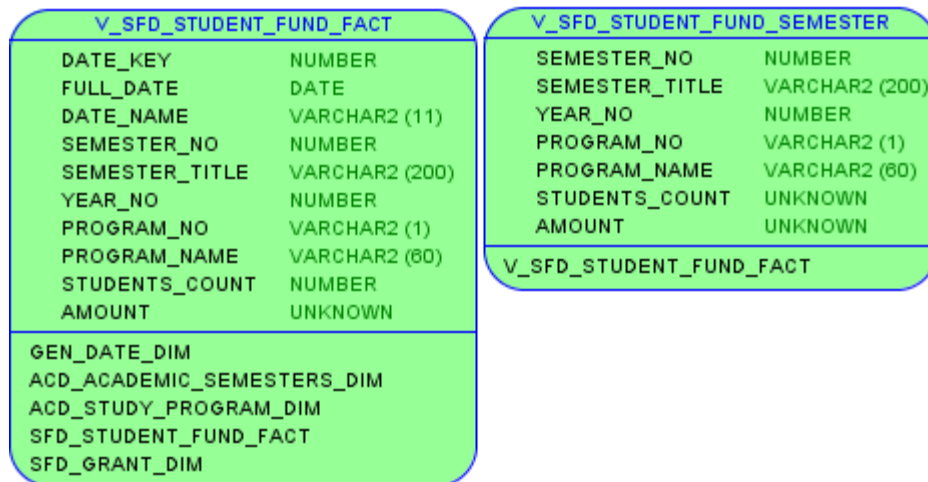


Figure (3.22): Student Fund Totals Views

4.2.2.3 Finance Department Data Marts

Finance department requirements about students are designed in one data mart which is Financial Collection described below.

Financial Collection Data Mart

Financial Collection Data Mart contains details about daily fees collections from students. It has three measures which are total students count paid their fees of specified day, total amount collected from students, and total transactions count. The granularity of this fact table identified as 1 row per day, per semester, per department per teller. Thus, fact table has three foreign keys to Date, Academic Semester, and Department dimension tables as shown in Figure 3.23.

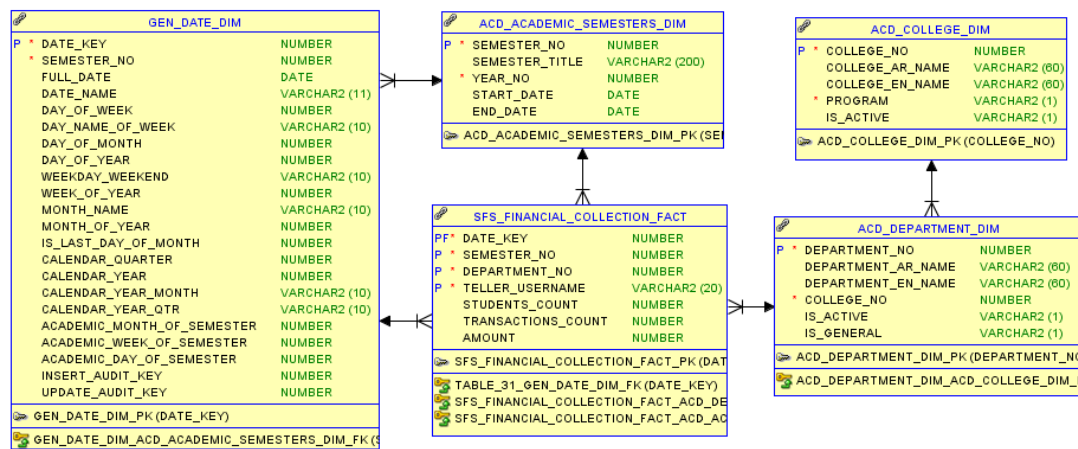


Figure (3.23) Financial Collection Data Mart

Depending on requirement analysis of finance department, we designed logical views shown in Figure 3.24. The base view “V_SFS_FIN_CLCT_FACT” joins the fact table with all dimension tables. Then both “V_SFS_FIN_CLC_DATE_TELLER” and “V_SFS_FIN_SEMESTER_DEPARTMENT” are designed based on the main view. Next, “V_SFS_FIN_CLCT_SEMESTER_TELLER” and “V_SFS_FIN_SEMESTER_COLLEGE” are designed based on “V_SFS_FIN_CLCT_TELLER” and “V_SFS_FIN_SEMESTER_DEPARTMENT” respectively. The purpose of each view is as follow:

- “V_SFS_FIN_CLCT_DATE_TELLER” calculates total fees collected in specific date for each teller regardless department or college.
- “V_SFS_FIN_CLCT_SEMESTER_TELLER” calculates total fees collected during a semester for each teller regardless department or college.

- “V_SFS_FIN_SEMESTER_DEPARTMENT” calculates total fees collected during a semester for each department.
- “V_SFS_FIN_SEMESTER_COLLEGE” calculates total fees collected during a semester for each college.

| | |
|--|--|
| V_SFS_FIN_CLCT_FACT DATE_KEY NUMBER FULL_DATE DATE DATE_NAME VARCHAR2 (11) SEMESTER_NO NUMBER SEMESTER_TITLE VARCHAR2 (200) YEAR_NO NUMBER DEPARTMENT_NO NUMBER DEPARTMENT_AR_NAME VARCHAR2 (60) DEPARTMENT_EN_NAME VARCHAR2 (60) COLLEGE_NO NUMBER COLLEGE_AR_NAME VARCHAR2 (60) COLLEGE_EN_NAME VARCHAR2 (60) TELLER_USERNAME VARCHAR2 (20) STUDENTS_COUNT NUMBER TRANSACTIONS_COUNT NUMBER AMOUNT NUMBER GEN_DATE_DIM ACD_ACADEMIC_SEMESTERS_DIM ACD_DEPARTMENT_DIM ACD_COLLEGE_DIM SFS_FINANCIAL_COLLECTION_FACT | V_SFS_FIN_CLCT_DATE_TELLER DATE_KEY NUMBER FULL_DATE DATE DATE_NAME VARCHAR2 (11) SEMESTER_NO NUMBER SEMESTER_TITLE VARCHAR2 (200) YEAR_NO NUMBER TELLER_USERNAME VARCHAR2 (20) STUDENTS_COUNT UNKNOWN TRANSACTIONS_COUNT UNKNOWN AMOUNT UNKNOWN V_SFS_FIN_CLCT_FACT V_SFS_FIN_SEMESTER_COLLEGE SEMESTER_NO NUMBER SEMESTER_TITLE VARCHAR2 (200) YEAR_NO NUMBER COLLEGE_NO NUMBER COLLEGE_AR_NAME VARCHAR2 (60) COLLEGE_EN_NAME VARCHAR2 (60) STUDENTS_COUNT UNKNOWN TRANSACTIONS_COUNT UNKNOWN AMOUNT UNKNOWN V_SFS_FIN_SEMESTER_DEPARTMENT V_SFS_FIN_SEMESTER_DEPARTMENT SEMESTER_NO NUMBER SEMESTER_TITLE VARCHAR2 (200) YEAR_NO NUMBER DEPARTMENT_NO NUMBER DEPARTMENT_AR_NAME VARCHAR2 (60) DEPARTMENT_EN_NAME VARCHAR2 (60) COLLEGE_NO NUMBER COLLEGE_AR_NAME VARCHAR2 (60) COLLEGE_EN_NAME VARCHAR2 (60) STUDENTS_COUNT UNKNOWN TRANSACTIONS_COUNT UNKNOWN AMOUNT UNKNOWN V_SFS_FIN_CLCT_FACT V_SFS_FIN_CLCT_SEMESTER_TELLER SEMESTER_NO NUMBER SEMESTER_TITLE VARCHAR2 (200) YEAR_NO NUMBER TELLER_USERNAME VARCHAR2 (20) STUDENTS_COUNT UNKNOWN TRANSACTIONS_COUNT UNKNOWN AMOUNT UNKNOWN V_SFS_FIN_CLCT_DATE_TELLER |
|--|--|

Figure (3.24): Financial Collection Views

In this project, Data hierarchy was very important because of the relationship business wise between the data. Hierarchies are meaningful, standard way to group the data within a dimension so you can begin with the big picture and drill down to lower levels to investigate anomalies. According to Kimball, hierarchies are the main paths for summarizing the data. Data hierarchy is an arrangement of data consisting of sets and subsets such that every subset of a set is of lower rank than the set. In the context of the data warehouse, it can be used to provide paths that can be used to roll up and drill down when analyzing the data. The data hierarchy is applicable to the dimension table and it allows for organization of data.

3.3 System Implementation

The system development is the actual implementation of the analysis and design carried out. In this phase of the project, we designed the data warehouse (Fact and dimension tables), the ETL (Extract, Transform and Load) and the front end application for the purpose of this project.

Validation process involved the confirmation by examination and provision of objective that an information system has been implemented correctly and conforms to the need of the user and intended use.

The main focus of this phase is developing procedures to validate the data that has been extracted and moved data in a form that can then be loaded into the warehouse. Finally, the data must be analyzed to determine whether or not certain elements should be cleansed prior to putting it into the warehouse. (Burton & Green, 2016)

The system development stage can now be embarked upon after the actual understanding of the expectation of the business users has been captured.

3.3.1 Design of the Physical Database

The models of IUG data warehouse is designed using Oracle SQL Developer Data Modeler. Thus, we can easily generate the SQL scripts of any database engine as shown in Figure 3.25. In our case, we have generated scripts for Oracle database engine. Next, the actual design of the database is carried out by executing scripts on data warehouse source using the Oracle SQL Developer IDE.

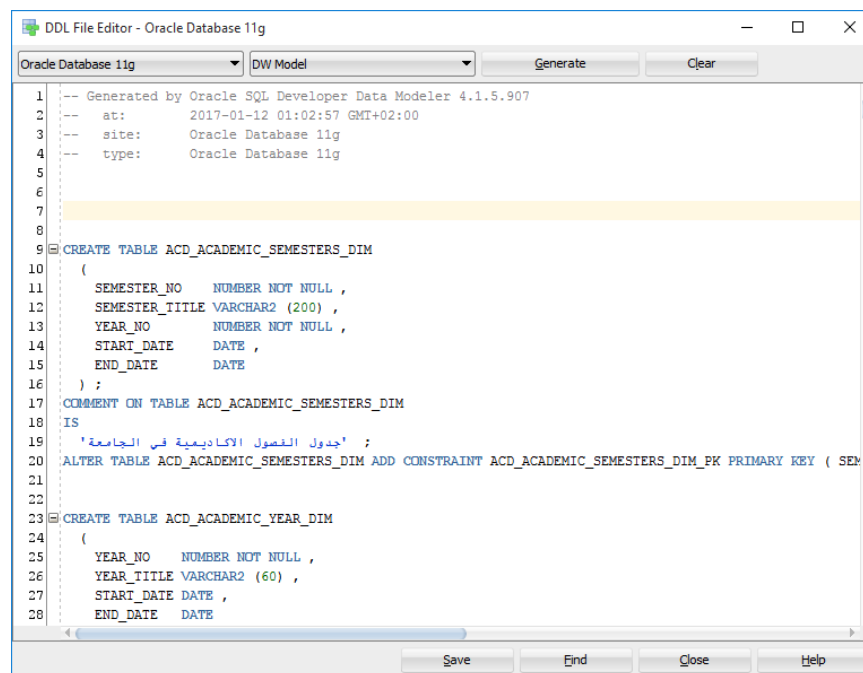


Figure (3.25): DDL Generator in Oracle SQL Developer Data Modeler

As a part of physical design of data warehouse, we created foreign keys between fact tables and every dimension table related to fact table to guarantee the referential integrity between tables. Referential integrity is a concept of establishing a parent-child relationship between two tables, with the purpose of ensuring that every row in the child table has a corresponding parent entry in the parent table.

For this project as mentioned earlier, we adopted Kimball approach of data warehousing. So based on Figure 2.2, the staging phase is responsible of loading data from different sources in IUG (i.e. OLTP database, flat files, or web services) to data warehouse tables. This process is deployed on dedicated server on IUG. The physical design of data warehouse tables was explained in 3.2.4.

3.3.2 Design of the ETL Process

ETL process is implemented after the logical and physical designs of the data warehouse are completed.

As mentioned in the beginning of the project that the IUG has different OLTP databases to extract from. ETL is the process of retrieving and transforming data from the source system and putting it into the data warehouse. With the scope of this project, we used Talend Open Studio to design the ETL process in order to load the data into the data store.

The ETL process is based on the principles provided by Ralph Kimball in (Kimball & Caserta, 2011). After the physical design is implemented on data warehouse database, staging tables are designed. Staging tables are almost having the same structure as source system's tables. In these tables, the data is cleansed and validated along with other quality checking and improvement processes.

In Figure 3.26, we show an example of loading staging tables in our implementation using Talend Open Studio. In this example we extract Academic Years data from source system, after then we map columns between source dataset and destination source which is STG_ACADEMIC_YEARS. In the same way and after the previous process completes (OnSubjobOk in Figure 3.26), we load Academic

Semesters and Dates. In Dates loading process, we join Kimball dates dataset (The Microsoft Data Warehouse Toolkit, 2nd Edition, 2017) with semesters in staging area.

Similarly, we load all staging tables in the data warehouse which will be the base for extracting data to dimension and fact tables.

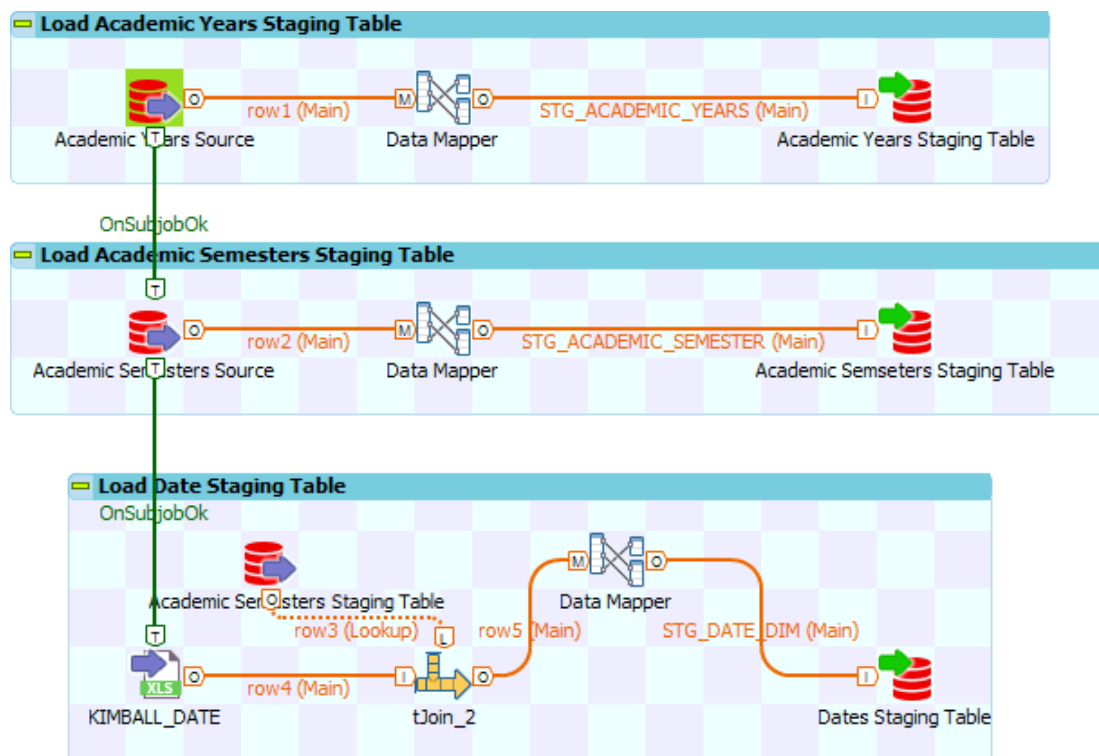


Figure (3.26): Example of Loading Staging Tables

From the staging tables, data is loaded firstly into dimension tables, then facts tables are loaded. Dimension tables are loaded first because of referential integrity that was described in previous section.

The data in dimension tables increases and changes slowly. These changes of data may or may not be tracked based on business requirements. There are different methods to be adopted for each of dimension tables as known as Slowly Changing Dimensions (SCD). These methods are based on Kimball's approach of Slowly Changing Dimensions in a data warehouse and described below.

SCD Type 1: Overwrite existing record

In slowly changing dimension Type 1, no history of dimension changes is kept in the database. The old dimension's attribute value is simply overwritten by the new one. This case is used when the business requirements state that no history of data is required for any analysis.

Figure 3.27 and Figure 3.28 illustrate SCD Type 1 when changing a department name in Departments dimension table.

| DEPARTMENT_NO | DEPARTMENT_AR_NAME | DEPARTMENT_EN_NAME | COLLEGE_NO | IS_ACTIVE | IS_GENERAL |
|---------------|--------------------|----------------------|------------|-----------|------------|
| 805 | هندسة البرمجيات | Software Engineering | 8 | 1 | 1 |

Figure (3.27): SCD Type 1 Before Changing a Record

As shown in figure below, the department name in DEPARTMENT_AR_NAME and DEPARTMENT_EN_NAME columns are totally overwritten by the new value.

| DEPARTMENT_NO | DEPARTMENT_AR_NAME | DEPARTMENT_EN_NAME | COLLEGE_NO | IS_ACTIVE | IS_GENERAL |
|---------------|--------------------|----------------------|------------|-----------|------------|
| 805 | هندسة الحاسوب | Computer Engineering | 8 | 1 | 1 |

Figure (3.28): SCD Type 1 After Changing a Record

SCD Type 2: Create new additional record

In this methodology, a new row is added for every change on the specified columns. Using SCD Type 2 methodology guarantee keeping track of dimension changes history. This type could be implemented by using three additional columns for dimension table, start_date, end_date, and is_active flag to indicate which record is most recent.

This type of SCD, is the most commonly used and preferred for important data because of its ability to store unlimited history as well as to store the time of that change.

In Figure 3.29 we show a student record before moving from a neighborhood to another. And after changing NEIGHBORHOOD_ID from 1 to 3, a new record is created with the new value as shown in Figure 3.30.

| STUDENT_NO | STUDENT_EN_NAME | NEIGHBORHOOD_ID | SCD_START | SCD_END | ACTIVE |
|------------|-----------------|-----------------|-----------|---------|--------|
| 120140001 | Belal Shbair | 1 | 13-APR-14 | (null) | 1 |

Figure (3.29): SCD Type 2 Record Before Update

The SCD_END of the old column is set by the date of the update operation, and ACTIVE flag is set to 0. Then the same date of update operation is used to be set for SCD_START of the new record as shown in figure below.

| STUDENT_NO | STUDENT_EN_NAME | NEIGHBORHOOD_ID | SCD_START | SCD_END | ACTIVE |
|------------|-----------------|-----------------|-----------|-----------|--------|
| 120140001 | Belal Shbair | 1 | 13-APR-14 | 21-OCT-15 | 0 |
| 120140001 | Belal Shbair | 3 | 21-OCT-15 | (null) | 1 |

Figure (3.30): SCD Type 2 After Updating a Record

The ACTIVE flag is used to easily select all active records instead of using effective dates which may put some complexity for the query.

SCD Type 3: Add new column

In this type usually only the current and previous value of dimension is kept in the database. This type of SCD is used when the business requirements state that only limited history needs to be stored in the dimension table. This is the least commonly needed technique, and it is not needed in our design for IUG according to business requirements.

