Design and Implementation of an Enterprise Data Warehouse

Edward M. Leonard
Marquette University

DESIGN AND IMPLEMENTATION OF AN ENTERPRISE DATA WAREHOUSE

By

Edward M. Leonard, B.S.

A Thesis submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

December 2011

ABSTRACT
DESIGN AND IMPLEMTATION
OF AN ENTERPRISE DATA WAREHOUSE

Edward M. Leonard, B.S.

Marquette University, 2011

The reporting and sharing of information has been synonymous with databases as long as there have been systems to host them. Now more than ever, users expect the sharing of information in an immediate, efficient, and secure manner. However, due to the sheer number of databases within the enterprise, getting the data in an effective fashion requires a coordinated effort between the existing systems. There is a very real need today to have a single location for the storage and sharing of data that users can easily utilize to make improved business decisions, rather than trying to traverse the multiple databases that exist today and can do so by using an enterprise data warehouse.

The Thesis involves a description of data warehousing techniques, design, expectations, and challenges regarding data cleansing and transforming existing data, as well as other challenges associated with extracting from transactional databases. The Thesis also includes a technical piece discussing database requirements and technologies used to create and refresh the data warehouse. The Thesis discusses how data from databases and other data warehouses could integrate. In addition, there is discussion of specific data marts within the warehouse to satisfy a specific need. Finally, there are explanations for how users will consume the data in the enterprise data warehouse, such as through reporting and other business intelligence.

This discussion also includes the topics of system architecture of how data from databases and other data warehouses from different departments could integrate. An Enterprise Data Warehouse prototype developed will show how a pair of different databases undergoes the Extract, Transform and Load (ETL) process and loaded into an actual set of star schemas then makes the reporting easier. Separately, an important piece of this thesis takes an actual example of data and compares the performance between them by running the same queries against separate databases, one transactional and one data warehouse. As the queries expand in difficulty, larger grows the gap between the actual recorded times of running that same query in the different environments.

ACKNOWLEDGMENTS

Edward M. Leonard, B.S.

I would like to thank my family for their love and support during my work on this master's thesis. I would especially like to thank my sister, Jamie Ousterout for her time and skills to help me edit this thesis.

I would like to thank Marquette University for my employment for the chance to work daily with the technologies in data warehousing and business intelligence.

Finally, I would like to thank the members of the committee for their time to be part of this thesis and their guidance during my coursework. Dr. Praveen Madiraju provided me a solid basis to continue my database studies in the form of this thesis. As director, he enthusiastically worked with me to complete this study. Dr. Harris, as my program advisor, helped to convince me to write this thesis as part of my studies for the Masters of Science in Computing program. Dr. Kaczmarek, whose seminar in business intelligence class earlier this year that sparked my interest to learn more in this discipline.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: Introduction

The vision for this thesis is to study components of a theoretical Enterprise Data Warehouse within the context of a higher education environment. The reason for such an implementation would give users at all levels of the university an integrated, secure and consistent data source from which they could report on and set their business needs more efficiently than possible without one.

Problem

The implementation of an Enterprise Data Warehouse, in this case in a higher education environment, looks to solve the problem of integrating multiple systems into one common data source. With the diverse roles that a college has both on the academic and nonacademic sides of the business, data about a person, whether they are an employee, alumnus, student, donor, or possibly all of the above, are more likely to reside in different transactional databases across the campus. Much of the information exists in a silo in and of itself, and the information does not translate across the spectrum to help the business of higher education grow. An example of this may be that the admissions department has information about the occupation of an incoming student's parent, information that may be important to the fundraising department since the parent's income from this occupation could possibly lead to a boon in donations; however, this useful information is not shared between these two departments. Certainly, having a data warehouse that shares this kind of information with the masses could cause internal strife or possible breaches of security. Therefore, devising a plan that restricts data, as appropriate, makes reasonable sense.

One universal problem of not having an Enterprise Data Warehouse is how users consume the data in the form of actual reports. So often, those who need the information or require knowledge from the data they utilize must wait for a report based on someone else's schedule. Furthermore, once they get the report, they may have to manipulate data within an application such as Microsoft Excel to fit their needs, and this can often lead to error or miscommunication. As both public and private organizations recover from the corporate fraud of the last decade, it is now more important than ever to report results correctly, which calls into question the current procedures. Rather than executives or managers having their morning coffee waiting for the delivery of reports, it should be possible for them to be able to go get their own information when they need it.

Explanation

A major component of Business Intelligence is the use of a data warehouses for making better decisions based on the information it provides. The data warehouse has a history of storing data in a way that is optimal for reporting and other analysis, such as OLAP, or Online Analytical Processing. The data warehouse uses a combination of efforts of hardware and software not only to acquire the data but also to present data for users to access it in their desired format. Once the data is in a data warehouse, there is the availability to develop them into data marts, which specialize in the arrangement of data for specific business purposes. Finally, there are hosts of reporting tools available that make the data warehouse a user-friendly experience for those who want to pull the data themselves rather than waiting for

distribution from others. The creation and evolution of the data warehouse make it an invaluable tool that makes Business Intelligence possible.

Motivation

There are many contributing factors involved when considering the implementation of an Enterprise Data Warehouse. Although executing such a project could require a significant time, resource and/or monetary investments on the part of a company, there are many motivating factors to move forward with the implementation of such a project.

The most significant motivation to implement a data warehouse is to have a better platform on which to report data. Combining the data from all the other databases in the environment, the data warehouse becomes the single source for users to obtain data. All reporting would be based on a single database, rather than on individual repositories of data. Using a data warehouse, a report writer does not need to learn multiple databases or attempt to try to join data between more than one database, a task which may prove difficult.

In addition to having one single database to report from, there are other advantages to not reporting from transactional databases. Namely, reporting from the transactional database can slow down the general database performance by taxing resources already allotted to the system. Running such a report against certain parts of a database could potentially cause slower experience for the user, because it could render a database or the application using it unresponsive. Finally, data can be in the process of an update or delete transaction in the database, which could render the report incorrect while it is running.

Another advantage to completing reports against the data warehouse is the fact that there can be a universal reporting tool and report distribution system deployed in order to allow a single platform for users. The advantages of such a tool would allow for ensuring timely distribution, consolidation, management and distribution of data as well as the ability to maintain an audit trail of the data.

Once the users have the data from the data warehouse, they can work with the data in order to make better decisions for their business. Data presented in a data warehouse is available for massaged by users in which users can work with data in Excel, Power Pivot, pivot tables based off OLAP, cubes and Key Performance Indicators (KPIs). By using warehouse data, just about anyone can create complex models, build charts and calculations, manage a variety of reporting functions, analyze and make decisions.

Besides making reporting easier for a user and having data better available for consumption, from the Information Technology (IT) department's point of view, the data warehouse will make it easier to control security. Rather than controlling security on each database for scores of users, with the data warehouse, there will be just one database to grant/deny users and rights. Furthermore, IT will have control and manageability over enterprise data that users can then access to fulfill their business needs.

Further helping control the IT department's manageability of databases, the data warehouse affords the opportunity for smaller databases to have their data represented in an Enterprise Data Warehouse. Most transactional databases have no associated data warehouse platform and there would never otherwise be a reason to have a data warehouse as part of

normal business operations for that database. Even if there was a need for the warehouse for that particular database, there might be too much time, resources or consulting fees to develop just one data warehouse, which would be helpful to only a small group of users. By having an Enterprise Data Warehouse, even a small database could be included as part of a large-scale solution.

Perhaps, one of the most underrated reasons for a data warehouse is the fact that it is able to maintain historical data. A basic transactional database does not keep historic data as well as a data warehouse does because a user normally quickly updates or deletes this data on a regular basis during the course of a day. In a basic transactional database, what a value was only hours before could disappear forever, and there would be no record of its existence. A data warehouse can accumulate old data and potentially store the data forever. As data change and rules adapt to directives, the storing of historical data may lend a chance to snapshot data within the data warehouse and, as a result, store earlier behaviors as they relate to data for current or future users to report upon.

The final reason for developing an Enterprise Data Warehouse is that a reasonable data model exists deployed for users so that consistency is the key to consuming the data across the business. Regardless of the source of data, the data model gives users a common way to get the data they need to satisfy their own purposes. This data model exists for use across the business to enhance the need for getting data in reports to make better decisions.

Now more than ever, businesses rely on accurate, secure and up-to-date information in their reports to help their business operations and comply with increasingly stringent

regulations. Having a common source of data for users and taking away the burden reporting places on a transactional database can improve the efficiency and data sharing across the business. The solution of the data warehouse replaces Excel and other reporting platforms with a modern-day, centralized reporting and analysis solution. The ability to consolidate all significant and related data for a single version of the truth is fundamental to reporting accuracy. Such an example for having more accurate reports has been since the implementation of the Sarbanes-Oxley legislation, making compliance mandatory.

Qualifications

I am presenting a real example of moving reporting from a transactional database to a data warehouse. This example compares the difference between running the same query against a view in a transactional database against a query of the same data in a denormalized, or grouped, star schema containing the same data and amount of records as the view. Using a series of timed measures, the advantages of reporting from a data warehouse will be clear. The comparisons will show the time differences for the varying queries, which would translate overall improved reporting time by users. There are advantages and disadvantages to each, but in general, it will be advantageous to report upon the data warehouse. One of the major advantages of implementing an Enterprise Data Warehouse is to improve reporting time and present data in a way so that decision makers can utilize the data to make better assessments and improve efficiency within their business unit.

The conclusion of this composition will be a working prototype of an Enterprise Data Warehouse, which incorporates simulated databases on smaller scales that describe the

implementation in a higher education environment. First, a student record transactional database built in SQL Server 2008 will come together in a specific data mart in an Enterprise Data Warehouse. Second, the "HR" database from Oracle will expand into a data mart specific to itself to show all the components of employee. This solution would provide the ability to report upon both components. For example, it would be possible to show a professor's relation to classes they teach from the student system along with their salary from the employee system. Without the Enterprise Data Warehouse, such a query would not be easy when linking between two database management systems, to say nothing of relating the actual subject of the report.

CHAPTER 2: Background

As previously discussed, the purpose of this thesis is to speak to the advantages of implementing an Enterprise Data Warehouse in a higher education environment. By doing so, we have the chance to integrate the data from the multiple databases used by various applications within departments across the campus. With all this information spread across the organization, including the possibility of a person existing on more than one system, having one single source of truth becomes a significant reason for implementing an Enterprise Data Warehouse. Otherwise, departments only report within their own silos, and this would lead to the very possible instance of not sharing information appropriately across the enterprise. Even if the information is already distributed, many users would share the material through e-mail or another method that would not be appropriate or condoned and many would fail to share this material in a reasonable and secure manner. With the data in one single source and governed securely (as deemed appropriate by the Information Technology department), data are available to the users on one platform and readily available to those who need the information.

Similar to a major corporation but also somewhat unique to the business of higher education, there are many departments that comprise the structure of a university. For each of these departments on campus, there will be some type of application used to help their own operation conduct business as needed, and this will most likely contain a database that stores all the information used by the application. The application and database could be large, small or somewhere in between but would find a place in the Enterprise Data Warehouse to benefit the needs of the reporting services within the organization. As an added layer of complexity,

the current database may also have its own data warehouse, but contain only independent information from that specific database and not that from the enterprise. Despite this revelation, the Enterprise Data Warehouse would integrate the specific data warehouse data rather than pulling directly from the transactional database. This was exactly the solution at the University of California Berkeley, where they integrated an Enterprise Data Warehouse that, "…would integrate data from the broad spectrum of campus information systems and warehouses so that the data could be used to meet a wide variety of needs (Warehouse Initiative, 3)."

When examining the needs for the Enterprise Data Warehouse, it is important to take a full view of the departments on campus and evaluate what they could possibly contribute to this project. In any higher education environment, the student system would be the cornerstone of the database reporting, and this system would contain all current and past student information about classes and the student themselves. Next, the financial system would track the monetary flows showing the cost of doing business within the university, including endowment and paying professors and administrators for completing their work. To take this group further, a practical human resources system would help employees navigate their recurring employment tasks. Another group that would have a robust database and need for reporting would be the alumni system for fundraising or engagement. There would also be other smaller departments such as a facilities department, police department and library, to name a few. All of these individual databases in an Enterprise Data Warehouse "would provide an umbrella structure for linking and expanding information from existing systems to support reporting and analysis (Warehouse Initiative, 6)."

The following diagram in Figure 1 attempts to layout the schematic of the possible Enterprise Data Warehouse.

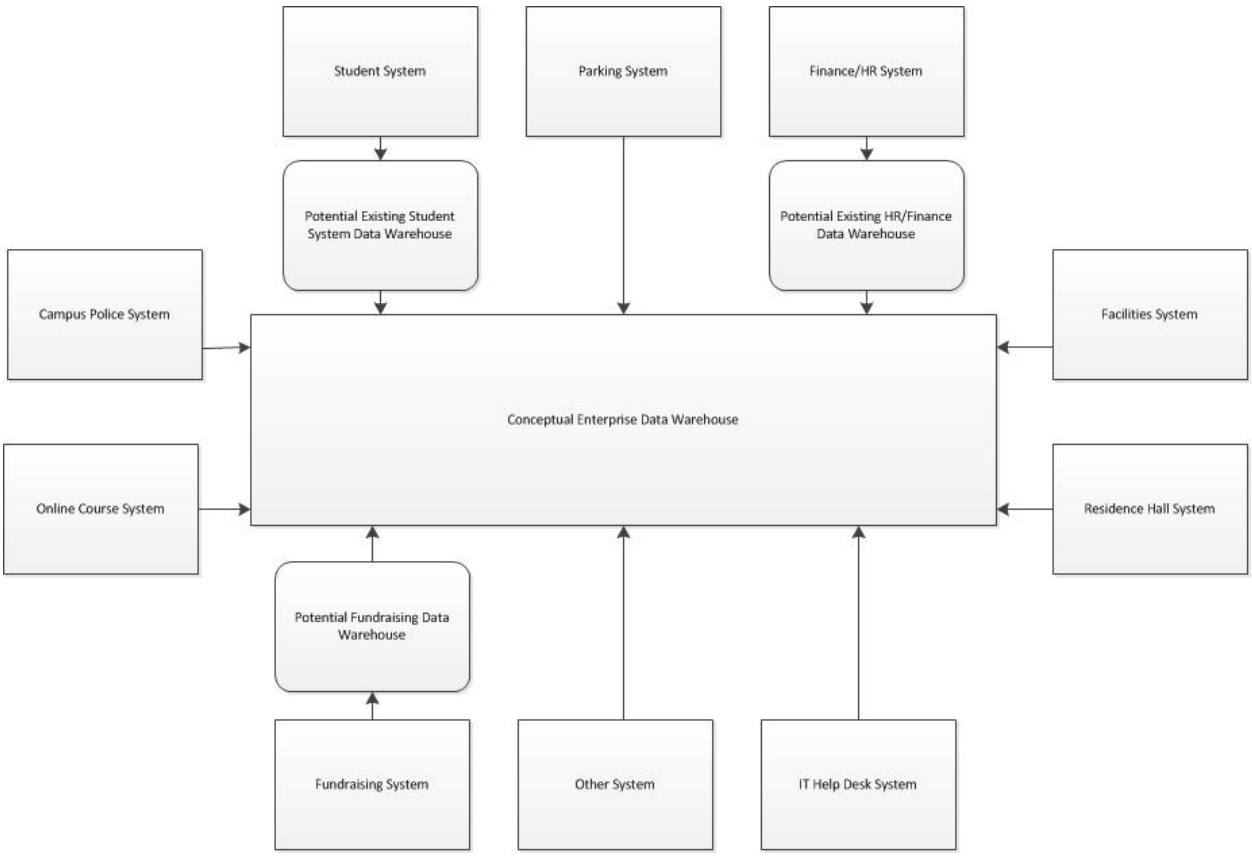


Figure 1: Conceptual Enterprise Data Warehouse.

With the creation of an Enterprise Data Warehouse, there might be a number of questions about the ultimate benefit of having the collection of data in one place, especially if the university is functioning just fine without one. The idea of the Enterprise Data Warehouse would be to offer reporting to deliver a full view of a person as he relates to the university, whether he is a student, alumnus or employee. The Enterprise Data Warehouse would have information about the person that would give users the knowledge to better execute their needs. An example would be the fundraising department wanting to learn more about an

alumnus they would be interested in reaching out to in order to submit a proposal for a donation. The Enterprise Data Warehouse would provide a place for such a query to review before talking to this alumnus so that the fundraisers could make a better decision about how they might obtain their donation. Such a query would be able to provide the information about what courses the student took as an undergraduate, what jobs he held while in school and maybe even any parking tickets they may have received during their time on campus. Additionally, they would also want to verify the alumnus' propensity to give based on prior gifts from the fundraisers' own database, notwithstanding the other information from the other systems.

Besides the fundraising proposal possibility, there is another usage of the fundraising database information and the student system. The advantage of having these databases denormalized into a data warehouse for reporting upon is that there would be the ability to share information better than can be done now. This situation entails how the fundraising department could solicit parents of students for gifts to the university. Although it sounds like an absurd idea to ask for donations from parents who already may be paying for their children to attend college, there remains a good possibility for successful fundraising. The goal is to import parents into the fundraising database from the student system, which sounds easy on paper but has a level of complexity that an Enterprise Data Warehouse can solve. Imagine that a new student lists his parent's name and address on a college application, and this information is subsequently entered into the student system. When the fundraising department wishes to enter them into their system, there is a possibility of a couple immediate problems. The initial questions are: First, does the parent already exist in the fundraising system because another

child already attends the university? Next, is the parent already in the system as an alumnus or other donor? Finally, is the parent already in the fundraising system under a different address? Because the systems do not talk to each other right now, there is a possibility of a potentially generous parent not going through the data import because of some technicality or human error during the final import. The solution is to have an Enterprise Data Warehouse so that the data, once entered, come over to a common platform where the parent data, logically arranged in a data mart, exist for use on a recurring basis.

Having data from multiple databases in one report, leads to better information sharing and reduces the possibility of performance implications. Imagine that an audience wants to view data from the student employment system, which must link to the student system for the purpose of the report. In this instance, it may be true that having a report based on the student employment system may run very quickly and return data in a desired fashion, but as soon as the link comes into the student system, progress slows significantly. The solution once again is to have the data arranged in such a way that the pressure is taken off the transactional database and placed onto an Enterprise Data Warehouse. While the data is only as current as the last ETL refresh, having the data on an Enterprise Data Warehouse saves both time for the report writer and others using the system. Further, the information stored in the student system exists as it does because it has to be in that database in such a way for the purpose of the application whereas the data warehouse would have this data arranged for the purpose of this particular report rather than the application itself.

There is really no greater justification of having an Enterprise Data Warehouse than being able to consume and report data better and faster than the way in which users are used to doing it now. With the reporting tools available now, as there are many to choose from in the marketplace, one might believe that things could not improve much better than they are now. However, as we further explore the implementation of the data warehouse, we will find that there are better choices available for reporting. Further, how we analyze data will significantly improve with the data warehouse infrastructure in place. This is justified by exploring term and tools such as dash boarding and the online analytical possibilities of "OLAP," to say nothing of the automated and secure delivery of data to users that gives them a chance to pull rather than be pushed to, as is necessary these days.

