Swimming in SQL

Turning Aspiration into Success

Brian Ciampa

Contents

I'm so glad that you've downloaded a copy! May I ask a quick favor before you start reading? Would you mind sending me a tweet to let me know that you're reading this ebook? I would love to know that you are interested in data warehousing and help where I can!

Click To Tweet

Introduction

Karen rubbed her eyes as she finished reading the email. It provided an interesting but exhausting challenge. The SELECT statement was already quite long. The request outlined in the email would add to it. Karen found a motivating song, inserted her ear buds, and began profiling the data that she needed to provide. Argon Enterprises offered her a great opportunity as an analyst, drawing upon her strengths associated with using data and attention to detail. The initial projects associated with regulatory reporting were very interesting, allowing her to groom her SQL skills and gain exposure as a reliable provider of data. She had proven that she was organized, professional, and very logical when it came to problem solving.

However, the requests had become somewhat unwieldy. SELECT statements were beginning to take an entire lunch hour to run and she was often pointed out by the DBAs as someone whose queries hogged system resources. Frequent ad-hoc report requests often gave way to requests for more detailed data, requiring more joins, more complexity, and even some further manipulation in Excel. Also, a revenue report provided to one department sometimes differed from a report of the same name provided to another department. The directors saw things differently and that was a bit difficult to manage at times.

Karen had recently noticed the creation of the Business Intelligence group a few cubicle rows away from hers. Other than meeting the director a time or two, she had no insight into their projects. She did, however, have interest. An all-hands meeting described that group as being able to place organization around Argon's data at a global level, along with standard data definitions. Revenue would be the same for everyone. Although Karen understood that conceptually, she was unclear exactly how they would do that.

She Googled terms like “Business Intelligence” and “Data Warehousing”, in order to become familiar with the concepts and design. However, despite her ability to become somewhat educated on the mission of business intelligence, she had no practical understanding of a data warehouse. Even if she could get time on the director's calendar to discuss a possible role on that team, she had no resume experience associated with building such a system.

The song was becoming distracting. Karen removed the ear buds and continued adding to her SQL, hoping the DBAs wouldn't mind.

****

The scenario described above in one version or another is very common. Corporate America is full of individuals who develop a unique ability to extract, massage, and then deliver data in the context of a specific organization. However, these people may become overwhelmed as the number and complexity of requests grow. The organization may be at risk since the logic associated with this type of data gathering is often baked into Excel formulas and/or the brain of those doing the manual extraction. The reality is that a data warehouse is needed and these people have the talent and motivation to take their careers in the direction of data warehousing. However, they have no experience. This reality can lead to two injustices. On the one hand (perhaps the most obvious), individuals are missing out on the opportunity to pursue a career path that matches their gifts and interests. On the other hand, the Data Warehouse and Business Intelligence industries are missing out on the valuable contribution of some very talented individuals. If you find yourself in this position, this book is intended to expose you to some resources that may help.

Are you new to Data Warehousing, yet intrigued by it? The chapters that follow will offer a brief introduction to the concept, followed by some resources that will help you to break the barrier that Karen, in the scenario above, has encountered. If you're interested, a great career path awaits!

Valuable Data

Taken from the Valuable Data Blog – http://valuabledata.blogspot.com

What is data warehousing? Why is it important? If I'm an executive of an organization and we just spent a few million dollars implementing a large enterprise system for our finance or human resources data, why are you asking me for MORE money to implement something called a data warehouse? Why do I need two systems? What's it going to do for me? This is a valid question and below I'll try to explain why a decision maker can benefit from such a system.

An operational system involves a database that is built in something called third-normal form. In a nutshell, this means that the system is optimized for getting data into it but not ideal for analyzing that data (at a high level). A data warehouse is a database that is denormalized (the opposite of third-normal form). This type of system is optimized for analyzing the data but is not optimized for putting data into it. Consider this example...

When you were young you may have had a piggy bank into which you placed coins. Putting coins into that bank is not hard. You simply drop them into the slit in the top. However, those banks are not ideal for understanding how much money they contain. First, they are designed to encourage saving, so getting the money out is not the easiest thing for a kid to do. Second, even if the bank is opened and you are able to peer inside, you will just see a pile of coins. It's anybody's guess as to how much money it represents.

