Paper 83-2010

Introduction to Business Intelligence for SAS Users

Tom Miron, Systems Seminar Consultants, Madison, WI

Abstract

Here, Business Intelligence, or BI, refers to the gathering of data from external sources and the compilation and presentation of that data to end users. If you're creating reports, graphics, predictive models, or other data presentations that drive business decisions then you are involved with business intelligence. This paper introduces key BI concepts and techniques and shows why and how they are useful for any SAS reporting or presentation project, from ad hoc report to full-blown data warehouse.

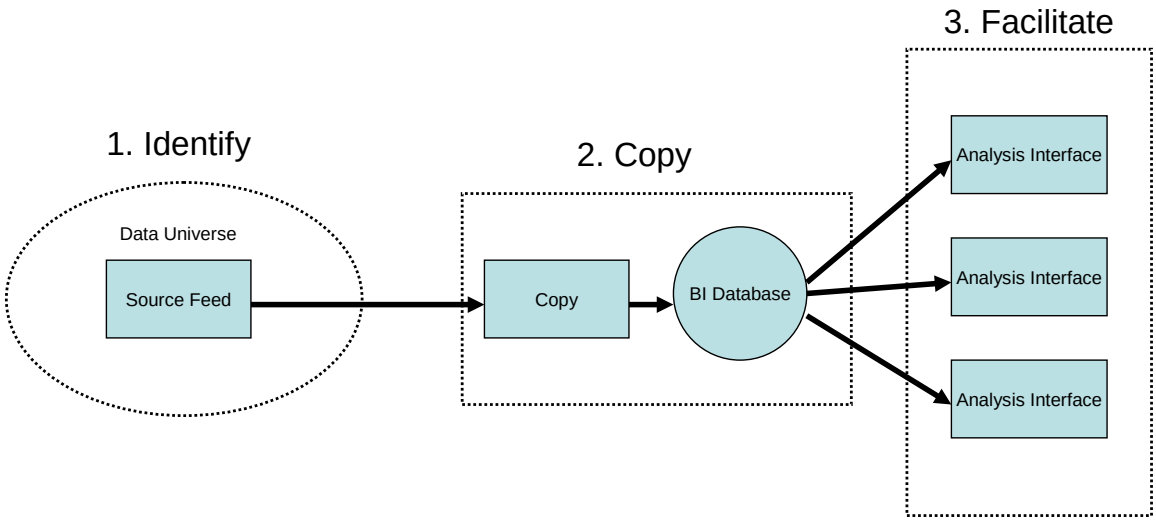
What is Business Intelligence?

Business Intelligence is a slippery term. For our purpose Business Intelligence, or BI, does not refer to any software product. It refers to the compilation and presentation of data to end users. In this sense, BI is often associated with data warehouse, data mart, decision support system (DSS), executive information system (EIS), and other acronym-worthy monikers. Here we will treat all these as BI, as one notional grouping, while keeping in mind that there are many, often conflicting and fiercely defended definitions of these terms that may draw a sharp distinction among them.

Using our definition, the fundamental character of business intelligence involves three elements:

1. identify and understand source data
2. create a processed copy of source data in a user accessible BI database
3. create or facilitate a user interface that allows analysis and presentation of the BI data

We will concentrate on elements 1. and 2.



Three Elements of BI

Important points

- If you're creating reports, graphics, predictive models, or other data presentations that drive business decisions then you are involved with business intelligence.
- BI principles covered in this paper are useful for any data presentation system even small, informal, or ad hoc systems.

What We'll Cover

Business intelligence, by any definition, cannot be comprehensively covered within the confines of this paper. Our purpose here is to cover a couple of the main tenets of BI, explain how they can be generally applied, and relate them to SAS software.