With the many great possibilities that an Enterprise Data Warehouse offers a business, it is important to remember that undertaking such an endeavor requires a determined group of people within the company that desire to create the system. There needs to be a strong executive sponsor that can spearhead the effort with leadership, a solid business reason that provides the motivation, and finally an organized project, which needs to have management led by the standards of the Project Management Institute. Each piece of this coordination to develop the data warehouse is possibly more challenging than the physical development of the warehouse itself, as providing the motivation and financial support is particularly demanding. In these trying financial times where budgets are not only limited but also much scrutinized by company leadership, it would be up to the executive sponsor and business itself to champion such a large undertaking. It would then be up to the IT department to carry out the project to create the warehouse, as the deployment is critical.

One of the most important parts of implementing an Enterprise Data Warehouse, is that there must be a strong executive sponsor to champion the effort of introducing the warehouse. Executive sponsors are defined as: "Strong business sponsors [that] are influential leaders within the organization. (Kimball, Lifecycle, 16)." Besides having a strong interest or desire in the data warehouse, they are probably strong members of the business community within the company and well respected enough to build a following for this undertaking. Also important is the virtue of leadership, which is vital to direct others to go through the various steps of development of this warehouse. Furthermore, "Effective sponsors are able to deal with the short-term problems and setbacks because they are focused on the long-term success of the project (Kimball, Lifecycle, 17)." Not all parts of the implementation may be easy or succinct, but having the follow-through to work past any issues and continue to drive users to the goal of a better data and reporting environment is a quality of a good executive sponsor. It is also reasonable for the sponsor to be part of a small group to head the effort rather than just one single person; however, the project can still be successful with just one individual.

The group behind the leadership of a resilient executive sponsor is the main benefactor of an Enterprise Data Warehouse, which is the business unit itself who will consume all of the benefits from the successful deployment. It is said that, "Organizations that successfully implement DW/BI solutions typically have a powerful sense of urgency for improved access to information caused by one or more compelling business motivations (Kimball, Lifecycle, 17)." The business may be used to or able to be successful with their current methods of information and using data, but there is always room for improvement and wanting to save time and obtain faster results will be possible with a data warehouse implementation. The desire for such

improvement usually comes in the form of a sales pitch for how implementation of a warehouse can improve the way a business currently does its work by showing flashy dashboards and promising new data in greater speed, hence creating a buzz. Armed with this, "DW/BI systems that align with a strategic business motivations and initiatives stand a good chance of succeeding." (Kimball, Lifecycle, 17) This reasons to believe that if, for example, an improved reporting process would better satisfy a need of a business unit, having a data warehouse would become a priority for development.

While having the strong backing of the executive sponsor and business to create the Enterprise Data Warehouse in the first place, it is up to the Information Technology project management office to lead the effort to deliver the data warehouse solution. While the business unit or executive sponsor will envision a complete working solution in a relatively short duration of time, the IT department will be required to handle the staffing and seeing the project through to completion. While the sponsor and business unit may propose a strategy that is grand, the execution is crucial, and the project manager will need to lead both staff members within the IT department plus those of the business units as well as report progress and issues to the "Project Management" group, composed of the executive sponsor along with company leadership.

The project team members are the group that will work on this project in addition to their current occupations or perhaps even on a full-time basis to see this undertaking through to completion. The project team would be comprised of those who have a sincere understanding of how to implement such a project or may exist on a contracting basis to help

complete the task of building the data warehouse. The fact that "It is common for the DW/BI team to take on both development and operational duties given the highly iterative nature of the DW/BI project development cycle (Kimball, Lifecycle, 35)," shows that the project is composed of mostly technical but also supporting players who help analyze and educate users. They find themselves sequestered to a project room to work together so that they can share their knowledge through open discussions or organized meetings. Furthermore, there could be changes or issues that arise where the speedy resolution relies on these problem solvers being able to join together in short time.

Once the project kickoff meeting, composed of all the impacted users and leadership members is complete, the formal execution of the project itself starts with a project plan and requirements gathering. The requirements for a data warehouse may seem clear-cut as explained thus far, but at this stage, there is room for input from everybody involved in the project. This input can be gathered through a series of interviews to find everyone's true desires for the project. Ralph Kimball states that in this stage, it is not necessary just to find out what people want out of the data warehouse, but more so find out more about them as it relates to their job. He says that, "You need to talk to them about their jobs, their objectives, and their challenges (Kimball, Lifecycle, 65)." This makes sense, as the data warehouse would better benefit the community of users if it could make jobs easier and provide a solution to some of the things that are difficult about their jobs. Upon completion of this series of steps along with other requirements of information gathering, a regular project plan comes to fruition, setting the scope of how the project will finish within stages needed to be to a success. Because "coordination of sponsorship, stewardship and best practices at the program level is

crucial to maximize the potential business return from your DW/BI investment (Kimball, Lifecycle, 58),” it is therefore important to keep on task and provide real solutions to existing procedures.

Since the project at a high level entails moving data from one source to another, there may be an important step needed before moving too deeply into this endeavor, and this may perhaps be carried out as a separate project at a department level, rather than as part of the enterprise project to implement the data warehouse. Before implementing a data warehouse, there comes into question the state of the data within the transactional databases in the first place and whether some separate work is necessary in order to cleanse the data before moving it out to a data warehouse environment. One memorable comparison always comes back to the description of being “garbage,” whether it is referring to garbage in and garbage out or leaving the garbage by the curb when referring to databases. Regardless of the evaluation, there remains the not well-kept secret that every database can contain components of data that users do not like but have learned to live with, and this needs evaluation before moving to a data warehouse environment.

There are many procedures and steps necessary to follow when moving data to a data warehouse, which usually are easier to do in the early stages of development. An article on data quality (Scott) in “End-To-End Data Quality,” discusses the task or choices involved with ensuring quality when upgrading to a BI system in which operational data is about to be loaded. This article evaluates not only data that might come over to the BI system but also the data in the current operational system. The main overlaying idea is to have quality data when moving

to the new BI system, in most likelihood a data warehouse system. The author faults all the operational systems that fail to have any standards built into data entry screens, which could potentially save the entry of bad data (Scott, 1). Furthermore, the author asks: 1) How good is the data? 2) What sort of errors are present? 3) How important is it to be correct in the BI system? And, finally, 4) How important is it to be correct on the operational source (Scott, 1)? It may not be possible to clean up all the problems of the operational database and there will certainly be projects to address this once the data moves over to the data warehouse, but there is a significant reason to attempt to complete this in the earliest stages.

Besides fixing the perceived data defects that exist or are unknown in the current transactional system, changing the behavior or guidelines before the data enters the system is also an effective cleanup undertaking. In another article (McKnight) in "Data Quality for the Next Decade," the author describes the data issues as defects that will lead to bad quality service to customers and problems internal to the business. He breaks down the defects into 11 different sections, including: 1) A lack of integrity of reference between data values across the model, 2) Inconsistent formatting, and 3) Unmet requirements for field values based on other values and miscalculations, to name a few. The interesting conclusion is that the author believes that rather than costly and difficult cleanup projects to cleanse the data, better diligence is necessary on fixing these issues before data enters the database in the first place. He concludes that organizations should protect, "information assets, data modeling, data quality, data architecture and data entry." (McKnight, 1) This translates to the idea that there should be some standardization or potentially an application change to ensure better data in the first place.

Concerning the need for standardization, a real world example exists within the higher education environment in no matter what school one would evaluate the student system. Continually up for debate within the department of the registrar or academic side of a university, is the question of how to best define a new freshman within the school: Is the new student considered an official freshman upon acceptance into the university, upon enrolling in classes, or upon completion of the first semester? Depending on the definition, or lack thereof, the bigger problem is trying to track this within the student system transactional database and eventually the data warehouse where other users may want to consume the data within their report. Having a plan or set of standards before entering the data instead of cleaning it up later would be the better solution based off the previous article by McKnight where it is better, in this case, to change the human behavior instead of that of the system. Any further cleansing or standardization would still be necessary during the ETL development stage of the project, but this is due a reasonable attempt before the data meets the system.

Before even beginning the process to construct an Enterprise Data Warehouse, there are many important pieces to this project that provide the background to the data warehouse itself and will ultimately lead to the successful implementation of this project. The most important piece is the end result of the data warehouse that shows how the entire collection of departments within a business, or in this case a higher education environment, come together to give users the opportunity to report on the big picture rather than on an individual silo of data. Before the project can get off the ground, it is necessary not only to have strong buy-in from the business but also a strong leader to champion the data warehouse and share the project vision to the teams of users who will eventually use the data warehouse. Almost as

important to the vision is to complement the effort with a strong project management office to carry out the execution of the project. Users of the system or those on the project team will be able to provide input and share how they want to solve some of their current challenges. They will also undertake the task to clean up or standardize their current data situation as it pertains to them. The results of having the data warehouse must justify the investment of time and monetary expense, and a very clear understanding of the background is necessary to understand all implications of the project before implementation.

CHAPTER 3: Business Intelligence

The motivating application here is for users to have access to the Enterprise Data Warehouse itself, which contains the data from other databases from across the entire business. Although most databases that are in use in an environment will have an application sitting on top of it for better usability, a data warehouse is just a very large database that does not have a traditional application tier. Nonetheless, there are applications that are part of a data warehouse, but they are there to complement the warehouse as components that provide a different purpose. First, there is an ETL (Extract, Transform and Load) application that is part of a solution that moves data from the transaction source database. Some popular ETL applications are SQL Server Integration Services, the OBIEE application from Oracle and Informatica, to name only a few. The most recognized application piece of a data warehouse is in actuality the reporting tools used as part of the presentation layer. The reporting application is generally not part of the warehouse, but it is the public face of the data warehouse and is the part that most users are familiar. There are host of tools to report or query the data in the warehouse, but they are generally homogeneous to the warehouse and can be used with other databases for reporting or data presentation. Nonetheless, quite a few other components are for use specifically for the data warehouse, such as the tools that a vendor will be eager to sell you for some long-term use and license fee. However, for the point of attaching a motivating application to a data warehouse, the most important piece is the application that handles the presentation layer, which is the information-sharing component, specific to reporting.

Reporting

Now more than ever, businesses rely on accurate, secure and up to date information in their reports to help their business operations and comply with increasingly stringent regulations. The issue is that there can be data in many different locations, and those outside of IT can hastily assemble reports without regard for the accuracy of the data. Such a case can cause questions about the reliability of the data in the report and can lead to bad decisions and possibly even larger issues down the road. In the context of having an Enterprise Data Warehouse, "BI tools ensure timely dissemination of data for reporting and analysis across the enterprise. In addition, the tools that consolidate, manage and distribute data also maintain an audit trail of the data – where it came from, what happened to it, and who receives it (Solving, 1)." An effective business intelligence solution must ensure timely distribution of data, allow for reporting and analysis across the enterprise, provide easy-to-use tools that consolidate, manage and distribute data and most importantly, maintain an audit trail of the data.

The culmination of the data warehouse is in how effectively the business intelligence solution interfaces with the data. The functionalities deal with the presentation layer of the data warehouse. There is the question of the tool's connection capabilities should you look to more than one environment like SQL or Oracle. Also questioned is the scheduling and distribution for sharing data on a recurring basis as planned. The security involving user authentication and governance is also important. Is it possible for customization, which would allow changes in the future? Finally, can you export of data and is there Microsoft Office compatibility with exported data? The most popular tools for the purpose of reporting are SAP

Business Objects, MicroStrategy, IBM Cognos, Actuate, Jaspersoft and Pentaho. There are questions about buy vs. build, functionality and, finally, the current popular tools available. Reasons to consider when buying or building reports include: the total number of reports, desired distribution and ad-hoc capabilities. With superior tools available, the question businesses face during implementation is still "whether to buy or build a reporting tool for your business intelligence needs" (Reporting, 1). In addition to basic reporting, it is more interesting to explore the other possibilities that exist within the realm of data warehousing, such as data mining for analysis, and dashboards.

When deciding which business intelligence tool to use, because of the significant investment and concern about usability, management must determine the best type of product to implement. The "Hype Cycle for Business Intelligence, 2010" model helps organizations understand the available technologies and their maturity and markets. The model helps businesses decide which technology innovations to adopt, postpone or ignore, as well as to figure out the appropriate time to adopt. Based off this model, businesses can make well-based decisions on the type of BI tool to implement. The higher the expectations and the less time involved cause the chart to rise, until realities set in to lower it. The model levels off somewhat then rises based on more traditional and standard BI products that have been implemented. The second part of the article explains the Priority matrix, which shows the BI technologies based on the level of benefit against the years to mainstream adoption. For example, dashboards have a high benefit with a very low number of years of adoption.

In the "Magic Quadrant for Business Intelligence Platforms" by Gartner Research, the quadrant shows where the various BI tools land in relation to others when analyzing the impact

of the BI tool. The four quadrants include niche players, challengers, leaders and visionaries. The vertical axis shows the ability to execute whereas the horizontal axis shows the completeness of vision. The leaders and niche players have most of the groups, as the leaders are easily the most recognizable while the niche players are part of a certain business need. The article spends time looking at the market trends for what users are looking for in their BI tools and where preferences are these days. They walk through each product including the popular tools of large companies, such as SAP, Microsoft and Oracle and discuss the products offered with the strengths and concerns associated with each tool.

Despite the clean and eye-friendly applications that are synonymous with business intelligence as they relate to the data warehouse, the first tool available for use on the presentation layer is the basic query tool. The query tool "allow[s] users to query the dimensional model directly and define a result set. (Kimball, Lifecycle, 479)." Not only would a business intelligence developer use this tool to compile enhancement programming, troubleshoot issues and complete data profiling, they would also use it to develop queries to create enhancements or develop new reports or data mining solutions. In the past, some of the older reporting methods used basic or advanced SQL to query a database and deliver the results in a comma-separated value file or into a spreadsheet application. While that would still be possible in the data warehouse solution, it is a better practice for the Information Technology department to regulate the data and limit the opportunity for basic users to query the database openly and restrict access to developers or programmers who need it for a specific purpose.

The focus of this section is: “The reason for generating reports is to transform data into information for the business user” (Whitehorn, 31). The most popular tool is still likely to be the ad-hoc reporting that most users are very familiar with because they would have relied on this method for years and those doing other reporting for the past decade or more are still very likely to use this same reporting with the data warehouse. Some of the most popular reporting tools for users remain the SAP Crystal-reporting tool and the “Report Builder” tool from Microsoft. The Microsoft tool is, “An additional tool, Report Builder, aimed squarely at the power user, was enhanced in SQL Server 2008” (Whitehorn, 30). Much like in the way Crystal reports are run through the Business Objects Enterprise application, Microsoft reports are deployed to SQL Server Reporting Services, which acts in the same capacity as a web tool for distributing reports to users. These report repositories give the IT department the ability to govern access to reports contained within the data warehouse, and they also give the users the ability to pull data rather than having to push the report to them as in the past.

One of the most popular reporting trends today that takes advantage of a data warehouse implementation is the use of dashboards, which provide the chance for user interaction when getting information. The dashboard “provide[s] the interface tools that support status reporting and alerts across multiple data sources at a high level, and also allow drilldown to more detailed data” (Kimball, Lifecycle, 491). A typical dashboard will usually include datasets, graphs, spark lines, key performance indicators and summary data. One of the advantages to having a dashboard is that in many cases, it will be interactive to the user and be able to show projections to have users make better decisions or zoom into a troubled area. A dashboard takes aim at a small group or targeted audience, as they are very detail-

oriented to a specific area of focus. In describing their transition from an old reporting system to dashboards in conjunction with their Enterprise Data Warehouse, Florida State University commented: “It was a very fundamental change. Prior to this, we delivered most of our enterprise resource planning type reporting in spreadsheet format. And this completely transitioned it to an online system where people could get access to the data they needed and to queries to get it in a different format if they really wanted it” (Schaffhauser, 3).

Another popular use of business intelligence in a data warehouse environment is having the ability to use advanced OLAP (Online Analytical Processing) and data mining. OLAP can organize data into multi-dimensional structures to allow users to choose dimensions to view. Data mining would “explore the data in a warehouse for useful patterns and relationships” (Kimball, Lifecycle, 479). A popular tool for this is Microsoft’s SQL server Analysis Services, which is a powerful analytical tool included in the suite of business intelligence products. The tool gives users the ability to design cubes based off a data warehouse’s star schema, then adds measures and key performance indicators, known as KPIs. The new Power Pivot tool used in Excel 2010 puts this power into the user’s hands in an easier fashion, and this “means no aggregates are required to get this speed, simplifying development and management, and giving predictable fast performance” (Whitehorn, 31). Giving a user access to these types of analytics further shows the ability for easier analysis of the sheer data that will be available in the data warehouse.

All the examples of reporting give users the ability to consolidate all data within the data warehouse to create a single version of truth, which is fundamental to reporting accuracy. There are many tools available for use in a business intelligence system and there are resources

to find out which one works best for the enterprise compiled for users to decipher. Because many of the DW/BI tools are similar in nature and scope, what is important most to users is that they fit the business well and are ultimately user-centric in deployment, especially as users are more involved in consuming business intelligence than ever before. In general, Kimball concludes that business intelligence tools must be usable, content rich, clean, current, and interactive and value oriented (Kimball, Lifecycle, 500). By having a data warehouse in place, the enterprise can take advantage of the newest technologies available in this field and empower the users to make better decisions with the information available to them.

CHAPTER 4: System Architecture

The Enterprise Data Warehouse has a design scheme that makes it conform to the current and studied methodologies of data warehousing where the result is a star schema design. The traditional data warehouse design is mostly due to the work of Ralph Kimball of the Kimball Group and will be a source of concentration for the following discussion as it relates to the system architecture. The most poignant of terms introduced by Kimball is that of dimensions and fact type tables as the way that data need to be arranged for the eventual organization into data marts and reporting. Also provided is the manner in which data moves from the operational source, such as transactional databases, over to data warehouse itself. The data itself moves through a tool known as an "ETL" which stands for Extract, Transform and Load (which itself stands for the steps taken with the data to build the data warehouse). Once the data are loaded sufficiently, the data in the appropriate tables give the users the ability to report their data as needed.