Now, consider if you changed the environment of those coins a bit. You remove the coins from the bank and put them into money rolls. That's a more tedious process than simply dropping coins through a slit at the top of the piggy bank. However, once that is done, it will be much easier to see how much money the bank contained. Looking at coins that are rolled up (structured) in this way makes them optimized for counting (analysis).

In much the same way, when you go to a retail store and purchase a shirt you don't want to have to stand at the register for a long time waiting for the computer to process the transaction. As a result, those systems are built in third-normal form so that the data can be placed into the system quickly and you can be on your way. The transaction may look something like this...

Date	Item Number	Item Description	Department	Amount Paid
11/14/2011	876FTR	Men's Polo	Men's Dept	$12.99

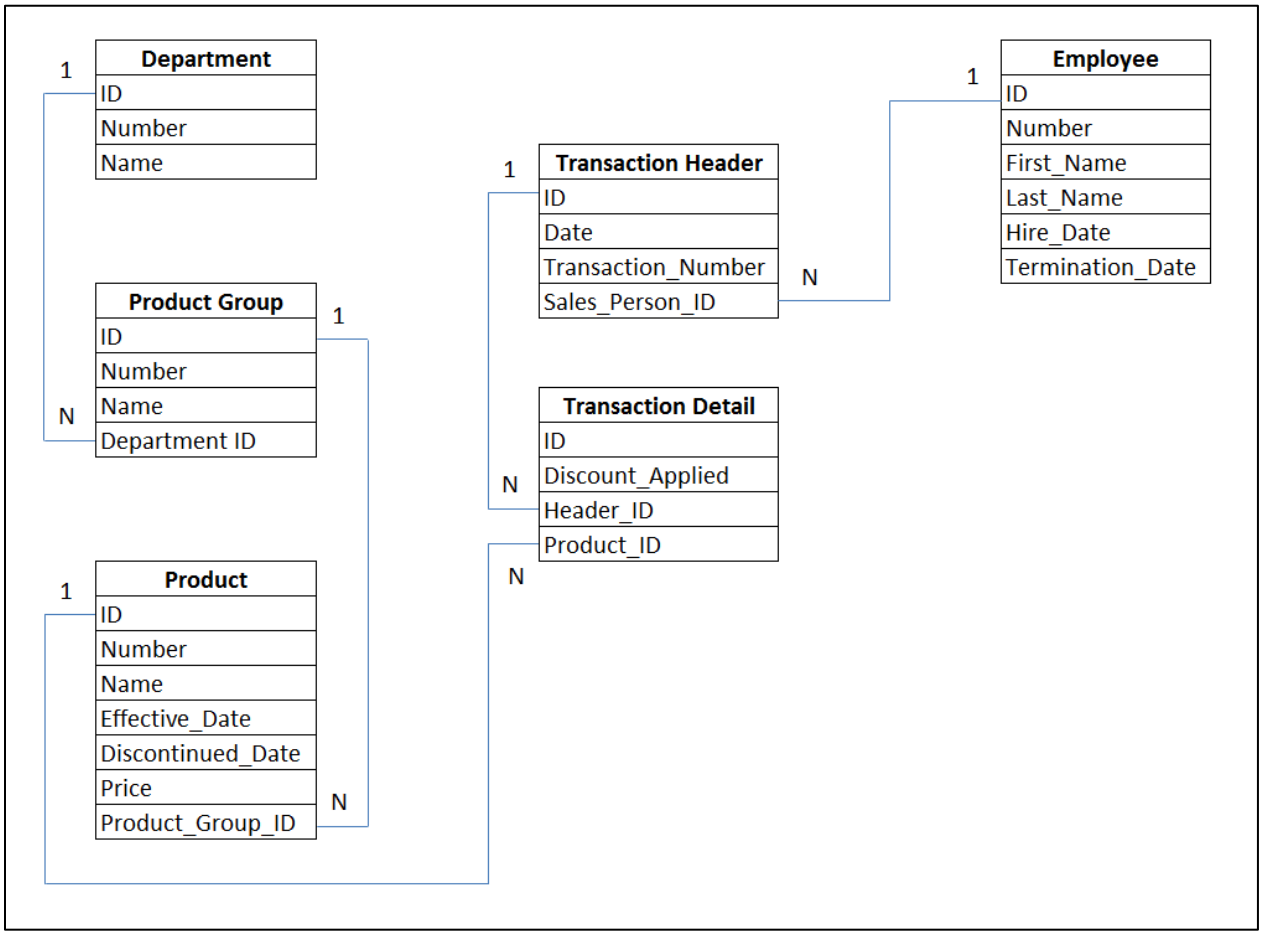
Now, what if 100,000 items were sold by several stores in one day? There would be 100,000 records like this in the system (with different data of course). Multiply that by 365 (days in a year) and you see the amount of data that we're talking about. Now, suppose an executive is wondering when Men's Polo shirts are most often sold. Summer? Spring? October? July? Examining all of those thousands of records is not practical; much like trying to determine how much money is contained within the pile of coins in the piggy bank. However, if a program ran each night (usually called an ETL job) that copied this data into a data warehouse and structured it in such a way that you can easily see into which year, quarter, season, and month this transaction fell, it would be much easier to answer those questions for upper management. When data is made available to decision makers in this way it becomes truly valuable. Data that is just sitting in an operational system is just data. Data that is structured in such a way that it not only tells a story but you can easily see what is that story is a pathway to making better decisions.