For illustration, Figure 3.31 shows sample record of a customer, the dimension table in this case can store only current type and last type of the customer.

| Before the change: | | | |
|--------------------|---------------|--------------|---------------|
| Customer_ID | Customer_Name | Current_Type | Previous_Type |
| 1 | Cust_1 | Corporate | Corporate |
| | | | |
| After the change: | | | |
| Customer_ID | Customer_Name | Current_Type | Previous_Type |
| 1 | Cust_1 | Retail | Corporate |

Figure (3.31): SCD Type 3 Example

Source: (Morzy, 2012)

SCD Type 6: a combination of SCD types 1, 2, and 3

In this type, a dimension has columns of Type 1 which will be overwritten by the new value, and it has columns of Type 2 that will be traced over time, and finally it has columns of Type 3 that have limited history.

In our design, we only have dimensions of SCD type 1, SCD type 2, and SCD type 6 that is a combination of previous two types. For example, Student dimension in Figure 3.29 has SCD type 2 for NEIGHBORHOOD_ID column and SCD type 1 for STUDENT_EN_NAME column. When student name changes, it is overwritten by the new value for last active record as shown in Figure 3.32.

| STUDENT_NO | STUDENT_EN_NAME | NEIGHBORHOOD_ID | SCD_START | SCD_END | ACTIVE |
|------------|-----------------|-----------------|-----------|-----------|--------|
| 120140001 | Belal Shbair | 1 | 13-APR-14 | 21-OCT-15 | 0 |
| 120140001 | Belal W. Shbair | 3 | 21-OCT-15 | (null) | 1 |

Figure (3.32): SCD Type 6 After Record Update

Loading SCD tables of Type 1 in Talend Open Studio goes directly through data mapper to destination tables as shown in Figure 3.33. The process starts by extracting data from Academic Years Staging Table, then it is filtered in Data Mapper. Finally, the data is prepared to be loaded in ACD_ACADEMIC_YEAR_DIM table. After completing academic years loading process, academic semesters loading process starts. Similarly, data is extracted from staging table, then it handled in Data Mapper, then it is loaded to dimension table.

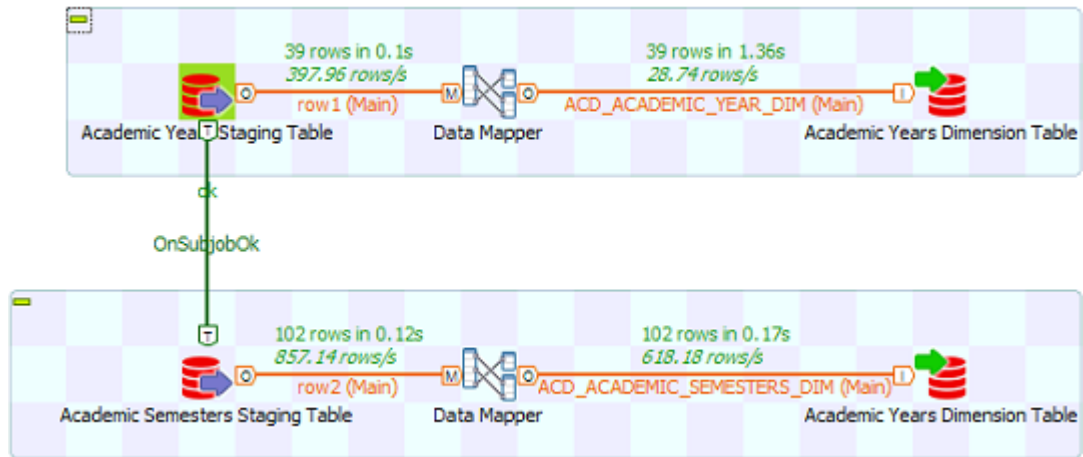


Figure (3.33): SCD Type 1 Implementation in Talend Open Studio

All SCD tables are handled in same way and can be summarized in Figure 3.34. Note that data sources with green icons will start in parallel processes since there is no direct relationship between the target dimensions' tables. Other data sources need to be triggered explicitly by the preceding loading process. For example, loading Subjects dimension table need Department dimension to be up to date since there is foreign key from Subject dimension table to Department dimension table. Similarly, Department dimension table need College dimension table to be loaded first for same reason. On the other hand, Grants dimension table i.e. do not need to wait any other loading process to be finished, so loading Grants dimension table can be started in parallel with other independent loading processes.



Figure (3.34): SCD Type 1 Dimensions Load Process

Fact tables are loaded after loading all dimension tables. To summarize loading fact tables' process, we demonstrate as an example loading colleges' total GPA for students over academic semesters in Figure 3.35. Loading other fact tables is almost going through the same stages.

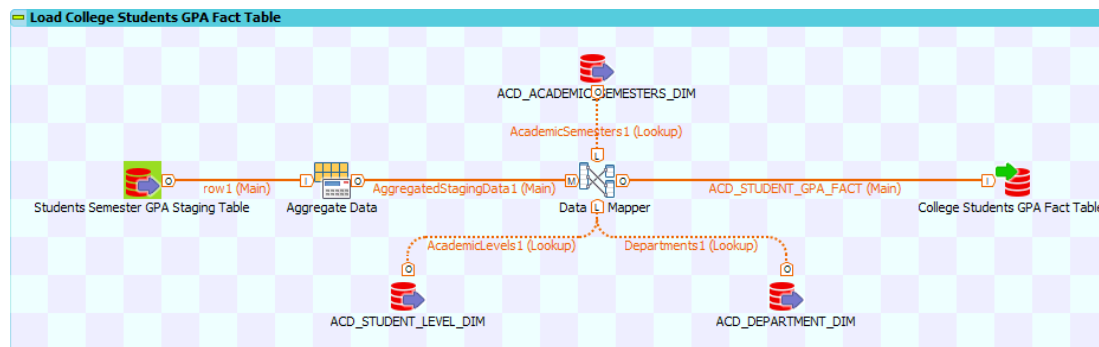


Figure (3.35): Fact Table Loading Process

Loading fact table starts by extracting staging data that is stored in staging area. For this example, data in staging table is stored in the following format shown in Figure 3.36. According to the business requirements, staging table holds for each student, stores his success hours for a semester and total success hours for entire academic life. Also, it holds student's GPA for the semester and average GPA for all semesters.

Schema of Students Semester GPA Staging Table

Students Semester GPA Staging Table

| Column | Db Column | Key | Type | DB Type |
|--------------------------------|--------------------------------|--------------------------|---------|---------|
| STUDENT_NO | STUDENT_NO | <input type="checkbox"/> | Integer | INT |
| SEMESTER_NO | SEMESTER_NO | <input type="checkbox"/> | Integer | INT |
| DEPARTMENT_NO | DEPARTMENT_NO | <input type="checkbox"/> | Integer | INT |
| STUDENT_ACADEMIC_LEVEL | STUDENT_ACADEMIC_LEVEL | <input type="checkbox"/> | Integer | INT |
| STUDENT_SEMESTER_GPA | STUDENT_SEMESTER_GPA | <input type="checkbox"/> | Double | DOUBLE |
| STUDENT_SEMESTER_SUCCESS_HOURS | STUDENT_SEMESTER_SUCCESS_HOURS | <input type="checkbox"/> | Integer | INT |
| STUDENT_TOTAL_GPA | STUDENT_TOTAL_GPA | <input type="checkbox"/> | Double | DOUBLE |
| STUDENT_TOTAL_SUCCESS_HOURS | STUDENT_TOTAL_SUCCESS_HOURS | <input type="checkbox"/> | Integer | INT |

OK Cancel

Figure (3.36): Example of a Staging Table

After extracting the records from staging area, we make data manipulation by aggregating data over departments, semester, and academic levels. The next step is to

map aggregated data with dimension tables to obtain the surrogate keys of these dimension tables. In Data Mapper, we join aggregated data key columns with dimension tables to utilizing filters as shown in Figure 3.37, in this example we show how to join Department dimension with aggregated data by specifying last active records of departments SCD table, this operation known as “lookup” in Talend Open Studio. Similarly, we join aggregated data set with Academic Level and Academic Semester dimensions.

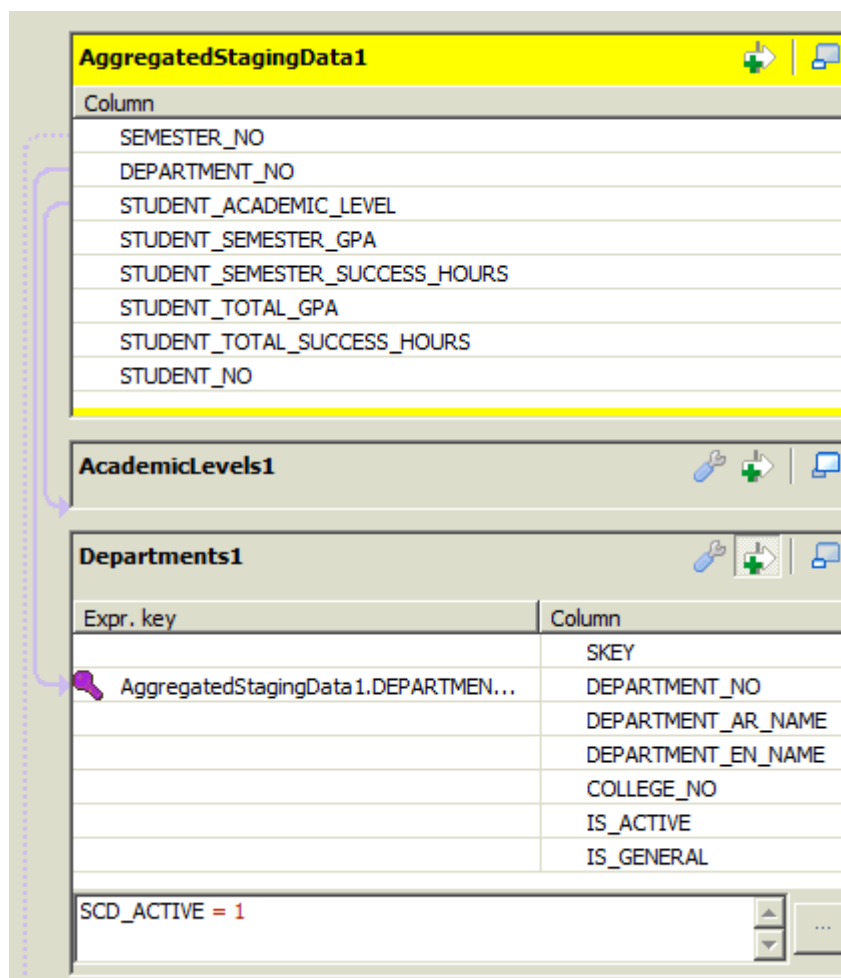


Figure (3.37): Data Mapper Lookups Example

After setting up lookup tables, the mapping process completes by specifying measures from aggregated data set and surrogate keys in dimension tables as demonstrated in Figure 3.38 below.

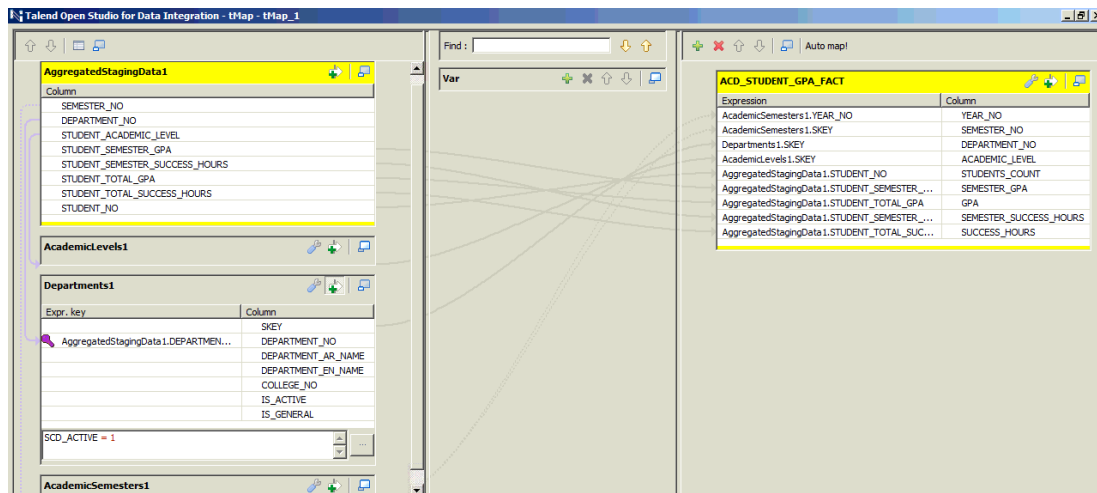


Figure (3.38): Preparing Fact Table Data Using Data Mapper

Finally, the last step of loading a fact table is to set the destination table, which is in our case “ACD_STUDENT_GPA_FACT” table.

ETL process is scheduled to run in daily basis. Every day, at the lowest active time of operational systems, ETL processes run for all jobs sequentially. It searches for new transactions that occurred after last fetching time to be loaded in staging area to be prepared for ETLs.

3.4 System Verification and Maintenance

The best method for verifying the data in the warehouse is to prepare reports on the data in the warehouse and compare it to figures based on the data subsequent to putting into the warehouse which are perceived to be correct. It is seldom believed that users verify the data because they are quite familiar with the detailed type of data they are after. Lastly, Maintenance is essential at each and every stage of the life-cycle. Primarily this entails documentation of processes, applications and most significantly, metadata.

Chapter 4

Proposed Materialized View Selection Model

Chapter 4

Proposed Materialized View Selection Model

In this chapter, we discuss in detail our proposed dynamic Materialized View Selection model.

As discussed in literature review, most of existing approaches for materialized view selection problem are static. There are some dynamic approaches that handle adding new views, removing existing views, and updating view base relations. There is lack in studies that handle dynamic materialized view selection based on view usage frequency.

For dynamic materialized view selection, we propose a model that detects view usage frequency over identified number of historical years, and then we prioritize materialization depending on view's usage frequency. Our model is shown in Algorithm 1.

| | |
|---|---|
| Algorithm 1. The pseudo code of the proposed materialized view selection algorithm based on frequency matrix. | |
| Purpose: | 1. To determine which view can be materialized according to its usage frequency over academic/financial year, and total query processing and maintenance costs. |
| Input: | 2. List of views, V , of size n , Checking Day d , Minimum View Materialization probability δ , Number of history years, h , to be considered |
| Output: | 3. Prioritized list of views, MV , that can be materialized |
| Procedure: | 4. (optional): exclude all views that are manually ignored by business users from L 5. Phase 1: for each view, calculate its usage for given history years h <ol style="list-style-type: none"> let $F = \emptyset$ for $i = 1$ to $n \rightarrow$ loop for $j = h$ to $1 \rightarrow$ loop let $t = \text{last_year}(d) - (h - j)$ $U \leftarrow$ retrieve usage frequencies for view V_i in year t for each value v in $U \rightarrow$ loop set $v = v * (\frac{1}{h})$ (detracting value over years) end loop add U to F_i end loop end loop 6. Phase 2: for each view, check the similarity between features in F_i and features of day d <ol style="list-style-type: none"> $f_d \leftarrow \text{extract_features}(d)$ for $i = 1$ to n loop |

| | |
|--|---|
| | III. $vc \leftarrow \text{find_similarity}(F_i, f_d)$ IV. <i>if</i> $vc \geq \delta$ V. <i>add</i> L_i <i>to</i> MV VI. <i>Reorder output views list</i> MV <i>based on</i> vc VII. <i>end if</i> VIII. <i>end loop</i> 7. (optional): <i>consider all views that are manually flagged to be materialized and add them to</i> MV |
|--|---|

In Algorithm 1, the program accepts the following inputs:

- List of views, V , of size n : this is the list of views to select subset of them to be materialized using our model.
- A certain day, d : the proposed model is triggered in daily basis, so that it is running in every day of the academic/financial year to check current list of views, and select subset of views to be materialized according to historical usage of each view in past days that are similar to day d .
- Minimum view materialization probability δ : the proposed model is configurable to accept the minimum similarity ratio between features of input day, d , and historical usage over days of each view. In case $\delta = 1$, it means materialize any view if it has maximum execution trials in days exactly similar to day d . In case $\delta = 0$, it means materialize all input views. The value of δ must be greater than 0 and it should be adjustable according to materialization failure and success attempts.
- Number of history years, h : usage data are recorded over years, but there must be a limit for historical data to be considered. For example; usage data of ten years ago may not be as important as past year. However, our model limits usage data to considered of the past h years only. This input is set according to organization's policy and vision.

The purpose of Algorithm 1 is to return a list of views to be materialized. This list is ordered according to materialization priority.

The proposed model formed by two phases, the first and last steps are optional which are manually handled by the user. These two steps are designed to force remove/add views from/to materialization as defined by user.

In phase 1, for each input view, we calculate its frequency matrix over identified historical years. The values in the frequency matrix is calculated according to the formula in line VII of Algorithm 1:

$$v_{new} = v_{old} * \left(\frac{j}{h}\right) \quad (4.1)$$

Where

v_{old} : the original value of usage frequency of view

h : number of historical years to be considered

j : value between h and 1, which represent checking year's distance from current year

For example, assume the original frequencies of using VIEW_1, VIEW_2, VIEW_3, and VIEW_4 for last 5 years (2016, 2015, 2014, 2013, and 2012) were as shown in Table 4.1.

Table (4.1): Frequency Matrix of Last Five Years

| View\Year | 2016 | 2015 | 2014 | 2013 | 2012 | Total |
|-----------|------|------|------|------|------|-------|
| VIEW_1 | 9 | 7 | 15 | 13 | 18 | 62 |
| VIEW_2 | 1 | 3 | 9 | 12 | 17 | 42 |
| VIEW_3 | 12 | 10 | 1 | 1 | 0 | 24 |
| VIEW_4 | 0 | 0 | 5 | 11 | 13 | 29 |

Applying formula (4.1) we get detracted frequency matrix over years as shown in Table 4.2. As we go far away from current year, the usage frequency is being less important in comparison of closer years.

Table (4.2): Detracted Frequency Matrix of Last Five Years

| View\Year | 2016 | 2015 | 2014 | 2013 | 2012 | Total |
|-----------|------|------|------|------|------|-------|
| VIEW_1 | 9 | 5.6 | 9 | 5.2 | 3.6 | 32.4 |
| VIEW_2 | 1 | 2.4 | 5.4 | 4.8 | 3.4 | 17 |
| VIEW_3 | 12 | 8 | 0.6 | 0.4 | 0 | 21 |
| VIEW_4 | 0 | 0 | 3 | 4.4 | 2.6 | 10 |

The frequency of using all views in last year is not affected, but the usage frequencies in 2015 was detracted by (4/5). The usage frequencies in 2014 was detracted by (3/5) as well, and those for 2013 and 2012 were detracted by (2/5) and (1/5) respectively.

The purpose of this step is to give higher importance for recent usage of views than older usage records. In our example in Table 4.2, we can obviously note that VIEW_3 got higher priority to be materialized than VIEW_2 and VIEW_4, since it is getting higher usage frequencies in recent years compared to 2012, 2013 and 2014, that while VIEW_2 and VIEW_4 is not highly used in recent years as shown in Table 4.1. VIEW_1 still has highest priority for materialization that because its big usage frequencies in years 2012, 2013, and 2014 compared to VIEW_3 usage frequencies in same periods.

Finally, in phase 2 we find the similarity between features of day d and historical usage days' features. In this thesis, we focus on academic day features which are:

- Academic semester: any day in the academic year of the university calendar is related to a specific semester. This feature will be one of the following values: first semester, second semester, third semester.
- Month of the academic semester: first and second semesters are typically last by four or five months, while third semester (summer term) usually lasts after two months. Thus, MONTH_OF_SEMESTER should be value between 1 and 5.