Data Warehouse High Level Architecture

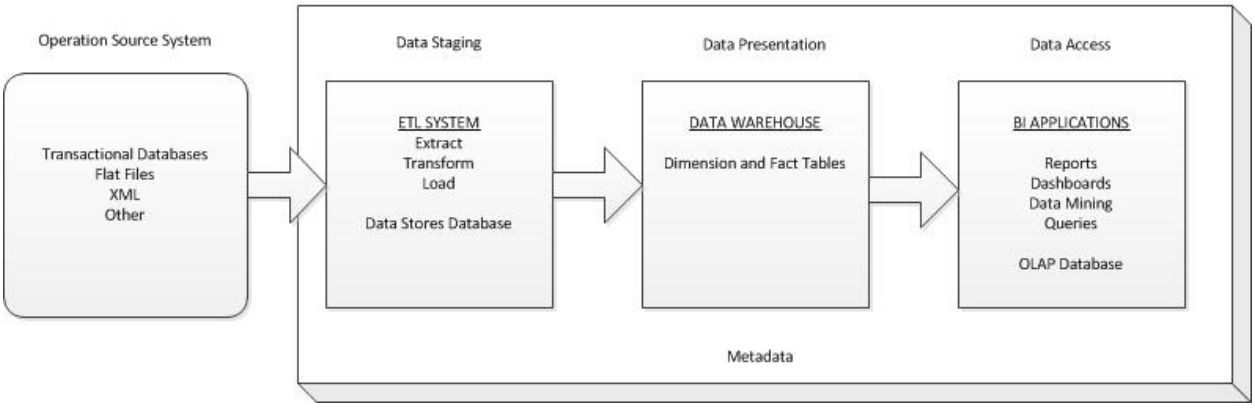


Figure 2: The High-level Architecture of a Data Warehouse.

Data Warehouse Layers

The initial construction and recurring updates of data have evolved over time to offer an efficient way to load data. As shown in Figure 2, within the scope of a data warehouse, "there are four separate and distinct components to be considered as we explore the data warehouse environment—operational source systems, data staging area, data presentation area, and data access tools" (Kimball, Toolkit, 7). The operational source is the transactional production databases that have typically been the location for all of our data. Next, the data staging area is a database off limits to users, but it is the first stop for the data from the transactional system where data comes together through the ETL and meets its transformations before it continues to the actual data warehouse. The data presentation area is the actual data warehouse full of data that is in dimension and fact tables, the standard formatting method of data warehouses. Finally, it is vital that the presentation layer exists to provide users with an experience that gives them the data in a format they need to make decisions effectively.

The physical architecture of the data warehouse begins at the data-staging layer of the data warehouse model. Generally, the ETL, which is the tool that extracts, transforms and loads the data from the data sources, is a visual application that uses various graphic user interfaces tools to transport the data from source to destination within the warehouse. For most of the ETL applications, it would need to sit upon a GUI operating system such as Windows or Linux while some could run on a UNIX-based operating system. When looking at the database that would act as the data store, it would make the most sense to locate this

database on the same general area as the data warehouse itself so that data would not need to travel so far or across servers. Depending on the database used, this could be anything from a Windows Server to a Linux server or HP-UNIX. Because these databases could be large, as they may quickly copy much of the data out of the transactional databases, this must be considered along with the space needed for the main data warehouse.

The most important hardware consideration for this project relies upon the decisions that will support perhaps the largest database that will exist within the entire company. The hardware from memory, CPUs and especially storage need to be at the top of the list of priorities when purchasing hardware for deployment. Businesses should consider not only what the database needs today for the project installation, but also what future improvements to the warehouse database will be required. Kimball states: "The major relational database vendors are still the platforms of choice, covering about 80 percent of the DW/BI market" (Kimball, Lifecycle, 165). Although it is probably obvious which database systems he refers to, it is imperative to make sure that these systems can handle the potential size and storage needs that will be necessary, perhaps into the terabyte range coupled with reasonable disk partitioning. When thinking about storage space, businesses should also give, consider the backup and recovery plan for that warehouse, especially since creating backup files could be very large in size.

Finally, the data access layer will require an entirely different environment. Although the same overlapping technologies are possible, one should not house these components in the same location because systems and the data warehouse layer need to exist separately from one another. In the case of the data access layer, there would once again be more of a need for an

application server. However, if using an OLAP engine, it may be necessary to have storage space for a related data-mining database for cubes and other related parts to this service. For the data layer, "There are servers to handle data access from the web, query management, enterprise reporting, authentication, metadata databases, and more" (Kimball, Lifecycle, 167). Regardless, most of the need at this layer refers to application and web parts that give users clear and clean access to their data by supporting the tool. Choices can also exist for the platform, such as the SAP Business Objects application, which can easily run on UNIX or Windows. The enterprise must evaluate what best fits the business's situation.

ETL Load and Process

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. Although it is possible to build an ETL that would complete its task as the go-between a transactional database and the eventual data warehouse "star schema," it is logical to extract the data and store it temporarily before transforming. Well known as a "Data Staging" area or Operational Data Store, this area of the data warehouse exists only to pull data out of the operational system, then rearrange and cleanse (also known as transforming) data before being loaded into an organized data warehouse. See Figure 1, which shows where one would find the staging area in a typical data-warehousing environment. A data warehouse, and especially one that is for an enterprise, has data coming from multiple sources, and this could make an ETL run for much longer than would be feasible to have a data warehouse in the first place. Therefore, quickly pulling all source data makes sense at this scale. Once the data comes out of the

transactional databases, having all the data away from business operations and in a secluded location gives the ETL a chance to stage and transform at its convenience. Once the data go through the transformations in a reasonable way, the data would next be going through the easy step of the ETL where it loads the dimension and fact tables of the data warehouse.

There is probably no better analogy of this part of a data warehouse then the Kimball definition of the data staging area of an ETL process. He says that it compares to the kitchen of a restaurant where raw food becomes a fine meal that customers will enjoy. As Kimball sees the comparison, "In the data warehouse, raw operational data is transformed into a warehouse deliverable fit for user query and consumption (Kimball, Toolkit, 8)." Taking it one-step further, the steps where the food becomes a meal, a trained chef will complete this task just as a trained business intelligence developer will design and build an ETL to do the same with the data that come in. Furthermore, whereas a customer out to dinner will likely never see the actual kitchen where the meal comes together, the average end user will never know the database or dataset that contains the data.

One of the main reasons that a data warehousing staging area is necessary is to begin the process of working with the large amount of data that creates the final organized warehouse product. In the simplest definition, the reason that a data staging is necessary is, "Due to varying business cycles, data processing cycles, hardware and network resource limitations and geographical factors, it is not feasible to extract all the data from all Operational databases at exactly the same time (What is a Data Mart, 39)." Since data will be coming from multiple data sources of varying sizes and in various types of forms, there needs to be a place for this data in large-scale data moving operations. One example would be that in the realm of

an Enterprise Data Warehouse in a higher educational environment, data coming from the finance or student system may need to be extracted more often than perhaps a smaller database that deals with the parking system. In addition, having the data stored outside of the transactional system will greatly reduce the taxing work of an ETL that would be transforming as it extracts, rather than having the data away from the users who will otherwise be annoyed from a high-power ETL.

Another advantage of having the operational data store is the chance to find out more about the data that will eventually be loaded in the data warehouse to determine the quality of the data that users will want to report in their new business intelligence portal. The ability to participate in data profiling gives data warehousing technicians a set of standards to follow and test the data as data come from the transactional databases. "Data profiling is the surveying of the source data landscape to gain an understanding of the condition of the data sources (Rogers, 1)," and gives the IT staff the chance to analyze data as it starts its journey to the data warehouse. One advantage of this step is that as data begin the extraction stage, if counts are abnormally large or small, some action becomes possible on the part of staff to determine if there is an error in the ETL itself or rather if there is a need to change the ETL for it to act more efficiently going forward. There can also be further statistical analysis completed to show the advantages of having a data warehouse in the enterprise, as management may be interested in seeing summary numbers to show their investment paying off.

With the data safely extracted from the transactional database, perhaps the most advantageous part of the data staging area is the chance to transform the data so that reporting within the data warehouse can be strengthened. While the data in the operational

databases is most likely efficient and undoubtedly correct for the application it supports, having the chance to transform data for the data warehouse is a crucial step and a chief advantage of a data warehouse's daily operations. Besides the data that is in place already from the transactional databases, there is the ability to re-stage the data in the forms of temporary tables as well as other ways to transform the raw data before the data go into a data warehouse. Another option is to tackle long-tabooed subjects of data standardizations that would otherwise not be possible with the front door application system, under which users already have developed bad habits. Thus, "Data standardization and matching is a set of processes that primarily consists of the design and execution, standardizing jobs to create uniformity around specific mandatory data elements" (Rogers, 1). Not only could old habits be fixed, having a standardization could conform with newer regulations such as Sarbanes Oxley, or in the case of higher education, HIPPA.

There are exciting possibilities when it comes to transforming data, which in the past would not be possible or allow for any updates. Kimball says that, "there are numerous potential transformations, such as cleansing the data (correcting misspellings, resolving domain conflicts, dealing with missing elements, or parsing into standard formats), combining data from multiple sources, DE duplicating data, and assigning warehouse keys (Kimball, Toolkit, 8)." Having data in multiple sources that come together for one eventual data destination almost requires intervention by the ETL to make the data conform to standards. To the point of misspellings, there is a grand opportunity to do what users will still not do, as they should sometimes. With other available transformations, such as in the SQL Server Integration Services ETL tool, there exists the opportunity to add columns based on a derivation of data

along with another popular features, known as a “Fuzzy Lookup.” The Fuzzy Lookup “joins a data flow input to reference table based on column similarity; the Similarity Threshold setting determines the closeness of allowed matches” (Veerman, 94). This lookup will take the information provided in the ETL, and based on a predetermined acceptable percentage of likelihood of a match, give the user the chance to make a very educated decision as to what to report or leave out.

The data staging area is one of the most significant pieces of the data warehouse solution because it offers chances to organize data from all sources into one place before sending the data to the final source where the users will make their queries run. As Kimball analogizes with the kitchen of a restaurant, the data staging area will not be transparent to the users of most every spectrum of the community aside from highly specialized developers responsible for the ETL. Nonetheless, the extraction of data would be better organized rather than haphazardly assembled or cached in an ETL and would reside in tables similar to the source of the data. Although not required, it appears to be a prudent decision to stage data before loading the data warehouse, especially with a larger system or scope of the data warehouse. While having the data in a staged location, it gives the ETL the chance to complete the middle step of transforming the data and fixing any abnormalities along the way. Once complete, the loading of data into dimensions and facts that are the usual makeup of the data warehouse is then the next rational step.

Metadata

A significant piece of the data warehouse process as it pertains to warehouse construction refers to the metadata, or information about the data that would be loaded. An article by T.T. Wai and S.S. Aung describes the use of metadata as part of a data warehouse and even further, it defines the many parts of metadata, which is not always clear. The article defines Metadata as follows: "Metadata is not the actual data; but rather information that addresses a number of data characteristics such as names and definitions of data elements, where the data comes from, how it is collected, and what transformations it goes through before being stored in the data warehouse(Wai and Aung, 671)." The metadata might refer to the field length and type rather than being the data and the authors further explain that, "meaningful metadata identifies important relations among data elements that are critical to using and interpreting the data available through the end user query tools (Wai and Aung, 671)." Metadata refers to the different data warehouse layers, beginning with data loading and transformation where it describes the source data and any changes. Next, it associates with data management, associating with the database management system information containing tables, views and indexes. Finally, metadata normally generates queries, such as histories and profiles.

Sometimes there can be unique uses of metadata with an ETL process of moving data from data from the source to destination. An article by Jian Li and Bihua Xu discusses a slightly different architecture setup for an ETL used to develop an oil drilling data warehouse. A typical

ETL design includes data extraction, data transform and data load to a data warehouse. The authors discuss developing an ETL to do the same tasks, but their idea is based on the metadata of the data itself, which uses data buffers during the ETL process to store the data during processing. One of the advantages is that within the ETL process, "Data extract, and transform and load can be executed individually and parallel. That accelerates the efficiency of ETL (Li and Xu, 2568)." The ETL process divides into different layers, with the first and third layer as the source and destination, respectively. The middle layer contains five models that process the ETL. In summary, this shows an ETL that has different architecture to improve efficiency and use more of the tools available to complete the ETL process.

Dimensional Modeling

The major process of organizing and configuring the data is the phase known as "Dimensional Modeling," which is the step that configures and organizes the final destination for the data. Kimball states: "Dimensional modeling is widely accepted as the preferred approach for DW/BI presentation. Practitioners and pundits alike have recognized that the data presented to the business intelligence tools must be grounded in simplicity to stand any chance of success" (Kimball, Lifecycle, 233). This affirms that it is most logical and widely practiced to transform the data by DE normalizing it into the dimensional model composed of dimension and fact tables. Although it is very possible to use an ETL to move data to a data warehouse and make it presentable in the same form as the source, this does not make much sense and fails to take advantage of the benefits of accepted data warehouse architecture. Furthermore, if one is going to go through the effort of building a data warehouse in the first

place, it reasons that the end users will want to have their data available in the most easily relatable fashion for their business intelligence.

Star Schema

The star schema is the focal point of the data warehouse development, as it provides a denormalized database structure for reporting the data that is vital to the business needs. The star schema itself is simply defined as, “a dimensional design for a relational database” (Adamson, 10). The schema gets its name because of its appearance (seen in Figure 3 below), as it appears in the shape of a star with the dimension tables that surround the focal point of the star schema, the fact table. The surrogate keys are the integer indexed primary key in the dimension tables that becomes the foreign key just for linking to the fact table, and this creates the basis for the star schema. Once the links are in place to the dimension tables, the star begins to take shape, and this is very common to data warehousing. The fact table possesses the many keys to the related dimension tables and makes the star schema possible. Since the fact table describes a business process, “it should provide a comprehensive set of measurements, even if some are redundant” (Adamson, 39). This affords the possibility of having reporting made easier by having data already readily available within that table.

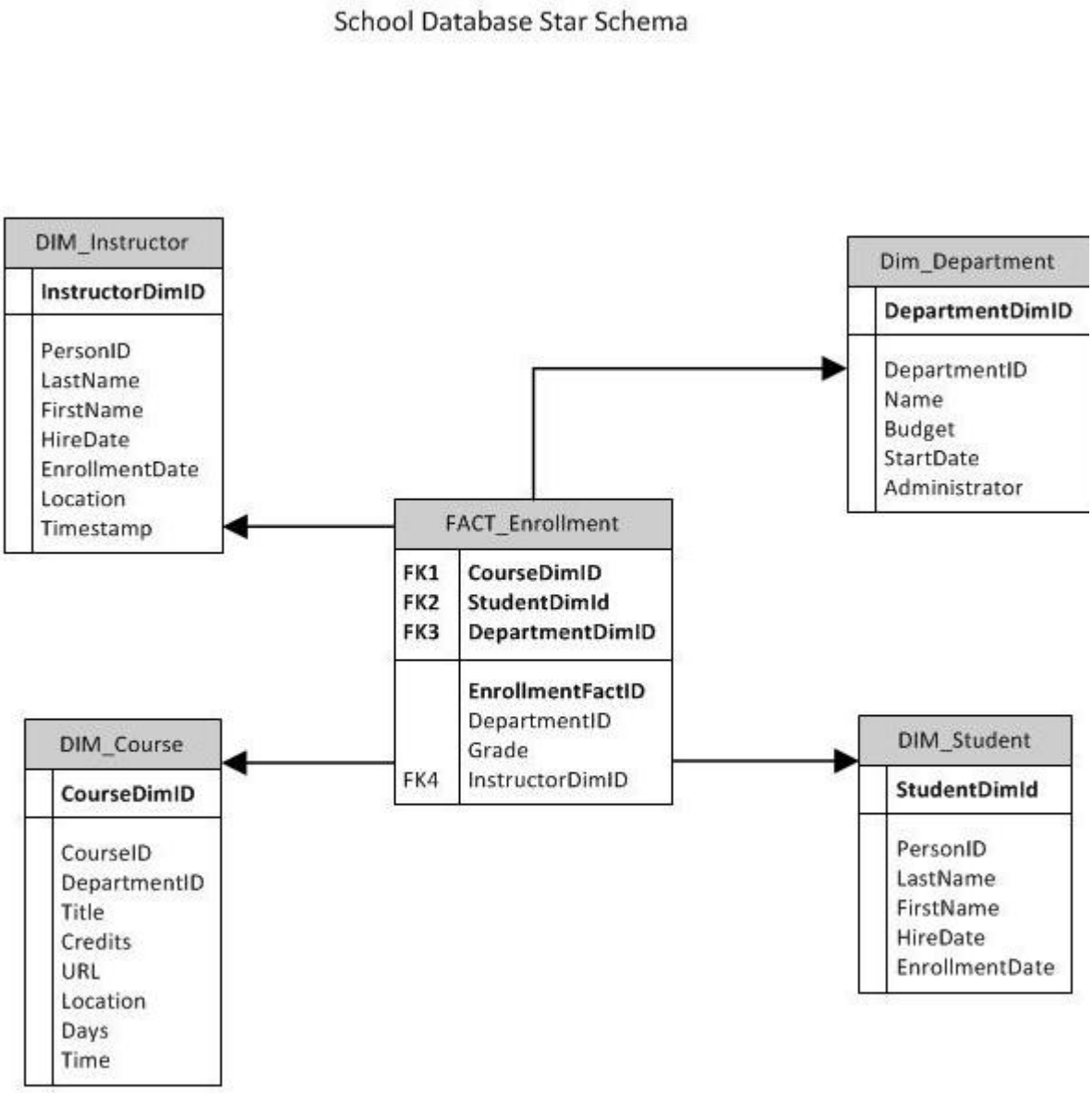


Figure 3: The Star Schema

The most similar table to that of the source data and the most valuable to the star schema itself is the dimension table. The dimension table contains the bulk of the data in the data warehouse, and it has the job of storing the attributes of data for that later move to the

fact tables. The dimension table will "consist of highly correlated clumps of attributes to represent the key objects of the business, such as products, customers, employees or facilities" (Kimball, Lifecycle, 242). The dimension tables will contain, for example, the address for someone in one table and all of their associated names, whether it be the first, middle or last in another dimension table. The dimension tables are normally the larger tables in the data warehouse compared to the fact tables and will contain reportable data once a query runs against them. In a higher education data warehouse, the data included will be the information about courses, instructors, students and employees. Figure 4 illustrates the composition of a student dimension table. The data is loaded to the dimension tables first in the ETL process because it there will be a primary key, which later is referenced in the fact table. The primary key to the fact table will be the indexed, integer field that has assignment as a dimension ID of an incrementing number from one by one, known as an Identity column in SQL Server. This dimension ID will copy to the fact table for referencing purposes, as the fact table builds from the ETL.

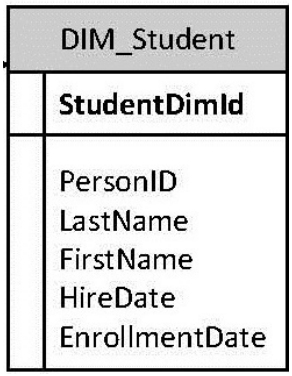


Figure 4: A Dimension Table

There are two types of dimension tables in a data warehouse based on the fact that they store history or they do not store historical data from the transactional data warehouse. The first and more agile of the dimension tables known as a "slowly changing dimension" that keeps up-to-date with data in the data warehouse by eliminating any history or keeping it to represent a current picture of data in the transactional databases. The first type of slowly changing dimensions would delete the existing row of data should an updated record be identified in the ETL process. The author warns against always using this type because of ease and to do so only "when the old value has lost all business significance" (Kimball, Lifecycle, 257). The second way to deal with this dimension is to insert the new row and store the old record with some type of date feature to denote which record the fact table should use for current reporting. Finally, the ETL process can still complete the insert as mentioned in the previous scenario, but would add an attribute, such as the word or note, to denote "current" and hence show the row of data that should be used. Considerations should also be made as to whether space concerns can be addressed with the storage of outdated information despite the fact it may be useful in some way.