The piggy bank example might not be a perfect comparison. In a data warehousing environment the data exists in third-normal form in the source system and is then copied from the source (not removed from the source) and placed into the data warehouse. You can't "copy" the coins so that they remain in the piggy bank and exist as rolled coins too. However, I'll trust that the example provides some benefit.

Technical Value Add

Taken from the Valuable Data blog – http://valuabledata.blogspot.com

In the last post we looked a bit at the theory of data warehousing and the value that it adds. Now I would like to examine the value add from a more technical perspective. Suppose you have the following tables in an operational system.



Now, suppose that an executive asks the following question: Which employees sold products in the summer or fall from the "Apparel" product group?

From that conversation you are tasked with providing a report that shows…

1.) Employee Name
2.) Employee Status
3.) Product Name
4.) Product Status
5.) Season
6.) Actual Sales Price

Since this data has not been warehoused, a SQL statement similar to the following will probably be required (assuming Oracle syntax).

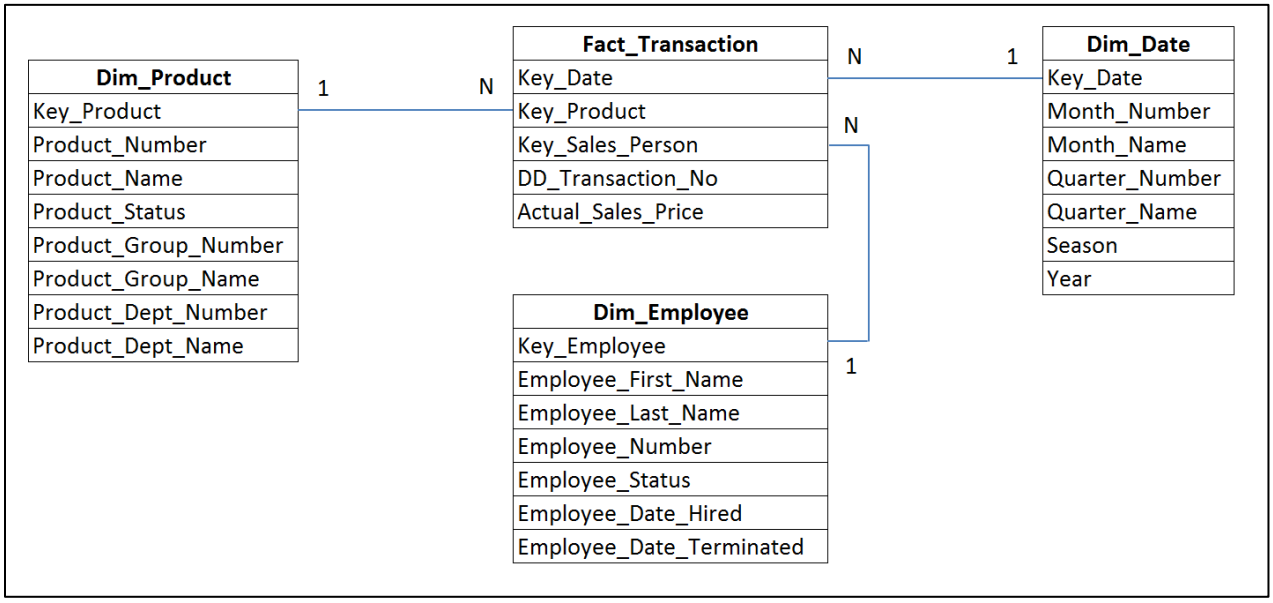
```
SELECT      a.last_name,
            a.first_name,
            decode(a.termination_date, null, 'Active','Inactive') as "Employee Status",
            e.name,
            decode(e.discontinued_date, null, 'Active','Inactive') as "Product Status",
            case    when extract(month from b.date) in (11,12,1,2) then 'Winter'
                    when extract(month from b.date) in (3,4,5) then 'Spring'
                    when extract(month from b.date) in (6,7,8) then 'Summer'
                    when extract(month from b.date) in (9,10) then 'Fall'
            end as "Season",
            e.price – c.discount_applied as "Actual Sales Price"
FROM        employee a,
            transaction_header b,
            transaction_detail c,
            product_group d,
            product e
WHERE       a.id = b.sales_person_id
            AND b.id = c.header_id
            AND e.id = c.product_id
            AND d.id = e.product_group_id
            AND case        when extract(month from b.date) in (11,12,1,2) then 'Winter'
                            when extract(month from b.date) in (3,4,5) then 'Spring'
                            when extract(month from b.date) in (6,7,8) then 'Summer'
                            when extract(month from b.date) in (9,10) then 'Fall'
            end in ('Summer','Fall')
            AND d.name = 'Apparel'
```