- Week of the academic semester: this feature is recorded since some views are usually used in specific weeks during academic semester, so that we would better to materialize these view on the specific week rather than materializing it during the whole academic month of semester.

These features are selected as the directors of Admission & Registration, Finance, and Student Fund departments recommended, their recommendation was based on daily report usage. All these features are extracted as in line I of phase 2 in Algorithm 1, then it was passed to the similarity check algorithm. Some enterprises may select other features for similarity check, such as: months of the year, weeks of the year, weeks of the month, fiscal year quarter, or month of quarter. That can be easily implemented using our model.

Our similarity check algorithm is shown in Algorithm 2.

| Algorithm 2. The pseudo code of similarity check function. | |
|--|--|
| Purpose: | 1. To find the similarity ratio between given day features and detracted frequency matrix |
| Input: | 2. Features of view i usage over years, F_i , Day features f_d |
| Output: | 3. Similarity ratio of f_d to F_i , value between 0 and 1 |
| Procedure: | I. let $S = 1$ II. set $L = [SEMESTER, [MONTH_OF_SEMESTER, WEEK_OF_SEMESTER]]$ III. for each level m in L IV. set $S_{temp} = 0$ V. for each feature l in ordered list L of level m VI. $u \leftarrow$ calculate total usage frequency of view i where $l = f_d(l)$ VII. $u_{max} \leftarrow$ select maximum usage frequency of view i for all l possible values VIII. $S_{temp} = S_{temp} + \frac{u}{u_{max}}$ IX. end foreach X. $S_{temp} = \frac{S_{temp}}{\text{size of } L \text{ in level } m}$ XI. If $S_{temp} = 0$ XII. $S = 0$ XIII. go to end XIV. end if XV. $S = S * S_{temp}$ XVI. end foreach end \rightarrow return S |

Algorithm 2 accepts two inputs which are:

- Checking Day Features, f_d : formed a list of features of the input day which are extracted in step I of phase 2. For example, the day of October 01, 2016 has the following features: {SEMESTER: 1, MONTH_OF_SEMESTER: 2, WEEK_OF_SEMESTER: 5}.
- List of detracted usage frequency of view i , F_i : frequency matrix as discussed before is designed for last h years, the data in this matrix is hierarchal as shown in Figure 4.1, which means SEMESTER is related to ACADEMIC YEAR, and MONTH OF SEMESTER and WEEK OF SEMESTER is related to SEMESTER. Frequency matrix is simply formed by aggregation queries over historical usage records.

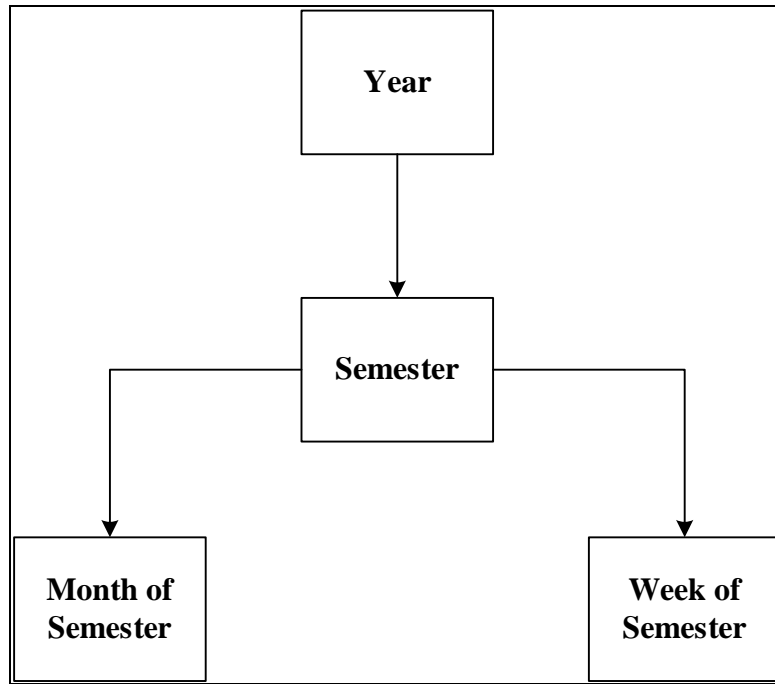


Figure (4.1): Usage Frequency Hierarchy

In the first step of similarity check algorithm, we consider similarity ratio between entry day and historical usage of view i is 100% similar. Next we start iterating over defined features list [Semester, Month of Semester, Week of Semester] and calculate usage frequency for each feature. For any set of features in same level, if there were not any historical usage records similar to input day's feature, then we set similarity ratio to zero. On the other hand, if there were some records, we calculate the maximum usage frequency for all values of checking level by reserving parent

level value; for example, if we are checking the similarity for Month of Semester, we calculate the maximum frequency usage over all five possible values [1, 2, 3, 4, and 5] for selected Semester (parent level) as shown in step VIII of Algorithm 2. For features in same level (MONTH_OF_SEMESTER and WEEK_OF_SEMESTER in our case), we calculate the average similarity ratio of them (step X in Algorithm 2). Then, the last step, we reduce similarity check by factor $\frac{u}{u_{max}}$, where u is total usage frequency of view i for selected feature, and u_{max} is the maximum usage frequency for view i over all possible values of the selected feature.

For illustration, assume frequency matrices for VIEW_5 is as shown in Table 4.3 and Table 4.4, and assume the feature of checking day was {SEMESTER: 1, MONTH_OF_SEMESTER: 2, WEEK_OF_SEMESTER: 6}. We are going to calculate the similarity for each feature in the features list. We will start by checking the similarity of SEMESTER feature which is the parent in the hierarchy. Then we will check the similarity for both MONTH_OF_SEMESTER and WEEK_OF_SEMESTER features. Since these last two features are in same level (see Figure 4.1), the similarity of this level is the average ratio of them. The result of this level is multiplied with parent level (SEMESTER) to get final similarity ratio.

By running our similarity check algorithm, we start by checking SEMESTER's frequencies, which results:

$$u = \text{total usage in } 1^{st} \text{ semester} = 15 + 17 = 32$$

Then we calculate the maximum usage frequency over all semesters which is:

$$u_{max} = \max(32, 19, 0) = 32$$

We got S here not reduced for checking similarity between semesters since $S = S * \frac{u}{u_{max}} = 1 * \frac{32}{32} = 1$.

Next, the algorithm goes deeper to next level by checking similarity between month of semester, for this feature we get:

$$u = \text{total usage in } 2^{\text{nd}} \text{ month of } 1^{\text{st}} \text{ semester} = 15$$

Then we calculate the maximum usage frequency over all months of current semester (1st Semester) which is:

$$u_{\max} = \max(17, 15) = 17$$

$$\text{However, } S_{\text{temp}} \text{ become } S_{\text{temp}} = S_{\text{temp}} + \frac{u}{u_{\max}} = 0 + \frac{15}{17} = 0.88.$$

Table (4.3): Frequency Matrix of VIEW_5 Over Months

| | First Semester | Second Semester | Third Semester |
|-----------------------|----------------|-----------------|----------------|
| 1 st month | 17 | 0 | 0 |
| 2 nd month | 15 | 19 | 0 |
| 3 rd month | 0 | 0 | 0 |
| 4 th month | 0 | 0 | 0 |
| 5 th month | 0 | 0 | 0 |

Next, we check the similarity of week of the month based on Table 4.4:

$$u = \text{total usage in } 6^{\text{th}} \text{ week of } 1^{\text{st}} \text{ semester} = 12$$

Then we calculate the maximum usage frequency over all week of current semester (1st Semester) which is:

$$u_{\max} = \max(7, 7, 3, 12, 3) = 12$$

That makes S_{temp} increases since view i was being used most in 6th week of first semester, $S_{\text{temp}} = S_{\text{temp}} + \frac{u}{u_{\max}} = 0.88 + \frac{12}{12} = 1.88.$

Table (4.4): Frequency Matrix of VIEW_5 Over Weeks

| | First Semester | Second Semester | Third Semester |
|----------------------|----------------|-----------------|----------------|
| 1 st week | 7 | 0 | 0 |
| 2 nd week | 0 | 0 | 0 |
| 3 rd week | 7 | 0 | 0 |

| | | | |
|-----------------------|----|----|---|
| 4 th week | 3 | 0 | 0 |
| 5 th week | 0 | 10 | 0 |
| 6 th week | 12 | 9 | 0 |
| 7 th week | 3 | 0 | 0 |
| 8 th week | 0 | 0 | 0 |
| 9 th week | 0 | 0 | 0 |
| 10 th week | 0 | 0 | 0 |

Finally, for current level we calculate the average by $S_{temp} = \frac{S_{temp}}{\text{size of } L \text{ in level } m} = \frac{1.88}{2} = 0.94$. That makes $S = S * S_{temp} = 1 * 0.94 = 0.94$

In our example, VIEW_5 has 94% chance to be called on the checking day. The final decision is based on the user's input δ (minimum similarity ratio) as shown on line IV in Algorithm 1.

After calculating similarity ratio, and if it passed the checking conditions on line IV, then the view is considered to be materialized.

Chapter 5

Results and Discussion

Chapter 5

Results and Discussion

In this chapter, we test our model over TPC Benchmark H (TPC-H) showing significant reduction in total MVPP cost in Section 5.1. Additionally, in Section 5.2 we show the effect of using the data warehouse in IUG environment and how materializing its views can provide even better performance.

5.1 TPC Benchmark™H (TPC-H)

The component of TPC-H schema is defined to consist of eight tables (base relation) including REGION, NATION, CUSTOMER, SUPPLIER, PART, PARTSUPP, LINEITEM AND ORDERS. The relationships between these tables in TPC-H schema are illustrated in Figure 5.1.

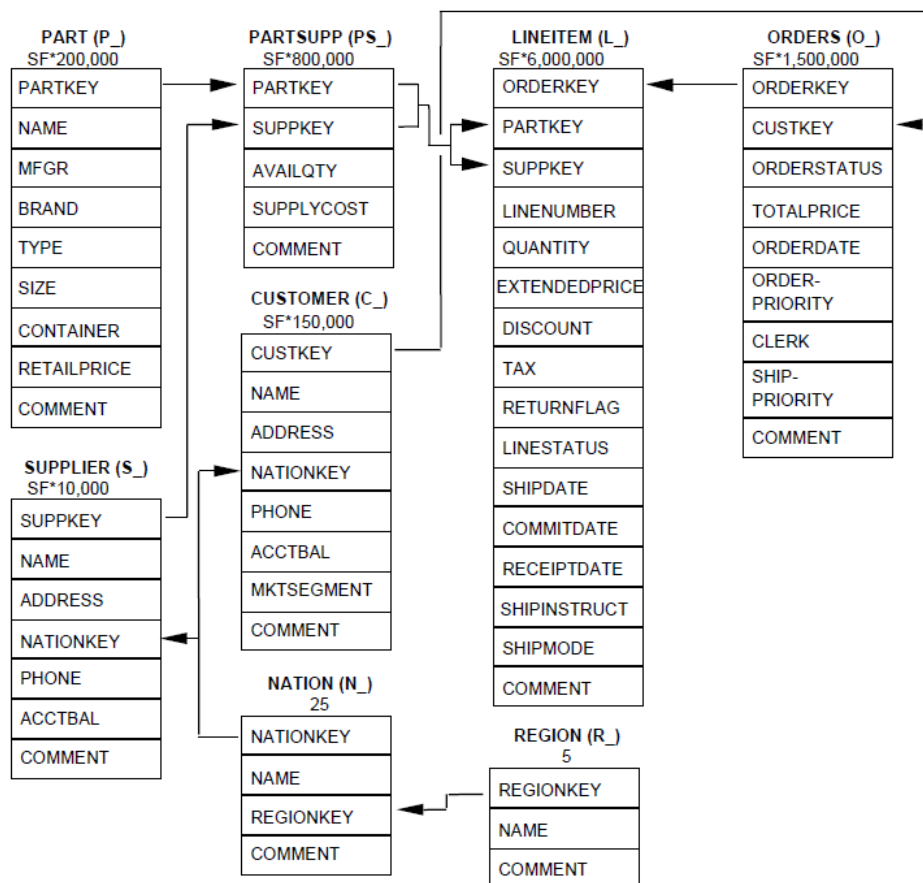


Figure (5.1): The TPC-H Schema

Source: (TPC Benchmark H, 2017)

The number below each table name represents the cardinality of the table. Some tables are factored by Scale Factor (SF) to obtain the chosen database size. Scale factors that used for the test database must be chosen from the set of fixed scale factors defined as follows (TPC Benchmark H, 2017, p. 79):

1, 10, 30, 100, 1,000, 3,000, 10,000, 30,000, 100,000

We run TPC-H schema by Oracle11gR2 with database size defined with reference to Scale Factor 1 (i.e., SF = 1; approximately 1 GB), the minimum required for a test database. The size of the NATION and REGION tables are fixed to 25 and 5 rows respectively. The TPC-H Schema Table Sizes of each base relation are presented in Table 5.1.

Table (5.1): The TPC-H Schema Table Size

| Table Name | Relation Size (in Tuples) | Record Size (in bytes) | Table Size (in MB) |
|------------|------------------------------|---------------------------|-----------------------|
| SUPPLIER | 10,000 | 159 | 2 |
| PART | 200,000 | 155 | 30 |
| PARTSUPP | 800,000 | 144 | 110 |
| CUSTOMER | 150,000 | 179 | 26 |
| ORDERS | 1,500,000 | 104 | 149 |
| LINEITEM | 6,000,000 | 112 | 641 |
| NATION | 25 | 128 | <1 |
| REGION | 5 | 124 | <1 |

Source: (Phuboon-ob, 2009)

Table 5.1 presents each table and its relation size (tuples or rows), and the size of each row (in bytes). The size of tables is number of rows multiplied by size of each row.

Query Set for Materialized View Selection over TPC-H

Query1 to Query7 were introduced by (Phuboon-ob, 2009) for static materialized view selection problem using Two-Phase Optimization (2PO) algorithm

based on MVPP structure. However, in (Suchyukorn, 2013) the researcher improved the query processing cost of the cheapest MVPP of Query1 to Query7 by her MVPP re-optimization algorithm, and she used 2PO to select set of views to be materialized. In this research and for TPC-H benchmark experiments, we based our effort on the re-optimized MVPP generated by (Suchyukorn, 2013). The details of Query1 is discussed in this section for illustration, while the other queries Query2 to Query7 are listed in Appendix 2. The Query1 to Query7 are denoted as Q1, Q2, Q3, Q4, Q5, Q6 and Q7.

The notations used in relational algebra query tree are as follow:

- σ_a represents the select operation, where a is a selection condition on one or more attributes of a relation.
- π_b represents the project operation, where b is a list of one or more attributes of a relation.
- \bowtie represents the inner join operation.
- γ represents an aggregation function.

The details of Query1 and its relational algebra query tree is described as follows:

Query Q1 produces the minimum supply cost of each nation of suppliers in specific region, ASIA. Its relational algebra tree is shown in Figure 5.2.

Query Q1:

```
SELECT n_name, Min(ps_supplycost)
FROM   part, partsupp, supplier, nation, region
WHERE  p_partkey = ps_partkey
      AND s_suppkey = ps_suppkey
      AND s_nationkey = n_nationkey
      AND n_regionkey = r_regionkey
      AND r_name = 'asia'
GROUP BY n_name;
```

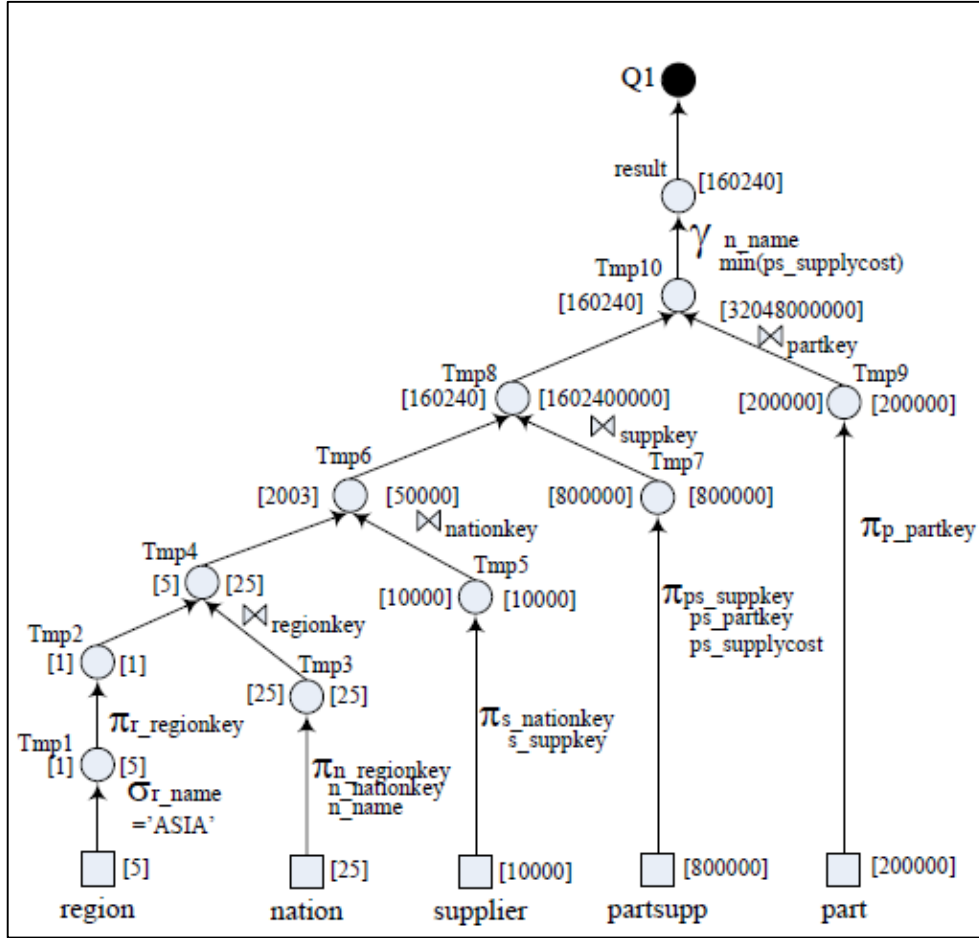


Figure (5.2): Relational Algebra Query Tree of Query Q1

Relational Algebra Query Tree is usually used to express the execution plans for database management systems (DBMS), it is the base item to build the MVPP structure.

The cost of an execution plan does not have a unit. “The value of this column does not have any particular unit of measurement; it is merely a weighted value used to compare costs of execution plans” (Oracle Corporation, 2014, p. 334).

The rest of queries Q2 through Q7 are listed in Appendix 2: TPC-H Schema Queries and Relational Algebra Trees for reference.

Depending on relational algebra trees of queries Q1 to Q7 and the cost (relation size) of each query, the researchers generated their re-optimized MVPP. Then they executed 2PO to select list of views to be materialized that resulting minimum query

processing and maintenance costs. Figure 5.3 shows the resulted re-optimized MVPP and selected nodes to be materialized by 2PO algorithm. The cost for each operation node is labeled at the right side of the node, the number of rows produced by this operation is labeled at the left side of the node.