The final piece of the dimensional model is the use of the fact table, the culmination and focal piece of the star schema, which provides the anchor for the eventual reporting design. The fact table stores the foreign key dimension ids from the previously built dimension tables. In order to be a fact table, "all fact tables have two or more foreign keys that connect to the dimension tables' primary keys" (Kimball, Toolkit, 18). This setup gives the user a chance to relate a specific action or transaction shown in the fact table and relate it to attributes in dimension tables for reporting. In an example in the prototype later in this thesis, and as

shown in Figure 5, the FACT_Enrollment table gives the user a chance to relate to the dimension table of the student to find their name, which may be vital to their report. Instead of having this data simply in the fact table already, it improves efficiency to link to the dimension for this information on a needed basis rather than consuming a table that contains more attribute information than it needs to have.

FACT_Enrollment	
FK1 **FK2** **FK3**	**CourseDimID** **StudentDimId** **DepartmentDimID**
	EnrollmentFactID DepartmentID Grade
FK4	InstructorDimID

Figure 5: Fact Table Example

In addition to storing the keys for related dimension tables, the fact table is famous for storing some type of measurement because "fact tables store the performance measurements generated by organization's business activities or events" (Kimball, Lifecycle, 238). Unless known as a "factless fact" table to the business intelligence developer, there will generally be at least one measure associated with a business transaction that the fact table represents. Taking the previous example of the FACT_Enrollment table, the measure associated with the table is that of a numerical representation of a grade, noted in the transaction where the student earns a grade by an instructor, and it is also available to the user to put in their report. In addition, the fact table will have its own unique identifier as a key for the table. Finally, the fact table has the ability to be stored as a "Snapshot Fact," which freezes the fact table at a point in time for reference later should it be necessary for historical purpose.

With all this data regularly loaded and standing up to queries constantly, especially in a setting where there can normally be records numbering from the hundreds of thousands to millions, setting indexes to the star schema tables within the data warehouse are crucial. The dimension tables will have an index applied to the dimension id, normally the primary key for that table. This would act as the clustered index since "the clustered index is usually unique or nearly unique; by default a table's primary key index is defined as a clustered index" (Kimball, Lifecycle, 346). Having a clustered index as in the case of the tables here will yield better results by a query being set against the star schema and attributes its speed to this application. Since fact tables also will have a unique fact id as the primary key, the same results of indexing will occur for these tables. It is the reference of fact tables to dimension id through the storage of the dimensions primary key as a foreign key to them, that data will readily available to query.

Data Marts

The final piece of the dimensional model of data that comprises a data warehouse is the term known as a "data mart," which is the further classification of an existing star schema that is part of a data warehouse. The data mart is a subset of data that usually pertains to a part of the business and may need only be concerned with a few fact tables and star joined dimension tables for queries and reports. Furthermore, once the data marts exist within the context of the data warehouse, users can provide input for updates or further customization without affecting other data marts. There are many ways to describe the data mart, but one of the best definitions that I have come across is in terms that most anyone outside of an IT environment would be familiar. Since most people are familiar with a shopping mall, everybody is also

accustomed with the fact that a shopping mall contains many stores, both large and small. Similarly, one can accurately describe that a data warehouse is similar to a shopping mall and a data mart is similar to a store within the mall and a store or collection of stores would represent star schema(s). Nonetheless, the coining and use of data marts, while simple and easy to understand to users and undoubtedly useful, have drifted away in recent years in favor of having the same structure but with less usage of this term.

Regardless of whether one uses the term "data mart" or not, there is still going to be certain star schemas used by only certain groups, leaving other parts of the data warehouse untouched by them. Before getting too deep into the makeup of data marts, it is intriguing to understand that Kimball says, "When we first started using the data mart terminology in the 1990s, we were describing process-centric detailed databases that represented a subset of enterprise's overall data architecture" (Kimball, Lifecycle, 248). This remains very true as he describes data marts as being just a small part of the overall solution in a data warehouse. However, those in IT stopped using the term because the term data mart, in their eyes and in general was "hijacked to refer to independent, non-architected, department databases" (Kimball, 248). Such an instance refers to someone spinning up a separate database or small data warehouse just for his reporting needs alone and not keeping his eye on the bigger picture. Regardless of Kimball disowning the term data marts, their idea is still relevant, especially in the case of a large Enterprise Data Warehouse where users are going to need to find their data amongst a sea of star schemas.

In order to create data marts, the very important prerequisite is the existence of the data warehouse, which is comprised of multiple star schemas, developed and named based off

the dimensional model. It is true that it might just be easier for a group of users or a department perhaps to have its own data warehouse for their own transaction database and they could accomplish the same basic results with a data mart or small collection of data marts. The advantage of having the data warehouse or in this case, the Enterprise Data Warehouse, is that there may be needs to traverse across data marts into other data marts, which would not be possible in their own silo-type of data warehouse. However, the largest advantage with an Enterprise Data Warehouse is that "you can shortcut the process of hunting the various candidate data sources, digging into their respective data environments to see what is there and what the quality looks like" (Simon, 8). Data marts offer a great advantage of having better control of data that users will need to find on a regular basis.

Once the data marts exist as part of the large data warehouse solution, each area now has the responsibility of consuming the data within the mart for their own needs. We are reminded that, "The most notable difference of a data mart from a data warehouse is that the data mart is created based on a very specific and predefined purpose and need for a grouping of certain data. A data mart is configured such that it makes access to relevant information in a specific area very easy and fast" (What Is a Data Mart, 1). If a basic user needs to go out to find their relevant star schema, it is very possible that within the scope of an Enterprise Data Warehouse, users could get lost or stumble across data that would net be relevant or applicable to their reporting needs. If the data mart presents the data properly for their use, they will quickly be able to develop the reports that they need to have in order to reap the benefits of a data warehouse in the first place.

In the case of an Enterprise Data Warehouse, there are times when one data mart may not be enough for the uses or organization to accomplish their tasks, and they would therefore need access to more than one specific data mart. By rule, "Within a single business organization, there can be more than one data mart. Each of these data marts is relevant or connected in some way to one or more business units that its design was intended for" (What Is a Data Mart, 1). This shows that if, for example, a user in the registrar department needed access to the student employment database information, it would not be impossible to find them through the proper security arrangement within the organization of the data mart. Later, I will illustrate such an undertaking by linking dimension tables from a pair of star schemas that resemble a snowflake type schema for the purpose of a specific report. Conversely, a user in said registrar department would never have any interest in finding data in the fundraising area and perhaps have no right to do so in the first place. Therefore, there could be some data marts, comprising a star schema, which all users would have access to use while others would drill down to only what is necessary based on security and job assignment.

There is a pair of advantages to the security settings available for controlling user's interaction when reporting within the data warehouse. First, unlike the transactional source database, it will not be necessary for users to generally have any ability or need to write to the data warehouse, as the ETL will handle such tasks to write or will the database developers and database administrators, instead. Secondly, and most importantly, there is a better opportunity to secure the access that a user will have to query the data warehouse, based off various rules. It is said that, "it is difficult in the relational database because the relational engine on its own doesn't have enough metadata to create multiple-level security roles"

(Kimball, Toolkit, 217). Having the metadata available in the data warehouse offers the chance to give partial restrictions to data as necessary, which would not be available to do in the transactional database otherwise.

Once the data warehouse is complete and the data marts are arranged, there is a better system in place for the users to better use the system to access the data that is relevant only to them. With a better-organized system, including business areas having their data well organized, the business users can better zero in on where changes need to occur or new reports need to be developed.

Snowflake Schema

There is yet another interesting schema in the data-warehousing world, known as a snowflake schema, and this is another way to bridge data with existing star schemas. The snowflake schema will appear when "the relationships between dimension attributes are made explicit in a dimensional design" (Adamson, 157). With data being similar in some databases or even within the same database, it is possible for such a relationship to exist within a collection of star schemas as a need may arise within the scope of the business requirement to do so. Much like how the star schema gets its appearance from the shapes it takes as designed; the snowflake is named due to its branching off the dimension table from a star schema. In the prototype, I found a snowflake type join helpful to link data from two star schemas. While vastly different in purpose, they have common relation enough to be able to design a report for the business.

ETL Alternative

There are many ways to build a star schema, but traditionally the best way is to design the schemas based on the needs of the business. However, there are a few instances where a star schema can be generated based on the input data. Three academics have devised a comprehensive way to generate a data warehouse schema based on the transactional database and a program developed to evaluate the data. In this case, they differentiate their proposal by affirming that "the work is not limited to designing the schema but it also populates the data from the clustered data source to the automatically generated schema tables" (Usman, 37). They use hierarchical clustering to help generate not only the star schema but also the snowflake schema and even the galaxy. Their program generates the schema and then deploys the final product to the database server. Such a program is a big improvement over the usual data modeling and manual development of a star schema then designing and programming the ETL to load the data to the data warehouse.

Their proposed model for automatic schema is comprised of a few parts to complete the process of taking raw data to the eventual schema in the data warehouse. Their first layer deals with the data mining where the chosen data set has assigned a hierarchy cluster is part of that data set. The "component also handles clustering parameters which include linkage method (Single, Complete or Average), clustering directions (rows, columns, both) and similarity/distance measure (Euclidean, Manhattan, Simple matching for nominal or Pearson correlation coefficient (Usman, 39). This is important because the hierarchal clustering is integral to the next step used to degenerate the schema. In the next layer, the clustered data becomes part of the schema generation by reading in the data now clustered, in the previous

step. Next, the type of table, that is dimension or fact, is determined and table relationships are also determined. Finally, the data moves into the final tables that make up the desired schema, be it the star, snowflake or galaxy.

The authors then demonstrate the use of their tool using data from a text file that eventually ends in a star schema. They indicate that they use the Euclidean algorithm for the experiment, and "the reason for this choice is that Euclidean distance is the most commonly used type of distance measure in cluster analysis" (Usman, 40). The program they use to execute the process is a basic C# program, which does go through the steps necessary to build a star schema in this experiment. It is interesting to think about whether this type of methodology using an algorithm would be effective rather than the manual designing, building and loading of the schema through an ETL. Still, it questionable whether this type of program can apply to every input situation or whether it would be unique as an ETL would be anyway, but there is no doubt that it would save an immense amount of time if this can be used universally. On the scale of an Enterprise Data Warehouse where inputs come from across campus systems, perhaps an individual application for each input would be reasonable.

CHAPTER 5: Implementation

This section includes a sample implementation of an Enterprise Data Warehouse on a significantly smaller scale that has only two star schemas. The solution shown here is an example of using the Business Intelligence tools with SQL Server and a SQL Server database, the same software that is appropriate for a full-scale implementation.

Enterprise Data Warehouse Prototype

The culmination of the thesis comes together with a prototype of an Enterprise Data Warehouse in a higher education environment. The purpose if this initial design of the Data Warehouse is to show not only how a series of tables can become a denormalized star schema, but also to show how an enterprise data warehouse can integrate data between two star schemas from two different systems all together.

Although this is on a smaller scale, the prototype will depict two transactional databases commonly found on a college campus, which are vital to the community in which they serve. The next step is to take this data out of the database by extracting it, then transforming it to put it a logical grouping, then finally loading it into an organized star schema. To show the user what is available, there are a few simple reports that will show data that a user at the college may find useful. The report illustrates the unique opportunity that a user can have in one report. The data that began as two very different databases and are in two different star schemas still are able to relate within the data warehouse. As the prototype illustrates, the ability to link data could be helpful in situations where one would need to have a comprehensive report that would otherwise come from report writers in separate

departments. It bears noting that this is the simplest of examples for the creation of an Enterprise Data Warehouse, but it shows the concept of the bigger picture of implementing a data warehouse. The final data warehouse is a SQL Server 2008 R2 database, which uses SQL Server Integration Services as the ETL tool and SQL Server Reporting Services as the reporting tool.

The first transactional database is the "HR" database, which comes with an Oracle XE 10.2 (Express Edition) installation. As depicted in Figure 6, the HR database portrays a database that would keep track of employees and job assignments as well as salary and personal information. Within this HR database, "each employee has an identification number, e-mail address, job identification code, salary, and manager" (Bauwens, 19). This database used in a college environment could not only to track regular employees but also the faculty who are also in the student system, which will be mentioned next.

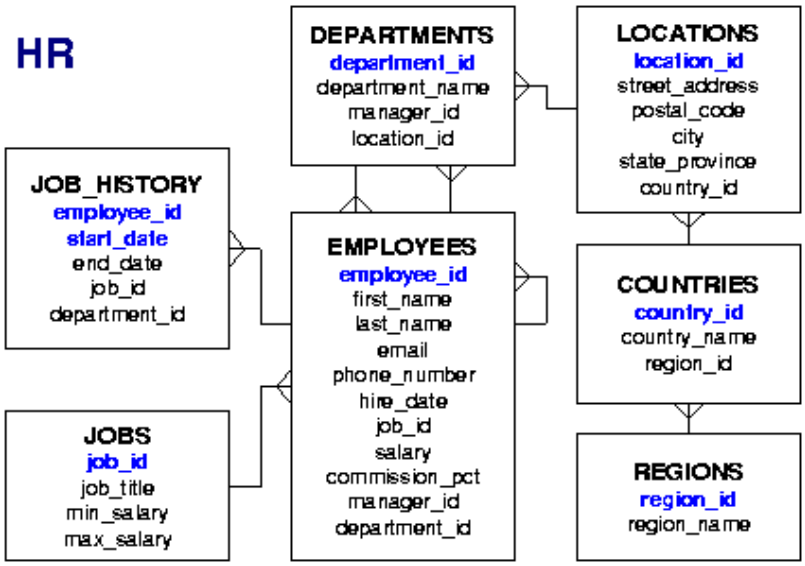


Figure 6: The HR Schema from Oracle Express Edition

The other transactional database that is part of this prototype is a database named "School," which is an example database that portrays a student system, a vital part of any college. This database keeps track not only of students but also instructors, classes, and locations, and it also has the potential for expansion if necessary. The database is in a Microsoft SQL Server 2008R2 relational database management system, hosted on a Windows operating system, as would typically be the case with this type of database software. The database is available from the Microsoft Developer Network, which shows the relations of a theoretical student system as well as installation instructions. In Figure 7, the schema is as designed as follows:

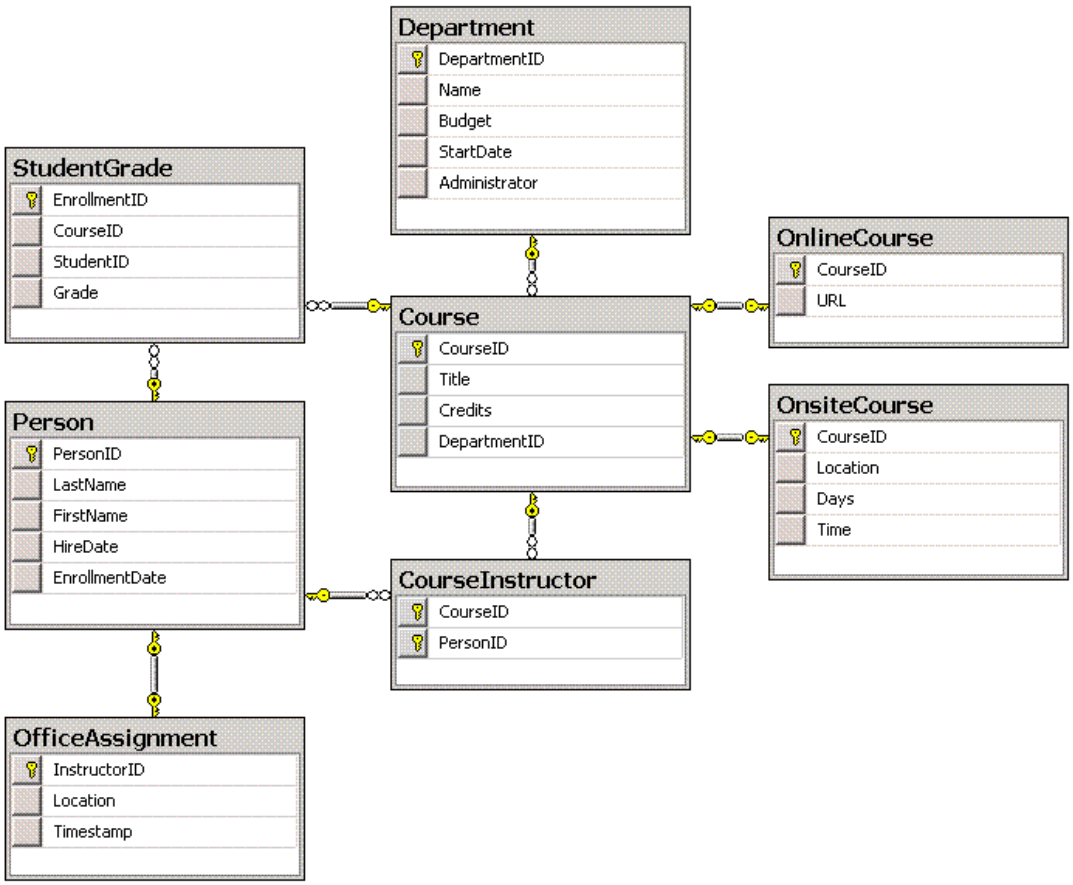


Figure 7: The "School" Schema from Microsoft SQL Server

The design of the data warehouse will involve a star schema for each transactional database, with the student system represented as the enrollment data, whereas the HR database represents the employee. The dimension tables are direct from the database tables, adding a dimension ID that is an identity-valued field and incrementing one by one. Some of the dimension tables will have data from other tables, such as a lookup table. The Fact table will have another unique Fact ID with all the related Dimension IDs along with the vital measures that will easily report data. For the sake of this example process and because of the size, all of the destination tables need to start empty before being loaded with data. The idea is to normally increment rather than loading the entire data warehouse with each run. However, due to the size and purpose, this data warehouse will not involve every detail of a data warehouse, but it will still convey the basis design and result.

ETL Construction

Below in Figure 8, is the main screen of the ETL in SQL Server Integration Services, used to generate the "School" Data Warehouse. Each schema task is within a "Sequence Container," which completes all steps inside the container before proceeding. The green arrows mean that the next task completes only upon a successful conclusion of the task.

.