The rest of this presentation is divided as follows:

- Understanding Operational vs. BI Data
- ETL or something like it
- Dimensional Modeling

Understanding Operational vs. BI Data

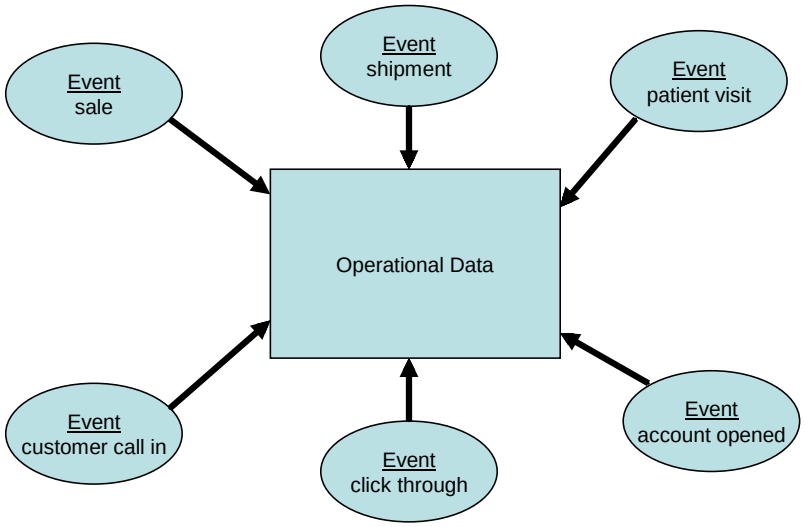
The world of BI and its relatives is littered with fuzzy, inscrutable terminology. I'll try to clarity a few terms here. First is "operational data."

Operational data

Operational data (op-data) originates with business events. In fact, an operational data item is a record of a business event. Some examples:

- a customer order
- a patient visit
- a banner ad clicked on
- an account opened

These are all critical business events that need to be recorded as accurately and as quickly as possible and with as many business-relevant attributes as possible.



Operational Data Represent Business Events

Each op-data item represents a real, unsummarized, unedited event. A key attribute of most op-data is time: When did this event occur?

As-it-happens

Op-data is typically recorded as it happens. Examples:

- A log record is cut when the page under a banner ad is served.
- An order entry record is added when a customer makes an order.
- A ship log record is entered when an item goes out the door.

A well-designed system will introduce a minimum amount of time between the real-world event and the insertion of its op-data record. This as-it-happens nature relates closely to another feature of op-data: system of record, discussed below.

Typically, you have no control over the structure and content of operational data and often no control over the timing of its availability. Op-data is controlled by, for the benefit of, those creating it. You are simply a user. The concerns of the op-data system are speed of entry (efficient table insert), integrity (don't want to lose an order), and availability (don't want a customer on hold to 'til the system's back up). These are not the concerns of BI.

System of record
Operational systems are often systems of record. This means, for example, a customer order does not really exist until it is recorded in the order entry system. The order entry system is the system of record for orders.

In an order entry system, for example, there may be a window between the time a customer relates a phone order to a sales rep and the point the sales rep keys the order into the order entry system. If the system goes down in that window the order will only be a fond memory for the sales rep. It will not exist as an order. Obviously, reducing the event-to-entry risk window is a prime design consideration for operational systems.

Virtual op-data
More generally, you should think of operational data as relative to your project's data flow. Example: The sales department creates a daily extract of all sales activity. This is the input to your market analysis BI system. From the view point of your system, the extract is op-data and the system of record. If a sale is not on the extract it doesn't exist.

If you do not control the structure, content, or timing of your input you can consider it op-data for the purpose of our BI discussion here. The extract described above could be called "virtual op-data."

Op-data isolation
Because operational systems are important for time-sensitive business processes they need to be shielded from access by unruly end users. And, because they are systems of record, they cannot be updated or manipulated by users. End-users cannot alter the structure of the op-data database or assign their own meaning to data elements. Operational systems are, or should be, isolated from end user access. End users need their own version of business data: the BI system.

A First Principle
Op-data is the starting point of your system. One of the first steps in any BI project is to know your op-data. This means: clearly define the delivery method, understand the timing, semantics, and variances. These characters of op-data need to be part of your project documentation. The op-data feed, by definition, is out of your control so it must be taken as given and become a parameter for your system design.