While this is not the longest SQL statement ever written, it is pretty verbose. The employee status, product status, actual sales price, and season fields need to be derived or calculated. That derivation or calculation must happen within the SQL statement for each row that is returned, which will eat up resources. If the business rules for the definition of a season change, then they will need to be changed in both the select clause and the where clause.

Suppose that beginning in 2011, management decided that November would be considered a "Fall" month but wanted to keep history. In other words, November transactions that occurred in 2010 or earlier would fall into the "Winter" bucket but those in 2011 and later will fall into the "Fall" bucket. Imagine the SQL statement that would need to be written for that!

In reality, this example may be a bit simplistic. In fact, if this requirement came through and it was the only requirement of its kind, it may be worth it to just write the SQL. However, once decision makers start to see data in a slightly different way they tend to want more and more and more…so the SQL statements get more complex. Once the analytical requirements reach a certain point, the data needs to be restructured.

Now, consider if the data from the tables is denormalized and placed into a data warehouse (see below).



In this case, the SQL statement to produce the same report will be something like this…

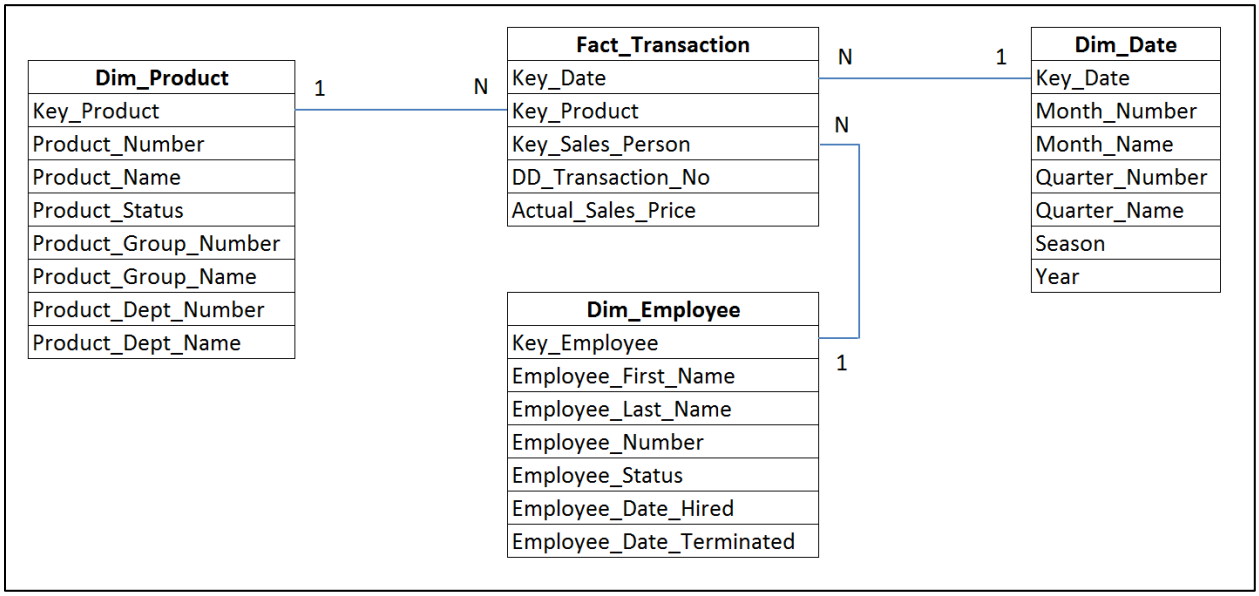
```
SELECT      b.employee_last_name,
            b.employee_first_name,
            b.employee_status,
            c.product_name,
            c.product_status,
            d.season,
            sum(a.actual_sales_price)
FROM        fact_transaction a,
            dim_employee b,
            dim_product c,
           dim_date d
WHERE       a.key_date = d.key_date
           AND a.key_product = c.key_product
           AND a.key_sales_person = b.key_employee
           AND d.season in ('Summer','Fall')
           AND c.product_group_name = 'Apparel'
GROUP BY    b.employee_last_name,
            b.employee_first_name,
            b.employee_status,
            c.product_name,
            c.product_status,
            d.season
```

That is much simpler. Also, the SQL statement does not contain any of the business rules. Each night, an ETL job (ETL is an acronym for extract, transform, and load) essentially assigns dates to the appropriate quarter, year, season, etc., assigns products to the appropriate product group, derives statuses, and performs other tasks that would be tedious within a select statement. If the business rules that define a season, for example, change then that can be taken care of in the ETL job. The SQL statement can remain the same. This makes pulling data for the purposes of analysis much simpler. If you are used to thinking about data in third-normal form then this will require a slightly different perspective. However, the value add can be great in helping executives analyze their data quickly.

Star Schema

Taken from the Valuable Data blog – http://valuabledata.blogspot.com