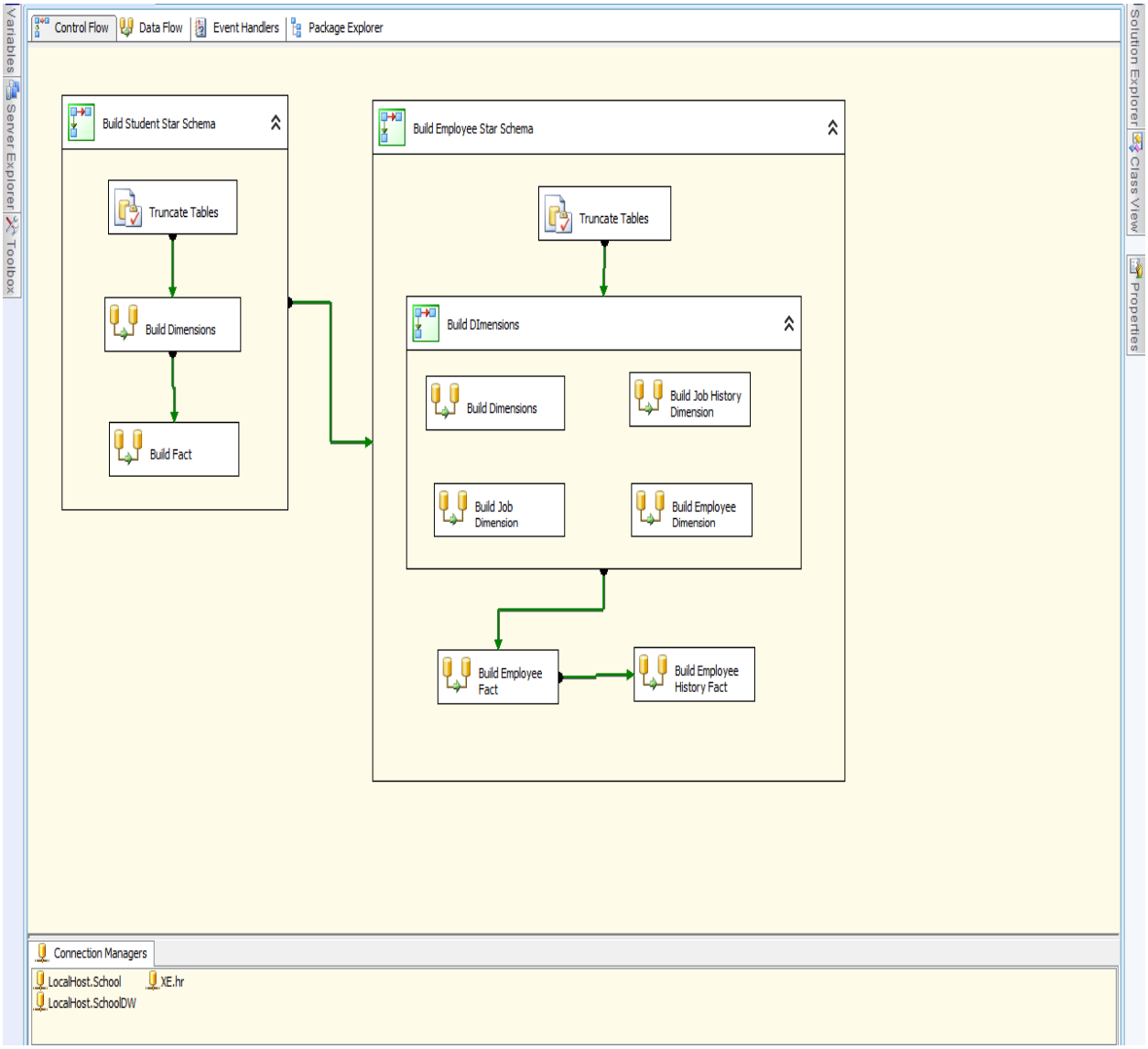


Figure 8: The Complete ETL Program.

Student Star Schema

The first part of the ETL as shown in Figure 9 will entail the building of the student related star schema, beginning with the dimension tables and ending in the loading of the fact table.

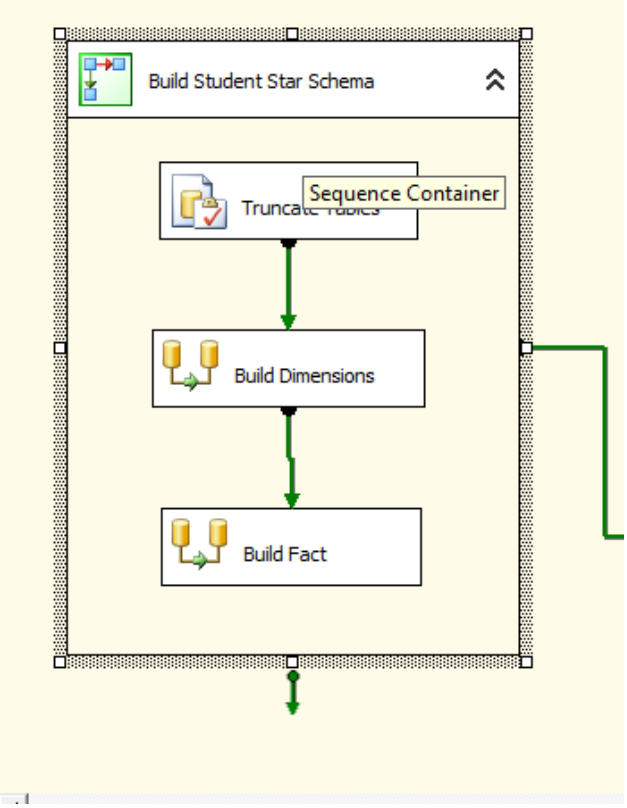


Figure 9: The Sequence Container to Build the Student Star Schema.

The next "Data Flow" or component of the ETL will build the dimensions. Within the data flow, multiple extracts, transformations and loading occur at the same time. Since data extract from multiple tables, all the work of the ETL can occur at the same time. In a larger scale ETL, this may not be the most appropriate action because you would want to handle it one at a time. The activities within the task start with an extract task and will select the table by selecting the actual table or using SQL. If it is necessary to link a value from another table, you can use a "lookup" task to relate to another table to add that value if necessary before loading to the destination table. For example, when extracting the instructor and before loading to the dimension table, it is necessary to lookup the office location for the instructor.

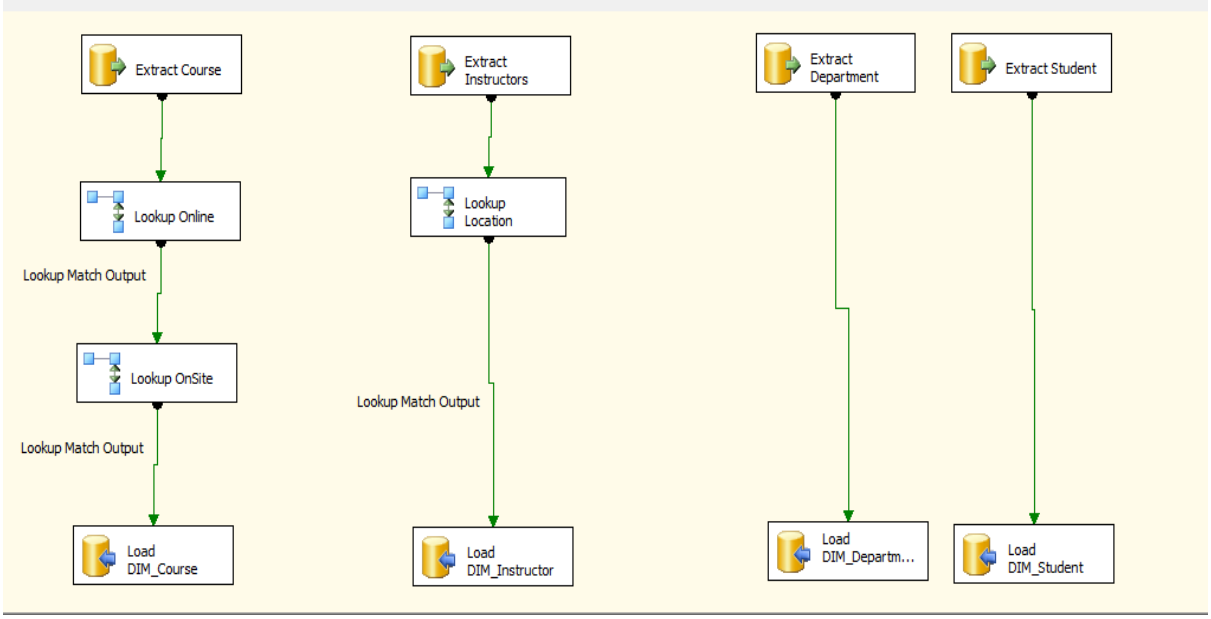


Figure 10: All Dimension Extracts, Transforms and Loads.

The code used in Figure 10 to complete the extracting and transformations are rather simple SQL, which will pull the necessary data and make joins to the other transactions systems. Here is the code needed to extract for the school dimensions:

DIM_Course

```
SELECT CourseID, Title, Credits, DepartmentID
FROM COURSE
```

DIM_Instructor (where an instructor is denoted as having a "hire date")

```
SELECT *
FROM DBO.PERSON
WHERE HIREDATE IS NOT NULL
```

DIM_Department

```
SELECT *
FROM DBO.DEPARTMENT
```

DIM_Student

```
SELECT *
FROM DBO.PERSON
WHERE HIREDATE IS NULL
```

This series of tasks is within the data flow container to build the fact table. It is important to note that since the fact table will use the dimension surrogate keys, the dimension tables must always be loaded first.

Once the dimensions are loaded and after they are loaded completely, the fact table comes together with the data from the other dimensions. The process begins with extracting data from the table in which we want to make the focus of the star schema. In addition, when extracting the table, there can be additional SQL code to extract what measures will carry over to the final fact table. In between, there will be the entire dimension IDs from all the previously created dimension tables.

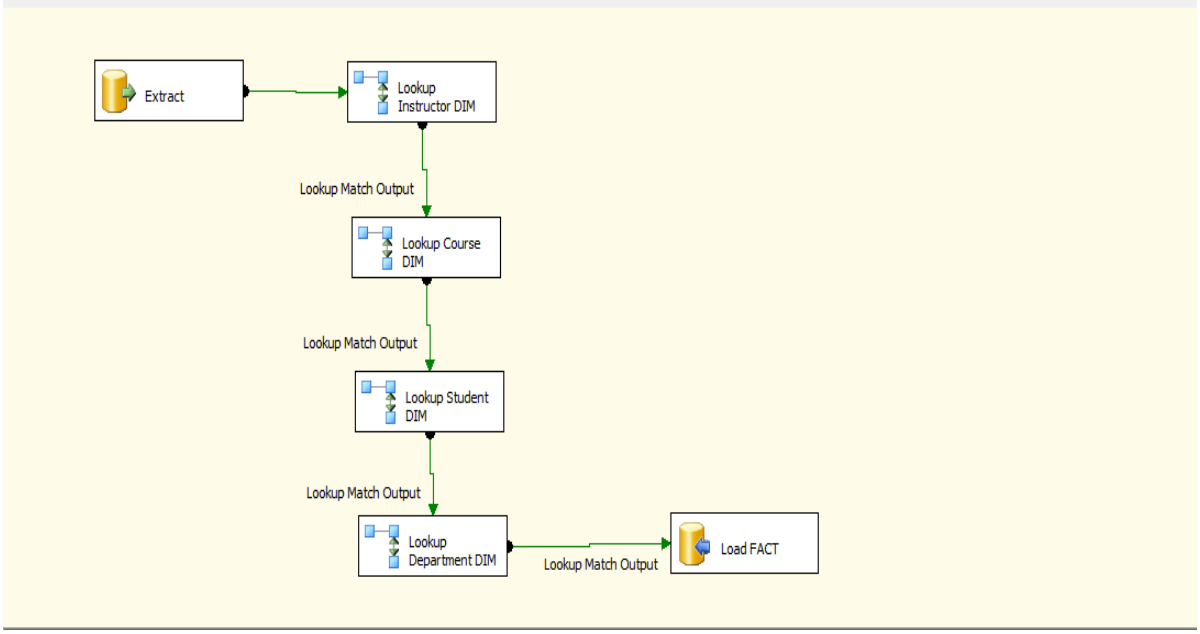


Figure 11: Task to Build the FACT_Student Table.

The code used in Figure 11 to complete the extracting and transformations are once again rather simple SQL, which will pull the necessary data and make joins to the other transactions systems. Here is the code needed to extract for the school fact table:

Main Extract

```
SELECT S.*, PersonID
FROM dbo.StudentGrade S
Left Join dbo.CourseInstructor C ON C.CourseID = S.CourseID
```

The remaining joins happen through a transform task known as a "Lookup." The lookup shown in Figure 12 will handle the join using the graphic user interface by linking the available columns from the first extract, then join to the appropriate dimension table using the join of like fields.

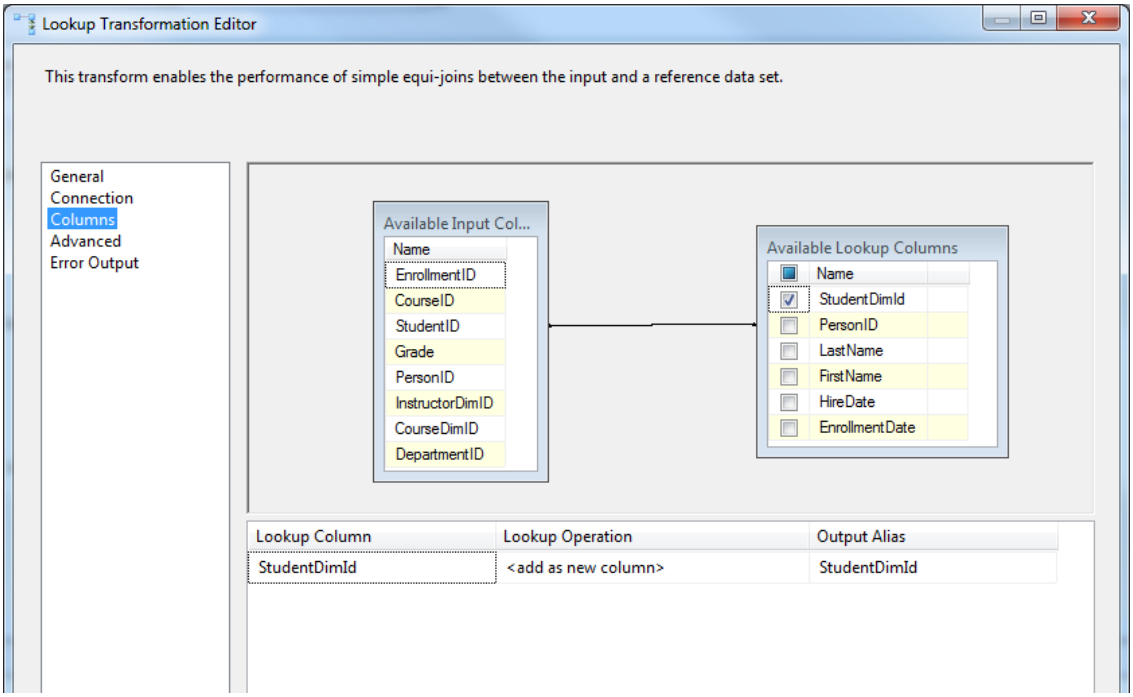


Figure 12: The Lookup Transform Editor's Lookup and Join.

Employee Star Schema

The next grouping of sequences will involve the building of the employee star schema within the prototype Enterprise Data Warehouse. This star schema comes entirely from the Oracle database "HR" that contains sample employee data and has the potential to be used to link to the student star schema previously created.

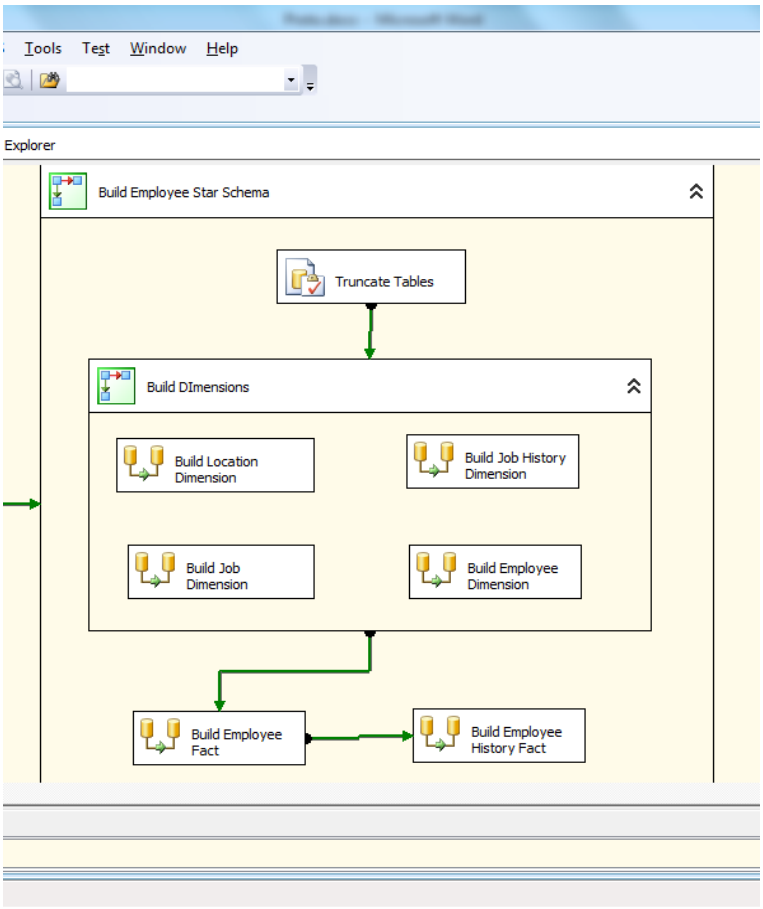


Figure 13: The Sequence Container to Build the Employee Star Schema.

As in the previous sequence container, the tables have a step to truncate in order to clear out the existing data in the dimension and fact tables for the purpose of this demonstration. The next sequence container in Figure 13, which is within the container to build the employee schema, will build each dimension. In the previous build of the student schema, all dimension tables build within one-step rather than having them separated into separate data flows as shown above. From experience, I find that there are compatibility issues in SQL Server Integration Services that do not translate to Oracle well, most likely because of the SQL Server architecture. Nonetheless, the task will occur without further issue and can still be run concurrently in order to save time. When building the dimension tables, there are

times when I once again have instances where some values have lookup transformation and others that directly load to from the Oracle table to the SQL Server Dimension table. The reason for this is because, as in the example of the locations dimension, it is ideal to eliminate a need for a an additional dimension table in countries, when it can easily be looked up based on the value and loaded from another small countries table. Another example of this is the lookup of the department and manager tables during the development of the job history dimension.

Most of the dimension tables are extracted using the simple selection options available in SSIS rather than using SQL. Although the native selection for SSIS is T-SQL, using that against an Oracle database would not be successful. However, since the ETL supports querying using the SQL native to the particular database, this is usually not an issue. In this prototype, due to the simplicity of the data, I am instead using the full table selection as revealed in Figure 14, no different from a “Select *” or select all.