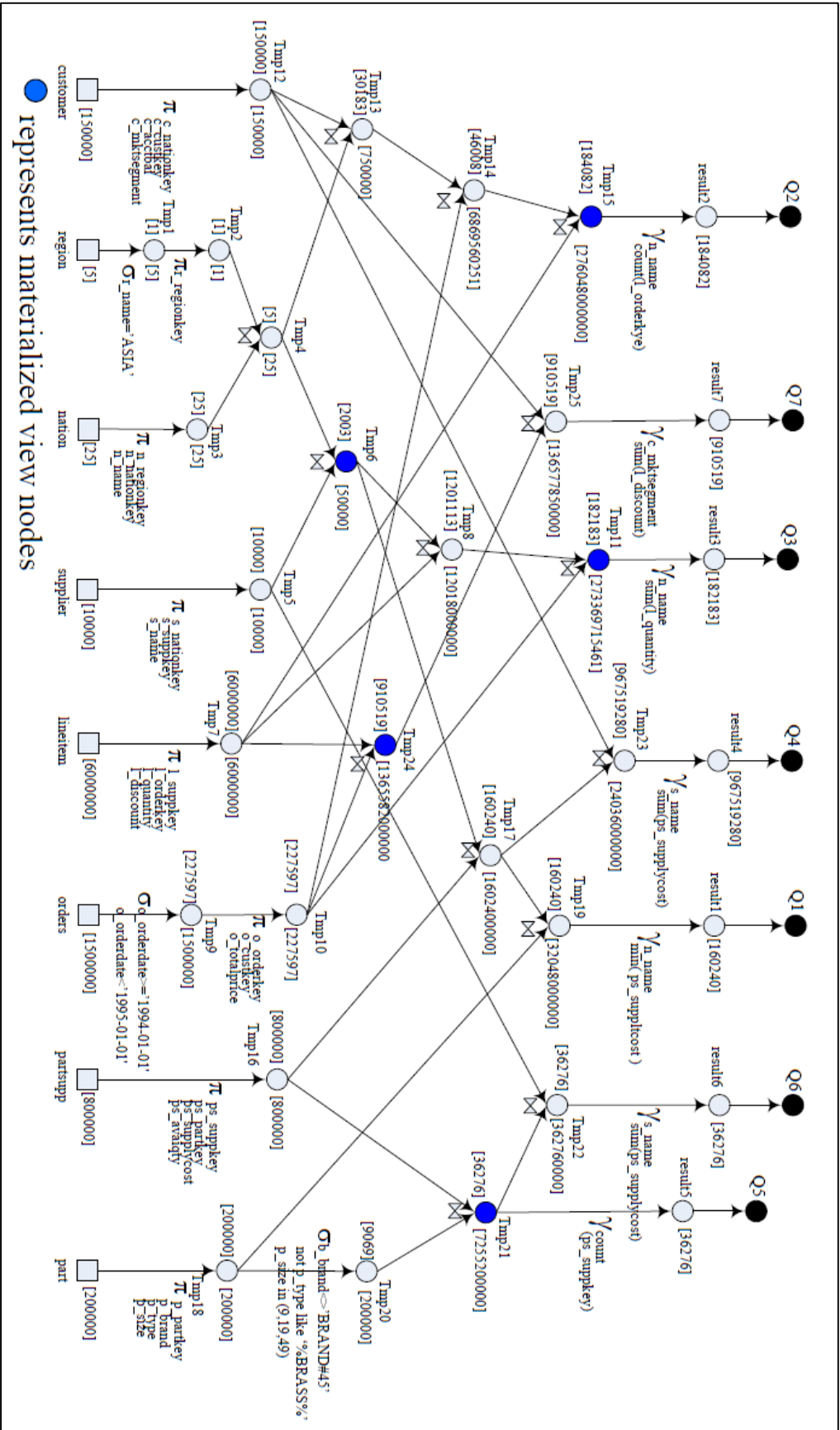


Figure (5.3): MVPP with Materialized View Nodes Selected by 2PO

To calculate the total cost of MVPP in Figure 5.5 and Figure 5.6, we return to Maintenance and Query Processing Costs of Re-Optimized MVPP that are calculated in (Suchyukorn, 2013), which are illustrated in Tables 5.2, and 5.3.

Table (5.2): The Maintenance Cost of the Re-Optimized MVPP

| Materialized View | Number of Base Relations | Constructed from Nodes | Maintenance Cost |
|-------------------------------|--------------------------|---|--------------------------|
| Tmp6 | 3 | Tmp1, Tmp2, Tmp3, Tmp4, Tmp5, Tmp6 | 180,168 |
| Tmp11 | 5 | Tmp1, Tmp2, Tmp3, Tmp4, Tmp5, Tmp6, Tmp7, Tmp8, Tmp9, Tmp10, Tmp11 | 1,426,977,515,570 |
| Tmp15 | 5 | Tmp1, Tmp2, Tmp3, Tmp4, Tmp12, Tmp13, Tmp9, Tmp10, Tmp14, Tmp7, Tmp15 | 1,414,630,939,520 |
| Tmp21 | 2 | Tmp16, Tmp18, Tmp20, Tmp21 | 14,512,800,000 |
| Tmp24 | 2 | Tmp7, Tmp9, Tmp10, Tmp24 | 2,731,179,455,194 |
| Total Maintenance Cost | | | 5,587,300,890,452 |

Source: (Suchyukorn, 2013, p. 77)

Table 5.2 shows the maintenance cost of each materialized node of re-optimized MVPP in Figure 5.3 that are (tmp6, tmp11, tmp15, tmp21, and tmp24). However, it also shows the number of base relations that could affect the materialized node. For example, node Tmp6 has three base relations which are Region, Nation, and Supplier. The total of MVPP Maintenance Cost was introduced by (Yang, Karlapalem, & Li, 1997) and is calculated by the formula:

$$C_{maintenance} = \sum_{v \in M} f_u C_m(v) \quad (5.1)$$

Where:

M is the set of materialized views,
 f_u is the frequency of updating base relations,
 $C_m(v)$ is the cost of maintenance when v is materialized.

For Tmp6, $C_m(Tmp6)$, it is constructed on three base relations, and accesses nodes Tmp1, Tmp2, Tmp3, Tmp4, Tmp5 and the node itself. The cost of each node is 5, 1, 25, 25, 10000 and 50000, respectively as shown in Figure 5.3. Thus, the materialized view maintenance cost of Tmp6 is $3*(5 + 1 + 25 + 25 + 10000 + 50000)$ that is 180,168. Other materialized nodes costs calculated similarly.

Table (5.3): The Query Processing Cost of the Re-Optimized MVPP

| Query | Access from Node | Query Processing Cost |
|------------------------------------|---|------------------------|
| Query number 1 (Q1) | Tmp6, Tmp16, Tmp17, Tmp18, Tmp19, result1 | 67,303,124,486 |
| Query number 2 (Q2) | Tmp15, result2 | 2,208,984 |
| Query number 3 (Q3) | Tmp11, result3 | 2,550,562 |
| Query number 4 (Q4) | Tmp6, Tmp16, Tmp17, Tmp12, Tmp23, result4 | 53,213,742,566 |
| Query number 5 (Q5) | Tmp21, result5 | 362,760 |
| Query number 6 (Q6) | Tmp21, Tmp5, Tmp22, result6 | 3,265,582,968 |
| Query number 7 (Q7) | Tmp24, Tmp12, Tmp25, result7 | 409,739,463,114 |
| Total Query Processing Cost | | 533,527,035,440 |

Source: (Suchyukorn, 2013, p. 77)

On the other hand, the Query Processing Cost is calculated by:

$$C_{queryprocessing} = \sum_{q \in Q} f_q C_q(M) \quad (5.2)$$

Where:

M is the set of materialized views,
 Q is the set of queries,
 f_q is the frequency of executing queries,
 $C_q(M)$ is the cost to compute q from the set of materialized views M .

For Query Q1 in Table 5.3, its total frequency of executing the query is 2, Q1 accesses the nodes named Tmp6, Tmp16, Tmp17, Tmp18, Tmp19, and result1. The cost of each node is 2003, 800000, 1602400000, 200000, 32048000000, and 160240, respectively. So, the query processing cost of query Q1, $C_q(Q1)$ is $(2)*(2003 + 800000$

+ 1602400000 + 200000 + 32048000000 + 160240) = 67303124486. The cost of Tmp6 is the number of resulting rows because it is a materialized node. Similarly, other costs are calculated and listed in Table 5.3.

The total cost of the MVPP is calculated by (Yang, Karlapalem, & Li, 1997):

$$C_{total} = \sum_{q \in Q} f_q C_q(M) + \sum_{v \in M} f_u C_m(v) \quad (5.3)$$

That is the total cost of the re-optimized MVPP by Suchyukorn, (Suchyukorn, 2013), is:

$$\begin{aligned} C_{total} &= C_{queryprocessing} + C_{maintenance} \\ &= 533,527,035,440 + 5,587,300,890,452 = 6,120,827,925,892 \end{aligned}$$

To show the effect of our proposed model on the total cost of resulted MVPP, we need to design the usage frequency matrix of queries, Q1 to Q7. Assuming these 7 queries are executed over a fiscal year, and the features needed to be selected are [QUARTER, MONTH_OF_QUARTER, WEEK_OF_QUARTER] as shown in Figure 5.4.

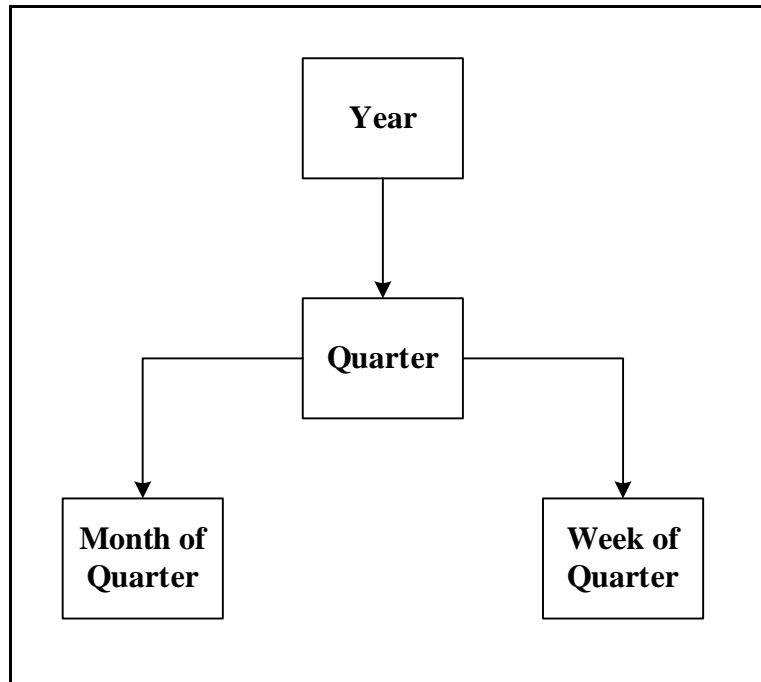


Figure (5.4): TPC-H Usage Frequency Hierarchy

Considering the usage frequencies are as listed in Table 5.4. Each query in the table have total number of executions over the years after detracting values according to our model.

Table (5.4): Usage Frequency Matrix of Queries Q1 through Q7 Over Financial Year's Quarter Weeks

| | 1 st Quarter | 2 nd Quarter | 3 rd Quarter | 4 th Quarter |
|-----------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| 1 st week | Q2(6), Q3(7), Q7(3) | Q5(5), Q6(9) | Q1(2) | Q2(6) |
| 2 nd week | Q1(2), Q6(9) | Q4(2) Q7(3) | Q6(9) | Q5(5), Q3(3) |
| 3 rd week | ... | ... | ... | |
| ... | ... | ... | ... | |
| 14 th week | ... | ... | ... | |

In those days with features similar to 1st week of 1st quarter in fiscal year, the list of views selected to be materialized in the re-optimized MVPP will be changed. Thus, views that flagged to be materialized in the trees of queries other than Q2, Q3, and Q7 will be canceled. In this case, tmp21 in Figure 5.3 will be unflagged as being materialized view. The resulted MVPP will be as shown in Figure 5.5.

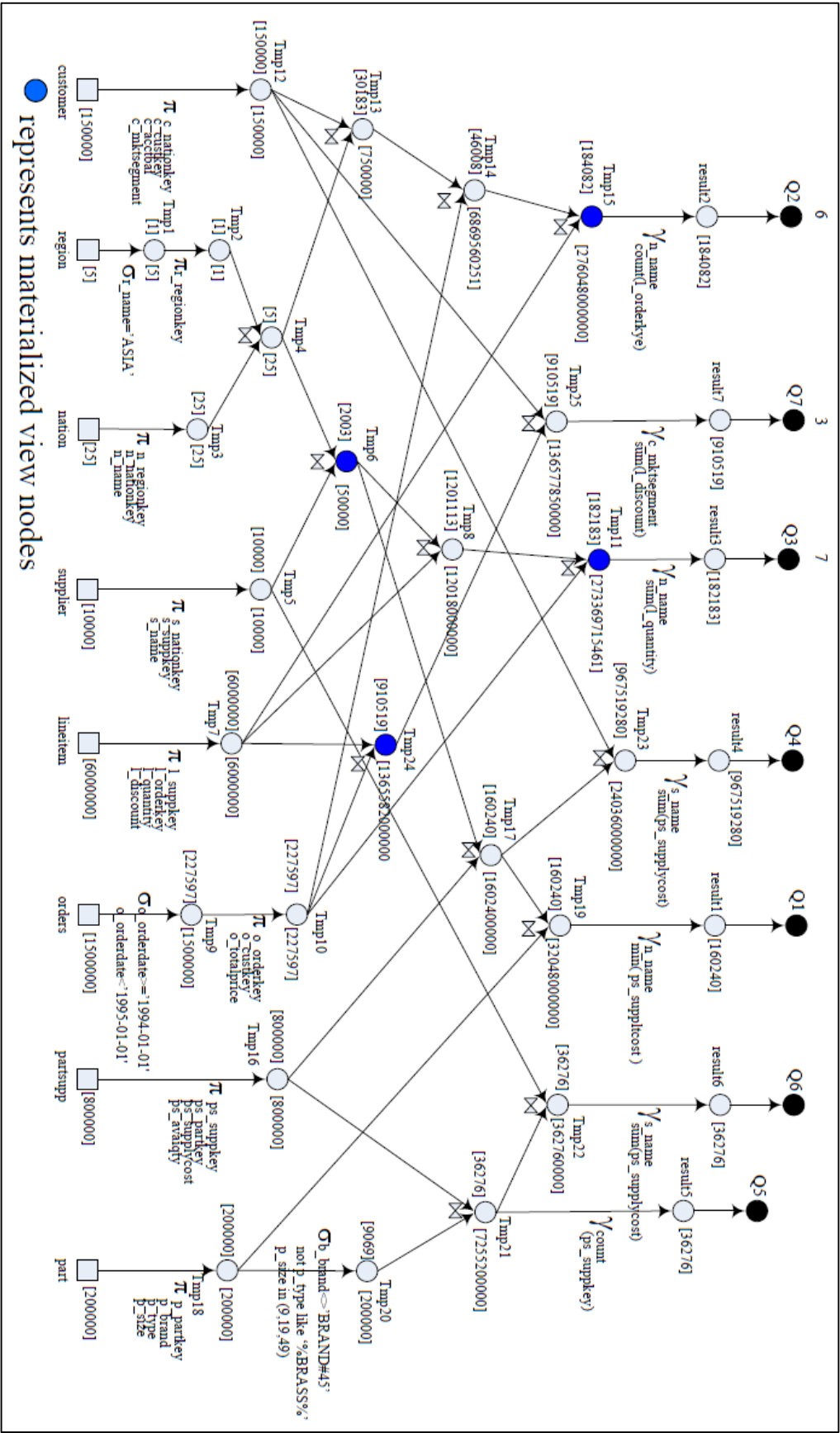


Figure (5.5): MVPP in 1st Week of 1st Quarter

The total maintenance cost of MVPP in Figure 5.5 is calculated in same way of the re-optimized MVPP in Table 5.2.

Table 5.5 illustrates the maintenance costs of 5 materialized nodes shown in Figure 5.5. Total Maintenance Cost, $C_{maintenance}$, is 5,572,788,090,452.

Table (5.5): The Maintenance Cost of the MVPP in 1st Week of 1st Quarter

| Materialized View | Number of Base Relations | Constructed from Nodes | Maintenance Cost |
|-------------------------------|--------------------------|---|--------------------------|
| Tmp6 | 3 | Tmp1, Tmp2, Tmp3, Tmp4, Tmp5, Tmp6 | 180,168 |
| Tmp11 | 5 | Tmp1, Tmp2, Tmp3, Tmp4, Tmp5, Tmp6, Tmp7, Tmp8, Tmp9, Tmp10, Tmp11 | 1,426,977,515,570 |
| Tmp15 | 5 | Tmp1, Tmp2, Tmp3, Tmp4, Tmp12, Tmp13, Tmp9, Tmp10, Tmp14, Tmp7, Tmp15 | 1,414,630,939,520 |
| Tmp24 | 2 | Tmp7, Tmp9, Tmp10, Tmp24 | 2,731,179,455,194 |
| Total Maintenance Cost | | | 5,572,788,090,452 |

Since the usage frequency of queries Q1, Q4, Q5, and Q6 is zero in 1st week of 1st quarter (checking days), then the query processing cost for those queries is 0. Table 5.6 shows the query processing cost of Queries Q1 to Q7. The total query processing cost, $C_{queryprocessing}$, of this MVPP is 409,744,222,660.

Table (5.6): The Query Processing Cost of the MVPP in 1st Week of 1st Quarter

| Query | Access from Node | Query Processing Cost |
|---------------------|---|-----------------------|
| Query number 1 (Q1) | Tmp6, Tmp16, Tmp17, Tmp18, Tmp19, result1 | 0 |
| Query number 2 (Q2) | Tmp15, result2 | 2,208,984 |
| Query number 3 (Q3) | Tmp11, result3 | 2,550,562 |
| Query number 4 (Q4) | Tmp6, Tmp16, Tmp17, Tmp12, Tmp23, result4 | 0 |

| | | |
|------------------------------------|--|------------------------|
| Query number 5 (Q5) | Tmp21, Tmp20, Tmp18, Tmp16, result5 | 0 |
| Query number 6 (Q6) | Tmp21, Tmp20, Tmp18, Tmp16, Tmp5, Tmp22, result6 | 0 |
| Query number 7 (Q7) | Tmp24, Tmp12, Tmp25, result7 | 409,739,463,114 |
| Total Query Processing Cost | | 409,744,222,660 |

From above discussion, we can summarize the total cost of the MVPP in 1st week of 1st quarter is:

$$C_{total} = C_{queryprocessing} + C_{maintenance}$$

$$= 409,744,222,660 + 5,572,788,090,452 = 5,982,532,313,112$$

To calculate the total cost of the MVPP in 2nd week of the 1st quarter, Figure 5.6 illustrates the nodes which should be materialized. Nodes Tmp6 and Tmp21 are selected to be materialized according to our usage frequency matrix. The total maintenance cost of these two nodes is $180,168 + 14,512,800,000 = 14,512,980,168$.