BI principle number one:

1. *Clearly define and understand the content, timing, and quality of your (virtual) op-data.*

BI = not op-data
We come all this way to describe what BI is not. It is not operational (or virtual operational). BI systems create and present a processed copy of op-data. Working with this BI copy gives you a controlled environment in which to develop, test, validate, document, impose compliance controls, summarize, graphically present, and pretty much do what you please with precious business data without affecting critical operational systems. This control allows you to reproduce results and directly control timeliness. It allows you to launch complex, long running queries. It allows you to detect and manage structural and data semantic changes in the op-data that may be introduced without warning.

The key word here is "control." Your BI system, humble as it may be, is in a controlled environment, isolated from the vagaries and sensitivities of op-data.

BI is after the fact
It should be clear that BI systems do not show users what's going on right now. They represent an after-the-fact snapshot of some aspect of the enterprise. That said, some BI systems may be very close to real-time. A system showing page hits in the last hour or even last minute, for example. Even such a system is most probably not directly displaying actual operational data. It will show a frequently taken extract.

Time is critical
Since BI is after the fact, there must be some discreet time cut off point represented in BI data. For example, a weekly sales report shows sales through last Monday. Midnight on Monday is the cut off point. The cut off point and the unit of time it represents (day, week, year, etc.) are critical features of even the simplest system. An ill-considered choice here will cause problems for everything that happens downstream.

Example:
A weekly sales summary is provided by your sales department. This is your virtual op-data. The weekly sales summary data is for the seven day period Sunday through Saturday. Your marketing department wants a Monday through Sunday summary. Your BI system can take two weeks of input data and apply some proration algorithm to approximate Monday through Sunday sales. Two problems: 1. Marketing's Monday-Sunday report is delayed by six days; 2. About one seventh of the sales are an approximation.

The example above relates to granularity and alignment. "Granularity" can be thought of as level of detail. If the input data were at a finer grain than required by the BI presentation, say daily data, you could summarize to any arbitrary courser grain, say weekly Monday through Sunday, with no delay. This example illustrates BI principle number two:

2. Make sure your input data's time attribute is at a finer level of detail (grain) than required for your BI system or ensure that it time aligns with your presentation requirements.

One way data flow
Data flows from the operational side of a business to the BI side in a one-way channel, that is, the results of BI side data manipulations are not used to update operational data. Changing operational data after the fact is a violation of their function as a system of record. This one way valve is more than just an emergent feature of most BI systems, it is a critical internal control. In other words, you don't want marketing to analyze last week's sales data in their BI system then update the source data to reflect better performance, aka cooking the books. So BI principle number three:

3. Your BI system should have explicit or implicit access controls that prevent update of (virtual) op-data by the BI system.

The Role of ETL

ETL is Extract, Transform, and Load. ETL is most often associated with a formal data warehouse, but we can generalize to our more expansive definition of BI: ETL gets the data from the (virtual) operational system to the BI system. Whether formal or not, the functions of ETL or an ETL-like process are the same:

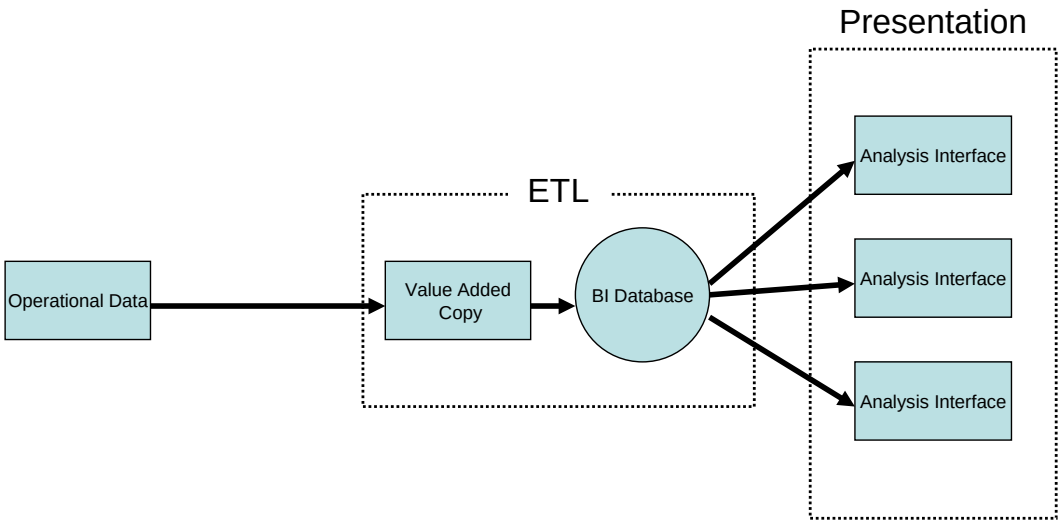
Basic functions:

- Copy operational data to the BI side in a timely and innocuous (to the operational system) manner.
- Create a fixed-in-time snapshot of the op-data.

Optional functions:

- Validate, backup, check-point, summarize, version stamp, create meta-data, apply access rules, and on and on.

All BI systems require at least the basic ETL functions above. Going forward we include informal ETL-like processes, that perform at least the basic functions above, under the term ETL.



ETL Transforms Operational Data to BI Data

In most BI systems, ETL requires the most effort and maintenance. ETL must accommodate real-life situations such as:

- op-data feeds from multiple sources
- data feeds with differing or perhaps random time alignments
- data feeds of dubious quality
- data feeds apt to change format without warning

We can give ETL only a cursory look here. But any BI system will have an ETL (or ETL-like) component and needs to consider ETL functions.

Separate ETL from other processes
The ETL process should be a discreet component within the BI. This means the ETL should not be integral with the presentation code or processes of the BI. This isolates the presentation side from changes made to the ETL and vice-a-versa. The ETL does not interface directly with the presentation side, but creates a standardized file or table structure that the presentation component draws on. We'll discuss one of these table structures, the dimensional model, below. This fixed data structure interface allows various presentation systems to use the data without rewriting the ETL. It also facilitates maintenance, validation, and debugging by making the pre-presentation data directly available for examination. Even a small system can benefit from this structure.

ETL anatomy
The Extract part of ETL involves just getting the data inside the boundary of your BI system. Ideally, this first step will make a copy of the data that is as close as possible to op-data feed while still in a format usable by the BI or ETL. In SAS, this may be a DATA step dedicated to reading the data feed and creating a SAS table of the feed with no transformation, validations, filtering, etc.

Example extract step:

```
data feedDaily;

   infile 'salesBI.feed.y2010m10d11.txt';

   input
      @01  dateField           $char10.
      @17  salesDollarsCents   $char14.
      @37  customerID          $char10. ;

run;
```

Note that this DATA step does not cast the input date and sales fields to SAS numerics. The point here is to get the data into the BI system and under your control. A straightforward copy of the data allows you to validate, debug, and test maintenance on the untransformed data using SAS rather than having to refer to data outside your control, that is, the original feed file.

BI principle number four:

4. Your BI system should have a discreet ETL (or ETL-like) process component that begins with a data copy step.

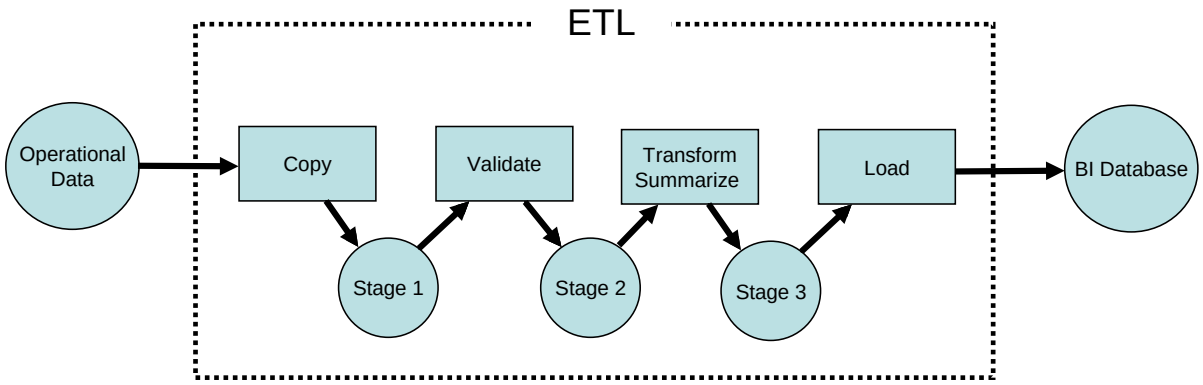
Life-cycle efficiency is the goal
Of course, you could do much more in a single DATA step than simply copy the data, perhaps generate your entire BI output. But, given the current state of disk capacity and processing power and barring truly huge volumes of data, the inefficiency of creating additional working copies of the feed data will, in most cases, not be an issue. The goal is overall life-cycle efficiency that considers clarity and development and maintenance effort.