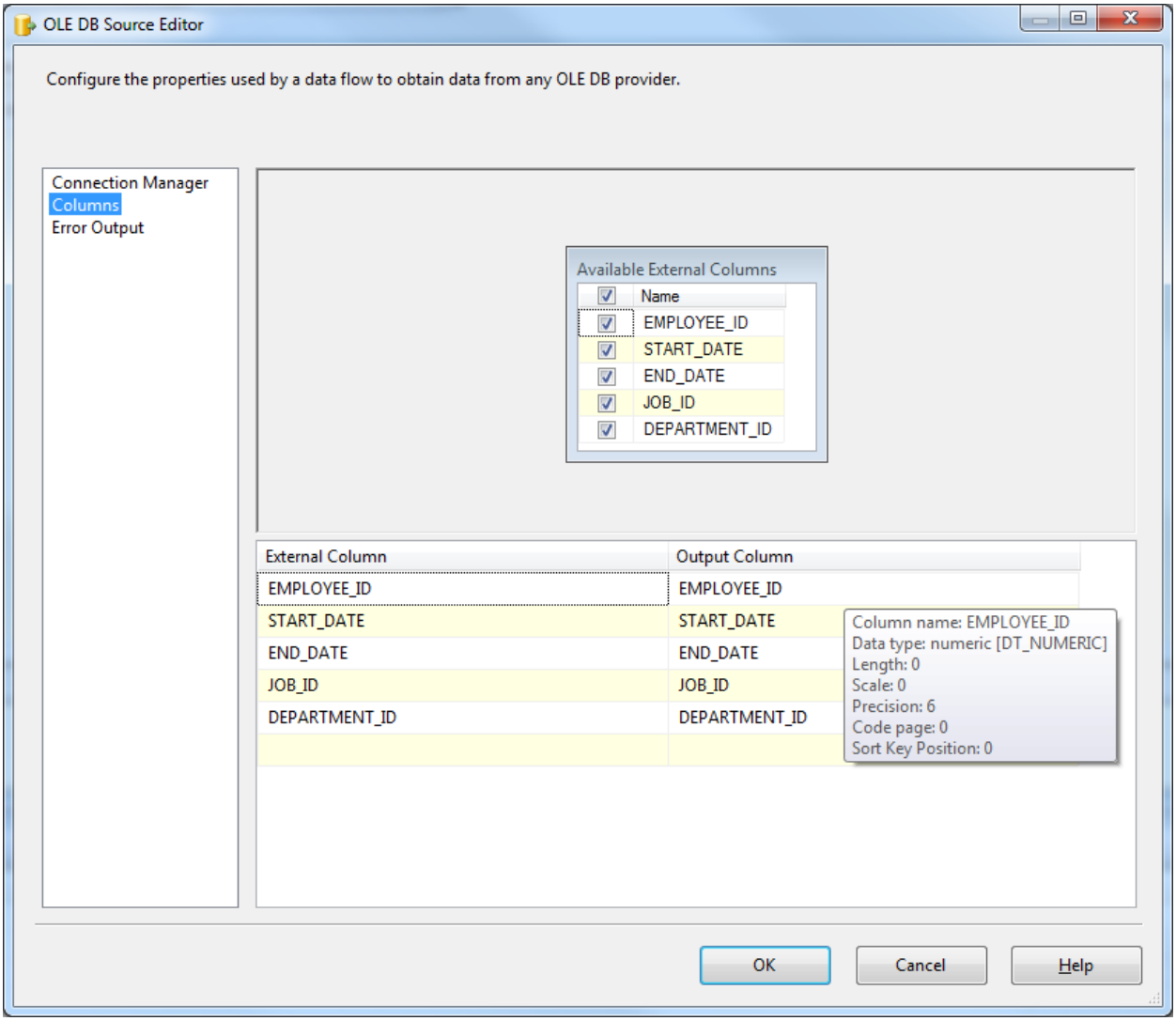


Figure 14: The Extraction Source Editor to Pull All Data from a Table.

The final step of building the example of the data warehouse is where the employee fact tables come together to have the dimension IDs and relevant measures within the table. There is a fact table for the employee but there is also one for employee history. This star schema takes on more of a form of a parent-child relationship because it links two fact tables together rather than just one fact table and dimension tables. The main reason for having the second fact table that has the name "Employee History" is that the source database table of "Job

History" does not lend itself well to the main "Employee" fact table because of the historical data whereas the "Employee" fact table is more the custodian of current data. One of the most important reasons for having an Enterprise Data Warehouse is the ability to hold historical data but also the ability to link easily to the current fact table. Salary, job title and percent commission are measures for the main employee fact whereas the department name is the measure for the employee history fact.

The final arrangement of the database tables are in Figure 15:

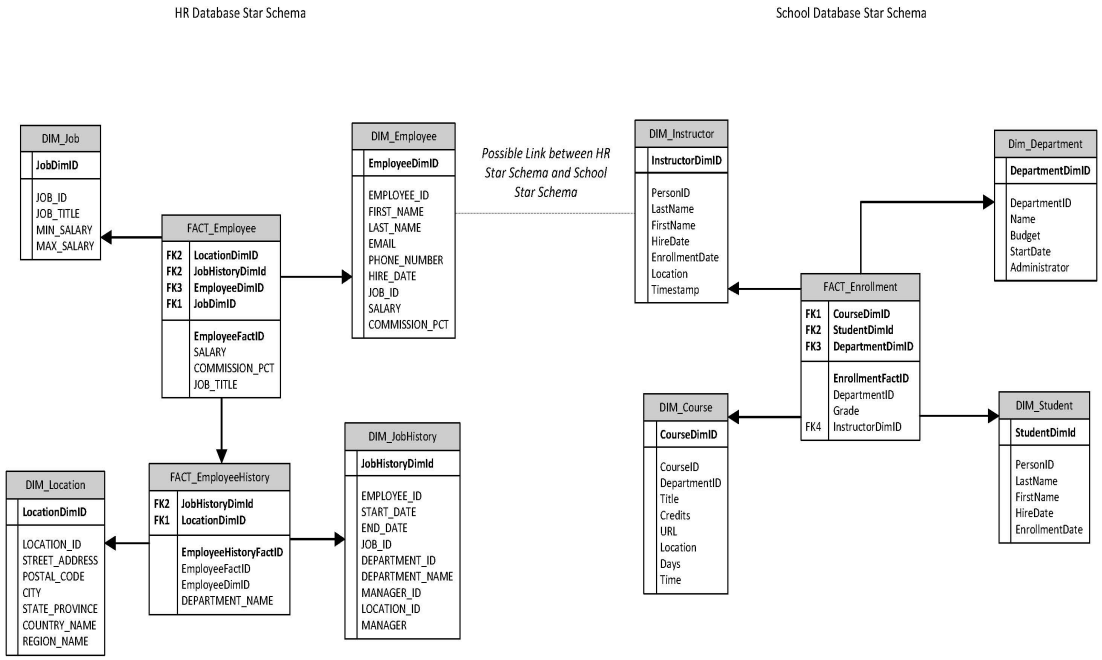


Figure 15: The "School Data Warehouse" Schemas.

Reports

Now that the prototype data warehouse exists, the question is what we can do with the data to show the advantages of an Enterprise Data Warehouse. The exercise here would be to

find an instructor from the SQL Server school database and find their salary from the Oracle HR database. To provide background, upon hire, a user will enter a new instructor into the HR database then assign an employee id. Upon their assignment to a class, a completely different user enters the instructor with the employee id referenced from the HR database. The instructor would have a separate "Person_ID" for that Person table. The data warehouse would then process the updates into the data warehouse and write these updates to the dimension and fact tables. As such, by forming a link to the Dim_Instructor and DIM_Employee tables through the Employee ID, one can link between the two realms in a "snowflake" schema instance where dimension tables are able to link. In my example, I chose an instructor to show his salary. From the student system, or School Database, there is no mention of salary. By linking through the DIM_Instructor Table to the DIM_Employee table in the HR star schema, you can further link to the FACT_Employee salary measure. The report shows how one can accurately link and report the salary data for the instructor.

The example report I have developed simulates how a report writer within an organization can consume the data that is now available in the Enterprise Data Warehouse, but on a smaller scale. The report, using SQL Server Integration Services (SSRS), which is part of the Microsoft BI suite of products used as part of this solution, is the appropriate way to show an example of the ability to link amongst star schemas. By building the report to carry out the above query, it is simple and fast to report upon the example within the report. The code to build the report is:

```
SELECT
  FACT_Employee.EmployeeFactID
  ,FACT_Employee.EmployeeDimID AS [FACT_Employee
EmployeeDimID]
  ,FACT_Employee.JobDimID
```

```
  ,DIM_Employee.EmployeeDimID AS [DIM_Employee
EmployeeDimID]
  ,DIM_Employee.EMPLOYEE_ID
  ,DIM_Employee.SALARY AS [DIM_Employee SALARY]
  ,DIM_Instructor.InstructorDimID
  ,DIM_Instructor.PersonID
  ,DIM_Instructor.LastName
  ,DIM_Instructor.FirstName
  ,FACT_Employee.SALARY AS [FACT_Employee SALARY]
FROM
  FACT_Employee
  INNER JOIN DIM_Employee
    ON FACT_Employee.EmployeeDimID =
DIM_Employee.EmployeeDimID
  INNER JOIN DIM_Instructor
    ON DIM_Instructor.EmployeeId = DIM_Employee.EMPLOYEE_ID
```

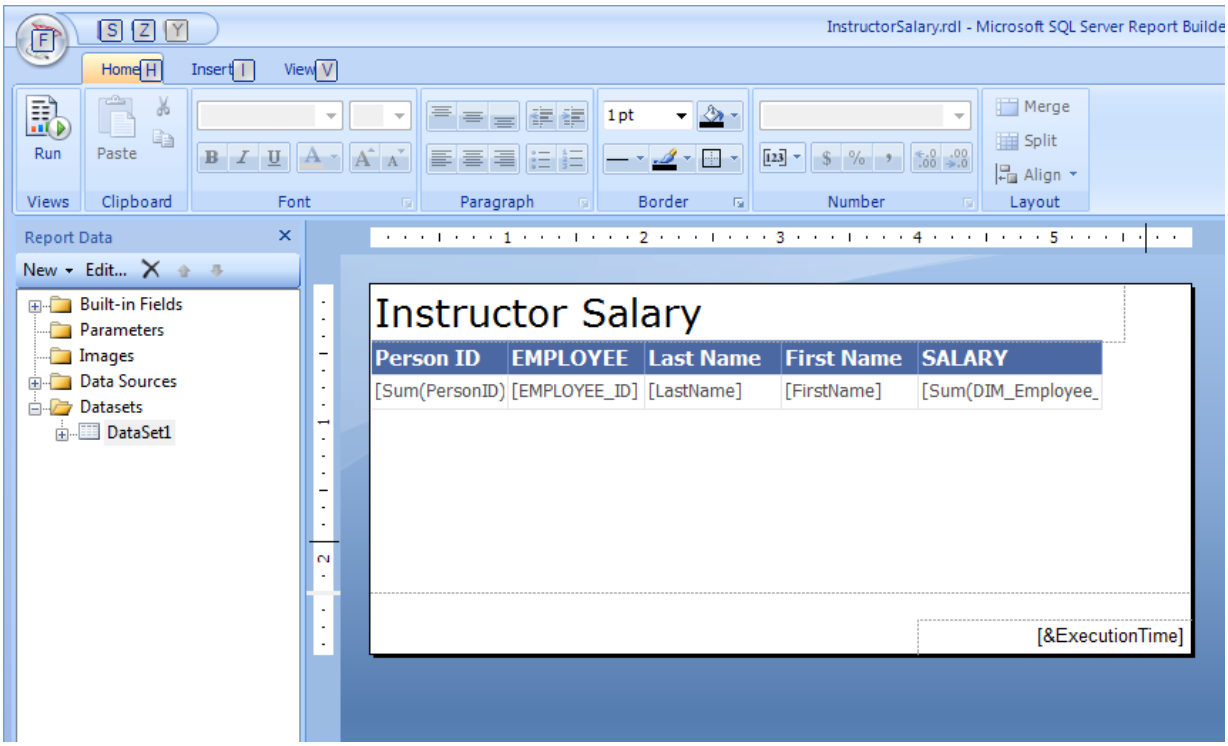


Figure 16: The Resulting Report Design for Instructor Salary.

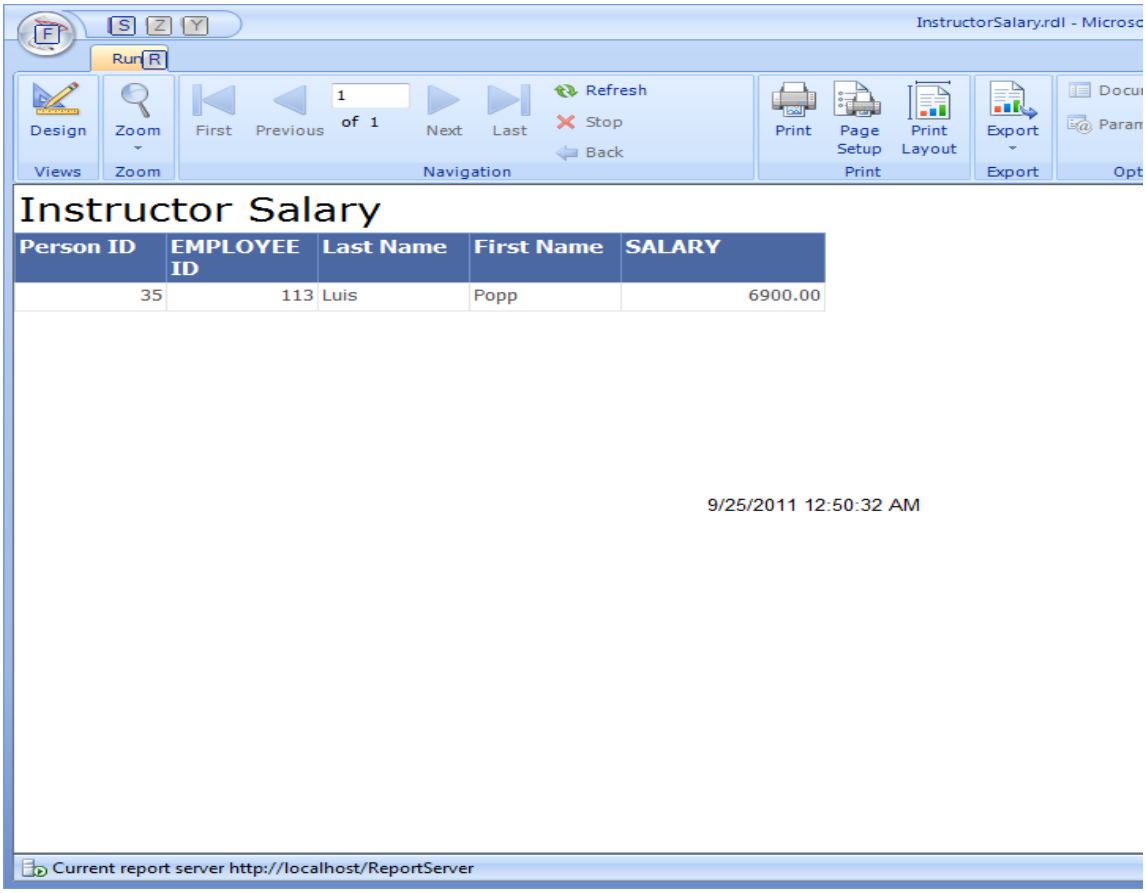


Figure 17: The Report Results for Instructor Salary.

The next example report I have developed simulates how a report writer could develop a grade report for the semester. The report, using SQL Server Integration Services (SSRS) shows how to create a simple report linking the fact table to the related Dimension tables. By building the report to carry out the above query, it is simple and fast to report upon the example within the report. The code to build the report is:

```
SELECT E.Grade, I.LastName as Instructor, C.CourseID,
C.Title, C.Credits, S.LastName, S.FirstName, D.Name
FROM FACT_ENROLLMENT E

INNER JOIN DIM_Instructor I ON E.InstructorDimid =
I.InstructorDimID
INNER JOIN DIM_Course C ON C.CourseDimID = E.CourseDimID
INNER JOIN DIM_Student S ON E.StudentDimId = S.StudentDimId
```

```
INNER JOIN Dim_Department D ON E.DepartmentDimID =
D.DepartmentDimID
```