Table (5.7): The Maintenance Cost of the MVPP in 2nd Week of 1st Quarter

| Materialized View | Number of Base Relations | Constructed from Nodes | Maintenance Cost |
|-------------------------------|--------------------------|------------------------------------|-----------------------|
| Tmp6 | 3 | Tmp1, Tmp2, Tmp3, Tmp4, Tmp5, Tmp6 | 180,168 |
| Tmp21 | 2 | Tmp16, Tmp18, Tmp20, Tmp21 | 14,512,800,000 |
| Total Maintenance Cost | | | 14,512,980,168 |

Similar to previous steps of Table 5.6, the total query processing cost of the MVPP in 2nd week of 1st quarter is the query processing costs of Q1 and Q6.

$$C_{queryprocessing} = 67,303,124,486 + 3,265,582,968 = 600,830,159,926$$

Table (5.8): The Query Processing Cost of the MVPP in 2nd Week of 1st Quarter

| Query | Access from Node | Query Processing Cost |
|------------------------------------|--|------------------------|
| Query number 1 (Q1) | Tmp6, Tmp16, Tmp17, Tmp18, Tmp19, result1 | 67,303,124,486 |
| Query number 2 (Q2) | Tmp15, Tmp7, Tmp14, Tmp10, Tmp9, Tmp13, Tmp4, Tmp3, Tmp2, Tmp1, Tmp12, result2 | 0 |
| Query number 3 (Q3) | Tmp11, Tmp10, Tmp9, Tmp8, Tmp7, Tmp6, result3 | 0 |
| Query number 4 (Q4) | Tmp6, Tmp16, Tmp17, Tmp12, Tmp23, result4 | 0 |
| Query number 5 (Q5) | Tmp21, result5 | 0 |
| Query number 6 (Q6) | Tmp21, Tmp5, Tmp22, result6 | 3,265,582,968 |
| Query number 7 (Q7) | Tmp24, Tmp10, Tmp9, Tmp7, Tmp12, Tmp25, result7 | 0 |
| Total Query Processing Cost | | 600,830,159,926 |

From the above discussion, we can summarize the total cost of the MVPP in 2nd week of 1st quarter as:

$$\begin{aligned}
 C_{total} &= C_{queryprocessing} + C_{maintenance} = 600,830,159,926 + 14,512,980,168 \\
 &= 615,343,140,094
 \end{aligned}$$

Table (5.9): The Query Processing Cost, Maintenance Cost and Total Cost of 1st and 2nd Weeks of 1st Quarter

| | Cost of Query Processing | Cost of Maintenance | Total Cost |
|---|--------------------------|---------------------|-------------------|
| All-Virtual Views | 8,427,206,080,471 | 0 | 8,427,206,080,471 |
| All-Materialized Views | 1,940,978,234 | 7,686,779,440,303 | 7,688,720,418,537 |
| 2PO | 533,527,035,440 | 5,587,300,890,452 | 6,120,827,925,892 |
| 1 st week of 1 st quarter | 409,744,222,660 | 5,572,788,090,452 | 600,830,159,926 |
| 2 nd week of 1 st quarter | 600,830,159,926 | 14,512,980,168 | 615,343,140,094 |

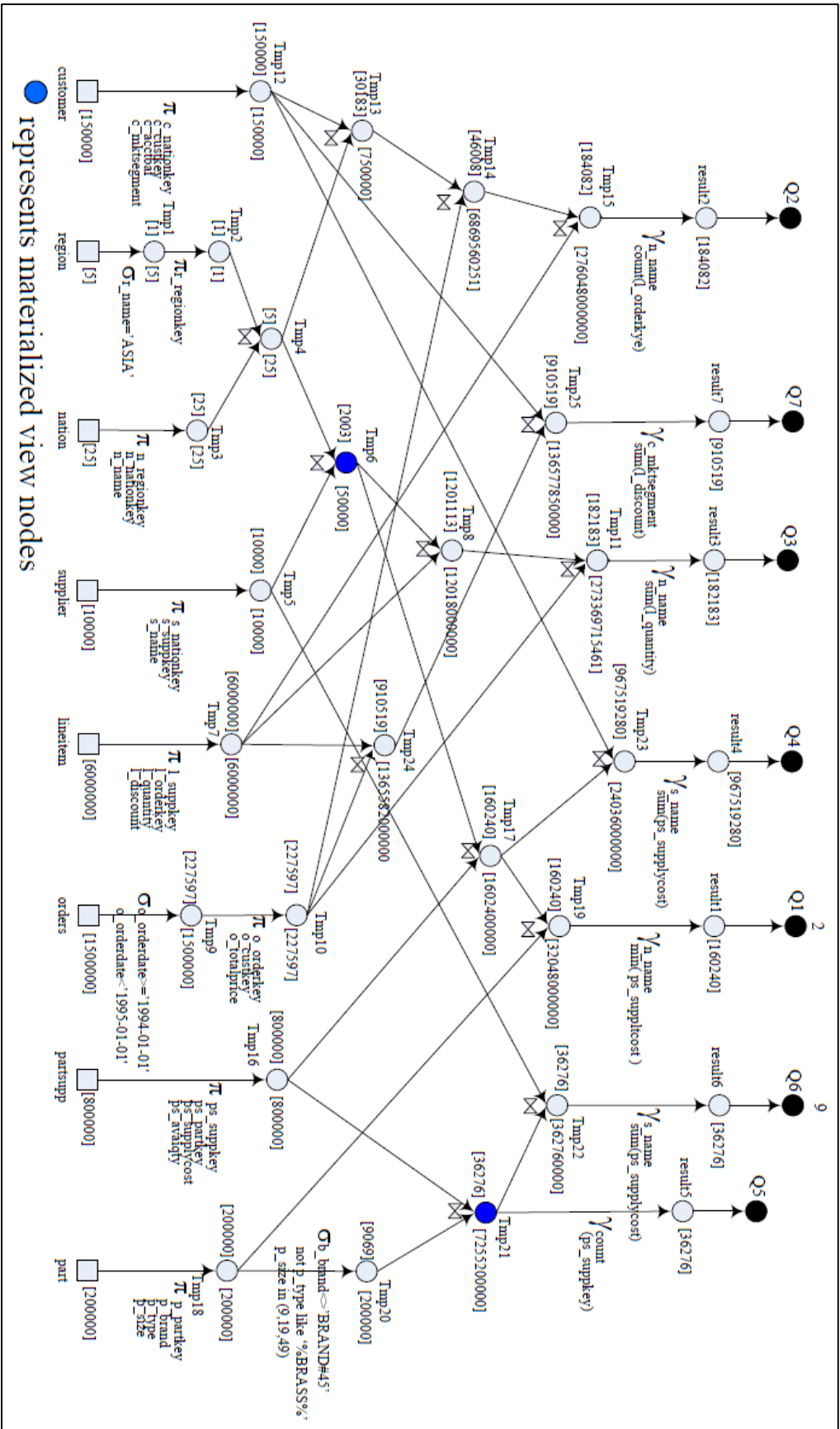


Figure (5.6): MVPP in 2nd Week of 1st Quarter

5.2 Legacy System versus Data Warehouse versus DW Materialized Views

In this section, we compare IUG database's performance before and after implementing the data warehouse. We show the time consumed and total query cost for each data mart using legacy system, then we show how the data warehouse optimized it. Additionally, we show the effect of materializing data warehouse's views on the database performance for each data mart.

The time is recorded while executing each query, it is calculated by Oracle Optimizer. The cost of queries is also calculated by the Oracle's optimizer, Cost-Based Optimizer CBO, which is a function of the CPU_COST and IO_COST as discussed in (Oracle Corporation, 2014) and (Hellström, 2017), it represents units of work or resource used in an operation. The optimizer uses disk I/O, CPU usage, and memory usage as units of work.

- **High School Results Data Mart**

For high school results data mart, we executed the query for last year, last three years, last ten years, and all years (empty filter). The results are shown in Table 5.10.

Table (5.10): High School Results Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|-------|--------------------------|------|-------------------------------|------|
| | | time (s) | cost | time (s) | cost | time (s) | cost |
| last year | 20305 | 0.322 | 2811 | 0.085 | 58 | 0.030 | 6 |
| last three years | 60457 | 0.511 | 6344 | 0.111 | 162 | 0.051 | 6 |
| last ten years (max) | 247075 | 1.198 | 10112 | 0.245 | 273 | 0.081 | 7 |

From the results above, data warehouse returns the results four to five times faster than legacy system. Data warehouse gives even better performance as data grows as shown in Figure 5.7.

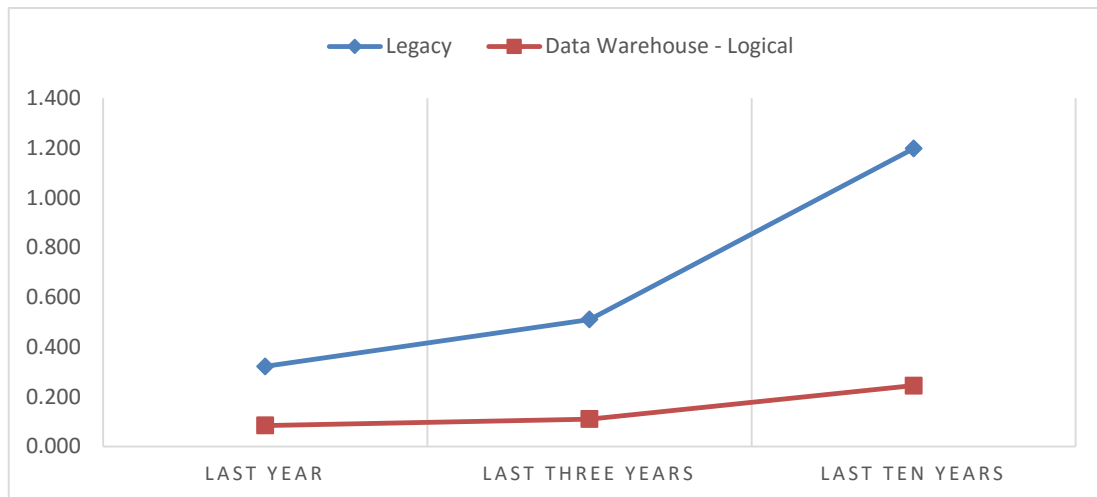


Figure (5.7): High School Results - Legacy System Vs Data Warehouse

On the other hand, materializing the data warehouse's views gives the system better performance. For high school results data mart, materializing its views would save up to 67% of the execution time over data warehouse's tables. As shown in Figure 5.8, it may give even better performance as data grows.

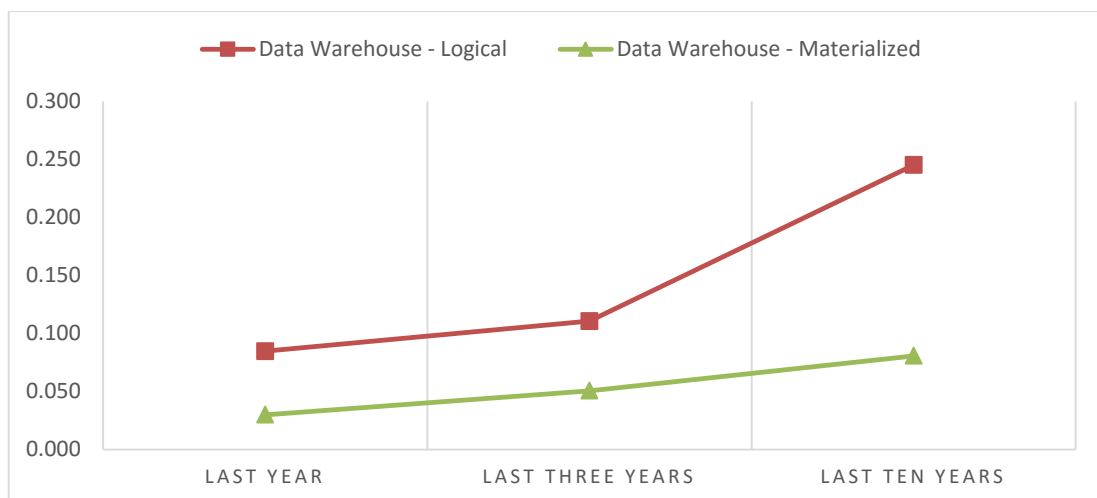


Figure (5.8): High School Results - Logical Vs Materialized

- **Students Registration Data Mart**

We've executed students' registration query over last semester, last three semesters, and last five years as shown in Table 5.11.

Table (5.11): Students Registration - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|-------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 93 | 14.768 | 2984 | 0.018 | 9 | 0.008 | 9 |
| last three semesters | 360 | 46.907 | 6119 | 0.035 | 15 | 0.019 | 9 |
| last five years | 1208 | 217.147 | 10944 | 0.093 | 15 | 0.058 | 9 |

From the results above, data warehouse returns the results about 2000 times faster than legacy system, this is due to the complex PLSQL functions that used to calculate students' registration hours in legacy system, while in the data warehouse the data is prepared and saved instantly into data warehouse after semester starts. Figure 5.9 compares time consumed in legacy system with time consumed in the data warehouse to return students' registration data.

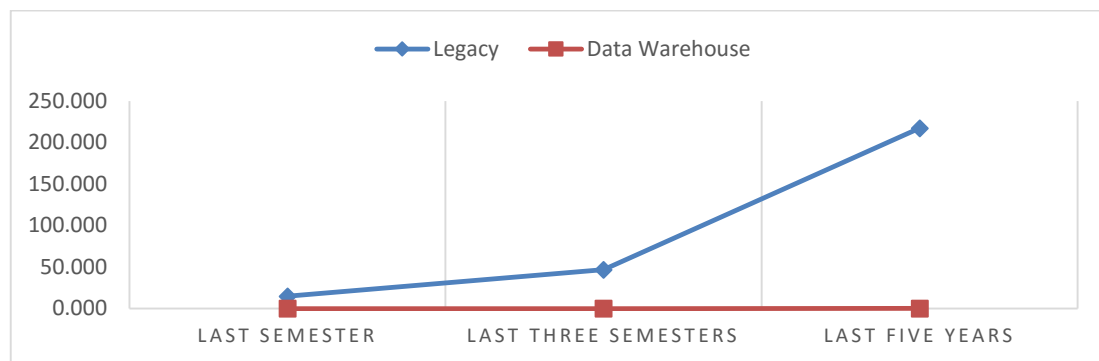


Figure (5.9): Students Registration - Legacy System Vs Data Warehouse

On the other hand, materializing the data warehouse's views gives the system better performance. For students' registration data mart, materializing its views would save up to 56% of the execution time over data warehouse's tables. As shown in Figure 5.10, it may give even better performance as data grows.

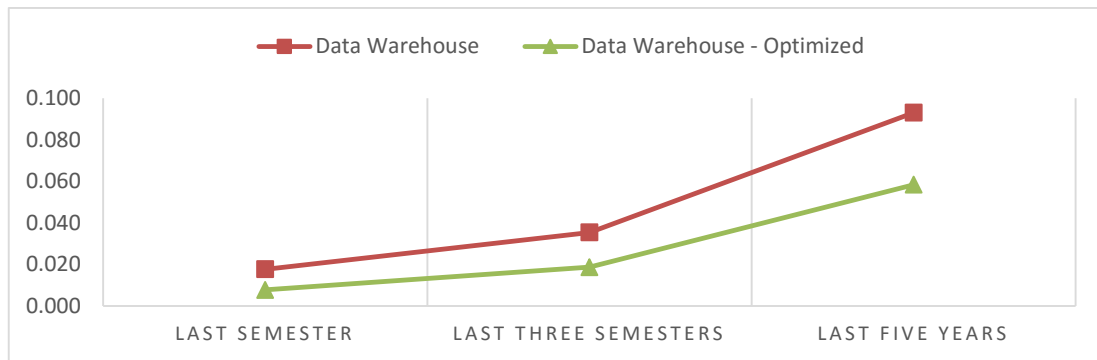


Figure (5.10): Students Registration - Logical Vs Materialized

The rest of experimental results about other data marts are listed in Appendix 1: Legacy System Vs Data Warehouse Vs Materialized Views. In Table 5.12 we summarize the performance improvement of each data mart in two cases:

- Legacy System Vs Data Warehouse: the effectiveness of using data warehouse.
- Data Warehouse Logical Views Vs Data Warehouse Materialized Views: we show how materializing data warehouse's views can give even better performance.

Table (5.12): Summary of the Comparison between IUG Legacy System, Data Warehouse, and Materialized Views

| Fact | Legacy vs DW | | DW Logical vs Materialized | |
|--|---------------------|-------------------|----------------------------|-------------------|
| | Faster (× times) | Saved Time (%) | Faster (× times) | Saved Time (%) |
| High School Results | 4.89 | 79.54 | 3.04 | 67.07 |
| Students Admission | 36.30 | 97.25 | 6.56 | 84.75 |
| Students Registration | 2334.91 | 99.96 | 2.30 | 56.60 |
| Registered Students Geographical Location | 40.29 | 97.52 | 16.67 | 94.00 |
| Transfers between Colleges | 9.69 | 89.68 | 5.15 | 80.57 |
| Exam Conflicts | 22.78 | 95.61 | 6.30 | 84.13 |
| SFD Students Registration | 211.74 | 99.53 | 1.72 | 41.71 |
| Cash Grants | 84.75 | 98.82 | 5.75 | 82.61 |
| Deferred Grant | 27.11 | 96.31 | 1.75 | 42.86 |
| Student Fund Summary | 14.72 | 93.21 | 18.50 | 94.59 |

| | | | | |
|-----------------------------|-------|-------|-------|-------|
| Financial Collection | 69.77 | 98.57 | 17.55 | 94.30 |
|-----------------------------|-------|-------|-------|-------|

From Table 5.12 we can conclude that, using data warehouse in IUG saves more than 90% of total time used in legacy system for most of data marts. On the other hand, materializing these data marts can save even more than 80% of total time consumed while using the data warehouse in IUG for most of data marts.

Chapter 6

Conclusion and Future Works

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In this thesis, we discussed the importance of data warehouse for organizations and companies. We highlighted two of industry approaches for designing DW which are top-down and bottom-up approaches, and we showed the advantages and disadvantages of each approach. For IUG, we adopted bottom-up (Ralph Kimball) approach. Also, we discussed case studies of implementing DW in higher education environments and their approaches of the implementation.

This research, in current phase, focuses on designing and implementing data warehouse which will help mainly in decision-making process that is related to the students. We developed data marts for some of the departments that are core for students, which are Admission and Registration, Student Affairs, and Finance departments. Other departments in IUG will be considered in future phases.

Additionally, we argue number of materialized view selection algorithms. We found that most of works in this field have been provided different approaches for the selection of views to materialize considering view maintenance cost and storage space. There is lack in researches in selection of views to be materialized based on dynamic thresholds such as view's usage frequency. Thus, we designed a model for materialized view selection based on view's usage frequency in different periods over academic/financial year.

Our experiments show that the proposed model can be integrated with existing materialized view selection algorithms to give better materialization decisions, and even better database's performance since it restrict the set of views before materialization process.

Finally, we discussed the effect of the data warehouse on the IUG database's performance and query processing time. When comparing the results of running the queries, it is clear in every instance that the data return dramatically quicker from the organized star schema in the data warehouse than from the transactional database.

Adding to that, materializing data warehouse's views will give even better performance of overall data warehouse system.

6.2 Future Works

In the current phase and due to limited time frame of this research, we designed and implemented a data warehouse for three departments only in IUG. They are Admission and Registration, Students Affairs, and Finance departments. In the future, we will complete the design and the implementation of the data warehouse for other departments in IUG.

Additionally, for our dynamic materialized view selection model, we would like to design an algorithm to calibrate the similarity ratio, value of δ , in Algorithm 1 dynamically. This algorithm would be based on correct and wrong decisions about views materialization.

Moreover, since the current model is running in daily basis to check for views' materialization, we would like to design a dynamic model to detect the appropriate time intervals for this process.

Finally, the proposed model accepts list of features for the similarity check algorithm. This list of features is static for all views. We are planning to make this list of features to be related for each view.