Once internalized, data may be transformed, validated, summarized, etc. with downstream DATA and PROC steps. You may produce several intermediate tables that represent discreet stages of your ETL. Again, in most cases, the performance cost of creating multiple tables is more than offset by reduced development and maintenance time.

Example ETL with extract, transform, and load steps:

```
/*
Extract
Includes generational backup and feed file tag
*/
data feedDaily(genmax=8);

   keep
      feedFileName dateField salesDollarsCents customerID;

   length
      inFileName         $256
      feedFileName       $256
      dateField          $10
      salesDollarsCents  $14
      customerID         $10;

   infile 'salesBI.feed.y2010m10d11.txt' filename=inFileName;

   feedFileName = inFileName;

   input
      @01  dateField           $char10.
      @17  salesDollarsCents   $char14.
      @37  customerID          $char10. ;

run;

/*
Transform 1
Validation
Assume a customer name look up SAS format is
available to the step.
*/
data stage01Daily;

   keep dateNumeric salesNumeric customerID customerName feedFileName;

   length customerName $60;

   set feedDaily;

   dateNumeric = input( dateField, ? mmddyy10.);

   if dateNumeric = . then do;
      <invalid date handler>
   end;

   salesNumeric = input(salesDollarsCents, ? dollar14.2 );

   if salesNumeric = . then do;
      <invalid sales handler>
   end;

   customerName = put( customerID, ? $customerName. );

   if customerName = ' ' then do;
      <invalid customerID hander>
   end;

   if _error_ then delete;

run;
```

```
/*
Transform 2
Presentation table requires specific character formats.
*/
data stage02Daily;

   keep date customerID customerName sales feedFileName;

   length
      date         $10
      sales        $14 ;

   set stage01Daily;

   date = put(dateNumeric, yymmdd10. );
   sales = put(salesNumeric, 14. );

run;

/*
Load
Add new data to presentation store
*/
proc datasets lib=salesBI nolist;
   append base=salesBI.customerSales data= stage02Daily;
run;
```



ETL Steps with Intermediate Data Stages

Note on the Presentation Component
The output end of a Bi system is presentation. Presentation may be part of your project or may be the responsibility of others. Presentation may be as simple as ad hoc SQL queries or may include an interactive interface, static and active reports, published HTML, dashboards, etc. This is the sexy part of any BI and without presentation the BI has no reason to exist. There's a vast array of information on presentation so we leave the BI presentation layer with just an acknowledgement.

Four BI Principles
The four principles presented above are useful for any system that creates presentations or presentable data from operational data, even if not considered a formal Business Intelligence system.

1. Op-data is a parameter of your project. Clearly define and understand the content, timing, and quality of your (virtual) op-data.

2. Make sure your input data's time attribute is at a finer level of detail (grain) than required for your BI system or ensure that it time aligns with your presentation requirements.

3. Your BI system should have explicit or implicit access controls that prevent update of (virtual) op-data by the BI system.

4. Your BI system should have a discreet ETL (or ETL-like) process component that begins with a simple copy step and ends by loading new, processed, data to a table or database.