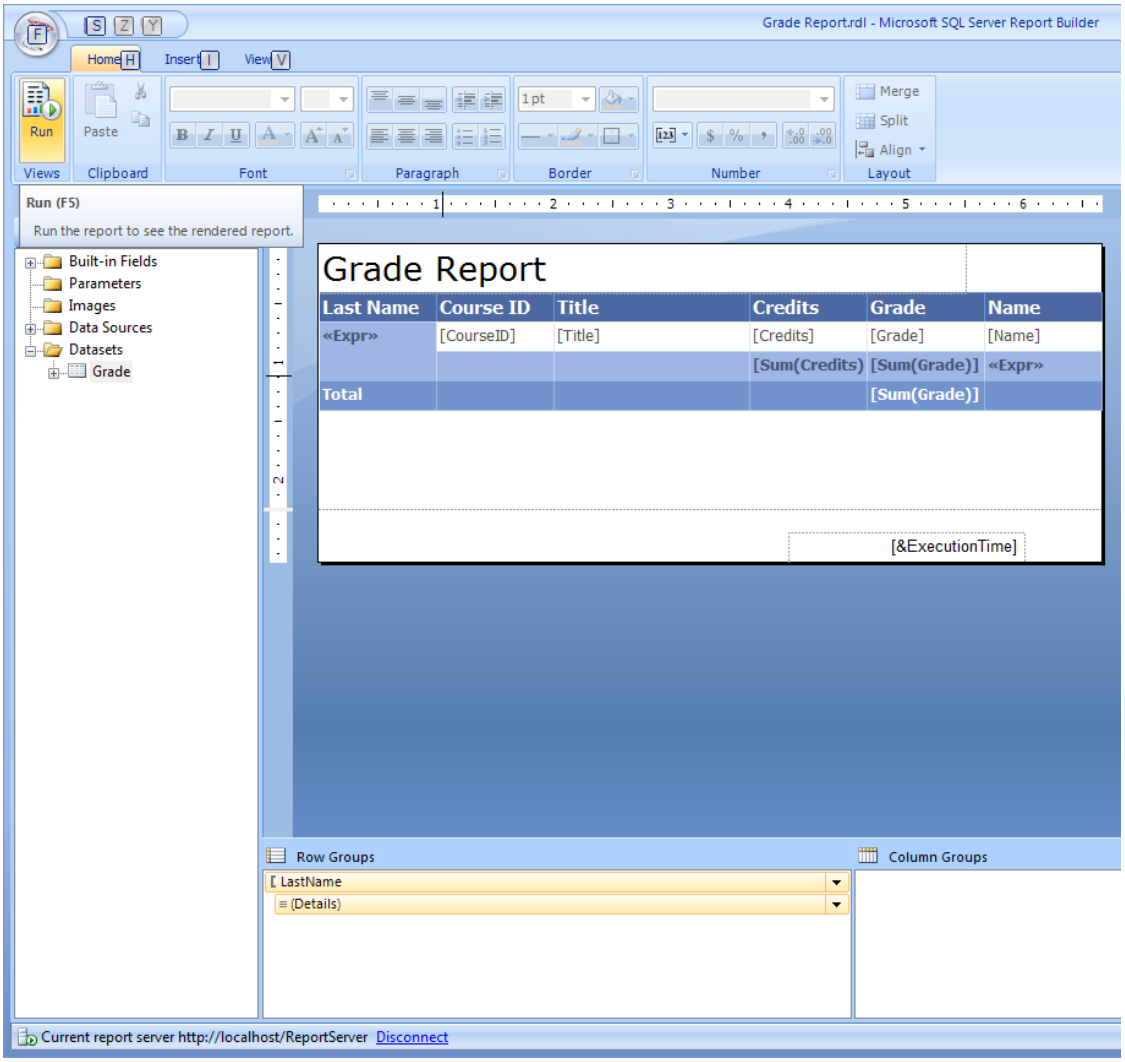
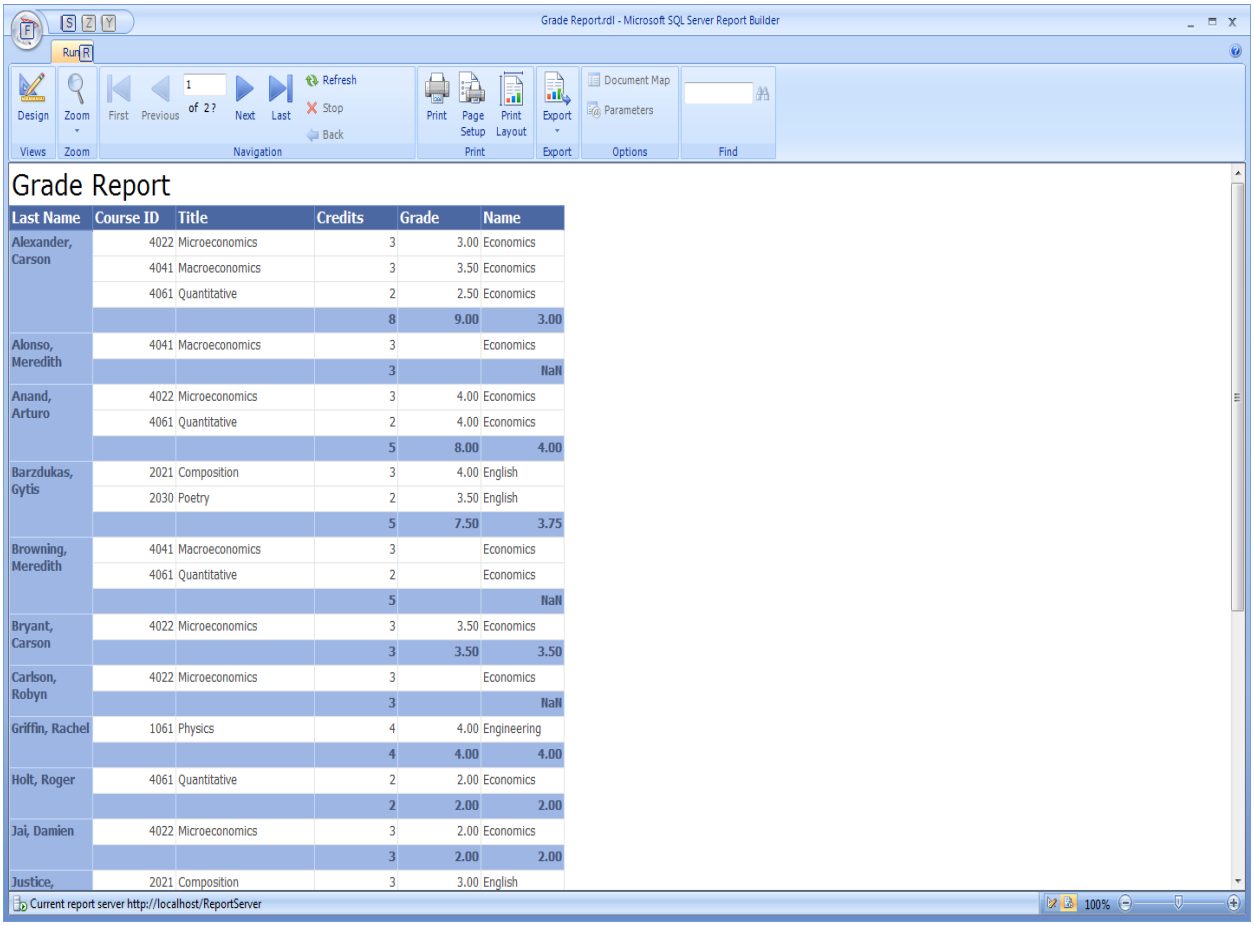


Figure 18: The Resulting Report for the Grade Report.



Last Name	Course ID	Title	Credits	Grade	Name
Alexander, Carson	4022	Microeconomics	3	3.00	Economics
	4041	Macroeconomics	3	3.50	Economics
	4061	Quantitative	2	2.50	Economics
			8	9.00	3.00
Alonso, Meredith	4041	Macroeconomics	3		Economics
			3		NaN
Anand, Arturo	4022	Microeconomics	3	4.00	Economics
	4061	Quantitative	2	4.00	Economics
			5	8.00	4.00
Barzdukas, Gytis	2021	Composition	3	4.00	English
	2030	Poetry	2	3.50	English
			5	7.50	3.75
Browning, Meredith	4041	Macroeconomics	3		Economics
	4061	Quantitative	2		Economics
			5		NaN
Bryant, Carson	4022	Microeconomics	3	3.50	Economics
			3	3.50	3.50
Carlson, Robyn	4022	Microeconomics	3		Economics
			3		NaN
Griffin, Rachel	1061	Physics	4	4.00	Engineering
			4	4.00	4.00
Holt, Roger	4061	Quantitative	2	2.00	Economics
			2	2.00	2.00
Jai, Damien	4022	Microeconomics	3	2.00	Economics
			3	2.00	2.00
Justice,	2021	Composition	3	3.00	English

Figure 19: The Report Results for the Grade Report.

The final example report I have developed simulates how a report writer could develop an employee title history with salary for the human resources department. The report, using SQL Server Integration Services (SSRS) shows how to create a simple report linking the fact table to the related Dimension tables. Since the "salary" is a measure of the FACT_Employee table, the data are already there for the report and only need to link to the DIM_Employee table. By building the report to carry out the above query, it is simple and fast to report upon the example within this report. The code to build the report is:

```
SELECT FE.SALARY, JOB_TITLE, EMPLOYEE_ID, LAST_NAME
FROM dbo.FACT_Employee FE
INNER JOIN DIM_Employee E ON E.EmployeeDimID =
FE.EmployeeDimID
```

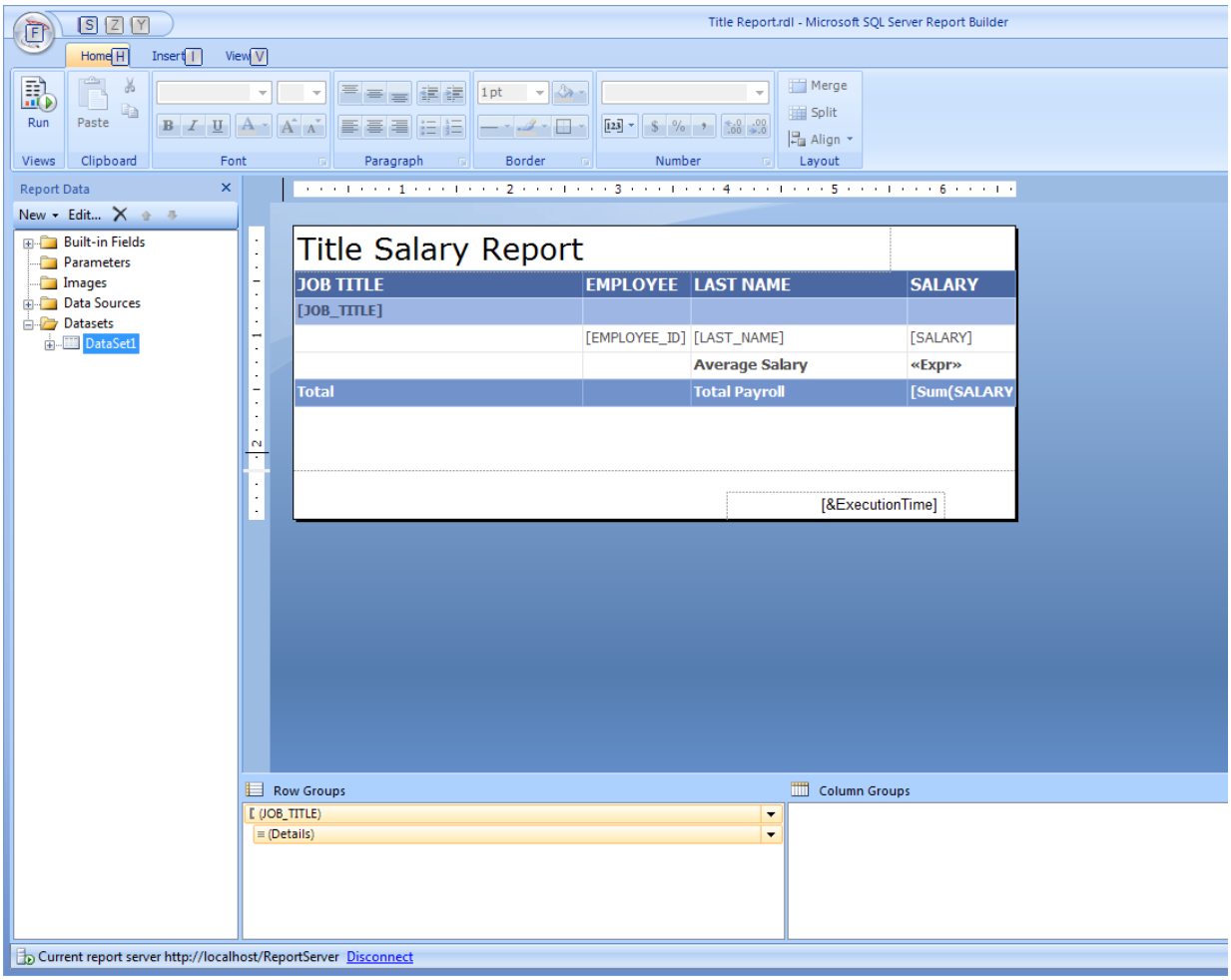


Figure 20: The Resulting Report for Salary Report Based on Title.

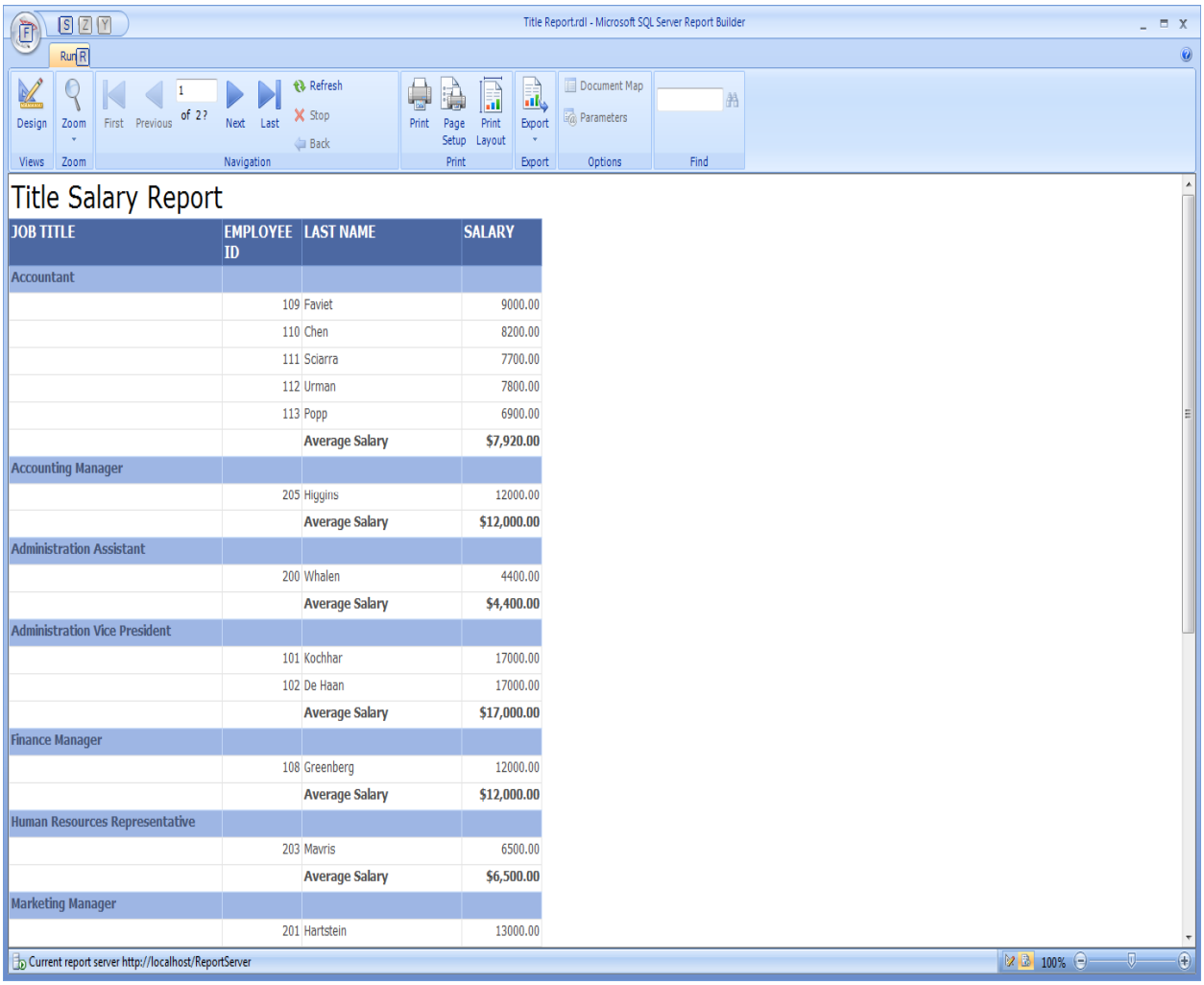


Figure 21: The Report Results for the Salary Report.

The following is an example of integrating data between a student system and the employee or HR database. Most of the time, it would be somewhat difficult if not impossible to link between the transactional databases. Sending the data to the enterprise data warehouse gives users the opportunity to successful integrate the data and report upon data from two different systems.

CHAPTER 6: Performance Evaluation

Overview

The Enterprise Data Warehouse is able to show how reporting can be improved, but it cannot offer a chance to compare the same report and query against an existing system. The purpose of this exercise is to take a real world example of the possible improvement in query time, comparing a transactional database and a denormalized star schema in a data warehouse. In order to make the data warehouse a viable option and to make it a worthwhile endeavor, it is important to show that there will be a big improvement in query time.

This effort involves solving a problem that existed where the solution was to create a data warehouse star schema. In a large database, it is especially time consuming to run reports against due to the complexity of the view and multiple joins to other tables. Due to the complexity of the data, I took apart the view piece by piece and designed a working ETL and arranged the tables into a star schema. The goal was to have a fact table that matches the count of records as the view does so that the exact same data that is present in the results of the database view would also be in the fact table. The fact table links to a collection of dimension tables, which contain the bulk of the attributes that the view returns.

The expectation in this testing set is that the data returned from the data warehouse will be less time consuming than from running the same query against the transactional database. The reason that there is a probability for a difference in the query times is due to the essence of the organization of the data warehouse. The source system, or transactional database, is a typically large “normalized” database designed, “to organize data into stable

structures, and thereby minimize update anomalies" (Sanders et al. 1). In the star schema used for this exercise, the data appears in a denormalized state, which is typical in the design and implementation of star schemas, organized to optimize querying. In this example, one of the star schema dimension tables built by the ETL uses a denormailization technique, which collapses tables, through joining by one to one relationships, adding only the necessary columns into a dimension table rather than the need for completing additional joins at runtime. For example, one of the main dimension tables used for the warehouse testing set collapses five tables from the transactional system to one, using only the necessary columns from each for the query. Having the data from one or more source tables organized into a logical dimension table will improve query results in the warehouse by having the columns readily available, having had the ETL complete the table join processing up front. By using a the view in the source system that was developed to query data for a report, it attempts to denormalize the data for the purpose of the report, but "since most DBMS products process view definitions at run time, a view does not solve performance issues" (Sanders et al. 7). The desire to move away from the complexity of this or a similar view that used as part of this testing set could provide the ultimate motivation for using a data warehouse to complete reporting for users.

It bears noting that the ETL for this star schema runs for approximately two minutes as it executes the SQL that the view does, but it runs in separate sections for the dimension tables then runs in for the fact table. Two natural downsides to having this data warehouse star schema is the time it takes to load the data plus the fact that the data will only be up-to-date as of the last refresh of the ETL. Since there is an initial time investment to load the data warehouse, it reasons that since the same time is necessary to run the query as to run the

refresh of the data warehouse, there is no real reason to do so in the first place. However, on a larger scale, if this data is to be repeatedly queried and reported upon all day, having a better performing system overall is a logical choice.

Testing Plan

In order to test the differences between the two systems, I designed the test simulations to best compare the two database queries. The source of the view is in an Oracle database while the data warehouse is in a SQL Server database, loaded from the Oracle database. There are three sets of testing, using the same query, translated in either Oracle SQL or Transact SQL or T-SQL against the SQL Server database. The query tool used for the Oracle database is SQLPlus, where headers suppress and timing is set. Conversely, I used the SQLCMD task from the Windows command prompt to login and run the timed queries against the SQL data warehouse.

Testing Set 1: Selecting All and Top

This set of queries select all from the database, querying all records then the top 100 to 100,000 records increasing by a factor of 10 each time. These sets of queries will test the database engines to tax it by selecting all, which is the most intensive query. After reviewing all the results in Figure 22, it is interesting to see big gap in time between querying the view versus the data warehouse.