In a prior post we looked at the value-add of a data warehouse using the following example. These relationships may seem a bit unusual to somebody who is not familiar with data warehousing, so I wanted to touch on this just a bit. When tables relate to each other in this way, they are a part of a star schema. Each star schema has two types of tables – Fact tables and Dimension tables.



Fact Tables: A Fact table contains the data that is to be measured. In the example above (a simple example) the fact_transaction table is the fact table. The actual_sales_price is the item that is measured. In this case, only one measureable item is included. If we wanted to add another measureable item (i.e., sales_price_before_discount) we could do that. This table can contain any number of additive items that fit the grain. What is grain? A fact table's grain is basically the answer to this question: What does each row in the fact table represent? In this case, the grain is one row for each transaction line item. So, if a customer purchases three items, then that will result in three rows (among all of the others that pertain to other transactions) in this fact table. Generally speaking (of course, there are a few exceptions) the fact table itself does not contain a description of this measurable data. Another kind of table fulfills this role.

Dimension Tables: A Dimension table describes the data that is contained in the fact tables. The idea is that you aggregate the measureable objects (i.e., sum the actual sales price) and then group by the dimension objects. You can analyze the actual sales prices for each product, for each year, for each month, and so on. Notice that the dim_product table, for example, contains the lower-level product data and the higher-level product group data. A developer who is creating tables in third-normal form would probably place this lower-level data and higher-level data into two different tables. In the data warehousing world, this may not happen. Looking at the raw data in this table may show something like…

Product Department Name	Product Group Name	Product Name
Men's Dept	Shoes	Brand A Sneakers
Men's Dept	Shoes	Brand B Dress Shoes
Women's Dept	Accessories	Wrist Watch
Women's Dept	Accessories	Beaded Necklace
Children's Dept	Athletic Wear	Sweat Pants

Notice that some of the data repeats. This allows you to easily see the actual sales price at a high level (i.e., price per department) and then as you add additional descriptive data to the select statement and group by it you can drill down to a more granular level (i.e., price per department per product group). In some cases, a descriptive element belongs by itself. Notice the dd_transaction_no field in the fact_transaction table. This describes the fact in terms of the transaction. In other words, which transaction did this refer to? This doesn't belong in any of the existing dimension tables. Rather than just create a dimension table with a key and this transaction number, we can place it directly into the fact table. It is known as a degenerate dimension in this case.