Dimensional Modeling: What and Why

Next we'll look at the output side of the ETL, specifically the dimensional data model (DM). The output of the ETL is commonly stored in a series of tables. These tables are queried directly by end users or may be input to a reporting system or interactive user interface. A common data structure for these output tables is the dimensional model. The dimensional model is often used for formal data warehouse/data mart, but is useful for humbler systems as well.

A comprehensive look at DM is not possible here, but we will touch on a few key ideas.

Facts and Dimension
Dimensional modeling is founded on the separation of data into two types: facts and dimensions. A fact is represented by a row in a fact table. A dimension by a row in a dimension table. Facts and dimensions are related by keys so the dimensional model requires a table-based, relational data system. SAS datasets work fine as the foundation of a dimensional model.

Facts
Each row in a fact table represents a business event. Recall that BI systems are all about events. A fact contains a numeric measure of interest associated with the event. "Fact" and "measure" are used synonymously. Together, they refer to numeric items of interest associated with the business event.

Example events and facts:

Event		Fact or Measure
1.	a sale	the sale dollar amount
2.	a banner ad click through	its existence
3.	a shipment	the number of packages
4.	a manufacturing process step	its was completed

For a sale and many other business transaction events the obvious measure is the dollar amount, as with event 1. above. Other events may be counts as with the shipment event, number 3.

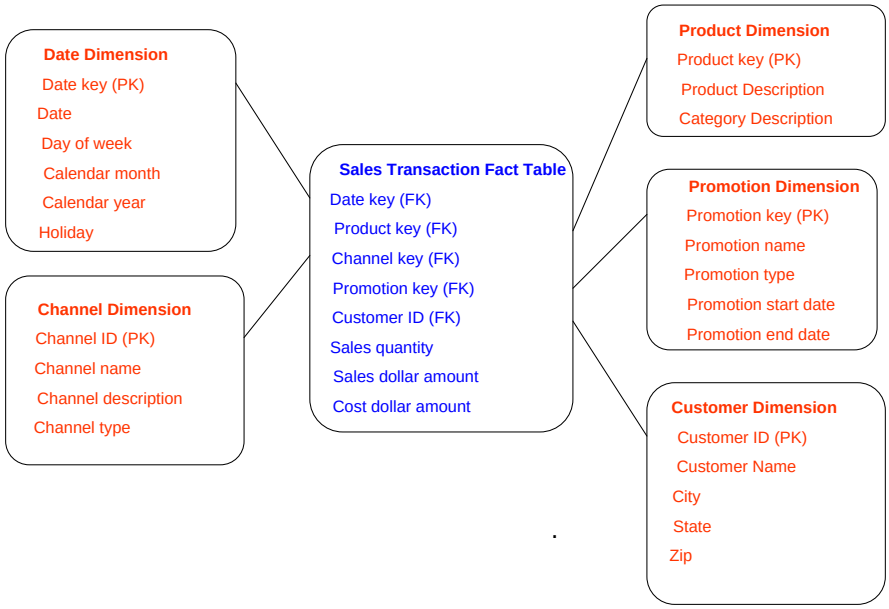
Note events 2. and 4. What is the numeric measure here? Without digressing into a full discussion of this type of event, you can think of this numeric measure as a count always equal to one.

Dimensions
Dimensions are attributes of a fact. They are defined and exist independent of facts. Example dimensions based on the events and facts above:

Event		Possible Dimensions
1.	a sale	date sold, customer who bought it, product sold
2.	a banner ad click through	timestamp of the click, promotion associated with the ad, server that served the page
3.	a shipment	timestamp from carrier, order number fulfilled, carrier, customer shipped to
4.	a manufacturing process step	timestamp of completion, process step ID, product, batch number

Note that time is a common and often critical dimension of a fact. Facts represents events. Events occur at a specific time.

Relating Facts and Dimensions
The typical DM structure is a single fact table with many related dimension tables. Facts and dimensions are related by keys. The keys identify specific rows in a dimension table.



A Dimensional Model for Sales

Note that the diagramed relationship among facts and dimensions resembles a (lopsided) star. Thus the dimensional model is often referred to as a star schema.