The Reference List

References

- Agrawal, S., Chaudhuri, S., & Narasayya, V. (2000). Automated Selection of Materialized Views and Indexes for SQL Databases. *26th International Conference on Very Large Databases* (pp. 496-505). Cairo: Morgan Kaufmann.
- Al-Kordi, Z. (2017, January 1). Admission and Registration acting manager. (B. Shbair, Interviewer)
- Aouiche, K., Jouve, P.-E., & Darmont, J. (2006). Clustering-Based Materialized View Selection in Data Warehouses. *ADBIS'06 Proceedings of the 10th East European conference on Advances in Databases and Information Systems* (pp. 81-95). Thessaloniki: Springer.
- Ballard, C., Farrell, D. M., Gupta, A., Mazuela, C., & Vohnik, S. (2006). *Dimensional Modeling: In a Business Intelligence Environment*. IBM Redbooks.
- Breslin, M. (2004). Data Warehousing Battle of the Giants: Comparing the Basics of the Kimball and Inmon Models. *Business Intelligence Journal*.
- Burton, P., & Green, S. (2016, December 12). *Meta Data: The Key To Data Warehouse Design*. Retrieved from Institute for Systems Research: <http://www.isr.umd.edu/Courses/ENSE623/DataWarehouse>
- Chan, S. S. (1999). The Impact of Technology on Users and the Workplace. *New Directions for Institutional Research*, 3-21.
- Choudhari, Y. D., & Shrivastava, S. K. (2012). Cluster Based Approach for Selection of Materialized Views. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(7), 315-318.
- Derakhshan, R., Stantic, B., Korn, O., & Dehne, F. (2008). Parallel Simulated Annealing for Materialized View Selection in Data Warehousing Environments. *ICA3PP 2008: Algorithms and Architectures for Parallel Processing*, 121-132.
- Devlin, B. (2010). Beyond business intelligence. *Business Intelligence Journal*.
- Eckerson, W. (2003). *Smart Companies in the 21st Century: The Secrets of Creating Successful Business Intelligence Solutions*. Seattle: A 101communications Publication.
- Ganapavarapu, V. B. (2014). *Designing and Implementing a Data Warehouse using Dimensional Modeling*. Albuquerque: The University of New Mexico.
- Gong, A., & Zhao, W. (2008). Clustering-based Dynamic Materialized View Selection Algorithm. *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference* (pp. 391-395). Shandong, China: IEEE.
- Gou, G., Yu, J. X., & Lu, H. (2006, May). A* search: an efficient and flexible approach to materialized view selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(3), 411-425.
- Gupta, H. (1997). Selection of views to materialize in a data warehouse. *Database Theory - ICDT '97*, 1186, 98-112.
- Gupta, H., & Mumick, I. S. (2005, January). Selection of Views to Materialize in a Data Warehouse. *IEEE Transactions on Knowledge and Data Engineering*, 17(1), 24-43.
- Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques*. Waltham: Elsevier.
- Hellström, I. (2017, March 08). *Oracle SQL & PL/SQL Optimization for Developers Documentation*. Retrieved from Read the Docs: <https://media.readthedocs.org/pdf/oracle/latest/oracle.pdf>

- Himanshu, G., & Mumick, I. S. (1998). Selection of Views to Materialize Under a Maintenance Cost Constraint. *Database Theory - ICDT'99 - 7th International Conference* (pp. 453-470). Jerusalem: Springer .
- IBM Business Analytics. (2011). *A Step-by-Step Approach to Successful Business Intelligence*. Retrieved from IDG Enterprise: http://resources.idgenterprise.com/original/AST-0066459_YTW03194CAEN.pdf
- IBM Knowledge Center. (2017, January 24). Retrieved from IBM Knowledge Center: https://www.ibm.com/support/knowledgecenter/SS9UM9_9.1.0/com.ibm.datatools dimensional.ui.doc/topics/c_dm_design_cycle_2_idgrain.html
- Inmon, W. H. (2002). *Building the Data Warehouse*. New York: Wiley.
- Kalnis, P., Mamoulis, N., & Papadias, D. (2002). View selection using randomized search. *Data & Knowledge Engineering*, 89-111.
- Kimball, R. (1996). *The Data Warehouse Toolkit*.
- Kimball, R., & Caserta, J. (2011). *The Data Warehouse ETL Toolkit*. John Wiley & Sons.
- Kimball, R., & Ross, M. (2010). *The Kimball Group Reader; Relentlessly Practical Tools for Data Warehousing and Business Intelligence*. Indianapolis: Wiley.
- Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The definitive guide to dimensional modeling*. Indianapolis: Wiley.
- Lee, M. (2001). Speeding Up Materialized View Selection In Data Warehouses Using a Randomized Algorithm. *International Journal of Cooperative Information Systems*, 10(3), 327-353.
- Leonard, E. M. (2011). *Design and Implementation of an Enterprise Data Warehouse*. Wisconsin: Marquette University.
- Lijuan, Z., Xuebin, G., Linshuang, W., & Qian, S. (2009). Efficient Materialized View Selection Dynamic Improvement Algorithm. *Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference* (pp. 294-297). Tianjin, China: IEEE.
- Moody, D. L., & Kortink, M. A. (2000). From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design.
- Morzy, T. (2012, January 1). *Slowly Changing Dimension*. Retrieved from Technologie Przetwarzania Danych: http://tpd.cs.put.poznan.pl/accounts/pdf/SPHD/ModelowanieHD/Slowly_Changing_Dimensions.pdf
- Moullin, M. (2007). Performance measurement definitions. *International Journal of Health Care Quality Assurance*, 3(20), 181-183.
- Oracle Corporation. (2014, June). *Oracle Database Performance Tuning Guide*. Oracle.
- Phuboon-ob, J. (2009). *Materialized View Selection Based on Two-Phase Optimization*. National Institute of Development Administration.
- Phuboon-ob, J., & Auepanwiriyaikul, R. (2007). Two-Phase Optimization for Selecting Materialized Views in a Data Warehouse. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 1(1), 119-123.
- Poe, V., Klauer, P., & Brobst, S. (1996). *Building a Data Warehouse for Decision Support*. Prentice Hall.
- Rainardi, V. (2008). *Building a Data Warehouse: With Examples in SQL Server*. Apress.

- Ramakrishnan, T., Jones, M. C., & Sidorova, A. (2012). Factors influencing business intelligence (BI) data collection strategies: An empirical investigation. *Decision Support Systems*, 486–496.
- Rangarajan, S. (2016, September 1). *Data Warehouse Design – Inmon versus Kimball*. Retrieved from The Data Administration Newsletter: <http://tdan.com/data-warehouse-design-inmon-versus-kimball/20300>
- Ribeiro, J. S. (2016). *Business Intelligence to support NOVA IMS Academic Services BI System*. Lisbon: Universidade Nova de Lisboa.
- Santos, M. Y., & Ramos, I. (2009). *Business Intelligence - Tecnologias da Informação na Gestão do Conhecimento*. Lisbon: Editora de Informática.
- Shwedeh, K. (2017, January 1). Director of Academic Affairs. (B. Shbair, Interviewer)
- Suchyukorn, B. (2013). *Dynamic materialized view selection based on two-phase optimization*. Bangkok: National Institute of Development Administration.
- Suchyukorn, B., & Auepanwiriyaikul, R. (2013). Re-Optimization MVPP Using Common Subexpression for Materialized View Selection. *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, 7(7), 1152-1159.
- Suknović, M., Čupić, M., Martić, M., & Krulj, D. (2005). Data Warehousing and Data Mining - A Case Study. *Yugoslav Journal of Operations Research*, 125-145.
- The Microsoft Data Warehouse Toolkit, 2nd Edition*. (2017, January 27). Retrieved from The Kimball Group: <http://www.kimballgroup.com/data-warehouse-business-intelligence-resources/books/microsoft-data-warehouse-dw-toolkit/>
- TPC Benchmark H. (2017, May 15). Retrieved from Transaction Processing Performance Council (TPC): http://www.tpc.org/TPC_Documents_Current_Versions/pdf/TPC-H_v2.17.2.pdf
- Valluri, S. R., Vadapalli, S., & Karlapalem, K. (2002, February). View Relevance Driven Materialized View Selection in Data Warehousing Environment. *Australian Computer Science Communications*, 24(2), 187 - 196.
- Vizgaitytė, G., & Skyrius, R. (2012). Business Intelligence in the Process of Decision Making: Changes and Trends. *EKONOMIKA*, 147-157.
- Wang, Z., & Zhang, D. (2005). Optimal Genetic View Selection Algorithm Under Space Constraint. *International Journal of Information Technology*, 11(5), 44-51.
- Yang, J., Karlapalem, K., & Li, Q. (1997). Algorithms for Materialized View Design in Data Warehousing Environment.
- Yang, J.-H., & Chung, I.-J. (2006, June). ASVMRT: Materialized View Selection Algorithm in Data Warehouse. *International Journal of Information Processing Systems*, 2(2), 67-75.
- Yu, J. X., Yao, X., Choi, C.-H., & Gou, G. (2003, November). Materialized View Selection as Constrained Evolutionary Optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 33(4).
- Zahra, K. (2015, June 20). *Inmon or Kimball? The debate of the century*. Retrieved from Kurt Zahra: <https://kurtzahra.com/2015/06/20/inmon-or-kimball-the-debate-of-the-century/>
- Zhang, C., Yao, X., & Yang, J. (2001, August). An evolutionary approach to materialized views selection in a Data Warehouse Environment. *IEEE Transactions On Systems, Man, And Cybernetics—part C*, 31(3), 282-294.

Appendices

Appendix 1: Legacy System Vs Data Warehouse Vs Materialized Views

A1.1 Students Admission Data Mart

Table (A1.1): Students Admission - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|------------------|---------|---------------|------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last year | 432 | 0.178 | 536 | 0.006 | 41 | 0.001 | 5 |
| last three years | 1895 | 0.564 | 1059 | 0.020 | 45 | 0.003 | 5 |
| last ten years | 8804 | 1.283 | 2175 | 0.035 | 47 | 0.009 | 5 |
| ALL | 23982 | 1.893 | 2175 | 0.082 | 50 | 0.031 | 5 |

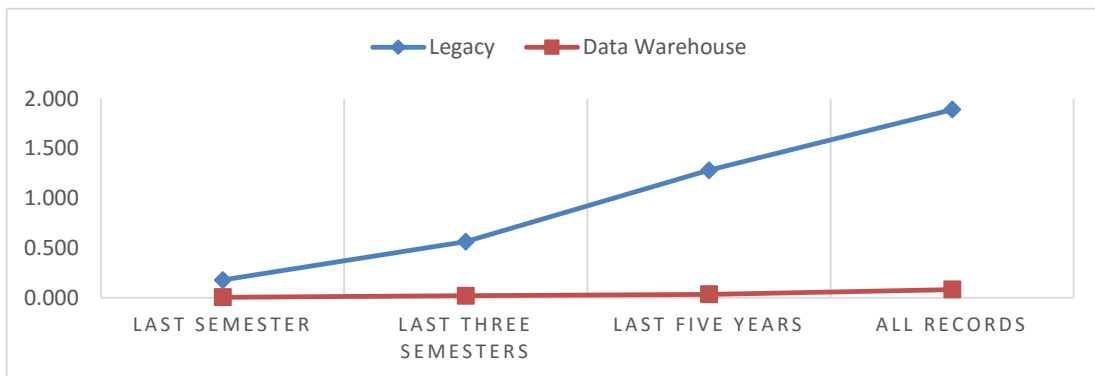


Figure (A1.1): Students Admission - Legacy System Vs Data Warehouse

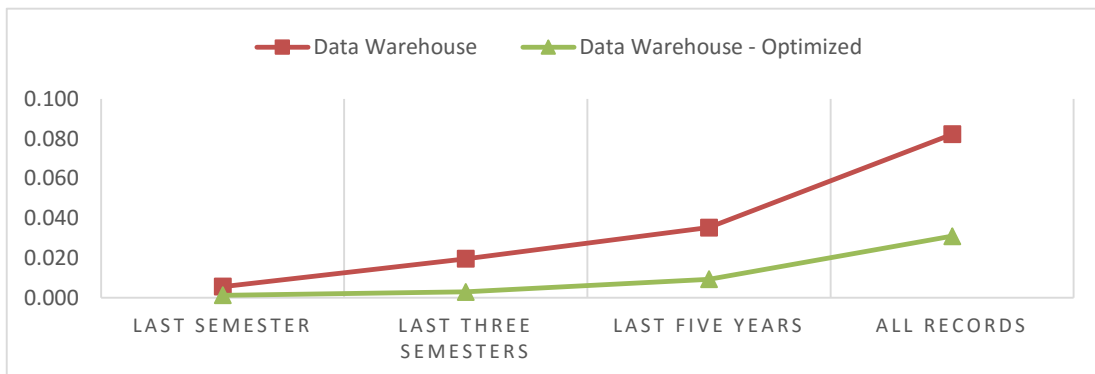


Figure (A1.2): Students Admission - Logical View Vs Materialized View

A1.2 Geographical Location Data Mart

Table (A1.2): Geographical Location- Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|-------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 1809 | 0.355 | 2738 | 0.017 | 80 | 0.001 | 4 |
| last three semesters | 5419 | 1.067 | 5144 | 0.028 | 87 | 0.002 | 4 |
| last five years | 17685 | 2.283 | 9239 | 0.057 | 304 | 0.004 | 4 |
| all records | 77706 | 8.174 | 21771 | 0.220 | 2294 | 0.025 | 4 |

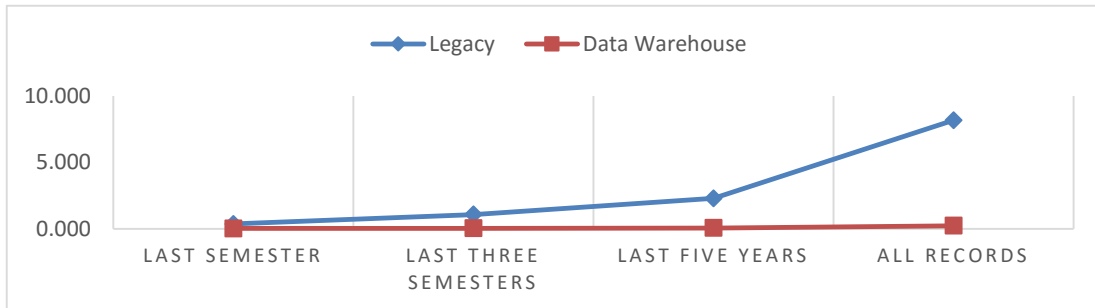


Figure (A1.3): Geographical Location- Legacy System Vs Data Warehouse

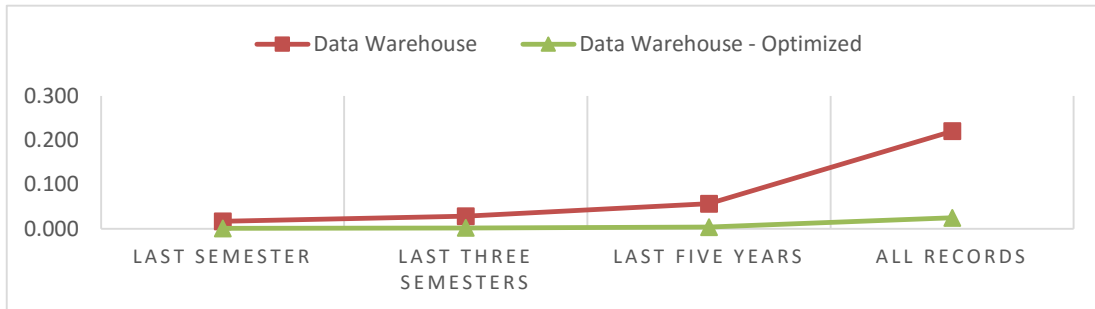


Figure (A1.4): Geographical Location- Logical View Vs Materialized View

A1.3 Transfers between Colleges Data Mart

Table (A1.3): Transfers between Colleges - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|------------------|---------|---------------|------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last year | 224 | 0.110 | 617 | 0.011 | 33 | 0.005 | 5 |
| last three years | 735 | 0.132 | 622 | 0.024 | 34 | 0.012 | 5 |
| last ten years | 3052 | 0.303 | 1372 | 0.065 | 239 | 0.035 | 5 |
| ALL | 11480 | 1.111 | 1391 | 0.290 | 538 | 0.056 | 5 |

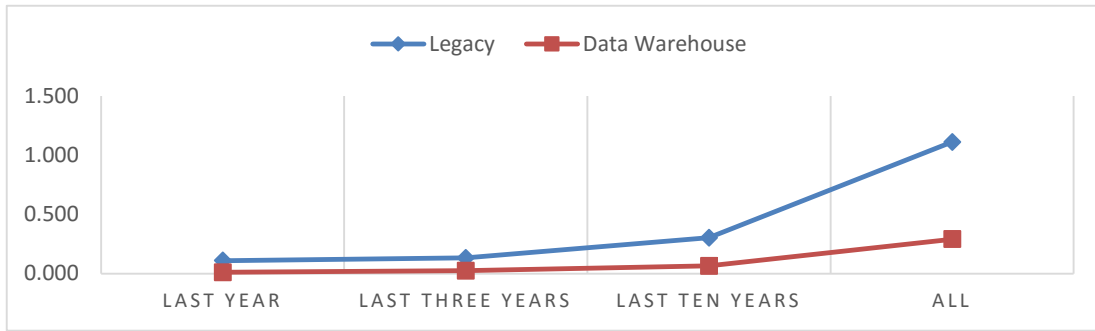


Figure (A1.5): Transfers between Colleges - Legacy System Vs Data Warehouse

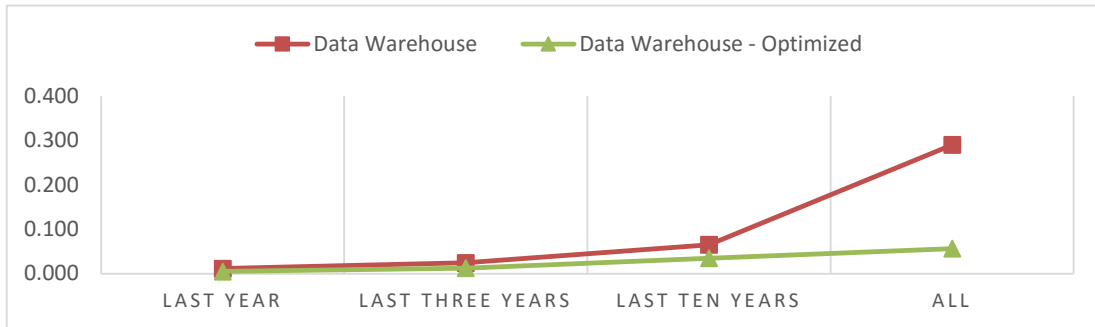


Figure (A1.6): Transfers between Colleges- Logical View Vs Materialized View

A1.4 Exam Conflicts Data Mart

Table (A1.4): Exam Conflicts - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|-------|--------------------------|-------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 1016 | 0.232 | 1677 | 0.014 | 56 | 0.003 | 5 |
| last three semesters | 3077 | 0.600 | 2883 | 0.026 | 60 | 0.007 | 5 |
| last five years | 10368 | 1.293 | 5305 | 0.061 | 271.5 | 0.019 | 5 |
| ALL | 44593 | 4.643 | 11581 | 0.255 | 1416 | 0.041 | 5 |