Fact tables and dimension tables are the two main types of tables used in data warehousing. Others exist which we may examine in the future. For now, these are the fundamentals of this type of system.

Extract, Transform, and Load (ETL)

Taken from the Valuable Data blog – http://valuabledata.blogspot.com

In prior posts, we've looked a bit at the basic structure of a star schema but we have not looked at populating a star schema. Populating a data warehouse is accomplished using an extract, transform, and load (ETL) job. This is a job that pulls data from the source system, transforms it into a structure appropriate for the data warehouse, and then loads that data into the data warehouse. The concept of ETL is not unlike any other craft or trade. Consider the following…

1.) A math student reads and understands the math problem from the textbook (extract), uses a piece of scrap paper, if calculators are not allowed, to find the solution (transform), and places the answer onto the final answer sheet (load).
2.) A carpenter purchases wood from a supplier (extract), uses his workshop to design, measure, sand, cut, etc. (transform), and then delivers the final product to the customer (load).

Usually, the transformation piece is accomplished inside of a separate schema in the database, called a staging area. The data from the source system is placed into this area and then restructured so that it is appropriate for the star schema, much like the math student uses the scrap paper and the carpenter uses the workshop. It is generally accepted that this area is for the ETL developer's eyes only. End users do not have access to data in the staging area, much like a customer is not involved in the carpenter's workshop and the math student does not place his chicken scratch on the sheet with the final answer. Once that data has been transformed, it is placed into the star schema and is ready for analysis.

It may be worth noting that some variations of ETL, such as ELT (extract, load, and transform), ETLT (extract, transform, load, and transform), and others, are used to describe methods by which jobs will populate a data warehouse using a different order than the standard extract then transform then load order. In order to encompass all of these terms, the term data integration (or something similar) is often used to describe data warehouse population in a general sense.

Conclusion

Karen glanced at her newly created website once more before locking her computer and walking to Bill's office. She eagerly anticipated the discussion as well as what may come as a result.

The last few weeks had been both exhausting and enjoyable at the same time. The book, *The Data Warehouse Workshop*, had exposed her to some resources that allowed her to practice the techniques needed to build a data warehouse. Some sample data was provided, free tools, and a description of business processes. Upon the completion of those exercises, she had been provided with a suggested framework, thanks to *The Data Warehouse Portfolio,* for packaging her work in the form of a web-based portfolio. Bill, the director of Business Intelligence at Argon, had agreed to give her 15 minutes out of his schedule to discuss the portfolio.

She stopped in the doorway of his office and knocked lightly on the open door. He looked up from his computer.

"Karen!" he said as he motioned her in.

"I really appreciate your time," she said as she sat down opposite his desk.

"Not a problem," he said. "I looked over the link you sent me. Very nice work…"

"I enjoyed going through the process," she added. As Bill asked why she took certain approaches to populating some of the tables in her ETL, Karen enjoyed explaining those approaches. Working through the ETL from the ground up had given her great familiarity with her portfolio as well as confidence that she could work well in this arena. Bill continued to look impressed.

"So, you've never done any data warehousing work before?" he asked.

"Other than that, no," she said. "But I work with Argon's data every day, writing lots of SQL and creating ad-hoc reports. I'm intrigued by data warehousing and just wanted to share that with you."