The base SAS system provides all the features you need to implement a fully formed dimensional model.

Why DM?
Breaking data into facts and dimensions involves planning, analysis, and a level of complexity greater than a flat, one-table structure. After all, SAS is perfectly happy with all dimensions, aka CLASS variables, residing in the same table with their associated analysis variables. This is what PROCs, such as PROC MEANS, expect input to look like. But here's three of the many reasons the dimensional model is worth considering.

1. *The potential number of fact rows is large compared to the number of unique dimension values.*

   Since the fact table stores only the numeric fact itself and numeric keys to dimension tables, fact tables are disk space efficient. Imagine a 10,000,000 row fact table related to a customer dimension the includes a 60 character customer name and a 120 character customer address, 180 characters per customer. In a flat table that's:

   10,000,000 X 180 = 1.8GB

   just for the customer information.

   Plus, consider what happens if you need to update a customer address.

   In the dimensional model you store only one row per customer and relate that row to the fact table via a key of perhaps 4 bytes:

   10,000,000 X 4 = 40MB
   40MB v. 1.8GB for customer info storage in the fact table

   The DM model can effectively store and make accessible truly huge amounts of data.

2. Your dimensions are shared and/or must conform to a business standard.

Following our example above, a separate customer table can be maintained by a unit of the business that specializes in customer information. Such a table can be shared among all BI systems. This ensures that everyone is using the same critical customer information and that maintenance and change are managed from a single point. In the context of DM this is called "conformed" data.

Further, personally identifiable information (PII) can be stored in a segregated table with its own access rules. The fact table can be freely queried and the PII is still related to the fact rows, but the PII attributes (name, SSN, etc.) cannot be read without proper credentials. The PII table can be managed and audited separately.

3. All queries against a DM take the same form.

Most DM queries have a common form. This reduces the learning curve for analyst's ad hoc queries and simplifies hooking DM data into various presentation systems such as drill reports, and dashboards. In the dimensional model all queries look like the one below which asks for all January sales in the West region. "Sales" is the fact table. "Customer" and "date" are dimension tables.

```
select
    salesAmount, region, month
from
    sales, customer, date
where
    sales.customerKey = customer.customerKey
       and
    sales.dateKey = date.dateKey
       and
    date.month = 'January'
       and
    customer.region = 'West'
```

The query WHERE clause first relates the fact table customer and date keys to their respective rows in the customer and date tables then selects on column values in those dimension tables, i.e. month and region.

Most DM queries follow these two steps:

1. relate dimension keys then…
2. select on dimension values

Of course, other selection criteria and summarization may be present as well.

More DM Good Stuff

There is much more to DM. And there's good reason it's considered the BI gold standard data structure. A few other DM features:

- straightforward summarization, disaggregation, and event joining, i.e., drill up, drill down, and drill across
- facilitates control of dimension attribute changes, e.g., what happens when a customer changes name?
- relative simplicity v. a fully normalized table structure
- facilitates conformed data, i.e., a single version of key business information such as customer and product attributes

The DM Takeaway

Hopefully this discussion gives you a feel for what DM is about. DM can be shrouded in mystic, which is sad because its beauty lies in its simplicity. For straightforward static reports it may not be appropriate. Beyond that, DM should be considered for any BI system.

Wrap Up

Business Intelligence can be defined as the systems that support business analysts and decision makers as opposed to the operational systems that record business events as they happen. BI systems present data to users. Well constructed BI systems have some common features that are worth considering for most reporting and presentation projects. These include clear definition and segregation of input operational data, time alignment between the presentation and source data, and a discreet data capture process (ETL).

BI systems benefit from the use of the dimensional data model. DM is a time-testing technique that enhances performance, flexibility, and clarity. DM simplifies and optimizes queries. DM facilitates system extension, change, and maintenance.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Tom Miron
Systems Seminar Consultants
2997 Yarmouth Greenway Dr.
Madison, WI 53711
608 625-4541
608 278-9964
tmiron@sys-seminar.com