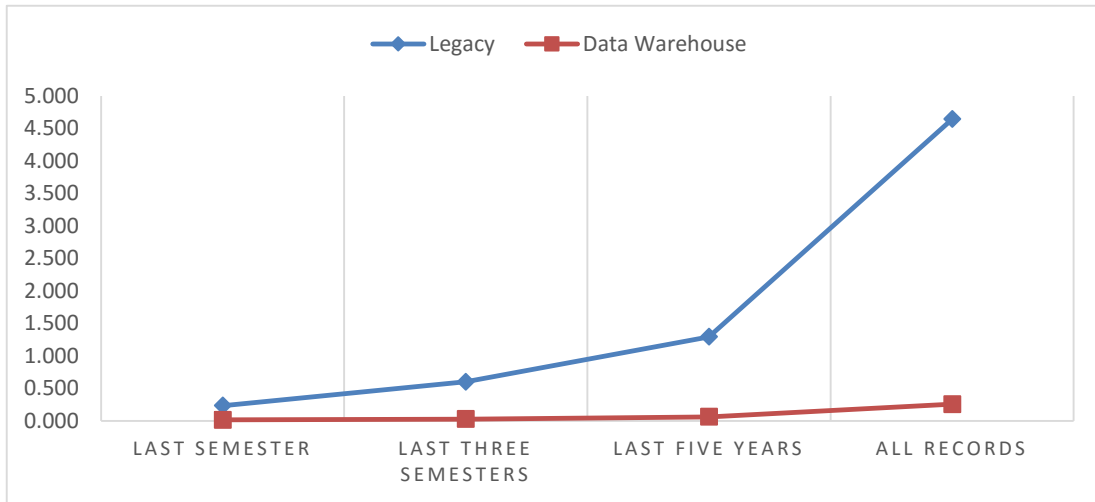


Figure (A1.7): SFD Exam Conflicts - Legacy System Vs Data Warehouse

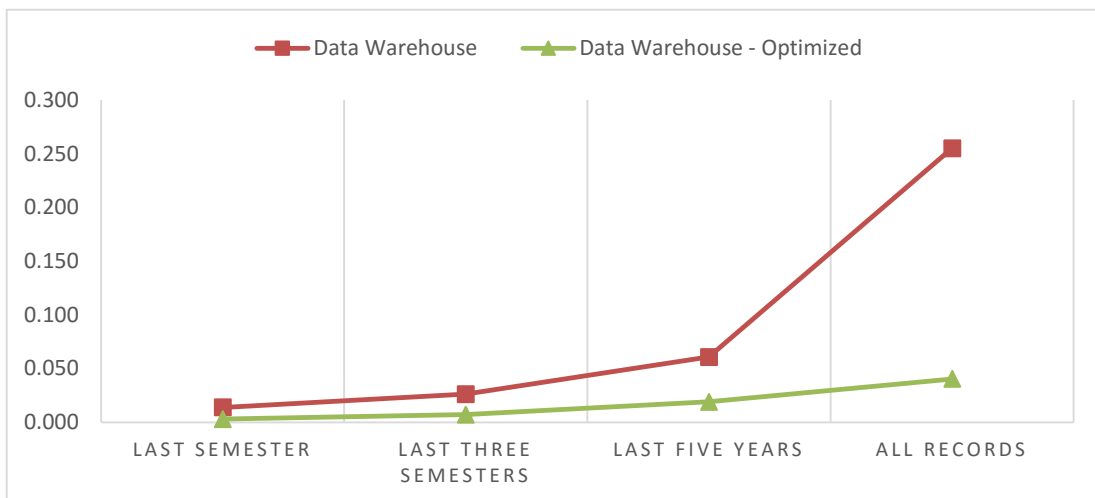


Figure (A1.8): Exam Conflicts - Logical View Vs Materialized View

A1.5 SFD Students Registration Data Mart

Table (A1.5): SFD Students Registration - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|-------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 23338 | 10.067 | 2984 | 0.048 | 6 | 0.028 | 6 |
| last three semesters | 65860 | 18.022 | 6119 | 0.088 | 15 | 0.051 | 15 |
| last five years | 231172 | 46.347 | 10944 | 1.094 | 21 | 0.638 | 21 |
| ALL | 845925 | 115.930 | 38251 | 1.773 | 35 | 1.033 | 35 |

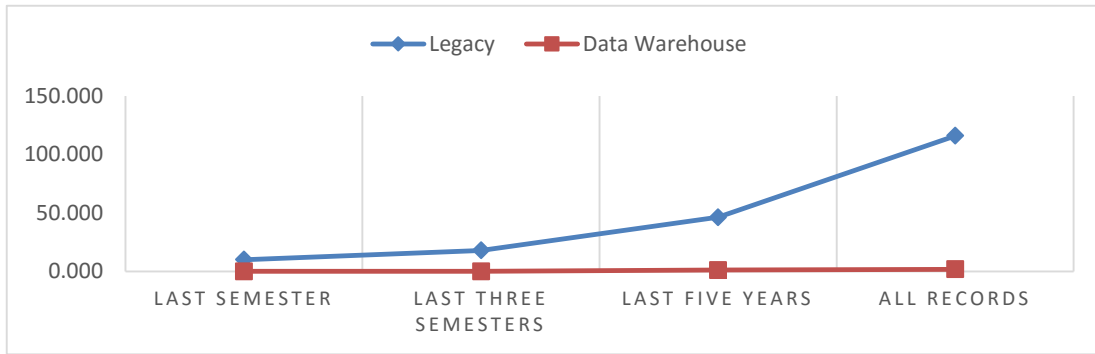


Figure (A1.9): SFD Students Registration - Legacy System Vs Data Warehouse

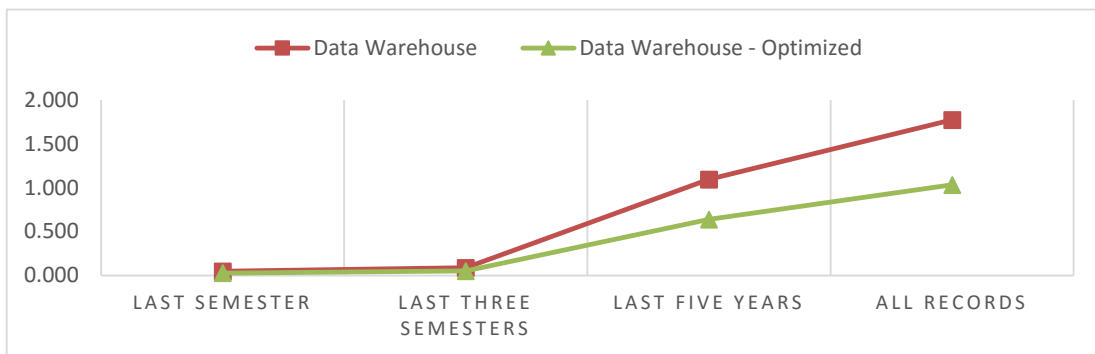


Figure (A1.10): SFD Students Registration - Logical View Vs Materialized View

A1.6 Cash Grants Data Mart

Table (A1.6): Cash Grants - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 287 | 0.226 | 870 | 0.003 | 4 | 0.001 | 3 |
| last three semesters | 861 | 0.296 | 1107 | 0.008 | 3 | 0.002 | 3 |
| last five years | 1954 | 0.342 | 1550 | 0.015 | 3 | 0.003 | 3 |
| ALL | 2533 | 0.358 | 1644 | 0.017 | 3 | 0.003 | 3 |

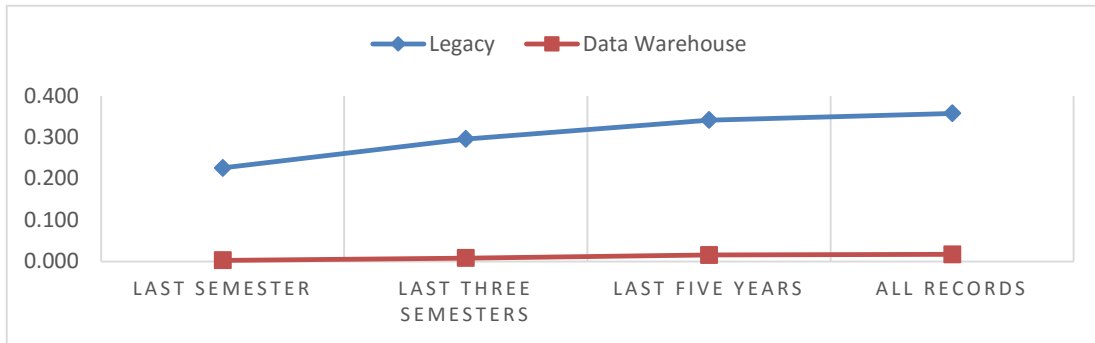


Figure (A1.11): Cash Grants - Legacy System Vs Data Warehouse

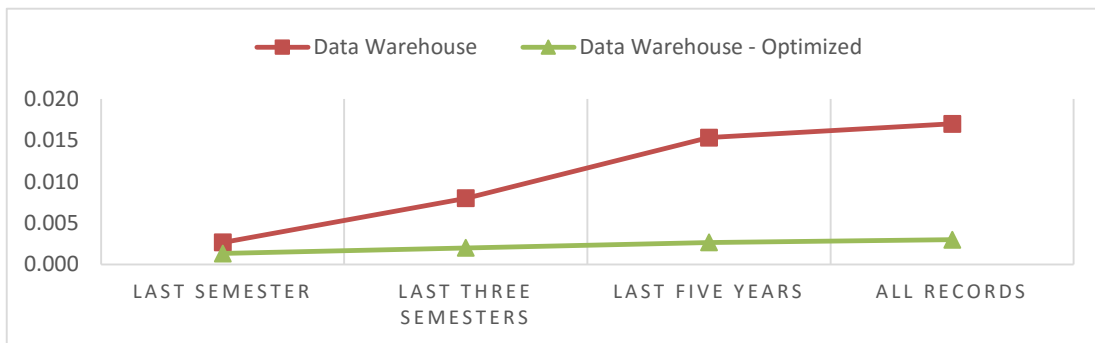


Figure (A1.12): Cash Grants - Logical View Vs Materialized View

A1.7 Deferred Grant Data Mart

Table (A1.7): Deferred Grant - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 2 | 0.043 | 311 | 0.002 | 4 | 0.001 | 4 |
| last three semesters | 6 | 0.067 | 320 | 0.003 | 3 | 0.002 | 3 |
| last five years | 16 | 0.081 | 334 | 0.003 | 3 | 0.002 | 3 |
| ALL | 27 | 0.093 | 335 | 0.004 | 3 | 0.003 | 3 |

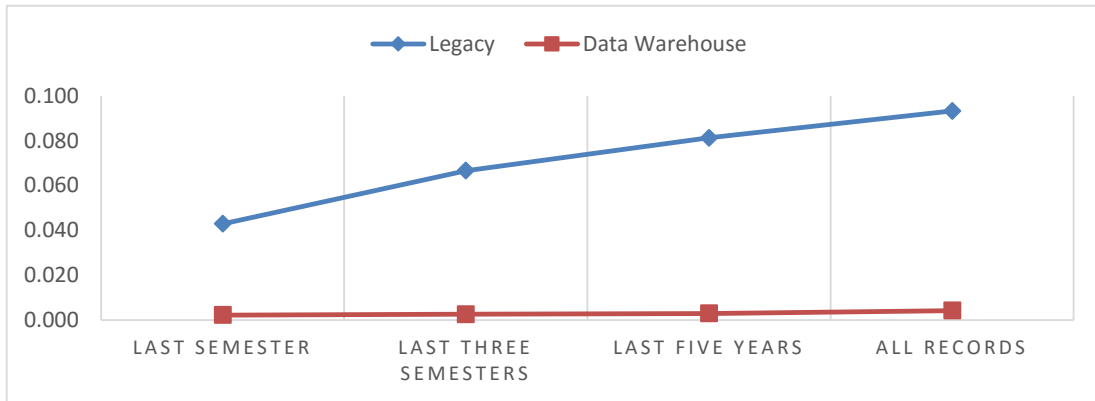


Figure (A1.13): Deferred Grant - Legacy System Vs Data Warehouse

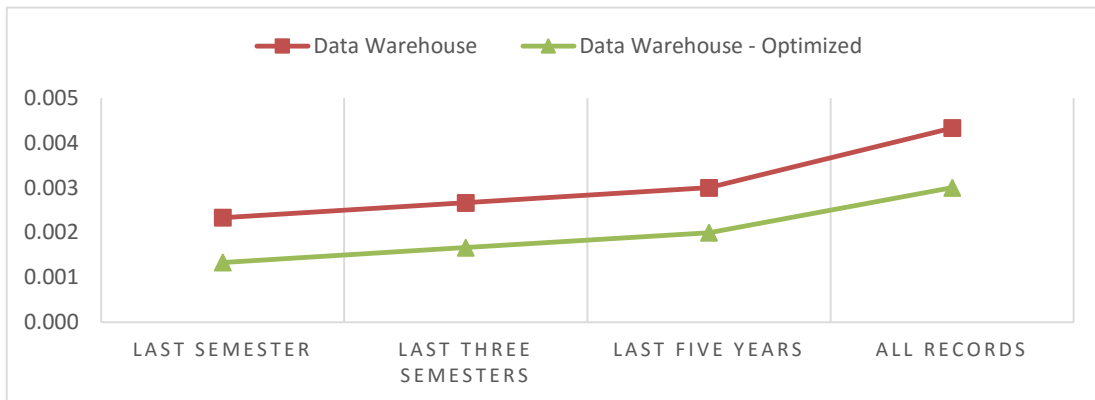


Figure (A1.14): Deferred Grant - Logical View Vs Materialized View

A1.8 Student Fund Summary Data Mart

Table (A1.8): Student Fund Summary - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|-------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 4191 | 4.024 | 41722 | 0.273 | 124 | 0.032 | 6 |
| last three semesters | 14379 | 4.465 | 46446 | 0.931 | 309 | 0.087 | 7 |
| last five years | 62343 | 9.018 | 60797 | 1.875 | 522 | 0.101 | 7 |
| ALL | 84432 | 11.618 | 81211 | 2.238 | 684 | 0.210 | 9 |

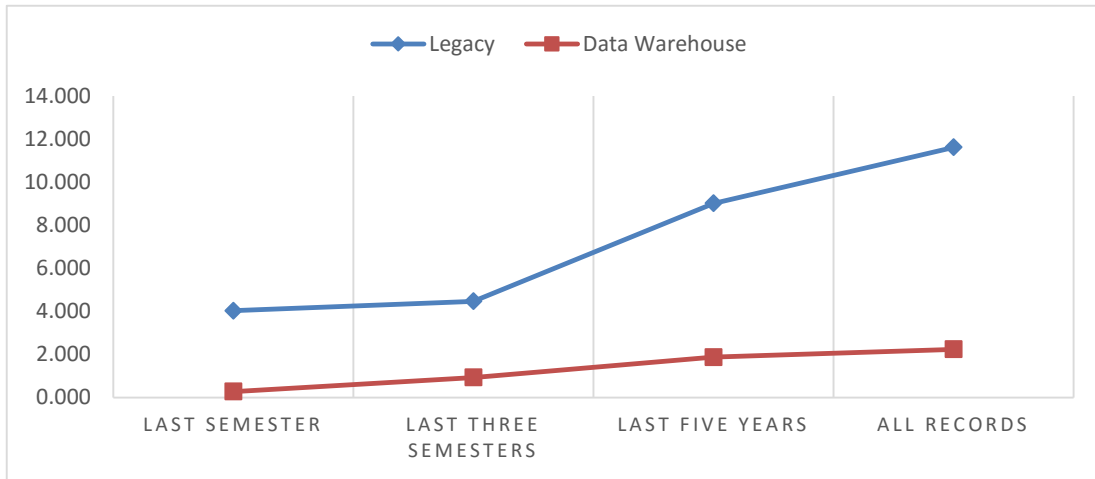


Figure (A1.15): Student Fund Summary - Legacy System Vs Data Warehouse

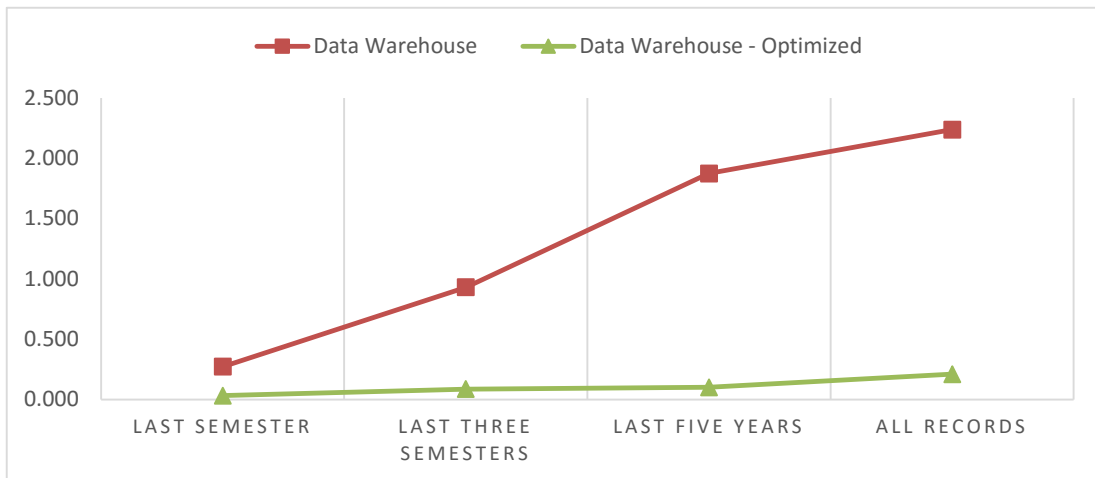


Figure (A1.16): Student Fund Summary - Logical View Vs Materialized View

A1.9 Financial Collection Data Mart

Table (A1.9): Financial Collection - Experimental Results

| Filters | Records | Legacy System | | Data Warehouse - Logical | | Data Warehouse - Materialized | |
|----------------------|---------|---------------|-------|--------------------------|------|-------------------------------|------|
| | | time | cost | time | cost | time | cost |
| last semester | 12462 | 0.912 | 12568 | 0.033 | 152 | 0.002 | 5 |
| last three semesters | 31540 | 2.553 | 19386 | 0.064 | 643 | 0.004 | 5 |
| last five years | 110651 | 6.533 | 25470 | 0.159 | 643 | 0.010 | 5 |
| ALL | 442283 | 27.629 | 41529 | 0.396 | 646 | 0.034 | 5 |