"Well, I'm intrigued by your intrigue," he added as he stared at her portfolio on his desktop monitor. He then received a calendar reminder. "I have a meeting now but I'll tell you what…let me get with your supervisor a little later. I'm impressed by your work here. I think you have a lot to offer so let me see if we can get you more and more involved in Argon's business intelligence initiatives. I'll email

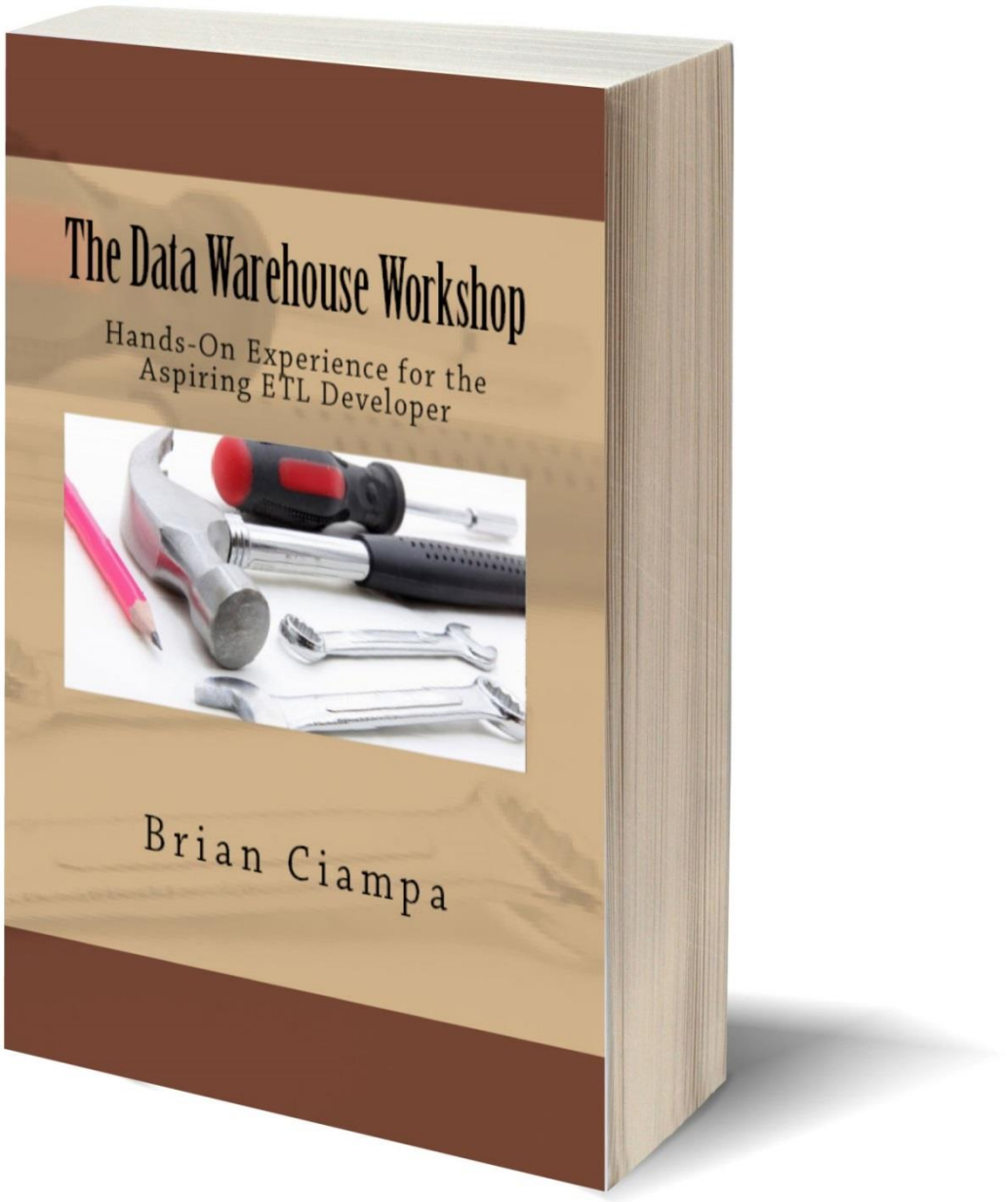
you a little later to give you the specifics of a small project to work on. We'll see where it goes from there."

He stood up and locked his computer. "Thanks for taking the initiative to reach out. I think this will work out really well for both of us."

The Data Warehouse Workshop

Hands-On Experience for the Aspiring ETL Developer

Available on Amazon
Click for more information



The Data Warehouse Portfolio

Don't Just Ask For a Job…Display Your Value

Available on Amazon

Click for more information