	View	DM	Difference
Select TOP 100	00:06.9	00:00.0	00:06.9
Select TOP 1000	00:07.9	00:00.2	00:07.7
Select TOP 10000	00:18.9	00:01.7	00:17.2
Select TOP 100000	02:14.8	00:17.8	01:57.015
Select *	02:42.0	00:35.5	02:06.5

Table 1: Testing Set 1 Results (in Minutes)

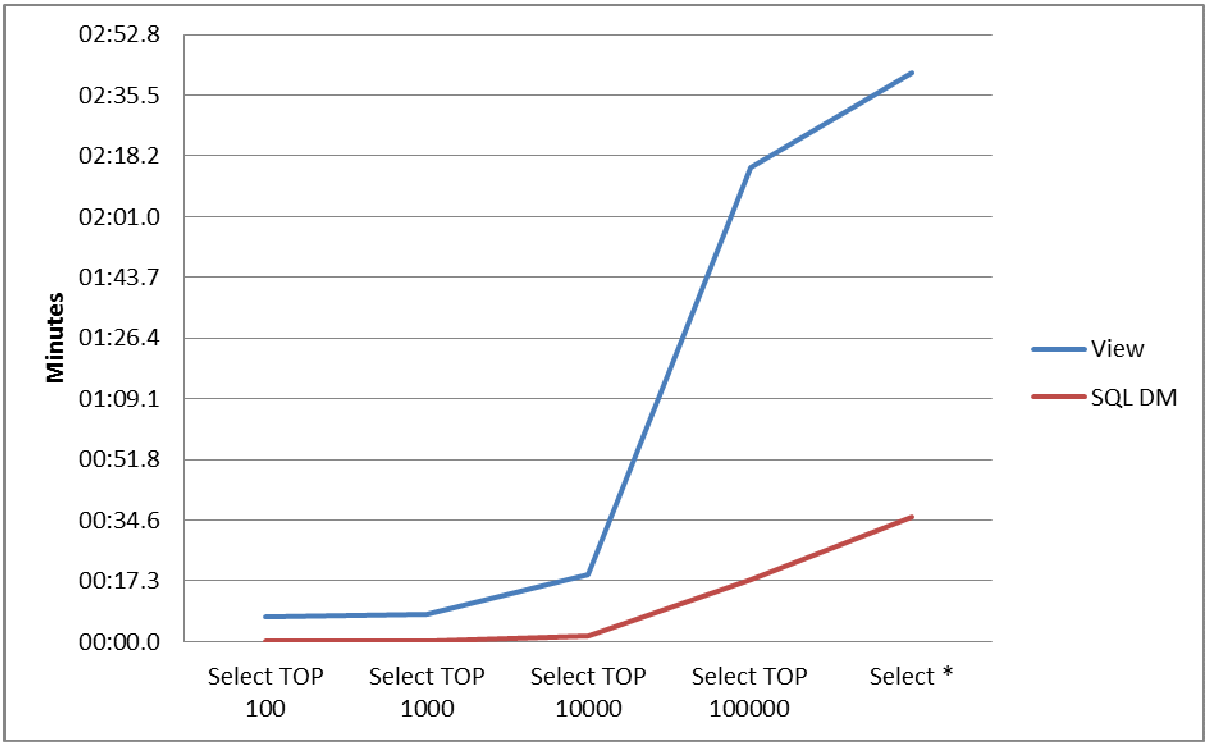


Figure 22: Graphical Results of Select All Queries.

Testing Set 2: Selecting All Random

This next set of queries select all from the database, querying random records then the 1 to 100,000 records increasing by a factor of 10 each time. This time, the idea is to see the data select random records rather than the top numbers. After reviewing all the results of

Figure 23, it is once again clear to see that there continues to be big gap in time between querying the view versus the data warehouse. One item of note is that the numbers slightly close in at under two minutes, but not enough to say that the warehouse query is not an improvement.

	View	DM	Difference
SELECT 1 Random	00:03.5	00:00.2	00:03.3
SELECT 100 Random	00:03.6	00:00.2	00:03.4
SELECT 1000 Random	00:04.8	00:00.4	00:04.4
SELECT 10000 Random	00:19.9	00:02.0	00:17.9
SELECT 100000 Random	02:38.7	00:57.3	01:41.357

Table 2: Testing Set 2 Results (in Minutes)

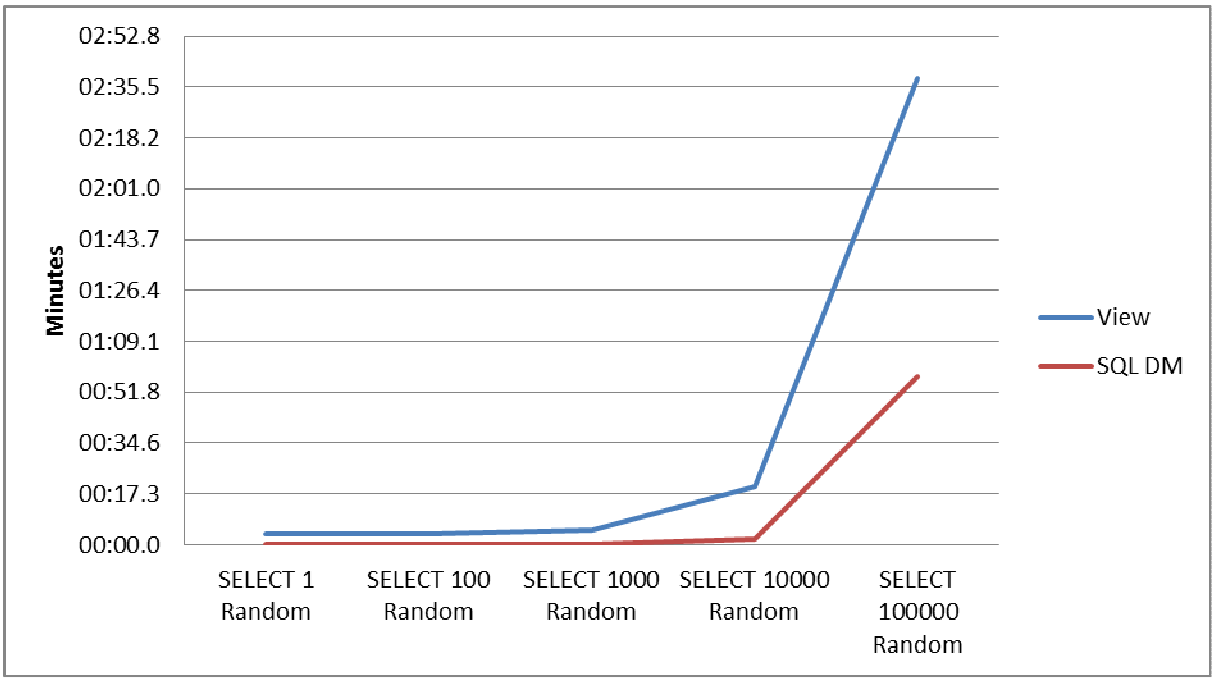


Figure 23: Graphical Results of Select All Random Queries.

Testing Set 3: Selecting Various

The final set of queries select all from the database but this time by using a where clause rather than all records. Although all the data were easy to query from the view because of the complexity, the respective query on the data warehouse side will have the where clause using a field from the dimension table. Regardless of the query, there is still a large gap of time between the queries, shown in Figure 24.

	View	DM	Difference
SELECT * WHERE User returns 100 Records	00:00.6	00:00.2	00:00.4
SELECT * WHERE Account returns 1000 Records	00:05.0	00:00.5	00:04.4
SELECT * WHERE Account Type returns 10000 Rec.	00:29.1	00:14.9	00:14.2
SELECT * WHERE Flag returns 100000 Records	04:02.0	01:06.4	02:55.7

Table 3: Testing Set 3 Results (in Minutes)

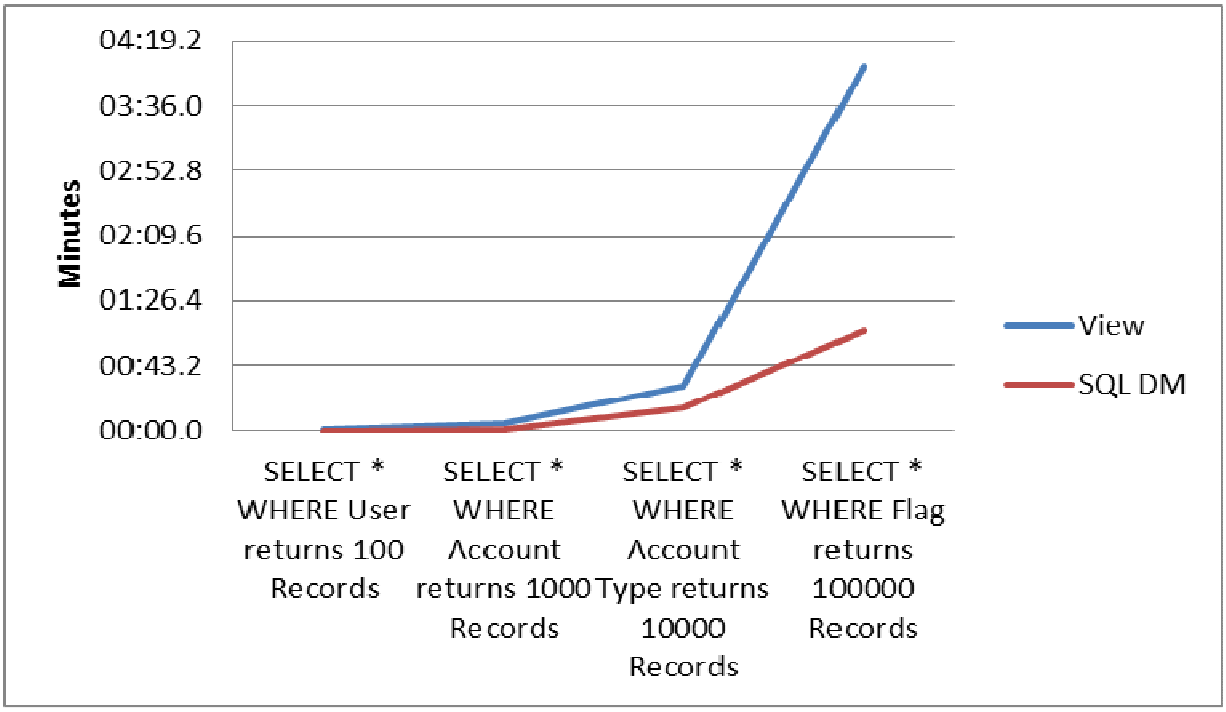


Figure 24: Graphical Results of Select All Where Clause Queries.

CHAPTER 7: CONCLUSION

This thesis introduced the idea of the implementation of an Enterprise Data Warehouse within the context of a higher education environment, in order to better integrate systems for simpler and improved reporting. In order to implement a data warehouse, it is necessary to have a strong business commitment spearheaded by an executive sponsor who believes in the project and is responsible for keeping the project team motivated. The IT department becomes involved to execute the project to install the data warehouse into the enterprise and integrate it with the existing systems.

The Enterprise Data Warehouse implementation is most effective using the up-to-date techniques of data modeling and studied practices as perfected through the years, which is the basis for data warehousing worldwide. By organizing data into a collection of star schemas in the form of dimension and fact tables for efficient organization by the need of the business, user reporting improves. When it comes to reporting, there are many improved tools available instead of the simple ad-hoc reporting or query tools, as enterprise reporting and dash boarding becomes a possibility.

The thesis concludes by studying some actual studies, involving construction of an Enterprise Data Warehouse example as well as studying an actual case of improving queries. By designing and implementing small scale, downsized Enterprise Data Warehouse, it was simple to see how star schemas assemble from an organized and efficient ETL tool from a transactional database out of multiple sources. With the practical exercise, we looked into running the same query against the same data but in very different systems. 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.

Such practical examples as well as the study of data warehouse architecture and reporting capabilities, show the advantages of implementing an Enterprise Data Warehouse for an improved experience for users who are querying and reporting on data for better-informed decisions.

BIBLIOGRAPHY

Adamson, Christopher, *Star Schema: The Complete Reference*. New York: McGraw Hill, 2010.

Bauwens , Christian. Et al. "Oracle Database Sample Schemas 10g Release 2 (10.2)." *Oracle*. June 2005. Oracle Corporation. 27 May 2011. http://download.oracle.com/docs/cd/B19306_01/server.102/b14198.pdf

Bitterer, Andreas. "Hype Cycle for Business Intelligence, 2010." *Gartner Research*. 16 August 2010. Gartner, Inc. 10 April 2011.

"Creating the School Sample Database." http://msdn.microsoft.com/en-us/library/bb399731.aspx. Microsoft Corporation, n.d. Web. 26 May 2011.

"Enterprise Data Warehouse Initiative EDW Planning Project." *Phase I Final Report* 23 Feb. 07. University of California – Berkeley. Web. 7 Mar 2011.

Kimball, Ralph. Et al. *The Data Warehouse Toolkit*. Second Edition. New York: Wiley Computer Publishing, 2002.

---. *The Data Warehouse Lifecycle Toolkit*. Second Edition. New York: Wiley Publishing, Inc., 2008.

Li, Jian; Xu, Bihua; "ETL tool research and implementation based on drilling data warehouse," *Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on* , vol.6, no., pp.2567-2569, 10-12 Aug. 2010 doi: 10.1109/FSKD.2010.5569836 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5569836&isnumber=5569072

McKnight , William. "Data Quality for the Next Decade." *Information Week*. Nov 2010. Web. 20 Oct 2011. http://www.information-management.com/issues/20_7/data-quality-for-the-next-decade-10019112-1.html.

"Reporting Tool Selection in Data Warehouses." *1keydata.com*. n.d. Web. 20 Oct 2011. http://www.1keydata.com/datawarehousing/toolreporting.html.

Rogers, Denise. "Data Warehousing Architecture - Designing the Data Staging Area." *Database Journal*. 22 June 2010. Web. 20 Oct 2011. http://www.databasejournal.com/sqletc/article.php/3888696/Data-Warehousing-Architecture---Designing-the-Data-Staging-Area.htm.

Sallam, Rita. Et al. "Magic Quadrant for Business Intelligence Platforms." *Gartner Research*. 27 January 2011. Gartner, Inc. 25 April 2011.

G. Sanders, S. Shin, "Denormalization Effects on Performance of RDBMS," *hicss*, vol. 3, pp.3013, 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3, 2001

Schaffhauser, Diane. "Florida State U Transforms Reporting with Business Intelligence." *Campus Technology*. April ,2010.

Scott, Peter. "End to End Data Quality." *Rittman Mead*. Web. 20 Oct 2011. http://www.rittmanmead.com/2008/10/end-to-end-data-quality/.

Simon, Alan. *90 Days to the Data Mart*. New York: Wiley Computer Publishing, 1998.

"The Data Warehouse Staging Area." *data-warehouses.net*, n.d. Web. 20 Oct 2011. http://data-warehouses.net/architecture/staging.html.

Usman, M.; Asghar, S.; Fong, S.; , "Data mining and automatic OLAP schema generation," *Digital Information Management (ICDIM), 2010 Fifth International Conference on* , vol., no., pp.35-43, 5-8 July 2010 doi: 10.1109/ICDIM.2010.5664622 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5664622&isnumber=5662242

Veerman, Erik. Et al. *MCTS Self-Paced Training Kit (Exam 70-445): Microsoft SQL Server 2005*. Redmond: Microsoft Press, 2008.

"What is A Datamart." *learn.geekinterview.com*, n.d. Web. 20 Oct 2011. http://www.learn.geekinterview.com/data-warehouse/data-marts/what-is-data-mart.html.

Wai, T.T.; Aung, S.S.; , "Metadata Based Student Data Extraction from Universities Data Warehouse," *2009 International Conference on Signal Processing Systems* , vol., no., pp.670-673, 15-17 May 2009 doi: 10.1109/ICSPS.2009.101 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5166871&isnumber=5166728

Whitehorn, Mark. Et al. "Best Practices for Data Warehousing with SQL Server 2008 R2." *Microsoft Technet*. December 2010. Microsoft Corporation. 24 Jun. 2011. http://technet.microsoft.com/en-us/library/gg567302.aspx

Appendix A

Technical Term Glossary

Attribute – A column (field) in a dimension table (Kimball, Toolkit, 391).

Business intelligence (BI) - A generic term to describe leveraging the organization's internal and external information assets for making better business decisions (Kimball, Toolkit, 393).

Cube - Name for a dimensional structure on a multidimensional or online analytical processing (OLAP) database platform, originally referring to the simple three-dimension case of product, market, and time (Kimball, Toolkit, 395).

Data Mart – A logical and physical subset of the data warehouse's presentation area (Kimball, Toolkit, 396).

Data Mining – A class of undirected queries, often against the most atomic data, that seek to find unexpected patterns in the data (Kimball, Toolkit, 397).

Data Warehouse - The conglomeration of an organization's data warehouse staging and presentation areas, where operational data is specifically structured for query and analysis performance and ease-of-use (Kimball, Toolkit, 397).

Database – A computer application whose sole purpose is to store, retrieve, and modify data in a highly structured way (Kimball, Toolkit, 398).

Deormalized - Allowing redundancy in a table so that the table can remain flat, rather than snowflaked or normalized, in order to optimize performance and ease-of-use (Kimball, Toolkit, 398).

Dimension Table – A table in a dimensional model with a single-part primary key and descriptive attribute columns (Kimball, Toolkit, 399).

Enterprise data warehouse (EDW) - The conglomeration of an organization's data warehouse staging and presentation areas (Kimball, Toolkit, 400).

ETL - Set of processes by which the operational source data is prepared for the data warehouse (Kimball, Toolkit, 401).

Extensible Markup Language (XML) – A cousin of HTML that provides structured data exchange between parties. XML contains data and metadata but no formatting information (Kimball, Toolkit, 402).

Fact Table - In a star schema (dimensional model), the central table with numeric performance measurements characterized by a composite key, each of whose elements is a foreign key drawn from a dimension table (Kimball, Toolkit, 402).

Foreign Key – A column in a relational database table whose values are drawn from the values of a primary key in another table. In a star-join schema, the components of a composite fact table key are foreign keys with respect to each of the dimension tables (Kimball, Toolkit, 402).

Index – A data structure associated with a table that is logically ordered by the values of a key and used to improve database performance and query access speed (Kimball, Toolkit, 404).

OLAP – OLAP is a loosely defined set of principles that provide a dimensional framework for decision support (Kimball, Toolkit, 408).

Operational Data Store (or "ODS") – A physical set of tables sitting between the operational systems and the data warehouse or a specially administered hot partition of the data warehouse itself (Kimball, Toolkit, 408).

Primary Key – A column in a database table that is uniquely different for each row in the table (Kimball, Toolkit, 410).

RDBMS - Database management system based on the relational model that supports the full range of standard SQL. Uses a series of joined tables with rows and columns to organize and store data (Kimball, Toolkit, 411). Oracle from the Oracle Corporation is a RDBMS as well as SQL Server from the Microsoft Corporation.

Slowly changing dimensions (SCD) -The tendency of dimension rows to change gradually or occasionally over time. A type 1 SCD is a dimension whose attributes are overwritten when the value of an attribute changes. A type 2 SCD is a dimension where a new row is created when the value of an attribute changes. A type 3 SCD is a dimension where an alternate old column is created when an attribute changes (Kimball, Toolkit, 413).

Snowflake Schema – A normalized dimension where a flat, single-table dimension is decomposed into a tree structure with potentially many nesting levels. In dimensional modeling, the fact tables in both a snowflake and star schema would be identical, but the dimensions in a snowflake are presented in third normal form, usually under the guise of space savings and maintainability (Kimball, Toolkit, 413).

SQL - Structured Query Language, the standard language for accessing relational databases (Kimball, Toolkit, 414).

Star Schema - The generic representation of a dimensional model in a relational database in which a fact table with a composite key is joined to a number of dimension tables, each with a single primary key (Kimball, Toolkit, 414).

Surrogate Key - Integer keys that are sequentially assigned as needed in the staging area to populate a dimension table and join to the fact table. In the dimension table, the surrogate key is the primary key. In the fact table, the surrogate key is a foreign key to a specific dimension and may be part of the fact table's primary key, although this is not required (Kimball, Toolkit, 414).

Table - Collection of rows (records) that have associated columns (fields) (Kimball, Toolkit, 414).

View – SQL statement that creates logical copies of a table or a complete query that can be used separately in a SELECT statement (Kimball, Toolkit, 417).