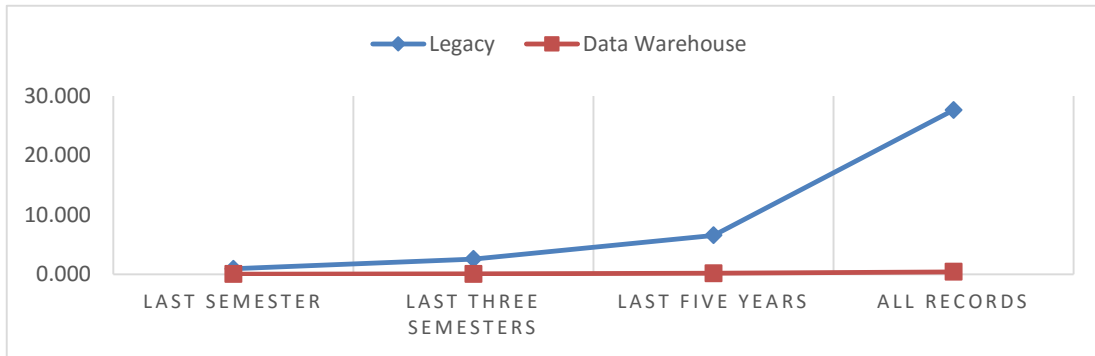


Figure (A1.17): Financial Collection - Legacy System Vs Data Warehouse

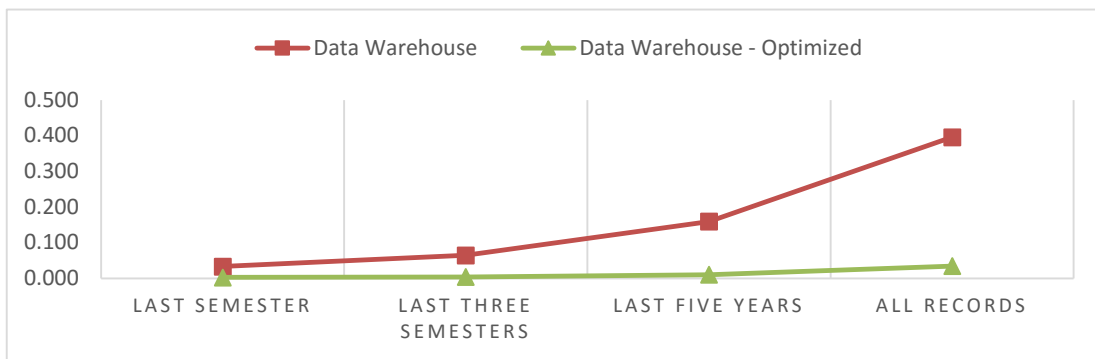


Figure (A1.18): Financial Collection - Logical View Vs Materialized View

Appendix 2: TPC-H Schema Queries and Relational Algebra Trees

A2.1 Query Q2:

```

SELECT n_name, COUNT(l_orderkey)
FROM   customer, orders, lineitem, nation, region
WHERE  c_custkey = o_custkey
      AND o_orderkey = l_orderkey
      AND c_nationkey = n_nationkey
      AND n_regionkey = r_regionkey
      AND r_name = 'asia'
      AND o_orderdate >= '1994-01-01'
      AND o_orderdate < '1995-01-01'
GROUP BY n_name;

```

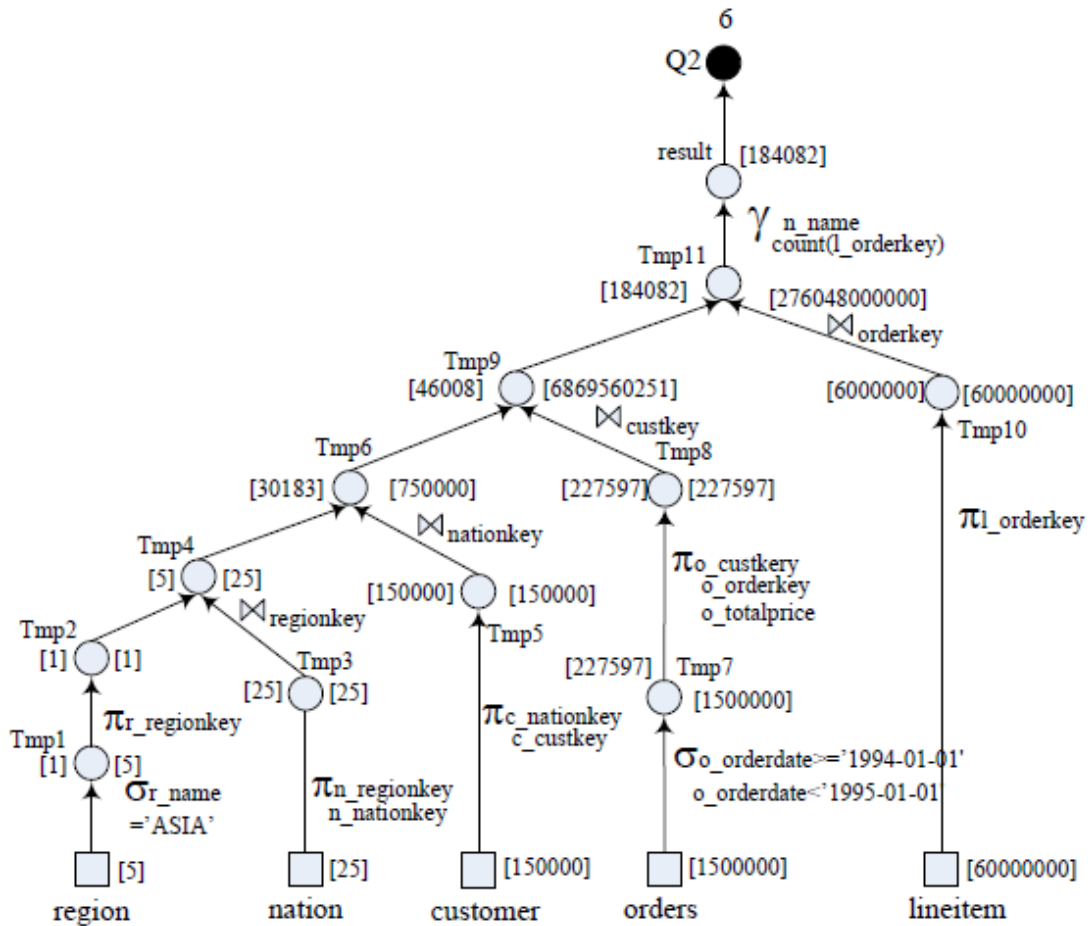


Figure (A2.1): Relational Algebra Query Tree of Query Q2

A2.2 Query Q3:

```

SELECT n_name, Sum(l_quantity)
FROM   orders, lineitem, supplier, nation, region
WHERE  o_orderkey = l_orderkey
      AND l_suppkey = s_suppkey
      AND s_nationkey = n_nationkey
      AND n_regionkey = r_regionkey
      AND r_name = 'asia'
      AND o_orderdate >= '1994-01-01'
      AND o_orderdate < '1995-01-01'
GROUP BY n_name;

```

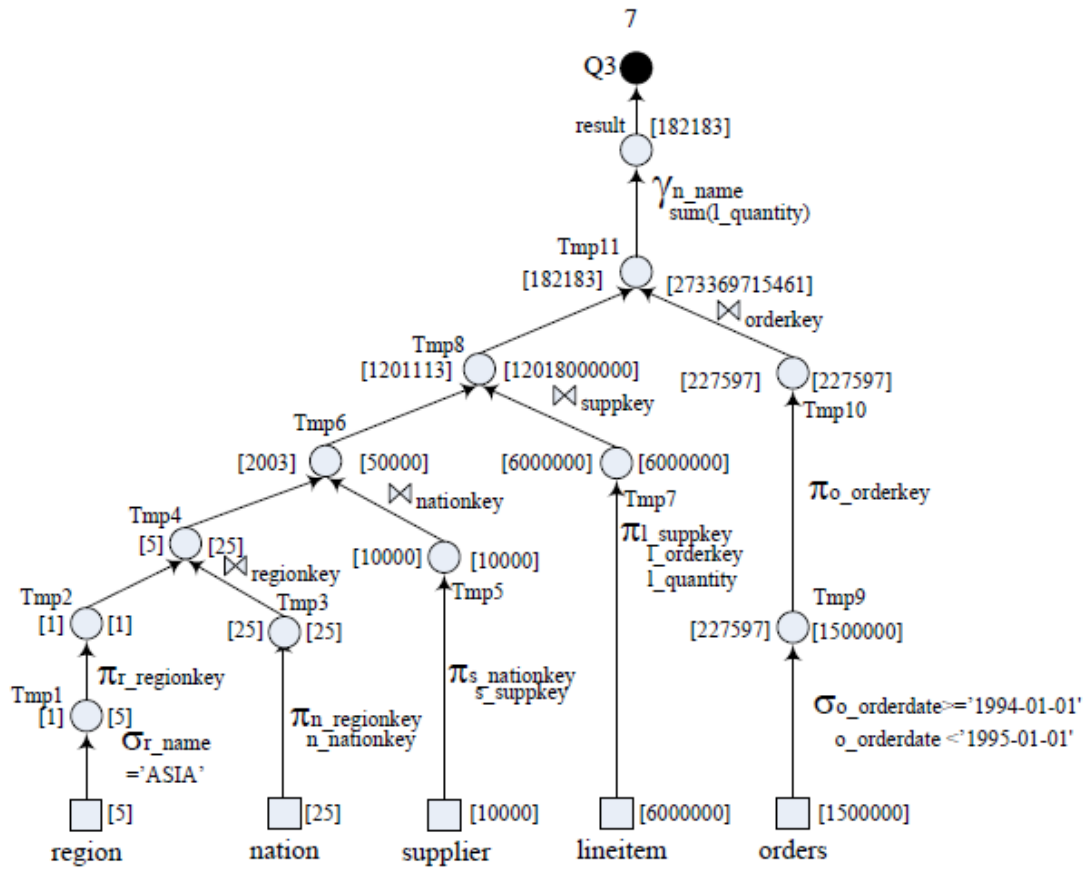


Figure (A2.2): Relational Algebra Query Tree of Query Q3

A2.3 Query Q4:

```

SELECT s_name, Sum(ps_supplycost)
FROM   partsupp, supplier, customer, nation, region
WHERE  ps_suppkey = s_suppkey
      AND c_nationkey = s_nationkey
      AND s_nationkey = n_nationkey
      AND n_regionkey = r_regionkey
      AND r_name = 'asia'
GROUP BY s_name;

```

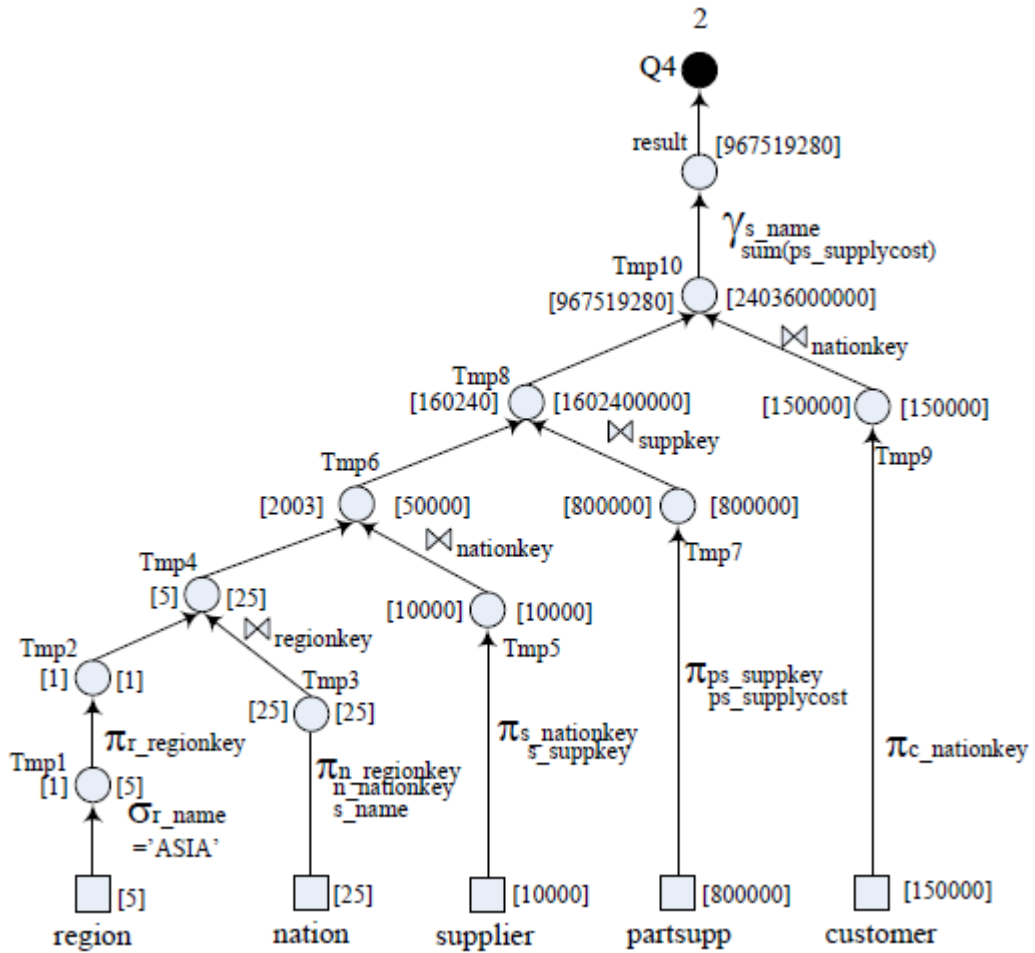


Figure (A2.3): Relational Algebra Query Tree of Query Q4

A2.4 Query Q5:

```
SELECT Count(ps_suppkey)
FROM   partsupp, part
WHERE  p_partkey = ps_partkey
      AND p_brand <> 'brand#45'
      AND NOT p_type LIKE '%brass%'
      AND p_size IN ( 9, 19, 49 );
```

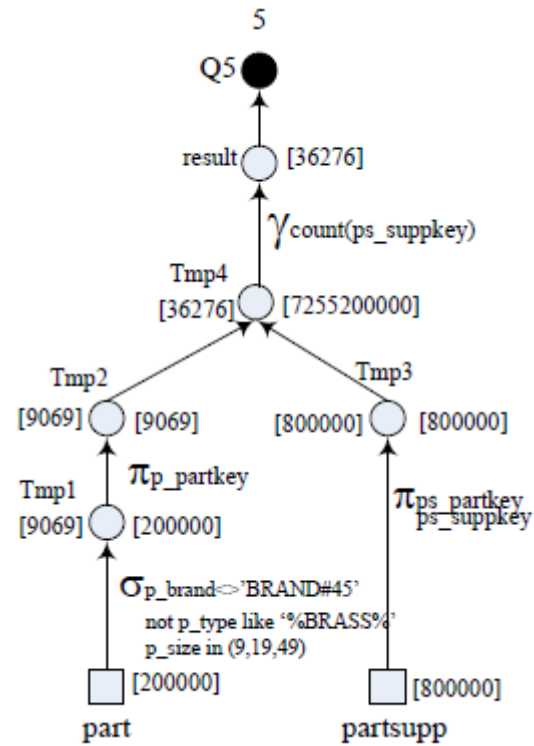


Figure (A2.4): Relational Algebra Query Tree of Query Q5

A2.5 Query Q6:

```

SELECT s_name, Sum(ps_supplycost)
FROM   supplier, partsupp, part
WHERE  s_suppkey = ps_suppkey
      AND p_partkey = ps_partkey
      AND p_brand <> 'brand#45'
      AND NOT p_type LIKE '%brass%'
      AND p_size IN ( 9, 19, 49 )
GROUP BY s_name;

```

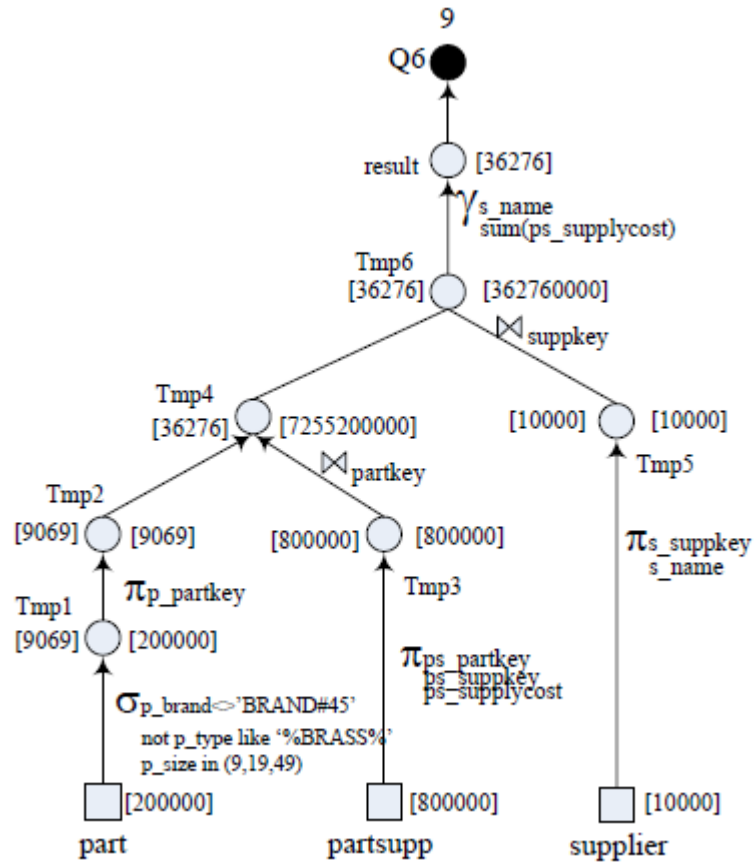


Figure (A2.5): Relational Algebra Query Tree of Query Q6

A2.6 Query Q7:

```

SELECT c_mktsegment, Sum(l_discount)
FROM   customer, orders, lineitem
WHERE  c_custkey = o_custkey
      AND o_orderkey = l_orderkey
      AND o_orderdate >= '1994-01-01'
      AND o_orderdate < '1995-01-01'
GROUP BY c_mktsegment;

```

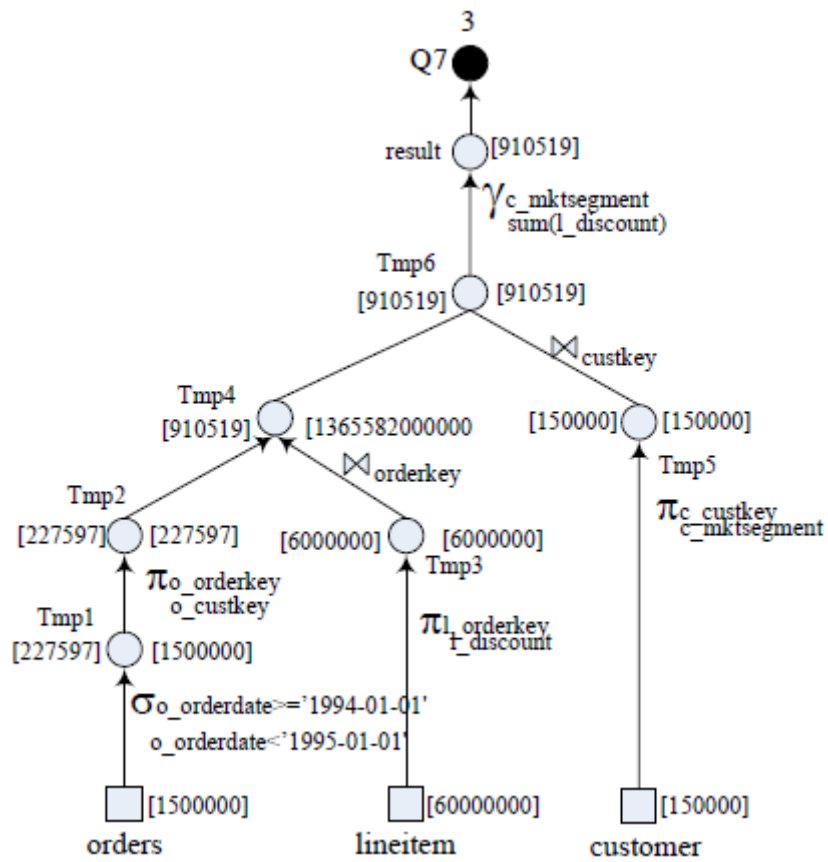


Figure (A2.6): Relational Algebra Query Tree of Query Q7